Mgr inż. Mateusz ŻARSKI

Politechnika Śląska w Gliwicach
Wydział Budownictwa

WIZJA KOMPUTEROWA I TRANSFER LEARNING W INSPEKCJACH BETONOWYCH OBIEKTÓW MOSTOWYCH

***PROMOTOR*: PROF. INŻ. MAREK SALAMAK**

POSIADANIE WYOBRAŹNI MOŻE BYĆ DLA NIEKTÓRYCH SZOKUJĄCE

HAVING AN IMAGINATION MAY BE SHOCKING TO SOME PEOPLE

INSPIROBOT AI

Spis treści

Wykaz Oznaczeń....VII
Słownik skrótów....VIII
Słownik terminów występujących w rozprawie....X
1. Wprowadzenie....1
1.1. Uzasadnienie wyboru tematu pracy....2
1.2. Tezy i cele pracy....5
1.3. Omówienie układu pracy....6
1.4. Metody badawcze stosowane w pracy....8
2. Przegląd literatury....9
2.1. Ocena dostępności literatury....10
2.2. Aktualny stan wiedzy i praktyka przeprowadzania inspekcji obiektów mostowych....11
2.2.1. Aktualny stan wiedzy w systemach utrzymania i zarządzania infrastrukturą mostową....11
2.2.2. Inspekcje obiektów mostowych na świecie....13
2.2.2.1. Europa....14
2.2.2.2. Stany Zjednoczone....15
2.2.2.3. Australia i Nowa Zelandia....17
2.2.2.4. Japonia....18
2.2.2.5. Republika Południowej Afryki....19
2.2.2.6. Brazylia....19
2.2.3. Przeglądy obiektów mostowych w Polsce....20
2.2.3.1. Wyposażenie terenowego inspektora mostowego....23
2.2.3.2. Zasady stosowania skali ocen punktowych....23
2.2.3.3. Systemy wspomagające gospodarkę mostową w Polsce....25
2.2.3.4. Podsumowanie....27
2.3. Nowoczesne metody w zarządzaniu infrastrukturą mostową....28
2.4. Podsumowanie przeglądu literatury w kontekście rozprawy....30
3. Sztuczna inteligencja – geneza, rozwój i budowa algorytmów....33
3.1. Inteligencja a sztuczna inteligencja....33
3.1.1. Zarys historyczny....35
3.1.2. Sztuczna inteligencja współcześnie....36
3.2. Metody sztucznej inteligencji....38
3.3. Sieci neuronowe....40
3.4. Konwolucyjne sieci neuronowe....44
3.4.1. Budowa konwolucyjnych sieci neuronowych....45
3.4.1.1. Warstwa konwolucyjna....46
3.4.1.2. Warstwa aktywacyjna....47
3.4.1.3. Warstwa łącząca....48
3.4.1.4. Warstwa w pełni połączona....48
3.4.1.5. Warstwa losowych przerwań....49
3.4.2. Architektury konwolucyjnych sieci neuronowych....50
3.4.3. Trening sieci....53
3.5. Sztuczna inteligencja w inżynierii budowlanej....57
3.5.1. Wizja komputerowa i sztuczna inteligencja w inżynierii budowlanej....58
4. Propozycja metody wspomagającej wykrywanie uszkodzeń powierzchni betonowych....60
4.1. Założenia projektowanego oprogramowania....61
4.1.1. Sieć neuronowa jako detektor obiektów na obrazie....62
4.1.2. Wybór architektury sieci neuronowej....66
4.1.3. Alternatywne metody uczenia sieci neuronowych....69
4.1.3.1. Fine tuning....69
4.1.3.2. Transfer learning....70

4.1.3.3. Wybór metody treningu Algorytmu....72
4.1.4. Metoda oceny proponowanego rozwiązania....73
4.2. Środowisko programistyczne i wykorzystane narzędzia....75
4.3. Kolekcja i ocena danych treningowych....76
4.3.1. Wzbogacanie danych....78
4.4. Budowa i podział zbioru uczącego wspomagane programowo....79
4.5. Implementacja metod TL....81
4.6. Implementacja detektora obiektów....85
4.7. Ocena skuteczności proponowanego rozwiązania....86
4.8. Alternatywne możliwości użycia proponowanej metody....89
5. Eksperyment porównawczy....91
5.1. Założenia eksperymentu....91
5.1.1. Sieć neuronowa uczona od wag losowych....91
5.1.2. Porównanie z aktualnym stanem wiedzy....94
5.2. Możliwości zwiększenia skuteczności proponowanego rozwiązania....95
5.3. Podsumowanie wyników eksperymentu....97
6. Sztuczna inteligencja w ekstrakcji danych z obrazu....99
6.1. Założenia rozbudowy proponowanego rozwiązania....99
6.2. Technologie umożliwiające pomiar cech fizycznych obiektu na obrazie....100
6.3. Projekt i budowa urządzenia pomiarowego....103
6.3.1. Wykorzystane komponenty i technologie....103
6.3.2. Budowa i oprogramowanie urządzenia....104
6.3.2.1. Implementacja logiki obsługi urządzenia pomiarowego....106
6.3.2.2. Implementacja oprogramowania aplikacji mobilnej....107
6.3.3. Testy dokładności urządzenia pomiarowego....108
6.4. Ekstrakcja fizycznych cech obrazu z wykorzystaniem maszyn wektorów nośnych....111
6.4.1. Zapis uszkodzeń w formie wektorowej....112
6.5. Ocena dokładności narzędzia SmartMeasure....114
6.6. Ocena przydatności rozwiązania....115
7. Podsumowanie i wnioski....118
7.1. Podsumowanie....118
7.2. Ocena stopnia osiągnięcia zakładanego celu pracy....120
7.3. Kierunek dalszych prac....121
7.4. Wnioski końcowe....123
Podziękowania....124
Bibliografia....126
Streszczenie....141
Summary....141
Załącznik z kodami źródłowymi utworzonych na potrzeby rozprawy aplikacji....142
A Data_Augmentation....143
B Dataset_builder....144
C Extract_features....148
D Train_model....149
E Moving_window....150
F Measuring_device_logic....151
G Get_px_size....153
H Get_poly....155

WYKAZ OZNACZEŃ

dist – długość zmierzonej celowej

E – końcowa wartość funkcji kosztów sieci neuronowej

f – ogniskowa

FN – (ang.) False Negative (fałszywie negatywny <wynik klasyfikacji>)

FP – (ang.) False Positive (fałszywie pozytywny <wynik klasyfikacji>)

I – macierz wejściowa do filtra konwolucyjnego

K – macierz filtra konwolucyjnego

lr – (ang.) Learning rate – współczynnik uczenia się sieci

net_j – wartość wyjściowa sieci przy obliczaniu błędu neuronu w warstwie *j*

o_j – wartość wyjściowa neuronu *o* w warstwie *j*

obj_d – wielkość obiektu w pikselach

p – wielkość zerowego wypełnienia macierzy (ang. zero padding)

s – krok macierzy (ang. stride)

$sensor_{px}$ – wielkość sensora w pikselach

$sensor_{real}$ – fizyczna wielkość sensora

TN – (ang.) True Negative (prawdziwie negatywny <wynik klasyfikacji>)

TOT – (ang.) Total samples (całkowita ilość próbek)

TP – (ang.) True Positive (prawdziwie pozytywny <wynik klasyfikacji>)

$w_{i,j}$ – waga połączenia między neuronem w warstwie *i* i *j*

SŁOWNIK SKRÓTÓW

AASHTO – American Association of State Highway and Transportation Officials

AI – Artificial Intelligence (patrz SI)

ANN – Artificial Neural Network (patrz *słownik terminów...* – sztuczna sieć neuronowa)

BMS – Bridge Management System

CNN – Convolutional Neural Network (patrz *słownik terminów...* – konwolucyjna sień neuronowa)

CV – Computer Vision (patrz wizja komputerowa)

DOT – Department Of Transportation

FCN – Fully Convolutional Network

FHWA – Federal Highway Administration

GBMS – Bauwerk Management System

GDDKiA – Generalna Dyrekcja Dróg Krajowych i Autostrad

IT – Information Technology (Technika Informatyczna)

IM – Inspektor mostowy

ML – Machine Learning (patrz *słownik terminów...* – uczenie maszynowe)

MP – megapiksel

MRWA – Main Roads Western Australia

NCHRP – National Cooperative Highway Research Program

PKP – Polskie Koleje Państwowe

RGB – Red Green Blue (patrz słownik terminów – RGB)

RGB-D – Red Green Blue Depth (patrz *słownik terminów...* – RGB-D)

SANRAL – South African National Roads Agency Ltd.

SI – Sztuczna inteligencja

SHM – Structural Health Monitoring

SPM – Systematic Preventive Maintenance

SSD – Single Shot Detector

SSN – Sztuczna sieć neuronowa

TIM – Terenowy inspektor mostowy

USDOT – United States Department Of Transportation

SŁOWNIK TERMINÓW WYSTĘPUJĄCYCH W ROZPRAWIE

Algorytm wstecznej propagacji (ang. *backpropagation algorithm*): składający się z dwóch faz algorytm stosowany w trakcie treningu **sieci neuronowej** (patrz sztuczna sieć neuronowa), do obliczania gradientów funkcji straty w odniesieniu do wag poszczególnych neuronów sieci [1].

Architektura sieci neuronowej: pojęcie architektury **sieci neuronowej** odnosi się do rozmieszczenia neuronów w warstwach i wzorców połączeń między warstwami, funkcji aktywacji i metod uczenia się. Model **sieci neuronowej** i architektura **sieci neuronowej** określają, w jaki sposób sieć przekształca dane wejściowe w wyjściowe [2].

Detektor obiektów: detektor obiektów, to algorytm komputerowy uwzględniający wykorzystanie cech charakterystycznych obiektu i jego danych przestrzennych w celu odnalezienia go na obrazie cyfrowym. Wykorzystuje dane liczbowe generowane w trybie asynchronicznym przez uczenie algorytmu. Problem wykrywania obiektów można postrzegać jako odnoszący się do terminów symbolicznych w informacji wizualnej poprzez wykorzystanie struktury składniowej i semantycznej [3].

Fine tuning: jedna z metod wspomagania procesu uczenia się **sieci neuronowej** polegająca na ponownym treningu konwolucyjnej **sieci neuronowej**, począwszy od wag wynikających z treningu przeprowadzonego na podobnym zbiorze danych [4].

Głębokie uczenie (ang. *Deep Learning, DL*): głębokie uczenie, to poddziedzina **uczenia maszynowego** (patrz **uczenie maszynowe**), które natomiast jest poddziedziną sztucznej inteligencji. Są to wielopoziomowe metody reprezentacyjne, składające się z prostych, lecz nieliniowych modułów transformujących dane wejściowe na każdym z poziomów algorytmu. Kluczowym aspektem głębokiego uczenia jest fakt, że moduły te formułowane są poprzez ogólne procedury uczenia się algorytmu [5].

Graf: uporządkowana struktura danych, składająca się z wierzchołków i krawędzi, przy czym poszczególne wierzchołki (zwane też węzłami) mogą być połączone krawędziami (skierowanymi lub nieskierowanymi) w taki sposób, iż każda krawędź zaczyna się i kończy w którymś z wierzchołków [6].

Konwolucyjna sieć neuronowa (ang. *Convolutional Neural Network, CNN*): konwolucyjna sieć neuronowa jest rodzajem algorytmu głębokiego uczenia się, korzystającym w swojej architekturze z warstw wykonujących operację konwolucji lub wzajemnej korelacji [7].

Krotka: uporządkowana struktura danych, najczęściej pojedynczego typu. Wartości występujące w krotce traktować można jako ciąg matematyczny o ustalonej długości.

Punktowa ocena stanu technicznego obiektu: ocena w skali od 0 do 5, nadawana elementowi lub jego części składowej na podstawie wizualnej inspekcji przeprowadzonej przez t**erenowego inspektora mostowego**, zgodnie z zasadami stosowania skali punktowej ocen GDDKiA w trakcie przeglądu podstawowego lub rozszerzonego [8].

RGB: model przestrzeni barw opisanymi przez trzy współrzędne, odpowiadające kolejno intensywności koloru czerwonego (R), zielonego (G) i niebieskiego (B) na obrazie. Na podstawie różnych kombinacji intensywności barw składowych, otrzymać można szeroką gamę barw pochodnych. W technologii komputerowej, najczęściej stosowany zapis barwy to **krotka** składająca się z trzech liczb 8-bitowych odpowiadających każdemu z kolorów. Dzięki takiemu zapisowi otrzymać można ponad 16 milionów

barw. Tym samym akronimem, powszechnie określa się także obrazy cyfrowe zapisane w tej przestrzeni barw.

RGB-D: standardowy model RGB obrazu rozszerzony o czwarty kanał zawierający informacje o odległości punktu od kamery. Podobnie jak w modelu RGB, najczęstszym zapisem w technologii komputerowej jest **krotka**, zawierająca cztery 8-bitowe liczby. Akronim ten stosowany jest także jako określenie obrazu cyfrowego zapisanego przy pomocy tego modelu.

Segmentacja semantyczna (ang. s*emantic segmentation*): **segmentacja semantyczna**, lub segmentacja obrazu, to zadanie polegające na utworzeniu klastrów z pikseli obrazu zawartych w tej samej klasie znajdującego się na nim obiektów. Jest to zatem zadanie polegające na predykcji klasy obiektu na poziomie pojedynczego piksela [9].

Segmentacja instancji (ang. *instance segmentation*):operacja analogiczna do **segmentacji semantycznej**, jednak wzbogacona o podział wydzielonych z obrazu klastrów na niezależne obiekty (instancje) [10].

Sztuczna sieć neuronowa (ang. *Artificial Neural Network, ANN*): klasa algorytmów **uczenia maszynowego**, które uczą się na podstawie danych i specjalizują się w rozpoznawaniu wzorców, a ich powstanie było inspirowane strukturą i funkcją mózgu. Do rodziny algorytmów ANN wlicza się głębokie uczenie i w większości przypadków oba te terminy stosuje się zamiennie [11].

Transfer learning: jedna z metod wspomagania procesu uczenia się **sieci neuronowej** polegająca na wykorzystaniu jako ekstraktora cech charakterystycznych konwolucyjnej **sieci neuronowej** trenowanej na podobnym zbiorze danych, bez ponownego treningu na docelowym zbiorze danych [12].

Terenowy inspektor mostowy: pracownik rejonu dróg , odpowiedzialny za przeprowadzanie, dokumentowanie oraz przechowywanie dokumentacji przeglądów obiektów mostowych. W trakcie prowadzenia robót na obiekcie mostowym, jest odpowiedzialny także za ich koordynowanie, nadzorowanie i odbiór. Termin ten pierwotnie był stosowany do pracowników GDDKiA. W rozprawie używany jest zamiennie z jego uogólnieniem – inspektor mostowy, niespecyficznym dla pojedynczej jednostki zarządzającej obiektem mostowym.

Uczenie maszynowe (ang. *Machine Learning, ML*): **uczenie maszynowe** jest uważane za poddziedzinę sztucznej inteligencji, która zajmuje się algorytmami i technologiami uzyskiwania wiedzy z danych. Jako produkt informatyki, **uczenie maszynowe** próbuje podejść do problemów algorytmicznie, a nie wyłącznie za pomocą metod matematycznych [13].

Wizja Komputerowe (ang. *Computer Vision, CV*): interdyscyplinarna dziedzina naukowa zajmująca się zagadnieniem uzyskania możliwości wysokopoziomowej interpretacji obrazów cyfrowych przez komputery [14].

1. WPROWADZENIE

Obiekty mostowe, to kluczowe i najbardziej rozpoznawalne obiekty inżynierskie, które należą do krytycznych elementów infrastruktury drogowej. Stanowią rodzaj drogowych obiektów inżynierskich, obok tuneli, przepustów i konstrukcji oporowych. Jedną z ich najważniejszych cech, jest bezpieczeństwo. Odnosi się ono tak do aspektu nośności i stateczności jak do bezpieczeństwa użytkownika obiektu oraz zdolności do zachowania tych cech w ciągu cyklu życia obiektu (trwałość obiektu inżynierskiego) [15]. Jest to jednocześnie jego najistotniejsza cecha z punktu widzenia potrzeb społeczeństwa będącego docelową grupą użytkowników. Katastrofy budowlane obiektów mostowych, oprócz wywołania olbrzymich kosztów materialnych i społecznych, są zawsze bardzo spektakularne i zwykle szeroko nagłośniane w mediach. To niestety pośrednio przyczynia się też do obniżania poziomu społecznego zaufania do bezpieczeństwa całej infrastruktury publicznej.

To właśnie tragiczne i głośne katastrofy mostów o wysokim znaczeniu na świecie [16] przyczyniły się do rozwoju pierwszych systemów zarządzania obiektami mostowymi w latach 60. (ang. Bridge Management System, BMS) [17]. Regulacje w tym zakresie wymusiły wykonywanie regularnych inspekcji. Przede wszystkim w formie cyklicznej wizualnej oceny stanu technicznego. Pomimo tego obowiązku i obiektywnej poprawy bezpieczeństwa infrastruktury mostowej, na całym świecie wciąż dochodzi do poważnych awarii i kolejnych tragedii.

W Polsce, obowiązek przeprowadzania okresowych przeglądów stanu technicznego obiektów mostowych nałożony został na zarządców dróg publicznych już w 1985 przez Ministerstwo Komunikacji [18], choć wcześniej po katastrofie mostu na ulicy 3 maja w Zabrzu w 1959 roku wprowadzono obowiązek założenia ewidencji mostów kolejowych z kartą rysunkową [19] oraz wprowadzono pojęcie utrzymania mostowego obiektu drogowego [20]. Natomiast stosowane do dziś instrukcje w tym zakresie zostały opracowane na przełomie stuleci [21]. Niestety nie były one przygotowane do elastycznego dopasowywania ich do rozwoju technologicznego w inżynierii lądowej. Szczególnie w odniesieniu do dynamicznie zmieniających się narzędzi informatycznych z grupy IT (ang. Information Technology). Na uwagę zasługuje także fakt, iż w świetle zwiększających się w sposób ciągły nakładów na utrzymanie obiektów infrastruktury drogowej, obecnie obowiązujące w praktyce metody przeprowadzania przeglądów obiektów mostowych okazują się być przestarzałe [22]. Nie rozwiązują tego problemu także próby implementacji elektronicznych systemów monitoringu stanu technicznego typu SHM (ang. Structural Health Monitoring) w najważniejszych

mostach. Zarządcy wciąż nie potrafią określić swoich potrzeb w tym zakresie, a twórcom tych systemów nie udaje się utworzyć wystarczająco skutecznych algorytmów identyfikujących uszkodzenia i alarmujących o zagrożeniach. Pokazuje to, że wizualne inspekcje wykonywane przez inspektorów mostowych wciąż będą potrzebne. Systemy SHM nie są w stanie ich całkowicie zastąpić.

W niniejszej pracy podjęto próbę wprowadzenia nowoczesnych technologii wykorzystujących metody sztucznej inteligencji (SI) i wizji komputerowej (ang. Computer Vision, CV) na potrzeby przeprowadzania przeglądów obiektów mostowych, w szczególności zbudowanych z betonu. Podjęto się także oceny ich wpływu na efektywność przeprowadzania przeglądu z uwzględnieniem różnych technik treningu algorytmu sztucznej inteligencji oraz porównano algorytm opracowany na potrzeby pracy z aktualnym stanem wiedzy w tej dziedzinie. Zaprezentowano także połączenie technik uczenia maszynowego z tradycyjnymi metodami wizji komputerowej wspieranymi przez autorskie układy elektroniczne jako punkt wyjścia do dalszej automatyzacji zagadnienia przeglądów obiektów mostowych i połączenia go z informacyjnymi modelami BIM (ang. Building Information Modeling) tych obiektów.

1.1. UZASADNIENIE WYBORU TEMATU PRACY

Na wybór tematu tej pracy duży wpływ miał opis kilku tragicznych wydarzeń. W pierwszym, około północy, 2 sierpnia 2016 roku, Surjeet Kumar, mechanik z Mahad w Indiach został wyrwany ze snu przez potężny huk. Gdy dobiegł do okna, zobaczył światła samochodów gasnące w toni rzeki Savitri. W katastrofie ponad stuletniego mostu na rzece Savitri, wybudowanego jeszcze przez Imperium brytyjskie, życie straciło 28 osób [23]. Po wielogodzinnym ulewnym deszczu, 14 sierpnia 2018 roku, we włoskim mieście Genua doszło do innej katastrofy – zawaleniu uległ pylon wraz z przylegającymi przęsłami ponad pięćdziesięcioletniego mostu zaprojektowanego przez Riccardo Morandiego (viadotto Polcevera w ciągu autostrady A10), pochłaniając ze sobą kolejne 43 ofiary. Poza olbrzymią tragedią dla rodzin ofiar i całej społeczności regionu, katastrofa ta poskutkowała także szkodami na ponad 350 mln euro [24]. 14 marca 2019 roku doszło do kolejnej katastrofy obiektu mostowego. W wyniku awarii, zawaleniu uległa kładka piesza na stacji kolejowej w Bombaju, odbierając życie sześciu i poważnie raniąc 30 osób [25].

Wszystkie trzy wymienione powyżej katastrofy, mimo że wydarzyły się w różnych miejscach, dotyczyły obiektów o różnym wieku, przeznaczeniu i konstrukcji, dzielą ze sobą jedną cechę wspólną. Wszystkie spowodowane były nieprawidłowościami lub zaniedbaniami w kluczowym dla bezpieczeństwa obiektu elemencie ich utrzymania – okresowych przeglądach stanu technicznego. Mimo ich regularnego przeprowadzania, manualne zarządzanie pozyskanymi w ich takcie danymi nie pozwoliło na dostatecznie szybkie uruchomienie łańcucha decyzyjnego. Tym samym, nakłady w

wysokości ułamka kwoty szkód spowodowanych katastrofami, potencjalnie mogłyby uratować życie lub zdrowie poszkodowanych oraz zapobiec marnotrawstwu znacznej ilości środków publicznych.

Efektywność wydatkowania środków publicznych to kolejny ważny aspekt wskazujący na istotność poruszanego w rozprawie zagadnienia. Badania wskazują [26], że w zależności od sposobu utrzymania obiektu infrastruktury mostowej, całkowity koszt jego cyklu życia może być mniejszy nawet czterokrotnie przy zastosowaniu utrzymania prewencyjnego (bieżącej naprawy uszkodzeń obiektu) w połączeniu z naprawami kluczowymi dla stanu technicznego obiektu niż przy zastosowaniu wyłącznie napraw elementów w stanie krytycznym. Aby strategia ta została poprawnie wprowadzona w życie, obiekt mostowy musi być cyklicznie poddawany dokładnym przeglądom stanu technicznego, pozwalającym na odszukanie i identyfikację uszkodzeń drobnych, przed ich rozwinięciem się w uszkodzenia zagrażające konstrukcji obiektu. Jednocześnie wraz z wciąż rosnącymi na świecie nakładami na utrzymanie istniejących obiektów oraz przy stopniowym ograniczaniu wydatków na budowę nowych [27,28], nie widać, aby tempo wdrażania nowych technik inspekcji nadążało za rozwojem technologii. Zwłaszcza technologii cyfrowych. Dodatkowym, alarmującym czynnikiem związanym z obecnymi standardami przeprowadzania przeglądów infrastruktury mostowej jest opisana w badaniach niska dokładność detekcji uszkodzeń przez inspektora [29] i subiektywizm nadawania obiektom oceny [30]. Stwierdzono, że przy najczęściej przeprowadzanych inspekcjach wizualnych, poniżej 30% inspektorów odnotowuje uszkodzenia drobne w postaci rys ustroju nośnego, a oceny nadawane elementom obiektów, mimo stosowania się do obowiązujących wytycznych wahają się w zakresie dwóch punktów od średniej wartości oceny dla 95% elementów.

Niska skuteczność działań inspektorów przeprowadzających przeglądy obiektów infrastruktury mostowej może wynikać z wielu czynników. Od wykorzystywania w trakcie pracy przestarzałych technik, przez trudne i niebezpieczne warunki pracy (patrz Rys. 1) po liczbę zadań do zrealizowania w ciągu roku. Oprócz corocznie budowanych nowych obiektów mostowych wymagających odbiorczego, a następnie cyklicznych przeglądów (średni wzrost liczby obiektów zarządzanych przez GDDKiA w latach 2012-2016 wynosił ponad 450 obiektów na rok [31]), z roku na rok rośnie liczba obiektów wysłużonych, z dużą liczbą uszkodzeń. Obecny globalny trend, potwierdzony przez dane statystyczne z USA i Japonii wskazuje, że średni wiek obiektu mostowego zbliża się lub nawet przekracza 50 lat [32,33]. Rozwój sieci infrastruktury drogowej nie idzie jednak w parze ze zwiększającą się liczbą inspektorów zatrudnionych do jej zarządzania. Według danych statystycznych Najwyższej Izby Kontroli, zatrudnienie w Generalnej Dyrekcji Dróg Krajowych i Autostrad między 2014 a 2018 rokiem nieznacznie spadło [34]. Struktura zatrudnienia wygląda podobnie w innych krajach – poziom zatrudnienia w amerykańskiej agencji FHWA w korespondującym okresie spadł o około 7% [35]. Oznacza to, że przy zwiększającej się liczbie obiektów wymagających kontroli i braku wdrażania nowoczesnych metod zwiększających wydajność pracy inspektorów mostowych (IM), zmniejsza się jednocześnie ich liczebność.

Rys. 1: Warunki pracy inspektora mostowego. Źródła: a) Harcon Corporation, b) Working AT Height [243]

Na ten oczywisty problem odpowiedzią może być zwiększenie efektywności i wydłużenie efektywnego czasu pracy terenowego inspektora mostowego dzięki zastosowaniu nowoczesnych metod wykorzystujących wizję komputerową oraz algorytmy sztucznej inteligencji.

W przeciągu ostatnich dwóch dekad dokonał się zauważalny postęp tak w dziedzinie fotografii cyfrowej jak i mocy obliczeniowej komputerów osobistych i biurowych. W samej technologii telefonów komórkowych, średnia wielkość sensora kamery wzrosła od około 0,3 MP (megapikseli) w 2002 do prawie 14 MP w 2017 roku [36]. Trend ten jest jeszcze bardziej widoczny w sensorach bardziej nowoczesnych kamer, które już wchodzą na rynek. Ich maksymalna rozdzielczość przekracza obecnie 150 MP. Pozwala to na dokładną rejestrację niewielkich szczegółów, takich jak rysy na powierzchniach betonowych, lub pozyskiwanie obrazów z dużo większych odległości, co wcześniej nie było możliwe. Z uwagi na relatywnie duże zapotrzebowanie na moc obliczeniową algorytmów sztucznej inteligencji, także rozwój technologii w tej dziedzinie miał kluczowy wpływ na prezentowane w rozprawie zagadnienie. W ciągu ostatnich dwudziestu lat, średnia ilość operacji wykonywanych przez procesor w trakcie jednego cyklu pracy wszystkich rdzeni procesora zwiększyła się ponad stukrotnie. Zależność ta pozostaje do tej pory w zgodzie z prawem Moore'a [37], na podstawie którego oczekiwać można dalszego, szybkiego postępu w rozwoju wymagających obliczeniowo algorytmów komputerowych także w przyszłości.

Dynamiczny rozwój tej dziedziny obserwowany jest we wiodących ośrodkach badawczych (np. Uniwersytet Stanforda, MIT), lecz skupiają się one na technologiach dostosowanych do potrzeb korporacji działających w branży IT, a więc podmiotów mających dostęp do funduszy znacznie przekraczających możliwości finansowe jednostek zarządzających infrastrukturą drogową. Z tego względu, jak na razie zainteresowanie tematem rozwoju prezentowanych w rozprawie technologii w inżynierii budowlanej jest wciąż ograniczone. Zwrócić uwagę należy jednak na prekursorów stosowania algorytmów sztucznej inteligencji w Polsce. Na pewno należy do nich profesor Zenon

Waszczyszyn z Politechniki Krakowskiej [38] razem ze swoimi następcami. Wprowadzają oni algorytmy SI do wizji komputerowej również w inżynierii budowlanej [39].

Odnotować należy także fakt istnienia metod wykorzystujących wyłącznie tradycyjną wizję komputerową w zagadnieniu przeprowadzania przeglądów technicznych obiektów mostowych (np. [40,41]). Te jednak nie cechują się wystarczającą dokładnością czy niezawodnością wskazań w stosunku do metod prezentowanych w niniejszej rozprawie.

Wszystkie wymienione powyżej przesłanki przemawiają za wyborem przedstawionego tematu rozprawy dyplomowej i pozwoliły na przyjęcie tezy i celów pracy.

1.2. TEZY I CELE PRACY

Przedmiotem rozprawy jest ocena możliwości zastosowania metod sztucznej inteligencji i wizji komputerowej do wspomagania inspekcji obiektów infrastruktury drogowej, a w szczególności obiektów mostowych. Oczywiście prezentowane tutaj rozwiązania przenieść można również na inne typy konstrukcji. Skupiają się one na wizualnej detekcji pojedynczego typu uszkodzenia (rysa o rozwartości nieprzekraczającej 0,2 mm), lecz podstawa ich funkcjonowania jest uniwersalna i może zostać zastosowana do innych rodzajów defektów, specyficznych dla różnych typów konstrukcji. Jednym z istotnych założeń pracy było zoptymalizowanie opisywanych algorytmów SI pod kątem zasobów jednostek o ograniczonym budżecie tak, aby były one możliwe do zaimplementowania na każdym szczeblu zarządzania obiektem budowlanym.

Główna teza pracy brzmi następująco:

Wdrożenie algorytmów sztucznej inteligencji do analizy obrazów pozyskanych w trakcie wykonywania okresowego przeglądu obiektu mostowego podniesie efektywność pracy terenowego inspektora mostowego, a zastosowanie technik transfer learningu zwiększy wszechstronność i ogólnodostępność proponowanego rozwiązania w porównaniu do klasycznie uczonych algorytmów SI.

Do wykazania tezy głównej pracy posłużyły następujące tezy pomocnicze:

- Dzięki zastosowaniu techniki transfer learningu, w inżynierii budowlanej możliwe jest efektywne wykorzystanie sieci pierwotnie trenowanych na powszechnych, ogólnych zbiorach danych niezwiązanych z zagadnieniem uszkodzeń powierzchni betonowych.
- Uzyskany w wyniku pracy algorytm będzie miał zdolność do generalizowania wiedzy między różnymi przypadkami powierzchni betonowej w podobnym stopniu co algorytm wykorzystujący sieć neuronową z wagami wszystkich warstw inicjalizowanymi w sposób losowy.

- Metody wizji komputerowej oraz automatyzacja pomiaru długości celowej, umożliwiają automatyczne pozyskanie fizycznych wymiarów uszkodzenia na obrazie.

Cele pracy, których realizacja potwierdzi tezy główne i pomocnicze:

- przegląd literatury w zakresie obowiązujących w kraju i na świecie standardów przeprowadzania okresowych przeglądów obiektów mostowych,
- przegląd literatury pod kątem nowoczesnych technik w utrzymaniu i zarządzaniu infrastrukturą mostową,
- ocenę aktualnego stanu wykorzystania technik sztucznej inteligencji w inżynierii budowlanej oraz w aspekcie przeglądów obiektów infrastruktury drogowo-kolejowej,
- przedstawienie metod pozyskiwania zbiorów uczących oraz budowania, uczenia i wykorzystania konwolucyjnych sieci neuronowych do analizy obrazu,
- budowę zbioru algorytmów uczenia maszynowego umożliwiających detekcję uszkodzeń na powierzchni betonu oraz test jego dokładności na uszkodzeniu typu rysa o rozwartości poniżej 0,2 mm,
- budowę podręcznego urządzenia pomiarowego działającego w systemach operacyjnych popularnych smartfonów i tabletów wraz z dedykowanym algorytmem, który pozwala na odczyt fizycznych cech uszkodzenia z obrazu cyfrowego,
- krytyczną ocenę możliwości zastosowania proponowanych rozwiązań w praktyce wykonywania przeglądów obiektów budowlanych.

1.3. OMÓWIENIE UKŁADU PRACY

Na niniejszą rozprawę składa się siedem rozdziałów. W pierwszym umieszczony został wykaz oznaczeń, słownik skrótów, a także uzasadnienie wyboru tematu pracy, jej tezy, główne cele oraz zakres. Znalazło się w nim także omówienie układu pracy oraz wykorzystanych metod badawczych.

Drugi rozdział zawiera przegląd literatury, na którym oparta została część zasadnicza pracy. Rozdział ten podzielony został na części odpowiadające kolejnym zagadnieniom poruszanym w rozprawie. W częściach tych wydzielony został opis aktualnego stanu wiedzy na temat metod przeprowadzania przeglądów infrastruktury mostowej na świecie i w Polsce, praktyka prowadzenia inspekcji oraz przegląd prac naukowych w zakresie nowoczesnych technologii w zarządzaniu infrastrukturą. Część ta skupia się tak na technologiach będących aktualnie w powszechnym użytku, jak i na najnowszych metodach przed wdrożeniem ich w oficjalnych instrukcjach.

Rozdział trzeci skupia się na omówieniu pojęcia sztucznej inteligencji. Rozpoczyna się od ogólnego opisu SI w ujęciu rozwoju i zastosowania metod, a następnie przechodzi do krytycznej

oceny przeszłych i obecnych prognoz dotyczących sztucznej inteligencji. W dalszej części rozdział opisuje metody SI w kontekście obecnego wykorzystania w inżynierii budowlanej oraz obecne w literaturze połączenie z wizją komputerową. Następnie przedstawiony jest opis teoretyczny głębokich metod SI wykorzystywanych w rozprawie – klasycznych sieci neuronowych oraz ich wariantu – konwolucyjnych sieci neuronowych. Opis ten został dodatkowo podzielony na sekcje dotyczące kolejno budowy, architektury i treningu sieci. Ostatnia część rozdziału zawiera przykładowe zastosowania konwolucyjnych sieci neuronowych, różne od tych wymienionych w pierwszej części rozdziału.

W czwartym rozdziale przedstawiona została propozycja metody wykorzystującej wizję komputerową oraz sztuczną inteligencję do wspomagania inspekcji obiektu mostowego. Opisane zostały narzędzia, które posłużyły do jej utworzenia, środowisko programistyczne oraz wykorzystane biblioteki. Następnie opisany został proces zbierania i oceny zbioru uczącego z wykorzystaniem autorskich algorytmów wizji komputerowej wraz z opisem metod sztucznego rozszerzania danych zbioru uczącego. W kolejnej części rozdział skupia się na przedstawieniu proponowanej architektury sieci, a następnie przechodzi do opisu wykorzystanych technik wspomagających jej uczenie, z których ostatecznie zostaje wybrana jedna. Rozdział zawiera także opis metod, które umożliwiają wykorzystanie sieci poddanej treningowi jako detektor obiektów na obrazie oraz ocenę skuteczności w wykrywaniu rys na powierzchniach betonowych proponowanego rozwiązania. W tym rozdziale przedstawiony został opis opracowanego na potrzeby pracy narzędzia inżynierskiego – zbioru algorytmów uczenia maszynowego KrakN, służącego do automatyzacji wykrywania uszkodzeń powierzchni betonowych.

Piąty rozdział zawiera opis eksperymentu badawczego, będącego podstawą do potwierdzenie głównej tezy rozprawy. Opisane zostały założenia tego eksperymentu, a także jego przebieg. Wyznaczono wskaźniki wydajności sieci między proponowanym rozwiązaniem, a siecią, której wagi przed treningiem inicjalizowane były od wartości losowych. Porównane zostały także możliwości treningu algorytmów przy wykorzystaniu budżetowych stacji obliczeniowych odpowiadających możliwościom typowego komputera klasy biurowej, stacji specjalistycznej wykorzystującej dedykowany procesor obliczeniowy oraz maszyny wykonującej obliczenia w chmurze. Proponowane rozwiązanie jest też porównane z aktualnymi, wykazującymi się największą dokładnością rozwiązaniami obecnymi w literaturze. W końcowej części rozdziału opisane zostały dodatkowe możliwości zwiększenia skuteczności wykrywania uszkodzeń proponowanego rozwiązania oraz podsumowanie wyników eksperymentu.

Rozdział szósty podejmuje tematykę powiązania bezwymiarowego obrazu cyfrowego RGB (patrz *słownik terminów*... – RGB) z fizycznymi wymiarami obiektów na obrazie. Przedstawione zostały techniki pozwalające na pomiar fizycznych cech obiektów na obrazie, a następnie projekt i budowa urządzenia współpracującego z systemem Android, umożliwiającego pomiar długości celowej

w trakcie przechwytywania obrazu. Moduł ten otrzymał własną nazwę – SmartMeasure i został dokładnie opisany tak ze względu na jego fizyczną budowę, jak i oprogramowanie logiki jego działania. Ostatnia część rozdziału przedstawia możliwości ekstrakcji cech fizycznych obiektów na obrazie z wykorzystaniem zbudowanego dodatku oraz metod uczenia maszynowego, a także podejmuje się oceny dokładności opracowanego rozwiązania.

Ostatni rozdział rozprawy to podsumowanie pracy oraz wnioski wyciągnięte z przeprowadzonych badań. Oceniony został w nim stopień osiągnięcia zakładanego w rozprawie celu. Rozprawa została także podsumowana oraz omówione zostały kierunki prowadzenia dalszych prac.

1.4. METODY BADAWCZE STOSOWANE W PRACY

Aby udowodnić stawiane tezy, rozprawa wykorzystuje szereg metod badawczych. Podstawowa z nich, powszechnie stosowana w pracach naukowych, to metoda analizy i krytyki piśmiennictwa. Jest to metoda mająca charakter pomocniczy. Z jej pomocą przeprowadzony został przegląd literatury w drugim rozdziale, a także opis metod sztucznej inteligencji w pierwszej części rozdziału trzeciego. Wykorzystana została także do krytycznej oceny źródeł przywołanych w rozprawie oraz do wskazania luk w obecnym stanie wiedzy i metodach stosowanych przy inspekcjach obiektów mostowych. Dzięki niej możliwe było usystematyzowanie obowiązujących w tej dziedzinie pojęć, a także stanowiła ona punkt wyjścia do dalszych badań.

Ostatnią z metod badawczych wykorzystanych w rozprawie jest metoda eksperymentalna. Polega ona na wprowadzeniu do procesu badawczego czynnika eksperymentalnego – w przypadku rozprawy, techniki transfer learning do treningu sieci neuronowej oraz dodatkowej aparatury pomiarowej w procesie pobierania obrazu, a następnie obserwacji zmian powstałych na jego skutek. Ponadto, metoda ta weryfikuje poprawność ustalonego prawdopodobnego przebiegu zmian eksperymentu po wprowadzeniu czynnika eksperymentalnego. Jest to tym samym metoda, dzięki której zostają udowodnione teza główna oraz tezy pomocnicze rozprawy. Została ona wykorzystana w rozdziale piątym i szóstym rozprawy.

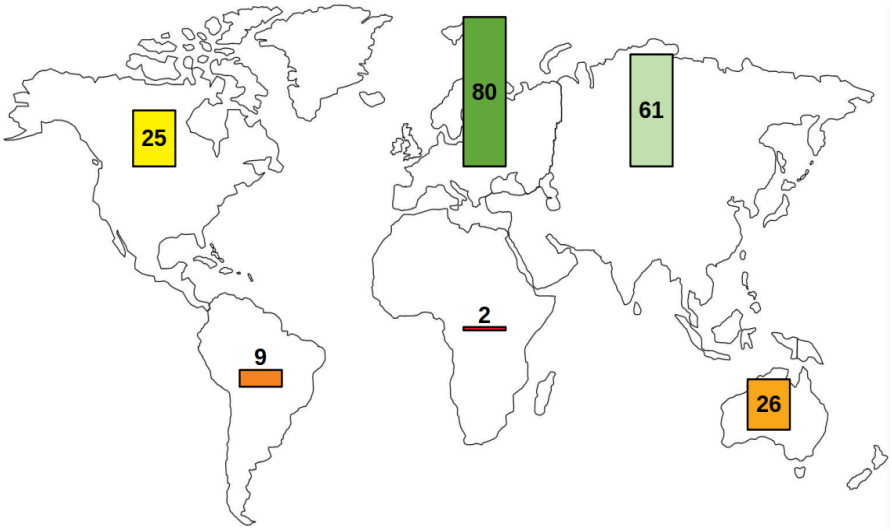
2. PRZEGLĄD LITERATURY

Na potrzeby rozprawy w pierwszej kolejności wykorzystana została metoda analizy i krytyki dostępnego piśmiennictwa do przeprowadzenia przeglądu literatury obejmującej dziedzinę inspekcji okresowych infrastruktury mostowej oraz badań stanu technicznego konstrukcji. Przegląd ten podzielony został na trzy główne części: aktualny stan wiedzy, praktyki stosowane w inspekcjach obiektów mostowych w Polsce i na świecie, a także nowoczesne metody i trendy w dziedzinie w pracach o międzynarodowym zasięgu. Jednocześnie, nie obejmował on metod związanych z rozdziałem 3 prezentowanej rozprawy, gdyż jego tematyka – metody wykorzystujące sztuczną inteligencję – stanowi odrębne zagadnienie naukowe, będące poddziedziną ogólnej tematyki przeprowadzania inspekcji obiektów mostowych, a także jest studium przypadku.

Istotnym jest także zastosowany podział rozdziału. Został on wprowadzony dla podkreślenia różnic między obecną praktyką opisywanych metod, a najnowszymi osiągnięciami w dziedzinie. Tak w przypadku praktyki oraz literatury krajowej jak i zagranicznej formalnie obowiązującej w inspekcjach obiektów mostowych, istnieje wyraźny rozdźwięk między stanem wiedzy, a stanem praktycznym. Możliwe powody takiego stanu przedstawione zostały w podsumowaniu rozdziału.

W trakcie studium literatury brane pod uwagę były ogólnodostępne źródła obejmujące zagraniczne i polskie regulacje, zalecenia i instrukcje dla jednostek zarządzających infrastrukturą mostową, artykuły naukowe, konferencyjne, książki, monografie oraz podręczniki. W przeglądzie zostały wykorzystane także artykuły zawierające podsumowanie aktualnego stanu wiedzy oraz prace zbiorcze. Większość z zebranych materiałów pochodzi z internetowych zasobów prac naukowych, a także z oficjalnych stron agencji rządowych zajmujących się utrzymaniem obiektów mostowych.

Rys. 2 przedstawia podział zebranych materiałów ze względu na lokalizację jednostki badawczej autorów pracy. Widać, że niemal 70% materiałów opisujących utrzymanie i przeprowadzanie inspekcji obiektów mostowych pochodzi z Europy i Azji. Może być to wynikiem równoległego istnienia wielu miejscowych regulacji opisujących sposoby utrzymania obiektów mostowych wynikających z liczby krajów na kontynencie europejskim, podczas gdy na przykład w Ameryce Północnej większość materiałów pochodzi z dwóch krajów (USA i Kanady).



Rys. 2: Podział studiowanej literatury z uwagi na lokalizację.

W trakcie przeprowadzania przeglądu literatury dotyczącego tej części rozprawy, zebrane zostały w sumie 203 publikacje z zakresu utrzymania, przeprowadzania inspekcji obiektów mostowych, zarządzania infrastrukturą mostową, nowoczesnych metod identyfikacji uszkodzeń mostów oraz naprawy i zabezpieczania obiektów mostowych przed uszkodzeniami. Większość ze studiowanych prac (169 publikacji) stanowi artykuły naukowe i materiały konferencyjne, natomiast pozostałe to lokalne instrukcje i rozporządzenia dotyczące przeprowadzania inspekcji obiektów mostowych.

2.1. OCENA DOSTĘPNOŚCI LITERATURY

Znakomitą większość materiałów (ponad 86%) stanowią prace opublikowane nie wcześniej niż przed 2018 rokiem. Niewielki odsetek (ok. 5,5%) stanowią natomiast prace starsze niż 10 lat (opublikowane przed 2010 rokiem), wraz z ich późniejszymi aktualizacjami. Są to najczęściej lokalne instrukcje przeprowadzania inspekcji technicznych infrastruktury mostowej.

W przypadku raportów technicznych i instrukcji, jeśli prezentowane dane były nieaktualne, dołączone zostały do nich późniejsze aktualizacje lub przy ich braku, nie były brane pod uwagę w przeglądzie. Część zawartych w przeglądzie pozycji to dokumenty na bieżąco aktualizowane, nad którymi kontynuowane są prace mające przystosowywać je do rozwoju opisywanej dziedziny, jak na przykład [42–44]. Zapewnia to jednocześnie ścisłe powiązanie literatury wchodzącej w skład przeglądu z praktyką przeprowadzania inspekcji.

Zawarte w przeglądzie prace naukowe i referaty konferencyjne zostały wyszczególnione także z uwagi na ich popularność w serwisach cytowań. Około 70% materiałów zawartych w przeglądzie posiadało co najmniej jedno cytowanie w serwisach Web of Science lub Scopus, natomiast najliczniej cytowany artykuł posiadał 8 cytowań w innych, cytowanych pracach naukowych.

W przeglądzie literatury znalazły się także pozycje polskich autorów. Jest to przede wszystkim instrukcja do przeprowadzania przeglądów drogowych obiektów inżynierskich opracowana na zlecenie Generalnej Dyrekcji Dróg Krajowych i Autostrad [21] oraz instrukcja ID-16 wykonana na zlecenie Polskich Kolei Państwowych [45], których szerszy opis wraz z przedstawieniem zasad nadawania elementom obiektów ocen punktowych [8] znajduje się w sekcji 2.2.3.. Oprócz wymienionych instrukcji, przegląd uwzględnił także najnowsze oryginalne prace naukowe pochodzące z polskich uczelni i przedstawicieli przemysłu, opisujące między innymi nowoczesne metody stosowane w przeglądach obiektów mostowych, oraz wpływ nowoczesnych technologii cyfrowych na rozwój dziedziny utrzymania infrastruktury mostowej.

Należy mieć jednak na uwadze, iż opisywana dziedzina rozwija się w sposób niezwykle dynamiczny, a coraz większy nacisk kładziony jest na technologie wykorzystujące sztuczną inteligencję, których opis znajduje się w rozdziale 3.

2.2. AKTUALNY STAN WIEDZY I PRAKTYKA PRZEPROWADZANIA INSPEKCJI OBIEKTÓW MOSTOWYCH

Sekcja ta opisuje aktualny stan wiedzy w dziedzinie utrzymania oraz przeprowadzania przeglądów infrastruktury mostowej. Opisywana literatura dotyczy metod oraz programów i strategii utrzymania aktualnie wykorzystywanych w utrzymaniu i badaniach infrastruktury mostowej. Metody, których wykorzystanie nie jest powszechne lub ogranicza się obecnie wyłącznie do testów, opisane zostały w sekcji 2.3.

2.2.1. AKTUALNY STAN WIEDZY W SYSTEMACH UTRZYMANIA I ZARZĄDZANIA INFRASTRUKTURĄ MOSTOWĄ

W świetle wykonanego studium literatury, kierunki rozwoju aktualnie wykorzystywanych metod utrzymania i zarządzania infrastrukturą mostową, podzielić można na cztery główne grupy:

- metody ciągłego monitorowania stanu technicznego obiektu mostowego oraz nowoczesne metody prowadzenia pomiarów na obiekcie,
- opis uszkodzeń, czynników mogących wpływać na powstawanie i rozwój defektów obiektów mostowych oraz metody zarządzania uszkodzeniami,
- techniki naprawcze, rehabilitacyjne oraz modernizacyjne,
- budowa i ewaluacja modeli zarządzania, utrzymania, optymalizowania procesu zarządzania oraz szacowania całkowitych kosztów życia obiektów mostowych.

Każda z wymienionych powyżej grup rozwoju opisywanej dziedziny znajduje odzwierciedlenie w bieżąco publikowanych artykułach oraz sprawozdaniach technicznych agencji rządowych

zarządzających infrastrukturą drogową, jednak techniki w nich opisywane stosowane są wyłącznie w projektach pilotażowych.

Do pierwszej z wymienionych grup zaliczyć można opisy zastosowań nowoczesnych czujników i urządzeń pomiarowych, dokładne opisy technik monitorowania mostów, studia przypadków łączące obie poprzednie dziedziny, a także prace opisujące techniki monitorowania infrastruktury wykorzystujące wiele technik pomiarowych. W pracach przedstawiających działanie czujników, zwrócić uwagę można na stosowanie czujników światłowodowych [46] oraz coraz szersze zastosowanie sensorów bezprzewodowych [47–49]. Wiele prac opisuje także wykorzystanie czujników typu MEMS [50,51] czy wykorzystanie pomiarów impedancji elektromechanicznej do detekcji i monitorowania rozwoju uszkodzeń [52]. Na uwagę zasługują także prace łączące wykorzystanie czujników w połączeniu z modelami autoregresywnymi, pozwalającymi na predykcję zmian stanu technicznego obiektu w czasie [53] oraz podejście wykorzystujące techniki skanowania laserowego w poszukiwaniu uszkodzeń [54]. Wykorzystywane jest także monitorowanie infrastruktury z wykorzystaniem wizyjnych technik DIC [55].

Prace przedstawiające techniki monitorowania obiektów mostowych skupiają się na opisie poprawnych metod stosowania i optymalnego rozmieszczenia czujników z uwzględnieniem niepewności pomiarowych [56–58], oraz ciągłego monitoringu infrastruktury mając na uwadze zarządzanie gospodarką mostową [59,60]. Znaleźć można także prace włączające do systemu monitorowania infrastruktury metodykę BIM [61].

Na osobną uwagę zasługują prace opisujące studia przypadków zastosowań konkretnych technik pomiarowych najczęściej do obiektów typu *landmark*, mostów o szczególnym znaczeniu gospodarczym lub kulturowym. Do obiektów tego typu wykorzystywane są techniki nieniszczące [62,63] lub w przypadku napraw i modernizacji – obiekt taki poddawany jest dodatkowemu monitorowaniu [64].

Kolejną z grup aktualnych kierunków rozwoju opisywanych metod, jest opis uszkodzeń, ich przyczyn i zarządzania nimi w trakcie cyklu życia obiektu mostowego. Z regionów szczególnie narażonych na konkretne typy obciążeń wyjątkowych, pochodzą prace opisujące wpływ na konstrukcje mostów między innymi: powodzi [65], wpływów sejsmicznych (trzęsień ziemi) [66,67] i huraganów [68], a także studia przypadków opisujące regiony narażone na kombinacje tych czynników [69]. Poza obciążeniami wyjątkowymi, szeroko opisane zostały też wpływy i wynikające z nich uszkodzenia związane z użytkowaniem obiektów mostowych, takie jak wpływ ruchu ulicznego i kolizji [70,71], a także pożarów na obiekcie [72]. Opisane zostały również pozostałe wpływy środowiskowe [73] wraz z idącym z nimi zagrożeniem korozyjnym, modelowanym na przykład w sposób probabilistyczny [74]. Do tej grupy zaliczyć można także prace poświęcone zarządzaniu

powstałymi uszkodzeniami mające na celu podnieść efektywność wydatkowania pieniędzy publicznych na utrzymanie infrastruktury [75].

Przedostatnią z wymienionych grup rozwoju dziedziny utrzymania i zarządzania infrastrukturą mostową, są jej naprawy i modernizacje. Z grupy tej wydzielić można trzy podstawowe działania podejmowane w celu wydłużenia cyklu życia obiektu mostowego. Pierwszym z nich są działania prewencyjne będące bezpośrednią odpowiedzią na czynniki zagrażające konstrukcji obiektu [76,77]. Kolejnym są działania polegające na modernizacji konstrukcji, dostosowującej ją do zmieniających się warunków użytkowania [78]. Ostatnim działaniem jest opracowywanie modeli optymalizujących strategie rehabilitacyjne [79].

Ostatnią grupą rozwoju omawianych metod, są metody zarządzania infrastrukturą mostową. Skupiają się one na przewidywaniu zapotrzebowania na nowe obiekty i naprawy oraz modernizacje już istniejących przy uwzględnieniu niszczącego wpływu środowiska [80] i ciągłego wzrostu obciążenia użytkowego mostów [81,82]. W odróżnieniu jednak od prac związanych z zarządzaniem pojedynczym uszkodzeniem lub obiektem, prace te skupiają się na strategicznym zarządzaniu budową i planowaniem sieci obiektów mostowych na wyznaczonym obszarze [83–85]. Inne opracowania dotyczące tej grupy rozwoju technologii zarządzania infrastrukturą mostową przedstawiają problematykę zarządzania ryzykiem związanym z obsługą sieci mostów w sposób analogiczny do problematyki zarządzania ryzykiem związanym z kapitałem [86]. Ostatnia grupa prac związana z zarządzaniem obiektami mostowymi na poziomie strategii utrzymania, prezentuje możliwości wdrożenia systemów BIM do cyklu życia obiektów mostowych [87], także w połączeniu z SHM w zarządzaniu gospodarką mostową [88]. Przedstawia także alternatywne jednostki stosowane w systemach SHM w przypadku braku możliwości posługiwania się wielkościami walutowymi [89].

Wymienione w sekcji grupy i metody w dziedzinie utrzymania infrastruktury mostowej, pomimo dużego rozbudowania tematycznego, nie poruszały podstawowego zagadnienia związanego z zarządzaniem obiektami mostowymi. Przeprowadzane przez IM inspekcje bieżące i podstawowe są bowiem pierwszym krokiem, w którego następstwie wykorzystać można opisane metody. Kolejne sekcje rozprawy przedstawią inspekcje mostów od strony praktyki IM na świecie i w Polsce.

2.2.2. INSPEKCJE OBIEKTÓW MOSTOWYCH NA ŚWIECIE

Najważniejszym elementem bieżącego utrzymania sieci infrastruktury mostowej jest poprawne przeprowadzanie jej przeglądów okresowych. Jest to bowiem najczęściej pierwszy, a czasem jedyny moment, w którym inspekcyjny aparat agencji zarządzającej obiektem ma szansę na bezpośredni kontakt z konstrukcją – każde kolejne wydarzenie związane z utrzymaniem obiektu (ekspertyza, działanie naprawcze, modernizacja) zapoczątkowane jest przeglądem okresowym, a każde uszkodzenie które zostanie w jego trakcie pominięte, nie zostanie uwzględnione na żadnym kolejnym etapie zarządzania obiektem. Z tego względu, działania te mają szczególną wartość w

zarządzaniu infrastrukturą, mimo że jak wskazuje na to struktura zatrudnienia terenowych inspektorów mostowych (TIM) w Polsce oraz na świecie (IM), ich finansowanie jest ograniczane (patrz s. 1.1.).

2.2.2.1. EUROPA

Zgodnie z zestawieniem przedstawionym w sekcji 2., Europa jest kontynentem, z którego pochodzi największa liczba prac poświęconych utrzymaniu infrastruktury. Jedną z przyczyn tego jest relatywnie duża liczba niezależnych od siebie agencji zarządzających infrastrukturą drogową, spośród których każda posiada własne wytyczne utrzymania zarządzanymi obiektami, różniące się od siebie wyłącznie w niewielki sposób. Mnogość tych systemów jest przyczyną prób ich unifikacji na poziomie kontynentu, a prace w tej tematyce są regularnie publikowane [90]. Tymczasem jednak, wobec braku wdrożenia jednolitych wytycznych oceniania stanu technicznego infrastruktury mostowej na poziomie ogólnoeuropejskiem, w opisie zostanie wyszczególnionych kilka krajów europejskich.

W Wielkiej Brytanii standardy utrzymania infrastruktury mostowej zostały zunifikowane między częściami składowymi Zjednoczonego Królestwa. Są one regulowane przez dokument BD 63/07 z 2007 roku [91], który w 2020 roku został zastąpiony przez jego aktualizację – dokument CS 450 [42]. W dokumentach tych opisane są metody przeprowadzania przeglądów dla wszystkich obiektów, których długość przekracza trzy metry. Inspekcje zostały podzielone na cztery typy, które podobnie jak w Polsce różnią się od siebie zakresem. Pierwsza z nich to kontrola powierzchniowa wykonywana przez inspektorów drogownictwa. Określa ona bieżącą przydatność obiektu do użytkowania. Druga to generalna inspekcja wykonywana w dwuletnich odstępach, składająca się z kontroli wizualnej bez wykorzystania sprzętu specjalistycznego. Jest przeprowadzana dla łatwo dostępnych elementów obiektu oraz skrajni przeszkody. Kolejnym typem inspekcji jest inspekcja główna. Jest to dokładna kontrola wizualna konstrukcji wykonywana przez przeszkolony zespół co najmniej raz na sześć lat. Ostatnim typem inspekcji jest kontrola specjalna, analogiczna do polskiej ekspertyzy.

Przeglądy obiektów mostowych we Francji odbywają się w bardzo podobny sposób do brytyjskich, z uwagi na fakt iż standardy obowiązujące w obu krajach opracowywane były wspólnie przez angielsko-francuskie konsorcjum Bridge Liaison Group w latach dziewięćdziesiątych [92]. Najbardziej wyraźnymi różnicami jest jednak zakres stosowania wytycznych (dla każdego obiektu z prześwitem większym niż dwa metry) oraz odstępy czasowe między przeprowadzanymi przeglądami. Kontrole powierzchniowe odbywają się raz na trzy lata, natomiast główne w przedziałach między rokiem a dziewięcioma latami, w zależności od kondycji obiektu.

W Niemczech dokumentem opisującym metodologię przeprowadzania przeglądów infrastruktury jest DIN 1076 z 1999 roku [93]. Podobnie jak w Wielkiej Brytanii i Francji, wyróżnia

on cztery typy przeglądów obiektu mostowego i nieznacznie różni się terminami ich wykonywania. Dokument zakłada przeprowadzanie kontroli generalnej raz na trzy lata, natomiast głównej przynajmniej co sześć lat. Podobnie jak w pozostałych przypadkach, inspekcje prowadzone są głównie metodą wizualną. Ponadto agencja wdrażająca wymieniony wcześniej dokument, poleciła wdrożenie systemu informatycznego GBMS (typ systemu BMS – ang. Bridge Management System) do zarządzania danymi około dziesięciu tysięcy mostów będących pod zarządem Ministerstwa Transportu [94].

Charakterystyka utrzymania infrastruktury drogowej w Rosji także jest zbliżona do pozostałych krajów europejskich. Obecnie stosowane metody weszły do powszechnego użytku w 1996 roku, a w 2002 roku zostały aktualizowane do wykorzystania z informatycznymi systemami zarządzania gospodarką mostową [95]. Obecnie wyszczególnia się trzy typy przeglądów: stałe, bieżące i okresowe. Podobnie jak w Polsce, przeglądy przeprowadzane są w sposób wizualny, a obiekt mostowy został podzielony na jego elementy składowe (standardowe), jednak w tym przypadku podział oprócz elementów konstrukcyjnych obejmuje także połączenia i łożyska, materiał konstrukcyjny oraz wykończeniowy. Elementy należące do osobnych grup oceniane są oddzielnie. Stosowany jest także system oceny punktowej z lingwistycznym opisem uszkodzeń. Ponadto wyniki przeglądów są zapisywane w cyfrowej bazie danych MONSTR.

2.2.2.2. STANY ZJEDNOCZONE

W Stanach Zjednoczonych podstawą prawną narzucającą utrzymanie infrastruktury mostowej jest kodeks przepisów federalnych [96], zaś wdrożenie standardów utrzymania infrastruktury spoczywa na agencji rządowej FHWA. Agencją ją wspomagającą i odpowiedzialną za przygotowywanie standardów inspekcji jest NCHRP. Standardy te obowiązują na szczeblu federalnym, natomiast lokalne strategie utrzymania zatwierdzane są przez odpowiednie Wydziały Transportu (DOT) poszczególnych stanów. Dokumentami opisującymi metodykę przeprowadzania przeglądów obiektów mostowych oraz strategię efektywnego utrzymania infrastruktury mostowej są chronologicznie pierwszy „National Bridge Inspection Standards" z 1971 roku [97], „Bridge Inspection Practices" [98] wykonany na zlecenie NCHRP oraz „Bridge Preservation Guide" autorstwa FHWA [99]. Dwa ostatnie powstały pod kierownictwem Wydziału Transportu USA (USDOT).

Pierwszy skodyfikowany opis praktyki przeprowadzania przeglądów drogowych obiektów inżynierskich w USA został opracowany w następstwie katastrofy mostu Silver Bridge w Point Pleasant w 1967 roku, która pochłonęła życie 46 osób [100]. Od tamtej pory standardy te na bieżąco są aktualizowane także z wykorzystaniem aktualnych praktyk wdrażanych przez stanowe DOT oraz wykonanego w 2003 roku przeglądu kwestionariuszy zebranych przez FHWA i AASHTO (rządowej agencji transportowej) z krajów takich jak Niemcy, Francja, Wielka Brytania oraz RPA. Także z tego

względu, metody utrzymania obiektów mostowych są coraz bardziej zbliżone do tych stosowanych w Europie. W toku rozwoju własnych metod zarządzania infrastrukturą drogową, uformowany został system Pontis wykorzystywany obecnie pod nazwą AASHTOWare Bridge Management Software przez ponad 40 stanowych Wydziałów Transportu [101].

Inspekcje infrastruktury przeprowadzane są przez dwuosobowe zespoły inspektorów mostowych zatrudnionych w stanowych Wydziałach Transportu, jednak w szczególnych przypadkach do niewielkich konstrukcji może zostać oddelegowany pojedynczy pracownik. Występuje osiem różnych typów inspekcji, z których jedynie trzy: rutynowa, podwodna oraz dotycząca krytycznych dla konstrukcji obiektów odbywają się w interwałach określonych przepisami. Stosowany jest czterostopniowy system oceny stanu technicznego elementów konstrukcji.

Inspekcja rutynowa w zależności od wytycznych stanowych, odbywa się w przedziałach rocznych lub dwuletnich. Wykonywana jest metodą wizualną wspomaganą przez pomiar niwelacyjny obiektu oraz skrajni, ocenę stanu technicznego przeszkody, a w przypadku mostów drewnianych wiercenia kontrolne. Wykonywana jest z dostępem do dokumentacji poprzednich przeglądów, w celu śledzenia zmian stanu technicznego między inspekcjami. W przeciwieństwie do przeglądów przeprowadzanych w Polsce, ocena wizualna obiektu nie musi odbywać się z niewielkiej odległości (*arms-length inspection*).

Inspekcja podwodna wykonywana jest w odstępach 60 - 72 miesięcznych. Jest ona wykonywana dla jedynie 6% obiektów mostowych (mimo, że około 84% mostów w ewidencji USA przekracza jakiś typ wodnej przeszkody terenowej). Inspekcję przeprowadza się w przypadkach, w których ocena konstrukcji z brzegu i sondowanie cieku wodnego są niemożliwe (w praktyce dla cieków głębszych niż 2 metry). Jej celem jest wizualne określenie wypłukiwania materiału spod posadowienia obiektu mostowego.

Ostatnia z inspekcji przeprowadzanych w sposób regularny, to kontrola elementów krytycznych dla konstrukcji obiektu. Jest ona przeprowadzana w odstępach dwu do trzyletnich. Poddawane są jej obiekty uznawane za skomplikowane (*complex*), do których zaliczane są mosty ruchome, wiszące i podwieszane. Do ich przeglądu wyznacza się personel o szczególnym przeszkoleniu.

Pozostałe typy kontroli stanu technicznego mostów w USA nie mają odgórnie narzuconych interwałów. Do tego typów inspekcji zalicza się kontrola uszkodzenia, będąca doraźną oceną skutków np. wypadku drogowego, inspekcja bliska będąca odpowiednikiem przeglądu rozszerzonego, odbywająca się z odległości około jednego metra (*arms-length*) dla wytypowanych obiektów oraz inspekcja szczegółowa poszczególnych elementów konkretnego mostu. Ostatnie typy to przegląd wstępny przy oddaniu obiektu do eksploatacji oraz ekspertyza.

W zakresie globalnej strategii utrzymania infrastruktury, w USA przeważa taktyka napraw prewencyjnych, co może być skutkiem posiadania relatywnie młodej infrastruktury. Strategia ta zakłada ciągłe wydatkowanie stosunkowo niewielkich kwot w nakładach na czyszczenie obiektów, malowanie stali i uszczelnianie pęknięć w celu uniknięcia dużych wydatków związanych z poważnymi naprawami. W tym celu na poziomie federalnym wdrożony został program systemowej konserwacji zapobiegawczej (SPM), bazujący na modelu opisującym pogarszanie się stanu technicznego elementów mostu w czasie [102]. System ten jest powiązany z finansowaniem pomocy federalnej dla stanowych Wydziałów Transportowych.

2.2.2.3. AUSTRALIA I NOWA ZELANDIA

W Australii i Nowej Zelandii za utrzymanie obiektów mostowych odpowiada ponad 800 publicznych i prywatnych agencji [103]. Analogicznie do USA, na szczeblu stanowym istnieje 9 agencji publicznych, natomiast organem na szczeblu rządowym jest Wydział Infrastruktury, Transportu, Rozwoju regionalnego i Komunikacji (The Department of Infrastructure, Transport, Regional Development and Communications) [104].

Każda z dziewięciu lokalnych agencji posiada własne wytyczne administracyjne co do zarządzania obiektami mostowymi dłuższymi od trzech metrów, lecz w dużym stopniu pokrywają się ze sobą nawzajem. Jest to spowodowane oparciem wdrożonych wytycznych o amerykański system Pontis. Na szczeblu krajowym, procedury zostały opisane przez rządową agencję Austroads w dokumencie z 2018 roku [43]. Został on zaadaptowany także w Nowej Zelandii.

W australijskim modelu utrzymania obiektów mostowych występują trzy poziomy kontroli. Pierwszy poziom to inspekcje przeprowadzane okresowo, co sześć do dwunastu miesięcy lub wykonywane poza harmonogramem w przypadku wystąpienia czynników zagrażających konstrukcji (np. powódź). Są one przeprowadzane w sposób wizualny dla widocznych i łatwo dostępnych elementów obiektu, przez inżyniera utrzymania infrastruktury, a jej wyniki służą bezpośrednio do planowania robót doraźnych. Kolejny poziom to kontrola mająca określić obecny stan techniczny konstrukcji, przeprowadzana przez wyszkolonego inspektora mostowego w towarzystwie inżyniera projektowego lub specjalisty. Podobnie jak poprzednia, ma ona formę badania wizualnego, lecz w tym przypadku kontrolowane są z bliska wszystkie elementy konstrukcji ponad powierzchnią wody. W jej trakcie elementom konstrukcji nadawane są oceny w skali cztero- lub sześciostopniowej, a maksymalny interwał jej przeprowadzania to od czterech do dziesięciu lat w zależności od zaleceń lokalnej jednostki administracyjnej. Jest ona podstawą do wdrożenia działań naprawczych na obiekcie. Ostatni, trzeci poziom inspekcji przeprowadzany jest, gdy taka konieczność zostanie wskazana na poprzednim poziomie kontroli obiektu. Inspekcje takie przeprowadzane są przez specjalistyczne grupy mające za zadanie wykonanie testów wytrzymałościowych elementów konstrukcji, dających

informacje o jej stanie technicznym a także nośności i szacowanej pozostałej długości cyklu życia mostu. W jej trakcie wykonywane są głównie dokładne oględziny obiektu oraz testy nieniszczące, a także długotrwały monitoring obiektu.

Jednocześnie, Australia jest jednym z pierwszych krajów, w których wprowadzone zostały informatyczne systemy zarządzania gospodarką mostową. Pierwszy z nich – MRWA został wdrożony już pod koniec lat osiemdziesiątych. Jest też posiadaczem najdłużej działającego informatycznego systemu administracyjnego zarządzającego infrastrukturą drogową na szczeblu rządowym – w 2002 roku działał on już nieprzerwanie od trzydziestu lat. Inne systemy uruchamiane były w poszczególnych jednostkach administracyjnych i używane były między innymi do zbierania informacji o obiektach mostowych i ich stanie technicznym, katalogowania raportów z inspekcji oraz ustalania racjonalnego budżetu na naprawy.

2.2.2.4. JAPONIA

Pierwotnie, agencją zarządzającą infrastrukturą drogową na poziomie administracyjnym w Japonii była Japońska Spółka Dróg Publicznych powołana w 1956 roku. Zarządzała ona całą siecią drogową w kraju aż do 2005 roku, kiedy została sprywatyzowana i podzielona na trzy osobne spółki odpowiedzialne za różne regiony kraju. Na podstawie kontraktów rządowych zajmują się one planowaniem, budowaniem i zarządzaniem inżynierskimi obiektami drogowymi w Japonii, choć dopiero w 2013 roku zostały prawnie zobowiązane do stosowania prewencyjnego modelu utrzymania infrastruktury powiązanego z regularnym przeprowadzaniem kontroli stanu technicznego [105].

Okresowe przeglądy infrastruktury w Japonii wykonywane są przez czteroosobowe zespoły składające się z inspektora mostowego, jego asystenta, technika mostowego oraz pomocnika technicznego. Instrukcja [106] wymaga formalnego wykształcenia wyłącznie od głównego inspektora, natomiast wobec pozostałych osób przeprowadzających przegląd jedynie doświadczenia w wykonywanym zawodzie.

Przeglądy okresowe infrastruktury drogowej w Japonii składają się z dwóch typów inspekcji – kontroli wstępnej i rutynowej. Kontrolę wstępną przeprowadza się przed otwarciem obiektu, natomiast rutynowe w pięcioletnich interwałach. Kontrola rutynowa przeprowadzana jest w sposób wizualny, bez dodatkowego wyposażenia inspektora. W jej trakcie nie wykonuje się także pomiarów obiektu ani uszkodzeń, a punktowa ocena nadawana elementom konstrukcji zawiera się w czterostopniowej skali określającej konieczność przeprowadzenia działań naprawczych, gdzie ocena I określa brak konieczności przeprowadzania napraw, a IV stan awaryjny. Nie występują ponadto inne opisane w instrukcji typy inspekcji, lecz w trakcie przeglądu inspektor mostowy może nadać elementowi ocenę „S” oznaczającą konieczność wykonania dodatkowych badań [107].

2.2.2.5. REPUBLIKA POŁUDNIOWEJ AFRYKI

W Republice Południowej Afryki administratorem infrastruktury drogowej jest Południowoafrykańska Narodowa Agencja Drogowa (SANRAL). Oprócz niej istnieje dziewięć lokalnych jednostek zarządzających wyodrębnionymi elementami infrastruktury drogowej. W odróżnieniu od pozostałych wymienionych krajów, w RPA przeglądy obiektów mostowych wykonywane są przez zewnętrzne firmy wynajmowane w tym celu przez jednostki administracyjne, natomiast główna agencja rządowa zajmuje się wyłącznie szkoleniem prywatnych inspektorów mostowych. SANRAL jest także autorem południowoafrykańskiej instrukcji przeprowadzania przeglądów infrastruktury, z której korzystają jednostki lokalne [108].

Terenowi inspektorzy mostowi w RPA muszą spełnić dwa warunki, aby uzyskać uprawnienia do wykonywania przeglądów obiektów mostowych. Pierwszym jest pięcioletni staż pracy jako projektant konstrukcji mostowych, a drugim odbycie dwudniowego szkolenia pod okiem instruktora. Szkolenie oprócz przeprowadzania przeglądu obiektu mostowego obejmuje także naukę obsługi wdrożonego w RPA systemu BMS [109].

Kontrola stanu technicznego konstrukcji mostowych w RPA przebiega na trzech szczeblach. Podstawowym szczeblem jest bieżące monitorowanie infrastruktury. Odbywa się ono relatywnie często, lecz w sposób nieregularny, nie podlegający reżimom minimalnej częstotliwości. Personel przeprowadzający działania polegające na monitorowaniu konstrukcji zgłasza wyłączenie zlokalizowane uszkodzenia, więc w przypadku braku uszkodzeń, kontrola nie jest odnotowywana. Odbywa się ona zazwyczaj przy okazji zdarzeń drogowych lub innych wydarzeń losowych. Drugim szczeblem są inspekcje główne. Odbywają się one co maksymalnie pięć lat, zgodnie z cyklem programu naprawczego infrastruktury w RPA, a w ich trakcie korzysta się z metod wizualnych do oceny stanu technicznego obiektu. W ocenie wizualnej wykorzystuje się opracowany w RPA system ocen punktowych DERU (*Degree, Extent, Relevancy, Urgency*), wykorzystujący opis każdego uszkodzenia według jego podstawowych charakterystyk: stopnia, wielkości, znaczenia oraz pilności uszkodzenia. Stosując system DERU, każde uszkodzenie klasyfikowane jest w pięciostopniowej skali (0-4) [110]. Ostatnim szczeblem kontroli stanu technicznego infrastruktury w RPA jest inspekcja weryfikacyjna. Poddawane są jej losowo niektóre konstrukcje, a do oceny ich stanu technicznego wyznacza się inspektora mostowego z minimalnym stażem wynoszącym 17 lat.

2.2.2.6. BRAZYLIA

Infrastruktura mostowa w Brazylii podlega pod Brazylijski Federalny Departament Transportu, oraz pod pomniejsze jednostki organizacyjne mu podlegające. Bieżące utrzymanie i inspekcje są przeprowadzane na podstawie krajowej normy DNIT 010/2004 z 2004 roku [111],

podobnie jak system wdrożony w Australii, opartej częściowo o amerykański projekt Pontis a także starsze wytyczne pochodzące z USA, takie jak standardy inspekcji mostów z 1997 roku.

Inspekcje wykonywane są przez terenowego inspektora mostowego wraz z pomocnikiem technicznym. W szczególnych przypadkach, wykonywane są przez starszego inspektora, posiadającego co najmniej dziesięć lat doświadczenia w przeprowadzaniu inspekcji.

Norma wyróżnia pięć rodzajów kontroli przeprowadzanych w trakcie cyklu życia obiektu mostowego. Pierwszą z nich jest kontrola odbiorcza, wykonywana przed oddaniem obiektu do użytku. Jest dokładnie dokumentowana w sposób fotograficzny, gdyż ma służyć jako odniesienie dla wszystkich kolejnych kontroli, a także działań naprawczych prowadzonych na obiekcie. Przeprowadza się ją także po przebudowie lub modernizacji mostu. Kolejnym typem inspekcji jest kontrola rutynowa, przeprowadzana raz na dwa lata. W jej trakcie wykonuje się wizualne oględziny obiektu i sprawdza się powstanie nowych i rozwój starych uszkodzeń. Trzecim typem inspekcji są kontrole nadzwyczajne. Przeprowadza się je w następstwie zdarzeń wyjątkowych w cyklu życia obiektu, takich jak katastrofy naturalne. Mają one na celu umożliwienie szybkiej wyceny zaistniałych szkód, oszacowanie aktualnej nośności obiektu oraz przyspieszenie procesu remontu. Następnym rodzajem są wykonywane co najmniej raz na pięć lat przeglądy specjalne. Są to szczegółowe inspekcje wizualne, przeprowadzane analogicznie do przeglądu rozszerzonego w Polsce. W jego trakcie, inspekcji poddawane są trudno dostępne elementy konstrukcji oraz wykonywana jest dokładna dokumentacja fotograficzna. Przegląd ten może być także przeprowadzony gdy jego potrzeba zostanie wskazana w trakcie kontroli rutynowej. Ostatnim typem przeglądu jest inspekcja pośrednia, służąca do kontroli pojedynczych uszkodzonych elementów wskazanych w trakcie innych typów kontroli. Raporty z wymienionych powyżej przeglądów przechowuje się w formie papierowej w dokumentacji obiektu.

2.2.3. PRZEGLĄDY OBIEKTÓW MOSTOWYCH W POLSCE

W Polsce przeglądy obiektów mostowych regulowane są następującymi dokumentami:

- Ustawa z dnia 7 lipca 1994 r. − Prawo budowlane (Dz. U. 1994 Nr 89 poz. 414) nakładające na właściciela lub zarządcę obiektu budowlanego konieczność okresowej kontroli jego stanu technicznego [112],
- Ustawa z dnia 21 marca 1985 r. o drogach publicznych (Dz. U. 1985 Nr 14 poz. 60) [18], nakładająca na zarządcę drogi publicznej między innymi obowiązek przeprowadzania okresowych kontroli stanu technicznego istniejących na niej obiektów inżynierskich,
- Obwieszczenie Marszałka Sejmu Rzeczypospolitej Polskiej z dnia 6 grudnia 2013 r. w sprawie ogłoszenia jednolitego tekstu ustawy o transporcie kolejowym [113] – obwieszczenie będące aktem ujednolicającym postanowienia Ustawy z dnia 28 marca 2003 r. o transporcie

kolejowym [114], nakładającej między innymi konieczność kontrolowania stanu technicznego kolejowych obiektów inżynierskich na wymieniony w ustawie organ właściwy,

- Instrukcje przeprowadzania przeglądów drogowych obiektów inżynierskich [21] – wprowadzona na podstawie Zarządzenia nr. 35 Generalnego Dyrektora Dróg Krajowych i Autostrad instrukcja obejmująca przeglądy bieżące, podstawowe, rozszerzone i szczegółowe oraz ekspertyzy obiektów inżynierskich wraz z opisem wymaganego w trakcie inspekcji sprzętu, sposobu dokumentacji, finansowania oraz analizy wyników i procedury podejmowania decyzji na podstawie przeglądu,
- Zasady stosowania skali ocen punktowych stanu technicznego i przydatności do użytkowania drogowych obiektów inżynierskich [8] – podręcznik wydany przez GDDKiA mający na celu usystematyzowanie ocen nadawanych elementom obiektu w trakcie oceny wizualnej, zawierający przykładowe obrazy występujących uszkodzeń i będący suplementem do wymienionej wcześniej instrukcji,
- Instrukcja utrzymania kolejowych obiektów inżynierskich na liniach kolejowych do prędkości 200/250 km/h – ID-16 [45] – wydana na zlecenie PKP Polskich Linii Kolejowych (PLK) instrukcja analogiczna do GDDKiA, zawierająca wytyczne przeprowadzania oględzin, kontroli rocznej, pięcioletniej oraz przeglądu specjalnego kolejowych obiektów mostowych. Zawiera także wytyczne dotyczące dokumentacji, zasad BHP w trakcie przeglądów oraz planowania koniecznych robót utrzymaniowych wynikłych w trakcie inspekcji,
- Wytyczne oceny stanu technicznego drogowych obiektów inżynierskich – WR-M-81 [115] – bliźniaczy do *Instrukcji przeprowadzania przeglądów drogowych obiektów inżynierskich* dokument wprowadzony na szczeblu Ministerstwa Infrastruktury do dobrowolnego stosowania na każdym etapie życia drogowego obiektu inżynierskiego. Podobnie jak dokument, na którym się opiera obejmuje czynności związane z utrzymaniem obiektu, jednak odwołuje się do ogólnego zarządcy obiektu.

Jednocześnie należy zaznaczyć, że mimo iż instrukcje GDDKiA oraz PLK dotyczą różnych typów obiektów inżynierskich, to ich zawartość w dużej części pokrywa się ze sobą. Odpowiadają sobie kolejno terminy konkretnych przeglądów (przegląd bieżący – oględziny, przegląd podstawowy – kontrola roczna itd.), wyposażenie inspektora mostowego oraz typy i kody lokalizowanych na obiekcie uszkodzeń (np. korozja stali konstrukcyjnej: KS – S-KS). Różnice między dokumentami polegają w większości na zasadach zachowania bezpieczeństwa ze względu na specyfikę ruchu na obiektach o różnym przeznaczeniu. Z tego powodu w rozprawie skupiono się na opisaniu specyfiki przeglądu technicznego w oparciu o chronologicznie pierwszą instrukcję GDDKiA do przeglądów drogowych obiektów inżynierskich z pominięciem różnic w zasadach zachowania bezpieczeństwa na obiekcie, które wykraczają poza dziedzinę pracy dyplomowej. Pozostałe różnice w instrukcjach (jak

na przykład nazwy rejonów organizacyjnych lub stanowisk w jednostkach zarządzających infrastrukturą) należy traktować analogicznie. Decyzja ta została podyktowana także większą dostępnością obiektów drogowych tak w przypadku materiałów z przeprowadzonych inspekcji oraz fizycznej dostępności do obiektów, które mogły służyć jako testowe w rozprawie.

Analogicznie do powyższego, pomimo że GDDKiA odpowiada za utrzymanie wyłącznie dróg (i znajdujących się w ich ciągu obiektów inżynierskich) krajowych, ekspresowych oraz autostrad, wykonana na zlecenie agencji instrukcja weszła do powszechnego użytku także w innych zarządach. Jednostki rozporządzające drogami wojewódzkimi i miejskimi (Zarządy Dróg Wojewódzkich – ZDW, Zarządy Dróg Miejskich – ZDM) najczęściej przyjmują instrukcję GDDKiA jako obowiązującą na drogach znajdujących się pod ich zarządem na podstawie lokalnego zarządzenia (np. zarządzenie Dyrektora ZDW w Gdańsku [116]).

Przeglądy okresowe drogowych obiektów mostowych w Polsce dzielą się na trzy typy zróżnicowane ze względy na częstotliwość i zakres przeglądu. Pierwszym z nich jest przegląd podstawowy wykonywany nie rzadziej niż raz w roku. Jego celem jest określenie aktualnego stanu technicznego tak samego obiektu jak również jego otoczenia i instalacji na obiekcie. Jest wykonywany poprzez wizualne oględziny obiektu nieuzbrojonym okiem oraz podstawowe pomiary. Efektem przeglądu jest nadanie obiektowi oceny punktowej.

Drugim typem jest przegląd rozszerzony, wykonywany nie rzadziej niż raz na pięć lal (lub trzy lata dla obiektów o dużym natężeniu ruchu), którego celem oprócz określenia stanu technicznego obiektu jest także ocena jego stanu wizualnego. Podobnie jak przegląd podstawowy wykonywany jest poprzez oględziny wizualne, jednak z odległości nieprzekraczającej jednego metra. W jego trakcie sporządzana jest także dokładna dokumentacja fotograficzna obiektu.

Ostatnim typem jest przegląd szczegółowy, wykonywany po wytypowaniu w trakcie innego przeglądu, lecz nie rzadziej niż raz na pięć lat. Jego głównym celem jest wykazanie celowości poprawy cech użytkowych w ramach przebudowy obiektu oraz określenie konieczności przeprowadzenia dodatkowej ekspertyzy. W odróżnieniu od pozostałych typów przeglądów, w jego trakcie oprócz oględzin wizualnych obiektu, przeprowadza się także manualne badania mechaniczne, pomiarowe i specjalistyczne.

W kolejnych sekcjach rozdziału opisane zostaną wyposażenie terenowego inspektora mostowego oraz zasady stosowania ocen punktowych w przeglądach podstawowym, rozszerzonym i szczegółowym. W rozprawie pominięte zostały przeglądy bieżące, gdyż wykonywane są przez drogomistrza – pracownika technicznego rejonu GDDKiA, nie będącego inspektorem mostowym. Nie uwzględniona została także ekspertyza, gdyż wykracza ona poza instrukcję GDDKiA przeprowadzania przeglądów.

2.2.3.1. WYPOSAŻENIE TERENOWEGO INSPEKTORA MOSTOWEGO

Podstawowe wyposażenie terenowego inspektora mostowego jest wspólne dla przeglądów podstawowego, rozszerzonego oraz szczegółowego. Składają się na nie przyrządy takie jak dalmierz laserowy, szkicownik, aparat cyfrowy, suwmiarka, czy nawet dłuto i wiertarka. Narzędzia te zostały wybrane tak, aby umożliwić TIM przeprowadzenie badań polowych zgodnie z instrukcją.

Tym samym, wyposażenie pozwalające na wykonanie przeglądu podstawowego składa się z wielu przyrządów, z których większość spełnia pojedynczą funkcję. Poza samą dokuczliwą ilością przenoszonego wyposażenia, po uwzględnieniu trudnych warunków pracy inspektora (patrz Rys. 1), część przyrządów może okazać się nieporęczna lub nawet niemożliwa do efektywnego zastosowania. Ekwipunek jednocześnie nie był aktualizowany o nowoczesne urządzenia nie wymagające uciążliwej, pracy manualnej inspektora, które byłyby istotnie ułatwiającym jego pracę dodatkiem.

2.2.3.2. ZASADY STOSOWANIA SKALI OCEN PUNKTOWYCH

W ocenie stanu technicznego obiektu wykorzystuje się pojęcie punktowej skali oceny elementu obiektu mostowego. Skala ta została przedstawiona w instrukcji w sposób tabelaryczny przez podanie wyłącznie podatnego na subiektywną interpretację, lingwistycznego opisu stopnia uszkodzenia. Skala i kryteria oceny elementów obiektu mostowego została przedstawiona poniżej, w Tab. 1.

Tab. 1: Skala i kryteria oceny elementów konstrukcyjnych wg. instrukcji GDDKiA

Ocena	Stan	Opis stanu elementu
5	odpowiedni	bez uszkodzeń i zanieczyszczeń możliwych do stwierdzenia podczas przeglądu
4	zadowalający	wykazuje zanieczyszczenia lub pierwsze objawy uszkodzeń pogarszających wygląd estetyczny
3	niepokojący	wykazuje uszkodzenia, których nienaprawienie spowoduje skrócenie okresu bezpiecznej eksploatacji
2	niedostateczny	wykazuje uszkodzenia obniżajcie przydatność użytkową, ale możliwe do naprawy
1	przedawaryjny	wykazuje nieodwracalne uszkodzenia dyskwalifikujące przydatność użytkową
0	awaryjny	uległ zniszczeniu lub przestał istnieć

Jak widać w Tab. 1, przedstawiony opis lingwistyczny stanu elementu pozwala na pewną dowolność w jego ocenie. Dowolność ta widoczna jest szczególności podczas wyłącznie wizualnych badań obiektu bez użycia aparatury pomiarowej pozwalającej na stwierdzenie faktycznego stopnia zagrożenia wynikającego z zaobserwowanego uszkodzenia. Skala ta nie oddaje również różnic między poszczególnymi elementami obiektu (oprócz rozróżnienia między elementy konstrukcyjne a izolację ocenianą w skali trójstopniowej). Pozwala także na nadawanie wysoce subiektywnych ocen, które mogą nie być powtarzalne między inspekcjami różnych inspektorów. Z uwagi na to, utworzony został

suplement do instrukcji przeprowadzania przeglądów, zawierający dokładne wytyczne co do stosowania ocen punktowych.

Suplement ten, pod nazwą „Zasady stosowania skali ocen punktowych stanu technicznego i przydatności do użytkowania drogowych obiektów inżynierskich" [8] opublikowany został już w trzy lata po publikacji instrukcji GDDKiA i został przewidziany do stosowania w czasie wykonywania przeglądów podstawowych oraz rozszerzonych.

Składa się on z dwóch głównych części. Pierwsza z nich przedstawia zasady stosowania ocen punktowych dla elementów konstrukcyjnych oraz izolacji różnych drogowych obiektów inżynierskich, natomiast druga opisuje określanie przydatności obiektu do użytkowania.

W pierwszej części, obiekt inżynierski został podzielony na jego elementy składowe, analogicznie do podziału występującego w raporcie z kontroli rocznej i pięcioletniej mostu. Następnie dla każdego z wyszczególnionych elementów podane zostały możliwe rodzaje defektów wraz z odpowiadającą im oceną dla danego zaobserwowanego zakresu procentowego – w przypadku gdy na elemencie występuje wiele uszkodzeń jednocześnie, końcową oceną elementu jest minimalna ocena wynikająca z najpoważniejszego z nich. Podane zostały także kody opisanych uszkodzeń. Fragment tabeli opisującej możliwe uszkodzenia dla dźwigarów stalowych wraz z kryterium nadania oceny przedstawia Tab. 2. Oprócz tabelarycznego zestawienia uszkodzeń, w dokumencie zawarto także uwagi dotyczące szczególnych przypadków defektów mogących występować na elemencie oraz ilustrowane przykłady, opisane odpowiednimi kodami, wraz z nadaną oceną. Przykładową ilustrację uszkodzenia węzła stalowego dźwigara kratownicowego przedstawia Rys. 3.

Tab. 2: Przykładowa tabela oceny elementu konstrukcyjnego (dźwigarów stalowych)

Lp.	**Rodzaj uszkodzeń**	**Zakres uszkodzeń [%]**					**Przykładowe kody uszkodzeń**
		0	≤5	10	20	≥30	
1	Zanieczyszczenia, wegetacja roślin	5	4	3	3	2	NS, WS
2	Zniszczona powłoka antykorozyjna	5	4	3			AS
3	Korozja dźwigarów	5	4	3	2	1	KS
4	Rysy i pęknięcia	5	1				RS

Druga część suplementu opisuje wytyczne do oceny przydatności obiektu do użytkowania, rozdzielone na poszczególne drogowe obiekty inżynierskie. Analogicznie jak w przypadku nadawania ocen punktowych elementom konstrukcji, w tej części suplementu tabelarycznie zestawione zostały przypadki nieprawidłowości zagrażających bezpieczeństwu użytkowania obiektu z odpowiadającymi im ocenami punktowym. Ponadto, także w tej części suplementu zamieszczone zostały ilustrowane przykłady nieprawidłowości ograniczających przydatność obiektu do użytkowania wraz z ich opisem oraz oceną punktową. Przykład ilustracji obrazującej zagrożenie bezpieczeństwa ruchu publicznego pieszych przedstawia Rys. 4.

Korozja węzła dźwigara kratownicowego. Osłabienie spowodowane ubytkami korozyjnymi oszacowano na 10%

Kod uszkodzenia: AS, KS

Ocena: 2

Rys. 3: Przykładowa ilustracja uszkodzenia węzła dźwigara kratownicy [8]

Jak wynika z zaprezentowanego opisu i przykładów, suplement do instrukcji przeprowadzania przeglądów drogowych obiektów inżynierskich w dużym stopniu wspomaga pracę i ujednolica wyniki nadawane drogowym obiektom inżynierskim przez terenowych inspektorów mostowych. Tym niemniej, także on operuje wyłącznie na podatnej na błąd, wizualnej ocenie rozległości i intensywności uszkodzenia. Jest to tym samym przesłanka, iż obowiązujący aktualnie system oceny infrastruktury mostowej nie jest idealny i wprowadzenie do przeglądów metod nie wymagających subiektywnej oceny elementów na podstawie głównie obserwacji wizualnej, zwiększy wydajność przeprowadzanych przeglądów, ujednolicając jednocześnie nadawane obiektom oceny. Nie bez znaczenia pozostaje też sposób kolekcjonowania danych o obiekcie, który aktualnie opiera się na manualnym wpisywaniu danych do gotowych formularzy raportów, a następnie przechowywanie ich w formie papierowej lub ręczną digitalizację podatną na utratę danych.

Uszkodzenie pokrywy kanału kablowego stwarzające zagrożenie dla pieszych

Ocena przydatności do użytkowania:

- bezpieczeństwo ruchu publicznego: 0
- szerokość skrajni na obiekcie: 5

Rys. 4: Przykładowa ilustracja zagrożenia bezpieczeństwa ruchu pieszego [8]

2.2.3.3. Systemy wspomagające gospodarkę mostową w Polsce

Odpowiedzią na część z sygnalizowanych w poprzedniej sekcji problemów, są systemy informatyczne wspomagające zarządzanie gospodarką mostową. Występują one jako systemy centralne, zarządzające bazą wiedzy o całej infrastrukturze należącej do danej jednostki organizacyjnej oraz systemy lokalne wykorzystywane na danym obszarze lub do zarządzania pojedynczymi

obiektami. Sekcja ta opisuje niektóre z tych systemów stosowane aktualnie lub w minionych latach w Polsce.

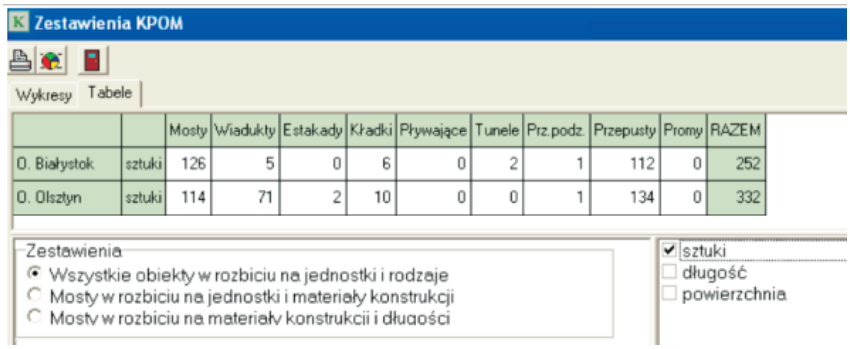
Programem wykorzystywanym przez GDDKiA jest SGM 2000 [117]. Został on pierwotnie opracowany w 1992 roku, natomiast wdrożony do powszechnego użytku rok później. Służy do zarządzania, gromadzenia i ewidencjonowania danych o drogowych obiektach inżynierskich. Składa się z kilku modułów, które odpowiadają kolejno za gromadzenie danych ewidencyjnych i generowanie dokumentów obiektu inżynierskiego, nadawanie, gromadzenie i edycję ocen z przeglądów stanu technicznego konstrukcji oraz generowanie ich wykazów. Pozostałe funkcjonalności systemu to możliwość generowania raportów oraz przeglądania bazy danych SQL systemu. SGM jest jednak systemem wyraźnie przestarzałym, którego ostatnia aktualizacja dostosowywała go do zarządzenia GDDKiA z 2008 roku. Nie powstał jeszcze uwspółcześniony odpowiednik tego systemu dla zastosowań centralnych. Fragment interfejsu systemu przedstawia Rys. 5. Widać, że odstaje on od standardu najnowszych aplikacji, a przechowywane dane są reprezentowane i wprowadzane w mało czytelny, tabelaryczny sposób. Może to prowadzić do trudnych do odnalezienia pomyłek.

Analogicznym systemem centralnym, wykorzystywanym do zarządzania siecią kolejowych obiektów inżynierskich jest System Zarządzania mostami Kolejowymi SMOK [118]. Został on opracowany i wdrożony na terenie całej Polski w roku 1997. Umożliwia ewidencję kolejowych obiektów inżynieryjnych, a także pozwalał na gromadzenie i przetwarzanie danych o ich uszkodzeniach. W czasie powstania, było to narzędzie innowacyjne, gdyż jako jedno z pierwszych na świecie wykorzystywało technologię sieci hybrydowych w implementowanych funkcjach ekspertowych. Funkcje te pozwalały na generowanie oceny stanu technicznego obiektu na podstawie wprowadzonych uszkodzeń. Niestety podobnie jak w przypadku systemu SGM, jego rozwój został porzucony, lecz znacznie wcześniej, bo po ostatniej aktualizacji w 2000 roku.

Przykładem systemu stosowanego przez administracje lokalne (to znaczy nie będącego w użytku do obsługi krajowej bazy obiektów mostowych) jest np. System Zarządzania Obiektami Komunikacyjnymi SZOK [119]. System ten utworzony pod koniec lat dziewięćdziesiątych przez autorów wspomnianego wcześniej systemu SMOK, wykorzystywany był przez niektóre jednostki zarządzające drogami powiatowymi, gminnymi i miejskimi. Oferował on podobne funkcjonalności co jego odpowiednik dla konstrukcji kolejowych.

Ostatnią grupą systemów wspomagających gospodarkę mostową są systemy zaprojektowane do obsługi pojedynczych obiektów. W odróżnieniu od wspomnianych w rozprawie wcześniej systemów, akumulują one dane nie tylko z zabudowanych w konstrukcji sensorów, ale także z przeglądów i bieżących raportów technicznych. Są one najczęściej obsługiwane przez zarządcę obiektu (np. GDDKiA) i obejmują mosty szczególnie istotne z punktu widzenia sieci komunikacji miejskiej. Przykładem takiego systemu może być Rubikon [120], opracowany dla mostu

autostradowego im. Armii Krajowej w Toruniu w ciągu autostrady A1. Zawiera on model cyfrowy obiektu wykorzystywany do nawigacji, opisywania uszkodzeń oraz lokalizowania mierników używanych w trakcie przeglądów obiektu.



Rys. 5: Fragment interfejsu modułu ewidencji programu SGM

Wymienione systemy, mimo usprawniania procesu zarządzania infrastrukturą mostową, jedynie w niewielkim stopniu wspomagały pracę terenowego inspektora mostowego, a ich rozwój został wstrzymany ponad dekadę temu. Tymczasem rozwój technologii mobilnych, fotografii cyfrowej oraz algorytmów wykorzystujących algorytmy sztucznej inteligencji daje potencjał do opracowania metod mogących wydatnie poprawić dokładność i wydajność procesu zarządzania infrastrukturą już na etapie podstawowego i rozszerzonego przeglądu obiektu, a także procesu zarządzania pojedynczymi uszkodzeniami.

2.2.3.4. PODSUMOWANIE

W sekcji tej przedstawione zostały metody przeprowadzania inspekcji obiektów mostowych w Polsce oraz wybranych krajach na świecie. W opisanych procedurach doszukać się można wielu podobieństw. Najważniejszym z nich jest wprowadzenie konieczności przeprowadzania przeglądów obiektów inżynierskich na poziomie rządowym oraz utworzenie krajowych wytycznych obejmujących główne drogi w państwie, lecz adaptowane także do dróg lokalnych. W większości z wymienionych krajów (wyjątkiem jest RPA) przeglądy przeprowadzane są przez agencje rządowe zarządzające infrastrukturą. Podobieństwa widać także w podziale rodzajów kontroli okresowych na szczeble różniące się od siebie pracochłonnością, interwałami ich przeprowadzania czy zbieraną dokumentacją lub w powszechnym w większości krajów systemie ocen punktowych nadawanych elementom obiektów, bazującym na zidentyfikowanym stopniu uszkodzenia. Kolejnym istotnym podobieństwem jest powszechność stosowania prostych metod wizualnych w trakcie przeglądów. Wszystkie z wymienionych systemów korzystają z oceny wizualnej, a niektóre z nich (na przykład w Japonii) do oceny elementów obiektu w ogóle nie wykorzystują żadnej aparatury pomiarowej.

Porównywalne są także sposoby akumulacji danych w formie zdjęć dołączanych do raportów z przeglądów oraz sposób przechowywania dokumentów. W większości z przytoczonych przykładów

systemów zarządzania infrastrukturą, generowane raporty przechowywane są na dwa sposoby – w formie elektronicznej oraz tradycyjnej, papierowej. Jednocześnie dane w postaci elektronicznej akumulowane są przez często wysłużone systemy informatyczne, których geneza sięga wczesnych lat dziewięćdziesiątych. Ostatnim istotnym podobieństwem jest kładzenie coraz większego nacisku na działania prewencyjne (szczególnie widoczne w wytycznych FHWA), które przy sprawnym monitorowaniu stanu technicznego konstrukcji mogą ograniczyć całkowite wydatki na utrzymanie infrastruktury.

Liczne podobieństwa między systemami wytłumaczyć można chronologią ich rozwoju. Rozwinięte systemy powstające w Europie i USA później, po niewielkich modyfikacjach migrowały dalej, między innymi do Australii i Brazylii, zachowując charakterystyczne cechy pierwotnych systemów.

Tym samym, metody wykorzystywane w innych krajach na świecie, porównać można do polskich standardów utrzymania infrastruktury mostowej. Dostrzec można bazowe elementy wspólne dla wszystkich systemów, określające przydatność metod stosowanych w praktyce. Elementami tymi jest **łatwość stosowania, szybkość i brak konieczności wykorzystywania rozbudowanych narzędzi specjalistycznych** w trakcie podstawowo przeprowadzanych przeglądów, co wpływa na ich wydajność i niski koszt. Mając zatem na celu poprawę skuteczności i wydajności pracy inspektora mostowego, należy wziąć pod uwagę te trzy podstawowe cechy. Musi je spełniać narzędzie inżynierskie, aby mogło zostać uznane za użyteczne w zagadnieniu przeprowadzania przeglądów obiektów mostowych.

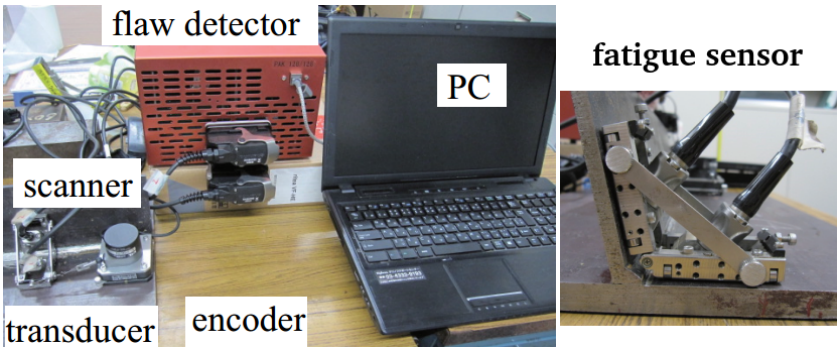
2.3. NOWOCZESNE METODY W ZARZĄDZANIU INFRASTRUKTURĄ MOSTOWĄ

Podejmując tematykę nowoczesnych metod nad którymi prowadzone są obecnie badania lub wdrażane są w trybie pilotażowym do wspomagania zarządzania infrastrukturą mostową, należy dokonać ich podziału na metody mające wspomagać przeprowadzanie przeglądów obiektów mostowych oraz te, które mają zostać wykorzystane w kolejnych etapach zarządzania infrastrukturą. Podział ten wykazuje zdecydowaną przewagę metod z tej drugiej grupy – większość obecnie prowadzonych prac skupia się na rozwiązaniach, które nie spełniają niektórych lub wszystkich trzech podstawowych założeń użytecznego narzędzia inspektora mostowego wymienionych w poprzedniej sekcji.

Na bazie zarysowanego powyżej podziału przeprowadzony został syntetyczny przegląd obecnie prowadzonych prac, a ich efekty zostały zakwalifikowane jako metody nie dotyczące zagadnienia inspekcji, pośrednio wspomagające kontrolę obiektu lub użyteczne z punktu widzenia pracy inspektora mostowego.

Do pierwszej grupy zostały zakwalifikowane metody, które przez swoją pracochłonność, konieczność stosowania sprzętu specjalistycznego oraz brak możliwości zastosowania doraźnego w procesie inspekcji nie spełniają wymagań użytecznego narzędzia inspektora mostowego, natomiast z powodzeniem mogą być wykorzystywane w trakcie ekspertyz. Zaliczyć można do nich instalowanie rozbudowanych układów czujników badających np. wpływy zmęczenia na spoinę pachwinową (Rys. 6). Wykorzystywane są także kosztowne i często delikatne urządzenia, takie jak skanery laserowe do poszukiwania drobnych uszkodzeń powierzchni betonowych [121] czy kamery na podczerwień w pomiarach wewnętrznych uszkodzeń elementów betonowych [122]. Jednak warunki użycia tych urządzeń sprzyjają szybkiemu ich uszkodzeniu.

W drugiej grupie metod umieścić można wszystkie prace opisujące metody zarządzania danymi z inspekcji oraz metody monitoringu ciągłego konstrukcji. Mimo że nie wspomagają one bezpośrednio w pracy IM, to dzięki ich zastosowaniu można w sposób wydatny podnieść jej efektywność. Najnowsze prace, oprócz korzystania z systemów monitorowania korozji stali konstrukcyjnej, montowanych na stałe na obiekcie [123], wykorzystują nowsze metody, takie jak pomiary wykonywane przez poruszające się po obiekcie pojazdy transportu publicznego [124,125]. Innym sposobem pomiaru, który może sprawdzić się tak podczas ciągłego monitoringu konstrukcji jak i w trakcie przeglądu obiektu mostowego, jest zastosowanie czujników Bluetooth do pomiaru wilgotności (szczelności) elementów betonowych. Mogą być one pozostawione na obiekcie na stałe, lub stać się częścią wyposażenia IM [126]. Prace opisujące metody zarządzania danymi skupiają się na wykorzystaniu technik monitoringu ciągłego w połączeniu z systemami SHM [127], systemów BMS integrujących w sobie aplikacje sieciowe, mobilne, rozproszone bazy danych oraz bieżące pomiary sensorów wbudowanych w obiekt [128], a także przedstawieniu możliwości technologicznych połączenia BIM, poszerzonej rzeczywistości i przechowywania danych w chmurze w kontekście przemysłu 4.0 [129]. Innym kierunkiem rozwoju metod akumulacji danych o obiekcie jest projekt SeeBridge, skupiający się na semantycznym wzbogacaniu modeli BIM obiektów mostowych [130,131].



Rys. 6: Przykład czujnika wpływu zmęczenia [246]

Na ostatnią grupę składają się metody, których użycie w trakcie przeprowadzania inspekcji może wydatnie wpłynąć na efektywność pracy IM. Zaliczyć można do nich całą gamę rozwiązań wdrażających algorytmy wizji komputerowej [132]. Najnowsze osiągnięcia w tej dziedzinie, wykorzystane w zagadnieniach utrzymania infrastruktury skupiają się na identyfikowaniu defektów powierzchniowych przy zastosowaniu metod tradycyjnej wizji komputerowej. Stosowane są między innymi metody binaryzacji obrazu [133], rozdzielające przestrzeń barw [41] oraz różne typy algorytmów do detekcji krawędzi [134–137]. Powstały także metody starające się szacować wielkość uszkodzenia na podstawie obrazu cyfrowego [138,139]. Oprócz identyfikacji uszkodzeń, metody wizyjne znajdują zastosowanie także w szacowaniu struktury ruchu pojazdów na obiekcie (będące częścią przeglądu szczegółowego) [140] oraz w połączeniu z modelami BIM – do monitorowania deformacji elementów konstrukcji [141]. Z technikami wizji komputerowej, do przeglądów infrastruktury używane są także drony [142], które w niektórych krajach już znalazły zastosowanie są między innymi w inspekcjach dróg i kopali odkrywkowych [143].

Do metod wizualnych zaliczyć można także rozwiązania wykorzystujące widzenie komputerowe z zastosowaniem wizji stereoskopowej, lecz obecnie prowadzone prace pozostają na wczesnym etapie rozwoju i stosowane są w nich głównie z ręcznie synchronizowane kamery konwencjonalne [144], w statycznych próbach obciążenia konstrukcji.

Osobną kategorię stanowią metody obliczeniowe, gdyż nie da się ich w prosty sposób opisać kryteriami charakteryzującymi narzędzie użyteczne z punktu widzenia IM, a same w sobie nie stanowią narzędzia opracowanego z myślą o wyłącznie jednym zastosowaniu. Są wykorzystywane na każdym z etapów zarządzania infrastrukturą, od oceny raportów z inspekcji wizualnych [145] i predykcji odpowiedzi konstrukcji na podstawie danych z monitoringu ciągłego [146], aż po probabilistyczną ocenę długości cyklu życia obiektu w zarządzaniu bazą infrastruktury [147]. Na szczególną uwagę zasługują relatywnie nowe w dziedzinie budownictwa metody stosujące algorytmy sztucznej inteligencji. Po okresie gwałtownego wzrostu zainteresowania nimi w innych gałęziach przemysłu, zaczęły być wprowadzane do inżynierii budowlanej tak od strony projektowej [148–150], jak i w kontekście utrzymania infrastruktury [151]. Jednocześnie, w opisywanej dyscyplinie jest to dział metod obliczeniowych cieszący się obecnie największym wzrostem zainteresowania.

2.4. PODSUMOWANIE PRZEGLĄDU LITERATURY W KONTEKŚCIE ROZPRAWY

Przegląd literatury dostarczył informacji na temat aktualnej praktyki w przeprowadzaniu inspekcji obiektów mostowych, a także utrzymania infrastruktury w Polsce i na świecie. Krytycznej ocenie poddana została także najnowsza literatura naukowa w dziedzinie opisująca aktualny stan wiedzy tak pod kątem jej dostępności jak i ilości cytowań. Następnie, literatura została podzielona ze względu na miejsce pochodzenia oraz konkretne zagadnienie, którego dotyczy. Wskazane zostały także obecne kierunki rozwoju w dziedzinie, odniesione do praktyki pracy inspektora mostowego.

Rysuje się wyraźna różnica między aktualną praktyką przeprowadzania przeglądów infrastruktury, a nowoczesnymi metodami jej badań. Różnica ta po części daje się wyjaśnić specyfiką pracy i wymaganiami stawianymi przed IM, które nakładają na niego konieczność szybkiej pracy, częściowo kosztem jej dokładności. Nie bez znaczenia są także koszty wdrożenia nowoczesnych metod, dla których cena pojedynczego badania może przekraczać koszty całego wyposażenia IM. Widać więc, że agencjom zajmującym się utrzymaniem infrastruktury drogowej na świecie, zależy na optymalnym stosunku kosztów przeglądu do szybkości jego wykonania i dokładności.

Na podstawie powyższych konkluzji oraz przeprowadzonego przeglądu literatury wyciągnięte zostały wnioski dotyczące możliwych kierunków rozwoju narzędzi wspomagania pracy IM, a także określone zostały główne wymagania, jakie muszą one spełniać. Jednocześnie, dzięki analizie aktualnych trendów w pracach naukowych i piśmiennictwie, określone zostały najbardziej racjonalne kierunki dalszego rozwoju prac nad tego typu narzędziem, wśród których zaobserwowana została coraz większa dominacja metod opartych na wizji komputerowej. Nie bez znaczenia jest także obserwowany w ostatnich latach, raptowny wzrost zainteresowania metodami sztucznej inteligencji w dziedzinie.

Pierwszy z kierunków który został wybrany jako możliwość rozwoju techniki wizyjnej utrzymania stanu technicznego konstrukcji, zakładał wykorzystanie do monitorowania kamer hiperspektralnych. Dzięki nim możliwe jest wielospektralne obrazowanie elementów obiektu, co pozwala uwidocznić na obrazie wczesne uszkodzenia (np. zanieczyszczenia chlorkami, słabe wykwity) niewidoczne jeszcze gołym okiem. Mimo że technologia ta przeżywa dynamiczny rozwój i już znalazła zastosowanie między innymi w medycynie, rolnictwie, chemii, muzealnictwie oraz kryminalistyce, to narzędzia potrzebne do jej zastosowania nie spełniają założeń opisanych w sekcji 2.2.3.4. Kamery hiperspektralne są urządzeniami kosztownymi, nieporęcznymi (Rys. 7) a ich wykorzystanie wymaga dużego nakładu czasu w porównaniu do tradycyjnych aparatów. Ponadto, nie są obecnie powszechnym narzędziem w dziedzinie inżynierii budowlanej.

Drugim kierunkiem było zastosowanie konwencjonalnego obrazowania z wykorzystaniem aparatu cyfrowego, wzbogaconego o metody wizji komputerowej, które w sposób algorytmiczny wspomagałyby identyfikację uszkodzeń. Metody te jednak odznaczają się dużą niedokładnością, a także podatnością na zmiany oświetlenia, które dyskwalifikują je z użycia w trakcie przeprowadzania inspekcji. Metody te nie pozwalają także na dostosowanie się do uszkodzeń o innych parametrach niż wstępnie przewidziane, to znaczy algorytmy te nie są w stanie generalizować swoich wskazań poza ich pierwotny model.

Ostatecznie, aby zaproponowane w rozprawie narzędzie było jak najbardziej wszechstronne i odporne na zmienne warunki oświetlenia oraz specyfikę pracy konkretnego IM, a także spełniało wszystkie wymienione wcześniej warunki użyteczności, wybrane zostało połączenie metod wizji

komputerowej oraz algorytmów sztucznej inteligencji. Połączenie to może przysłużyć się rozwojowi metod utrzymania infrastruktury mostowej w kierunku większej automatyzacji przeglądów, wpływającej na zwiększenie dokładności i wydajności pracy inspektora mostowego oraz zmniejszenie pracochłonności dokumentowania przeglądu. Może także wpłynąć na popularyzację metod SI w dziedzinach, w których do tej pory nie były wykorzystywane. Wpisuje się to jednocześnie w obecny trend intensywnego wzrostu wykorzystania SI w inżynierii budowlanej.

Rys. 7: Kamera hiperspektralna na statywie [244]

3. SZTUCZNA INTELIGENCJA – GENEZA, ROZWÓJ I BUDOWA ALGORYTMÓW

Przed przystąpieniem do opisu konkretnych algorytmów wchodzących w skład dziedziny sztucznej inteligencji, istotne jest usystematyzowanie dotyczących jej pojęć. W tym kontekście istotne jest zdefiniowanie pojęć inteligencji i sztucznej inteligencji na pograniczu pól psychologii i nauk technicznych. Definicję tę należy skonfrontować z ewolucją SI począwszy od jej pierwszych koncepcji, a skończywszy na czasach najnowszych.

Dopiero po zapoznaniu się z tematyką SI, można przejść do opisu jej konkretnych algorytmów oraz przykładów ich zastosowania w inżynierii budowlanej, także w kontekście inspekcji.

3.1. INTELIGENCJA A SZTUCZNA INTELIGENCJA

Wywodzące się z łaciny słowo „*intellego*", od którego pochodzi jego uwspółcześniony odpowiednik – „inteligencja", oznacza „rozumiem" lub „dostrzegam", a jego dosłowne tłumaczenie, powiązane z oryginalną etymologią wywodzącą się z połączenia słów „*inter*" oraz „*legō*" znaczy „czytam -" lub „wybieram między". Termin ten pierwotnie jednak nie był bliski jego dzisiejszemu znaczeniu i długo pozostawał ściśle powiązany z naukami teologicznymi, a później filozofią. Dopiero w XIX wieku, za sprawą prac na gruncie psychologii autorstwa Johna Stuarta Milla i Alfreda Bineta został adaptowany do nauk psychologicznych, a jego znaczenie zaczęło zbliżać się do obecnego [152].

Tym niemniej, nawet po ponad stu latach od wprowadzenia pojęcia inteligencji do nauk psychologicznych, jej ogólna definicja pozostaje kontrowersyjna, a środowisko naukowe zajmujące się nią pozostaje podzielone. Do najszerzej przyjętych obecnie definicji inteligencji należą:

- ogólna zdolność umysłowa obejmująca między innymi umiejętność rozumowania, planowania, rozwiązywania problemów, myślenie abstrakcyjne i kompleksowe, szybkie uczenie i uczenie z doświadczenia [153],
- zachowanie adaptacyjne nakierowane na cel [152],
- wyjątkowa zdolność **człowieka** do zmiany lub modyfikacji swojego funkcjonowania poznawczego w celu dostosowania się do zmieniających się wymagań sytuacji [154].

Nie są to definicje powszechne dla całego środowiska naukowego. Wśród innych, nie uwzględnionych w nich aspektów zaznacza się między innymi możliwość istnienia innych,

niemierzalnych psychometrycznie typów inteligencji (systemów zdolności), konieczność uwzględnienia w definicji wpływów kulturowych i procesów nauczania, a także nie postrzeganie inteligencji jako cechy ekskluzywnej dla gatunku ludzkiego [155,156].

W odróżnieniu od ogólnej definicji inteligencji wobec której na płaszczyźnie naukowej nadal toczone są spory, jej odniesienie do cechy ludzkiej jest dalece mniej kontrowersyjne. Jej szczególną definicję wykorzystywaną w psychologii nauczania opisać można jako ludzką zdolność umysłową charakteryzującą się złożoną zdolnością poznawczą oraz wysokim poziomem motywacji i samoświadomości [157]. Zdolność ta może zostać opisana w sposób liczbowy przez szereg wyznaczników, spośród których wymienić można najbardziej powszechny – iloraz inteligencji IQ, oraz inne wyznaczniki psychometryczne, takie jak *współczynnik g* (ang. general intelligence factor) czy *c* (ang. collective intelligence factor) dla inteligencji zbiorowej [158].

Przechodząc z gruntu psychologii i inteligencji obserwowanej i mierzonej u ludzi na nauki techniczne i obecnie opracowywane algorytmy sztucznej inteligencji, zauważyć można że część z przytoczonych wcześniej definicji oraz metryk nie znajduje zastosowania w ich opisie. Bezpośrednią tego przyczyną jest dzisiejszy stopień zaawansowania sztucznej inteligencji, która dotychczas wypełniała wyłącznie definicję „słabej" lub „wąskiej" SI. Oznacza to, że jej możliwości, w przeciwieństwie do „silnej" sztucznej inteligencji mającej w założeniu posiadać zdolność rozumienia i nauczenia się dowolnego, złożonego zadania intelektualnego, ograniczają się do wykonywania pojedynczych zadań, takich jak rozpoznawanie mowy, tekstu lub obrazów, czy wyszukiwanie optymalnych rozwiązań złożonych, ale należących do pojedynczego typu problemów [159]. Także w rozprawie, pod pojęciem metod sztucznej inteligencji, tak tych opisywanych w dalszych sekcjach jak i użytych w budowie narzędzia wspomagającego pracę IM, rozumiana będzie „wąska" sztuczna inteligencja.

W świetle powyższego, inteligencję w odniesieniu do obecnego rozwoju technologii komputerowych zdefiniować można za twórcami przedsiębiorstwa DeepMind, będącego pionierem w najnowszej historii rozwoju metod sztucznej inteligencji. Określili oni sztuczną inteligencję jako miarę zdolności do osiągania celów w szerokim spektrum środowisk [160]. Do jej pomiaru w odróżnieniu od testów ogólnych stosowanych w badaniach psychometrycznych u ludzi, wykorzystywane są metryki opisujące działanie konkretnego algorytmu dla wybranego zadania, takie jak dokładność, czułość i precyzja. Zostały one opisane i użyte w ocenie proponowanego w dalszej części rozprawy rozwiązania w rozdziale 4. oraz do porównania go z aktualnym stanem wiedzy w sekcji 5.1.2..

Obecnie, naukę o sztucznej inteligencji podzielić można na pola wyznaczające jej współczesne kierunki rozwoju. Pokrywają się one z dziedzinami, w których algorytmy SI są stosowane w sposób komercyjny. Należy do nich między innymi reprezentacja wiedzy, wykorzystywana w indeksowaniu danych i wspieraniu procesów decyzyjnych na przykład w

medycynie [161], automatyczne planowanie [162], uczenie maszynowe wraz z głębokim uczeniem wykorzystywane w wizji komputerowej i budowie algorytmów klasyfikujących oraz przetwarzanie języka naturalnego (ang. Natural Language Processing) [163] w systemach komputerowych.

Jednocześnie zestawiając ze sobą pojęcia inteligencji naturalnej oraz sztucznej, należy wspomnieć o problemach natury etycznej rodzących się razem z rozwojem SI. Mimo, że problematyka ta w połączeniu ze sztuczną inteligencją była obecna już od początków rozważań na jej temat, to związana z nią dyskusja znacznie przybrała na sile w przeciągu ostatnich dwóch dekad. Najczęściej wymienianymi problemami natury etycznej jest ryzyko masowej fali bezrobocia po upowszechnieniu się algorytmów SI w znacznej liczbie branż przemysłowych [164] oraz problem tworzenia bytów o inteligencji równej człowiekowi [165]. Ten ostatni jednak pozostanie w sferze rozważań teoretycznych aż do momentu opracowania „silnej" sztucznej inteligencji.

3.1.1. ZARYS HISTORYCZNY

Koncepcja inteligentnej maszyny lub sztucznej, rozumnej istoty sięga legend i mitów z czasów starożytnych. Nie była ona jednocześnie ekskluzywna dla pojedynczych cywilizacji – podobne rozważania pojawiały się na całym świecie, niezależnie od panującej kultury czy wyznania. W mitologii greckiej przykładem motywu inteligentnej maszyny może być mit o Talosie, uformowanym z brązu strażniku Krety [166]. Innymi przykładami rozważań na temat inteligentnych maszyn we wczesnych cywilizacjach są także egipskie ruchome posągi będące obiektami kultu lub wywodzący się z tradycji hebrajskiej Golem.

Były to jednak wyłącznie przejawy dywagacji zakorzenionych w religii lub filozofii, a faktyczne przykłady zbliżonych do przedstawianych w nich maszyn pojawiły się znacznie później, wraz z pierwszymi automatami (nazywanymi też z greki automatonami). Były to proste maszyny zasilane przy pomocy siły mięśni lub sprężyn napędowych, naśladujące w sposób mechaniczny podstawowe funkcjonowanie i zachowanie organizmów żywych. Pierwsze wzmianki o maszynach tego typu pochodzą już z cywilizacji starożytnych, lecz obecnie nie sposób dowieść prawdziwości ich cech opisywanych w ówczesnych pracach. Dobrze udokumentowane lub zachowane do teraz najwcześniejsze przykłady automatów pochodzą z okresu renesansu, a na ich pojawienie się wpływ miał rozwój technik budowy zegarów, w których często były zabudowywane. Najbardziej rozpoznawalne z nich to Kaczka Vaucansona, w sposób automatyczny przedstawiająca układ trawienny kaczki oraz Automat Maillardeta, nazywany również Młodym Artystą. Automaty te jednak, oprócz bycia świadectwem kunsztu rzemieślnika, nie posiadały praktycznych zastosowań [167].

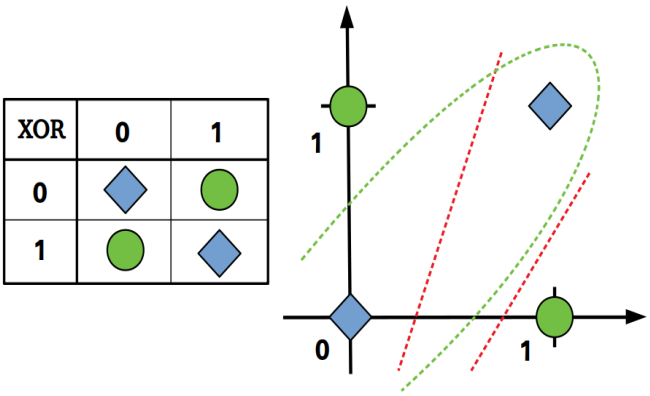
Podwaliny pod metody, które obecnie nazywane są sztuczną inteligencją oraz później budowę i rozwój komputerów zostały położone wraz z pracą matematyków, takich jak Gottfried Leibniz (opis logiki matematycznej), George Boole (logika algebraiczna) czy Alan Turing (informatyka teoretyczna). Ich prace, a także wielu nie wymienionych powyżej matematyków, oprócz opisu logiki

wykorzystanej później przy budowie i programowaniu pierwszych komputerów, zostały wykorzystane jako narzędzia do budowy algorytmów, które obecnie stosowane są w zagadnieniach obejmujących dziedzinę sztucznej inteligencji. Ta jednak, w swojej obecnej formie narodziła się dopiero na przełomie lat czterdziestych i pięćdziesiątych dwudziestego wieku.

3.1.2. SZTUCZNA INTELIGENCJA WSPÓŁCZEŚNIE

Rozwój współczesnych metod sztucznej inteligencji był ściśle powiązany z rozwojem dziedziny neurologii, która w latach czterdziestych i pięćdziesiątych ubiegłego wieku poczyniła olbrzymie postępy w zakresie poznania sposobu funkcjonowania mózgu. W okresie tym poznano między innymi budowę mózgu jako złożonego z sieci neuronów komunikujących się ze sobą za pośrednictwem impulsów elektrycznych. Odkryto także sposób powiązania obwodowego i ośrodkowego układu nerwowego, a także dzięki testom przeprowadzanym na zwierzętach, reprezentację bodźców zewnętrznych (wzrokowych, dotykowych itd.) jako impulsy generowane przez neurony w mózgu [168]. Odkrycia te zaczęto przenosić na grunt matematyki przez algorytmiczne odwzorowanie działania neuronów, co w 1943 roku poskutkowało utworzeniem matematycznego opisu działania neuronu [169], a w 1958 pierwszym prostym algorytmem neuronowym o nazwie Perceptron, zaimplementowanym maszynowo do rozpoznawania obrazów wielkości 20 na 20 pikseli [170]. Graf przedstawiający budowę Perceptrona w swojej podstawowej formie – złożonego z pojedynczego neuronu, przedstawia Rys. 10 w sekcji 3.3. Olbrzymi entuzjazm spowodowany nowymi odkryciami w tej dziedzinie skutkował optymistycznymi prognozami autorstwa czołowych naukowców w dziedzinie, których zdaniem pierwsze algorytmy „silnej" sztucznej inteligencji miały zostać utworzone jeszcze przed 1975 rokiem [171]. W tym okresie uformował się też powszechny dzisiaj termin „sztuczna inteligencja", użyty po raz pierwszy oficjalnie jako propozycja grupy tematycznej dla letnich projektów studenckich na Dartmouth College w 1955 roku [172].

Jednak mimo dynamicznego rozwoju dziedziny wspieranego przez jej bogate finansowanie (np. „Grupy SI" przez agencję DARPA w latach 1963-1974 [173]) opracowane w tym okresie algorytmy sztucznej inteligencji były w stanie rozwiązywać jedynie problemy trywialne i ich wykorzystanie ograniczało się głównie do prostych gier (na przykład w szachach i warcabach). Kolejnymi problemami które napotkali naukowcy była niewystarczająca moc obliczeniowa ówczesnych komputerów, brak istniejących baz danych służących do treningu algorytmów oraz możliwości ich zebrania i zapisu na dostępnych nośnikach danych, a także zbyt wąski zasób instrukcji wykonywalnych przez procesory [174]. Podobnie obiecujący algorytm Perceptron został wkrótce całkowicie porzucony z uwagi na brak możliwości rozwiązywania przez niego podstawowych problemów nieliniowych XOR (Rys. 8). Z tych powodów na ponad dekadę, nazywaną „pierwszą zimą SI", zahamowane zostało finansowanie dziedziny, a wraz z nim niemal całkowicie jej rozwój.



Rys. 8: Problem wykluczającego LUB (XOR) z błędnymi rozwiązaniami liniowymi (czerwony) i poprawnym nieliniowym (zielony)

Powrót algorytmów SI do jednostek naukowych nastąpił dopiero w 1980 razem z rozwojem techniki komputerowej i upowszechnieniem się systemów eksperckich. W odróżnieniu jednak od opracowanych wcześniej sztucznych neuronów, systemy te opierały się na symulowaniu procesu decyzyjnego eksperta w danej dziedzinie poprzez binarne reguły logiczne „jeżeli – to" połączone w łańcuchy decyzyjne, opracowane bezpośrednio na zebranych danych. Oznacza to, że nie podlegały one treningowi, a swoje działanie zawdzięczały zapisanej w nich bazie wiedzy [175]. Było to możliwe dzięki dynamicznemu rozwojowi metod utrwalania danych, dających możliwość tworzenia baz zawierających setki tysięcy rekordów. To natomiast skłoniło badaczy do rozpoczęcia prac nad systemem eksperckim „silnej" sztucznej inteligencji zawierającym bazę wiedzy przeciętnego człowieka [165].

Systemy eksperckie prostocie działania oraz tworzenia zawdzięczały swoją największą przewagę nad tworzonymi poprzednio algorytmami – były użyteczne w zastosowaniach praktycznych nie ograniczających się do prostych gier. Jednocześnie, w tym samym okresie narodziło się odwrotne podejście do sztucznej inteligencji. Popularyzacja dziedziny robotyki wpłynęła na rozpowszechnienie poglądu, jakoby motoryka urządzenia była wyznacznikiem SI, natomiast sama możliwość symulowania inteligencji była jedynie jej mniej ważną pochodną [176]. Ostatnim z osiągnięć lat osiemdziesiątych w dziedzinie rozwoju SI było opracowanie algorytmu propagacji wstecznej (ang. backpropagation algorithm), dzięki któremu możliwy stał się trening sztucznych neuronów połączonych ze sobą w sposób szeregowy. Odkrycie to zaowocowało rozbudowaniem możliwości algorytmów neuronowych o rozwiązywanie problemów nieliniowych, co poskutkowało ich sukcesem komercyjnym w rozpoznawaniu tekstu i mowy na początku lat dziewięćdziesiątych, jednak algorytmy te nadal nie były wystarczająco efektywne, aby móc je stosować w bardziej złożonych problemach.

Tym razem jednak przewrotnie to rozwój technologii komputerowej przyczynił się do nadejścia kolejnego okresu „zimy SI". Postęp w budowie komputerów osobistych, takich jak Apple II projektowanych do użytku domowego, drastycznie zmniejszył opłacalność produkcji maszyn

projektowanych z myślą o algorytmach SI. Wkrótce komputery osobiste możliwościami obliczeniowymi przerosły maszyny dedykowane sztucznej inteligencji, których produkcji ostatecznie zaprzestano. To natomiast po pewnym czasie poskutkowało upadkiem systemów eksperckich opartych o maszyny Lisp, których nie dało się już utrzymać ani aktualizować [177], a społeczność naukowa ponownie zwróciła się ku sieciom neuronowym.

Przyjęto, że okres ostatniej „zimy SI" upłynął w 1997 roku, kiedy Garry Kasparov przegrał rozgrywkę szachową z komputerem Deep Blue. Zbiegło się to jednocześnie w czasie z publikacją artykułu opisującego algorytm propagacji wstecznej o wielokrotnie większej efektywności niż te z których korzystano wcześniej [178] w 1998 roku. Dało to początek kolejnemu okresowi dynamicznego rozwoju dziedziny sztucznej inteligencji, który zaowocował powstaniem poddziedzin takich jak uczenie maszynowe oraz głębokie uczenie. Obecnie, dzięki coraz wydajniejszym podzespołom komputerowym, dziedzina sztucznej inteligencji rozwija się nadal, a kolejne pola jej wykorzystania pojawiają się niemalże z dnia na dzień.

Jednocześnie, ostatni długi okres intensywnych prac nad algorytmami SI ponownie rozbudził entuzjazm wśród badaczy, zwracając ich uwagę na horyzont budowy „silnej" sztucznej inteligencji. Po raz kolejny wśród naukowców z dziedziny zaczęły pojawiać się głosy, iż osiągnięcie algorytmów działających na poziomie złożoności równym człowiekowi czeka ludzkość za około dwadzieścia pięć lat [179], jednak historia pokazuje, że szacunki te do tej pory były zdecydowanie przesadzone.

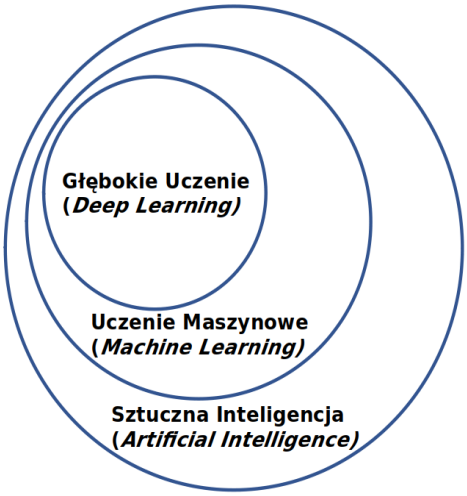
3.2. METODY SZTUCZNEJ INTELIGENCJI

Sztuczna inteligencja współcześnie obejmuje całą grupę metod i algorytmów, których celem jest wykonywanie przez maszynę operacji. W przypadku człowieka są one wykonywane intuicyjnie czy nawet niemal automatycznie. Tymczasem okazują się być wyzwaniem dla algorytmów komputerowych. Przykładem takich operacji jest rozpoznawanie zawartości obrazów, predykcja kolejnych wartości w szeregach czasowych czy tłumaczenie tekstu z jednego języka na drugi. Algorytmy te powstawały w różnych okresach i pierwotnie mogły nie być ze sobą powiązane w tej samej dziedzinie, lecz obecnie zostały uporządkowane w sposób hierarchiczny.

Ogólny podział metod sztucznej inteligencji można przedstawić jako zawierające się w sobie poddziedziny, począwszy od ogólnej dziedziny SI. Jej poddziedziną jest uczenie maszynowe, będące nauką o algorytmach i modelach statystycznych wykonujących pewne określone zadania bazując na metodach rozpoznawania wzorów oraz automatycznego wnioskowania. Do poprawnego działania takich algorytmów wymagany jest ich trening na zbiorach danych, zwanych zbiorami treningowymi. Algorytmów uczenia maszynowego nie tworzy się z myślą o wykonywaniu wyłącznie jednej, precyzyjnie określonej operacji, a pewnych działań ogólnych, których wynik uzależniony będzie od zbioru danych, na których są przeprowadzane. Przykładem jego wykorzystania może być zagadnienie klasyfikacji lub regresji logistycznej. Kolejnym poziomem podziału jest poddziedzina uczenia

maszynowego czyli głębokie uczenie. Zakłada ono hierarchiczną budowę algorytmu uczenia maszynowego, z którego wydzielić można płytkie oraz głębokie (ukryte) warstwy. Warstwy te cechują się tym, że oprócz ich znanej, założonej wstępnie architektury, działają jako czarna skrzynka i nie są interpretowalne wprost przez człowieka. Ich zawartość uzupełniana jest automatycznie, w iteracyjnym procesie trenowania algorytmu. Podobnie jak w przypadku uczenia maszynowego, algorytmy te mogą wykonywać operacje klasyfikacji i regresji, lecz ich zastosowanie pozwala na rozwiązanie o wiele bardziej złożonych problemów [180].

Zależności między kolejnymi poziomami algorytmów sztucznej inteligencji przedstawiono na Rys. 9.



Rys. 9: Ogólny podział algorytmów sztucznej inteligencji

W ogólnej dziedzinie sztucznej inteligencji, oprócz algorytmów wchodzących w skład uczenia maszynowego, wymienić można jeszcze trzy inne podejścia do SI. Pierwsze z nich to symboliczna sztuczna inteligencja, której reprezentacją są opisywane wcześniej systemy eksperckie bazujące na logice symbolicznej. Drugie to sieci bayesowskie (sieci decyzyjne) będące graficznymi modelami probabilistycznymi, umożliwiającymi przedstawienie prawdopodobieństwa wystąpienia kolejnego wydarzenia na podstawie zarejestrowanych wydarzeń w przeszłości. Ostatnie z nich to metody ewolucyjne, takie jak algorytmy i programowanie genetyczne oraz ewolucyjne.

Także uczenie maszynowe oprócz poddziedziny głębokiego uczenia zawiera inne metody sztucznej inteligencji. Należą do nich między innymi drzewa decyzyjne wykorzystywane w analizach statystycznych oraz eksploracji danych, algorytmy regresyjne, oraz wykorzystane w rozdziale 6. rozprawy maszyny wektorów nośnych (ang. Support Vector Machines). Do metod uczenia maszynowego nie wchodzącego w skład głębokiego uczenia, zaliczyć można także szczególne przypadki sztucznych sieci neuronowych, w których występuje ograniczona ilość warstw ukrytych (w zależności od opracowania, od jednej do trzech warstw). Istotną cechą algorytmów uczenia maszynowego jest konieczność przeprowadzenia treningu do umożliwienia poprawnego działania

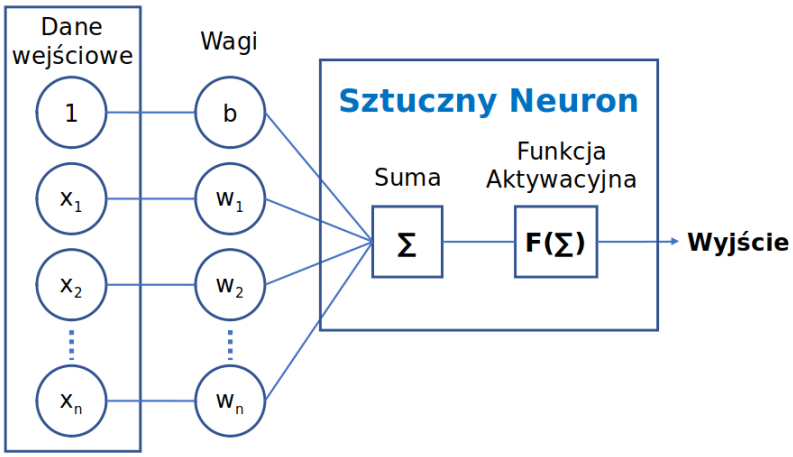
algorytmu. Trening ten może zostać przeprowadzony w różny sposób, z uwagi na typ algorytmu oraz wykorzystane dane treningowe. Algorytmy uczenia maszynowego ze wzmocnieniem wykorzystują dane zawierające jedynie nagrodę lub karę za przeprowadzoną operację. Algorytmy służące do regresji, klasteryzacji lub klasyfikacji korzystają natomiast ze zbiorów danych zawierających próbki informacji, które algorytm będzie przetwarzał w trakcie swojego działania. Z uwagi na sposób etykietowania danych treningowych, algorytmy podzielić można na uczone w sposób nadzorowany, nienadzorowany (czego przykładem może być klasteryzacja) oraz uczone w sposób częściowo nadzorowany, w którym tylko część z danych treningowych jest etykietowana. Taki sam podział występuje w poddziedzinie głębokiego uczenia.

Spośród wymienionych szczebli algorytmów sztucznej inteligencji, głębokie uczenie zawiera najmniej, bo tylko jeden typ algorytmu. Jest nim szczególny zbiór sztucznych sieci neuronowych (SSN), posiadających pewną liczbę warstw ukrytych w swojej architekturze. Pole obecnych zastosowań SSN jest niezwykle szerokie i obejmuje między innymi wizję komputerową, rozpoznanie mowy, przetwarzanie języka naturalnego czy tłumaczenie maszynowe. Zasada ich działania, budowa oraz jej szczególny przypadek – konwolucyjna sieć neuronowa zostaną opisane w kolejnych sekcjach rozprawy – jest to jednocześnie podstawowy algorytm uczenia maszynowego wykorzystany w dalszej, praktycznej części rozprawy.

3.3. SIECI NEURONOWE

Mimo zróżnicowanych architektur wykorzystywanych przez sztuczne sieci neuronowe, istota ich działania nadal oparta jest o koncepcję algorytmu Perceptron, złożonego z pojedynczego sztucznego neuronu, opracowaną w 1958 roku. Zakłada ona budowę grafu, w którym każdy z węzłów (nazywany sztucznym neuronem) wykonuje operację mnożenia danych wejściowych x_i przez wagę w_i, a uzyskanej wartości, powiększonej o jednostkową wartość odchylenia b (ang. bias) używa jako argumentu dla funkcji aktywacji neuronu f. Działanie to przedstawione zostało na Rys. 10 i wzorze (1).

Argumentem funkcji aktywacyjnej jest prosta funkcja liniowa o parametrach ($w_1 \dots w_n$) (stąd problem klasyfikacji XOR). Dodatek wartości odchylenia (b) pozwala na zmianę punktu zerowego funkcji. Funkcją aktywacyjną może być dowolna nieliniowa funkcja ciągła, która dzięki swojej nieliniowości umożliwia rozwiązywanie problemów liniowo nierozdzielnych. Pierwotnie była ona wzorowana na wykresie odpowiedzi elektrycznej aktywnego w mózgu neuronu podobnego do funkcji sigmoid (patrz Rys. 11), lecz w toku zastosowań praktycznych zaczęto stosować mniej skomplikowane krzywe, które jednocześnie osiągały lepsze rezultaty w testach algorytmów. Przykłady funkcji aktywacyjnych Sigmoid, Tanh, ReLu oraz Leaky ReLu zostały przedstawione na Rys. 11.



Rys. 10: Algorytm Perceptron złożony z pojedynczego neuronu

$$y = f\left(\sum_i x_i w_i + b\right) \tag{1}$$

Na Rys. 11, funkcje aktywacyjne różnią się od siebie zarówno zakresem przyjmowanych wartości jak i pochodnymi, pokazanymi w Tab. 3. Ma to szczególne znaczenie w procesie treningu sieci, podczas którego wartość pochodnej funkcji aktywacyjnej wyciągana jest z każdego neuronu w algorytmie. Jednocześnie, dla dużych wartości argumentów, pochodne funkcji Sigmoid oraz Tanh zmierzają do zera, co jest jedną z przyczyn „zanikania gradientu" (ang. gradient vanishing) w algorytmach składających się z wielu połączonych szeregowo neuronów, co uniemożliwia trening sieci [181]. Uniemożliwia to trening neuronów w początkowych warstwach algorytmu, tym samym zmniejszając jego efektywną głębokość, a w konsekwencji także dokładność. Problem ten jednak nie występuje w przypadku funkcji ReLu oraz jej nowszej wersji – Leaky ReLu. Z tego powodu obecnie są to najczęściej stosowane funkcje aktywacyjne w algorytmach neuronowych [182].

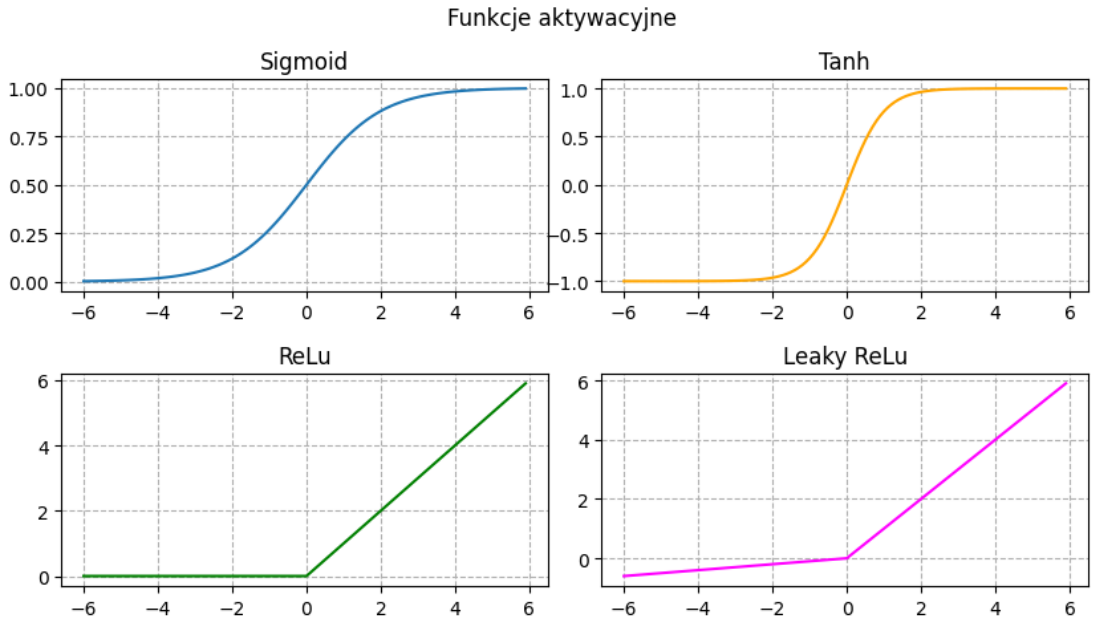
Innym rodzajem funkcji aktywacyjnej jest funkcja występująca na końcu algorytmu. W jej przypadku, oprócz wyznaczenia wartości wyjściowej neuronu, musi ona także zmapować wielowymiarowy wektor wyjściowy algorytmu do (w przypadku klasycznych algorytmów neuronowych) wektora jednowymiarowego i znormalizować go do wybranego rozkładu prawdopodobieństwa. W tym celu najczęściej wykorzystywana jest funkcja Softmax będąca znormalizowaną funkcją wykładniczą uogólniającą funkcję logistyczną do przypadku wielowymiarowego.

Opisywany do tej pory pojedynczy neuron występował w kontekście „algorytmu neuronowego". Algorytmem neuronowym jest zorganizowana w połączone ze sobą warstwy sieć sztucznych neuronów zwana sztuczną siecią neuronową. Sieci takie budowane są w celu eliminacji najpoważniejszej wady jedno neuronowego algorytmu Perceptron – możliwości klasyfikowania wyłącznie danych rozdzielnych liniowo. W sieci wyróżnić można warstwę wejściową (W_0), wyjściową (WW) oraz w przypadku głębokiego uczenia, wewnętrzne warstwy ukryte (W_1 do W_n). W

klasycznym przykładzie sztucznej sieci neuronowej każdy neuron z warstwy *Wn* połączony jest ze wszystkimi neuronami z warstw *Wn-1* oraz *Wn+1* tworząc sieć gęsto połączoną. Przykładowa klasyczna sieć neuronowa została przedstawiona na Rys. 12. Połączenia między neuronami w trakcie działania sieci (na przykład klasyfikowania danych z wektora wejściowego) pozwalają na przepływ danych wyłącznie od warstwy *W0* w kierunku warstwy *WW*. Jedynie w trakcie treningu sieci przepływ danych zostaje odwrócony.

Tab. 3: Wzory przykładowych funkcji aktywacyjnych i ich pochodne

Funkcja *Sigmoid*	**Pochodna**	**Funkcja *Tanh***	**Pochodna**
$f(x)=\frac{1}{1+e^{-x}}$	$f'(x)=\frac{e^{x}}{(e^{x}+1)^{2}}$	$f(x)=\frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$	$f'(x)=\frac{4e^{2x}}{(e^{2x}+1)^{2}}$
Funkcja *ReLu*	**Pochodna**	**Funkcja *LReLu***	**Pochodna**
$f(x)=\begin{cases}0\,dla\,x<0\\x\,dla\,x\geq 0\end{cases}$	$f'(x)=\begin{cases}0\,dla\,x<0\\1\,dla\,x\geq 0\end{cases}$	$f(x)=\begin{cases}0.01\,x\,dla\,x<0\\x\,dla\,x\geq 0\end{cases}$	$f'(x)=\begin{cases}0.01\,dla\,x<0\\1\,dla\,x\geq 0\end{cases}$



Rys. 11: Przykłady funkcji aktywacyjnych

Trening sieci jest wymagany, aby mogła ona poprawnie wykonywać przewidziane dla niej operacje. Do treningu w przypadku uczenia nadzorowanego, wykorzystuje się zbiór etykietowanych danych liczbowych, podzielonych na podzbiór treningowy, testowy oraz ewaluacyjny. Przykładowym rekordem danych etykietowanych może być krotka ({*1, 3, 5, 7*}, {*0*}). Oznacza ona, że dla wejściowego wektora cech {*1, 3, 5, 7*} przyporządkowana została klasa {*0*}. Wymiary klasyfikowanych wektorów lub macierzy cech oraz wektora wyjściowego sieci neuronowej muszą być stałe, co wynika ze statycznej cechy architektury sieci.

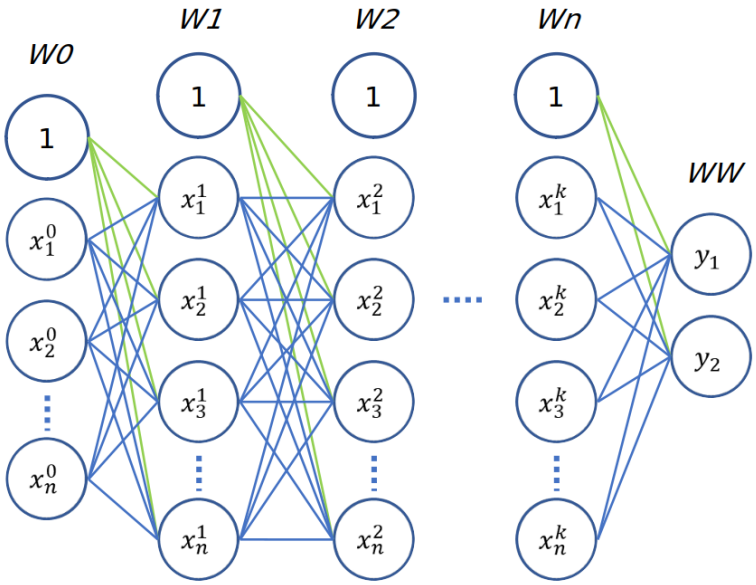
Poszczególne rekordy znajdujące się w zbiorze danych wykorzystywanych do treningu sieci nazywane są punktami danych (ang. datapoints). Zbiór musi być wystarczająco obszerny dla analizowanego zagadnienia (przyjmuje się, że w przypadku klasyfikacji obrazów jest to około tysiąc punktów danych dla każdej z klas [183]) oraz nie zawierać powtarzających się punktów danych. Jego podział na podzbiór treningowy, testowy oraz ewaluacyjny przeprowadza się w sposób zbliżony do proporcji 0,7:0,2:0,1, przy czym istotne jest, aby ten sam punkt danych nie występował w dwóch różnych podzbiorach. Zbiory treningowy i testowy wykorzystywane są w trakcie uczenia sieci. Treningowy – w trakcie regulacji wag w neuronach z wykorzystaniem algorytmów wstecznej propagacji, natomiast testowy – do wyznaczania metryk sieci po zakończonym treningu. Zbiór ewaluacyjny pozwala określić najbardziej korzystny sposób trenowania algorytmu i nie ma bezpośredniego wpływu na wyniki treningu.

Szczególnym typem zbioru danych służącym do treningu sieci neuronowej, są zbiory danych konkursowe, takie jak zbiór danych wizualnych ImageNet [184]. Są one przygotowywane w celu sprawdzenia możliwej do uzyskania dokładności klasyfikacji algorytmu na jak największych i najbardziej różnorodnych zbiorach danych. Jednocześnie, dzięki swojej różnorodności pozwalają na wyodrębnienie z trenowanej na jego bazie sieci neuronowej ogólnego ekstraktora cech charakterystycznych obrazu, nie posiadającego wstępnej skłonności wobec pojedynczego zestawu cech obiektu na obrazie.

Do przeprowadzenia procesu uczenia sieci neuronowej wykorzystuje się algorytmy propagacji wstecznej do wielokrotnego, iteracyjnego porównania wartości wyjściowej sieci z wartością oczekiwaną. Ilość przeprowadzonych porównań uzależniona jest od wielkości zbioru uczącego, założonej liczby iteracji oraz przebiegu treningu sieci. Bez użycia algorytmów wstecznej propagacji, samo porównanie wartości wyjściowych umożliwiałoby jedynie aktualizację wag w ostatniej warstwie ukrytej sieci (Wn), ponieważ wewnętrzne jej warstwy działają jak czarna skrzynka. Aby je aktualizować, algorytm wstecznej propagacji, oblicza wartości pochodnej funkcji aktywacyjnej każdego z neuronów w sieci, na podstawie wektora wejściowego do każdej z warstw. Pozwala to w konsekwencji na wykorzystanie metody prostego lub częściej stochastycznego gradientu (SGD – ang. Stochastic Gradient Descent) do aktualizacji wag w neuronach sieci w odniesieniu do funkcji kosztów. Aby przyspieszyć trening sieci, wykorzystuje się także algorytmy wspomagające SGD, takie jak algorytm Nesterova czy adaptacyjne szacowanie momentu ADAM (ang. adaptive moment estimation) [185].

Ogólny wzór (2) na podstawie którego przeprowadzana jest propagacja wsteczna przedstawiony został poniżej. Na jego podstawie obliczana jest pochodna cząstkowa błędu wartości wyjściowej sieci w odniesieniu do danej wagi połączenia neuronów w warstwach i i j $w_{i,j}$, funkcji kosztów E, wartości wyjściowej z danego neuronu sieci w warstwie j – o_j oraz końcowej wartości wyjściowej całej sieci net_j. Dzięki obliczeniom błędu sieci w odniesieniu do danej wagi, wyznaczyć

można wielkość kroku, o który należy zmienić wagę tak, aby globalnie zbliżyć sieć do wartości oczekiwanej. Proces ten powtarzany jest dla każdego neuronu w każdej warstwie trenowanej sieci aż do zakończenia treningu, kiedy wartości wag neuronów zapisywane są w sposób trwały w sieci [186–188].



Rys. 12: Ogólny schemat klasycznej sieci neuronowej

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{i,j}} \tag{2}$$

3.4. KONWOLUCYJNE SIECI NEURONOWE

Odmianą klasycznych sieci neuronowych są konwolucyjne sieci neuronowe, opracowane z myślą o analizie, klasyfikacji i przetwarzaniu obrazów, a także przetwarzaniu języka naturalnego. Swoją nazwę zawdzięczają podstawowej operacji wykonywanej w warstwach sieci – operacji konwolucji. Ich koncepcja powstała w latach osiemdziesiątych i oparta była na symulacji wielopoziomowego przetwarzania danych pochodzących z siatkówki przez kolejne warstwy neuronów w mózgu. Pozwalało to na zachowanie oryginalnej orientacji przestrzennej danych (w formie wielowymiarowych macierzy liczb) w sieci, kluczowej w zagadnieniu rozpoznawania obrazów. W zagadnieniu tym istotna jest nie tylko poszukiwana cecha obrazu, ale także jej pozycja względem innych cech charakterystycznych. Tym samym, w odróżnieniu od klasycznych sieci neuronowych, warstwy sieci konwolucyjnych nie są połączone w sposób gęsty na całej swojej głębokości [189]. Połączenia gęste w tych sieciach występują jedynie w ich ostatnich warstwach, odpowiedzialnych za ostateczną klasyfikację przetworzonych wcześniej danych wejściowych.

Chronologicznie pierwszą siecią konwolucyjną była opracowana w 1998 roku sieć LeNet (której nazwa pochodzi od nazwiska twórcy – Yann LeCun). Była to relatywnie płytka sieć,

posiadająca wyłącznie dwie warstwy konwolucyjne i dwie gęsto połączone. Jako funkcję aktywacyjną wykorzystywała krzywą Tanh. Była wykorzystywana do rozpoznawania ręcznie pisanych liter i cyfr na jednokanałowych obrazach wielkości 32 na 32 piksele, osiągając w tym zagadnieniu dokładność (odsetek poprawnych wskazań algorytmu wyrażony w procentach) sięgającą 98% [190], lecz nie sprawdziła się w bardziej złożonych zagadnieniach. Kolejna próba wykorzystania sieci konwolucyjnych w bardziej rozbudowanej formie została podjęta dopiero w 2012 roku z wykorzystaniem sieci AlexNet. Posiadała ona pięć warstw konwolucyjnych i trzy gęsto połączone. Mimo jedynie trzech warstw konwolucyjnych więcej niż jej poprzednik, była w stanie rozpoznawać trójkanałowe obrazy należące do zbioru ImageNet z dokładnością około 85% [184,191]. Jednocześnie, mimo pozornie niewielkiego rozbudowania, do treningu wymagała aż dwóch wydajnych procesorów graficznych jednocześnie. Ostatnią z konwolucyjnych sieci neuronowych odnoszącą sukcesy na zbiorze ImageNet, była sieć w VGG (ang. Visual Geometry Group) opracowana w 2014 roku, złożona z 16, 19 lub 23 warstw konwolucyjnych. W jej najpłytszym wariancie wykorzystywała ponad czternaście milionów parametrów i osiągała dokładność rzędu 95% na zbiorze ImageNet.

Najnowsze warianty sieci konwolucyjnych opracowane przez ostanie pięć lat, zamiast klasycznej budowy sekwencyjnej sieci, wykorzystują mikroarchitektury w połączeniu ze znacznie większą ilością trenowalnych warstw konwolucyjnych. Przykładami takich sieci jest sieć GoogLeNet, w której zastosowano mikroarchitekturę nazwaną Incepcion oraz ResNet wykorzystujący moduły szczątkowe do trenowania sieci neuronowych o głębokości dochodzącej do 150 warstw. Nie byłoby to możliwe przy zastosowaniu modelu sekwencyjnego, z uwagi na problem zanikającego gradientu w początkowych warstwach sieci. Sieci te w testach na zbiorze ImageNet osiągają dokładności wyższe niż ludzie, przez co sam konkurs zmienił najważniejszą konkurencję z rozpoznawania obrazu na lokalizację konkretnego obiektu na obrazie.

3.4.1. BUDOWA KONWOLUCYJNYCH SIECI NEURONOWYCH

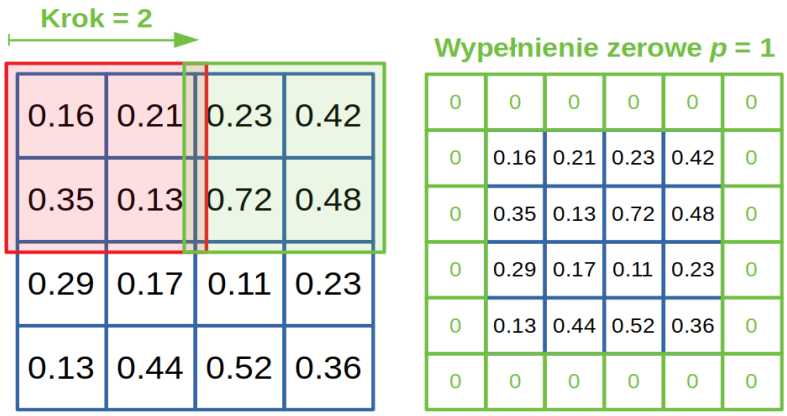
Jak wspomniano w poprzednich sekcjach, sztuczne sieci neuronowe składają się z połączonych ze sobą warstw, a ich szczególny przypadek – konwolucyjne sieci neuronowe w swojej budowie wykorzystują warstwy nie połączone w pełni między sobą, w celu zachowania przestrzennego rozłożenia danych wejściowych. Podstawową warstwą sieci konwolucyjnych jest warstwa konwolucyjna, lecz oprócz niej występują także:

- warstwa aktywacyjna,
- warstwa łącząca – (ang. Pooling layer),
- warstwa w pełni połączona (gęsto połączona).

Oprócz warstw, sieci konwolucyjne wykorzystują także operatory działania na macierzach, takie jak krok oraz zerowe wypełnienie. Pozwalają one w efektywny sposób zarządzać wymiarami

macierzy przetwarzanej przez daną warstwę sieci. W kolejnych sekcjach omówione zostaną podstawowe elementy składowe sekwencyjnej, konwolucyjnej sieci neuronowej.

Krokiem nazywana jest wielkość przeskoku między współrzędnymi macierzy wejściowej, o jaki przesuwa się macierz filtra konwolucyjnego. Zwykle wartość ta wynosi 1 lub 2. Zerowe wypełnienie to sztuczne zwiększenie wymiarów macierzy wejściowej, poprzez dodanie do niej wierszy i kolumn wypełnionych wartościami zerowymi tak, aby po przeprowadzonej operacji konwolucji, wymiar macierzy wyjściowej odpowiadał wymiarowi macierzy wejściowej. Działanie obu operatorów zostało przedstawione na Rys. 13.



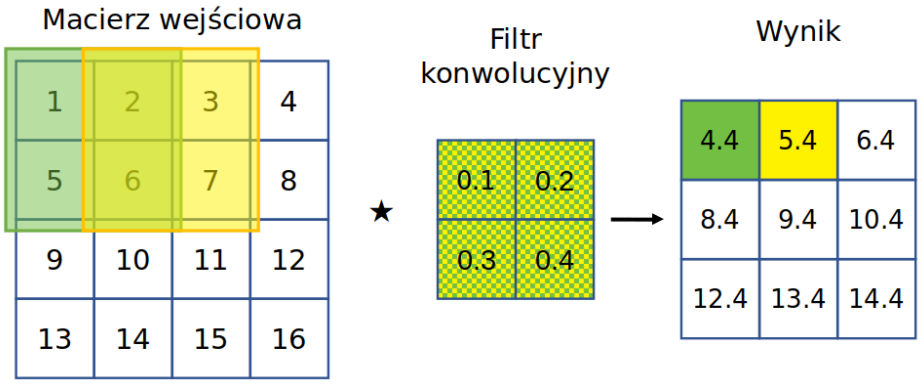
Rys. 13: Wizualizacja kroku o wartości 2 i zerowego wypełnienia o wartości 1

3.4.1.1. WARSTWA KONWOLUCYJNA

Warstwy konwolucyjne odpowiedzialne są za przeprowadzanie operacji konwolucji między macierzą wejściową $\boldsymbol{I}$, a macierzą $\boldsymbol{K}$ utworzoną z wag w każdej z warstw sieci. Macierz $\boldsymbol{K}$ nazywana jest filtrem konwolucyjnym. W praktyce jednak, z uwagi na mniejsze zapotrzebowanie na moc obliczeniową, operacja konwolucji w strukturach obliczeniowych zastępowana jest operacją wzajemnej korelacji macierzy, lecz ze względów historycznych, termin „konwolucja" pozostał obowiązujący. Działanie operacji konwolucji opisuje wzór (3), natomiast jej graficzny opis razem z zaznaczonymi pierwszymi dwoma etapami obliczeń dla kroku s wynoszącego 1, w sposób schematyczny przedstawia Rys. 14.

Na Rys. 14, filtr konwolucyjny ma mniejszy wymiar niż macierz wejściowa. Z tego powodu, operacja konwolucji przeprowadzana jest na kolejnych wycinkach macierzy wejściowej, a filtr konwolucyjny porusza się po niej według zadanego kroku. Wynikiem operacji konwolucji jest macierz o wymiarze mniejszym niż macierz wejściowa (końcowy wymiar macierzy zależy od wielkości filtra konwolucyjnego, zadanego kroku oraz wypełnienia zerowego macierzy wejściowej). Pozwala to na zachowanie przestrzennych cech obrazu. W pierwszych warstwach sieci rejestrowane są niewielkie kształty, natomiast kształty ogólne, takie jak długie krawędzie obiektów, rejestrowane są

w warstwach bliskich wyjściowej. Jednocześnie, w przypadkach głębokich sieci, aby wymiary macierzy wyjściowych kolejnych warstw nie zmniejszyły się poniżej wymiarów filtra konwolucyjnego, stosuje się opisany wcześniej operator zerowego wypełnienia.



Rys. 14: Przykład operacji konwolucji [11]

$$S(i,j)=(K*I)(i,j)=\sum_m \sum_n K(i+m,j+n)I(m,n) \tag{3}$$

Filtry konwolucyjne w warstwach konwolucyjnych rzadko występują pojedynczo. Zazwyczaj wykorzystywanych jest kilkadziesiąt do kilkuset filtrów naraz, znacznie zwiększając wymiary macierzy wejściowej. Na przykład, jeśli macierz wejściowa o wymiarach (256, 256, 3) (trójkanałowy obraz o wielkości 256 na 256 pikseli) zostanie przepuszczona przez filtr konwolucyjny o wymiarach (3, 3, 64) (64 filtry konwolucyjne o wymiarach 3 na 3), to dla kroku s równego 1 i braku zerowego wypełnienia, otrzymana mapa cech macierzy wejściowej będzie miała wymiary (254, 254, 64). Zależność ta opisana jest wzorem 4, w którym $\boldsymbol{I}$ oraz $\boldsymbol{K}$ to kolejno pojedyncze wymiary macierzy wejściowej oraz filtra konwolucyjnego.

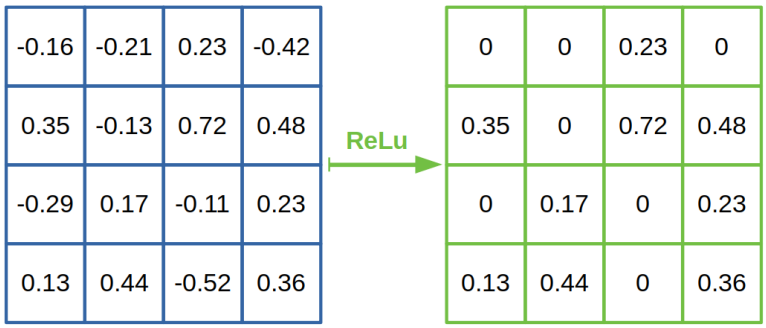
$$D=\left[\frac{(I-K+2p)}{s}\right]+1 \tag{4}$$

Wagi zawarte w filtrach konwolucyjnych oraz ostatnich, gęsto połączonych warstwach sieci to parametry algorytmu podlegające treningowi.

3.4.1.2. Warstwa aktywacyjna

Warstwa aktywacyjna w sieciach konwolucyjnych spełnia to samo zadanie co funkcja aktywacyjna w sieciach klasycznych. Wykonuje ona operację podstawienia do funkcji aktywacyjnej poszczególnych wartości w mapie cech macierzy otrzymanych w wyniku operacji konwolucji. W wyniku przeprowadzenia macierzy przez warstwę aktywacyjną otrzymywana jest macierz o niezmienionych wymiarach, zawierająca wyniki operacji dla każdego elementu macierzy wejściowej.

Warstwy aktywacyjne mogą wykorzystywać dowolną funkcję aktywacyjną, lecz obecnie najczęściej wykorzystywane są funkcje ReLu i Leaky ReLu. Przykład działania warstwy aktywacyjnej przedstawia Rys. 15.



Rys. 15: Przykład działania funkcji aktywacyjnej ReLu

3.4.1.3. Warstwa łącząca

Warstwy łączące wykorzystują operacje łączenia maksymalnego (ang. max pooling) oraz łączenia średniego (ang. average pooling) w celu selekcji z macierzy cech wartości będących odpowiedzią na najwyższe wagi filtru konwolucyjnego lub uśrednienia ich dla filtra zadanej wielkości. Operacja ta w założeniu odwzorowywać ma schemat działania biologicznych neuronów, w których promowane są najsilniejsze sygnały docierające z synaps.

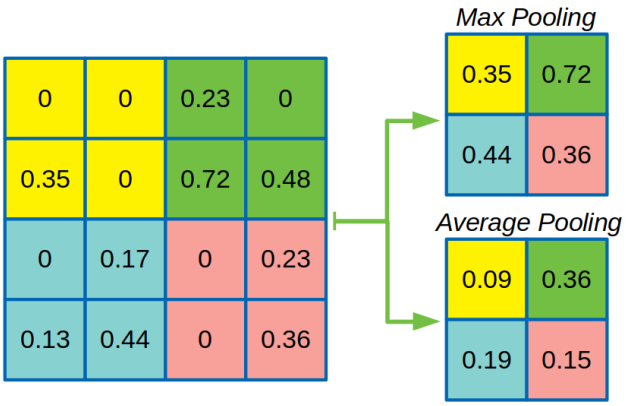
Podobnie jak w przypadku warstw konwolucyjnych, wymiary filtra warstwy łączącej są mniejsze niż wymiary macierzy wejściowej. Warstwy łączące zazwyczaj umieszczone są po warstwach aktywacyjnych, w celu zmniejszenia wymiarów macierzy cech. Przykładowa operacja zastosowania warstwy łączenia maksymalnego oraz średniego przeprowadzona na dwuwymiarowej macierzy z Rys. 15, została przedstawiona na Rys. 16. Na przedstawionym przykładzie zastosowano krok s o wartości 2.

3.4.1.4. Warstwa w pełni połączona

W przypadku sieci konwolucyjnych, warstwy w pełni połączone (ang. fully connected) nie różnią się znacznie od jej odpowiedników w klasycznych sieciach neuronowych. Podobnie jak w klasycznym odpowiedniku występują w szeregu zawierającym często więcej niż jedną warstwę. Jedyne różnice polegają na jej umiejscowieniu – wykorzystywana jest na końcu konwolucyjnej części sieci, zaraz przed kocową warstwą aktywowaną funkcją Softmax,. Ponieważ wynikiem działania warstwy konwolucyjnej, aktywacyjnej oraz łączącej zawsze jest wielowymiarowa macierz, przed przeprowadzeniem jej przez warstwę w pełni połączoną musi być ona przekształcona do

jednowymiarowego wektora. W dalszej kolejności, operacje przez nią wykonywane przebiegają analogicznie do klasycznej sieci neuronowej.

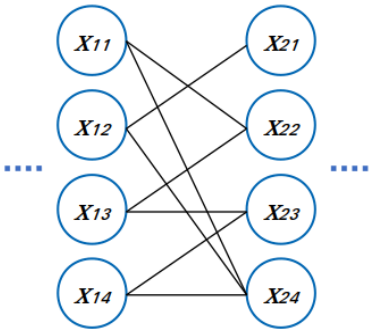
Jednocześnie, warstwy w pełni połączone zawierają większość z parametrów sieci neuronowej. Na przykład dla wspomnianej wcześniej sieci w architekturze VGG16, liczba parametrów (wag filtrów) zawarta we wszystkich warstwach konwolucyjnych wynosi około 14 milionów, podczas gdy w samych trzech ostatnich warstwach gęsto połączonych, liczba parametrów przekracza 130 milionów.



Rys. 16: Przykład operacji łączenia maksymalnego i średniego macierzy

3.4.1.5. Warstwa losowych przerwań

Często występującą w sieciach neuronowych warstwą jest warstwa losowych przerwań. Jej zadanie ogranicza się do losowego przerywania części połączeń między następującymi po sobie warstwami, najczęściej między warstwami w pełni połączonymi. Stosowana jest w celu ograniczenia wpływu pojedynczego neuronu na końcowy wynik wskazania sieci, zapobiegając treningowi stronniczego algorytmu. Wizualizacja działania warstwy losowych przerwań z parametrem $p = 0,5$ określającym prawdopodobieństwo przerwania połączenia między neuronami, została przedstawiona na Rys. 17.



Rys. 17: Przykład działania warstwy losowych przerwań

Warstwa ta jest jedyną z opisywanych, która zachowuje się w sposób dynamiczny w trakcie treningu sieci. Aby wytrenowany algorytm nie był stronniczy, przerywane połączenia między warstwami neuronów zmieniają się w sposób pseudolosowy z każdą iteracją treningu algorytmu.

3.4.2. ARCHITEKTURY KONWOLUCYJNYCH SIECI NEURONOWYCH

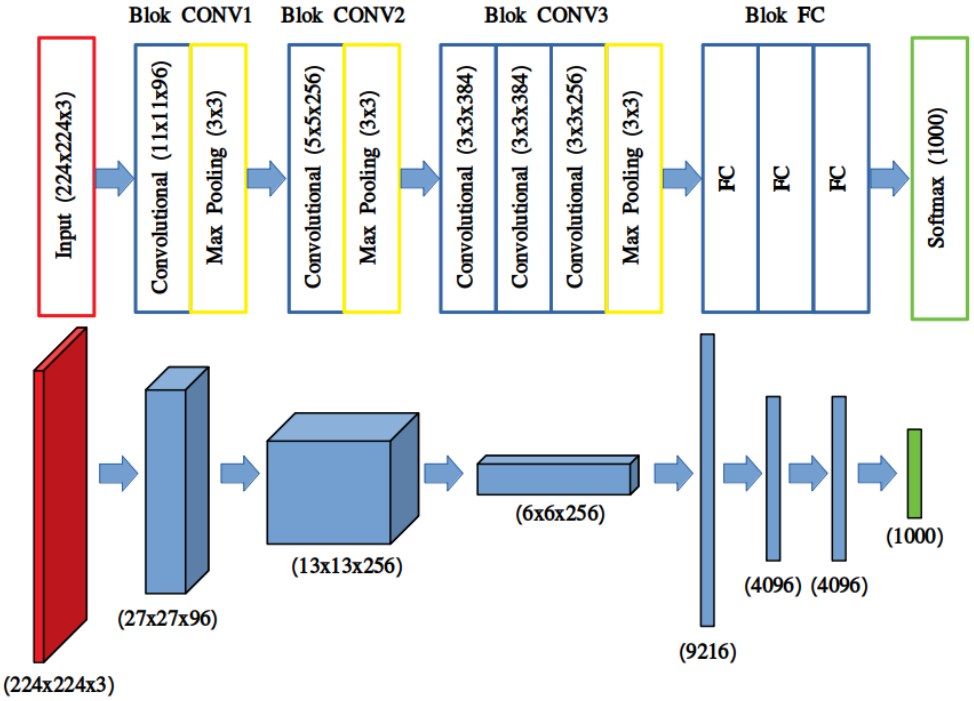
Sieci konwolucyjne wykorzystują przedstawione powyżej typy warstw w zorganizowany, niezmienny w trakcie treningu i wykorzystania sposób, nazywany architekturą sieci. Ma ona wpływ na możliwości uczenia się algorytmu oraz wydajność jego działania, nie tylko z uwagi na liczbę zastosowanych warstw, w których przebiega uczenie się algorytmu, ale także wymiary stosowanych filtrów konwolucyjnych i liczbę neuronów warstw w pełni połączonych. Mimo, że z reguły sieci o większej liczbie warstw konwolucyjnych osiągają większe dokładności w rozpoznaniu obrazu, istotne jest także umiejscowienie warstw odpowiadających za aktywację i łączenie macierzy. Typowy blok, z którego złożone są sekwencyjne sieci konwolucyjne został przedstawiony we wzorze (5).

$$[CONV] \rightarrow [ACT] \rightarrow [POOL] \stackrel{\text{def}}{=} [CONV] \rightarrow [POOL] \tag{5}$$

Wzór ten przedstawia dwa równoznaczne zapisy głównego bloku sieci neuronowej w najczęściej spotykanej w praktyce kolejności. Jednocześnie, w graficznym zapisie architektury sieci konwolucyjnych zazwyczaj pomija się oznaczanie warstw aktywacyjnych, gdyż występują razem z warstwami konwolucyjnymi.

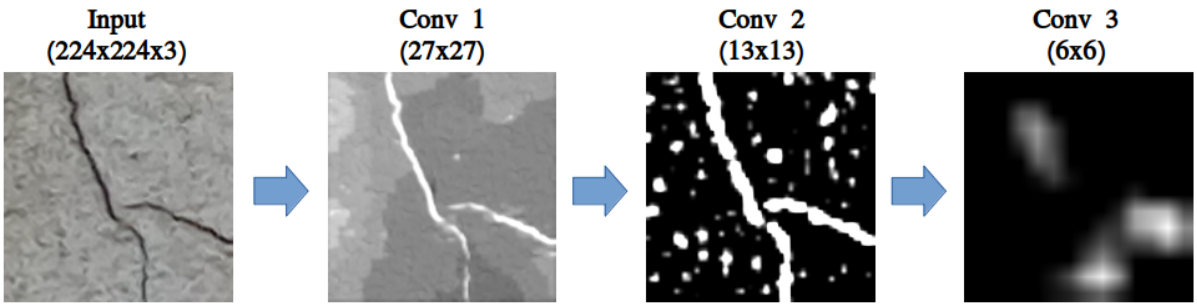
Przykładami sekwencyjnych sieci konwolucyjnych są sieci LeNet, AlexNet oraz sieci z rodziny VGG, wspomniane wcześniej w sekcji 3.4. Ich charakterystyczną cechą jest relatywnie prosta budowa, w której elementem powtarzalnym jest główny blok konstrukcyjny zapisany wzorem 4. Na górnej części Rys. 18 przedstawiona została architektura sieci AlexNet wraz wymiarami filtrów w poszczególnych warstwach. Część dolna rysunku zawiera reprezentację graficzną wymiarów generowanych przez nie macierzy cech.

Na przedstawionym przykładzie widać, że wraz z głębokością sieci zmniejsza się wymiar macierzy cech obrazu wejściowego, lecz rośnie liczba opisujących je macierzy. W końcowej części sieci zbiór macierzy cech transformowany jest do jednowymiarowego wektora, który następnie przetwarzany jest w sposób analogiczny do klasycznych sieci neuronowych. Ostatnia warstwa aktywacyjna wykorzystuje funkcję Softmax, aby zwrócić prawdopodobieństwo detekcji każdej z klas. Na przykładzie 1000 możliwych klas zbioru ImageNet, wektor wyjściowy zawiera dokładnie 1000 elementów.



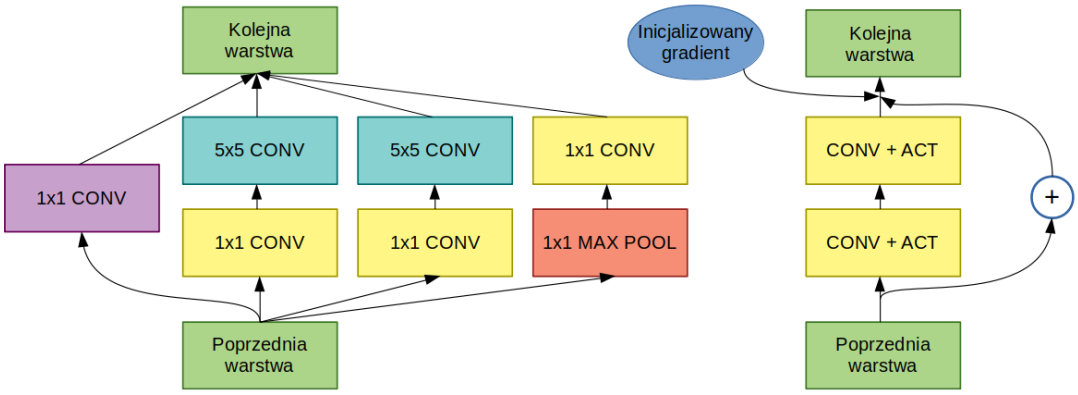
Rys. 18: Przykład sieci sekwencyjnej z graficzną reprezentacją wymiarów macierzy cech

W sekcji 3.2., przedstawiony został ogólny opis metod sztucznej inteligencji, w którym wewnętrzne warstwy sieci neuronowych zostały opisane jako czarna skrzynka. W ten sam sposób działają wewnętrzne warstwy konwolucyjne, lecz wyniki ich działania można wizualizować. Chodzi o mapy cech oraz mapy aktywacyjne powstałe w wyniku przetworzenia wstępnej macierzy obrazu przez bloki konwolucyjne. Wizualizacja map aktywacyjnych poszczególnych warstw konwolucyjnej sieci AlexNet, dla obrazu zawierającego uszkodzenie w postaci rysy powierzchni betonu została przedstawiona na Rys. 19. Na rysunku, w kolejnych warstwach wytrenowanej na danym zbiorze danych sieci promowane są części obrazu odpowiedzialne za główne cechy poszukiwanego na nim obiektu. W ostatniej warstwie mapa cech generowana przez warstwę konwolucyjną zawiera już tylko elementy uszkodzenia. Jednocześnie, ilustracja pokazuje tylko po jednym przykładzie mapy aktywacyjnej z każdego z bloków konwolucyjnych, podczas gdy w trakcie działania sieć AlexNet generuje w sumie 608 takich map.



Rys. 19: Wizualizacja map cech obrazu wejściowego w kolejnych warstwach sieci

Głębokości architektur sekwencyjnych sieci konwolucyjnych sięgają do 25 ukrytych warstw konwolucyjnych. Z reguły uznaje się, że dokładność sieci neuronowej rośnie wraz z jej głębokością. Niestety, w przypadku sieci sekwencyjnych (o architekturze, w której warstwy połączone są szeregowo), wraz z głębokością algorytmu rośnie także ryzyko wystąpienia problemu wspomnianego w sekcji 3.3. – zanikania gradientów. Może on doprowadzić do ograniczenia efektywnej głębokości sieci przez brak możliwości aktualizowania wag w pierwszych warstwach algorytmu. Z tego powodu głębokość sieci sekwencyjnych z reguły nie przekracza 30 warstw. Kolejnym problemem związanym z dużą liczbą warstw w algorytmie jest zapotrzebowanie na moc obliczeniową związaną z jego treningiem i wykorzystaniem. Nie przekłada się ona jednak w sposób liniowy na dokładność algorytmu. Na przykład różnica między dokładnością sieci VGG13 (13 warstw konwolucyjnych) i VGG19 (19 warstw konwolucyjnych) wynosi około jednego procenta, przy czym dodatkowe sześć warstw konwolucyjnych wydłuża czas obliczeń o niemal 30 procent [192].



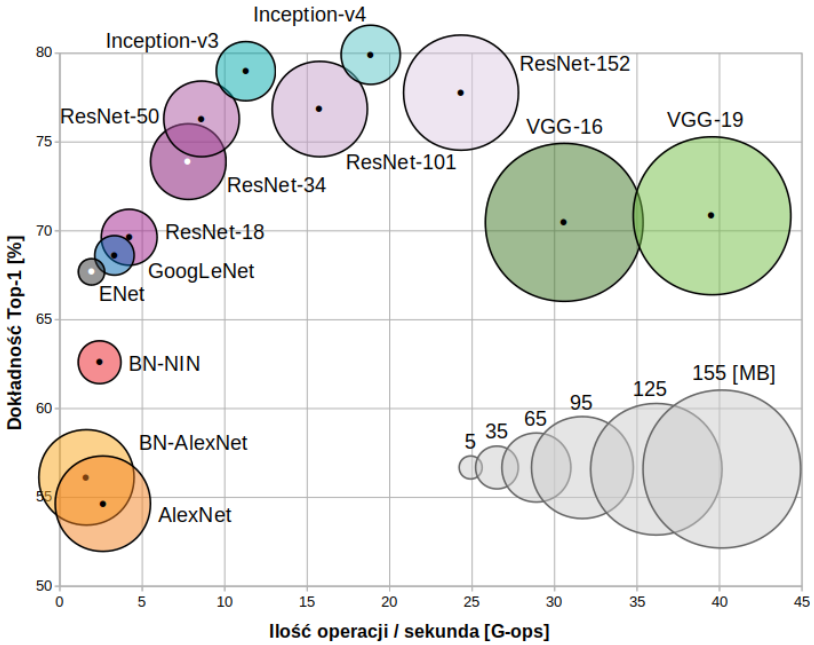
Rys. 20: Mikroarchitektury stosowane w sieciach GoogLeNet (po lewej) i ResNet (po prawej) [247,248]

Trening sieci konwolucyjnych zbudowanych z większej ilości warstw ukrytych jest możliwy poprzez zastosowanie mikroarchitektury sieci oraz ponowną inicjalizację gradientu w wewnętrznych warstwach sieci podczas treningu. Pierwszy ze sposobów zastosowano w sieci GoogLeNet, wykorzystującej moduł sumujący mapy aktywacyjne z warstw konwolucyjnych o filtrach różnej wielkości, wykonujących obliczenia równolegle względem siebie. Oba sposoby zostały natomiast zastosowane w sieci ResNet, wykorzystującej moduły resztkowe do utrzymania niezerowej wartości gradientu, a także technikę ponownej inicjalizacji gradientu w jednej lub dwóch głębokich, ukrytych warstwach konwolucyjnych. Wizualizacje elementów mikroarchiterktur sieci GoogLeNet oraz ResNet zostały przedstawione na Rys. 20.

W sekcji przedstawiono jedynie ogólny podział architektur konwolucyjnych sieci neuronowych. Ich konkretne zastosowania, oprócz wymienionych już sieci takich jak VGG16 czy ResNet implementowane były pod różnymi nazwami. Jednak nawet sieci posiadające tę samą bazową architekturę (jak rodzina sekwencyjnych sieci VGG) mogą występować w licznych wariacjach. Te

różnić mogą się od siebie na przykład stosowaniem dodatkowych warstw łączących lub różnych parametrów warstw przerwań. Należy mieć także na uwadze fakt, iż mimo że niektóre typy sieci mogą wydawać się przestarzałe lub nieefektywne w kontekście rozpoznawania obrazu, ich architektura nadal może pozostawać w użyciu do innych zagadnień. Przykładem takich sieci jest rodzina sieci VGG, które mimo osiągania gorszych wyników w rozpoznawaniu obrazu oraz mniej wydajnej ze względów obliczeniowych architektury niż sieci z grupy ResNet, nadal z powodzeniem jest wykorzystywana do segmentacji semantycznej obrazów jako sieć w pełni konwolucyjna FCN (ang. Fully Convolutional Network).

Na Rys. 21 przedstawiono zestawienie wybranych sieci konwolucyjnych z uwzględnieniem ich dokładności wskazania o najwyższym prawdopodobieństwie (dokładność Top-1), liczbie wykonywanych na sekundę operacji oraz wielkości modelu w MB [192].



Rys. 21: Zestawienie istniejących konwolucyjnych sieci neuronowych

3.4.3. TRENING SIECI

Jak zaznaczono w sekcji 3.3., uczenie sieci neuronowych jest procesem iteracyjnym, w którym wykorzystywany jest algorytm propagacji wstecznej do korygowania wag połączeń między neuronami. W analogiczny sposób korygowane są wagi występujące w filtrach konwolucyjnych warstw ukrytych konwolucyjnych sieci neuronowych. Uczenie sieci neuronowych wymaga wielokrotnych testów parametrów uczenia. Nie istnieją bowiem konkretne wytyczne, które pozwalałyby na wyznaczenie tych parametrów bez każdorazowego testu algorytmu. Proces ten często określany jest jako po części nauka, a po części sztuka [193].

Do uczenia sztucznych sieci neuronowych używa się parametrów, które nazywane są hiperparametrami. Należą do nich:

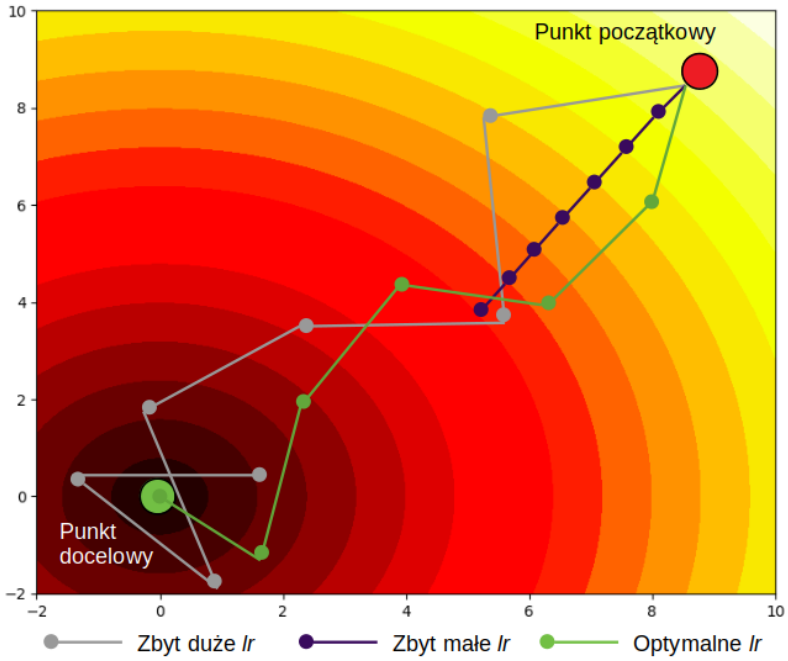
- liczba epok w trakcie których trenowana będzie sieć,
- wielkość pakietu analizowanych danych (ang. batch size),
- wybór funkcji optymalizacyjnej,
- współczynnik uczenia się (*lr* – ang. learning rate),
- tempo uczenia się (ang. learning rate momentum),
- zanik współczynnika uczenia się (ang. learning rate decay),
- charakterystyki niektórych warstw głębokich.

Trening sieci neuronowej rozpoczyna się od pewnych wstępnych wartości wag w neuronach. Wielkości tych wag mogą odgrywać istotną rolę w treningu sieci, gdyż niewłaściwa ich inicjalizacja może powodować zanik lub eksplozję (wzrost wykładniczy) wielkości wag w trakcie treningu sieci, uniemożliwiając poprawną naukę algorytmu. Jedną z możliwości inicjalizowania wag w algorytmach neuronowych jest wykorzystanie macierzy zerowych. Jednak symetryczność tych macierzy może prowadzić do obniżenia dokładności końcowego algorytmu. Z tego powodu, obecnie wagi inicjalizowane są najczęściej w sposób pseudolosowy z wykorzystaniem rozkładu normalnego lub jednostajnego ciągłego, na przykład w przedziale od -0,05 do 0,05 [194]. Po wstępnym przyjęciu wag dla wszystkich warstw algorytmu rozpocząć się może pierwsza epoka procesu uczenia sieci.

Epoką uczenia się algorytmu neuronowego nazywa się fragment procesu treningu, w którym zbiór treningowy przeprowadzany jest przez architekturę sieci. Liczba epok treningu algorytmu może wynosić od kilku do kilkuset, a nawet kilku tysięcy. Jest ona uzależniona od wielkości zbioru treningowego, typu sieci i specyfiki zadania. Wstępnie zakładana liczba epok uczenia algorytmu z reguły ustalana jest na wyższym poziomie niż finalnie wymagany. W praktyce trening przerywa się ręcznie, kiedy algorytm osiągnie zadowalające metryki.

Z liczbą epok treningowych bezpośrednio związana jest wielkość zbioru danych, które sieć analizuje. Aby przyspieszyć proces uczenia się sieci, aktualizacja wag w neuronach nie odbywa się wyłącznie raz, po upływie całej epoki (przeprowadzeniu całego zbioru uczącego przez sieć), ale za każdym razem, kiedy sieć podda analizie pojedynczy pakiet danych. Pojedynczy pakiet może przybierać rozmiary od kilkudziesięciu punktów danych do rozmiarów całego zbioru uczącego, w zależności od możliwości obliczeniowych maszyny, na której prowadzony jest trening. Niewielki rozmiar pakietu danych może skutkować znacznym wydłużeniem treningu, natomiast jeśli jej rozmiar będzie zbyt duży, aktualizacje wag w neuronach mogą okazać się zbyt duże by sieć mogła osiągnąć globalne minimum funkcji kosztów. To natomiast poszukiwane jest z wykorzystaniem funkcji

optymalizacyjnych takich jak SGD lub ADAM, wspomnianych wcześniej w sekcji 3.3.. Wybór konkretnej funkcji zależy od specyfiki zadania wykonywanego przez sieć neuronową i wymaga przeprowadzenia testów w trakcie treningu algorytmu, a wybór ten może być kluczowy dla finalnych metryk sieci.

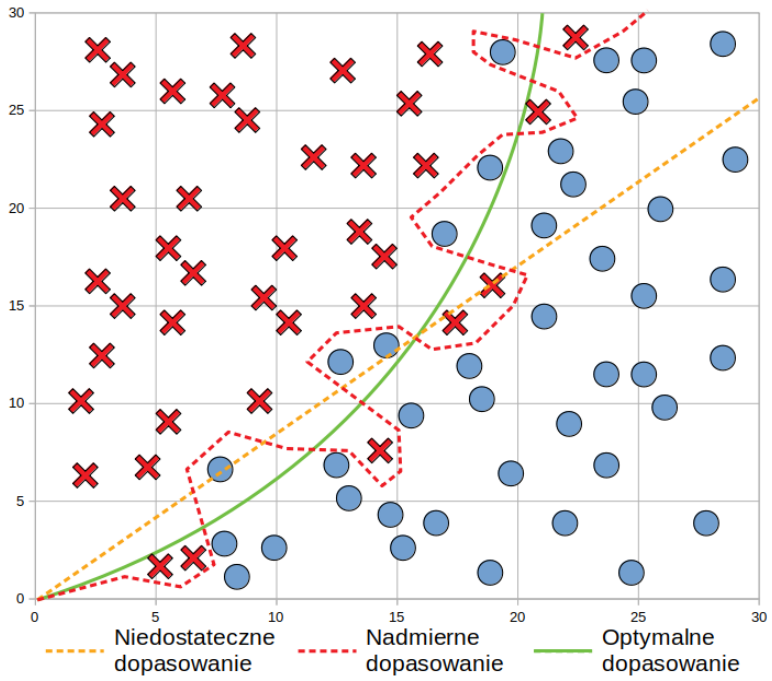


Rys. 22: Różnice w doborze wielkości współczynnika uczenia

Parametrem funkcji optymalizacyjnej jest współczynnik uczenia się oraz dwie jego dodatkowe charakterystyki – tempo wzrostu oraz zaniku tego parametru. Współczynnik uczenia się to parametr określający wielkość kroku zmiany wag w neuronach obliczany dla każdego pakietu danych w trakcie treningu. Jego wielkość może być stała w trakcie całego procesu uczenia sieci, lecz może to skutkować zatrzymaniem się nauki w lokalnym minimum funkcji kosztów, jeśli parametr został określony jako zbyt mały, lub ominięciem globalnego minimum, jeśli jego wartość została przyjęta jako zbyt duża. Opisane sytuacje zostały przedstawione na Rys. 22. Aby ograniczyć wpływ stałej wielkości współczynnika uczenia się na trening algorytmu, wykorzystuje się algorytmy powodujące dynamiczny zanik jego wielkości w trakcie treningu. Umożliwia to szybki postęp treningu na jego początku z wykorzystaniem dużych wartości *lr* oraz coraz dokładniejsze kroki aktualizujące wagi w sieci na końcowym jego etapie. Kolejnym parametrem wspomagającym uczenie sieci jest stosowanie algorytmów wprowadzających do funkcji optymalizacyjnej współczynnik tempa uczenia się (ang. learning rate momentum). Promuje on zmianę wag algorytmu neuronowego w ten sposób, że promowany kierunek aktualizacji wag pokrywa się z nachyleniem funkcji kosztów. Umożliwia to większe kroki aktualizujące wagi sieci w początkowej fazie treningu oraz zapobiega wydostawaniu się wag z globalnego minimum funkcji [195].

Ostatnim czynnikiem mającym wpływ na jakość treningu sieci jest stosowanie dodatkowych warstw, takich jak warstwa losowych przerwań, opisana w sekcji 3.4.1. Ich zastosowanie ma z reguły pozytywny wpływ na ostateczną dokładność algorytmu, zwiększając jego zdolności do generalizowania wiedzy. Jednak zbyt agresywne korzystanie z warstw przerwań (na przykład poprzez ustalenie parametru p na poziomie powyżej 0,8) może skutkować wydłużeniem się czasu treningu sieci.

Skończony trening algorytmu neuronowego musi zostać oceniony, gdyż z pozoru poprawnie uczona sieć neuronowa wciąż może przejawiać niekorzystne cechy, takie jak nadmierne (ang. overfitting) lub niedostateczne dopasowanie (ang. underfitting). Obie te cechy uwidaczniają się po przeprowadzeniu przez algorytm neuronowy danych ze zbioru ewaluacyjnego i skutkują obniżeniem dokładności wskazań algorytmu. Zostały zwizualizowane na Rys. 23.



Rys. 23: Wizualizacja cech dopasowania wytrenowanego algorytmu neuronowego

W przypadku wystąpienia niekorzystnych cech dopasowania sieci neuronowej, należy zmodyfikować cechy treningu lub bazę danych, na której uczona była sieć. Problem nadmiernego dopasowania zwalcza się przy pomocy zwiększenia zbioru uczącego, lub modyfikację punktów danych przed treningiem algorytmu z wykorzystaniem metod rozszerzania wykorzystanych już punktów (ang. data augmentation) Jeśli natomiast nie jest to możliwe, wykorzystuje się alternatywne metody treningu sieci, opisane w sekcji 4.5. Niedostateczne dopasowanie występuje zazwyczaj przy treningu nie dość rozbudowanych w stosunku do analizowanego zagadnienia sieci neuronowych. Aby

ograniczyć jego wpływ, należy zmodyfikować algorytm, na przykład poprzez dodanie dodatkowych warstw głębokich lub zmianę parametrów warstw wykorzystywanych obecnie.

3.5. SZTUCZNA INTELIGENCJA W INŻYNIERII BUDOWLANEJ

Na podstawie literatury podsumowującej liczbę publikacji związanych z tematyką sztucznej inteligencji w inżynierii budowlanej w latach 2009-2018 [38,149,196] zauważyć można dynamiczny wzrost zainteresowania metodami uczenia maszynowego. Metody, które odnotowały największy wzrost wykorzystania należały do logiki rozmytej, algorytmów ewolucyjnych oraz sieci neuronowych.

Zgodnie z ogólnym trendem w rozwoju technik sztucznej inteligencji jako dziedziny nauki, metody te niemal błyskawicznie zostały zaadaptowane do rozwiązywania problemów inżynierii budowlanej. Płytkie sieci neuronowe odnalazły zastosowanie w budownictwie już w latach 80-tych, kiedy wykorzystywane były do przeprowadzania obliczeń na komputerach klasy PET do rozwiązywania problemów, co do których nie istniały opracowane reguły lub heurystyka [197], jak na przykład zarządzanie placem budowy.

W miarę rozwoju algorytmów, do roku 2005 znalazły zastosowanie także między innymi w projektowaniu materiałów budowlanych [196], projektowaniu geotechnicznym [198] oraz wczesnych metodach oceny stanu technicznego konstrukcji na podstawie badania jej odpowiedzi dynamicznej [199]. Po tym okresie sztuczne sieci neuronowe wykorzystywane były także między innymi przy testach nieniszczących materiałów budowlanych oraz projektowaniu funkcji eksperckich wspomagających proces zarządzania obiektami budowlanymi [148,200,201]. Po opracowaniu bardziej złożonych architektur sztucznych sieci neuronowych na początku XXI wieku, takich jak opisywane wcześniej sieci konwolucyjne lub LSTM (ang. Long-Short Term Memory), dające możliwość analizy między innymi szeregów czasowych, to na ich wykorzystaniu skupił się ciężar prowadzenia badań w zakresie inżynierii budowlanej.

Jako jeden z najnowszych kierunków wykorzystania sieci neuronowych w inżynierii budowlanej wymienić można wspomagane przy pomocy algorytmów neuronowych symulacje fizyczne. Wykorzystują one sieci LSTM oraz VIN (ang. Visual Interaction Network) w połączeniu z wydajnymi procesorami graficznymi do przewidywania kolejnych kroków symulacji na podstawie treningu z wykorzystaniem profesjonalnego oprogramowania. Sieci te cechują się wysoką dokładnością symulacji, sięgającą 95% zgodności z oprogramowaniem profesjonalnym oraz mniejszemu od niego zapotrzebowania na moc obliczeniową maszyn roboczych. Daje to możliwość prowadzenia dokładnych i jednocześnie szybkich obliczeń fizycznych bez konieczności ponoszenia wysokich kosztów wykorzystywanych klasterów obliczeniowych [202,203].

3.5.1. WIZJA KOMPUTEROWA I SZTUCZNA INTELIGENCJA W INŻYNIERII BUDOWLANEJ

Metody obecne w inżynierii budowlanej wykorzystujące klasyczne metody wizji komputerowej wymienione i opisane zostały wraz z ich ograniczeniami w sekcji 2.3. Ograniczenia uniemożliwiające ich zastosowanie w zagadnieniach praktycznych inżynierii, doczekały się częściowego rozwiązania dzięki wykorzystaniu metod sztucznej inteligencji. Dzięki niej, metody oparte na wizji komputerowej mogą być dokładniejsze, mniej zależne od typu obserwowanej powierzchni oraz naświetlenia obrazu, a także mogą generalizować wiedzę poza wstępnie opracowany w metodzie model [204].

Przykładami wczesnego zastosowania metod sztucznej inteligencji w połączeniu z wizją komputerową w inżynierii budowlanej są między innymi metody wykorzystujące maszyny wektorów nośnych, algorytm k-najbliższych sąsiadów oraz inteligentne wstępne przetwarzanie obrazów [205–207]. Mimo, że w momencie publikacji były to metody przełomowe, ich autorzy nadal borykali się z podobnymi problemami co klasyczna wizja komputerowa. Dopiero upowszechnienie się konwolucyjnych sieci neuronowych pozwoliło na wykorzystanie potencjału wizji komputerowej w detekcji uszkodzeń obiektów budowlanych, co poskutkowało dużą liczbą publikacji w dziedzinie utrzymania obiektów budowlanych [208–210]. Na bazie wymienionych prac powstały także systemy wykorzystujące ciągły monitoring wizyjny z kamer przemysłowych CCTV (ang. Closed-Circuit TeleVision), na przykład do monitorowania elementów kanalizacji [211].

Kolejnym krokiem w rozwoju dziedziny było wykorzystanie bardziej rozbudowanych algorytmów, takich jak sieci w pełni konwolucyjne do segmentacji semantycznej lub instancyjnej obrazu, w celu ekstrakcji obszarów, na których zlokalizowano defekty [212,213]. Opisywane były także systemy połączone – klasyfikujące i jednocześnie segmentujące obraz na fragmenty zawierające defekt oraz nieuszkodzone, a także algorytmy segmentacyjne i klasyfikujące szeregi czasowe, wykorzystujące rekurencyjne sieci LSTM [151,214].

Jednak wszystkie wymienione powyżej metody, mimo bycia gotowymi do implementacji na szeroką skalę dzięki osiąganym przez siebie wysokim wynikom dokładności wskazań, nigdy nie zostały wprowadzone do powszechnego użytku na szczeblu zarządzania infrastrukturą mostową. Przesłanki, które częściowo mogą tłumaczyć zaniechanie w implementacji opisywanych algorytmów, zostały wymienione w sekcji 2.2.3.4. rozprawy. Żadna z metod nie spełnia wymienionych w sekcji 2.2.3.4. trzech podstawowych cech użytecznego narzędzia inżynierskiego. Metody te nie są łatwe w zastosowaniu dla osoby nieposiadającej gruntownej wiedzy w dziedzinie sztucznej inteligencji oraz nie istnieją gotowe narzędzia umożliwiające generowanie dla nich zbiorów uczących, co jest podstawowym warunkiem do ich wykorzystania. Z tego także powodu, nie mogą być zastosowane szybko. Sama ręczna generacja zbiorów treningowych jest poza zasięgiem możliwości lokalnych jednostek administrujących infrastrukturą drogową. Co więcej, z uwagi na duże zapotrzebowanie na

moc obliczeniową maszyn mogących uruchomić przedstawione algorytmy, wymagają one sprzętu specjalistycznego, który może okazać się zbyt drogi dla jednostek administracyjnych dysponujących ograniczonym budżetem.

Także sposób prezentacji niektórych z wymienionych prac uniemożliwia ich efektywne wdrożenie. W publikowanych pracach zwraca się uwagę na fakt braku dołączania do nich przez naukowców gotowych modeli wytrenowanych sieci oraz zbiorów uczących, przez co wyników ich pracy nie da się reprodukować ani w sposób rzetelny porównać z innymi dostępnymi rozwiązaniami [209].

Wszystkie z wymienionych powyżej problemów są możliwe do rozwiązania dzięki wykorzystaniu nowoczesnych technik wizji komputerowej przy budowie zbioru uczącego, treningu algorytmu oraz jego praktycznemu wdrożeniu. W kolejnych rozdziałach rozprawy zaprezentowane zostanie oryginalne narzędzie inżynierskie przewidziane do wykrywania uszkodzeń powierzchni betonowych. Dzięki zastosowaniu konwolucyjnych sieci neuronowych, proponowane rozwiązanie osiąga wysokie wyniki w zagadnieniu klasyfikacji obrazu. Jednocześnie wykorzystanie nowoczesnych technik budowy zbioru uczącego i treningu algorytmu umożliwia stosowanie go w dowolnym środowisku komputerowym. Zaprezentowana zostanie także technika łącząca metody SI i niestandardowej elektroniki w celu dalszej automatyzacji procesu inspekcji obiektów mostowych.

4. PROPOZYCJA METODY WSPOMAGAJĄCEJ WYKRYWANIE USZKODZEŃ POWIERZCHNI BETONOWYCH

Głównym celem prezentowanej rozprawy jest dostarczenie łatwego do wdrożenia, opartego na głębokim uczeniu narzędzia inżynierskiego do wykrywania uszkodzeń na powierzchniach betonowych oraz budowania zbiorów treningowych. Może być ono używane bezpośrednio przez inspektora mostowego w trakcie pracy na obiekcie, jak również do budowania własnych klasyfikatorów defektów nie ujętych w rozprawie. Zaproponowane narzędzie zapewnia skuteczność wykrywania drobnych defektów przewyższającą podejście manualne. Algorytmy użyte w rozprawie zostały zaimplementowane w postaci pakietu oprogramowania, który zapewnia możliwość modyfikacji komponentów do zarządzania i dostrajania klasyfikatorów zgodnie z określonymi przypadkami użycia, specyficznymi dla pracy danej jednostki zarządzającej infrastrukturą mostową. Wraz z tekstem rozprawy udostępnione zostało zbudowane oprogramowanie [215,216], a także przykładowy zbiór cech bazy danych do wykrywania rys o rozwartości do 0,2 mm, na której przeprowadzony był trening algorytmu [217]. Udostępnione zostały także wstępnie skompilowane pliki wykonywalne dla systemów Windows oraz Linux, a także sam klasyfikator. Publiczna dostępność oprogramowania i zbiorów danych opracowanych na potrzeby utrzymania infrastruktury ma kluczowe znaczenie dla wdrażania nowoczesnych metod w tym obszarze. Dodatkowym celem prezentowanej rozprawy jest także popularyzacja metod wizji komputerowej w połączeniu ze sztuczną inteligencją w inżynierii budowlanej.

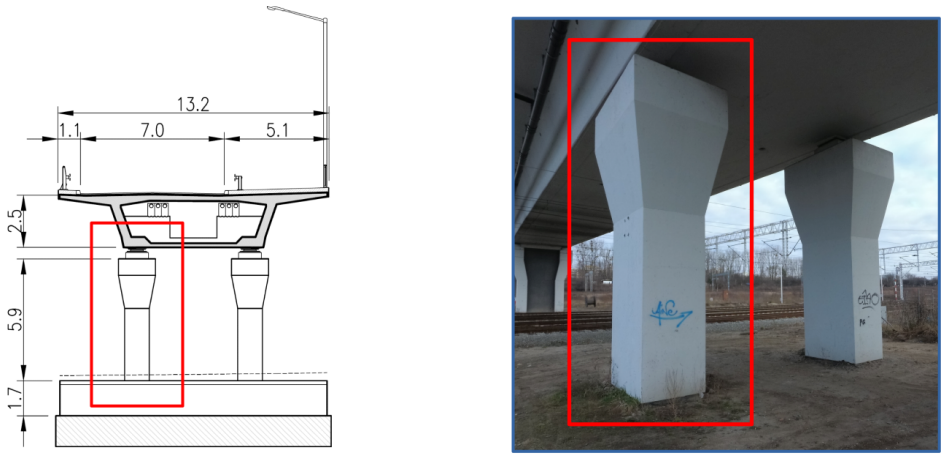
Jako defekt służący do testów prezentowanego podejścia wybrane zostały rysy o rozwartości poniżej 0,2 mm. Zostały one wybrane z uwagi na trudności związane z ich manualnym wykryciem przez nieuzbrojone oko oraz na ich powszechne występowanie w konstrukcjach żelbetowych. Są one także obecne w dziedzinie jako aktualne pole rozwoju metod automatycznej detekcji uszkodzeń, dzięki czemu możliwe jest porównanie prezentowanej metody z innymi, równoważnymi, opisywanymi w literaturze. Jednocześnie, prezentowane narzędzie inżynierskie może być wykorzystane do detekcji innych uszkodzeń powierzchniowych takich jak erozja czy wykwity korozyjne, w zależności od zbioru danych wykorzystanego do jego treningu.

W celu dowiedzenia zdolności proponowanego rozwiązania do generalizowania wiedzy poza zbiór danych, na którym było trenowane, do treningu wykorzystano hermetyczny zbiór danych pozyskany na jednym obiekcie. Jako obiekt treningowy wybrany został skrajny, południowy filar

estakady drogowej łączącej ulice Portową oraz Perseusza w Gliwicach (estakada Jana Heweliusza). Ta ponad 200 metrowa konstrukcja ze skrzynkowym dźwigarem głównym przebiega nad wielotorową linią kolejową. Jej filar został wybrany jako obiekt treningowy z uwagi na liczne zarysowania o niewielkiej rozwartości na całej swojej powierzchni. Lokalizacja obiektu na mapie została przedstawiona na Rys. 24, natomiast schemat rysunkowy i zdjęcie filaru na Rys. 25.

Rys. 24: Lokalizacja obiektu treningowego

Aby dowieść zdolności algorytmu do generalizowania wiedzy, zbiór testowy obejmował rysy o niewielkiej rozwartości oraz inne, o rozwartości dochodzącej do 1 mm. Zbiór testowy zawierał obrazy pochodzące z innych elementów wiaduktu Heweliusza, ale także obrazy pobrane na innych obiektach budowlanych w obrębie Gliwic. Obrazy pobierane były na powierzchniach o różnych charakterystykach i w różnych warunkach oświetlenia.



Rys. 25: Schemat rysunkowy i zdjęcie obiektu treningowego

4.1. Założenia projektowanego oprogramowania

Aby projektowane oprogramowanie spełniało opisane w sekcji 2.2.3.4. wymagania użytecznego narzędzia inżynierskiego, musi spełniać założenia rzutujące na sposób jego wykonania:

- możliwość uruchomienia narzędzia tak do treningu klasyfikatora jak i do jego wykorzystania na urządzeniu o niskiej mocy obliczeniowej,
- zdolność algorytmu do detekcji niewielkich uszkodzeń,
- łatwość w obsłudze treningu i przygotowywaniu zbioru uczącego,
- możliwość zastosowania algorytmów wspomagających trening,
- wybór odpowiedniej metryki algorytmu do oceny jego przydatności.

Ich wpływ na finalny projekt oprogramowania opisany został w następnych sekcjach.

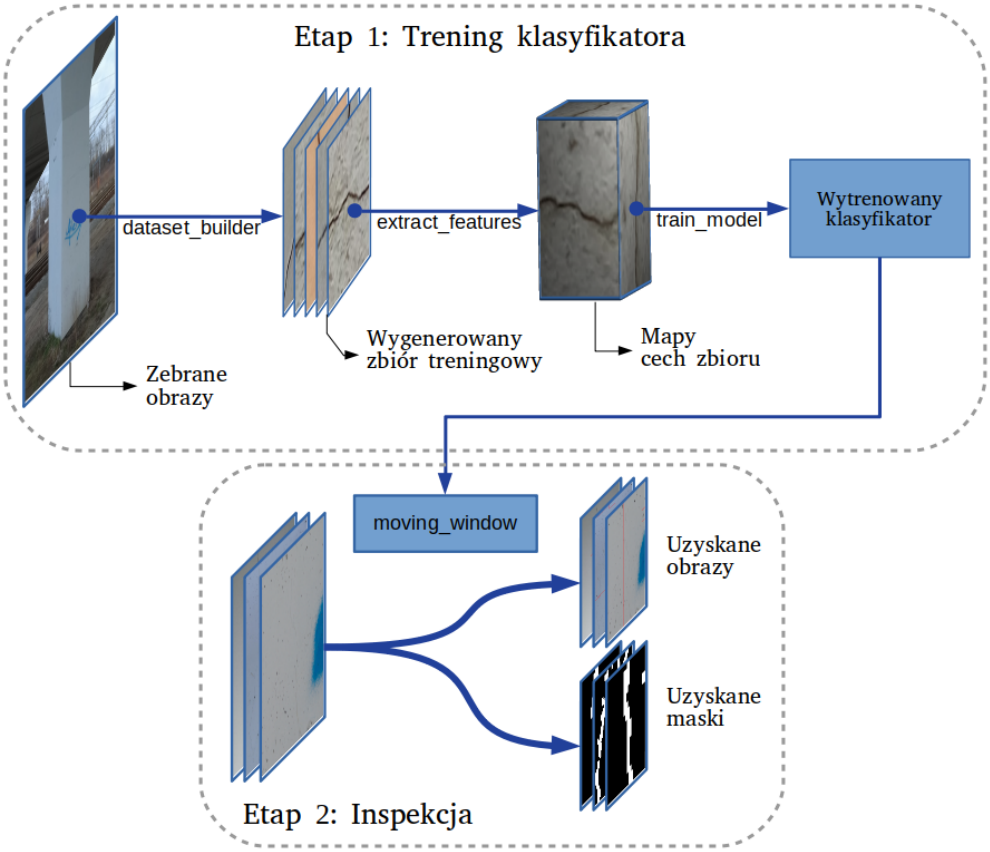
4.1.1. SIEĆ NEURONOWA JAKO DETEKTOR OBIEKTÓW NA OBRAZIE

Detektory obiektów na obrazie to algorytmy wykorzystujące zespoły konwolucyjnych sieci neuronowych (np. algorytm Faster R-CNN [218]) lub regresję krzywych ograniczających (np. algorytm YOLO [219]) do wskazywania lokalizacji obiektów na obrazie. Ich celem jest maksymalizacja wydajności detekcji w kontekście szybkości działania na urządzeniach mobilnych, w zastosowaniach IoT (ang. Internet of Things), a także monitoringu CCTV. Część z nich znalazła już zastosowanie w budownictwie jako detektory uszkodzeń, na przykład do wykrywania korozji stali konstrukcyjnej czy wad nawierzchni z wykorzystaniem bezzałogowych statków powietrznych BSP i narzędzi autorskich IoT [220,221]. Jednak pomimo zalet wynikających z niskiego opóźnienia ich działania nawet na mało wydajnych urządzeniach mobilnych, algorytmy te mają szereg wad wykluczających je z efektywnego i wszechstronnego zastosowania jako narzędzia inżynierskiego w pracy IM.

Podstawową wadą opisywanych rozwiązań jest sposób, w jaki osiągają dużą wydajność w analizie obrazu. W celu uzyskania wysokiej liczby analizowanych klatek na sekundę, powyższe algorytmy ograniczają liczbę możliwych obiektów wykrytych na obrazie i są domyślnie stosowane na obrazach o niskiej rozdzielczości, uniemożliwiającej detekcję niewielkich obiektów (jak na przykład rysy o małej rozwartości) [182]. Kolejną istotną przesłanką przemawiającą przeciw stosowaniu detektorów obiektów jest stosunkowo duża złożoność procesu przygotowania i etykietowania ich danych treningowych. Oprócz samej etykiety obiektu znajdującego się na obrazie, metody te wymagają w zbiorze uczącym także podania dokładnej jego lokalizacji. Mają też ograniczoną możliwość wykorzystania technik wspomagających ich trening.

Z tych względów zdecydowano się na zastosowanie metody poruszającego się okna (ang. sliding window), jak w [222] wraz z klasyfikatorem rozpoznającym zawartość semantyczną. Zakłada ona próbkowanie kolejnych kadrów pozyskanych z obrazu wejściowego, a następnie przeprowadzanie ich przez algorytm neuronowy w celu jego klasyfikacji. W ten sposób otrzymywana jest binarna maska zawierająca wszystkie obszary, na których zidentyfikowana została poszukiwana klasa obiektu.

Może ona następnie posłużyć do zarządzania obrazem wejściowym. Takie podejście rozwiązuje problem rozdzielczości obrazu wejściowego, a tym samym umożliwia wykrycie poszukiwanych rys o niewielkiej rozwartości. Co więcej, podejście to umożliwia wykorzystanie alternatywnych technik treningu algorytmu wraz z wybraną osobiście przez użytkownika architekturą konwolucyjnej sieci neuronowej. W dalszej części rozprawy, opisywany zbiór algorytmów włącznie z metodami budowy zbioru uczącego sieć neuronową, będzie występował pod nazwą KrakN.



Rys. 26: Etapy pracy ze zbiorem algorytmów KrakN

Ogólny opis pracy zbioru algorytmów KrakN, w tym metody wraz z ich kolejnością działania, przedstawiono na Rys. 26, gdzie oprócz etapów pracy, podano także nazwy konkretnych algorytmów przeprowadzających dane operacje. Należy zwrócić uwagę, że praca oprogramowania KrakN jest procesem dwuetapowym, podzielonym w celu zwiększenia efektywności programu oraz umożliwienia przeprowadzenia obliczeń w klastrach obliczeniowych w chmurze. Ponadto pierwsza, bardziej czasochłonna część algorytmu (trening klasyfikatorów) musi być wykonywana tylko raz na dany typ uszkodzenia lub gdy do klasyfikatora zostanie dodany nowy przypadek defektu. Uzyskane w ten sposób klasyfikatory mogą być następnie wykorzystywane przez wszystkich pracowników jednostki utrzymania obiektów mostowych bez konieczności dodatkowego treningu algorytmu lub posiadania wstępnych danych treningowych. Część druga, w której wykorzystywany jest wytrenowany wcześniej klasyfikator, może następnie być używana przy każdej inspekcji przeprowadzanej przez inspektora

mostowego w celu wsparcia jego pracy także bez dodatkowego treningu z pierwszego etapu algorytmu.

Porównanie wymienionego na początku sekcji podejścia opartego na detektorach obiektów z proponowanym w ramach zbioru algorytmów KrakN znajduje się w Tab. 4. Należy pamiętać, że KrakN nie narzuca ograniczeń rozdzielczości wykorzystywanych zdjęć, a ich rozmiar nie ma wpływu na możliwości ich analizy ani na dokładność wskazań. Co więcej, wykorzystując techniki wspomagania uczenia konwolucyjnej sieci neuronowej i automatyzując proces tworzenia treningowych zbiorów danych, znacząco poprawia się wydajność proponowanego rozwiązania.

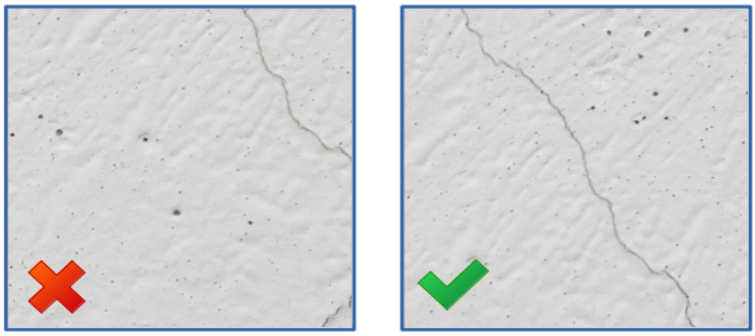
Tab. 4: Porównanie podejścia algorytmów KrakN do detektorów obiektów

	Detektory obiektów	**Algorytmy KrakN**
Rozdzielczość obrazu	Ograniczona możliwościami detektora	Równa rozdzielczości kamery
Wstępne zapotrzebowanie na dane treningowe	Wysokie zapotrzebowanie na dane, wymagane są etykiety obiektów wraz z ich pozycją na obrazie	Niskie zapotrzebowanie dzięki wspomaganiu procesu uczenia sieci, wymagane wyłącznie etykiety obiektów
Budowanie zbioru uczącego	Ręczne	Automatyczne, wspierane algorytmami KrakN
Hiperparametry treningu	Ustawiane ręcznie	Zarządzane algorytmami KrakN
Dokładność wskazań	Ograniczona możliwościami detektora	Ograniczona rozdzielczością kamery

Sekcja 3.4. przedstawia konwolucyjne sieci neuronowe jako algorytmy, które zostały opracowane głównie w celu klasyfikacji obrazów. Podstawową cechą tych algorytmów, która pozwoliła im osiągnąć bardzo dobre rezultaty w tym zagadnieniu, jest ich “świadomość” przestrzenna [185]. Oznacza ona, że podczas szkolenia sieć poznaje nie tylko charakterystyczne cechy wyglądu danego obiektu, ale także ich specyficzne położenie w relacji do siebie nawzajem na obrazie. Ponadto, w trakcie treningu sieci, dane te są utrwalane w wagach filtrów konwolucyjnych i z tego powodu nawet pozornie niewielka zmiana cech poszukiwanego na obrazie obiektu może poskutkować błędem wskazania sieci. Przykładem takiej zmiany cech może być nachylenie rysy na obrazie. Jeśli w zbiorze uczącym algorytmu dominowały obrazy rys poziomych, po treningu sieć będzie wykazywać mniejszą dokładność przy wykrywaniu rys o innym nachyleniu. Należy przy tym pamiętać, że CNN posiadają także zdolność do generalizowania wiedzy, więc o ile zbiór uczący można podzielić na różne klasy w zależności od pozycji, nachylenia czy częściowego zasłonięcia obiektu na obrazie, to najczęstszą praktyką jest budowa zbioru uczącego zawierającego wszystkie przypadki występowania poszukiwanego obiektu w zbalansowanej względem siebie liczbie obrazów.

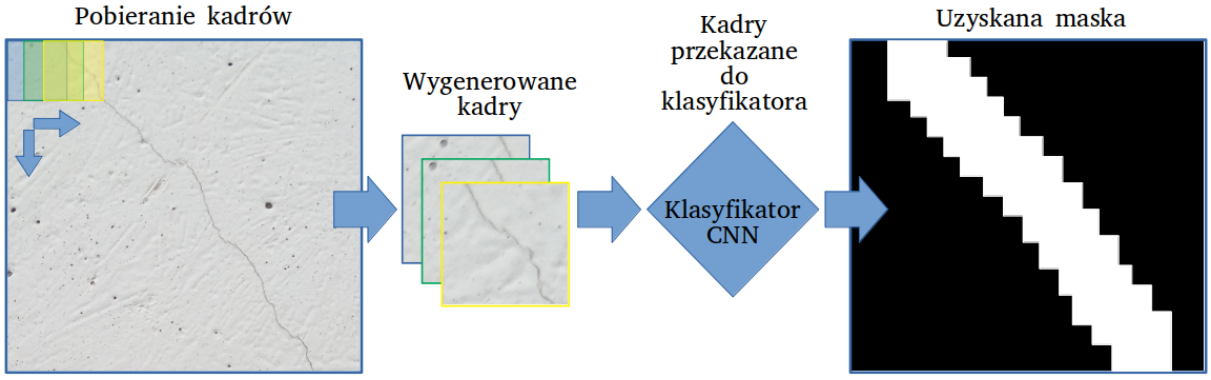
Powyższa charakterystyka ma duży wpływ na sposób reprezentacji danych, nad którymi pracuje algorytm neuronowy. W szczególności ma to znaczenie przy budowaniu zbioru danych. Należy bowiem zwrócić uwagę nie tylko na obecność danego obiektu na obrazie, ale także na jego

przestrzenne położenie, obroty i częściowe zasłonięcie. Z uwagi na tę konieczność, ogólna pozycja klasyfikowanego obiektu powinna być spójna we wszystkich punktach danych, a także przy zastosowaniu algorytmu podczas klasyfikacji tak, aby jak najwięcej cech poszukiwanego obiektu zawarta była w pojedynczym punkcie danych. Zasada ta została przedstawiona na Rys. 27, gdzie oba obrazy zawierają poszukiwany defekt, ale ten zawierający więcej cech uszkodzenia powinien być wykorzystany do uczenia algorytmu w celu uzyskania jego najlepszej skuteczności.



Rys. 27: Nieprawidłowe (lewa) i prawidłowe (prawa) położenie obiektu na obrazie

Opisywana wcześniej świadomość przestrzenna konwolucyjnych sieci neuronowych jest bardzo istotną cechą, w przypadku gdy są one wykorzystywane jako detektor obiektów przy użyciu techniki poruszającego się okna (Rys. 28). Aby zminimalizować liczbę przypadków, w których wybrany kadr obrazu pomija poszukiwany defekt skutkując fałszywie negatywną klasyfikacją, należy zastosować określone nakładanie się kadrów (ang. overlapping), uwzględnione jako parametr liczbowy z zakresu (0,0–1,0) w zbiorze algorytmów KrakN. Jednocześnie, mimo że nakładanie się obrazów poprawia dokładność klasyfikacji, znacznie zwiększa również ich liczbę przekazywaną do algorytmu klasyfikacyjnego, co z kolei wydatnie wydłuża czas pracy algorytmu. Z tego powodu, aby uzyskać zadowalającą dokładność przy jednoczesnym ograniczaniu całkowitego czasu obliczeń, wartość nakładania się okien należy utrzymywać w zakresie od 0,50 do 0,75 wielkości obrazu wejściowego. Wartość ta w pakiecie oprogramowania KrakN domyślnie wynosi 0,60 , lecz może być modyfikowana ręcznie przez użytkownika.



Rys. 28: Algorytm poruszającego się okna

Wynikiem działania algorytmu przesuwającego się okna jest zestaw sklasyfikowanych, nakładających się kadrów, które następnie wykorzystywane są do utworzenia maski zawierającej cały gabaryt uszkodzenia na początkowym obrazie. Maski takie mogą zostać użyte do wycinania defektów z obrazów wejściowych w celu dalszej analizy i katalogowania lub wzbogacania modeli obiektu o dodatkowe dane [131]. Opisywany w rozprawie pakiet algorytmów pozwala na uzyskanie zarówno wygenerowanych masek, ich współrzędnych odniesionych do lokalnych układów, jak i oryginalnych obrazów wzbogaconych o obwiednie zlokalizowanych na obrazach defektów do szybkiej oceny powierzchni betonu przez IM.

4.1.2. WYBÓR ARCHITEKTURY SIECI NEURONOWEJ

Jak wspomniano wcześniej (patrz s. 3.4.2.), każda z sieci neuronowych charakteryzuje się pewną konkretną architekturą, definiowaną jako określony układ warstw. Od architektury tej zależy nie tylko dokładność algorytmu, ale także jego zapotrzebowanie na moc obliczeniową oraz prędkość działania. Na przykład zastosowanie sieci sekwencyjnej, wykorzystującej 130 milionów parametrów wymaga około 495 MB pamięci operacyjnej przeznaczonej wyłącznie na przechowywanie wag filtrów sieci (przy użyciu 32 bitowych liczb typu float (ang. floating-point number)). Mając na uwadze powyższe, proces doboru architektury sieci wykorzystanej w rozprawie zakładał maksymalizację dokładności algorytmu przy jednoczesnym zachowaniu możliwości jego obsługi na maszynach roboczych nie posiadających wydajnych układów obliczeniowych. Stoi to jednocześnie w zgodzie ze wstępnymi założeniami dotyczącymi projektowanego narzędzia (możliwość zastosowania go w jednostkach o ograniczonym finansowaniu).

Przy doborze konkretnej architektury przetestowano cztery grupy architektur z wykorzystaniem komputera o typowych dla zastosowań biurowych parametrach, bez wykorzystania do obliczeń procesora graficznego (GPU). Maszyna obliczeniowa wyposażona była w procesor Intel J3455 o niskim poborze mocy oraz 4 GB pamięci RAM typu LPDDR-3 (typ pamięci RAM o niskim zapotrzebowaniu na energię). W trakcie eksperymentu pod uwagę brano następujące parametry testowanych algorytmów:

- domyślny rozmiar analizowanego obrazu,
- liczba parametrów sieci,
- dokładność wskazań na zbiorze danych ImageNet (predykcja określona jako trafna gdy poprawna odpowiedź znajdowała się w pierwszych pięciu najbardziej prawdopodobnych),
- test wydajności na maszynie obliczeniowej w średniej liczbie obrazów analizowanych na 1 sekundę obliczeń.

Tab. 5: Porównanie parametrów architektur sieci neuronowych

Architektura	Domyślne wymiary obrazu (px/px/px)	Liczba parametrów	Dokładność na zbiorze ImageNet	Test wydajności (obrazy/s)
LeNet	32/32/1*	60 tys.	~ 24,0 %	12,4
AlexNet	227/227/3	60 mln.	80,2 %	2,3
VGG16	224/224/3	138 mln.	91,2 %	1,2
ResNet-50	224/224/3	25 mln.	93,0 %	–**

*Sieć LeNet domyślnie akceptuje obrazy jednokanałowe

**Operacja równoległych obliczeń w architekturze ResNet nie jest obsługiwana przez niektóre starsze typy procesorów

Do testu wybrano przekrój architektur sieci neuronowych począwszy od płytkiej sieci sekwencyjnej LeNet, poprzez bardziej rozbudowane, głębokie algorytmy sekwencyjne, kończąc na sieci ResNet-50 z rozbudowaną mikroarchitekturą. Pierwsza z testowanych sieci – LeNet, domyślnie obsługuje jednokanałowe obrazy o rozdzielczości 32 na 32 piksele. Z tego powodu, obrazy ze zbioru ImageNet zostały poddane wstępnej obróbce poprzez konwersję do skali szarości oraz zmniejszeniu ich wymiarów ze średniej rozdzielczości 469 na 387 pikseli do wymiarów akceptowanych przez sieć. Poskutkowało to relatywnie niską dokładnością algorytmu, poniżej 25%, jako że w trakcie obróbki obrazów utracono znaczną ilość ich cech charakterystycznych. Ponadto, niska rozdzielczość wejściowa obrazu może potencjalnie utrudniać budowę zbioru uczącego w zastosowaniu praktycznym. Biorąc pod uwagę powyższe, architektura ta została wstępnie odrzucona. Mała liczba parametrów sieci pozwala jednak na bardzo szybkie obliczenia z wykorzystaniem procesora klasy biurowej.

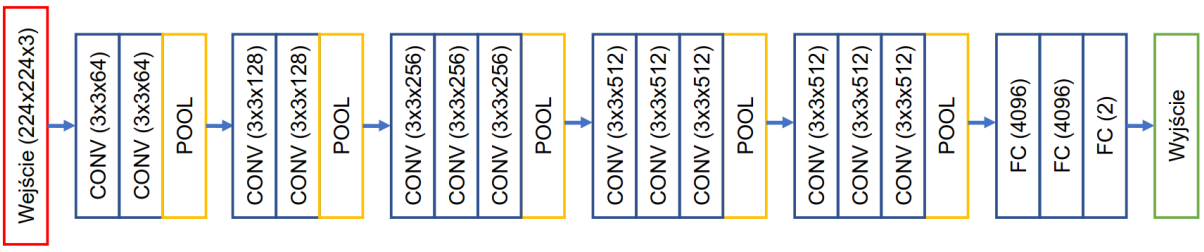
Kolejną testowaną architekturą była sieć AlexNet. Działa ona na obrazach o rozdzielczości pozwalającej na komfortową budowę zbioru uczącego, a jej ilość parametrów pozwala na osiągnięcie ponad 80% dokładności na zbiorze ImageNet. Pozostaje także relatywnie wydajna, pozwalając na analizę średnio ponad 2 obrazów na sekundę przy wykorzystaniu procesora biurowego. Podobną, lecz bardziej rozbudowaną architekturę posiada sieć VGG16. Dzięki znacznie głębszej budowie osiąga ponad 90% dokładność na zbiorze ImageNet, nie ograniczając przy tym znacznie wydajności obliczeń.

Ostatnią architekturą poddaną testowi była sieć ResNet-50. Wykorzystuje ona opisaną w sekcji 3.4.2. mikroarchitekturę wykonującą część obliczeń w sposób równoległy. Dzięki temu, przy posiadaniu mniejszej liczby parametrów osiąga lepsze wyniki w analizie zbioru ImageNet, przy potencjalnie krótszym czasie obliczeń niż klasyczne sieci sekwencyjne. Nie było jednak możliwe przeprowadznie testu wydajności, gdyż prowadzenie obliczeń równoległych nie jest obsługiwane przez instrukcje procesora maszyny testowej.

Z uwagi na wyniki przeprowadzonego testu, do budowy narzędzia wspomagającego pracę IM wybrana została sieć VGG16, oferująca zadowalającą dokładność odniesioną do szybkości działania algorytmu. Jej architektura w wariancie wykorzystanym w rozprawie, została przedstawiona na Rys.

29. Jednocześnie, przedstawiona w zbiorczej tabeli zawierającej wyniki eksperymentu (Tab. 5) dokładność dotyczy zróżnicowanego zbioru ImageNet rozróżniającego 1000 klas obiektów na obrazie, co może mieć istotny wpływ na dokładność wskazań algorytmu. W zastosowaniu jako narzędzie pracy IM, rozróżniające znacznie mniej klas obiektów, prezentowane rozwiązanie może osiągać wyższą dokładność. Jest to uwarunkowane zastosowaniem bardziej homogenicznego zbioru uczącego algorytm, w którym cechy poszczególnych obiektów będą bardziej się wyróżniały.

Jednocześnie z uwagi na zachowanie możliwości wykorzystania bardziej nowoczesnych maszyn obliczeniowych w zastosowaniu praktycznym, zbiór algorytmów KrakN pozostawiony został otwarty na połączenie go z innymi architekturami sieci neuronowych. Mimo domyślnego użycia sieci VGG16, możliwa jest ręczna zmiana bazowej części algorytmu, odpowiedzialnej za ekstrakcję cech charakterystycznych obrazu na dowolny inny, wytrenowany wcześniej algorytm. W tym celu wykorzystać można na przykład udostępniane wraz z biblioteką *Keras* modele sieci konwolucyjnych trenowanych na zbiorze ImageNet, przeznaczonych do ekstrakcji cech charakterystycznych obrazu (bez końcowej warstwy klasyfikującej). Pamiętać jednak należy, że wraz ze zmianą ekstraktora cech obrazu, należy od nowa poddać treningowi końcowy klasyfikator, zgodnie z opisem metod przedstawionym w sekcji 4.5. Tym samym pamiętać należy także o domyślnym rozmiarze obrazu akceptowanym przez wykorzystaną sieć.



Rys. 29: Architektura sieci VGG16 wykorzystanej w rozprawie

Dzięki umożliwieniu wykorzystania własnych ekstraktorów cech charakterystycznych obrazu w pakiecie KrakN, znacznie zwiększa się jego wszechstronność. Wykorzystanie płytszych architektur może umożliwić zastosowanie pakietu w połączeniu z technologią IoT w komputerach jednopłytkowych, na przykład do automatyzacji pomiarów wizualnych struktury ruchu na obiekcie. Natomiast stosowanie rozbudowanych mikroarchitektur wraz z wydajnymi układami procesorów graficznych może umożliwić analizowanie tekstur o wysokiej rozdzielczości pobranych w sposób automatyczny, na przykład przy pomocy urządzeń UAV, do detekcji i rozróżniania wielu pozornie podobnych do siebie uszkodzeń. Dzięki opisanym scenariuszom wykorzystania, zbiór algorytmów KrakN może zostać stosowany na różnych szczeblach zarządzania obiektami mostowymi, do wielu zadań, w których analizowane są dane wizualne.

4.1.3. ALTERNATYWNE METODY UCZENIA SIECI NEURONOWYCH

W sekcji 3.4.3. opisano pokrótce trening sieci neuronowych. Oprócz wyróżnionego tam problemu nadmiernego i niedostatecznego dopasowania sieci w wyniku treningu, występują jednak także inne trudności związane z nauką algorytmów neuronowych, wpływające na ich finalną dokładność. Zaliczyć można do nich między innymi wysokie zapotrzebowanie na zasoby obliczeniowe, czas trwania treningu, zanikający gradient czy inicjalizację wag w warstwach sieci.

W tej sekcji przedstawione zostaną dwie alternatywne metody treningu sieci neuronowej – f*ine tuning* oraz t*ransfer learning*. Opierają się one na częściowym wykorzystaniu wag pozyskanych w trakcie treningu sieci na osobnym zbiorze treningowym do wspomagania treningu na nowym zbiorze lub jako gotowy ekstraktor cech charakterystycznych obrazu wymagający jedynie końcowej warstwy klasyfikatora. Metody te zakładają transfer wiedzy między różnymi zagadnieniami. Oznacza to, że niezależnie od konkretnych obiektów poszukiwanych na obrazie, ich ogólna charakterystyka będzie podobna we wszystkich zbiorach danych (obecność krawędzi, różnice w kolorach, generalny kształt obiektów itp.), a wyodrębnione cechy w części konwolucyjnej sieci będą w pewnym stopniu uniwersalne między różnymi rozpoznawanymi obiektami. W ten sposób pozwalają na częściowe ominięcie najbardziej istotnych problemów z punktu widzenia zastosowania proponowanego rozwiązania w praktyce, obniżając uciążliwość treningu algorytmu od wag losowych, a także zmniejszając liczbę parametrów treningu. Tym samym zmniejszają zapotrzebowanie na moc obliczeniową maszyny obliczeniowej wykorzystującej algorytm w zagadnieniu klasyfikacji.

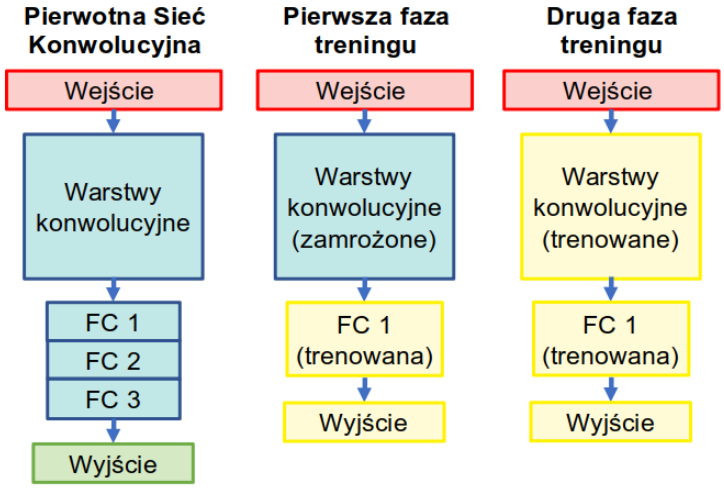
W końcowej części sekcji wybrana zostanie najbardziej praktyczna dla omawianego zagadnienia metoda, użyta następnie w części praktycznej rozprawy do budowy zbioru algorytmów KrakN.

4.1.3.1. FINE TUNING

Pierwszą z metod wspomagających uczenie sieci neuronowych jest technika dostrajania sieci (ang. fine tuning). Zakłada ona wykorzystanie całej architektury sieci konwolucyjnej wytrenowanej na obszernym, zewnętrznym zbiorze poprawnie etykietowanych danych jako bazy do treningu algorytmu na osobnym zbiorze. W pierwszej kolejności, z algorytmu usuwane są ostatnie warstwy w pełni połączone wraz z warstwą klasyfikatora Softmax. Następnie zastępowane są taką samą bądź mniejszą liczbą warstw w pełni połączonych i warstwą klasyfikującą. Warstwy te inicjalizowane są wagami w sposób losowy.

W ten sposób zbudowana nowa sieć, trenowana jest w dwóch fazach. W pierwszej, trening obejmuje wyłącznie nowo dodane warstwy inicjalizowane losowo, tak aby w trakcie treningu nie utracono filtrów z wcześniejszych warstw, które już są w stanie ekstrahować złożone cechy obrazu.

Druga faza treningu przeprowadzana jest na całej głębokości sieci. Podczas niej dostrajane są także warstwy pierwotnie zamrożone. Opisane fazy treningu różnią się od siebie długością i stosowanymi w ich trakcie parametrami. Ich długość pozostaje zazwyczaj w stosunku 1:3, gdzie dłużej poddawana jest treningowi cała sieć. Różnicowany jest także współczynnik uczenia się sieci. W pierwszej części treningu pozostaje on relatywnie wysoki tak, aby nowo dodane elementy algorytmu w jak najszybszy sposób dopasowały się wagami do jej istniejącej części. Następnie współczynnik ten jest obniżany do poziomu, w którym wagi całej sieci aktualizowane są już tylko w niewielkim stopniu, a algorytm poszukuje globalnego minimum. Proces tworzenia sieci konwolucyjnej uczonej przy pomocy dostrajania przedstawiony został na Rys. 30.



Rys. 30: Trening sieci neuronowej przy pomocy dostrajania [249]

Podejście to jednak nie pozostaje bez wad. Najpoważniejszą z nich jest konieczność w dalszym ciągu trenowania całej głębokości sieci neuronowej. Zatem mimo że algorytm może być trenowany przez mniejszą liczbę epok, nadal wymaga takich samych zasobów obliczeniowych co sieć trenowana od losowej inicjalizacji. W podobnym stopniu co przy treningu w sposób klasyczny zarządzać należy także parametrami treningu sieci, a to wymaga doświadczenia od osoby przeprowadzającej trening. Ponadto pojawiają się dodatkowe etapy treningu, którymi należy zarządzać – aktywowanie całej głębokości algorytmu po zadanej ilości epok treningu i krokowa zmiana wielkości parametru współczynniku uczenia.

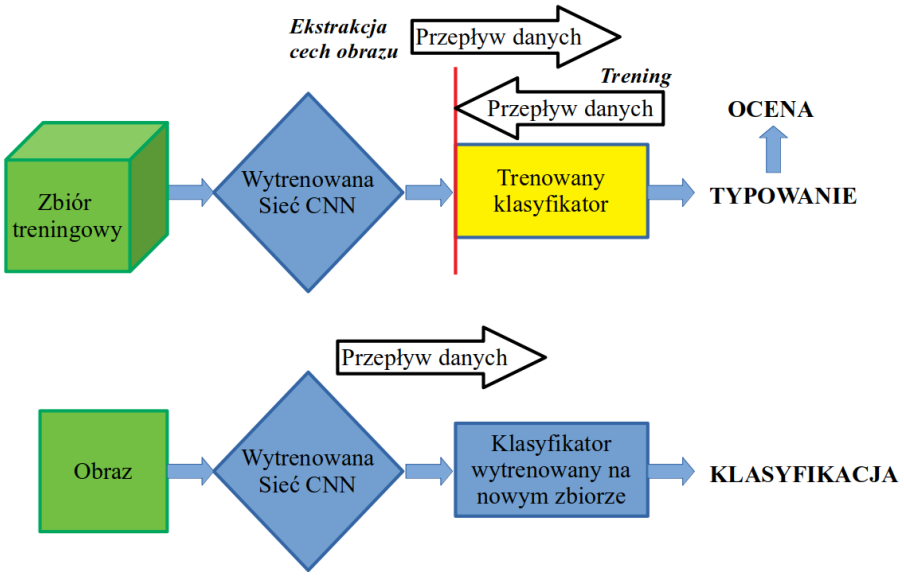
4.1.3.2. TRANSFER LEARNING

Drugą z metod wspomnianą we wstępie sekcji jest *transfer learning* (z uwagi na brak adekwatnego polskiego odpowiednika będzie używana nazwa transfer learning zamiennie z jej skrótem: TL). Podobnie jak w przypadku metody dostrajania sieci, TL wykorzystuje częściowo wagi, które algorytm nabył w trakcie treningu na osobnym zbiorze danych. Trening sieci z wykorzystaniem tej metody także rozpoczyna się od usunięcia z jej pierwotnej architektury ostatnich warstw w pełni

połączonych oraz klasyfikującej. Tym razem jednak nie są one zastępowane przez równorzędne warstwy sieci neuronowej, a najczęściej przez pojedynczy klasyfikator wykorzystujący funkcję Softmax, trenowany na danych wyjściowych z ostatniej warstwy konwolucyjnej.

W przypadku zastosowania TL, pierwotna sieć nie jest modyfikowana i służy wyłącznie jako ekstraktor cech charakterystycznych obrazu. Niesie to ze sobą konsekwencje w postaci możliwości efektywnego klasyfikowania jedynie tych obiektów, które wykazują podobne cechy (np. obecność linii) co obiekty obecne w pierwotnym zbiorze danych. Jest zatem szczególnie istotne, aby sieć używana jako ekstraktor cech obrazu, trenowana była na zbiorach danych obfitych w obiekty zawierające wiele cech charakterystycznych. Na przykład sieć uczona pierwotnie rozpoznawania elementów rozbudowanego zbioru danych zawierającego pojazdy, ludzi i inne obiekty złożone geometrycznie, może być wykorzystana w zagadnieniu rozpoznawania kształtów liści roślin (przykład jednolitego zbioru danych). Wykorzystanie jej w sytuacji odwrotnej nie będzie już jednak możliwe. Sposób treningu i wykorzystania sieci konwolucyjnej uczonej z wykorzystaniem metody TL został przestawiony na Rys. 31.

Poza wspomnianą wcześniej istotną wadą metody TL, posiada ona liczne zalety, takie jak brak konieczności zarządzania hiperparametrami opisującymi przebieg treningu sieci, krótszy czas treningu i mniejsze zapotrzebowanie na dane wejściowe. Ponadto, dobór parametrów końcowego klasyfikatora opartego o regresję logistyczną można w dużej mierze zautomatyzować, a jego wielkość będzie zdecydowanie mniejsza niż w przypadku w pełni połączonych warstw neuronowych. Ważną zaletą tego podejścia jest także duża dostępność gotowych sieci neuronowych, wytrenowanych wcześniej na rozbudowanych zbiorach danych.



Rys. 31: Trening (góra) i wykorzystanie (dół) sieci uczonej poprzez TL [245]

4.1.3.3. WYBÓR METODY TRENINGU ALGORYTMU

Aby ograniczyć pracochłonność proponowanej w rozprawie metody oraz zwiększyć przystępność narzędzia wykorzystywanego do treningu klasyfikatora przez inspektora mostowego, rozważone zostały różne scenariusze praktyki budowy algorytmu z wykorzystaniem opisanych w poprzednich sekcjach metod wspomagania treningu sieci. Uwzględnione zostały takie czynniki jak dostępność danych treningowych oraz ich zróżnicowanie względem oryginalnego zbioru danych, na których przeprowadzono trening podstawowej sieci konwolucyjnej. Scenariusze budowy algorytmu zostały przedstawione w Tab. 6. Opierają się na ogólnym opisie wytycznych treningu sieci konwolucyjnych przedstawionym w [12] i należy traktować je w sposób zgrubny. Ocena stopnia podobieństwa zbiorów danych oraz tego jaką ilość punktów danych w zbiorze można określić jako „dużą", zależy od analizowanego zagadnienia, uzależnionego od konkretnego typu uszkodzenia, które wykrywać ma algorytm.

W proponowanym w rozprawie zbiorze algorytmów KrakN założone zostało wykorzystanie architektur trenowanych pierwotnie na bogatym w cechy charakterystyczne zbiorze ImageNet [223]. Z tego powodu możliwe będzie wykorzystanie metod wspomagających trening algorytmu z wykorzystaniem tak dostrajania sieci jak i TL. Jednocześnie z uwagi na problem związany z brakiem obszernych, ogólnodostępnych zbiorów danych zawierających uszkodzenia obiektów mostowych, należy liczyć się z tym, że każda z jednostek zarządzających przeprowadzi proces kolekcji danych do zbioru samodzielnie. Jest to jednak proces długotrwały i pracochłonny, a obecnie dostępne dane w formie zdjęć cyfrowych mogą okazać się niewystarczającej jakości.

Tab. 6: Scenariusze rekomendowanych sposobów treningu sieci neuronowej

	Bogaty w cechy charakterystyczne oryginalny zbiór danych	**Oryginalny zbiór danych z obiektami o niewielu cechach charakterystycznych**
Mała ilość danych treningowych	Możliwe wykorzystanie TL	Rekomendowany trening sieci od wag losowych
Duża ilość danych treningowych	Możliwe wykorzystanie dostrajania końcowych warstw sieci	Rekomendowany trening sieci od wag losowych lub dostrajanie wszystkich warstw sieci

Z uwagi na powyższe, w opracowanym na potrzeby rozprawy zbiorze algorytmów, do wspomagania procesu uczenia sieci konwolucyjnej, wykorzystana została metoda TL. Jednocześnie, mimo wstępnego założenia wykorzystania sieci bazowej trenowanej na zbiorze ImageNet, algorytmy KrakN pozwalają na zastosowanie dowolnie trenowanej sieci jako ekstraktora cech charakterystycznych. W miarę rozpowszechniania się technik SI w dziedzinie inżynierii budowlanej, bazowa sieć będzie mogła zostać zastąpiona przez klasyfikator pierwotnie ekstrahujący dane ze zbiorów zawierających uszkodzenia obiektów mostowych. Pozwoli to jednocześnie zachować ciągłość stosowania prezentowanych w rozprawie metod oraz podnieść ich dokładność w miarę

upowszechniania się stosowania klasyfikacji obrazu przy pomocy SI w jednostkach zarządzających infrastrukturą mostową.

4.1.4. METODA OCENY PROPONOWANEGO ROZWIĄZANIA

Aby ocenić możliwość stosowania proponowanych metod w praktyce utrzymania betonowych obiektów mostowych, muszą one zostać ocenione z uwagi na dokładność ich wskazań w porównaniu do podejścia tradycyjnego oraz innych, obecnych w aktualnej literaturze sposobów detekcji uszkodzeń. W tym celu wykorzystane zostały metryki powszechnie wykorzystywane w ocenie wyników testów (na przykład medycznych, testów dokładności algorytmów predykcji) obliczane na podstawie tablicy pomyłek (ang. confusion matrix – macierz błędów) [224]. Tablicą pomyłek jest macierz zawierająca liczby wskazań przyporządkowane do pozytywnej i negatywnej etykiety, która umożliwia ocenę jakości klasyfikacji binarnej. Jej układ wraz z wartościami, które zawiera oraz ich skrótami występującymi w literaturze został przedstawiony w Tab. 7.

Tab. 7: Układ i wartości występujące w macierzy pomyłek

		Klasa rzeczywista	
		Klasa pozytywna	**Klasa negatywna**
Wyniki predykcji	**Wskazanie pozytywne**	Predykcja prawdziwie pozytywna (TP)	Predykcja fałszywie pozytywna (FP)
	Wskazanie negatywne	Predykcja fałszywie negatywna (FN)	Predykcja prawdziwie negatywna (TN)
Całkowita ilość próbek		Liczba próbek (TOT)	

Z wykorzystaniem wartości zawartych w macierzy pomyłek obliczyć można między innymi następujące, podstawowe metryki opisujące wytrenowany klasyfikator:

- dokładność (ang. accuracy),
- czułość (ang. recall),
- precyzja (ang. precision),
- zbalansowana dokładność (ang. balanced accuracy).

Są one obliczane z wykorzystaniem poniższych wzorów:

$$Dokładność = \frac{TP+TN}{TOT} \quad , \tag{6}$$

$$Czułość = \frac{TP}{TP+FN} \quad , \tag{7}$$

$$Precyzja = \frac{TP}{TP+FP} \quad , \tag{8}$$

$$Zbalansowana\,dokładność = \frac{Czułość + Precyzja}{2} \quad . \tag{9}$$

Metryki te wykorzystywane są zamiennie, w zależności od analizowanego zagadnienia. Najczęściej używana dokładność daje ogólne informacje na temat sprawności klasyfikatora, nie uwzględnia ona niezrównoważonego z uwagi na liczbę klas zbioru wskazań. Może zatem dawać mylne przekonanie o dokładności stosowanego rozwiązania, gdy widoczna jest nadreprezentacja pojedynczej klasy, jak w następującym przykładzie pokazanym w Tab. 8:

Tab. 8: Przykład macierzy błędu dla niezrównoważonych klas predykcji oraz obliczone metryki

	Klasa pozytywna	**Klasa negatywna**
Wskazanie pozytywne	5	15
Wskazanie negatywne	100	1000
	1120	

$$Dokładność = \frac{TP+TN}{TOT} = \frac{5+1000}{1120} = 90\% \quad ,$$

$$Czułość = \frac{TP}{TP+FN} = \frac{5}{5+100} = 5\% \quad ,$$

$$Precyzja = \frac{TP}{TP+FP} = \frac{5}{5+15} = 25\% \quad ,$$

$$Zbalansowana\,dokładność = \frac{Czułosć+Precyzja}{2} = \frac{5+25}{2} = 15\% \quad .$$

W powyższym przykładzie klasyfikator osiąga dokładność na poziomie 90% co pozornie wydaje się być doskonałym wynikiem. Widać jednak, że metryka ta nie uwzględnia faktycznego udziału wskazań prawdziwie pozytywnych (TP) w stosunku do fałszywie negatywnych (FN). Udział ten jest jednak kluczowy dla zagadnień w których wskazanie fałszywie negatywne może pociągać za sobą poważne konsekwencje, takich jak na przykład testy chorób zakaźnych lub omawiane w rozprawie zagadnienie wczesnej detekcji uszkodzenia na obiekcie mostowym.

Z tego powodu, w ocenie klasyfikatorów projektowanych do wykrywania konkretnego typu zdarzenia powszechnie stosuje się metryki pochodne, takie jak czułość i precyzja. Dla zaprezentowanego przykładu wynoszą one kolejno poniżej 5% oraz 25%, natomiast metryka zbalansowanej dokładności wynosi około 15%.

W rozprawie wykorzystane zostały dwie metryki. Pierwsza z nich to dokładność, użyta w celu oszacowania ogólnej sprawności z jaką opracowane rozwiązanie klasyfikuje dane wejściowe. Druga to czułość, gdyż ma ona kluczową wartość w przypadku nadreprezentacji klasyfikacji fałszywie negatywnych. Mają one bowiem największy wpływ negatywny na bezpieczeństwo ocenianej konstrukcji. Jednocześnie, z wyżej wymienionych powodów, za metrykę o większej wadze uznana została czułość.

Kolejnym istotnym aspektem związanym z oceną dokładności proponowanego rozwiązania są dane ewaluacyjne, przy pomocy których określane będą opisane metryki. Proponowana w rozprawie metoda wspomagania pracy IM zakłada wykorzystanie ograniczonych ilościowo danych do treningu algorytmu. Aby dodatkowo zweryfikować zdolność proponowanej metody do generalizowania wiedzy pozyskanej w trakcie treningu, wykorzystany zbiór treningowy jest homogeniczny i pochodzi z pojedynczej konstrukcji budowlanej. Założenie to nie obejmuje jednak danych, na których określone zostały metryki rozwiązania, jako że zbiór ewaluacyjny nie może być jednocześnie podzbiorem zbioru treningowego. Aby otrzymać rzetelne wyniki, sieć neuronowa nie może być oceniana przy pomocy punktów danych, które analizowała w trakcie treningu. Z tego powodu, do oceny wydajności proponowanego rozwiązania wykorzystane zostały tak dane zebrane na tym samym obiekcie testowym co dane treningowe (jednak nie użyte następnie do treningu) jak i pozyskane na innych obiektach. W sumie do ewaluacji zbioru algorytmów KrakN oraz porównania go z obecnymi w aktualnej literaturze innymi metodami wykorzystano ponad 3 tysiące punktów danych.

Jednocześnie nie jest poddawany ocenie sposób pobierania danych w trakcie przeglądu w przypadku zastosowania praktycznego. W celu uniezależnienia wyników obliczeń metryk proponowanego rozwiązania oraz rozwiązań występujących w literaturze, przyjęte zostało założenie, że proces pobierania danych jest zautomatyzowany (na przykład przy pomocy pojazdów BSP). Ma to istotne znaczenie dla oceny rozwiązania, gdyż w przypadku pobierania danych ręcznie, błąd algorytmu potęgowany byłby przez błąd osoby pobierającej dane. Założenie to pozwala także na porównanie oceny przedstawionego rozwiązania do danych literaturowych opisujących dokładność manualnego przeglądu wykonywanego przez IM lub metod ręcznych, wykorzystujących elementy wizji komputerowej.

4.2. ŚRODOWISKO PROGRAMISTYCZNE I WYKORZYSTANE NARZĘDZIA

Pakiet oprogramowania KrakN zbudowany na potrzeby rozprawy został opracowany przy użyciu ogólnodostępnego środowiska programistycznego o otwartym kodzie źródłowym. Wraz z wykorzystanymi bibliotekami podzielonymi na zagadnienia w których zostały użyte i numerami ich wersji zostało ono przedstawione w Tab. 9. Dzięki dokładnemu opisowi wykorzystanych narzędzi możliwe jest odtworzenie uzyskanych w rozprawie wyników na dowolnej maszynie obliczeniowej.

Oprogramowanie, biblioteki i wszystkie metody wykorzystane w pracy pozostają opatrzone licencją *open source* (otwarte, bezpłatne oprogramowanie) lub zostały opracowane na potrzeby rozprawy. Ponadto, algorytmy opisane w rozprawie zostały opracowane i wstępnie przetestowane na darmowym systemie Ubuntu 18.04 LTS GNU/Linux. Mimo, że biblioteki mogą się różnić w zależności od wykorzystywanego systemu operacyjnego, poddane zostały także testom w innych, komercyjnych środowiskach, w trakcie których ich działanie zostało określonej jako poprawne. Odpowiednia wersja pakietu oprogramowania w postaci wykonywalnych notatników języka Python

znajduje się w dołączonym do rozprawy repozytorium [216]. Zawiera ono także szczegółowe instrukcje dotyczące instalacji wymaganych dodatkowych bibliotek, a także pliki wykonywalne Windows.

Możliwe jest również wykorzystanie pakietu KrakN w środowiskach obliczeń online w chmurze, takich jak Google Colab, aby zminimalizować zapotrzebowanie na moc obliczeniową dostępną na maszynie lokalnej. Należy również zauważyć, że dzięki przystosowaniu opisywanego w rozprawie rozwiązania do środowisk przeprowadzających obliczenia online zmniejsza się dodatkowo pracochłonność wdrożenia rozwiązania związana z konfiguracją środowiska lokalnego.

Tab. 9: Wykorzystane w rozprawie środowisko programistyczne i biblioteki

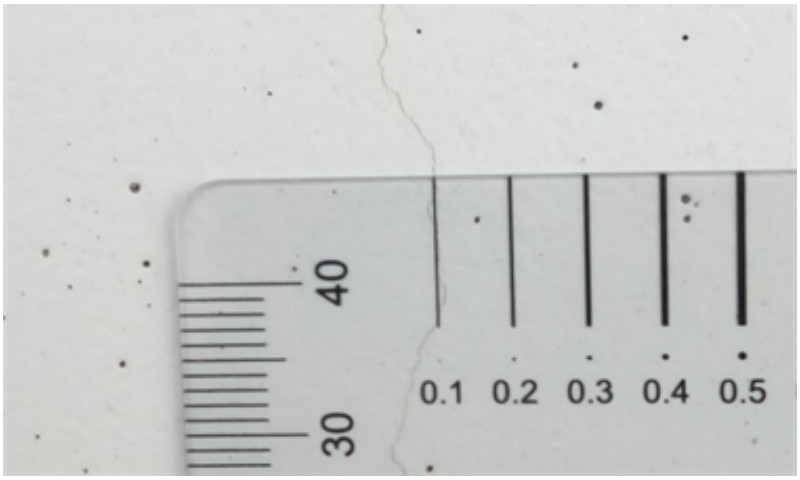
Środowisko programistyczne	Machine Learning	Obliczenia, zarządzanie macierzami	Narzędzia, zarządzanie bazami danych	Zarządzanie systemem operacyjnym
Python 3.6	Tensorflow 1.12 Keras 2.2 Scikit-learn 0.22	Numpy 1.16 OpenCV 4.0 Random	H5py 2.9 Pickle Progressbar 2.5	Os Imutils 0.5 PyGame 1.9

4.3. KOLEKCJA I OCENA DANYCH TRENINGOWYCH

W Internecie dostępnych jest wiele bezpłatnych zbiorów danych, które wykorzystać można do treningu algorytmu neuronowego. Jednym z nich jest zbiór danych zawierający obrazy pęknięć i zarysowań SDNET2018 [183]. Zbiory te jednak nie skupiają się na jednym typie lub zakresie rozwartości, co często może prowadzić do błędnego etykietowania określonego rodzaju defektu w świetle lokalnych przepisów. Na przykład wymieniony powyżej zbiór danych, mimo że według opisu zawiera obrazy rys na powierzchni betonu, przedstawia jako zarysowania ubytki o widocznie przerwanej ciągłości materiału, które mogłyby zostać określone jako pęknięcia [201]. Co więcej, detekcja uszkodzeń o tym rozwarciu decydowałaby już głównie o stanie awaryjnym elementu konstrukcji i miałaby ograniczoną przydatność jako narzędzie IM. Problem ten został już wyraźnie podkreślony w pracach naukowych [225], gdzie zauważono duże rozbieżności między deklarowaną a faktyczną dokładnością algorytmów SI do wykrywania uszkodzeń na powierzchni betonu. Poprzez fakt treningu algorytmów na bardzo wyraźnych uszkodzeniach o dużej rozwartości osiągane są pozornie wysokie dokładności algorytmów, które jednak nie rozpoznają kształtu lub charakterystycznych cech uszkodzenia, a działają w swej zasadzie jako detektory anomalii na powierzchni betonu, klasyfikując zabrudzenia jako rysy w testach niezależnych. Z tego powodu, mając na względzie wytyczne obowiązujące w przeglądach infrastruktury w Polsce oraz rzetelność treningu klasyfikatora, na potrzeby rozprawy przeprowadzona została niezależna kolekcja zbioru danych. W pracy przedstawiony został także pełny proces gromadzenia i etykietowania obrazów.

Jako studium przypadku przy tworzeniu zbioru danych wykorzystano rozlegle zarysowany filar mostu (patrz p. 4.). Obraz wraz z pomiarem przykładowego uszkodzenia ze zbioru treningowego

przedstawiony został na Rys. 32. Szczególną uwagę należy zwrócić na pomiar rozwartości rysy. Uszkodzenia o tak małej rozwartości nie są wykrywane przez rozwiązania opisywane w aktualnej literaturze. Zdjęcia do bazy zostały wykonane aparatem cyfrowym Sony Alpha DSLR-A500 wyposażonym w obiektyw o ogniskowej 18-55 mm f / 3,5-5,6 w dobrych warunkach oświetlenia (wczesne godziny popołudniowe przy braku zachmurzenia) w odległości 20-30 cm od powierzchni filara. Odległość wykonywania zdjęć została dobrana tak, aby na obrazach uzyskać gęstość pikseli odpowiednią do detekcji uszkodzeń o wielkości poniżej 0,2 mm (uzyskano gęstość powyżej 10 pikseli na 1 mm obrazu). Wszystkie zdjęcia zostały wykonane ręcznie. W efekcie uzyskano ponad 900 zdjęć o rozdzielczości 4248 na 2850 pikseli, pokrywających większość powierzchni filara. W praktyce zastosowania opisywanych metod do inspekcji dużych obiektów mostowych, zdjęcia do zbioru danych lub samej inspekcji uzyskać można za pomocą pojazdu BSP [226] lub ciągników samojezdnych wyposażonych w kamery CCTV jak w [227], co dodatkowo ograniczałoby pracochłonność zadania. Jednocześnie zwiększałoby to możliwości dotarcia do miejsc trudno dostępnych w trakcie inspekcji.

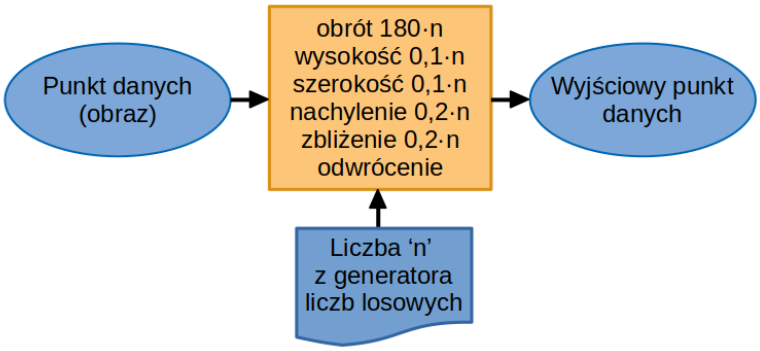


Rys. 32: Pomiar przykładowego uszkodzenia zawartego w zbiorze treningowym

Należy także podkreślić, że dzięki zdolności sieci konwolucyjnych do generalizowania wiedzy, w trakcie inspekcji użyć można kamery o innych parametrach niż podczas zbierania danych treningowych. Zależność ta jest dodatkowo poddana weryfikacji w trakcie ewaluacji algorytmu. Część danych ewaluacyjnych pochodzi z kamery o zbliżonych parametrach, lecz większość z nich została zebrana przy pomocy innych aparatów oraz telefonów komórkowych. Dzięki temu możliwe jest wykorzystanie powszechnie dostępnych tanich kamer obecnych w smartfonach lub dronach przy braku dostępności aparatów wysokiej rozdzielczości w celu dalszego obniżenia kosztów wdrożenia proponowanego rozwiązania. Jedynym istotnym czynnikiem jest wyłącznie zachowanie podobnej gęstości pikseli poszukiwanego na obrazie uszkodzenia.

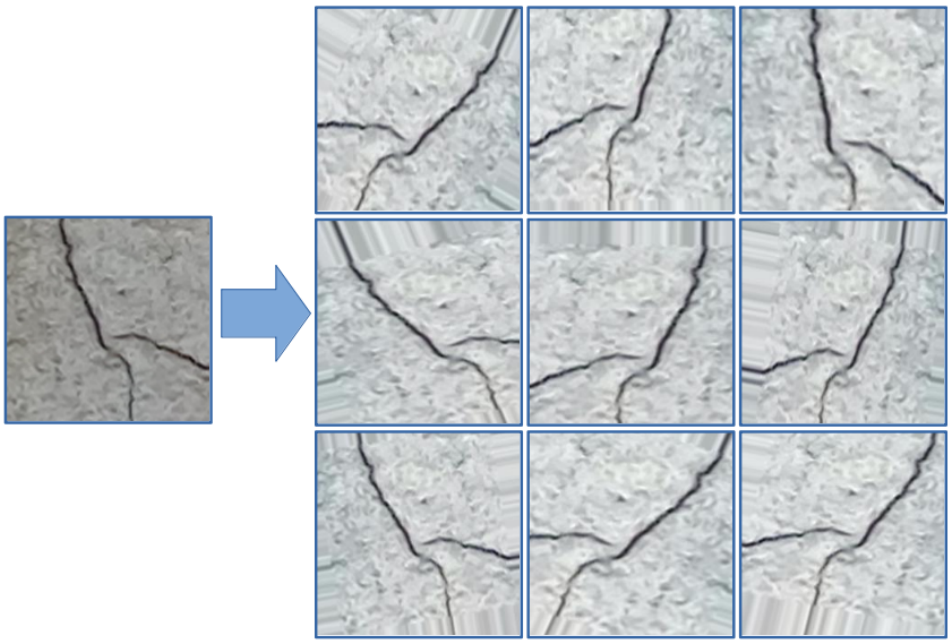
4.3.1. WZBOGACANIE DANYCH

W przypadku stosowania ograniczonych zbiorów danych, tak jak w zagadnieniu rozważanym w rozprawie, powszechną praktyką jest stosowanie procesu wzbogacania zbioru uczącego (ang. data augmentation). Pozwala on na znaczne rozszerzenie danych wykorzystywanych do treningu o cechy nieobecne w pierwotnym zbiorze, dzięki czemu algorytm neuronowy może nauczyć się bogatszych cech obiektu poszukiwanego na obrazie.



Rys. 33: Algorytm wzbogacający punkty danych

Proces ten zakłada pseudolosową manipulację obrazami zawartymi w zbiorze uczącym, a następnie dodanie tak obrobionych zdjęć jako osobne punkty danych. Metody wykorzystane w rozprawie zostały przedstawione na schemacie blokowym na Rys. 33, natomiast efekt wzbogacania przykładowego obrazu w zbiorze danych zobrazowano na Rys. 34.



Rys. 34: Efekt działania algorytmu do wzbogacania danych

Kod algorytmu przetwarzającego dane wejściowe w całości znajduje się w części A załącznika rozprawy zawierającego kody źródłowe aplikacji. Wykorzystany w rozprawie algorytm przeprowadza także normalizację danych na obrazie, więc kolory widoczne na obrazach przed wzbogacaniem danych będą różniły się odcieniem od tych po wzbogaceniu. Warto także zwrócić uwagę, że ze wzbogacania danych korzysta się wyłącznie na zbiorze treningowym. Modyfikacja zbioru testowego wpłynęłaby negatywnie na ocenę metryk, natomiast modyfikowanie zbioru ewaluacyjnego uniemożliwiłoby poprawne dobranie hiperparametrów modelu.

Na obrazie Rys. 34, pojedyncze zdjęcie rysy posłużyło do wygenerowania 9 innych obrazów, zawierających to samo uszkodzenie, lecz poddane transformacjom przestrzennym. Dzięki wykorzystaniu opisywanej techniki można znacząco rozbudować pierwotny zbiór danych. W zagadnieniu rozważanym w rozprawie, pojedynczy punkt danych z pierwotnego zbioru służył do generowania kolejnych 10 punktów.

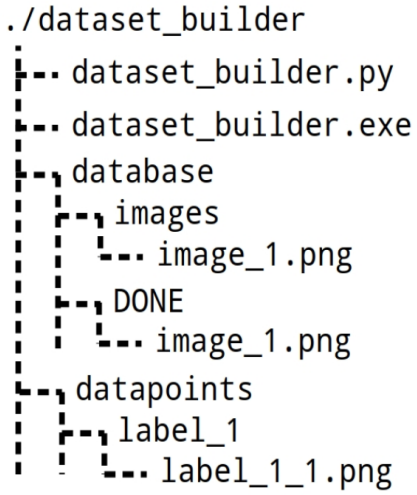
4.4. BUDOWA I PODZIAŁ ZBIORU UCZĄCEGO WSPOMAGANE PROGRAMOWO

Zebrane opisane w sekcji 4.3. obrazy służące do treningu algorytmu muszą one zostać przetworzone w etykietowane punkty danych o wymiarach odpowiadających wykorzystywanej architekturze sieci neuronowej. Proces wykonywany w sposób manualny jest czasochłonny i pracochłonny. Samo tworzenie zbiorów danych treningowych jest często opisywane jako najtrudniejszy i najbardziej kosztowny etap treningu detektorów obiektów opartych na algorytmach SI. Nie ma również gotowego oprogramowania, które znacznie zmniejszyłoby pracochłonność tego procesu, ponieważ punkty danych dla każdej architektury CNN mogą różnić się wymaganym rozmiarem.

Z tego powodu, utworzony na potrzeby rozprawy pakiet oprogramowania KrakN zawiera program komputerowy utworzony w celu usprawnienia podziału zbioru danych na etykietowane punkty danych, pod nazwą `dataset_builder.py`. W repozytorium dołączonym do rozprawy zawiera się również jego wersja w skompilowanym formacie `dataset_builder.exe` dla użytkowników systemów Windows. Aby z niego skorzystać, należy zachować konkretną strukturę katalogów przedstawioną na Rys. 35. Jednocześnie, ze względu na ograniczenia usługi obliczeń w chmurze Google Colab (ograniczona przestrzeń dyskowa oraz brak możliwości implementacji interaktywnego interfejsu) jest to jedyna część pracy z oprogramowaniem KrakN, której nie można wykonać online.

Katalog główny, zawierający plik `dataset_builder.py`, jest dalej podzielony na dwa podkatalogi. Katalog bazy danych (*database*) zawiera jeszcze nieprzetworzone obrazy w folderze *images* i już przetworzone obrazy w folderze *DONE*. W katalogu *datapoints* użytkownik oprogramowania powinien umieścić foldery z własnymi nazwami, odpowiadającymi żądanym nazwom etykiet klas (na przykład „zarysowanie” i „brak zarysowania”). W folderach tych punkty

danych będą przechowywane w postaci wycinków obrazu uzyskanych ze zdjęć wejściowych. Jeśli foldery nie zostaną nazwane ręcznie przez użytkownika, domyślnie zostanie utworzona jedna klasa zestawu danych pod nazwą „*Class_1*”. Za pomocą pojedynczego uruchomienia skryptu, użytkownik może podzielić swój zbiór danych na maksymalnie 9 oddzielnych klas. Liczbę tę można zwiększyć przy wielokrotnym wykonywaniu skryptu.

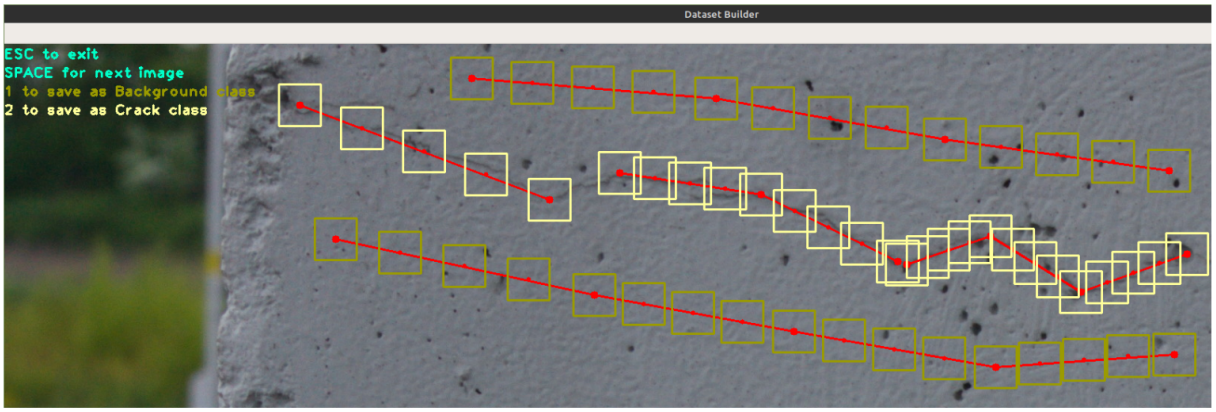


Rys. 35: Struktura folderów programu dataset_builder

Aby wyodrębnić punkty danych z obrazu wejściowego, użytkownik przy pomocy myszy wybiera ścieżki wzdłuż której poszczególne kadry zostaną wyodrębnione na obrazie wejściowym. Następnie wybiera jedną z wcześniej zadeklarowanych nazw klas, automatycznie wczytanych do programu. Domyślnie każdy odcinek ścieżki służy do wyodrębnienia 5 wycinków o domyślnych wymiarach 224 na 224 piksele. Użytkownik może zmienić te wartości, edytując plik `dataset_builder.py` jeśli architektura sieci klasyfikującej dane będzie wymagała obrazów o innych wymiarach. Ponieważ algorytm wykorzystuje metodę przesuwającego się okna, nie kompensuje automatycznie różnej skali poszukiwanego obiektu. Z uwagi na to użytkownik musi wprowadzić żądany współczynnik skali w pierwszym kroku budowy zbioru treningowego, w oparciu o rozmiar defektu na obrazie wejściowym. Wartość ta zostanie zapisana w nazwach wyjściowych plików obrazów. W prezentowanym studium przypadku współczynnik skali został ustawiony na 2 ze względu na niewielki rozmiar rozpatrywanej wady. Przykład wyglądu interfejsu opisywanego programu jak i jego działanie pokazano na Rys. 36.

Jak widać na Rys. 36, skrypt `dataset_builder.py` umożliwia szybkie wyodrębnienie punktów danych z obrazu wejściowego. Na zaprezentowanym przykładnie ponad 60 prawidłowo oznaczonych punktów danych zostało wyodrębnionych i umieszczonych w oddzielnych folderach z pojedynczego obrazu wejściowego przy pomocy jedynie kilku wskazań myszy. W studium przypadku, na podstawie ponad 900 obrazów wejściowych wyodrębniono ze zbioru danych ponad 8 tysięcy punktów danych dla każdej rozpatrywanej klasy. W sumie otrzymano ponad 16 tysięcy poprawnie

zaklasyfikowanych punktów danych, skatalogowanych w osobnych folderach. Po uwzględnieniu procesu wzbogacania zbioru uczącego, opisanego w poprzedniej sekcji, baza danych na której trenowany był algorytm neuronowy liczyła ponad 160 tysięcy obrazów. Cały proces wyodrębniania punktu danych z obrazów wejściowych trwał niecałe 4 godziny.



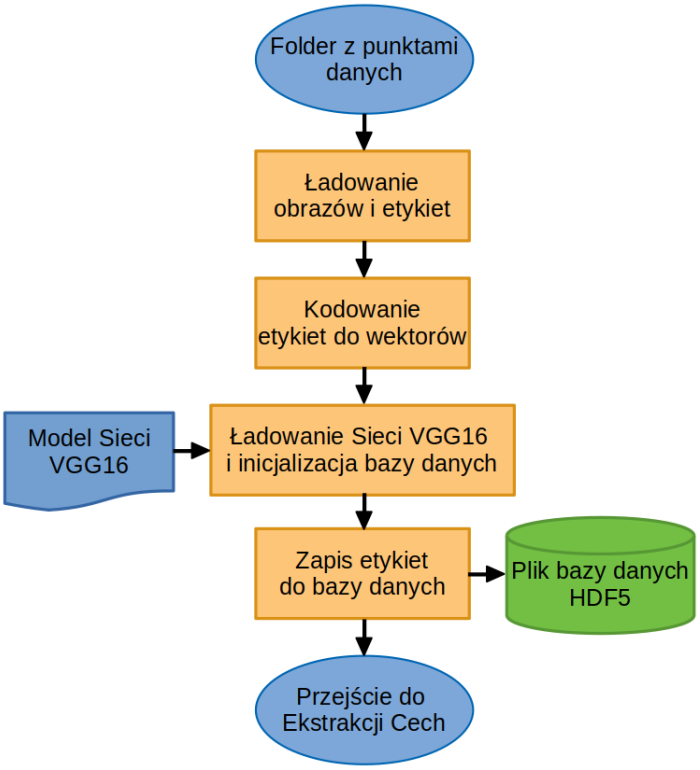
Rys. 36: Interfejs programu dataset_builder z dwiema przykładowymi klasami

W części B załącznika do rozprawy z kodami źródłowymi znajduje się kompletny kod źródłowy utworzonego na potrzeby pracy programu `dataset_builder.py`.

4.5. Implementacja metod TL

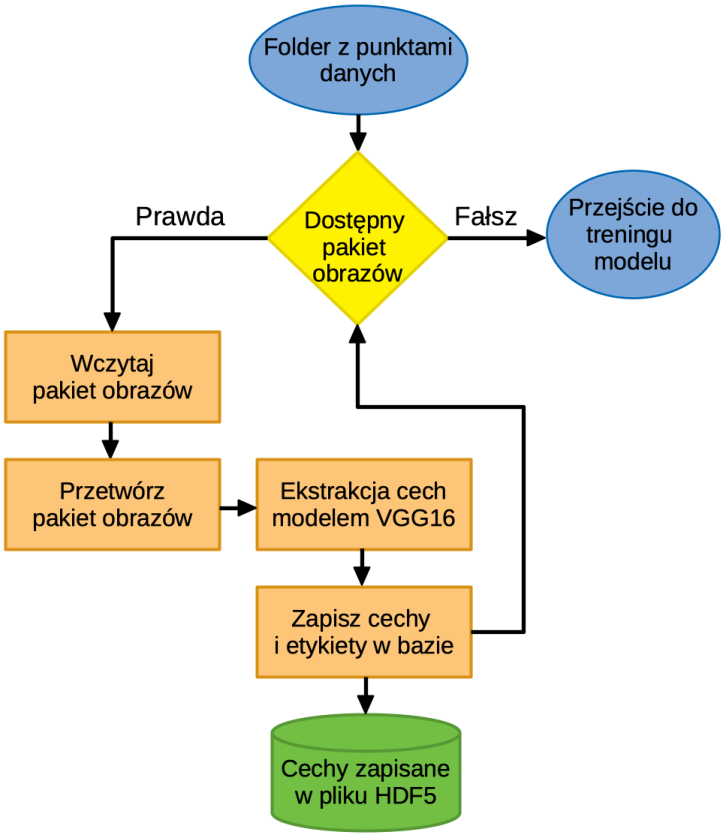
Jak zostało wspomniane w sekcji 4.1.3., budowa klasyfikatora danych wykorzystującego transfer learning składa się z dwóch faz – wyodrębniania cech charakterystycznych obrazu i właściwego szkolenia ostatniej warstwy klasyfikatora. Chociaż oba te elementy można wykonać w jednym ciągłym cyklu, często bardziej korzystne jest, jeśli wyodrębnianie cech i trening ostatniej warstwy klasyfikatora wykonywane są oddzielnie. W ten sposób można szybko i wielokrotnie szkolić klasyfikator z różnymi parametrami lub modelami regresji bez konieczności każdorazowego wyodrębniania cech ze zbioru danych, ponieważ cechy te są zapisywane w osobnym pliku. Warto również zauważyć, że w przypadku dużych zbiorów danych część wyodrębniania cech będzie zwykle bardziej czasochłonnym procesem niż samo szkolenie klasyfikatora, więc korzystne w tym przypadku będzie bieżące zapisywanie efektów pracy na wypadek na przykład awarii systemu komputerowego.

W opisywanym pakiecie oprogramowania KrakN proces wyodrębniania cech jest realizowany przez skrypt `extract_features.py`. Skrypt ten można uruchomić na komputerze lokalnym, ale dla lepszej wydajności ze względu na obliczenia w chmurze wykorzystujące wydajne procesory GPU, sugerowane jest użycie usługi Google Colaboratory. Kluczowe elementy algorytmu można znaleźć na Rys. 37 i Rys. 38.



Rys. 37: Schemat blokowy przetwarzania wstępnego etykiet

W pierwszej fazie wyodrębniania cech przedstawionej na rysunku Rys. 37 ładowane są etykiety klas pochodzące z nazw folderów utworzonej wcześniej bazy danych. Następnie etykiety są kodowane do wektorów za pomocą biblioteki `sklearn.preprocessing`. W kolejnym kroku skrypt ładuje do pamięci żądaną architekturę wcześniej wytrenowanej sieci konwolucyjnej bez ostatniej warstwy warstwy klasyfikatora (parametr `include_top` w skrypcie ma domyślnie wartość `False`). Należy zwrócić uwagę, że w przedstawionym przypadku zainicjalizowane wagi pochodzą z sieci trenowanej na zbiorze danych ImageNet, jak zostało wspomniane w sekcji 4.1.3. Ostatnie części inicjalizują obiekt, który będzie przechowywał wyodrębnione funkcje jako zestaw danych do uczenia klasyfikatorów w pliku bazy danych HDF5. W skrypcie parametr wskazujący wymiary wyodrębnionych cech pokrywa się z wymiarami wyjścia sieci, poprzez odczytanie wcześniej zapisanego przy tworzeniu bazy danych parametru skali. Po zainicjowaniu kodera etykiet i urządzenia zapisującego zestaw danych, cechy są wyodrębniane i zapisywane w pojedynczym pliku w seriach pakietów w następnym kroku.



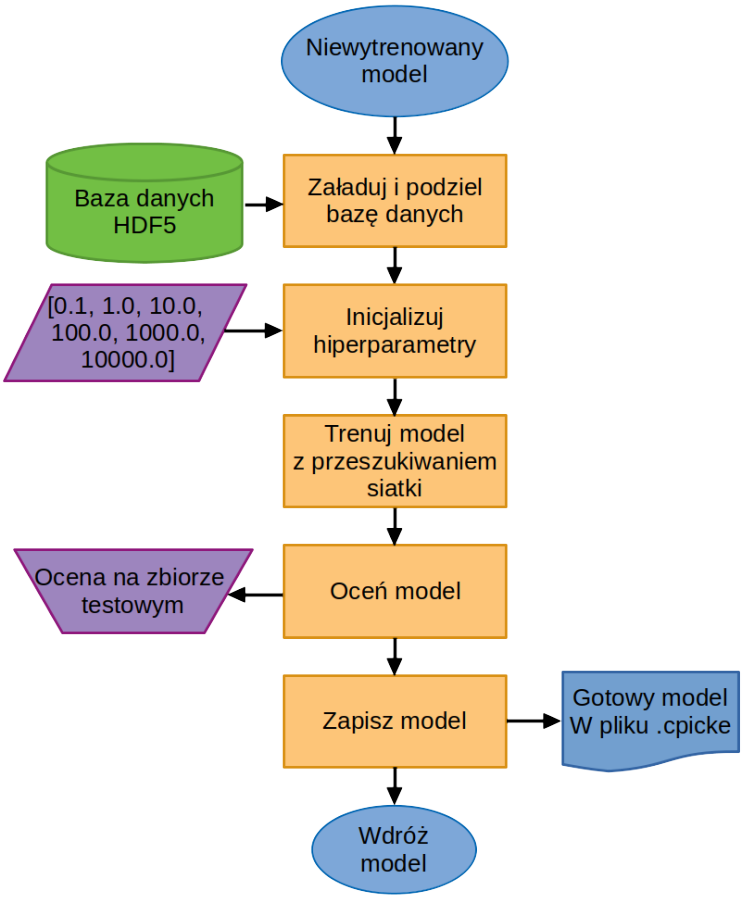
Rys. 38: Schemat blokowy ekstraktora cech obrazu

Podstawowa część ekstrakcji cech została przedstawiona na Rys. 38. Główna pętla algorytmu ładuje pakiet obrazów w wymiarach oczekiwanych przez model sieci konwolucyjnej, a następnie przetwarza go wstępnie zgodnie z instrukcją danego modelu CNN. W ostatnim kroku wstępnie przetworzone obrazy są gromadzone w jednym pakiecie, a ekstrakcja cech odbywa się z wykorzystaniem wcześniej załadowanego modelu VGG16, który zapisuje cechy charakterystyczne aktualnego pakietu obrazów. Cechy z odpowiednimi etykietami są zapisywane w wyjściowym zestawie danych, w pliku HDF5.

Wykonanie skryptu skutkuje utworzeniem pliku HDF5 w folderze *database* zawierającym wybrane cechy i etykiety wejściowego zbioru danych. Jego rozmiar silnie zależy od liczby obrazów wejściowych. Ponieważ w przeciwieństwie do obrazów wejściowych dane nie są w żaden sposób kompresowane, ich rozmiar będzie zwykle większy niż suma rozmiarów obrazów wejściowych. W prezentowanym studium przypadku rozmiar wyodrębnionych ze zbioru uczącego cech przekracza 3 GB. Z wykorzystaniem usługi Google Colab, wyodrębnienie cech zbioru danych z parametrem regulującym wielkość przetwarzanego jednorazowo pakietu przy *batchSize* = 128 zajęło mniej niż 3 minuty. Na komputerze lokalnym bez wykorzystania wydajnego procesora obliczeniowego operacja ta zajęłaby około 6 godzin.

Podstawowe fragmenty kodu źródłowego skryptu `extract_features.py` zostały przedstawione w części C załącznika do rozprawy.

Jak już zostało wspomniane w sekcji 4.1.3.2., TL umożliwia wykorzystanie wstępnie wytrenowanego modelu i wyszkolenie tylko ostatniej warstwy sieci neuronowej. W prezentowanym w rozprawie oprogramowaniu KrakN, proces ten jest realizowany w skrypcie `train_model.py`, a zasadniczą część procesu przedstawiono na Rys. 39. W opisywanym podejściu wykorzystany został algorytm przeszukiwania siatki na modelach regresji logistycznej, trenowanych różnymi hiperparametrami. Dzięki temu spośród wszystkich wytrenowanych modeli, pakiet KrakN automatycznie może wybrać model wykazujący najlepszą dokładność na zbiorze testowym.



Rys. 39: Trening modelu przed wdrożeniem

W pierwszej części algorytm wczytuje wcześniej wyodrębnione cechy charakterystyczne zbioru treningowego i dzieli punkty danych na zbiory do treningu i oceny modelu. Najczęściej używane wartości do podziału danych treningowych dzielą bazę danych w stosunku bliskim 0,75 na korzyść danych służących do treningu. Następnie tworzona jest lista parametrów treningowych (hiperparametrów). Wartości te najczęściej inicjowane w skali logarytmicznej, jak na Rys. 39. W dalszym kroku tworzony i trenowany jest zarówno algorytm przeszukiwania sieci, jak i model regresji logistycznej. Modele te są oceniane na pozostałej części zbioru danych. Użytkownik może następnie

ocenić wyniki treningu na podstawie przedstawionych metryk. W ostatnim kroku tworzony jest plik `KrakN_model.cpickle`, do którego zapisywane są parametry modelu osiągającego najlepsze wyniki na zbiorze testowym.. Proces szkolenia dla studium przypadku opisywanego w rozprawie, przy użyciu usługi Google Colab wykorzystującej tylko jednordzeniowy procesor, zajął mniej niż 20 minut, osiągając dokładność predykcji na poziomie 97% i 98% odpowiednio dla klas nieuszkodzonej powierzchni betonu oraz wykrytej rysy.

Kod źródłowy opisywanego powyżej skryptu `train_model.py` został przedstawiony w części D załącznika do rozprawy.

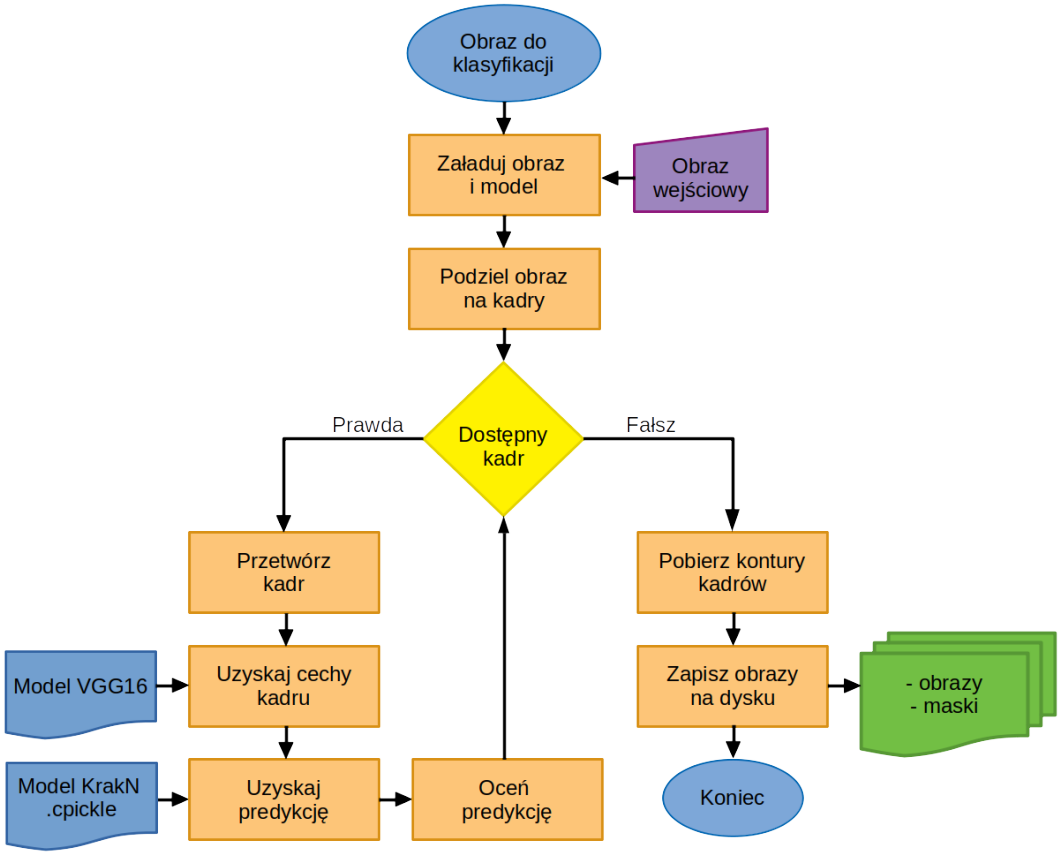
4.6. IMPLEMENTACJA DETEKTORA OBIEKTÓW

Aby wykorzystać wcześniej wytrenowany klasyfikator jako detektor obiektów, należy zaimplementować algorytm przesuwającego się okna. Na potrzeby prezentowanej rozprawy algorytm ten został zaimplementowany w skrypcie `moving_window.py` znajdującym się w repozytorium dołączonym do rozprawy. Główną pętlę algorytmu przedstawiono na Rys. 40.

W tej głównej pętli, wymiary obrazu są dostosowywane zgodnie z zapisanym parametrem skali bazy danych, ustawionym podczas korzystania ze skryptu `dataset_builder.py`. Jak zostało wspomniane w sekcji 4.1.1., wielkość kroku z którym wybierane są kolejne fragmenty obrazu wejściowego, jest ustawiony na 0,60 rozmiaru obrazu akceptowego przez sieć neuronową. Następnie kadr wejściowy jest wstępnie przetwarzany w taki sam sposób, jak w skrypcie ekstrahującym cechy charakterystyczne. W dalszym kroku wytrenowany poprzednio klasyfikator jest wykorzystywany do przewidywania klasy wybranego kadru. Po predykcji wybierana jest najbardziej prawdopodobna klasa obiektu, a wartość prawdopodobieństwa porównywana jest z progiem ufności. Próg ten jest ustawiony w celu zmniejszenia liczby fałszywie pozytywnych prognoz. Przy zaledwie dwóch klasach, prawdopodobieństwo powyżej 50% skutkowałoby przypisaniem obiektu do danej klasy. Jednak w celu podniesienia pewności predykcji, parametr ten domyślnie jest ustawiony na 0,95. Użytkownik może go zmienić ręcznie mając na uwadze liczbę klas jego klasyfikatora.

Po przeprowadzonej predykcji zapisywane są pliki masek dla każdej klasy, zgodnie z metodą przedstawioną na Rys. 28 w sekcji 4.1.1.. W ostatnim kroku, na podstawie obrazów masek, generowane są obrazy wyjściowe z defektami z każdej klasy zaznaczonymi czerwonymi prostokątami. Znaczniki prostokątów są generowane jako obwiednie obiektów odnalezionych na maskach obrazów, więc klasy w obrazie wyjściowym mogą się nakładać, co będzie szczególnie widoczne w przypadku klasy tła (betonu nieuszkodzonego). Z wykorzystaniem usługi Google Colab cały proces dla obrazu wejściowego o wielkości 4248 na 2850 piksele, na którym przeprowadzono ponad 2500 predykcji zajmuje mniej niż minutę. Przykładowe wyniki działania algorytmu przedstawione zostały w kolejnej sekcji.

Kod źródłowy opisywanego powyżej skryptu `moving_window.py` został zamieszczony w w części E załącznika do rozprawy.

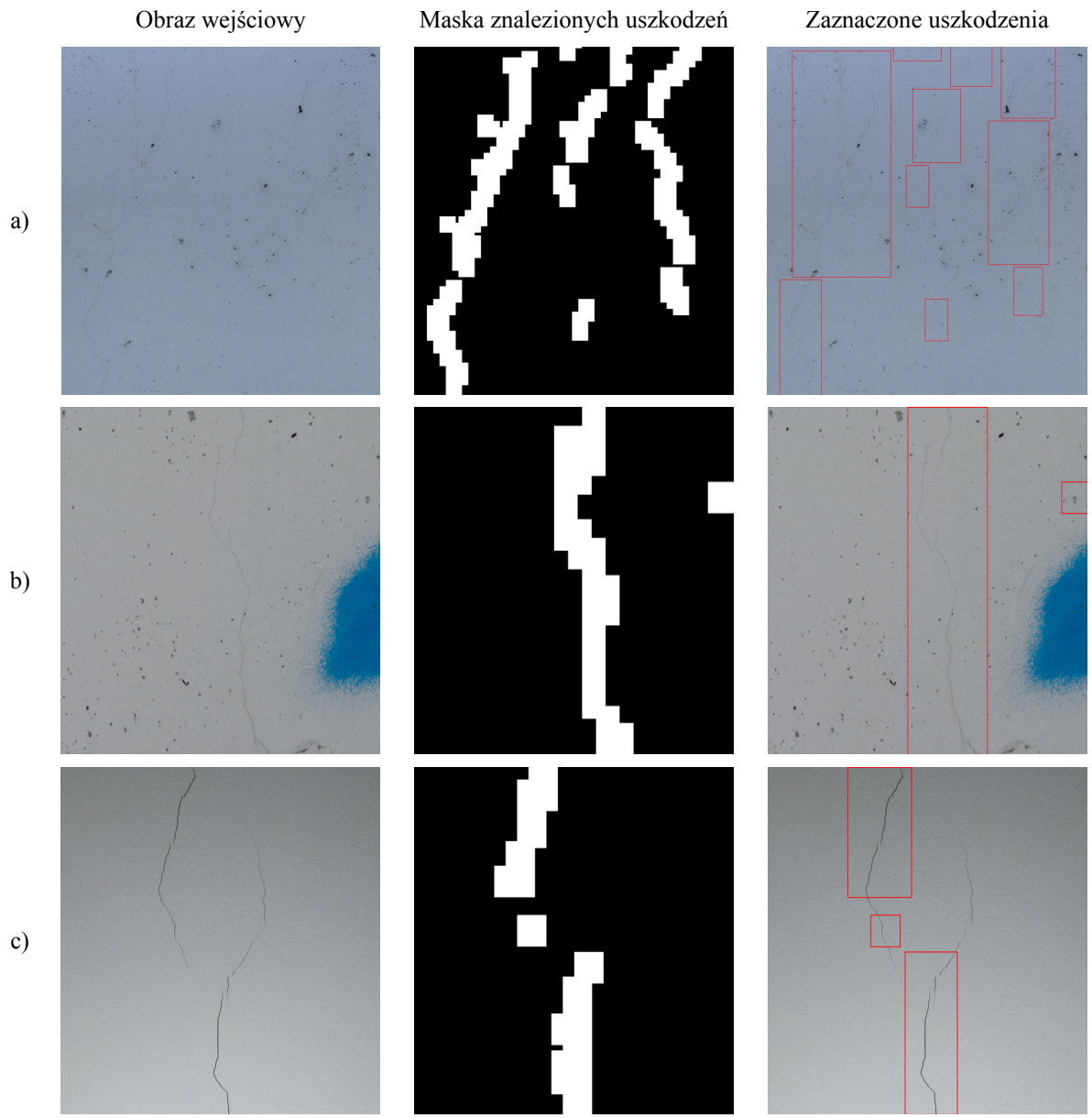


Rys. 40: Implementacja algorytmu poruszającego się okna

4.7. OCENA SKUTECZNOŚCI PROPONOWANEGO ROZWIĄZANIA

Na Rys. 41 przedstawiono przykładowe wyniki procedury zaimplementowanej w skrypcie `move_window.py`. W efekcie działania skryptu generowane są obrazy wyjściowe w postaci dwuklasowych masek oraz obrazy z wykrytymi defektami zaznaczonymi prostokątnymi obwiedniami. Wygenerowane maski można później wykorzystać do dalszego zarządzania defektami, natomiast obrazy z obwiedniami mogą służyć jako bieżące wskazówki przy pracy inspektora mostowego.

Poszczególne przypadki przedstawione na Rys. 41 pokazują możliwe wyniki implementacji oprogramowania KrakN. W jego praktycznym zastosowaniu stosunek wyników fałszywie dodatnich do fałszywie ujemnych może być inny. Za manipulację tą proporcją odpowiada zmienna `confidence_threshold`, w celu zminimalizowania zagrożenia fałszywie negatywnymi lub pozytywnymi predykcjami.



Rys. 41: Przykłady zastosowania pakietu algorytmów KrakN

Rys. 41 przedstawia przykłady wykrytych przy pomocy opisywanego oprogramowania defektów. W części a) ilustracji zamieszczono fragment przeanalizowanej 120-megapikselowej ortomozaiki wygenerowanej na filarze mostu, z którego pobierane były dane do zbioru testowego. Pokazuje ona zdolność pakietu KrakN do wykrywania defektów na obrazach o dużej rozdzielczości.. W przypadku tego przykładu, obiektem testowym był pierwszy, zachodni filar esktakady Heweliusz w Gliwicach, któremu wykonano 980 zdjęć. Zdjęcia te następnie posłużyły to utworzenia czterech ortofotomap o rozdzielczości 120 megapikseli dla każdej z czterech powierzchni filara. Część b) zawiera rysę z filaru nie będącego częścią zbioru uczącego. Należy zwróć uwagę, na fałszywie dodatnią predykcję w prawym górnym rogu obrazu wyjściowego. W części c) zamieszczono kolejny przykład działania algorytmu, choć w tym przypadku do pobierania danych wejściowych zastosowano niskobudżetowy (o cenie poniżej 100 EUR) aparat w smartfonie. Pomimo niskiej jakości zastosowanej

kamery i różnej od zbioru treningowego typu powierzchni, algorytm zachował zdolność wykrywania defektów.

Oprócz testów przeprowadzonych na obrazach, prezentowany w rozprawie zbiór algorytmów został przetestowany także z wykorzystaniem metod opisanych w sekcji 4.1.4..

Głównym założeniem eksperymentu było przetestowanie klasyfikatora na niewidzianych w trakcie treningu algorytmu danych. Zostały uzyskane na różnych materiałach, powierzchniach i powłokach różnego typu zarejestrowanych przez kamery o różnych parametrach. Zmianie podlegały także warunki oświetlenia. W sumie do ewaluacji proponowanego rozwiązania wykorzystano ponad 3 tysiące nowych obrazów. Aby umożliwić odtworzenie wyników badania eksperymentalnego, zestaw danych ewaluacyjnych oraz wytrenowany klasyfikator zostały umieszczone w repozytorium rozprawy. Na podstawie pozyskanych danych, zamieszczonych w macierzy błędów (Tab. 10), obliczone zostały metryki dokładności i czułości predykcji klasyfikatora.

Tab. 10: Tablica pomyłek dla algorytmu KrakN

	Przewidziane tło	**Przewidziana rysa**
Faktyczne tło	1577 (TN)	125 (FP)
Faktyczna rysa	122 (FN)	1630 (TP)
	Liczba próbek	3452

Poniżej przedstawiono metryki zbioru algorytmów KrakN, obliczone na podstawie tablicy pomyłek:

$$Czułość = 0{,}93,$$
$$Precyzja = 0{,}93.$$

Należy podkreślić, że w celu jak najwierniejszej symulacji działania IM, dane treningowe dla sieci zostały zebrane na pojedynczym obiekcie budowlanym. Ponadto zbiór danych ewaluacyjnych nie pokrywał się w żadnej części z danymi służącymi do treningu algorytmu. Pomimo tego, opracowany zbiór algorytmów zachował zdolność do generalizowania wiedzy na różne powłoki, warunki oświetlenia i charakterystyki kamery. Zachował jednocześnie wysoką dokładność i czułość predykcji. Należy zauważyć, że w celu prawidłowej oceny algorytmu próg ufności został ustawiony na 0,50, zgodnie z ilością klas. W praktycznych zastosowaniach, w których próg ufności jest ustawiany zgodnie z preferencjami użytkownika, stosunek kategorii fałszywie dodatnich do negatywnych może być inny niż przedstawiony w Tab. 10.

Pomimo uzyskania niższej dokładności predykcji niż podczas uczenia algorytmu, otrzymane wyniki są nadal znacznie wyższe niż te uzyskane podczas ręcznych inspekcji (patrz [228] oraz sekcja 1.1.). Najbardziej zaskakujący w wynikach testów był fakt, że opracowany algorytm wciąż osiąga ponad 50% wyższe wyniki od IM w testach manualnych w wykrywaniu małych defektów

powierzchni, mimo że został trenowany na pojedynczym obiekcie infrastruktury, a poszukiwane defekty są prawie nie do odróżnienia od tła gołym okiem. Fakt ten jest tym bardziej widoczny, że dane ewaluacyjne w większości zostały pozyskane przy pomocy telefonu komórkowego, wykonującego zdjęcia o zdecydowanie niższej jakości, niż nawet niskobudżetowe aparaty cyfrowe. Świadczy to o tym, że proponowana metoda może znaleźć zastosowanie w praktyce i jest obiecującym narzędziem do wspomagania automatyzacji procesu utrzymania infrastruktury.

4.8. ALTERNATYWNE MOŻLIWOŚCI UŻYCIA PROPONOWANEJ METODY

Jedną z najważniejszych cech prezentowanego w rozprawie podejścia jest zachowanie pełnej modułowości i otwartości kodu narzędzia. Umożliwia to swobodną manipulację dołączonymi do rozprawy skryptami w sposób umożliwiający ich rozbudowę i dostosowanie do konkretnych potrzeb danej jednostki zarządzającej obiektami mostowymi.

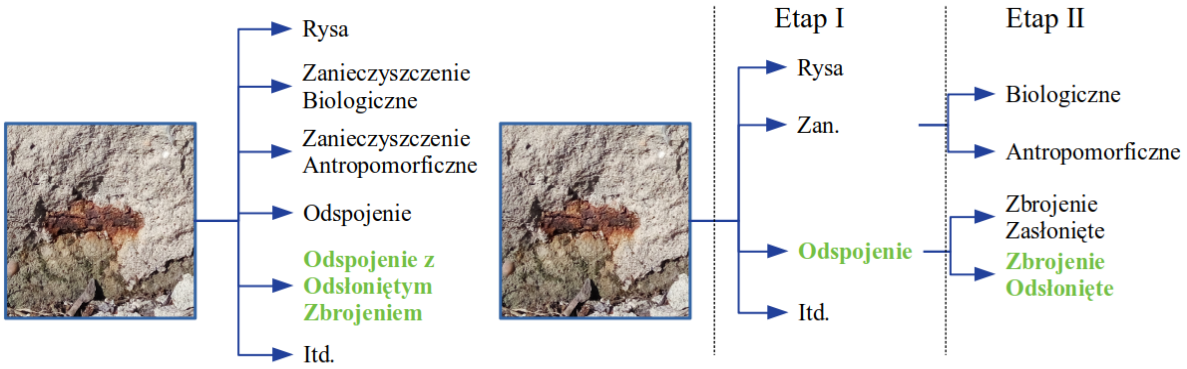
Aby zademonstrować wszechstronność pakietu oprogramowania KrakN, został on wykorzystany jako podstawa do odtworzenia podejścia opartego na wielostopniowej (wieloetapowej) klasyfikacji uszkodzenia (ang. multi-stage classifier), przedstawionego w [229]. Głównym założeniem wielostopniowej klasyfikacji jest możliwość uzyskania wyższej dokładności predykcji w porównaniu do typowego klasyfikatora, przy etapowaniu klasyfikacji na obiekty znacznie różniące się od siebie. Oba podejścia zostały zobrazowane na Rys. 42.

Podejście wieloetapowe przedstawione na Rys. 42, pozwala na dokładniejszy podział przypadków defektów poprzez grupowanie ich w podzbiory. Pozwala to na łatwiejsze zarządzane jednocześnie uszkodzeniami i ich klasyfikatorami. Ponadto, stosując wiele niezależnych od siebie klasyfikatorów, można używać mniejszych zbiorów danych uczących, co skraca czas treningu dla klasyfikatora. W przypadku konieczności dodania nowej klasy defektów do klasyfikacji, ponownego treningu wymaga tylko jeden klasyfikator z ciągu etapów klasyfikacji.

Na potrzeby zbadania możliwości wykorzystania proponowanego pakietu algorytmów w podejściu klasyfikacji wieloetapowej, w repozytorium rozprawy utworzono katalog `network_trainer_Staged`. Zawiera on szablon, który może posłużyć do wdrożenia KrakN w podejściu wieloetapowym. Pozwala on na implementację pakietu KrakN do wykorzystania obrazów wyjściowych i masek z pierwszego etapu klasyfikacji do kolejnego etapu analizy obrazu z wykorzystaniem innego klasyfikatora. To z kolei umożliwia dalszy podział defektów na podzbiory.

Aby przeprowadzić klasyfikację wieloetapową, użytkownik pakietu KrakN musi przeprowadzić trening nowych klasyfikatorów do wykorzystania w poszczególnych etapach klasyfikacji przy użyciu metod opisanych w sekcji 4.5. z nowym zbiorem danych. Jednocześnie należy zwrócić uwagę, że każdy z klasyfikatorów w etapach musi być oddzielnym modelem. Na przykład,

drugi etap analizy obrazu przedstawiony na Rys. 42 składałby się z dwóch klasyfikatorów – oddzielnego dla odspojenia i zanieczyszczenia.



Rys. 42: Różnica między klasyfikacją jednostopniową (po lewej) a wielostopniową (po prawej)

Aby oprogramowanie wykryło kolejne modele klasyfikatorów automatycznie, muszą zostać one nazwane tak, aby jego nazwa zawierała typ uszkodzenia z poprzedniego etapu, a sam model musi zostać umieszczony w podkatalogu `Classifiers`. Przystępując do drugiego etapu klasyfikacji, użytkownik musi umieścić obrazy i ich maski w odpowiednio nazwanych folderach w katalogu `network_trainer_Staged` i ponownie uruchomić skrypt `move_window_staged.py`. W rezultacie w katalogu `Output` zostaną utworzone nowe dane wyjściowe klasyfikacji składające się jak poprzednio, z oznaczonych obrazów i masek zawierających gabaryty uszkodzenia. Skrypt `move_window_staged.py` może przyjąć dowolną liczbę obrazów, masek i klasyfikatorów.

Powyższy przykład wykazuje, że zbiór algorytmów KrakN jest wszechstronny i podatny na modyfikacje, które pozwalają mu wykonywać bardziej złożone zadania po niewielkich zmianach. Jednocześnie jest w stanie zmieniać swoją bazową funkcjonalność na potrzeby wykonywania konkretnych zadań inspektora mostowego, pozostając przy tym mniej wymagającym obliczeniowo niż inne podejścia. Całkowity potencjał prezentowanego w rozprawie rozwiązania kryje się w możliwościach jego łatwej rozbudowy, dzięki której może zastąpić wiele równorzędnych rozwiązań przy minimalnym wkładzie w jego rozbudowę.

5. EKSPERYMENT PORÓWNAWCZY

W zastosowaniach praktycznych aspekty związane z możliwością wykorzystania opisywanych metod w rzeczywistych scenariuszach zarządzania infrastrukturą są równie ważne, jak parametry opisujące ich skuteczność w wykrywaniu uszkodzeń. Z tego powodu rozdział ten przedstawia analizę porównawczą, w której wzięte pod uwagę zostały oba te czynniki. Całkowity czas treningu algorytmu będzie mierzony przy użyciu maszyn obliczeniowych o różnych parametrach, a następnie porównywane będą metryki algorytmów na zbiorze ewaluacyjnym. Dzięki takiemu podejściu określona zostanie praktyczna wartość opisywanego podejścia zarówno od strony dokładności wskazań w porównaniu do metod obecnych w aktualnej literaturze, ale także od strony możliwości budowy i treningu algorytmów SI z wykorzystaniem typowych maszyn obliczeniowych znajdujących się w urzędach zarządzających infrastrukturą.

5.1. ZAŁOŻENIA EKSPERYMENTU

Eksperyment porównawczy został podzielony na dwa niezależne od siebie etapy. W pierwszym z nich proponowany zbiór algorytmów KrakN był porównany z siecią konwolucyjną uczoną od wag losowych. Sieć ta w założeniu osiągać ma podobną dokładność i czułość co KrakN, a główną porównywaną wartością będzie czas treningu algorytmu. Będzie posiadała także niezależną od pakietu KrakN architekturę, a jej trening zostanie przeprowadzony na tym samym zbiorze danych.

Drugi etap eksperymentu zakładał porównanie proponowanego w rozprawie rozwiązania do dwóch równorzędnych, obecnych w aktualnej literaturze rozwiązań, bazujących na wykorzystaniu sztucznej inteligencji. W tym etapie, oprócz zapotrzebowania na moc obliczeniową, porównywana była także dokładność i czułość rozwiązań. Porównana została także kluczowa dla algorytmu zdolność do generalizowania wiedzy między zbiorami danych różnych od pierwotnego, na którym przeprowadzony został trening.

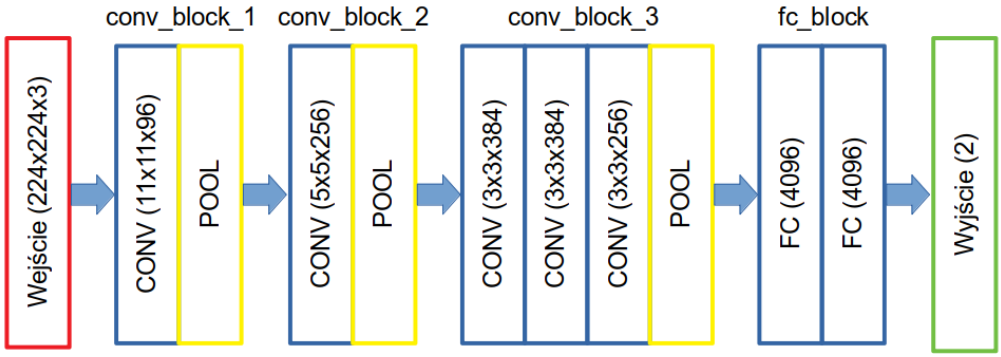
5.1.1. SIEĆ NEURONOWA UCZONA OD WAG LOSOWYCH

W celu przeprowadzenia badania porównawczego podkreślającego korzyści płynące z wykorzystania algorytmu opartego na TL, od podstaw przeprowadzono budowę i trening konwolucyjnej sieci neuronowej o architekturze sekwencyjnej, podobnej do sieci VGG16 wykorzystanej do implementacji algorytmu KrakN. Zgodnie z założeniami opisanymi w sekcji 5.1., do

treningu algorytmu wykorzystano ten sam zbiór uczący co w przypadku głównego, rozwijanego w rozprawie programu. Ponadto, sieć była oceniana także na tych samych danych ewaluacyjnych. Eksperyment miał na celu sprawdzenie czy sieci uczone z zastosowaniem TL mogą osiągać podobne metryki i zdolność do uogólniania wiedzy z wąskiego zbioru danych, tak jak sieci trenowane od wag losowych. Ocenie poddano także praktyczną możliwość adaptacji sieci szkolonych od podstaw w przypadku pracy na maszynach roboczych o ograniczonych zasobach.

Podczas szkolenia sieci do badań porównawczych wykorzystano kilka architektur o różnej liczbie parametrów i wartości hiperparametrów. Średni czas szkolenia jednego modelu z użyciem usługi Google Colab wyniósł 10 godzin, a łączny czas szkolenia wszystkich testowanych modeli ponad 50 godzin. Trening i testy sieci odbywały się pod stałym nadzorem. Podczas szkolenia modele były stale oceniane pod kątem unikania stronniczości sieci na korzyść jednej z klas. Ostateczną architekturę sieci wykorzystaną do badań eksperymentalnych można zobaczyć na Rys. 43.

Zaprezentowana na Rys. 43, architektura sieci użyta do badania porównawczego jest mniej rozbudowana w porównaniu z architekturą sieci użytej jako ekstraktor cech charakterystycznych obrazu w pakiecie KrakN (por. Rys. 29). Podczas procesu uczenia stwierdzono, że model z mniejszą liczbą warstw konwolucyjnych w których przebiega proces uczenia się sieci jest wystarczający dla rozpatrywanego problemu. W porównaniu z opisanym wcześniej modelem sieci VGG16, wykorzystuje on ponad dwukrotnie mniej dających się trenować parametrów. Mogła być to zatem przesłanka świadcząca o możliwości treningu sieci na mniej wydolnej maszynie obliczeniowej. Model ten znalazł się również w repozytorium rozprawy.



Rys. 43: Architektura sieci wykorzystanej w eksperymencie porównawczym

Biorąc pod uwagę możliwość zastosowania proponowanego rozwiązania w praktyce utrzymania infrastruktury mostowej przez jednostki rządowe, należy zauważyć, że wykorzystanie techniki transfer learningu w istotny sposób ogranicza zapotrzebowanie na zasoby obliczeniowe. Zależność ta została przedstawiona w Tab. 11. Zbiór algorytmów KrakN trenowany przy pomocy jednordzeniowego, biurowego procesora Intel Xeon zapewnia 20-krotne skrócenie czasu obliczeń w porównaniu z treningiem sieci uczonej od wag losowych. Nawet w przypadku użycia wydajnego

procesora graficznego (GeForce GTX 1080). Przewaga metod KrakN jest jeszcze wyraźniejsza, gdy dostępne są tylko mniej wydajne obliczeniowo procesory CPU, ponieważ konwencjonalne szkolenie tej samej sieci neuronowej z wykorzystaniem wyłącznie procesora zajęłoby prawie 20 dni ciągłych obliczeń. Ponadto wartości przedstawione w Tab. 11 pokazują tylko jedną iterację uczenia dla porównywanych metod. Podczas gdy zbiór algorytmów KrakN zarządza hiperparametrami treningu, pozwalając na jednorazowe uruchomienie treningu algorytmu, konwencjonalny sposób treningu sieci CNN wymaga zazwyczaj wielokrotnego ich testowania przed zakoczeniem treningu. Jest to istotna przesłanka przemawiająca za przewagą proponowanego rozwiązania nad algorytmami SI uczonymi w sposób tradycyjny.

Tab. 11: Porównanie czasu treningu algorytmów KrakN do konwencjonalnej sieci CNN

KrakN [CPU]	CNN [GPU]	CNN [CPU]
25 minut	~ 10 godzin	~ 444 godziny (19 dni)

W Tab. 12 przedstawiona została macierz błędów dla sieci konwolucyjnej uczonej od wag losowych.

Tab. 12: Macierz błędów dla konwencjonalnie uczonej sieci CNN

	Przewidziane tło	Przewidziana rysa
Faktyczne tło	1599 (TN)	117 (FP)
Faktyczna rysa	103 (FN)	1630 (TP)

Na podstawie danych zawartych w macierzy błędów (Tab. 12), obliczono następujące metryki sieci:

$$Czułość = 0{,}94,$$
$$Precyzja = 0{,}94.$$

Porównując wyniki treningów obu sieci można zauważyć, że metryki sieci trenowanej od podstaw są tylko nieznacznie lepsze niż w przypadku sieci trenowanej z pomocą TL. Obie sieci zyskały porównywalną możliwość generalizowania wiedzy w różnych zbiorach danych w oparciu o zbiór treningowy o ograniczonej wielkości i różnorodności. Oba algorytmy uzyskały również znacznie większą skuteczność w identyfikowaniu uszkodzeń niż powszechnie stosowane metody ręczne.

Jednakowoż, dla praktycznego zastosowania algorytmów sztucznej inteligencji równie ważna jak dokładność działania sieci jest możliwość ich szkolenia i wdrażania przez jednostki administracyjne przeprowadzające kontrole obiektów mostowych. Mimo że metody uczenia transferowego, uzyskują tylko nieco mniejszą dokładność niż klasyczne metody uczenia sieci, oferują znaczne zmniejszenie zapotrzebowania na moc obliczeniową podczas treningu, a tym samym pozwalają na szkolenie większej liczby klasyfikatorów w krótszym czasie do różnych zastosowań praktycznych. Co więcej, dzięki wykorzystaniu publicznych, wielokrotnie testowanych w rozległych

zbiorach danych, takich jak ImageNet, sieci jako ekstraktory cech charakterystycznych obrazu, podobne możliwości uogólniania wiedzy zostaną uzyskane w różnych przypadkach użycia, niezależnie od czasu poświęconego na trening algorytmu. Dzięki temu korzystając z algorytmów KrakN, nie jest konieczne każdorazowe dostrajanie hiperparametrów treningu sieci, gdy klasyfikator jest trenowany. Do jego obsługi nie jest także wymagana specjalistyczna wiedza z zakresu tworzenia i uczenia algorytmów CNN.

Tym samym jednak należy pamiętać, że aby uzyskać najwyższe parametry klasyfikacji sieci, zbiór danych służący do jej uczenia powinien być porównywalny z danymi, które algorytm ma klasyfikować w praktyce. Mimo że założenie dotyczące jednorodności zbioru uczącego było jednym z podstawowych założeń rozprawy, to w zastosowaniach praktycznych polecane jest rozszerzanie zbioru uczącego z każdym kolejnym obiektem infrastruktury poddanym inspekcji.

5.1.2. PORÓWNANIE Z AKTUALNYM STANEM WIEDZY

W drugim etapie eksperymentu porównawczego, zbiór algorytmów KrakN został porównany z dwoma obecnymi w najnowszej literaturze w dziedzinie podejściami – CrackNet [214] i DeepCrack [230] do wykrywania rys na powierzchni betonu. Oba wymienione algorytmy w swoim działaniu wykorzystują metody sztucznej inteligencji. Należy jednak zauważyć, że podejścia te są ograniczone do wykrywania zarysowań, podczas gdy KrakN jest otwartym oprogramowaniem, zdolnym do wykrywania wielu typów defektów, w zależności od zastosowanego zbioru uczącego.

W celu przeprowadzenia porównania, każdą z sieci oceniono na dwóch zbiorach ewaluacyjncyh. Nie były one jednocześnie poddane ponownemu treningowi, gdyż nie pozwalały na to zastosowanie przy ich budowie metody. Pierwszym z nich był oryginalny zbiór ewaluacyjny opisany w sekcji 4.1.4., zawierający rysy o rozwartości poniżej 0,2 mm. Drugi został wybrany spośród zbiorów danych znajdujących się w repozytorium Github projektu CrackNet [231] i zawierał rysy o znacznie większej rozwartości niż w opisywany poprzednio i składał się z 40 tysięcy obrazów. Wyniki przeprowadzonego porównania zapisano w Tab. 13 poniżej.

Tab. 13: Porównanie metryk sieci na różnych zbiorach ewaluacyjnych

	Zbiór danych KrakN [P/C]	**Zbiór danych CrackNet [P/C]**	**Spadek metryk**
KrakN	0,93 / 0,93	0,81 / 0,81	**0,12 / 0,12**
CrackNet	0,48 / 0,14	0,98 / 0,98	0,50 / 0,84
DeepCrack	0,26 / 0,40	0,97 / 0,79	0,71 / 0,39

P – Precyzja
C – Czułość

W Tab. 13, dokładność i czułość predykcji sieci różni się przy ewaluacji na różnych zbiorach danych. Wynika to z różnych możliwości sieci neuronowych do uogólniania wiedzy między pozornie

identycznymi zadaniami. Jednak dzięki wyodrębnieniu bardziej ogólnych cech obrazu wejściowego za pomocą bazy z sieci konwolucyjnej do zastosowania ogólnego i wykorzystaniu metody TL, algorytmy KrakN osiągają lepsze wyniki, w ocenianiu na danych różniących się od pierwotnego zbioru treningowego, co skutkuje mniejszym spadkiem metryk między zadaniami.

Dodatkowo, przeprowadzone zostało także porównanie z wykorzystaniem zbioru CrackNet do treningu algorytmów KrakN. Z użyciem tego zbioru danych, KrakN osiągnął wartość 0,99 zarówno dla dokładności jak i czułości, podczas ewaluacji. Następnie, bez ponownego szkolenia, osiągnął odpowiednio 0,77 i 0,91 dla dokładności i czułości, gdy oceniano go na pierwotnym zestawie danych. Oznacza to, że po raz kolejny utrata metryk dla algorytmu KrakN była niższa niż dla pozostałych podejść i wyniosła (0,22 / 0,08).

Ponadto, w trakcie eksperymentu porównawczego, ujawniona została kolejna, poważna wada algorytmów CrackNet i DeepCrack. W pierwszym teście z użyciem tych algorytmów, miały one przeanalizować pokazaną na Rys. 41 120-megapikselową ortomozaikę. Algorytmy te jednak nie były w stanie przetworzyć tak dużych obrazów bez przekraczania 25 GB pamięci VRAM. Takie zasoby maszyny obliczeniowej można spotkać częściej na wyspecjalizowanych stacjach do przetwarzania i renderowania wideo, natomiast zwykle nie występującą w stacjach niskobudżetowych. Okazało się tym samym, że użycie tych algorytmów wymagało wyjątkowo dużej ilości zasobów obliczeniowych, przy czym domyślnie dedykowane były wydajnym procesorom graficznym w maszynie obliczeniowej. Ostatnim poważnym problemem odnotowanym przy użyciu algorytmów DeepCrack i CrackNet była ich niska dokładność na zbiorze ewaluacyjnym. Aby algorytmy te były w stanie wykryć pęknięcia na obrazach, wartość progu ufności wykrycia defektów musiała zostać ustawiona na jedynie 6% (15/255). W przypadku algorytmu KrakN próg ten pozostawiony został na domyślnym poziomie 50% (próg zastosowany przy ewaluacji algorytmu KrakN).

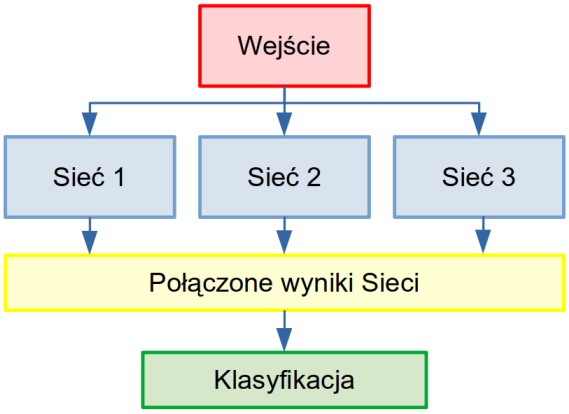
5.2. MOŻLIWOŚCI ZWIĘKSZENIA SKUTECZNOŚCI PROPONOWANEGO ROZWIĄZANIA

Dzięki otwartej strukturze prezentowanego w rozprawie podejścia, możliwe jest stosowanie dodatkowych technik mogących zwiększać skuteczność wykrywania uszkodzeń przez algorytm. Techniki te wymagają tylko niewielkich modyfikacji pakietu znajdującego się w repozytorium rozprawy i mogą zostać wdrożone nawet bez szerokiej znajomości metod uczenia maszynowego.

Przykładowa technika mogąca poprawić parametry pakietu KrakN to korzystanie z zespołów sieci (ang. network ensembles). Polega ona na klasyfikacji przy pomocy kilku poprawnie wytrenowanych klasyfikatorów jednocześnie. Nie jest to jednak technika dystynktywna dla sieci neuronowych. Jej pierwsze zastosowania sięgają algorytmów boostingu (zwiększających wagę aktualizowanych parametrów dla próbek błędnie klasyfikowanych) lub lasów losowych. W tej drugiej, trenowanych jest duża liczba algorytmów drzew decyzyjnych, które następnie wykorzystywane są razem (tworząc „las”) do głosowania nad prawdziwym rozwiązaniem zadania klasyfikacji.

Głosowanie to może następnie być rozstrzygnięte przez na przykład średnią ważoną lub kolejny klasyfikator wykorzystujący regresję logistyczną.

W przypadku sieci konwolucyjnych wykorzystywanych w zagadnieniu klasyfikacji, poszczególne sieci mogą być uczone z wykorzystaniem różnych zbiorów treningowych, a ich wskazaniom nadane mogą być różne wagi, użyte następnie w kolejnym zagadnieniu klasyfikacji lub do średniej ważonej. W ten sposób sterować można ręcznie stronniczością sieci składowych, uzależniając ją na przykład od typu analizowanej powierzchni lub rozwartości uszkodzenia. Dzięki temu zrównoważyć więc można różnice metryki czułości i precyzji wynikające z treningu na jednorodnych zbiorach danych. Główna zasada, w myśl której wykorzystywane są zespoły sieci została przedstawiona na Rys. 44.



Rys. 44: Schemat budowy zespołu sieci

Powód, dla którego zespoły sieci mogą osiągać wyższą dokładność klasyfikacji niż pojedyncze sieci, przedstawiony został w pracy seminaryjnej z zakresu uczenia maszynowego [232]. Została ona oparta na zasadzie nierówności Jensena. W kontekście zagadnienia klasyfikacji opisać ją można jako możliwość wystąpienia w zespole modelu o znacznie większej dokładności niż średnia dokładność innych modeli, przez co najbardziej racjonalnym wyborem jest użycie wszystkich modeli jednocześnie. Jest to dodatkowo podyktowane brakiem ścisłych kryteriów wyboru najlepszego modelu, który ponownie może wykazywać najkorzystniejsze metryki tylko dla pojedynczego zadania klasyfikacji. Tym samym, wybór zespołu zawsze powinien gwarantować lepsze wyniki klasyfikacji niż wybór losowego modelu.

W przypadku proponowanego w rozprawie rozwiązania nie występuje ograniczenie co do liczby modeli wykorzystywanych na potrzeby zespołu, ani sposobu w jaki przeprowadzana będzie końcowa klasyfikacja. Ponadto, metody zaimplementowane w skrypcie `train_model.py` po niewielkich modyfikacjach mogą być użyte do budowy osobnej aplikacji odpowiadającej za końcową klasyfikację zespołu sieci.

Innym sposobem, który można zastosować w celu poprawy dokładności sieci trenowanych na jednorodnych zbiorach danych, są techniki obejmujące dedykowane, wstępne przetwarzanie danych wejściowych. Dzięki jego zastosowaniu uwidocznić można na obrazie najbardziej istotne dla klasyfikacji uszkodzenia cechy. Można do tego użyć między innymi automatycznych detektorów krawędzi, normalizacji danych zawartych na obrazie lub ręcznie programowanych filtrów konwolucyjnych.

Dzięki wszechstronności opisywanego w rozprawie podejścia, jako dane do treningu sieci wykorzystać można także obrazy z innych niż widzialne spektrów. Oprócz kamer termowizyjnych, które znalazły już zastosowanie w wykrywaniu defektów na obiektach mostowych [233], stosować można także obrazy z kamer wielo- i hiperspektralnych. Należy jedynie pamiętać o odpowiedniej obróbce wstępnej obrazu w taki sposób, aby jego wymiary były akceptowalne przez sieć wykorzystaną jako ekstraktor cech charakterystycznych.

5.3. PODSUMOWANIE WYNIKÓW EKSPERYMENTU

W wyniku przeprowadzonego eksperymentu porównawczego wykazana została wyższość proponowanego w rozprawie rozwiązania zarówno nad podejściem tradycyjnym, wykorzystującym trening sieci konwolucyjnej od wag losowych jak i nad algorytmami opisywanymi w literaturze. Jednocześnie rozwiązanie to gwarantuje wyższą skuteczność wskazywania uszkodzeń niż inspektor mostowy w trakcie pracy manualnej (patrz dokładność wskazań inspektora – s. 1.1.). Testy przeprowadzone za pomocą zbioru ewaluacyjnego niepowiązanego ze zbiorem testowym udowodniły natomiast, że sieci trenowane z wykorzystaniem TL, pierwotnie uczone na zbiorach danych nie związanych z zadaniem, w którym mają być wykorzystywane, są w stanie generalizować wiedzę w stopniu pozwalającym na efektywne wykorzystywanie ich do precyzyjnej detekcji niewielkich uszkodzeń. **Potwierdza to tym samym pierwsze dwie tezy pomocnicze rozprawy.**

Ponadto, wykonane testy porównawcze wskazują użyteczność proponowanej metody w zastosowaniach praktycznych. W pierwszym z nich, sieć uczona w sposób tradycyjny wykazywała podobne metryki co proponowane w rozprawie rozwiązanie, lecz wymagała wielokrotnie więcej czasu i zasobów obliczeniowych do przeprowadzenia treningu. Aby algorytm trenowany od podstaw osiągnął dokładność porównywalną do algorytmów KrakN, w najlepszym przypadku (pojedynczy trening) musi być trenowany przez 10 godzin, co stanowi dwudziestokrotność czasu treningu z wykorzystaniem transfer learningu. Ponadto, wytrenowany model był zdecydowanie większy, przez co mniej praktyczny w przechowywaniu i wykorzystaniu na komputerach klasy biurowej (model trenowany od wag losowych zajmował ponad 500 MB przestrzeni dyskowej, podczas gdy model wygenerowany przez algorytmy KrakN, jedynie 55 MB).

Druga część eksperymentu wykazała, że proponowana w rozprawie metoda charakteryzuje się większą wszechstronnością niż inne, obecne w literaturze metody. Oprócz osiągania większej

zdolności do generalizacji nabytej w trakcie treningu algorytmu wiedzy, proponowana metoda wymaga także wykorzystania mniejszej ilości zasobów obliczeniowych. W przeciwieństwie do porównywalnych metod, dzięki swojej modułowej budowie i otwartemu dostępowi do kodu źródłowego, pozwala ona na wprowadzanie do niej modyfikacji, które dalszym stopniu będą poprawiać dokładność wskazań algorytmów lub rozbudują jego możliwości o wieloetapową klasyfikację uszkodzeń. Co więcej, dostarczone w repozytorium rozprawy gotowe metody pozwalają na budowę i trening własnych klasyfikatorów przez IM, bez posiadania rozległej wiedzy w tematyce uczenia maszynowego.

Z uwagi na powyższe, narzędzie inżynierskie wykonane na potrzeby rozprawy uznać można za użyteczne z punktu widzenia praktyki inspektora mostowego rozwiązanie. Spełnia ono bowiem wszystkie trzy warunki opisane w sekcji 2.2.3.4.:

- łatwość stosowania dzięki udostępnieniu metod do których wykorzystania nie potrzebna jest wiedza specjalistyczna z zakresu uczenia maszynowego,
- szybkość dzięki zastosowaniu techniki TL, wydatnie ograniczającej czas treningu algorytmu,
- brak konieczności wykorzystywania narzędzi specjalistycznych, a jedynie dowolnego aparatu cyfrowego.

W kolejnej części rozprawy przedstawiony zostanie dalszy rozwój prezentowanych metod, pozwalający na wdrożenie kolejnego stopnia automatyzacji pracy inspektora mostowego, przy zachowaniu ograniczonych kosztów wdrożenia usprawnień.

6. SZTUCZNA INTELIGENCJA W EKSTRAKCJI DANYCH Z OBRAZU

Opisywane do tej pory metody pozwalały na detekcję, a tym samym ilościowe zestawienie uszkodzeń obiektu pod postacią całkowitej liczby występowania instancji danego uszkodzenia na badanym elemencie. Podejście takie może znaleźć zastosowanie w przypadkach, w których całkowity wymiar uszkodzenia (nie mierzony w relacji do ogólnego wymiaru uszkodzonego elementu) nie jest istotnym elementem procedury nadania części konstrukcji jej oceny stanu technicznego. Przykładem takiego zastosowania może być przegląd podstawowy. Nie jest ono jednak wystarczające w pozostałych typach przeglądów, wymagających pomiaru fizycznych parametrów uszkodzeń.

Jednocześnie, obraz pozyskany przy pomocy aparatu cyfrowego, pozbawiony jest informacji pozwalających na nadanie fotografowanym obiektom wymiarów fizycznych. Jest on bowiem wyłącznie odwzorowaniem trójwymiarowej przestrzeni na rzutni będącej powierzchnią obrazu, w którym nieznane są parametry odwzorowania takie jak pozycja kamery względem odwzorowywanego punktu oraz orientacja kamery. Pozostałe, znane parametry (kamery i rzutni) nie są wystarczające do określenia cech fizycznych pozyskanego obrazu. Z tego powodu, niemożliwa jest ekstrakcja dodatkowych informacji o obrazowanych obiektach, bez modyfikacji metody uzyskiwania obrazu.

Mając na uwadze powyższe, opracowany został system modyfikujący metodę pozyskiwania obrazu z wykorzystaniem oryginalnie zaprojektowanego urządzenia pomiarowego, pozwalający na ekstrakcję cech fizycznych obrazu. Jego istotną częścią jest algorytm pozwalający na segmentację uszkodzenia z wykorzystaniem maszyn wektorów nośnych oraz wektoryzację uszkodzenia (rysy na powierzchni betonowej) w celu dokonania na nim precyzyjnych pomiarów. Całość proponowanej metody, dla uproszczenia została nazwana **SmartMeasure**.

6.1. ZAŁOŻENIA ROZBUDOWY PROPONOWANEGO ROZWIĄZANIA

W toku prac nad opisywanym w rozprawie rozwiązaniem, ustalone zostały założenia dotyczące systemu pozwalającego na ekstrakcję cech fizycznych uszkodzenia z obrazu cyfrowego. Podyktowane są one praktyką przeprowadzania przeglądów obiektów mostowych, dostępną technologią oraz kosztami budowy i ogólną dostępnością elementów rozwiązania. Projektowane rozwiązanie w założeniu miało spełniać także warunki użyteczności z punktu widzenia inspektora mostowego przedstawione w sekcji 2.2.3.4.

Główne założenia proponowanego rozwiązania SmartMeasure określono w następujący sposób:

- ciągłe działanie w czasie rzeczywistym,
- dokładność pomiaru obiektu na obrazie na poziomie 1 cm, w pomiarze prowadzonym z odległości nieprzekraczającej 20 cm,
- wykorzystanie powszechnie dostępnych aparatów cyfrowych w telefonach komórkowych,
- zapis informacji o fizycznej wielkości piksela na obrazie do standardu Exif (ang. Exchangeable Image File Format) [234], w chwili zapisu zdjęcia,
- możliwość wektoryzacji zidentyfikowanego uszkodzenia,
- jak najniższy budżet przeznaczony na zakup części i budowę urządzenia.

Wymienione założenia gwarantować mają tak użyteczność jak i możliwie największą dostępność proponowanego rozwiązania. Powinno ono być także proste w użyciu oraz oferować dokładność i powtarzalność pomiarów. Podobnie jak algorytm przedstawiony w rozdziale 4., nie powinno wymagać wiedzy specjalistycznej do obsługi.

Tym samym, należy zwrócić szczególną uwagę, projektowane rozwiązanie koncentrować się będzie na przeprowadzaniu wyłącznie pomiarów zgrubnych, służących do oszacowania ilościowego uszkodzenia. Z tego powodu, na tym etapie prac jako przypadek testowy wybrano długość rysy, a nie jej rozwartość. Przy czym zmiana rozwartości może zostać oszacowana pośrednio poprzez wykonanie wielokrotnego pomiaru całkowitej długości rysy w odstępach czasu. Chodzi o założenie, że jej dalsza propagacja może skutkować także zwiększeniem rozwartości.

Jednocześnie, dzięki udostępnieniu w rozprawie projektu budowy rozwiązania wraz z wykorzystywanymi przez nie algorytmami jest ono modyfikowalne. Pozwala to na dostosowanie go do specyficznych zastosowań, na przykład poprzez zmianę poszczególnych komponentów składowych lub części metod zawartych w algorytmie.

6.2. TECHNOLOGIE UMOŻLIWIAJĄCE POMIAR CECH FIZYCZNYCH OBIEKTU NA OBRAZIE

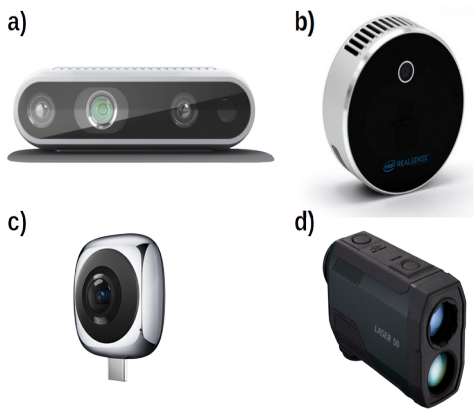
Jak zaznaczono na początku rozdziału, pomiar cech fizycznych, takich jak wymiary obiektu przedstawionego na obrazie cyfrowym utrwalonym przy pomocy konwencjonalnego aparatu cyfrowego, nie jest możliwe. Obraz taki pozbawiony jest wielkości opisujących cechy rzutu obiektu trójwymiarowego na płaszczyznę obrazu. Dlatego, aby wprowadzić do niego realne fizyczne wymiary, należy posłużyć się dodatkowymi technikami obrazowania, lub technikami wspomagającymi obrazowanie.

Istnieje wiele możliwości, dzięki którym uzyskać można powiązanie wymiarów fizycznych z wymiarami pojedynczego piksela, a co za tym idzie także obiektu (traktowanego jako zbiór pikseli) na obrazie. Wykorzystują one wiele różnych technik, dających rezultaty o zróżnicowanej dokładności, a gotowe systemy mogą osiągać koszty wielokrotnie przewyższające cenę aparatu cyfrowego. W trakcie prac nad proponowanym w rozprawie rozwiązaniem, rozważona została użyteczność i możliwość zaadaptowania do wykorzystania przez IM różnych rozwiązań. Opierały się one na wykorzystaniu zewnętrznych sensorów, specyfiki pracy kamery oraz zastosowaniu dodatkowych markerów w trakcie pomiaru.

Pierwszą rozważaną techniką było wykorzystanie dodatkowych sensorów w połączeniu z aparatem cyfrowym. Sensory te mogą wykorzystywać różne technologie, pozwalające na uzyskanie obrazu o znanej wielkości piksela wprost, trójwymiarowej chmury punktów lub długości celowej pozwalającej na obliczenie wielkości piksela. W zależności od sposobu pomiaru, ilości akumulowanych danych oraz dostępnego interfejsu komunikacji, występują jako niezależne urządzenia, lub urządzenia peryferyjne do zastosowania razem z aparatem lub smartfonem. Jako przykłady takich sensorów można wymienić:

- kamery z technologią wykrywania głębi na podstawie wizji stereoskopowej oraz rzucanej na obiekt siatki punktów w podczerwieni – np. Intel RealSense D435i [235] – Rys. 45, część a),
- kamery z technologią wykrywania głębi z wykorzystaniem skanowania laserowego sceny przy pomocy czujnika LiDAR – np. Intel RealSense L515 [235] – Rys. 45, część b),
- użycie dodatkowej pojedynczej kamery RGB komunikującej się ze smartfonem przy pomocy interfejsu USB, która umożliwi algorytmiczne uzyskanie wizji stereoskopowej do odczytu głębi obrazu – np. HUAWEI EnVizion [236] – Rys. 45, część c),
- dalmierz laserowy komunikujący się ze smartfonem poprzez łączność Bluetooth, umożliwiający pomiar długości celowej obiektywu głównego – np. Nikon LASER 50 Laser Rangefinder [237] – Rys. 45, część d).

Wykorzystanie zewnętrznego sensora jako narzędzia pomiarowego posiada szereg zalet, z których jako podstawowe wymienić można wysoką dokładność otrzymanego pomiaru, relatywnie niską trudność implementacji oraz możliwość użycia gotowego urządzenia. Wadami tego rozwiązania jest jednak wysoki koszt zewnętrznego sensora, który nie spełnia założenia o niskim koszcie rozwiązania. Charakteryzuje się też zamkniętą architektura i nie pozwala na modyfikację działania sensora w sposób algorytmiczny.



Rys. 45: Sensory i urządzenia peryferyjne umożliwiające ekstrakcję fizycznych wymiarów z obrazu

Kolejną techniką, której zastosowanie brano pod uwagę w rozprawie, jest wykorzystanie technik operowania kamerą, umożliwiających uzyskanie szeregu skorelowanych ze sobą obrazów. Przykładami takich technik jest między innymi zastosowanie zjawiska paralaksy oraz wykorzystanie metod fotogrametrycznych. Podejścia te są w pewnym stopniu do siebie podobne, gdyż oba wykorzystują zmianę położenia pojedynczej kamery w trakcie wykonywania powiązanych ze sobą zdjęć. W przypadku korzystania z paralaksy, otrzymywany jest obraz stereoskopowy, w którym bazą (odległością między dwoma punktami, obserwacji) jest przesunięcie poziome obiektywu między ujęciami. Jest ona jednak najczęściej wykorzystywana w zastosowaniach, w których możliwe jest użycie statywu mierzącego dystans między kolejnymi pozycjami kamery. Metody fotogrametryczne natomiast zakładają skorelowanie ze sobą wielu zdjęć tego samego obiektu, różniących się pozycją kamery. W następnym kroku, na bazie utworzonych zdjęć obliczane są kolejne pozycje kamery względem ujęć i na tej podstawie tworzona jest fotogrametryczna chmura punktów. Chmura ta jednak nie zawiera faktycznych wymiarów obiektu, a jedynie przedstawia jego elementy w rzeczywistej skali względem siebie. Do uzyskania wymiarów fizycznych wymagane jest podanie rzeczywistej odległości między dwoma, dowolnymi punktami chmury. Jest to także proces wymagający relatywnie dużych zasobów obliczeniowych, uniemożliwiających wykorzystanie w czasie rzeczywistym na smartfonie.

Ostatnią grupą technik, powszechnie wykorzystywaną do określania rzeczywistych wymiarów obiektów na dwuwymiarowym obrazie, jest wykorzystanie w utrwalanej scenie obiektów o znanym wymiarze. Obiektem takim jest najczęściej marker fotograficzny (fotopunkt), którego wymiar podany w zadanej jednostce jest zapisany w sposób umożliwiający automatyczny odczyt przez program do obróbki obrazów. Przykładem takiego markera może być kod QR (Rys. 46), zawierający informację o wielkości krawędzi kodu w formacie liczby naturalnej lub łańcucha znaków. Należy przy tym pamiętać, że fizyczna kopia markera musi posiadać deklarowane wymiary. Możliwe jest także ręczne wyskalowanie obrazu posługując się fizycznym obiektem znajdującym się na scenie, o wymiarach znanych IM. Wadami tych metod jest jednak konieczność fizycznego kontaktu z obiektem w przypadku stosowania markerów oraz podatność na błędy pomiarowe przy wykorzystaniu elementów

o znanej wielkości.–W tym przypadku różnica kilku pikseli na obrazie może mieć duży wpływ na pomiar, w szczególności jeśli docelowy obiekt znacznie różni się skalą od obiektu bazowego (np. duża różnica w skali między długością rysy, a długością przęsła obiektu).

Rys. 46: Przykład markera zawierającego informacje o swoich wymiarach

Mając na uwadze rozważane technologie oraz założenia dotyczące możliwości efektywnego rozbudowania proponowanego rozwiązania opisane w sekcji 6.1., jako metodę pomiaru wielkości piksela na obrazie wybrano wykorzystanie dodatkowego, zewnętrznego sensora. Aby jednak ograniczyć do minimum koszty rozwiązania, tym samym czyniąc je ogólnodostępnym dla IM, nie zostało wykorzystane gotowe urządzenie pomiarowe, a zamiast tego zaprojektowano i zbudowano oryginalny sensor do pomiaru długości celowej w trakcie wykonywania zdjęcia przy pomocy aparatu w smartfonie. Na podstawie tej zmierzonej wartości oraz znanych parametrów kamery, w sposób algorytmiczny jest możliwe obliczenie wielkości piksela na obrazie. Jedynym dodatkowym założeniem pomiaru wykonywanego w ten sposób jest zachowanie równoległości między mierzoną płaszczyzną, a płaszczyzną odwzorowania, gdyż wykonywany pomiar jest punktowy. Nie jest to jednak wymóg w sposób znaczny ograniczający przydatność metody.

6.3. PROJEKT I BUDOWA URZĄDZENIA POMIAROWEGO

6.3.1. WYKORZYSTANE KOMPONENTY I TECHNOLOGIE

Przy wyborze komponentów do sensora kierowano się założeniami dotyczącymi projektowanego urządzenia z sekcji 6.1. Uwzględniono deklarowaną przez producenta dokładność, rozdzielczość pomiaru, i cenę. Do projektu dodano także elementy ułatwiające prowadzenie pomiaru, nie będące jednak kluczowymi z punktu widzenia działania urządzenia pomiarowego. Wszystkie dobrane komponenty są ogólnodostępne na rynku. Wraz z handlową nazwą komponentu i funkcją pełnioną w urządzeniu pomiarowym zostały zestawione w Tab. 14.

Mikrokontroler A-Star zasilany jest przy pomocy napięcia o wysokości 5 V. Jest to wartość, którą można uzyskać wykorzystując port USB smartfona. Działa on z częstotliwością 16 MHz, co pozwala na uzyskanie liczby odczytów długości celowej znacznie przekraczającą potrzeby urządzenia pomiarowego. Z tego względu, aby zwiększyć czytelność odczytów oraz zmniejszyć zużycie energii

sensora, liczbę odczytów ograniczono w sposób algorytmiczny. Ultradźwiękowy czujnik odległości HC-SR04 charakteryzuje się zasięgiem od 2 do 200 cm. Mieści się więc w zakresie założonym w punkcie 6.1., przy czym jednocześnie dolna granica zasięgu nie stanowi przeszkody w odczycie, gdyż nie będzie wykorzystywana z uwagi na zakres ostrości aparatu. Czujnik pozwala na odczyt z rozdzielczością 0,1 cm.

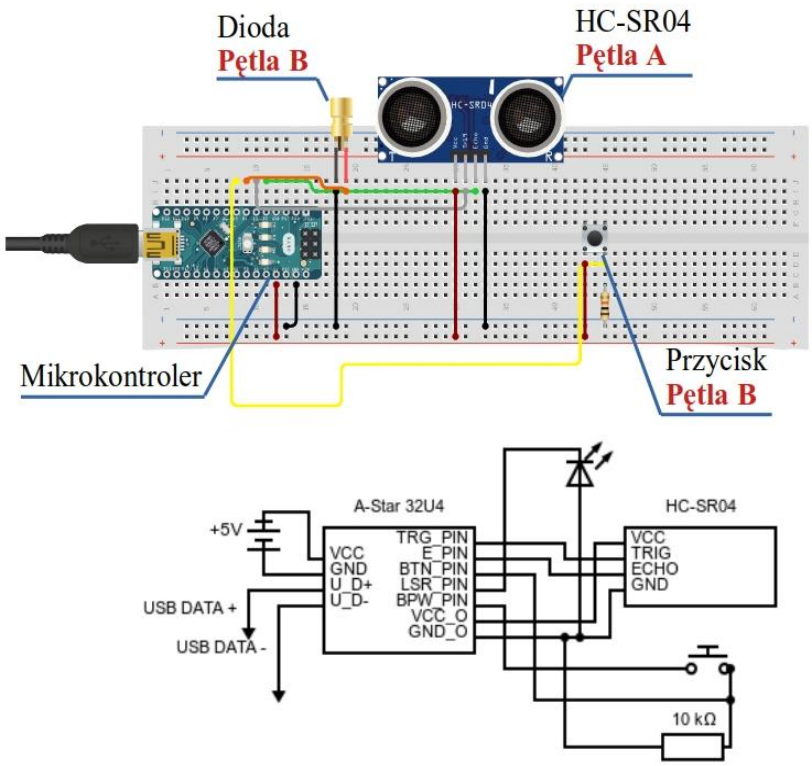
Tab. 14: Zestawienie komponentów projektowanego urządzenia pomiarowego

Komponent	Nazwa handlowa komponentu	Funkcja
Mikrokontroler	A-Star 32U4	Układ scalony zawierający mikrokontroler AVR Atmega32u4 służący do komunikacji z czujnikami peryferyjnymi oraz zarządzaniem elementami aktywnymi układu. Dzięki obsłudze portu SERIAL możliwa jest także komunikacja mikrokontrolera z wejściem USB smartfona.
Czujnik odległości	HC-SR04	Ultradźwiękowy czujnik odległości służący do pomiaru długości celowej w trakcie wykonywania zdjęcia z użyciem fali akustycznej o częstotliwości 40 kHz. Komunikuje się z mikrokontrolerem w sposób analogowy, natomiast przetworzenie sygnału analogowego na zmierzoną odległość odbywa się w sposób algorytmiczny.
Dioda laserowa	–	Dioda laserowa służy do informowania inspektora wykonującego pomiar o obranym celu. Może rzucać na mierzoną powierzchnię punkt lub znacznik geometryczny w formie np. krzyża, do wykorzystania jako dodatkowa informacja o zachowaniu równoległości płaszczyzny pomiaru i rzutowania.
Przycisk 4-pinowy	–	Przycisk służący do aktywacji i dezaktywacji diody laserowej – w zależności od zbioru treningowego użytego do treningu algorytmu wykorzystywanego do rozpoznawania defektów, obecność plamki lasera może mieć niekorzystny wpływ na wynik klasyfikacji.

Do programowania urządzenia pomiarowego oraz obsługującego go ze strony smartfona interfejsu wykorzystano dwa języki programowania. Język C – w odniesieniu do mikrokontrolera oraz Java – do aplikacji mobilnej dopisującej informacje o pomiarze celowej do danych Exif wykonanego zdjęcia. Szczegóły budowy i implementacji opisywanego urządzenia pomiarowego przedstawiono w sekcji 6.3.2. rozprawy.

6.3.2. BUDOWA I OPROGRAMOWANIE URZĄDZENIA

Budowę urządzenia pomiarowego rozpoczęto od wstępnego złożenia funkcjonalnego układu elementów na stykowej płytce prototypowej. W następnym kroku, po oprogramowaniu logiki operacji wykonywanych przez układ i zapisaniu jej w pamięci mikrokontrolera, sprawdzone zostało działanie układu ze względu na jego funkcjonalność. Na Rys. 47 przedstawiono wizualizację układu na płytce prototypowej oraz schemat połączenia elementów urządzenia pomiarowego zgodny z oznaczeniami zawartymi w opisie logiki i części F załącznika do rozprawy zawierającego kod źródłowy aplikacji. Ilustracja zawiera także wraz z oznaczenie dwóch wykonywanych w sposób ciągły przez układ pętli.



Rys. 47: Wizualizacja i schemat połączenia urządzenia pomiarowego

W projekcie nie zostały wykorzystane wszystkie dostępne połączenia cyfrowe i analogowe mikrokontrolera, więc możliwa jest dalsza rozbudowa układu o kolejne elementy. Na przykład o dodatkowe oświetlenie typu LED czy czujniki kontaktowe (np. czujnik wilgotności, pH). Jednocześnie, należy zaznaczyć, że dostępny na mikrokontrolerze procesor obsługuje jedynie pojedynczy wątek, natomiast w projekcie wykonywane są równolegle dwie pętle obliczeniowe (pętla A i B). Ten krok budowy urządzenia pomiarowego został podjęty, aby wykazać możliwość dalszej rozbudowy układu przy jednoczesnym zachowaniu dotychczasowej funkcjonalności. Na przykład poprzez algorytmiczną symulację działania procesora wielowątkowego. Jednocześnie, zastosowanie dwóch równolegle wykonywanych pętli obliczeniowych umożliwia niezależną od stanu odczytu odległości przez czujnik ultradźwiękowy obsługę celownika laserowego. Algorytm ten został opisany w kolejnej sekcji (6.3.2.1.), a jego ewentualna rozbudowa o dodatkowe czujniki może przebiegać w sposób analogiczny.

Głównymi elementami układu jest mikrokontroler A-Star 32u4 oraz ultradźwiękowy czujnik odległości HC-SR04. Ich wzajemna komunikacja realizowana jest w sposób cyfrowy, przy pomocy dwóch pinów (złącz punktowych) odpowiedzialnych za przesył i odczyt sygnału ultradźwiękowego *TRIG*, *ECHO* oraz dwóch pinów zasilających – *VCC* i *GND*. Pin *TRIG* odpowiada za generowanie sygnału ultradźwiękowego na podstawie informacji uzyskanej z mikrokontrolera. Następnie, sygnał odbity od powierzchni wraca do czujnika, a jego wystąpienie odczytywane jest na pinie *ECHO*. Na

podstawie różnicy w czasie między wystąpieniem wartości wysokich na tych dwóch pinach, w mikrokontrolerze odczytywana jest odległość od mierzonej powierzchni.

Na Rys. 47 przedstawiony został także sposób połączenia zaprojektowanego układu z urządzeniem zewnętrznym. Realizowane jest ono poprzez połączenie przez port Serial USB w standardzie 2.0 (występują jedynie szyny danych Data + i Data –). Przy pomocy tego połączenia wykorzystać można zaprojektowane urządzenie tak ze smartfonem, jak i dowolnym innym urządzeniem obsługującym port WE/WY (typu serial).

6.3.2.1. IMPLEMENTACJA LOGIKI OBSŁUGI URZĄDZENIA POMIAROWEGO

Jak zaznaczone zostało w poprzedniej sekcji – w logice obsługującej zaprojektowane urządzenie pomiarowe symulowano równoległą pracę dwóch wątków logicznych procesora, tak aby możliwe było wykonanie dwóch niezależnych operacji w jednym cyklu pracy pętli głównej mikrokontrolera. W tym celu zastosowano metody programistyczne wykorzystujące protowątki (ang. protothreads) [238], dodające operacjom w mniejszym stopniu obciążającym procesor dodatkowe warunki logiczne, pozwalające na pominięcie ich w trakcie wykonywania głównej pętli programu. Działanie pętli głównej, podzielonej na pętle A oraz B przedstawiono na Rys. 48.

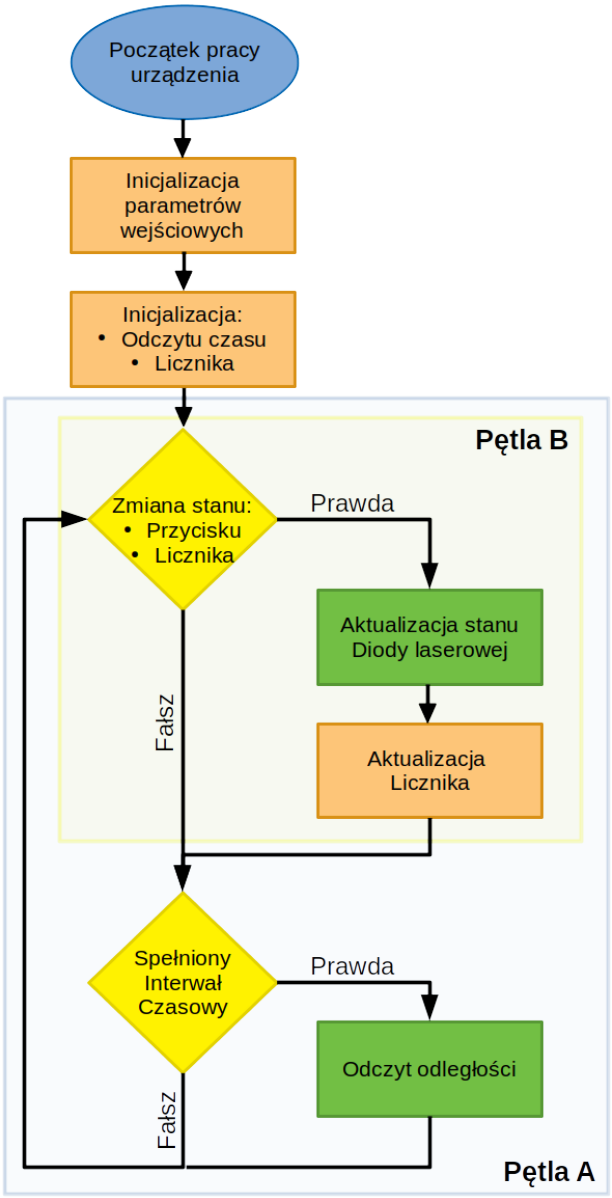
Rys. 48 przedstawia przepływ pracy wykonywanej w pętli ciągłej przez mikrokontroler A-Star. Algorytmiczny opis tej logiki umieszczony został w części F załącznika do rozprawy.

W pierwszej kolejności inicjalizowane są stałe pracy urządzenia, takie jak numery pinów odpowiadających elementom układu oraz parametr określający zakres dokładności pomiaru czujnikiem ultradźwiękowym, a także przypisywane są tryby pracy połączeń (jako wejścia i wyjścia sygnałów). Następnie inicjalizowane są zmienne zarządzające przebiegiem głównej pętli wykonywanej przez urządzenie.

Główna pętla wykonywana przez układ składa się z dwóch części – pętli A oraz B. Jak wspomniano wcześniej, w celu ograniczenia ilości operacji wykonywanych przez urządzenie, przed wykonaniem obu z pętli sprawdzane są warunki logiczne. Pozwala to na symulowanie działania dwóch równoległych wątków logicznych. W przypadku pętli A sprawdzany jest pojedynczy warunek – spełnienie interwału czasowego (pomiar wykonywany jest co pół sekundy), a pętla wykonywana jest niezależnie od wyniku warunku logicznego wykonania pętli B. Pętla B natomiast wykonywana jest jedynie, jeśli spełnione są oba warunki: naciśnięty został przycisk obsługujący diodę laserową oraz aktualizowana została zmienna licznika. Dzięki jej zastosowaniu, nie jest możliwe wykonywanie pętli B w przypadku gdyby przycisk pozostawał cały czas wciśnięty.

Przy każdym przebiegu pętli A, zmierzona wartość jest zapisywana w pamięci mikrokontrolera i zwracana przez port serial, jeśli zostanie wysłane przez niego zapytanie z urządzenia

zewnętrznego. Logika obsługująca urządzenie zewnętrzne wykorzystująca pomiar z urządzenia, została opisana w kolejnej sekcji rozprawy.



Rys. 48: Opis logiki działania urządzenia pomiarowego

6.3.2.2. *Implementacja oprogramowania aplikacji mobilnej*

Na potrzeby powiązania pomiaru długości celowej z wielkością piksela na obrazie, do zaprojektowanego urządzenia pomiarowego napisana została aplikacja mobilna, wykorzystująca wykonany pomiar w trakcie zapisu obrazu. Aplikacja zapisuje zmierzoną w centymetrach odległość do metadanych obrazu dołączanych jako powszechny standard Exif.

Metadane w formacie Exif zawierają między innymi informacje na temat rozdzielczości obrazu, czasu ekspozycji, ogniskowej obiektywu oraz wielkości obrazu w pionie i poziomie. Przy

pomocy zaprojektowanego urządzenia pomiarowego, oraz aplikacji mobilnej uzupełniane jest pole Exif „*SubjectDistance*", pozostające najczęściej puste, jeśli aparat przy pomocy którego wykonywane było zdjęcie nie posiada wbudowanego czujnika odległości. Dane te, pozwalają następnie na wyznaczenie w sposób algorytmiczny wielkości obiektu na obrazie w centymetrach, przy pomocy następującego wzoru [239]:

$$Wielkość\ obiektu = \frac{dist \cdot sensor_{real} \cdot obj_d}{sensor_{px} \cdot f} \quad , \tag{10}$$

gdzie:

- *dist* – długość celowej zmierzona przy pomocy urządzenia pomiarowego,
- $sensor_{real}$ – wielkość sensora w jednostce fizycznej (mm),
- obj_d – wielkość obiektu na obrazie w pikselach,
- $sensor_{px}$ – wielkość sensora w pikselach,
- *f* – ogniskowa.

Tym samym, wzór obliczający wielkość pojedynczego piksela na obrazie przybiera postać:

$$Wielkość\ piksela = \frac{dist \cdot sensor_{real}}{sensor_{px} \cdot f} \tag{11}$$

Jednocześnie, należy zwrócić uwagę na konieczność doboru korespondujących ze sobą wartości w trakcie przeprowadzania obliczeń. Wielkość sensora tak wyrażona w pikselach jak w milimetrach powinna opisywać ten sam wymiar sensora (pionowy lub poziomy). Ponadto, wielkości te mogą nie być ujęte w metadanych Exif obrazu. W takim przypadku należy ręcznie odszukać je w specyfikacji dostarczonej przez producenta sensora w aparacie.

Fragment kodu źródłowego aplikacji mobilnej komunikującej się z urządzeniem pomiarowym i obliczającej wielkość piksela na rejestrowanym obrazie został dołączony do rozprawy w części G załącznika.

6.3.3. TESTY DOKŁADNOŚCI URZĄDZENIA POMIAROWEGO

Gotowe urządzenie pomiarowe zamknięte w obudowie wytworzonej w technologii druku 3D z tworzywa ABS (powszechnego tworzywa polimerowego) zostało przedstawione na Rys. 49. Przedstawiony został także sposób stabilnego połączenia z obudową smartfona, zrealizowany przy pomocy płytki magnetycznej.

Rys. 49: Urządzenie pomiarowe wraz ze sposobem połączenia ze smartfonem

Testy urządzenia pomiarowego obejmowały porównanie wskazywanej przez urządzenie odległości w odniesieniu do profesjonalnego dalmierza laserowego (PRO DL-100) na dystansie od 15,0 do 60,0 cm w kroku wynoszącym 5,00 cm w zakresie 15,0 – 30,0 cm i 10,0 cm w zakresie 30,0 – 60,0 cm. W celu ułatwienia prowadzenia pomiaru i zapewnienia powtarzalności wyników między kolejnymi pomiarami, wykonana została specjalna obejma podtrzymująca oba urządzenia i utrzymująca je w pozycji pionowej. Na szynach łączących tablicę z zaznaczonym celem oraz platformę na której znajdowały się urządzenia pomiarowe naniesiona została podziałka z krokiem długości 5,00 cm. Obejmę wraz z porównaniem przykładowych wyników pomiaru przedstawia Rys. 50.

Rys. 50: Przykładowe pomiar urządzeniem pomiarowym i dalmierzem laserowym

Testy dokładności urządzenia pomiarowego obejmowały przeprowadzenie pięciu kolejnych pomiarów w każdej z wyznaczonych odległości (w sumie przeprowadzono 35 pomiarów przy pomocy obu urządzeń). Wyniki pomiarów, wraz z uśrednionym błędem pomiaru zaprojektowanego urządzenia zostały przedstawione w Tab. 15. W tabeli wyszczególniono także dystans 20,0 cm, dla którego obliczona została wielkość obserwowanego piksela, zgodnie z założeniami opisanymi w sekcji 6.1.. Należy jednak mieć na uwadze, iż nie jest to dokładność przeprowadzonego opisywaną metodą pomiaru. Wielkość ta zostanie wyznaczona w sekcji 6.5. na podstawie pomiarów faktycznych uszkodzeń powierzchni betonu z wykorzystaniem metody segmentującej obraz uszkodzenia w sposób semantyczny.

Tab. 15: Błąd pomiaru zaprojektowanego urządzenia

Dystans [cm]	**Średni błąd pomiaru [%]**
15,0	0,6
20,0	**0,6**
25,0	0,7
30,0	0,7
40,0	0,8
50,0	0,9
60,0	1,1

Jak wynika z pomiarów zapisanych w Tab. 15, średni błąd pomiaru prowadzonego przy pomocy zaprojektowanego urządzenia pomiarowego, rośnie od 0,6% dla odległości 15,0 cm do 1,1% na dystansie 60,0 cm. Widać tym samym, iż błąd ten uzależniony jest od odległości do celu, sięgając wartości średniej wynoszącej 0,66 cm dla odległości 60,0 cm. Należy jednak zauważyć, że z uwagi na możliwości obrazowania przy pomocy kamer aparatów w smartfonach, dystans na którym prowadzony jest pomiar nie powinien przekraczać 30,0 cm. W rozprawie założono dystans, przy dystansie 20,0 cm jako optymalny do wykonywania pomiarów.

W pomiarach wykonywanych na potrzeby rozprawy, posługiwano się smartfonem Xiaomi Redmi Note 2, wykorzystującym sensor optyczny **Samsung S5K3M2** oraz optykę o następujących parametrach:

- poziomy wymiar fizyczny sensora: $sensor_{real}$ = 4,69 mm,
- poziomy wymiar sensora w pikselach: $sensor_{px}$ = 4160 px,
- ogniskowa: f = 3,50 mm.

Powyższe parametry pozawalają na uzyskanie teoretycznej wielkości fizycznej pojedynczego piksela o wymiarze na obrazie:

$$Wielkość\ piksela = \frac{200\,mm \cdot 4{,}69\,mm}{4160 \cdot 3{,}50\,mm} = 0{,}064\,mm \tag{12}$$

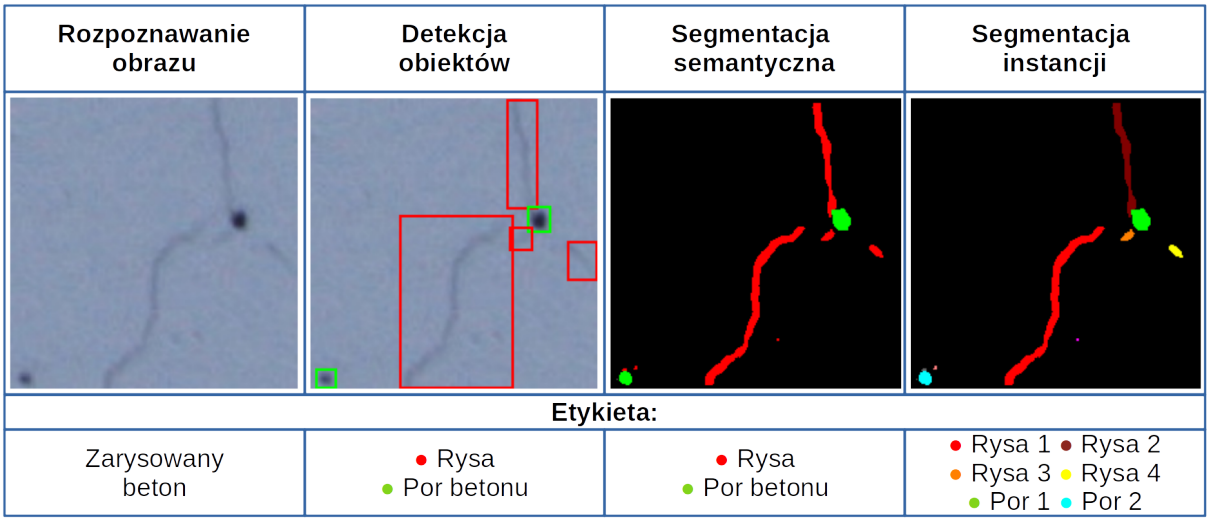
oraz uzyskanie gęstości pikseli na obrazie na poziomie:

$$Gęstość\ obrazu = \frac{1}{Wielkość\ piksela} = \frac{1}{0{,}064} = 15{,}52\,\frac{px}{mm} \tag{13}$$

6.4. Ekstrakcja fizycznych cech obrazu z wykorzystaniem maszyn wektorów nośnych

Kolejnym krokiem po wyznaczeniu fizycznej wielkości piksela na zarejestrowanym obrazie, jest wydzielenie z niego wyłącznie tych pikseli, które zawierają się w obiekcie oznaczonym jako uszkodzenie. Umożliwi to pomiar parametrów fizycznych rozpoznanego uszkodzenia, w usprawniając tym samym dalsze zarządzanie nim.

Wydzielenie z obrazu pikseli zawierających wyłącznie pojedynczą klasę obiektów to problem segmentacji semantycznej. W przypadku, w którym poszczególne instancje danej klasy rozpatrywane są jako pojedyncze obiekty jest to segmentacja instancji. Przykład takiej operacji przedstawiono na Rys. 51, gdzie na przykładzie analizy powierzchni betonu porównano ją także z omawianymi wcześniej zagadnieniami rozpoznawania obrazu oraz detekcji obiektu na obrazie.

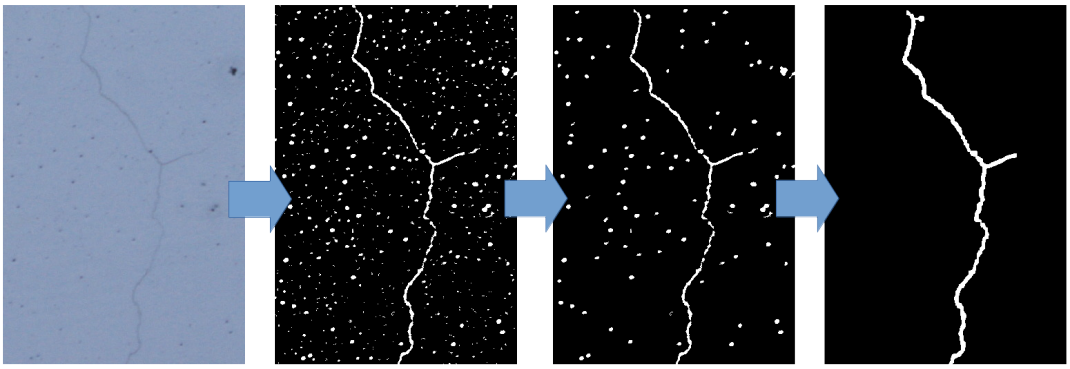


Rys. 51: Różnice między rozpoznawaniem obrazu, detekcją obiektów, segmentacją semantyczną i segmentacją instancji

Z uwagi na charakter rozpatrywanego w rozprawie uszkodzenia, wobec którego nie jest istotne odróżnienie od siebie przenikających się instancji, które traktować można wspólnie jako pojedyncze uszkodzenie, zdecydowano się na wykorzystanie segmentacji semantycznej. W tym celu utworzony został nowy moduł opracowywanego systemu, bazujący na algorytmie rozpoznawania kształtów z wykorzystaniem maszyn wektorów nośnych.

Maszyny wektorów nośnych to algorytm płytkiego uczenia maszynowego, opracowany na początku lat sześćdziesiątych XX wieku, a następnie zmodyfikowany w latach dziewięćdziesiątych poprzez zastosowanie transformacji punktów danych do wysoce wielowymiarowej przestrzeni (ang. kernel trick). Dzięki temu zabiegowi stało się możliwe rozwiązywanie problemów nieliniowych (patrz Rys. 8), będących niemożliwymi do rozwiązania przez oryginalną wersję metody [240].

Maszyny wektorów nośnych zastosowano tutaj jako algorytm rozpoznawania kształtu binarnej maski utworzonej na bazie posegmentowanego z wykorzystaniem klasycznych metod wizji komputerowej obrazu. W pierwszej kolejności obraz wejściowy jest normalizowany i nakładany jest na niego filtr rozmycia Gaussa. Następnie wykorzystywany jest adaptacyjny filtr progowy, oznaczający na obrazie wszystkie elementy, które wyraźnie odróżniają się od tła jako binarne maski, a najmniejsze obiekty będące szumem są usuwane. W ostatnim kroku, maski elementów służą jako wektor wejściowy do modelu maszyny wektorów nośnych, która na podstawie kształtu elementu decyduje czy jest to uszkodzenie w postaci rysy, czy np. wyłącznie zabrudzenie powierzchni. Przykład takiej operacji przedstawiony został na Rys. 52, natomiast sam algorytm został dołączony do rozprawy w pierwszej połowie kodu części H załącznika.



Rys. 52: Segmentacja obrazu z wykorzystaniem maszyn wektorów nośnych

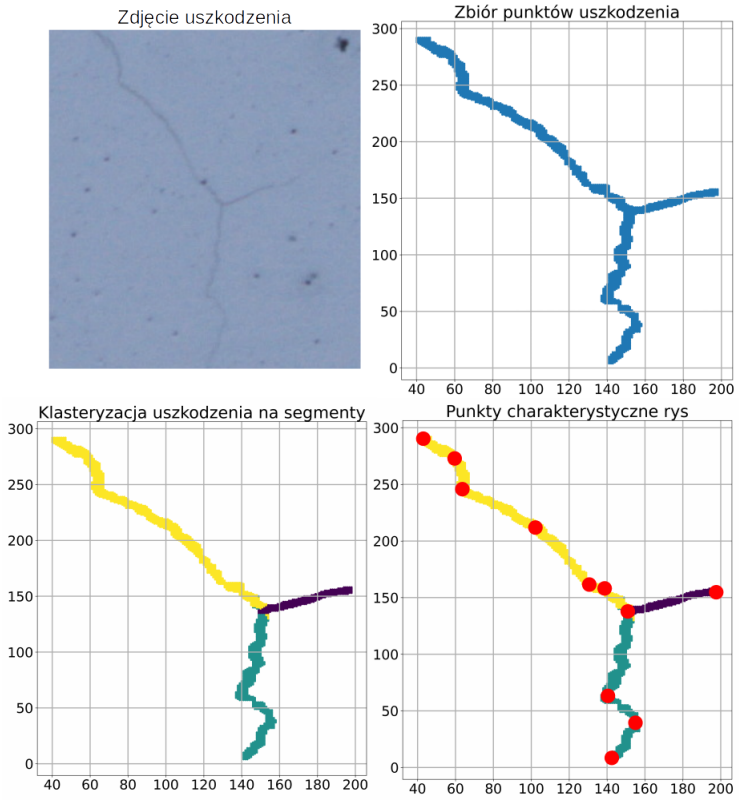
6.4.1. ZAPIS USZKODZEŃ W FORMIE WEKTOROWEJ

Kolejnym krokiem po segmentacji semantycznej uszkodzenia, jest jego wektoryzacja. Jest ona przeprowadzana w celu umożliwienia pomiaru uszkodzenia oraz dalszego nim zarządzania poprzez np. przeniesienie bezpośrednio do modelu BIM lub zapis charakterystyk uszkodzenia w innej bazie danych.

W celu wektoryzacji rysy utworzono dodatkowy algorytm, wykorzystujący metody uczenia maszynowego nienadzorowanego (klasteryzacja danych), regresję liniową oraz regresję wielomianową. W pierwszym kroku segment zawierający piksele uszkodzenia zapisywany jest jako dyskretny rozkład punktów o znanych współrzędnych. Punkty te odniesione są do układu, w którym

znajduje się obraz i odpowiadają jego pikselom. W kolejnym kroku model mieszany Gaussa wykorzystywany jest do uzyskania oddzielnych klastrów uszkodzenia, zawierających jego wyodrębnione części – odnogi rys. Liczba klastrów ustalana jest na podstawie zmian wielkości inercji (oceny jakości modelu segmentacyjnego, mierzonej jako suma kwadratów odległości punktów od centrum klastra). Z pomocą tej miary podział na liczbę klastrów przerywany jest, gdy jej zmiana nie przekracza 0,2 między kolejnymi iteracjami.

Po podziale na klastry, na każdym z nich przeprowadzana jest regresja liniowa. Dzięki niej określany jest kierunek główny odnogi uszkodzenia, a także jego punkt początkowy i końcowy w nowym układzie odniesienia. Układ ten to transformowany liniowo o kąt nachylenia prostej regresji układ pierwotny. Następnie punkty uszkodzenia służą jako dane wejściowe do modelu regresji wielomianowej dla stopni wielomianu od 3 do 10. Ostateczny wybór stopnia wielomianu dokonywany jest na podstawie miary współczynnika determinacji. Po przeprowadzeniu regresji i uzyskaniu wykresu ogólnego przebiegu rysy na elemencie betonowym, oznaczane są na nim punkty charakterystyczne przebiegu w dziedzinie wyznaczonej przez punkty początkowe i końcowe uszkodzenia. W ostatnim kroku uzyskane szeregi wektorów z każdego z klastrów łączone są w jeden zapis uszkodzenia. Etapy tego algorytmu w formie rysunkowej zostały przedstawione na Rys. 53, natomiast jego zapis w języku Python został dołączony do rozprawy w części H załącznika.

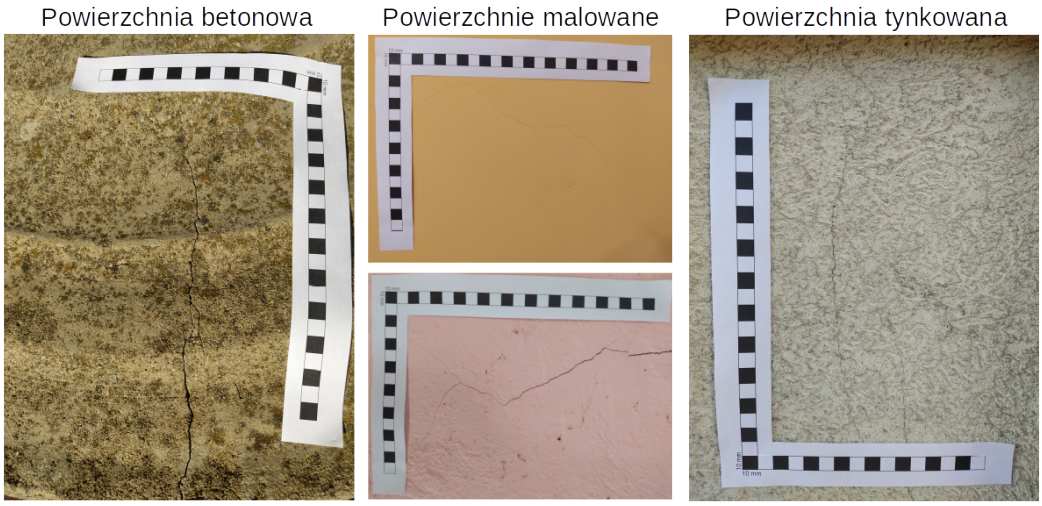


Rys. 53: Kolejne kroki algorytmu wektoryzacji rys

Po uzyskaniu uszkodzenia w zapisie wektorowym jest możliwy pomiar jego długości poprzez proste sumowanie kolejnych segmentów uszkodzenia wyznaczonych przez wektory. Następnie, przy znanej wielkości fizycznej pojedynczego piksela na obrazie uzyskać można rzeczywistą wielkość uszkodzenia.

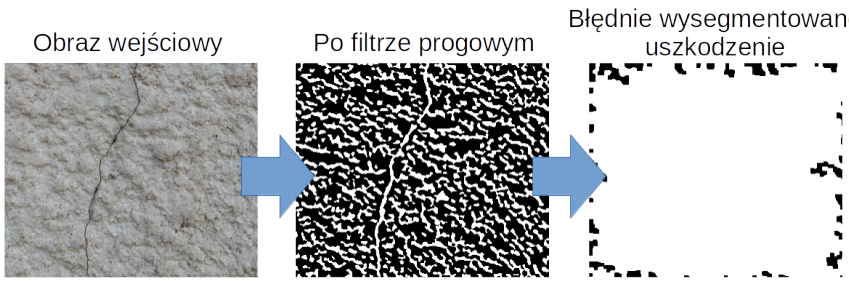
6.5. OCENA DOKŁADNOŚCI NARZĘDZIA SMARTMEASURE

Ocenę dokładności opisywanego rozwiązania przeprowadzono na podstawie porównania pomiaru wykonanego przy pomocy proponowanej metody do pomiaru ręcznego wykonanego z dokładnością do 1,00 mm. Metodę weryfikowano z wykorzystaniem 21 zdjęć zawierających trzy typy powierzchni, począwszy od malowanych, przez betonowe, a skończywszy na tynku fakturowanym, których zdjęcia wykonano z odległości nieprzekraczającej 20,0 cm. Na obrazach, dodatkowo w celu weryfikacji pomiaru, zamieszczono także fizyczną skalę z podziałką 10,0 mm. Przykłady obrazów na których weryfikowano działanie algorytmu zostały zamieszczone na Rys. 54.



Rys. 54: Przykłady obrazów użytych do weryfikacji dokładności metody

Z uwagi na zauważalne różnice w dokładności proponowanej metody w zależności od typu powierzchni, na której została wykorzystana, dokładność została obliczona dla każdego typu powierzchni osobno. Wyniki pomiaru dokładności metody zamieszczono w Tab. 16. Jednocześnie należy zaznaczyć, iż proponowana metoda nie była w stanie w sposób poprawny dokonać segmentacji semantycznej uszkodzeń na powierzchni wykończonej tynkiem fakturowym. Jest to spowodowane ilością powiązanych obiektów na powierzchni elementów będących cechami charakterystycznymi powierzchni otynkowanej. Z uwagi na to, nie jest możliwy pomiar wielkości uszkodzenia na tego typu powierzchni. Przykład segmentowanego obrazu po zastosowaniu adaptacyjnego filtra progowego oraz błędnie wysegmentowanego uszkodzenia przedstawiono na Rys. 55.



Rys. 55: Przykład błędnie wysegmentowanego uszkodzenia

Tab. 16: Średnie wartości błędy pomiaru proponowanej metody w zależności od typu powierzchni

Typ powierzchni	**Średni błąd pomiaru długości rysy[mm]**
Malowana	< 1,00
Betonowa	3,00
Tynkowana	**– (nie uzyskano wysegmentowanego uszkodzenia)**

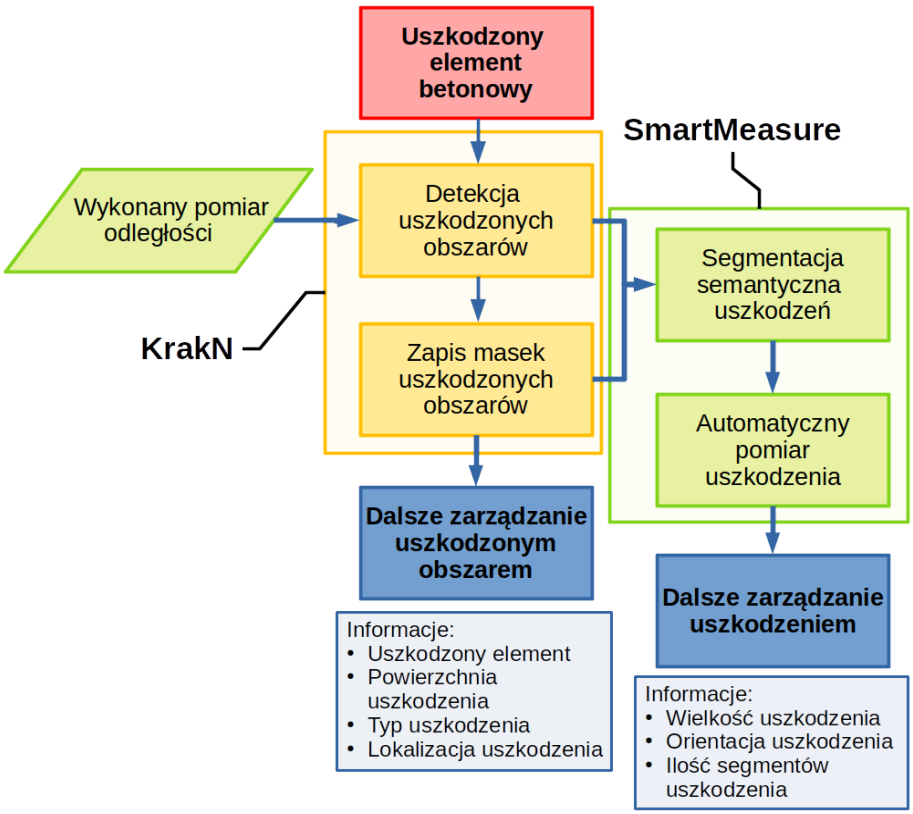
W Tab. 16, najmniejszy, nieprzekraczający 1,00 mm błąd metody został uzyskany przy pomiarze uszkodzeń powierzchni malowanych. Jest to wynikiem najmniejszej liczby nierówności powierzchni, zanieczyszczeń i innych charakterystycznych cech na powierzchni, przez co adaptacyjny filtr progowy nie łączył nie powiązanych ze sobą semantycznie obszarów. Większy błąd uzyskany na niemalowanej powierzchni betonowej, wynika on pośrednio z obecności na powierzchni porów, ale także z obliczeń uśrednionego błędu pomiaru dla różnych typów powierzchni betonowych. W testach rozwiązania wykorzystano zarówno czyste powierzchnie betonu architektonicznego, jak również stare powierzchnie betonowe wykonane z betonu żwirowego z widocznymi odpryskami oraz ziarnami kruszywa (patrz „Powierzchnia betonowa” na Rys. 54). Dla pomiarów wykonanych na betonie architektonicznym, błąd pomiaru był zbliżony do błędu uzyskanego na powierzchniach malowanych i także nie przekraczał 1,00 mm.

Tym samym spełnione zostało założenie dotyczące dokładności prowadzonych pomiarów, określone w sekcji 6.1. rozdziału.

6.6. Ocena przydatności rozwiązania

W rozdziale 6. rozprawy przedstawiona została koncepcja i wstępna implementacja systemu SmartMeasure, będącego dalszą rozbudową opisywanego w rozdziale 4. pakietu KrakN i umożliwiającego pomiar cech fizycznych uszkodzeń powierzchni betonowych. Składa się on z dodatku będącego peryferyjnym dla smartfona urządzeniem pomiarowym oraz współpracującym ze sobą oprogramowaniem mikrokontrolera, aplikacji mobilnej oraz aplikacji komputerowej. Oprócz projektu budowy, zastosowania i oprogramowania systemu, w sekcji 6.1. opisano także ogólne założenia dotyczące projektowanego dodatku.

Na Rys. 56 przedstawiony został proponowany przepływ pracy zastosowania pakietu KrakN w połączeniu z systemem SmartMeasure do wykrywania, pomiaru i przekazania do dalszego zarządzania uszkodzeń obiektów budowlanych. W pierwszej kolejności, przy wykorzystaniu pakietu KrakN dokonywana jest ocena jakościowa elementu poprzez wykrywane i identyfikowane uszkodzenia na podstawie obrazów. W drugiej kolejności, przy pomocy dodatku SmartMeasure przeprowadzana jest ocena ilościowa uszkodzeń – są one mierzone, a informacje o przeprowadzonym pomiarze dodawane są do danych Exif obrazu.



Rys. 56: Przepływ pracy rozwiązania proponowanego w rozprawie

Wykorzystując oba zaprezentowane w rozprawie rozwiązania możliwe jest w pierwszym kroku zautomatyzowanie wykrywania uszkodzeń obiektów budowlanych w trakcie prowadzenia przeglądu przez IM. Rozszerzenie metod umożliwiających detekcję uszkodzeń o zaproponowany system SmartMeasure, umożliwia w drugim kroku automatyczny pomiar cech fizycznych wykrytego uszkodzenia przy pomocy automatyzacji pomiaru długości celowej i wykorzystaniu technik wizji komputerowej, **dowodząc tym samym trzeciej tezy pomocniczej rozprawy**.

Jednocześnie, proponowany system rozszerzający funkcjonalność aparatu fotograficznego w smartfonie o pomiar długości celowej cechuje się znacznie niższym kosztem implementacji w porównaniu do innych systemów opisanych w sekcji 6.2. Jest przy tym wystarczająco dokładny i prosty w użyciu, aby był użyteczny przy prowadzaniu pomiarów w trakcie podstawowego i

rozszerzonego przeglądu technicznego obiektu mostowego. Spełnia także założenia dotyczące użyteczności urządzenia inżynierskiego, opisane w sekcji 2.4. rozprawy.

7. PODSUMOWANIE I WNIOSKI

7.1. PODSUMOWANIE

Przedmiotem niniejszej pracy było opracowanie propozycji systemu mogącego przyczynić się do zwiększenia efektywności pracy inspektora mostowego. W badaniach skupiono się na uszkodzeniu powierzchni betonowych w postaci rys lub pęknięć. Występują one stosunkowo często w konstrukcjach żelbetowych, natomiast w elementach sprężonych mogą świadczyć o poważnym zagrożeniu. Już nie tylko trwałości, ale nawet bezpieczeństwa konstrukcji. Rysy o niewielkiej rozwartości są zwykle trudne do wskazania i identyfikacji podczas tradycyjnych inspekcji wizualnych. Dlatego zdecydowano się skoncentrować właśnie na nich. Jednak uniwersalność i elastyczność zaproponowanego rozwiązania pozwala zastosować je również w przypadku innych defektów na betonowych powierzchniach, jak np. zabrudzeń, ubytków, korozji, wykwitów itp. Wystarczy do tego utworzyć stosowny zbiór danych uczących.

W pierwszej części rozprawy przedstawiono obszerny przegląd literatury poświęcony metodom oceny stanu technicznego obiektów mostowych. Na początku opisano metody wykonywania inspekcji na świecie powołując się na krajowe regulacje. Następnie zostały opisane inspekcje przeprowadzane w Polsce na podstawie instrukcji GDDKiA. Są one wykorzystywane powszechnie także przez lokalne jednostki zarządzające infrastrukturą. Opisany został sposób przeprowadzania inspekcji, a także nadawania oceny punktowej poszczególnym elementom obiektu i całej konstrukcji. Dzięki analizie literatury wykazano liczne podobieństwa między stosowanymi w różnych częściach świata instrukcjami przeprowadzania inspekcji oraz wskazano na podstawowe cechy rozwiązania użytecznego inżyniersko, które mogłoby przyczynić się do poprawy efektywności inspektora mostowego. Ocena literatury została podsumowana krótkim opisem najnowszych rozwiązań w dziedzinie inspekcji obiektów mostowych Spośród nich, na uwagę zasługują dynamicznie rozwijające się metody wykorzystujące wizję komputerową oraz metody sztucznej inteligencji. I to właśnie na te dwie grupy metod położony został największy nacisk w całej rozprawie.

Kolejna część pracy poświęcona została opisowi metod sztucznej inteligencji wraz z jej podziałem na poddziedziny, takie jak uczenie maszynowe i głębokie uczenie. Rozdział rozpoczyna się ogólnym opisem historii rozwoju koncepcji sztucznej inteligencji, a następnie samych metod SI. W dalszej części opisane zostały algorytmy sztucznych sieci neuronowych pod kątem ich budowy,

treningu oraz stosowania. W podobny sposób opisany został wariant sztucznych sieci neuronowych wykorzystywany w zagadnieniach wizji komputerowej – konwolucyjne sieci neuronowe. Przedstawiono elementy składające się na kolejne warstwy sieci konwolucyjnej, takie jak warstwy konwolucyjne, aktywacyjne i łączące oraz sposób ich połączenia w pełną architekturę. Opisano także problemy dotyczące treningu tego typu algorytmów oraz metody wspomagające trening. Rozdział zawiera także analizę literatury w dziedzinach łączących inżynierię budowlaną oraz metody sztucznej inteligencji, także w połączeniu z wizją komputerową. Ocena literatury dowodzi istnienia pola badawczego, które nie zostało jeszcze wypełnione w dziedzinie wspomagania przeprowadzania przeglądów obiektów mostowych. Wnioski płynące ze studium literatury pozwoliły na ukierunkowanie dalszych prac podejmowanych w rozprawie.

Na tej podstawie zaproponowano nowe narzędzie inżynierskie jako pakiet algorytmów KrakN, wspomagających pracę inspektora mostowego w wykrywaniu niewielkich uszkodzeń powierzchni betonowych. Wyszczególniono założenia proponowanej metody, spośród których zwrócić uwagę należy na zastosowanie rozpoznawania obrazu z wykorzystaniem konwolucyjnej sieci neuronowej w połączeniu z algorytmem przesuwającego się okna jako detektora obiektów. Dzięki temu, proponowane rozwiązanie nie wymaga zastosowania kosztownej infrastruktury obliczeniowej, gwarantując jednocześnie wysoką dokładność predykcji. W dalszym ciągu, z myślą o skalowalności rozwiązania, wybrano główną architekturę sieci konwolucyjnej poprzez przeprowadzenie testów wydajności popularnych architektur. Aby w większym stopniu ograniczyć zapotrzebowanie na zasoby obliczeniowe proponowanego rozwiązania, wykorzystano alternatywną metodę treningu algorytmu. Rozważono przy tym wykorzystanie metod *fine tuning* i *transfer learning*, spośród których ostatecznie wybrano *transfer learning* jako metodę o mniejszych wymaganiach obliczeniowych. Opisane zostały także metryki oceny dokładności algorytmów predykcyjnych oraz sposób ich interpretacji wraz z przykładem uzasadniającym równoczesne stosowanie różnych i uzupełniających się metryk.

W dalszej części rozprawy zaprezentowano sposób budowy i implementacji pakietu autorskich algorytmów KrakN. W opisie uwzględniono środowisko programistyczne, wykorzystane biblioteki oraz systemy operacyjne wspierane przez proponowane rozwiązanie. W praktycznej części implementacji algorytmów opisano metody kolekcji danych treningowych algorytmu spełniające założenia wstępne rozprawy (miarę rozwartości rysy) oraz sposoby sztucznego ich wzbogacania z użyciem metod *Data Augmentation*. Zaprezentowano także utworzone narzędzie wspomagające podział całego zbioru danych na podzbiory zawierające klasy uszkodzenia. Opis implementacji proponowanego rozwiązania zawiera także dokładne przedstawienie algorytmów składających się na pakiet KrakN w postaci graficznej. Proponowane rozwiązanie zostało także sprawdzone pod kątem dokładności predykcji i obliczone zostały jego metryki. Przedstawiono także możliwości rozbudowy rozwiązania, o dodatkowe funkcjonalności.

Kolejna część rozprawy zawiera dwa eksperymenty porównawcze, dowodzące użyteczności proponowanego rozwiązania. Pierwszy z nich porównuje algorytm sieci neuronowej trenowanej z wykorzystaniem transfer learningu do sieci uczonej od wag losowych. Oprócz metryk obu rozwiązań porównano także czas potrzebny na trening algorytmu, odniesiony do różnego typu jednostek obliczeniowych. W wyniku pierwszego porównania dowiedziono, iż proponowane w rozprawie rozwiązanie cechuje się podobnymi metrykami co algorytm neuronowy trenowany od wag losowych. Jednak jego trening, nawet z wykorzystaniem znacznie mniej wydajnych maszyn obliczeniowych, jest wydatnie krótszy. Drugi z eksperymentów zakładał porównanie algorytmów KrakN z aktualnym stanem wiedzy w dziedzinie wykrywania uszkodzeń powierzchni betonowych z użyciem algorytmów SI. Do eksperymentu wykorzystano sieci CrackNet i DeepCrack. Cechują się one obecnie najwyższą dokładnością spośród algorytmów, których modele zostały upublicznione. W wyniku drugiego eksperymentu porównawczego dowiedziono, iż oba z rozwiązań, do których się porównywano, obarczone są znacznie większym spadkiem metryk przy zastosowaniu na obrazach zawierających uszkodzenia o rozwartości poniżej 0,2 mm. Udowodniona została tym samym przydatność proponowanego rozwiązania.

W ostatniej części rozprawy postawiono sobie za cel rozbudowę proponowanego rozwiązania o system umożliwiający automatyczny pomiar parametrów uszkodzenia na podstawie danych zawartych na obrazie, ale wzbogaconych o informacje z systemu umożliwiającego pomiar długości celowej aparatu. System ten miał jednocześnie spełniać opisane wcześniej założenia dotyczące użytecznego narzędzia wspomagającego pracę inspektora mostowego. W szczególności miał być łatwy w użyciu, tani w budowie i eksploatacji oraz miał zapewniać skuteczność wykrywania uszkodzeń adekwatną do zastosowania w przeglądach podstawowym i rozszerzonym. W toku pracy zaprojektowano, a następnie utworzono i oprogramowano peryferyjny dodatek do smartfona pozwalający na precyzyjny pomiar długości celowej. W następnym kroku opracowany został algorytm przeprowadzający w pierwszym kroku segmentację semantyczną obrazu zawierającego uszkodzenie z wykorzystaniem maszyn wektorów nośnych, a w drugim wybierający charakterystyczne elementy uszkodzenia, pozwalając na jego wektoryzację. W ten sposób osiągnięto automatyczny pomiar wybranych parametrów uszkodzenia. System ten następnie został poddany ocenie przydatności, przy założeniu wykorzystania go w drugim etapie pomiaru, już po wstępnej detekcji uszkodzenia przy pomocy algorytmu KrakN.

7.2. OCENA STOPNIA OSIĄGNIĘCIA ZAKŁADANEGO CELU PRACY

Podstawowym celem pracy było zaproponowanie nowoczesnego rozwiązania, które zwiększyłoby efektywność okresowych inspekcji obiektów mostowych, jednocześnie nie zmieniając w sposób znaczny metody prowadzenia inspekcji oraz jej kosztów. Cele te zostały zawarte w tezie głównej oraz w trzech tezach pomocniczych przedstawionych w rozdziale 1. rozprawy.

W toku pracy udowodnione zostały tezy pomocnicze rozprawy. W rozdziale 4. dowiedziono pierwszej z nich. Algorytm trenowany przy pomocy techniki *transfer learning*, którego bazą była sieć neuronowa pierwotnie wykorzystywana do rozpoznawania obrazów ze zbioru ImageNet był w stanie nauczyć się rozpoznawać obiekty, na których nie był trenowany. Zaznaczyć należy jednocześnie, iż pierwotna baza danych nie zawiera obrazów uszkodzenia powierzchni betonu. Był to zbiór ImageNet powszechnie wykorzystywany w treningu i ocenie dokładności ogólnych algorytmów wizji komputerowej.

W rozdziale 5. udowodniona została druga teza pomocnicza. Dzięki zastosowaniu różnych zbiorów danych do ewaluacji utworzonego algorytmu potwierdzono zdolność algorytmu do generalizowania zdobytej wiedzy. Poprzez porównanie z istniejącymi rozwiązaniami, wykazano większą zdolność do generalizowania wiedzy w porównaniu do algorytmów trenowanych od wag losowych na docelowym zbiorze danych. Za sprawą wykorzystania metody *transfer learning*, z obrazu wydobywane są cechy, które mogą zostać pominięte, jeśli zbiór treningowy jest hermetyczny.

Rozdział 6. dowodzi ostatniej tezy pomocniczej pracy. Połączenie automatyzacji pomiaru długości celowej wraz z zastosowaniem wizji komputerowej opartej na metodach płytkiego uczenia maszynowego pozwoliło na pozyskanie cech fizycznych uszkodzenia po jego segmentacji na obrazie. Zaprojektowany na potrzeby rozprawy dodatek peryferyjny do smartfona zapewnia dostatecznie wysoką dokładność dla potrzeb przeglądów podstawowych i rozszerzonych.

Potwierdzone w rozprawie trzy tezy pomocnicze są podstawą do dowiedzenia tezy głównej. Wykazano, iż zastosowanie technik transfer learningu pozwala na zwiększenie wszechstronności i ogólnodostępności końcowego algorytmu przez jego większą zdolność do generalizowania wiedzy oraz zmniejszenie zapotrzebowania na moc obliczeniową wymaganą do jego treningu. Stosowanie wizji komputerowej w połączeniu z algorytmami uczenia maszynowego pozwala na automatyzację części czynności wykonywanych przez inspektora w trakcie inspekcji. Tym samym, wdrożenie proponowanych w rozprawie rozwiązań pozwoliłoby na zwiększenie efektywności pracy inspektora mostowego. **Potwierdza to jednocześnie tezę główną rozprawy**.

Na podstawie powyższego, zakładany cel pracy określony tezą główną i tezami pomocniczymi, można ocenić jako w pełni osiągnięty.

7.3. KIERUNEK DALSZYCH PRAC

Zaprezentowane w rozprawie rozwiązanie jest jedynie wstępem do rozwoju, a w perspektywie do całkowitej modernizacji metod pracy inspektorów mostowych i stanowić może podstawę dalszych prac badawczych.

W rozprawie wskazany został potencjał związany z metodami wizji komputerowej oraz uczenia maszynowego w automatyzacji detekcji i pomiarze uszkodzeń obiektów inżynierskich. Jest to

jednak potencjał, który także w prezentowanej rozprawie pozostaje nie w pełni wykorzystany. Temat ten jest bowiem zbyt obszerny na pojedyncze opracowanie, czego dowodem jest dynamika rozwoju dziedziny. Zarysowane zostały jednak kierunki, w których podążać musi rozwój tak, aby w największym stopniu udało się wykorzystać jej możliwości.

Pierwszym z nich jest dalsze ograniczanie zapotrzebowania na moc obliczeniową treningu algorytmów neuronowych. Kroki w tym kierunku zostały już poczynione, co zaowocowało rozwojem szeregu metod, mających na celu optymalizację sieci konwolucyjnych [241,242]. Dzięki technikom ograniczającym zapotrzebowanie na moc obliczeniową sieci neuronowych, w niedługim czasie może stać się możliwe wykorzystywanie ich w czasie rzeczywistym z użyciem procesorów telefonów komórkowych o niskiej wydajności. W odniesieniu do prezentowanych w rozprawie rozwiązań, pozwoliłoby to na całkowite pominięcie wykorzystania stacjonarnych maszyn obliczeniowych i zarządzanie uszkodzeniami na przykład z poziomu urządzenia mobilnego.

Kolejnym jest rozbudowa proponowanego rozwiązania tak, aby przy jego pomocy możliwy był pomiar rozwartości zlokalizowanej rysy. W sekcji 6.1. przedstawiającej założenia zasygnalizowana została jedna z metod, którą wykorzystać można do monitorowania zmian rozwartości rysy przy cyklicznych pomiarach. Nie pozwala ona jednak na dokładny pomiar fizycznych parametrów uszkodzenia. Aby pomiar taki był możliwy, konieczne jest podniesienie rozdzielczości pobranego obrazu w rejonie uszkodzenia. Oczywistą metodą jest zastosowanie aparatu o lepszych parametrach, lecz uszkodzenia o rozwartości poniżej 1 mm nadal mogą nie być dostatecznie dobrze reprezentowane pod kątem gęstości pikseli na obrazie. W celu rozwiązania tego problemu, przed segmentacją semantyczną obrazu, można poddać go wstępnej obróbce z wykorzystaniem modelu głębokiego uczenia maszynowego pozwalającego na sztuczne podniesienie rozdzielczości obrazu. Modele takie, nazywane modelami super-rozdzielczości, wykorzystywane są już obecnie np. w celu podniesienia jakości obrazów z kamer przemysłowych lub cyfrowej rekonstrukcji starych filmów. Być może jednak, po treningu na odpowiednim zbiorze danych, będą w stanie podnieść rozdzielczość obrazu rysy do poziomu umożliwiającego jej dokładny pomiar.

Ostatnim z istotnych, możliwych kierunków dalszego rozwoju prezentowanej pracy jest wykorzystanie potencjału leżącego w powiązaniu obrazu pozyskiwanego przy pomocy smartfonu z fizycznymi wymiarami obiektów. W prezentowanej rozprawie przedstawiono i rozwinięto jedynie jeden z możliwych sposobów pomiaru cech fizycznych obiektu na obrazie. Wartym jednak rozważenia jest wykorzystanie technik umożliwiających obrazowanie trójwymiarowe. Na przykład z wykorzystaniem zewnętrznego sensora lub smartfonów z wieloobiektywowymi aparatami pozwalającymi na stereoskopię. Dzięki zastosowaniu tych technik możliwy jest wielopunktowy pomiar odległości i uzyskanie chmury punktów. Na jej podstawie byłaby możliwa identyfikacja ilościowa uszkodzeń. Na przykład pomiar objętości ubytków betonu. Wymagałoby to jednak zastosowania architektur sieci neuronowych wykorzystujących trójwymiarową operację konwolucji.

7.4. WNIOSKI KOŃCOWE

Na inspektorach mostowych ciąży odpowiedzialność za stan techniczny kontrolowanych przez nich obiektów, a tym samym za bezpieczeństwo ich użytkowników. Są oni także jedynym łącznikiem między rzeczywistą konstrukcją i jednostką zarządzającą. Niedopatrzenia mogące wynikać z warunków ich pracy przenoszą się na całą infrastrukturę drogową, której kluczowym elementem są obiekty mostowe. Obecnie wyniki pracy inspektora mostowego zależą niemal wyłącznie od jego wyszkolenia i doświadczenia zawodowego. Inspektor nie posiada bowiem nowoczesnych narzędzi, które wydatnie mogłyby efektywnie podnieść skuteczność i komfort jego pracy.

Tym samym, jest to sytuacja trudna do zaakceptowania z punktu widzenia inspektorów i jednostki zarządzającej obiektem mostowym. Może także budzić obawy u świadomego użytkownika infrastruktury. Celowym jest zatem opracowanie i wdrożenie użytecznych narzędzi, które usprawniłyby proces przeprowadzania inspekcji – co było głównym celem niniejszej rozprawy.

Narzędzia i metody opracowane w ramach rozprawy, oprócz spełnienia zakładanych dla nich wymagań użytecznego narzędzia inżynierskiego, udowadniają także iż modernizacja wykorzystywanych w trakcie przeglądu narzędzi jest możliwa. Dzięki zastosowaniu autorskich rozwiązań, koszty takiej rozbudowy można znacznie ograniczyć, pozwalając tym samym na ich implementację w jak najszerszej skali. Może to mieć z kolei pozytywny wpływ na procesy decyzyjne, prowadzące ostatecznie do ich wdrożenia.

Opracowanie i wstępna implementacja proponowanych w rozprawie rozwiązań otwiera także drogę do podjęcia badań nad rozwojem podobnych systemów, przewidzianych do zastosowania nie tylko w procesie wykrywania uszkodzeń infrastruktury publicznej. Systemy te, odpowiednio rozbudowane mogą posłużyć także usprawnieniu procesu zarządzania innymi typami infrastruktury budowlanej, ograniczając koszty jej utrzymania przy jednoczesnym podniesieniu jej niezawodności i bezpieczeństwa.

PODZIĘKOWANIA

Pragnę podziękować wszystkim tym, bez wsparcia i pomocy których praca ta nigdy nie mogłaby powstać.

Dziękuję:

Panu Prof. inż. Markowi Salamakowi, promotorowi tej rozprawy, za umożliwienie mi podjęcia pracy naukowej i opiekę merytoryczną przez cały okres trwania moich studiów.

Panom Prof. dr hab. Jarosławowi Miszczakowi oraz dr Mateuszowi Ostaszewskiemu z Instytutu Informatyki Teoretycznej i Stosowanej Polskiej Akademii Nauk, za udzielone mi wsparcie merytoryczne.

Moim współpracownikom i kolegom z Katedry Mechaniki i Mostów Politechniki Śląskiej, za inspirujące dyskusje i wspólne seminaria.

Kończąc, pragnę złożyć także specjalne podziękowania mojemu ojcu Ireneuszowi i bratu Michałowi za wspieranie mnie przez całą moją drogę naukową, oraz Marcie, której cierpliwość wobec mnie zasługuje na osobny rozdział niniejszej rozprawy.

Mateusz Żarski

BIBLIOGRAFIA

[1] D. Rumelhart, H. Geoffrey, R. Williams, Neurocomputing: Foundations of Research, MIT Press, Cambridge, 1988. http://dl.acm.org/citation.cfm?%0Aid=65669.104451.

[2] S.A. Kalogirou, Designing and Modeling Solar Energy Systems, in: Sol. Energy Eng., Elsevier, 2009: pp. 553–664. https://doi.org/10.1016/B978-0-12-374501-9.00011-X.

[3] S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, V. Papastathis, M.G. Strintzis, Knowledge-Assisted Semantic Video Object Detection, IEEE Trans. Circuits Syst. 15 (2005) 1210–1224.

[4] N. Becherer, J. Pecarina, S. Nykl, K. Hopkinson, Improving optimization of convolutional neural networks through parameter fine-tuning, Neural Comput. Appl. 31 (2019) 3469–3479. https://doi.org/10.1007/s00521-017-3285-0.

[5] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature. 521 (2015) 436–444. https://doi.org/10.1038/nature14539.

[6] H. Adrian, Podstawy Informatyki - Grafy, AGH. (n.d.). http://home.agh.edu.pl/~horzyk/lectures/pi/ahdydpiwykl9.html (accessed December 1, 2019).

[7] R. Szeliski, Computer Vision: Algorithms and Applications, 2010.

[8] E. Michalak, L. Janas, A. Kaszyński, Zasady stosowania skali ocen punktowych stanu technicznego i przydatności do użytkowania drogowych obiektów inżynierskich, 2nd ed., Generalna Dyrekcja Dróg Krajowych i Autostrad, Warszawa, 2018. https://www.gddkia.gov.pl/frontend/web/userfiles/articles/m/materialy-pomocnicze-do-pobrania_6608/Zasady_stosowania_skali_ocen_Czesc_I_Wydanie_2.pdf.

[9] Paperswithcode.com, Semantic Segmentation, (2020). https://paperswithcode.com/task/semantic-segmentation (accessed August 9, 2021).

[10] A.M. Hafiz, G.M. Bhat, A Survey on Instance Segmentation: State of the art, (2020). https://doi.org/10.1007/s13735-020-00195-x.

[11] A. Rosebrock, Convolutional Neural Networks, in: Deep Learn. Comput. Vis. with Python Start. Bundle, PyImageSearch, 2017: pp. 171–198. https://www.goodreads.com/book/show/37846923-deep-learning-for-computer-vision-with-python-starter-bundle.

[12] A. Karpathy, Transfer Learning, Stanford Univ. (n.d.). http://cs231n.github.io/transfer-learning/ (accessed December 1, 2019).

[13] E. Varga, Machine Learning, in: Pract. Data Sci. with Python 3, Apress, Berkeley, CA, 2019: pp. 255–316. https://doi.org/10.1007/978-1-4842-4859-1_7.

[14] D.H. Ballard, C.M. Brown, Computer Vision, Prentice Hall Professional Technical Reference, 1982. https://dl.acm.org/doi/10.5555/578131.

[15] Rozporządzenie Ministra Transportu i Gospodarki Morskiej z dnia 30 maja 2000 r. w sprawie warunków technicznych, jakim powinny odpowiadać drogowe obiekty inżynierskie i ich usytuowanie., Poland, 2000. http://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20000630735.

[16] A.G. Lichtenstein, The Silver Bridge Collapse Recounted, J. Perform. Constr. Facil. 7 (1993) 249–261. https://doi.org/10.1061/(ASCE)0887-3828(1993)7:4(249).

[17] enacting jurisdiction United States, Federal-Aid Highway Act of 1968, 1968. https://lawcat.berkeley.edu/record/80374.

[18] Ustawa z dnia 21 marca 1985 r. o drogach publicznych., Poland, 1985. http://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU19850140060.

[19] poz 310 i 311 Dz.U.Nr 54, Ustawa z dnia 2 grudnia 1960 r. o kolejach, 1960.

[20] poz 90 Dz.U.Nr 20, Ustawa z dnia 29 marca 1962 r. o drogach publicznych, 1962.

[21] L. Janas, A. Jarominiak, E. Michalak, Instrukcje przeprowadzania przeglądów drogowych obiektów inżynierskich, 3rd ed., Stowarzyszenie Inżynierów i Techników Komunikacji RP, Oddział w Rzeszowie, Warszawa, 2020. https://www.gddkia.gov.pl/frontend/web/userfiles/articles/m/materialy-pomocnicze-do-pobrania_6608/Tekst instrukcji.pdf.

[22] M.C. Scutaru, C.C. Comisu, G. Boaca, N. Taranu, Bridge Maintenance Strategies – A brief comparison among different countries around the world, IABMAS 2018, Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. (2018) 231–238.

[23] I. Sidiqque, Mahad bridge collapse: One year later, still no closure, Mumbai Mirror. (2017). https://mumbaimirror.indiatimes.com/mumbai/cover-story/mahad-bridge-collapse-one-year-later-still-no-closure/articleshow/59923265.cms (accessed June 27, 2020).

[24] Genova, il crollo del ponte Morandi, La Repub. (2018). https://www.repubblica.it/cronaca/2018/08/18/news/crollo_genova_trovati_i_corpi_di_tre_dei_dispersi-204360358/ (accessed December 1, 2019).

[25] Foot overbridge collapses near CST station in Mumbai, over 30 injured and 6 dead, Econ. Times. (2019). https://economictimes.indiatimes.com/news/politics-and-nation/foot-bridge-collapses-near-cst-station-in-mumbai-several-injured-reported/articleshow/68413146.cms.

[26] J.S. Kong, D.M. Frangopol, Life-Cycle Reliability-Based Maintenance Cost Optimization of Deteriorating Structures with Emphasis on Bridges, J. Struct. Eng. 129 (2003) 818–828. https://doi.org/10.1061/(ASCE)0733-9445(2003)129:6(818).

[27] J. Kane, A. Tomer, Shifting into an era of repair: US infrastructure spending trends, Brookings.Edu. (2019). https://www.brookings.edu/research/shifting-into-an-era-of-repair-us-infrastructure-spending-trends/ (accessed January 1, 2020).

[28] E. Wsół, GDDKiA: Niemal 1,4 mld na utrzymanie infrastruktury w 2015 roku, Rynek Infrastruktury. (2015). http://www.rynekinfrastruktury.pl/mobile/gddkia-niemal-14-mld-na-utrzymanie-infrastruktury-w-2015-roku-51224.html (accessed December 1, 2019).

[29] B.A. Graybeal, B.M. Phares, D.D. Rolander, M. Moore, G. Washer, Visual inspection of highway bridges, J. Nondestruct. Eval. 21 (2002) 67–83. https://doi.org/10.1023/A:1022508121821.

[30] B.M. Phares, B.A. Graybeal, D.D. Rolander, M.E. Moore, G.A. Washer, Reliability and accuracy of routine inspection of highway bridges, Transp. Res. Rec. (2001) 82–92. https://doi.org/10.3141/1749-13.

[31] Serwis GDDKiA - dane statystyczne, (2020). https://www.gddkia.gov.pl/pl/a/6610/dane-statystyczne (accessed June 27, 2020).

[32] Roads In Japan 2018, Road Bur. Minist. Land, Infrastructure, Transp. Tour. (2018). http://www.mlit.go.jp/road/road_e/index_e.html.

[33] ASCE analysis of U.S. Department of Transportation, Fed. Highw. Adm. (2018). https://www.fhwa.dot.gov/bridge/nbi/ascii2018.cfm (accessed January 1, 2020).

[34] Wyniki Kontroli NIK, (2020). https://www.nik.gov.pl/kontrole/wyniki-kontroli-nik/ (accessed June 27, 2020).

[35] Federal Highway Administration Salaries, (2018). https://www.federalpay.org/employees/federal-highway-administration/2012 (accessed June 27, 2020).

[36] Counterclockwise: plotting the average camera resolution through the years, (2017). https://www.gsmarena.com/counterclockwise_average_camera-news-27838.php (accessed June 28, 2020).

[37] G.M. Moore, Cramming more components onto integrated circuits, Electronics. 38 (1965) 114. https://newsroom.intel.com/wp-content/uploads/sites/11/2018/05/moores-law-electronics.pdf.

[38] Zenon Waszczyszyn - Homepage, (n.d.). https://www.l5.pk.edu.pl/~zenwa/index.html (accessed June 28, 2020).

[39] M. Słoński, A comparison of deep convolutional neural networksfor image-based detection of concrete surface cracks, Comput. Assist. Methods Eng. Sci. 26 (2019) 105–112. https://doi.org/10.24423/cames.267.

[40] Y. AbdelRazig, L.-M. Chang, M. Skibniewski, Automated Quality Assessment of Constructed Surfaces through Intelligent Image Processing, in: Proc. 16th IAARC/IFAC/IEEE Int. Symp. Autom. Robot. Constr., 2017: pp. 693–699. https://doi.org/10.22260/isarc1999/0107.

[41] S. Lee, L.M. Chang, M. Skibniewski, Automated recognition of surface defects using digital color image processing, Autom. Constr. 15 (2006) 540–549. https://doi.org/10.1016/j.autcon.2005.08.001.

[42] CS 450 Inspection of highway structures, 0 ed., The National Archives, London, 2020.

[43] Austroads, Guide to Bridge Technology Part 7: Maintenance and Management of Existing Bridges, 2nd ed., Audtroads, Sydney, 2018.

[44] FHWA, 23 CFR 350 Subpart C—National Bridge Inspection Standards, 2017.

[45] Polskie Koleje Państwowe, INSTRUKCJA utrzymania kolejowych obiektów inżynierskich na liniach kolejowych do prędkości 200/250 km/h - Id-16, PKP, Warszawa, Polska, 2014.

[46] L.J. Butler, M.Z.E.B. Elshafie, C.R. Middleton, Pervasive Fibre-optic sensor networks in bridges: A UK case study, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1738–1745.

[47] M. Domaneschi, C. Apostoliti, G.P. Cimellaro, B. Glisic, K. Kliewer, Monitoring footbridges using wireless mesh networks, in: 9th Int. Conf. Bridg. Maintenance, Saf. Manag., 2018: pp. 909–922.

[48] T. Mimasu, Y. Goi, C.W. Kim, System identification of a bridge by a sparse-like system matrix, in: 6th Int. Symp. Life-Cycle Civ. Eng., 2018: pp. 929–936.

[49] D. Roach, Use of comparative vacuum monitoring sensors for automated, wireless health monitoring of bridges and infrastructure, SAND2018-5926C. (2018) 2747–2751.

[50] S. Rau, G. Morgenthal, Assessment of MEMS-based sensors for inclination measurements, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1577–1584.

[51] Y. Umekawa, H. Suganuma, Bridge displacement monitoring using acceleration measurement for efficient bridge management, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1710–1717.

[52] Q. Feng, Y.B. Liang, T.H. Yi, Grouting compactness monitoring of concrete-filled steel tube arch bridge using electro-mechanical impedance technique, Smart Struct. Syst. (2018) 541–546.

[53] D.G. Lu, W.H. Zhang, Z. Zhao, Health monitoring data modeling and reliability prediction of an actual bridge based on ARMA model, in: IABMAS 2018, Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., 2018: pp. 2089–2096.

[54] W. Anigacz, D. Beben, J. Kwiatkowski, Testing of bridge structures using laser scanning method, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1548–1555.

[55] P.S. Halding, J.W. Schmidt, C.O. Christensen, DIC-monitoring of full-scale concrete bridge using high-resolution Wideangle lens camera, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1492–1499.

[56] P. Hradil, K. Koski, Combined structural and traffic monitoring of steel suspension bridge, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1597–1604.

[57] J. Kullaa, Obtaining full-field response for optimal sensor placement, Struct. Multidiscip. Optim. (2018) 937–944.

[58] P. Olaszek, D. Sala, M. Kokot, M. Piątek, Railway bridge monitoring system using inertial sensors, Sensors. (2018) 1522–1529.

[59] J. Wuorenjuuri, Guide for a successful monitoring project from finland, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1718–1724.

[60] J. Bień, T. Kamiński, M. Kużawa, Monitoring in management of roadway bridges, in: IABMAS 2018, 2018: pp. 1839–1844.

[61] J.S. Jensen, F.R. Gottfredsen, Creating the basis for implementing BIMS in existing infrastructure components, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 172–178.

[62] T. Ishikawa, Y. Kuramitsu, H. Furuta, K. Tsukada, Inspection of bingo bridge by using high-sensitivity magnetic nondestructive testing, in: IABMAS 2018, 2018: pp. 1333–1336.

[63] M. Maizuar, L. Zhang, S. Miramini, P. Mendis, Structural monitoring of Eltham rail trestle bridge using advanced nondestructive testing techniques, ACMSM25. (2018) 650–653.

[64] H. Martín-Sanz, V. Dertimanis, L.D. Avendaño-Valencia, E. Chatzi, E. Brühwiler, Monitoring of the chillon viaduct after strengthening with uhpfrc, Struct. Infrastruct. Eng. (2018) 968–975.

[65] M. Nasim, S. Setunge, H. Mohseni, S.W. Zhou, Computational investigation on the piers of a u-slab bridge under raising flood intensity, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 2020–2027.

[66] B. Panchireddi, U. Yadav, J. Ghosh, Damage accumulation in aging highway bridges considering multiple earthquake events, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1044–1051.

[67] A.I. Sarkis, A. Palermo, O. Kammouh, G.P. Cimellaro, Seismic resilience of road bridges: Lessons learned from the 14 November 2016 Kaikōura earthquake, in: Proc. 9th Int. Conf. Bridg. Maintenance, Saf. (IABMAS 2018), 2018: pp. 1988–1995.

[68] D.Y. Yang, D.M. Frangopol, Renewal-theory-based life-cycle risk assessment of bridge deck unseating under hurricanes, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1996–2003.

[69] M.A. Valenzuela, N. Valenzuela, P. Moraga, F. Pineda, M. Márquez, R. Romo, Management of risk disasters: Application in Aysen and Valparaiso, Chile, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 2599–2605.

[70] E. Yamaguchi, Y. Tanaka, H. Tsuji, Influence of collision on structural performance of steel girder, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1966–1971.

[71] A. González, D. Martínez, E.J.O. Brien, M. Casero, J.J. Moughty, J.R. Casas, M. Vagnoli, R. Remenyte-Prescott, J. Andrews, F. Huseynov, J. Brownjohn, TRUSS-ITN methods for detecting bridge damage from response to traffic, in: 2018 Civ. Eng. Res. Irel. Conf. (CERI 2018), 2018: pp. 1761–1768.

[72] M.S. Kang, H. Pak, J.W. Kang, S.H. Kee, B.J. Choi, Structural behavior of a steel-concrete composite beam under fire condition, in: IABMAS Int. Conf., 2018: pp. 2146–2153.

[73] S. Liu, H. Zhu, L. Yang, M. Habib, Estimating climate effects on bridge decks deterioration, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1515–1521.

[74] V. Aryai, M. Mahmoodian, N.V. Ferdowsi, F. Ariai, Structural reliability analysis of corroding steel bridges using random-field representation, in: 9th Int. Conf. Bridg. Maintenance, Saf. Manag. 2018, 2018: pp. 1770–1775.

[75] M. Baeßler, F. Hille, A study on diverse strategies for discriminating environmental from damage based variations in monitoring data, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1557–1564.

[76] A. Toktorbai Uulu, H. Katsuchi, H. Yamada, Evaluation of adhered sea-salt particle amount to bridge, anti-adhesion countermeasures, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1958–1965.

[77] W. Lin, N. Taniguchi, T. Yoda, S. Satake, A preventive strengthening strategy for aged steel columns, J. Constr. Steel Res. (2018) 2470–2476.

[78] H. Singh, Repair and retrofitting of bridges – present and future, in: Proc. 9th Int. Conf. Bridg. Maintenance, Saf. (IABMAS 2018), 2018: pp. 1209–1215.

[79] M. Alsharqawi, T. Zayed, S.A. Dabous, Optimizing rehabilitation strategies for bridge decks under performancebased contracting setting, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 2529–2536.

[80] H.D. Tran, S. Setunge, Y.C. Koay, H. Luczak, A simulated maintenance costing using a Markov deterioration model for bridge c+omponents, in: Proc. 9th Int. Conf. Bridg. Maintenance, Saf. (IABMAS 2018), 2018: pp. 187–193.

[81] C. Jager, T. Pape, P. Shaw, R. Heywood, The bridge assessment maze, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1169–1176.

[82] M. Frizzarin, L. Mancassola, P. Franchetti, Quantifying increases in maintenance costs of Prestressed reinforced concrete (PRC) bridges due to increasing fatigue from heavier traffic loads, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 2760–2766.

[83] Y.C. Sung, K.C. Chang, C.C. Chen, C.C. Hsu, K.W. Chou, H.H. Hung, A novel life-cycle based management system for disaster mitigation of bridges, in: Maintenance, Safety, Risk, Manag. Life-Cycle

Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 901–907.

[84] T.M. Pape, P. Shaw, C. Doherty, G. Danicic, A. Robertson, D. Wilson, Bridge network planning for heavy vehicles in Queensland, CRC Press. (2018) 1442–1449.

[85] Y. Fujino, Bridge maintenance, renovation and management - research and development of governmental program in Japan, Struct. Infrastruct. Eng. (2018) 2–14.

[86] P.S. McCarten, Bridge risk management: Credibility gaps, in: IABMAS 2018, Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., 2018: pp. 1177–1184.

[87] M. Salamak, BIM w cyklu życia mostów, 1st ed., Wydawnictwo Naukowe PWN, Warszawa, 2020.

[88] B. Mcguire, THESIS - USING BUILDING INFORMATION MODELING TO TRACK AND ASSESS THE STRUCTURAL CONDITION OF BRIDGES, Colorado State University, 2014.

[89] M.S. Khan, S. Ghosh, J. Ghosh, C. Caprani, Sensitivity analysis of value of information framework, ASCE-ASME J. Risk Uncertain. Eng. Syst. (2018) 2081–2088.

[90] A. Strauss, S. Fernandes, J.R. Casas, L. Mold, J.C. Matos, Quality specifications and performance indicators for road bridges in Europe, in: IABMAS Int. Conf., 2018: pp. 1822–1831.

[91] Design Manual for Roads and Bridges, Inspection of Highway Structures, Des. Manaul Road Bridg. 3 Section (2007).

[92] B. Mahut, R.J. Woodward, Comparison of bridge management practice in England and France, Proc. 5th Int. Conf. Bridg. Manag. (2005) 163–170.

[93] P. Haardt, R. Holst, The German approach to Bridge Management, Tenth Int. Conf. Bridg. Struct. Manag. E-C128 (2008) 1–12.

[94] T.P.D. Mirzaei Z., Adey B. T., Klatter L., The IABMAS Bridge Management Committee – Overview of existing bridge management systems, 2014.

[95] Y. Ponomarev, Y. Yenyutin, V. Fedoseyev, G. Brodski, E. Brodskaia, Specific features of standard inspection of bridge structures in Moscow, Russia, Bridg. Struct. 2 (2006) 35–43. https://doi.org/10.1080/15732480600730896.

[96] Electronic Code of Federal Regulations (e-CFR), (n.d.). https://www.law.cornell.edu/cfr/text/23/chapter-I (accessed August 20, 2020).

[97] R.W. Joey Hartmann, National Bridge Inspection Standards, FHWA. (1971). https://www.fhwa.dot.gov/highwayhistory/national_bridge_inspection_standards.cfm.

[98] NCHRP, NCHRP SYNTHESIS 375: Bridge Inspection Practices, 2007. https://doi.org/10.17226/14127.

[99] FHWA, FHWA: Bridge Preservation Guide, FHWA Publication, 2011. https://trid.trb.org/view.aspx?c=viewmarked.

[100] J. Bień, Modelowanie obiektów mostowych w procesie ich eksploatacji, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2002. https://dbc.wroc.pl/dlibra/publication/1023/edition/1162/content?&action=ChangeMetaLangAction&lang=pl.

[101] Pontis, AASHTOWare Bridge Management Software, (n.d.).

[102] P.D. Thompson, National-scale bridge element deterioration model for the usa, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 2373–2379.

[103] G.D. Rummey, L.B. Dowling, Towards a Uniform Bridge Management System for Australia and New Zealand, Austroads Bridg. Conf. (2004). https://trid.trb.org/view/768186.

[104] M.M. Melhem, C. Caprani, A. Ng, Bridge Management in Australia and New Zealand: Current Approaches and Future Needs, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., CRC Press, 2018: p. 7.

[105] Y. Jeong, W.S. Kim, I. Lee, J. Lee, Bridge inspection practices and bridge management programs in China, Japan, Korea, and U.S., J. Struct. Integr. Maint. 3 (2018) 126–135. https://doi.org/10.1080/24705314.2018.1461548.

[106] MLIT, Manual for Bridge Periodic Inspection., Tokyo: Ministry of Land, Infrastructure, Transportation, and Tourism., Japan, 2014.

[107] S. Masahiro, T. Takashi, BRIDGE INSPECTION STANDARDS IN JAPAN AND US, in: 29th US - Japan Bridg. Eng. Work., 2013. https://www.google.pl/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjEhbDY4KnrAhWKJaYKHRAYA9sQFjAAegQIBRAB&url=https%3A%2F%2Fwww.pwri.go.jp%2Feng%2Fujnr%2Ftc%2Fg%2Fpdf%2F29%2F29-3-1_Shirato.pdf&usg=AOvVaw3p9oIWmreIWMYlIsF1C_dA.

[108] J.A. Whim, J. Rautenbach, A South African perspective on bridge maintenance management, 23rd Annu. South. African Transp. Conf. SATC 2004 Get. Recognit. Importance Transp. (2004) 233–241.

[109] A.J. Nell, A. Newmark, P.A. Nordengen, A Bridge Management System for the Western Cape Provincial Government, South Africa: Implementation and Utalization, (2008) 1–14.

[110] E.J. Kruger, A.A. Nyokana, Bridge management implemented by the South African national roads agency, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 206–213.

[111] R.P. Dutra, Inspeções em pontes e viadutos de concreto armado e protendido - Procedimento, Brasil, 2004.

[112] Kancelaria Sejmu RP, Ustawa z dnia 7 lipca 1994 r. − Prawo budowlane, Warszawa, n.d.

[113] Kancelaria Sejmu RP, Obwieszczenie Marszałka Sejmu Rzeczypospolitej Polskiej z dnia 6 grudnia 2013 r. w sprawie ogłoszenia jednolitego tekstu ustawy o transporcie kolejowym, Polska, 2015.

[114] Kancelaria Sejmu RP, USTAWA z dnia 28 marca 2003 r. o transporcie kolejowym, Kancelaria Sejmu RP, Polska, 2003.

[115] J. Rymsza, J. Biliszczuk, M. Mistewicz, WR-M- 81, Ministerstwo Infrastruktury, Polska, 2021. https://www.gov.pl/web/infrastruktura/wr-m.

[116] ZDW Gdańsk, Zarządzenie Nr 55/2006 z dnia 31.07.2006 r. Zastępcy Dyrektora Zarządu Dróg Wojewódzkich w Gdańsku w sprawie przyjęcia “Instrukcji przeprowadzania przeglądów drogowych obiektów inżynierskich,” ZDW, Gdańsk, 2006.

[117] M. Bednarczyk, System Gospodarki Mostowej– XX-lecie funkcjonowania, Drogownictwo. 5 (2010) 155–161. https://docplayer.pl/39188608-System-gospodarki-mostowej-xx-lecie-funkcjonowania.html.

[118] J. Bień, MOSTY KOLEJOWE – USZKODZENIA, AWARIE, KATASTROFY, in: Awarie Bud., Szczecin-Międzyzdroje, 2009. http://www.awarie.zut.edu.pl/files/ab2009/referaty/00_referaty_problemowe/03_Bien_J_Mosty_kolejowe-uszkodzenia,_awarie,_katastrofy.pdf.

[119] J. Bień, SYSTEMOWE WSPOMAGANIE ZARZĄDZANIA MOSTAMI DROGOWYMI I KOLEJOWYMI, in: Zesz. Nauk. Politech. Rzesz., Rzeszów, 2012: pp. 49–68.

https://oficyna.prz.edu.pl/fcp/YGBUKOQtTKlQhbx08SlkAWxhQAykrCDILDWdbE1VHWGFBWxslAxt1FSVcVnRKCkI8VQIDJCUODBYAKg8TXBEbIRBSBw/18/public/zeszyty_naukowe/wbiis/2012/budownictwo-pw-nowe/bud-59-03-1-pw2.pdf.

[120] SPIN, Rubikon, (2003). http://spin.neostrada.pl/projects_scientific.html#Rubikon (accessed September 19, 2020).

[121] M. Rabah, A. Elhattab, A. Fayad, Automatic concrete cracks detection and mapping of terrestrial laser scan data, NRIAG J. Astron. Geophys. 2 (2013) 250–255. https://doi.org/10.1016/j.nrjag.2013.12.002.

[122] H.Q. Tran, J. Huh, C. Kang, K. Kwak, J. Ahn, Assessment of lateral thermal diffusion of impulse thermography method in measuring size of non-planar defects, in: IABMAS Int. Conf., 2018: pp. 2873–2880.

[123] S.H. Jeon, J.M. You, J.H. Ahn, K. Il Cho, Y.S. Jeong, Measurement of time-dependent corrosion of steel bridge from corrosion monitoring, in: IABMAS 2018, Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., 2018: pp. 642–649.

[124] A. Yabe, T. Iye, A. Miyamoto, Automatic data collection system for structural health monitoring, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., 2018: pp. 1635–1641.

[125] N. Jin, T.S. Paraskeva, E.G. Dimitrakopoulos, Estimation of bridge frequencies from a passing vehicle, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 2566–2572.

[126] R. Helmerich, L. Moldenhauer, G. Voigt, F. Adao, E. Köppe, Considerations for identification of moisture in building materials using bluetooth®, in: Mater. Today Proc. 5.13, 2018: pp. 2752–2759.

[127] H.L. Li, Y.L. Ding, H.W. Zhao, S.T. Hou, Acceleration response analysis of a long-span steel arch bridge subjected to high-speed trains integrating SHM data, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 1689–1695.

[128] S.T. Hou, G. Wu, H.L. Li, An integrated model-based bridge management system, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 198–204.

[129] M. Salamak, M. Januszka, Brim bridge inspections in the context of industry 4.0 trends, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 2260–2267.

[130] R. Sacks, A. Kedar, A. Borrmann, L. Ma, D. Singer, U. Kattel, SeeBridge Information Delivery Manual (IDM) for Next Generation Bridge Inspection, ISARC 2016 - 33rd Int. Symp. Autom. Robot. Constr. (2016). https://doi.org/10.22260/ISARC2016/0100.

[131] R. Sacks, A. Kedar, A. Borrmann, L. Ma, I. Brilakis, P. Hüthwohl, S. Daum, U. Kattel, R. Yosef, T. Liebich, B.E. Barutcu, S. Muhic, SeeBridge as next generation bridge inspection: Overview, Information Delivery Manual and Model View Definition, Autom. Constr. 90 (2018) 134–145. https://doi.org/10.1016/j.autcon.2018.02.033.

[132] N. Catbas, C.Z. Dong, O. Celik, T. Khuc, A vision for vision-based technologies for bridge health monitoring, in: Proc. 9th Int. Conf. Bridg. Maintenance, Saf. (IABMAS 2018), 2018: pp. 54–62.

[133] H. Kim, E. Ahn, S. Cho, M. Shin, S.H. Sim, Comparative analysis of image binarization methods for crack identification in concrete structures, Cem. Concr. Res. 99 (2017) 53–61. https://doi.org/10.1016/j.cemconres.2017.04.018.

[134] R.S. Adhikari, O. Moselhi, A. Bagchi, Image-based retrieval of concrete crack properties for bridge inspection, Autom. Constr. 39 (2014) 180–194. https://doi.org/10.1016/j.autcon.2013.06.011.

[135] R.S. Adhikari, O. Moselhi, A. Bagchi, Image based retrieval of concrete crack properties, Gerontechnology. 11 (2012). https://doi.org/10.4017/gt.2012.11.02.458.00.

[136] G. Li, S. He, Y. Ju, K. Du, Long-distance precision inspection method for bridge cracks with image processing, Autom. Constr. 41 (2014) 83–95. https://doi.org/10.1016/j.autcon.2013.10.021.

[137] C. Koch, S.G. Paal, A. Rashidi, Z. Zhu, M. König, I. Brilakis, Achievements and Challenges in Machine Vision-Based Inspection of Large Concrete Structures, Adv. Struct. Eng. 17 (2014) 303–318. https://doi.org/10.1260/1369-4332.17.3.303.

[138] W. Wang, A. Zhang, K.C.P. Wang, A.F. Braham, S. Qiu, Pavement Crack Width Measurement Based on Laplace's Equation for Continuity and Unambiguity, Comput. Civ. Infrastruct. Eng. 33 (2018) 110–123. https://doi.org/10.1111/mice.12319.

[139] Y.-S. Yang, C. Wu, T.T.C. Hsu, H.-C. Yang, H.-J. Lu, C.-C. Chang, Image analysis method for crack distribution and width estimation for reinforced concrete structures, Autom. Constr. 91 (2018) 120–132. https://doi.org/10.1016/j.autcon.2018.03.012.

[140] F. Daize, E.A. Micu, E.J. O'Brien, A. Malekjafarian, Using images to estimate traffic loading on long-span bridges, in: 9th Int. Conf. Bridg. Maintenance, Saf. Manag., 2018: pp. 1411–1416.

[141] N. Hallermann, J. Taraben, G. Morgenthal, Bim related workflow for an image-based deformation monitoring of bridges, in: Proc. 9th Int. Conf. Bridg. Maintenance, Saf. (IABMAS 2018), 2018: pp. 157–164.

[142] D. Kaleta, D. Macheta, E. Reizer, M. Rajchel, Możliwości stosowania dronów do inspekcji mostów, Arch. Inst. Inżynierii Lądowej. (2017) 141–149. https://doi.org/10.21008/j.1897-4007.2017.24.10.

[143] M.A. Valenzuela, N. Valenzuela, A. Peña-Fritz, D. Torres, M. Márquez, UAV: First Chilean proposal of use on road bridge inspections, in: IABMAS 2018, Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., 2018: pp. 2350–2357.

[144] Y.-S. Yang, C.-M. Yang, C.-W. Huang, Thin crack observation in a reinforced concrete bridge pier test using image processing and analysis, Adv. Eng. Softw. 83 (2015) 99–108. https://doi.org/10.1016/j.advengsoft.2015.02.005.

[145] J. Bennetts, G. Webb, S. Denton, P.J. Vardanega, N. Loudon, Quantifying uncertainty in visual inspection data, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., 2018: pp. 2252–2259.

[146] S. Xu, D. Wang, R. Ma, A. Chen, H. Tian, Vortex-induced vibration prediction of bridges based on data fusion theory, in: IABMAS 2018, Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., 2018: pp. 2849–2856.

[147] S.A. Faroz, S. Ghosh, Bayesian integration of NDT with corrosion model for service-life predictions, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg. - Proc. 9th Int. Conf. Bridg. Maintenance, Saf. Manag. IABMAS 2018, 2018: pp. 2786–2792.

[148] S. Priyanka, Application of emerging artficial intelligence methods in structural engineering-a review, Www.Irjet.Net. 5 (2018). https://www.irjet.net/archives/V5/i11/IRJET-V5I11164.pdf.

[149] H. Salehi, R. Burgueño, Emerging artificial intelligence methods in structural engineering, Eng. Struct. 171 (2018) 170–189. https://doi.org/10.1016/j.engstruct.2018.05.084.

[150] A. Patil, L. Patted, M. Tenagi, V. Jahagirdar, M. Patil, R. Gautam, Artificial Intelligence as a Tool in Civil Engineering-A Review, IOSR J. Comput. Eng. . 36–39 (2017) 2278–8727. www.iosrjournals.org.

[151] S. Dorafshan, H. Azari, Deep learning models for bridge deck evaluation using impact echo, Constr. Build. Mater. 263 (2020) 120109. https://doi.org/10.1016/j.conbuildmat.2020.120109.

[152] S. Goldstein, D. Princiotta, J. Naglieri, Handbook of Intelligence, Springer London, London, 2015.

[153] L. GOTTFREDSON, Mainstream Science on Intelligence: An Editorial With 52 Signatories, History, and Bibliography, Wall Str. Journal, Mainstream Sci. Intell. (1994) 13–23. https://doi.org/10.1016/s0160-2896(97)90011-8.

[154] R. Feuerstein, The Dynamic Assessment of Cognitive Modifiability the Learning Propensity Assessment Device: Theory Instruments and Techniques, Jerusalem, 2002.

[155] U. Neisser, G. Boodoo, Et.al, Intelligence: Knowns and Unknowns, Am. Psychol. 51 (1996) 77–101.

[156] S. Legg, M. Hutter, A Collection of Definitions of Intelligence, in: Proc. 2007 Conf. Adv. Artif. Gen. Intell., 2007: pp. 17–24. https://doi.org/10.5555/1565455.1565458.

[157] K. Tirri, P. Nokelainen, eds., Measuring Multiple Intelligences and Moral Sensitivities in Education, SensePublishers, Rotterdam, 2011. https://doi.org/10.1007/978-94-6091-758-5.

[158] G. Dumitriu, The Factors of Intelligence Development and Individual Performance, BRAIN. Broad Res. Artif. Intell. Neurosci. 1 (2010) 102–106. https://www.edusoft.ro/brain/index.php/brain/article/view/46/143.

[159] J.R. Searle, Minds and brains without programs, Mindwaves. 3 (1980) 1–19.

[160] S. Legg, M. Hutter, Universal Intelligence: A Definition of Machine Intelligence, Minds Mach. 17 (2007) 391–444. https://doi.org/10.1007/s11023-007-9079-x.

[161] G.J. Kuperman, R.M. Reichley, T.C. Bailey, Using Commercial Knowledge Bases for Clinical Decision Support: Opportunities, Hurdles, and Recommendations, J. Am. Med. Informatics Assoc. 13 (2006) 369–371. https://doi.org/10.1197/jamia.M2055.

[162] M.A. Peot, D.E. Smith, Conditional nonlinear planning, (1992) 189–197. https://doi.org/10.1016/b978-0-08-049944-4.50027-6.

[163] Y. Goldberg, A Primer on Neural Network Models for Natural Language Processing, J. Artif. Intell. Res. 57 (2016) 345–420. https://doi.org/10.1613/jair.4992.

[164] M. Ford, G. Colvin, Will robots create more jobs than they destroy?, Guard. (2015). https://www.theguardian.com/technology/2015/sep/06/will-robots-create-destroy-jobs (accessed August 27, 2020).

[165] P. McCorduck, Machines Who Think, 2nd ed., Routledge, 2004.

[166] P. Grimal, Słownik mitologii greckiej i rzymskiej, 3rd ed., Zakład Narodowy im. Ossolińskich, 1997.

[167] G. Wood, Living Dolls: A Magical History Of The Quest For Mechanical Life, Guard. (2002). https://www.theguardian.com/books/2002/feb/16/extract.gabywood (accessed August 27, 2020).

[168] J.Y. Lettvin, H.R. Maturana, W.S. Mcculloch, W.H. Pitts, What the Frog's Eye Tells the Frog's Brain. Proceedings of the, Proc. IRE. 47 (1959) 1940–1959. https://doi.org/10.1109/JRPROC.1959.287207.

[169] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133. https://doi.org/10.1007/BF02478259.

[170] F. Rosenblatt, The Perceptron - A Perceiving and Recognizing Automaton, Rep. 85, Cornell Aeronaut. Lab. (1957) 460–1. https://doi.org/85-460-1.

[171] “The Thinking Machine” (1961) - MIT Centennial Film, Techtv.Mit.Edu. (2018). https://techtv.mit.edu/videos/10268-the-thinking-machine-1961---mit-centennial-film (accessed December 1, 2019).

[172] McCarthy, Minsky, Rochester, Shannon, A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE, (1955) 1–13. http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf.

[173] U.W. Eisenecker, Ai: The tumultuous history of the search for artificial intelligence, 1995. https://doi.org/10.3233/AIC-1995-8108.

[174] M.L. Minsky, S. Papert, Perceptrons : an introduction to computational geometry, MIT Press, Cambridge, 1988.

[175] C. Leondes, Expert Systems, MIT Press, 2001.

[176] H. Moravec, Mind children : the future of robot and human intelligence, Cambridge, 1988.

[177] H. Moravec, H.P. Newquist, S. Pub-, Book Reviews The Great 1980s AI Bubble : A Review of The Brain Makers, 15 (1994) 86–87.

[178] Y. LeCun, L. Bottou, O. Genevieve, K.-R. Muller, Efficient BackProp, Neural Networks: Tricks of the Trade. (1998). https://doi.org/10.5555/645754.668382.

[179] B. Goertzel, Human-level artificial general intelligence and the possibility of a technological singularity, Artif. Intell. 171 (2007) 1161–1173. https://doi.org/10.1016/j.artint.2007.10.011.

[180] B. Stoel, Use of artificial intelligence in imaging in rheumatology – current status and future perspectives, RMD Open. 6 (2020) e001063. https://doi.org/10.1136/rmdopen-2019-001063.

[181] S. Basodi, C. Ji, H. Zhang, Y. Pan, Gradient amplification: An efficient way to train deep neural networks, Big Data Min. Anal. 3 (2020) 196–207. https://doi.org/10.26599/BDMA.2020.9020004.

[182] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, V. Zocca, Python Deep Learning: Exploring Deep Learning Techniques and Neural Network Architectures with PyTorch, Keras and TensorFlow, 2018.

[183] S. Dorafshan, R.J. Thomas, M. Maguire, SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks, Data Br. 21 (2018) 1664–1668. https://doi.org/10.1016/j.dib.2018.11.015.

[184] ImageNet, Large Scale Visual Recognition Challenge 2012, ImageNet. (2012). http://www.image-net.org/challenges/LSVRC/2012/results.html (accessed August 29, 2020).

[185] F.-F. Li, J. Johnson, S. Yeung, CS231n: Convolutional Neural Networks for Visual Recognition, Stanford Vis. Lab. (2018). http://cs231n.stanford.edu/syllabus.html (accessed December 1, 2019).

[186] A. Ng, Machine Learning, Coursea. (2017) 88, 132, 137 141.

[187] M. Nielsen, Chapter 2: How the backpropagation algorithm works, Neural Networks Deep Learn. (2017).

[188] M. Mazur, A Step by Step Backpropagation Example, MattMazur.Com. (2015). https://mattmazur.com/2015/%0A03/17/a-step-by-step-backpropagation-example/ (accessed August 21, 2020).

[189] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biol. Cybern. 36 (1980) 193–202.

https://doi.org/10.1007/BF00344251.

[190] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE. 86 (1998) 2278–2324. https://doi.org/10.1109/5.726791.

[191] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM. 60 (2012) 84–90. https://doi.org/10.1145/3065386.

[192] S. Bianco, R. Cadene, L. Celona, P. Napoletano, Benchmark analysis of representative deep neural network architectures, IEEE Access. 6 (2018) 64270–64277. https://doi.org/10.1109/ACCESS.2018.2877890.

[193] A. Rosebrock, Spotting Overfitting and Underfitting, in: Deep Learn. Comput. Vis. with Python Start. Bundle, 2017: pp. 251–262.

[194] A. Rosebrock, Weight initialization, in: Deep Learn. Comput. Vis. with Python Start. Bundle, 2018: pp. 165–170.

[195] CS231n Stanford, CS231n Convolutional Neural Networks for Visual Recognition: Learning, Stanford Univ. (2017). https://cs231n.github.io/neural-networks-3/#sgd (accessed August 29, 2020).

[196] P. Lu, S. Chen, Y. Zheng, Artificial Intelligence in Civil Engineering, Math. Probl. Eng. 2012 (2012) 1–22. https://doi.org/10.1155/2012/145974.

[197] M.. Rafiq, G. Bugmann, D.. Easterbrook, Neural network design for engineering applications, Comput. Struct. 79 (2001) 1541–1552. https://doi.org/10.1016/S0045-7949(01)00039-6.

[198] M. Ochmański, G. Modoni, J. Bzówka, Prediction of the diameter of jet grouting columns with artificial neural networks, Soils Found. 55 (2015) 425–436. https://doi.org/10.1016/j.sandf.2015.02.016.

[199] X. Wu, J. Ghaboussi, J.H. Garrett, Use of neural networks in detection of structural damage, Comput. Struct. 42 (1992) 649–659. https://doi.org/10.1016/0045-7949(92)90132-J.

[200] J.P. Amezquita-Sanchez, M. Valtierra-Rodriguez, M. Aldwaik, H. Adeli, Neurocomputing in Civil Infrastructure, Sci. Iran. 23 (2016) 2417–2428. https://doi.org/10.24200/sci.2016.2301.

[201] J. Bień, Uszkodzenia i diagnostyka obiektów mostowych, 1st ed., Wydawnictwa Komunikacji i Łączności, Wrocław, 2010. http://www.wydawnictwopw.pl/index.php?s=karta&id=2309.

[202] N. Watters, A. Tacchetti, T. Weber, R. Pascanu, P. Battaglia, D. Zoran, Visual Interaction Networks: Learning a Physics Simulator from Video, in: Conf. Neural Inf. Process. Syst., 2017. https://doi.org/10.1063/1.1709499.

[203] J. Navrátil, A. King, J. Rios, G. Kollias, R. Torrado, A. Codas, Accelerating Physics-Based Simulations Using End-to-End Neural Network Proxies: An Application in Oil Reservoir Modeling, Front. Big Data. 2 (2019). https://doi.org/10.3389/fdata.2019.00033.

[204] Y.-J. Cha, W. Choi, O. Büyüköztürk, Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks, Comput. Civ. Infrastruct. Eng. 32 (2017) 361–378. https://doi.org/10.1111/mice.12263.

[205] J.-H. Chen, M.-C. Su, R. Cao, S.-C. Hsu, J.-C. Lu, A self organizing map optimization based image recognition and processing model for bridge crack inspection, Autom. Constr. 73 (2017) 58–66. https://doi.org/10.1016/j.autcon.2016.08.033.

[206] Y. AbdelRazig, L.-M. Chang, M. Skibniewski, Automated Quality Assessment of Constructed Surfaces through Intelligent Image Processing, Proc. 16th IAARC/IFAC/IEEE Int. Symp. Autom. Robot. Constr. (2017). https://doi.org/10.22260/isarc1999/0107.

[207] G. Li, X. Zhao, K. Du, F. Ru, Y. Zhang, Recognition and evaluation of bridge cracks with modified active contour model and greedy search-based support vector machine, Autom. Constr. 78 (2017) 51–61. https://doi.org/10.1016/j.autcon.2017.01.019.

[208] A. Zhang, K.C.P. Wang, Y. Fei, Y. Liu, S. Tao, C. Chen, J.Q. Li, B. Li, Deep Learning-Based Fully Automated Pavement Crack Detection on 3D Asphalt Surfaces with an Improved CrackNet, J. Comput. Civ. Eng. 32 (2018) 1–14. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000775.

[209] Y. Liu, J. Yao, X. Lu, R. Xie, L. Li, DeepCrack: A deep hierarchical feature learning architecture for crack segmentation, Neurocomputing. 338 (2019) 139–153. https://doi.org/10.1016/j.neucom.2019.01.036.

[210] F.T. Ni, J. Zhang, Z.Q. Chen, Pixel-level crack delineation in images with convolutional feature fusion, Struct. Control Heal. Monit. 26 (2019) 1–18. https://doi.org/10.1002/stc.2286.

[211] J.C.P. Cheng, M. Wang, Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques, Autom. Constr. 95 (2018) 155–171. https://doi.org/10.1016/j.autcon.2018.08.006.

[212] C.V. Dung, L.D. Anh, Autonomous concrete crack detection using deep fully convolutional neural network, Autom. Constr. 99 (2019) 52–58. https://doi.org/10.1016/j.autcon.2018.11.028.

[213] X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, X. Yang, Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network, Comput. Civ. Infrastruct. Eng. 33 (2018) 1090–1109. https://doi.org/10.1111/mice.12412.

[214] A. Zhang, K.C.P. Wang, Y. Fei, Y. Liu, C. Chen, G. Yang, J.Q. Li, E. Yang, S. Qiu, Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces with a Recurrent Neural Network, Comput. Civ. Infrastruct. Eng. 34 (2019) 213–229. https://doi.org/10.1111/mice.12409.

[215] {KrakN}: open-source framework for crack detection with transfer learning, (2020). http://github.com/.

[216] M. Żarski, MatZar01/KrakN: KrakN v1.0, Zenodo. (2020). https://zenodo.org/record/3764697 (accessed September 29, 2020).

[217] M. Żarski, Transfer Learning for leveraging computer vision in infrastructure maintenance [extracted features], Zenodo. (2020). https://zenodo.org/record/3755452 (accessed September 29, 2020).

[218] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, (2015). http://arxiv.org/abs/1506.01497.

[219] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, (2015). http://arxiv.org/abs/1506.02640.

[220] Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, O. Büyüköztürk, Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types, Comput. Civ. Infrastruct. Eng. 33 (2018) 731–747. https://doi.org/10.1111/mice.12334.

[221] Y. Du, N. Pan, Z. Xu, F. Deng, Y. Shen, H. Kang, Pavement distress detection and classification based on YOLO network, Int. J. Pavement Eng. (2020) 1–14. https://doi.org/10.1080/10298436.2020.1714047.

[222] S. Yokoyama, T. Matsumoto, Development of an Automatic Detector of Cracks in Concrete Using Machine Learning, Procedia Eng. 171 (2017) 1250–1255. https://doi.org/10.1016/j.proeng.2017.01.418.

[223] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, Li Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conf. Comput. Vis. Pattern Recognit., IEEE, 2009: pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848.

[224] D.M.W. Powers, Ailab, Evaluation : From Precision , Recall and F-Factor to ROC , Informedness , Markedness & Correlation, J. Mach. Learn. Technol. 2 (2007) 37–63. http://www.bioinfo.in/contents.php?id=51.

[225] B. Wójcik, M. Żarski, ASESMENT OF STATE-OF-THE-ART METHODS FOR BRIDGE INSPECTION: CASE STUDY, Arch. Civ. Eng. (2020).

[226] X. Zhong, X. Peng, S. Yan, M. Shen, Y. Zhai, Assessment of the feasibility of detecting concrete cracks in images acquired by unmanned aerial vehicles, Autom. Constr. 89 (2018) 49–57. https://doi.org/10.1016/j.autcon.2018.01.005.

[227] J.C.P. Cheng, M. Wang, Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques, Autom. Constr. 95 (2018) 155–171. https://doi.org/10.1016/j.autcon.2018.08.006.

[228] B.M. Phares, G.A. Washer, D.D. Rolander, B.A. Graybeal, M. Moore, Routine Highway Bridge Inspection Condition Documentation Accuracy and Reliability, J. Bridg. Eng. 9 (2004) 403–413. https://doi.org/10.1061/(ASCE)1084-0702(2004)9:4(403).

[229] P. Hüthwohl, R. Lu, I. Brilakis, Multi-classifier for reinforced concrete bridge defects, Autom. Constr. 105 (2019) 102824. https://doi.org/10.1016/j.autcon.2019.04.019.

[230] Y. Liu, J. Yao, X. Lu, R. Xie, L. Li, DeepCrack: A deep hierarchical feature learning architecture for crack segmentation, Neurocomputing. 338 (2019) 139–153. https://doi.org/10.1016/j.neucom.2019.01.036.

[231] C.F. Ozgenel, Concrete crack images for classification, (2018). https://doi.org/10.17632/5y9wdsg2zt.2.

[232] T.G. Dietterich, Ensemble Methods in Machine Learning, Springer London, London, 2000. https://link.springer.com/chapter/10.1007/3-540-45014-9_1.

[233] J. Yang, W. Wang, G. Lin, Q. Li, Y. Sun, Y. Sun, Infrared Thermal Imaging-Based Crack Detection Using Deep Learning, IEEE Access. 7 (2019) 182060–182077. https://doi.org/10.1109/ACCESS.2019.2958264.

[234] ExifTool, Exif Tags, (n.d.). https://exiftool.org/TagNames/EXIF.html (accessed August 4, 2021).

[235] Intel, Technologia Intel RealSense, (2021). https://www.intel.pl/content/www/pl/pl/architecture-and-technology/realsense-overview.html (accessed July 23, 2021).

[236] Huawei, HUAWEI EnVizion, (2021). https://consumer.huawei.com/es/support/accessories/envizion-360-camera/ (accessed July 27, 2021).

[237] Nikon, LASER 50 Laser Rangefinder, (2021). https://www.nikonusa.com/en/nikon-products/product/rangefinders/laser-50.html (accessed July 27, 2021).

[238] B. Artin, Protothreads, (2020). https://www.arduino.cc/reference/en/libraries/protothreads/ (accessed August 4, 2021).

[239] W. Fulton, Calculate Distance or Size of an Object in a photo image - The Math, Scantips.Com. (2015). https://www.scantips.com/lights/subjectdistance.html#math (accessed August 9, 2021).

[240] J.A. Miszczak, Wykład 05 - Maszyny Wektorów Wspierających, Miszczak.Eu. (2021). https://miszczak.eu/puma/2021/ (accessed August 9, 2021).

[241] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, D. Doermann, Towards Optimal Structured CNN Pruning via Generative Adversarial Learning, (2019).

[242] M. Żarski, B. Wójcik, K. Książek, J.A. Miszczak, Finicky transfer learning—A method of pruning convolutional neural networks for cracks classification on edge devices, Comput. Civ. Infrastruct. Eng. 37 (2022) 500–515. https://doi.org/10.1111/mice.12755.

[243] N.H. Pham, H.M. La, Design and implementation of an autonomous robot for steel bridge inspection, 54th Annu. Allert. Conf. Commun. Control. Comput. Allert. 2016. (2017) 556–562. https://doi.org/10.1109/ALLERTON.2016.7852280.

[244] RESONON, Outdoor Field System Portable Hyperspectral Imaging System, (2020). https://resonon.com/hyperspectral-outdoor-field-system (accessed August 20, 2020).

[245] A. Rosebrock, Networks as Feature Extractors, in: Deep Learn. Comput. Vis. with Python Pract. Bundle, 2017: pp. 31–47.

[246] H. Shirahata, Development of phased array ultrasonic test system for detection of fatigue crack of rib-to-deck weld of orthotropic steel deck system, in: Maintenance, Safety, Risk, Manag. Life-Cycle Perform. Bridg., 2018: pp. 2857–2864.

[247] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: 2015 IEEE Conf. Comput. Vis. Pattern Recognit., IEEE, 2015: pp. 1–9. https://doi.org/10.1109/CVPR.2015.7298594.

[248] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-Decem (2016) 770–778. https://doi.org/10.1109/CVPR.2016.90.

[249] A. Rosebrock, Fine-tuning Networks, in: Deep Learn. Comput. Vis. with Python Pract. Bundle, PyImageSearch, 2017: pp. 59–73. https://books.google.pl/books?id=TFDpvQEACAAJ.

STRESZCZENIE

Przedmiotem niniejszej rozprawy doktorskiej jest budowa nowoczesnego, opartego o algorytmy sztucznej inteligencji rozwiązania, mogącego w wydatny sposób podnieść efektywność pracy inspektora mostowego. Duży nacisk w rozprawie został położony na zastosowanie technik ograniczających zapotrzebowanie algorytmów na moc obliczeniową oraz niejednorodne zbiory danych uczących. Rozprawa dotyczy istotnego zagadnienia, jakim jest wykrywanie uszkodzeń powierzchni betonowych w trakcie inspekcji obiektów mostowych. Skoncentrowano się na powszechnym, lecz trudnym do wskazania i identyfikacji uszkodzeniu, jakim jest rysa o rozwartości nieprzekraczającej 0,2 mm.

Praca zawiera obszerne studium literatury obejmujące metody oceny stanu technicznego obiektów mostowych stosowane aktualnie na świecie oraz w Polsce, a także najnowsze rozwiązania w monitorowaniu stanu technicznego mostów opisane w pracach badawczych. Osobne studium literatury przygotowane zostało dla metod wykorzystujących algorytmy sztucznej inteligencji w zagadnieniach inżynierii budowlanej. Na podstawie przeprowadzonej analizy piśmiennictwa wnioskowano o cechach użytecznego narzędzia inżynierskiego mogącego podnieść efektywność inspekcji obiektów mostowych.

W dalszej części rozprawy opisano sposób budowy i implementacji narzędzia będącego przedmiotem pracy. Opisane zostały wykorzystane algorytmy i sposób ich działania, a także kryteria ich wyboru. Zaprezentowano również utworzone narzędzie wspomagające budowę oraz podział zbioru danych na podzbiory do treningu algorytmu. Proponowane rozwiązanie zostało także zweryfikowane pod kątem dokładności predykcji na niewidzianych wcześniej zbiorach danych, a także w porównaniu do aktualnego stanu wiedzy w dziedzinie wykrywania uszkodzeń, reprezentowanego przez gotowe, wytrenowane algorytmy głębokiego uczenia maszynowego.

W kolejnej części pracy rozbudowano proponowane rozwiązanie o system umożliwiający automatyzację wektoryzacji uszkodzenia oraz pomiaru jego parametrów na podstawie danych zawartych na obrazie, wzbogaconych o wyznaczoną długość celowej aparatu. System ten składał się z zaprojektowanego, peryferyjnego dodatku do smartfona mierzącego długość celowej, komunikującego się ze smartfonem w trakcie pobierania obrazu oraz logiki pozwalającej na segmentację uszkodzenia. Został on poddany ocenie przydatności jako drugi krok proponowanego w rozprawie narzędzia inżynierskiego.

W wyniku przeprowadzonych eksperymentów potwierdzono tezę główną rozprawy. Dysertacja została zakończona wskazaniem potencjalnych kierunków dalszych prac badawczych oraz opisem wniosków dotyczących narzędzi i metod wykorzystywanych w przeglądach betonowych obiektów mostowych.

SUMMARY

The subject of this dissertation is the development of a modern solution, based on artificial intelligence algorithms, that can significantly improve the efficiency of bridge inspector's work. A strong emphasis is placed on the application of techniques for reducing the algorithms' demand for computational power and heterogeneous sets of learning data. The dissertation addresses the important issue of detecting concrete surface defects during bridge inspections. The focus is on a common but difficult to pinpoint and identify damage such as a crack with a width not exceeding 0.2 mm.

The paper contains an extensive literature study covering the methods of assessing the technical condition of bridges currently used in the world and in Poland, as well as the latest solutions in bridge technical condition monitoring described in research papers. A separate literature study was prepared for methods using artificial intelligence algorithms in structural engineering problems. Based on the literature analysis, the features of a useful engineering tool that can increase the efficiency of bridge inspections were concluded.

In the further part of the dissertation the method of development and implementation of the bridge inspector support tool was described. The algorithms used and their mode of operation are described, as well as the criteria for their selection. The created tool for supporting the construction and division of the data set into subsets for algorithms training is also presented. The proposed solution was also verified in terms of prediction accuracy on previously unseen datasets, as well as in comparison to the current state of the art methods in defect detection, represented by current, trained deep machine learning algorithms.

In the next part of the work, the proposed solution was extended to include a system capable of automating the damage vectorization and measurement of its parameters based on the data contained in the image, enriched by the determined camera to target distance. This system consisted of a designed peripheral smartphone add-on measuring the distance, communicating with the smartphone during image capturing, and logic allowing for defect segmentation. It was subjected to usability evaluation as the second step of the bridge inspector support tool proposed in the dissertation.

As a result of the experiments, the main thesis of the dissertation was confirmed. The dissertation was concluded by indicating potential directions for further research work and describing conclusions regarding tools and methods used in the inspection of concrete bridges.

ZAŁĄCZNIK Z KODAMI ŹRÓDŁOWYMI UTWORZONYCH NA POTRZEBY ROZPRAWY APLIKACJI

Tab. 17: Wykaz algorytmów

Kod załącznika	Nazwa algorytmu	Opis algorytmu
A	`Data_Augmentation`	Aplikacja `data_augmentation.py` wykorzystywana jest do wzbogacania podstawowego zbioru danych o zmodyfikowane punkty danych. Wykorzystuje operacje takie jak losowe przesunięcia, obroty, zbliżenia oraz odbicia lustrzane aby zwiększyć różnorodność zbioru wykorzystywanego do treningu algorytmu SI.
B	`Dataset_builder`	Program `dataset_builder.py` jest wykorzystywany do budowy zbioru punktów danych jako pojedynczych obrazów w formacie `.png`, pogrupowanych w folderach odpowiadających klasom. Posiada prosty, graficzny interfejs w celu ułatwienia jej obsługi.
C	`Extract_features`	Aplikacja ta ma na celu budowę zbioru zawierającego cechy obrazów ze zbioru treningowego i zapis ich do dwóch plików w formacie `.hdf5`, odpowiadających zbiorowi treningowemu i testowemu. Ekstrakcja cech przeprowadzana jest przy pomocy modelu VGG16.
D	`Train_model`	Skrypt `train_model.py` przeznaczony jest do treningu modelu regresji logistycznej z wykorzystaniem cech zbioru treningowego pozyskanych z wykorzystaniem aplikacji `extract_features.py`. Efektem działania skryptu jest wytrenowany model w formacie `.pickle`.
E	`Moving_window`	Aplikacja `moving_window.py` służy do przeprowadzania inferencji z modelem na obrazie wejściowym. Jej wynikiem jest obraz wyjściowy zawierający defekty oznaczone obwiednią prostokątną oraz drugi obraz zawierający maski lokalizacji odnalezionych uszkodzeń.
F	`Measuring_device_logic`	Program ten odpowiada za logikę sterującą dodatkiem do smartfona pozwalającym na pomiar długości celowej. Oprócz obsługi pomiaru odległości, pozwala także na sterowanie diodą laserową umożliwiającą precyzyjne określenie miejsca prowadzonego pomiaru.
G	`Get_px_size`	Aplikacja `get_px_size.java` pozwala na zapis informacji o długości celowej do danych exif pobieranego obrazu. Jest to aplikacja przygotowana z myślą o smartfonach z systemem Android. W pliku `.java` znajduje się główna logika operująca sposobem wyświetlania, pobierania oraz zapisywania obrazu wraz z danymi exif.
H	`Get_poly`	Program `get_poly.py` pozwala na ekstrakcję rysy z obrazu w postaci wektorowej, w przestrzeni o znanych wymiarach fizycznych. Program wykorzystuje algorytm maszyn wektorów nośnych do segmentacji uszkodzenia na podstawie kształtu jego binarnej maski. Następnie z użyciem algorytmu klasteryzacji dzieli uszkodzenie na odnogi, na których w ostatnim kroku wyszukiwane są punkty charakterystyczne przy pomocy regresji wielomianowej.

A DATA_AUGMENTATION

data_augmentation.py

```
import cv2
from keras.preprocessing.image import ImageDataGenerator
from imutils import paths
import numpy as np

# get random number
random = np.random.randint(0, 100)

# set desired number of images
num = 10
current = 0

# select data dir
DATA_DIR = './datapoints'
# load image paths
image_paths = list(paths.list_images(DATA_DIR))

# initialize data augmentor
aug = ImageDataGenerator(rotation_range=30*random,\
          width_shift_range=0.1*random,\
          height_shift_range=0.1*random, shear_range=0.2*random, \
          zoom_range=0.2*random, horizontal_flip=True, fill_mode="nearest")

# load and preprocess each datapoint
for image_path in image_paths:
  image = cv2.imread(image_path)

  # add 3rd dimension to image matrix
  image = np.expand_dims(image, axis=0)

  # process image
  image_generator = aug.flow(image, batch_size=1, \
          save_to_dir=DATA_DIR.replace('datapoints', 'datapoints_aug'))

  # run generator for each image and selected iteration number
  for image in image_generator:
    current += 1
    if current == num
      break

print("All done!")
```

B DATASET_BUILDER

dataset_builder.py

```
#!python3
try:
    import platform
    import cv2
    from imutils import paths
    import numpy as np
    import pygame
    import os
    import itertools
    import random
except ImportError as e:
    print(e)
    print("One or more dependencies missing!\nOpen README file to check
          required dependencies.")
    if platform.system() == 'Linux':
        print("\nYou can install all dependencies using 'sudo chmod +x
              ./install_dependencies.sh & ./install_"
              "dependencies.sh' command in KrakN directory.")
    else:
        print("\nYou can install all dependencies using
              install_dependencies.bat in KrakN directory")
    quit()

# def 'on_trackbar' handlers to manage window position:
def on_trackbar_horizontal(val):
    global pos_x
    pos_x = cv2.getTrackbarPos(TRACKBAR_HORIZONTAL, WINDOW_NAME)
    crop = image[pos_y:pos_y + screen_height, pos_x:pos_x + screen_width]
    cv2.imshow(WINDOW_NAME, crop)
    cv2.moveWindow(WINDOW_NAME, 0, 0)

def on_trackbar_vertical(val):
    global pos_y
    global crop
    pos_y = cv2.getTrackbarPos(TRACKBAR_VERTICAL, WINDOW_NAME)
    crop = image[pos_y:pos_y + screen_height, pos_x:pos_x + screen_width]
    cv2.imshow(WINDOW_NAME, crop)
    cv2.moveWindow(WINDOW_NAME, 0, 0)

# def on_click mouse callback to point defects polylines
def on_mouse(event, x, y, flags, param):
    global global_centers
    if event == cv2.EVENT_LBUTTONDOWN:
        point_list.append((x + pos_x, y + pos_y))
        if len(point_list) == 2:
            cv2.line(image, point_list[0], point_list[1], (0, 0, 255), 2)
            x_step = (point_list[1][0] - point_list[0][0]) // (FRAME_COUNT
                     + 1)
            y_step = (point_list[1][1] - point_list[0][1]) // (FRAME_COUNT
                     + 1)
            if len(global_centers) == 0:
                global_centers.append(point_list[0])
            for i in range(1, FRAME_COUNT + 1):
                cv2.circle(image, (point_list[0][0] + i * x_step,
                    point_list[0][1] + i * y_step), 3, (0, 0, 255), -1)
                global_centers.append((point_list[0][0] + i * x_step,
                    point_list[0][1] + i * y_step))
```

```
            global_centers.append(point_list[1])
            point_list.pop(0)
        cv2.circle(image, (x+pos_x, y+pos_y), 5, (0, 0, 255), -1)

# get screen resolution to fit image crop
pygame.init()
info = pygame.display.Info()
screen_width = info.current_w
screen_height = info.current_h - 200
pygame.quit()

# set global constraints
WINDOW_NAME = "Dataset Builder"
TRACKBAR_HORIZONTAL = "Navigate width"
TRACKBAR_VERTICAL = "Navigate height"
FRAME_COUNT = 3
# set zoom factor
while True:
      try:
            ZOOM_FACTOR = float(input("Set zoom factor: "))
            break
      except:
            print("Error, use '.' as decimal separator")
FRAME_SIZE = int(224 // ZOOM_FACTOR)
HALF_SIZE = int(FRAME_SIZE // 2)
os_separator = os.path.sep
DATAPOINTS_STRING = r'.{}datapoints{}'.format(os_separator, os_separator)
IMAGE_FILE_STRING = r'.{}database{}Images'.format(os_separator,
os_separator)
FILE_DONE_STRING = IMAGE_FILE_STRING.replace("Images", "DONE")
FILE_ITERATOR = 0
FRAME_ITERATOR = 0
KEYS_IDS = list(range(49, 58))
KEYS_NAMES = [str(num) for num in range(1, 10)]
# add colors
color_list = list(itertools.combinations_with_replacement([0, 155, 255],
                  3))
random.shuffle(color_list)

# check files integrity
if not os.path.exists(DATAPOINTS_STRING):
    os.mkdir(DATAPOINTS_STRING)
if not os.path.exists(IMAGE_FILE_STRING):
    print("No database found, add images to ./database/Images directory")
    os.makedirs(IMAGE_FILE_STRING)
    os.makedirs(FILE_DONE_STRING)
    quit()

# check output directory exist or make new Crops file
class_list = os.listdir(DATAPOINTS_STRING)
if len(class_list) == 0:
    os.mkdir(DATAPOINTS_STRING + 'Crops')
    print('NO CLASSES FOUND\n\nAdded Crops class')
    class_list = os.listdir(DATAPOINTS_STRING)
# check if classes extend 9 files
if len(class_list) > 9:
    print("Too many classes in datapoints directory\nMax number of classes
          in single run is 9")
    quit()

# leave only relevant KEYS and colors
```

```
KEYS_IDS = KEYS_IDS[0:len(class_list)]
KEYS_NAMES = KEYS_NAMES[0:len(class_list)]
color_list = color_list[0: len(class_list)]

# make file for images done:
if not os.path.exists(FILE_DONE_STRING):
    os.mkdir(FILE_DONE_STRING)

# get paths from image folder:
imagePaths = list(paths.list_images(IMAGE_FILE_STRING))

for path in imagePaths:
    # load image
    image = cv2.imread(path)
    output_image = image.copy()
    # add instructions to image
    cv2.putText(image, "ESC to exit", (0, 20), cv2.FONT_HERSHEY_PLAIN, 1.3,
                (200, 255, 0), 2)
    cv2.putText(image, "SPACE for next image", (0, 45),
                cv2.FONT_HERSHEY_PLAIN, 1.3, (200, 255, 0), 2)
    for i in range(0, len(class_list)):
        cv2.putText(image, "{} to save as {} class".format(KEYS_NAMES[i],
                    class_list[i]), (0, 70 + i * 25),
                    cv2.FONT_HERSHEY_PLAIN, 1.3, color_list[i], 2)

    # set initial image constraints and variables list
    h, w = image.shape[:2]
    x_max = w - screen_width - 1
    y_max = h - screen_height - 1
    pos_x = 0
    pos_y = 0
    point_list = []
    global_centers = []

    # add named window
    cv2.namedWindow(WINDOW_NAME)
    # add track bars
    cv2.createTrackbar(TRACKBAR_HORIZONTAL, WINDOW_NAME, pos_x, x_max,
                       on_trackbar_horizontal)
    cv2.createTrackbar(TRACKBAR_VERTICAL, WINDOW_NAME, pos_y, y_max,
                       on_trackbar_vertical)
    # add mouse callback
    cv2.setMouseCallback(WINDOW_NAME, on_mouse)

    while True:
        on_trackbar_horizontal(0)
        on_trackbar_vertical(0)
        key = cv2.waitKey(20)
        if key in KEYS_IDS:
            for point in global_centers:
                # get top left point of crop
                top_left_x = point[0] - HALF_SIZE
                top_left_y = point[1] - HALF_SIZE
                bottom_right_x = point[0] + HALF_SIZE
                bottom_right_y = point[1] + HALF_SIZE
                # check if top left is less than 0
                if top_left_x < 0:
                    top_left_x = 0
                    bottom_right_x = top_left_x + FRAME_SIZE
                if top_left_y < 0:
                    top_left_y = 0
```

```
                    bottom_right_y = top_left_y + FRAME_SIZE
                # check if bottom right is greater than image size
                if bottom_right_x >= w - 1:
                    top_left_x -= (bottom_right_x - w)
                    bottom_right_x = top_left_x + FRAME_SIZE
                if bottom_right_y >= h - 1:
                    top_left_y -= (bottom_right_y - h)
                    bottom_right_y = top_left_y + FRAME_SIZE

                # resize to 224x224 and save crop to file
                tile = output_image[top_left_y:bottom_right_y,
                                    top_left_x:bottom_right_x]
                tile = cv2.resize(tile, (224, 224),
                                  interpolation=cv2.INTER_CUBIC)

                for i in range(0, len(class_list)):
                    if key == KEYS_IDS[i]:
                        # check if file exists
                        output_path = DATAPOINTS_STRING + class_list[i] +
                                      os_separator + class_list[i] + "_" +\
                str(FRAME_ITERATOR) + "_s_{} ".format(ZOOM_FACTOR) + ".png"
                        while os.path.exists(output_path):
                            FRAME_ITERATOR += 1
                            output_path = DATAPOINTS_STRING + class_list[i]
                            + os_separator + class_list[i] + "_" + \
                str(FRAME_ITERATOR) + "_s_{}".format(ZOOM_FACTOR) + ".png"

                        # show crops on image
                        cv2.rectangle(image, (top_left_x, top_left_y),
                        (bottom_right_x, bottom_right_y), color_list[i], 2)
                        cv2.imwrite(output_path, tile)

            # empty global centers and point list
            global_centers = []
            point_list = []

        if key == 32:
            pos_x = 0
            pos_y = 0
            # reset track bars
            cv2.setTrackbarPos(TRACKBAR_HORIZONTAL, WINDOW_NAME, 0)
            cv2.setTrackbarPos(TRACKBAR_VERTICAL, WINDOW_NAME, 0)
            # destroy window to initialize new trakc bars values
            cv2.destroyAllWindows()
            # move current image to DONE folder
            os.rename(path, path.replace('Images', 'DONE'))
            break

        # exit
        if key == 27:
            quit()

print("All images done! Press ENTER to exit.")
```

C EXTRACT_FEATURES

extract_features.py

```
# extract class labels and encode them
labels = [p. split (os. path .sep )[ -2] for p in imagePaths ]
le = LabelEncoder ()
labels = le. fit_transform ( labels )

# load VGG16 network excluding final FC layers
print (" loading network ...")
model = VGG16 ( weights =" imagenet ", include_top = False )

# initialize dataset writer
dataset = HDF5DatasetWriter ((len( imagePaths ), 512 * 7 * 7) ,
                             outputPath , " features ",
                             bufferSize )
                             dataset . storeClassLabels (le.Classes_ )

batchImages = [ ]
for (j, imagePath ) in enumerate ( batchPaths ):
  # load and resize image
  image = load_img ( imagePath , target_size =(224 , 224) )
  image = img_to_array ( image )

  # preprocess image by expanding dimensions and subtracting mean RGB value
  image = np. expand_dims (image , axis =0)
  image = imagenet_utils . preprocess_input ( image )

  # add image to batch
  batchImages . append ( image )

  # pass images the network
  batchImages = np. vstack ( batchImages )
  features = model . predict ( batchImages , batch_size = batchSize )

  # reshape features
  features = features . reshape (( features . Shape [0] , 512 * 7 * 7))

  # add features and labels to dataset
  dataset .add ( features , batchLabels )
```

D TRAIN_MODEL

train_model.py

```
# open database
db = h5py . File ( databasePath , "r")

# set the training / testing split index
i = int(db[" labels "]. shape [0] * 0.75)

# train Logistic Regression classifiers
print (" Tuning hyperparameters ... ")
params = {"C": [0.1 , 1.0 , 10.0 , 100.0 , 1000.0 , 10000.0]}
model = GridSearchCV (
                      LogisticRegression ( max_iter =512),
                      params , cv =3, n_jobs = jobs )
model .fit(db[" features "][:i], db[" labels " ][:i])

print (" Best hyperparameters : {}". format ( model . best_params_ ))

# evaluate model
print (" Evaluating ...")
preds = model . predict (db[" features "][i:])
print ( classification_report (db[" labels "][i:],
preds , target_names =db[" label_names "]))

# save model to disk
print (" Saving model ...")
f = open ( modelPath , "wb")
f. write ( pickle . dumps ( model . best_estimator_ ))
```

E MOVING_WINDOW

moving_window.py

```
while True :
  # crop single tile from image
  subImage = imageCV [ yTop : yTop + windowSize , xTop : xTop +
                       windowSize ]
  # resize sub image according to database settings
  subImage = cv2.resize ( subImage , ( tileSize , tileSize ),
                          interpolation =cv2.INTER_CUBIC )

  # extract image features with VGG16
  img_data = image.img_to_array ( subImage )
  img_data = np.expand_dims ( img_data , axis =0)
  img_data = preprocess_input ( img_data )
  features = model.predict ( img_data )
  features = features.reshape(( features.shape [0] , 512 * 7 * 7))

  # get prediction
  predict = classifier.predict_proba ( features )
  predictionList = list ( predict )[0]

  # assess predictions and add them to output images
  maxPredIndex = np.argmax ( predictionList )
  maxPred = predictionList [ maxPredIndex ]

  if maxPred >= confidenceThreshold:
    cv2.rectangle ( maskList [ maxPredIndex ],(xTop , yTop ),
                    ( xTop + windowSize , yTop + windowSize ),
                    (255 , 255 , 255) , -1)

  # add bounding boxes to input image
  for i in range (0, len ( labels )):
    contours , _ = cv2 . findContours ( maskList [i]. astype (’uint8 ’),
                         cv2. RETR_TREE , cv2. CHAIN_APPROX_SIMPLE )
    outputImage = imageCV.copy()

    for contour in contours :
      x, y, w, h = cv2 . boundingRect ( contour )
      cv2.rectangle(outputImage , (x, y), (x + w, y + h), (0, 0, 255) , 2)
      # write output image
      cv2.imwrite(outputPathDir + ’/’ + labels [i] + " _out " + ". png ",
                  outputImage )
```

F MEASURING_DEVICE_LOGIC

Measuring_device_logic.ino

```
#include <NewPing.h>
#include "Button.h"

#define TRG_PIN 12          //pulse triggering pin
#define E_PIN 11            //echo receiving pin
#define MAX_DISTANCE 200    //max measuring distance
#define BTN_PIN 5           //pushbutton pin
#define LSR_PIN 2           //laser diode pin
#define BPW_PIN 3           //laser power pin

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
Button pushButton(PUSHBUTTON_PIN);
bool laserOn = false;
bool buttonPressed = false;
int counter = 0;
long prevTime;
long currTime;

void setup() {
  Serial.begin(115200);
  pushButton.init();
  pinMode(LASER_PIN, OUTPUT);
  pinMode(BUTTON_POWER_PIN, OUTPUT);
  digitalWrite(BUTTON_POWER_PIN, HIGH);
  prevTime = 0;
}

void loop() {
  currTime = millis() / 500;
  //Serial.println(pushButton.read());
  if (pushButton.read())
  {
    if (counter == 0)
    {
      if (!laserOn)
      {
        laserOn = true;
        digitalWrite(2, HIGH);
      }
      else
      {
        laserOn = false;
        digitalWrite(2, LOW);
      }
    }
    counter += 1;
  }
  if (counter != 0 && !pushButton.read())
  {
    counter = 0;
  }
  int message = Serial.read();
  if (message != -1 || message == -1)
  {
    if (currTime != prevTime)
    {
      prevTime = currTime;
```

```
      Serial.println(currTime);
      Serial.print("Ping: ");
      Serial.print(sonar.ping_cm());
      Serial.println("cm");
    }
  }
  Serial.println(laserOn);
}
```

G GET_PX_SIZE

Get_px_size.java

```
package com.example.mateusz.serialtest;

import android.app.PendingIntent;
import android.content.Intent;
import android.hardware.usb.UsbDevice;
import android.hardware.usb.UsbDeviceConnection;
import android.hardware.usb.UsbManager;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;

import java.util.concurrent.ThreadLocalRandom;

import me.aflak.arduino.Arduino;
import me.aflak.arduino.ArduinoListener;

public class MainActivity extends AppCompatActivity implements
ArduinoListener {

    public TextView distanceLabel;
    public Arduino Pololu;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON
);
        setContentView(R.layout.activity_main);
        distanceLabel = findViewById(R.id.DistLabel);
        double base = 2.70;
        String sensor_spec = "IMX318";
        double sens_h = 6.811;
        double sens_px = 5512;
        Pololu = new Arduino(this);
        Camera camera;
    }
    @Override
    protected void onStart() {
        super.onStart();
        Pololu.setArduinoListener(this);
        camera = Camera.open();
        Parameters params = camera.getParameters();
        double focal = Double.parseDouble(params.fetFocalLenght());
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Pololu.unsetArduinoListener();
        Pololu.close();
    }
    @Override
```

```
    public void onUsbPermissionDenied() {
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Pololu.reopen();
            }
        }, 3000);
    }
    @Override
    public void onArduinoMessage(byte[] bytes) {
        String str = new String(bytes);
        double len = Double.parseDouble(str) + base;
        double px_size = (len * sens_h) / (sens_px * focal)
        String result = String.format("%.2f", px_size);
        display(result);
        String caller = "call";
        Pololu.send(caller.getBytes());
    }

    public void display(final String message){
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                distanceLabel.setText(message+"\n");
            }
        });
    }
    @Override
    public void onArduinoAttached(UsbDevice device) {
        distanceLabel.setText("Device attached!");
        Pololu.open(device);
    }
    @Override
    public void onArduinoDetached() {
        distanceLabel.setText("Device detached");
    }
    @Override
    public void onArduinoOpened() {
        String str = "Distance measurement";
        Pololu.send(str.getBytes());
    }
}
```

H GET_POLY

Get_poly.py

```
import cv2
import numpy as np
from imutils import paths
from joblib import load
from imutils import grab_contours
from math import degrees, atan2

class Point:
    def __init__(self, x, y, order=0, linked = []):
        self.x = x
        self.y = y
        self.order = order
        self.linked = []

    def get_coords(self):
        return (self.x, self.y)

    def set_order(self, new_order):
        self.order = new_order

    def add_link(self, point):
        if point != self and point not in self.linked:
            self.linked.append(point)


def distance(p1, p2):
    return (((p2.x - p1.x) ** 2) + ((p2.y - p1.y) ** 2)) ** 0.5


def boxLengths(lengths):
    lengths.sort()
    return lengths[:-1]


def getRatio(lengths):
    return lengths[1] / lengths[0]

def crop_center(x_tl, y_tl, k_size):
    return (x_tl + (k_size//2) + 1, y_tl + (k_size//2) + 1)

def if_close_average(point1, point2, prox):
     distance = (((point2.x - point1.x) ** 2) + ((point2.y - point1.y) **
2)) ** 0.5
    if distance < prox:
            return True, Point((point1.x + point2.x) // 2, (point1.y +
point2.y) // 2)
    else:
        return point1, point2

def get_points_distance(point1, point2):
     distance = (((point2.x - point1.x) ** 2) + ((point2.y - point1.y) **
2)) ** 0.5
    return (distance, point1, point2)

def set_point_order(point, lst, prox):
    order = 0
    distances = []
```

```
    # initial ordering
    for p in lst:
        if point == p:
            continue
        distances.append(get_points_distance(point, p))
    distances.sort(key=lambda x:x[0])
    for tup in distances:
        if tup[0] <= prox:
            order += 1

    point.set_order(order)
    return order

def order_account_angle(point, lst, prox, angle_thresh):
    within_prox = []
    distances = []
    angles = []
    linked = []

    # consider distances first:
    for p in lst:
        if p == point:
            continue
        distances.append(get_points_distance(point, p))
    distances.sort(key=lambda x:x[0])

    for tup in distances:
        if tup[0] <= prox:
            within_prox.append(tup[2])

    # consider angle now
    for p in within_prox:
        if p == point:
            continue
        # get relative point:
        x = p.x - point.x
        y = p.y - point.y

        # get angle
        angle = degrees(atan2(y, x))
        if angle < 0:
            angle = 360 + angle

        if len(angles) == 0:
            angles.append(angle)
            point.add_link(p)
            point.order += 1
        else:
            for a in angles:
                if a - angle_thresh <= angle <= a + angle_thresh:
                    continue
                else:
                    angles.append(angle)
                    point.add_link(p)
                    point.order += 1
                    break

def point_to_list_dst(point, lst):
    distances = []
    for p in lst:
```

```
        if point == p:
            continue
        distances.append(get_points_distance(point, p))
    distances.sort(key=lambda x:x[0])
    return distances

def cluster_finder(point, lst, prox):
    cluster = []
    distances = []

    for p in lst:
        if point == p:
            continue
        distances.append(get_points_distance(point, p))
    distances.sort(key=lambda x:x[0])

    if distances[0][0] <= prox:
        cluster.append(distances[0][1])
        cluster.append(distances[0][2])
    else:
        return [point]

    for i in range(1, len(distances)):
        dec = []
        for point in cluster:
               dec.append(get_points_distance(point, distances[i][2])[0] <=
prox)
        if False not in dec:
            cluster.append(distances[i][2])

    return cluster

def avg_from_cluster(cluster):
    x = []
    y = []
    for point in cluster:
        x.append(point.x)
        y.append(point.y)
    return Point(sum(x)//len(x), sum(y)//len(y))

def save_im(image, name):
    cv2.imwrite('./{}.png'.format(name), image)


# Dirs for SVM data
NOISE_DIR = './svm_seg/noise'
CRACK_DIR = './svm_seg/crack'
INPUT_DIR = './svm_seg/input'
CLF_PATH = './svm_seg/clf_c_0.001_g_1000.joblib'

images_paths = list(paths.list_images(INPUT_DIR))

# load classifier
classifier = load(CLF_PATH)

# initialize iterator
iterator = 0
# set proximity value
prox = 20
# set degrees threshold
deg_thresh = 10
```

```
# Open image
for path in images_paths:
    image = cv2.imread(path)

    # normalize, turn to grayscale & apply blur
    image_norm = cv2.normalize(image,  None, 0, 255, cv2.NORM_MINMAX)
    image_gray = cv2.cvtColor(image_norm, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(image_gray, (9, 9), 0)

    # Apply adaptive threshilding
                thresholded    =    cv2.adaptiveThreshold(blurred,    255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 13, 2)
    # filter out noise
    out_initial = np.zeros(image.shape[:2])
       contours,  hierarchy  =  cv2.findContours(thresholded,  cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    image2 = np.zeros((20, 20))

    for contour in contours:
        bounding_box = cv2.boundingRect(contour)
        x, y, w, h = bounding_box

        # filter out small particles:
        if w > 5 or h > 5:
            cv2.drawContours(out_initial, [contour], -1, 255, -1)

    # apply dilatation and erosion
    kernel_dilatation = np.ones((4, 4), np.uint8)
    kernel_erosion = np.ones((2, 2), np.uint8)
    after_morpho = cv2.dilate(out_initial, kernel_dilatation, iterations=2)
              after_morpho    =    cv2.erode(after_morpho,    kernel_erosion,
iterations=3).astype('uint8')

    # filter out by ratio
    out_final = np.zeros(image.shape[:2])
      contours,  hierarchy  =  cv2.findContours(after_morpho,  cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    save_im(out_initial, 'init')
    save_im(thresholded, 'thresh')

    for contour in contours:
        bounding_box = cv2.boundingRect(contour)
        x, y, w, h = bounding_box
        iterator += 1

        # consider only big elements
        if w > 50 or h > 50:
            rect = cv2.minAreaRect(contour)
            box = cv2.boxPoints(rect)
            box = np.int0(box)

            # assess data with svm
            temp_image = np.zeros(image.shape[:2], np.uint8)
            cv2.drawContours(temp_image, [contour], -1, 255, -1)
                                                              strip      =
temp_image[bounding_box[1]:bounding_box[1]+bounding_box[3],
bounding_box[0]:bounding_box[0]+bounding_box[2]]
                                    strip    =    cv2.resize(strip,    (50,    50),
interpolation=cv2.INTER_NEAREST)
            strip = np.reshape(strip, strip.size)
```

```
            prediction = classifier.predict([strip])

            if prediction == 1:
                cv2.drawContours(out_final, [contour], -1, 255, -1)

    out_s1 = np.zeros(image.shape[:2], np.uint8)
    out_s2 = np.zeros(image.shape[:2], np.uint8)
    # Now cracks are shown as objects on out_final
    # It's time to sgment them into series of polylines
    # select kernel for segmenting crack
    k_size = 11
    # filter out contours again:
       contours, hierarchy = cv2.findContours(out_final.astype('uint8'),
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    # add iterators for scanning image
    x_tl = 0
    y_tl = 0
    # add ending bools
    x_end = False
    y_end = False
    # and iterate for every crack
    for contour in contours:
        # add temp image for every crack:
        temp_crack_image = np.zeros(image.shape[:2], np.uint8)
        cv2.drawContours(temp_crack_image, [contour], -1, 255, -1)

        # initial Points list for every contour
        points_initial = []

        while True:
            crop = temp_crack_image[y_tl:y_tl+k_size, x_tl:x_tl+k_size]

            # get centroid of object in crop
            if 255 in crop:
                M = cv2.moments(crop)
                cX = int(M["m10"] / M["m00"])
                cY = int(M["m01"] / M["m00"])
                points_initial.append(Point(x_tl + cX, y_tl + cY))
                cv2.circle(out_s1, (x_tl + cX, y_tl + cY), 1, 255, -1)

            x_tl += k_size

            if x_tl > image.shape[1] and not x_end:
                x_tl = image.shape[1] - k_size
                x_end = True
            elif x_end:
                x_tl = 0
                y_tl += k_size
                x_end = False

            if y_tl > image.shape[0] and not y_end:
                y_tl = image.shape[0] - k_size
                y_end = True
            elif y_end:
                x_tl = 0
                y_tl = 0
                y_end = False
                x_end = False
                break

        # erase close points
```

```
        points_filtered = []

        for x in range(2):
            if x != 0:
                points_initial = points_filtered
                points_filtered = []
            while len(points_initial) >= 2:
                distances = []
                for i in range(0, len(points_initial) - 1):
                    for j in range(i + 1, len(points_initial)):
                        distances.append(get_points_distance(points_initial
[i], points_initial[j]))

                distances.sort(key=lambda x:x[0])

                dec, point = if_close_average(distances[0][1], distances[0]
[2], prox)
                if type(dec) == bool:
                    points_filtered.append(point)
                else:
                    points_filtered.append(dec)
                    points_filtered.append(point)
                points_initial.remove(distances[0][1])
                points_initial.remove(distances[0][2])

            points_filtered = list(set(points_filtered))

        for point in points_filtered:
            cv2.circle(out_s2, point.get_coords(), 1, 255, -1)

        # initially order points
        for point in points_filtered:
            order_account_angle(point, points_filtered, 1.5 * prox, 60)
            #set_point_order(point, points_filtered, 1.5 * prox)

        # show ordering on image
        for point in points_filtered:
              cv2.putText(out_s2, str(point.order), point.get_coords(), 2,
.5, 255)

        st1 = np.hstack([image_gray, out_s2])
    cv2.imshow("out1", st1)
    cv2.waitKey(0)

                  out_final    =    cv2.erode(out_final,    kernel_erosion,
iterations=1).astype('uint8')
    stacked = np.hstack([image_gray, out_final])

    cv2.imshow("out", stacked)
    cv2.waitKey(0)

save_im(out_final, 'out')

print("That's all folks!")

for p in points_filtered:
    for l in p.linked:
        if type(l) == Point:
            leng += 1
print(leng)
```