

**POLITECHNIKA KRAKOWSKA im. T. Kościuszki**



**Wydział Inżynierii Lądowej**

**Katedra Technologii Informatycznych**



**w Inżynierii L-10**

**Kierunek studiów:** Budownictwo

**Specjalność:** Budowle – informacja i modelowanie (BIM)

## **PRACA DYPLOMOWA MAGISTERSKA**

**Jakub Niemiec**

Nr Albumu: **141022**

GŁĘBOKIE SIECI NEURONOWE W OCENIE STANU TECHNICZNEGO  
OBIEKTÓW BUDOWLANYCH

DEEP NEURAL NETWORKS IN STRUCTURAL HEALTH MONITORING

Promotor: dr hab. inż. Marek Słoński, prof. PK

Ocena pracy: ...

Podpis promotora: ...

Data: ...

Kraków, czerwiec 2022

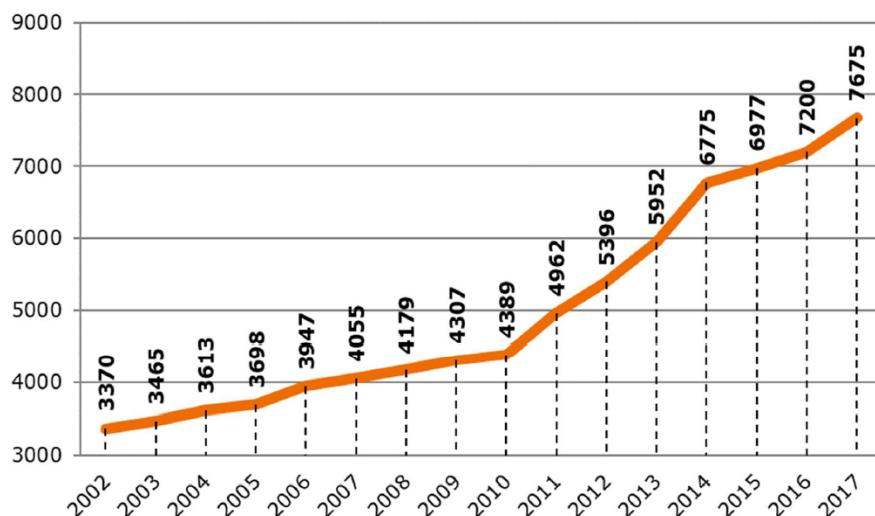
*Za cenne wskazówki podczas pisania pracy pragnę  
podziękować panu dr hab. inż. Markowi Słońskiemu*

# Spis treści

<b>1 Wprowadzenie</b>	<b>4</b>
1.1 Cel i zakres pracy . . . . .	8
<b>2 Przedstawienie problemu</b>	<b>9</b>
2.1 Segmentacja obrazu . . . . .	9
2.2 Głębokie sieci neuronowe . . . . .	10
2.2.1 Warstwa konwolucyjna . . . . .	12
2.2.2 Warstwa łącząca . . . . .	13
2.2.3 Odwrócona konwolucja . . . . .	14
2.2.4 Funkcje aktywacyjne . . . . .	14
2.3 U-Net . . . . .	17
2.4 Attention U-Net . . . . .	19
2.5 TransUnet . . . . .	22
<b>3 Definiowanie modeli i implementacja algorytmu mierzenia pęknięć</b>	<b>25</b>
3.1 Zbiór danych . . . . .	25
3.2 Funkcja kosztu . . . . .	27
3.3 Metryki oceny zachowania modelu . . . . .	27
3.4 Zestawienie wyników segmentacji obrazu . . . . .	30
3.5 Algorytm mierzenia pęknięć . . . . .	34
<b>4 Analiza obrazu laboratoryjnego</b>	<b>40</b>
<b>5 Podsumowanie i wnioski</b>	<b>44</b>

# Wprowadzenie

Zgodnie z raportem [30] Generalnej Dyrekcji Dróg Krajowych i Autostrad liczba obiektów mostowych w 2017 roku wynosiła 7675. Na rysunku 1.1 można zauważyć znaczny wzrost liczby obiektów od 2010 roku. Zgodnie z zapisem ustawy Prawo budowlane [16] nakłada na właściciela lub zarządcę obiektu budowlanego obowiązek wykonywania okresowych kontroli, polegających w szczególności na sprawdzeniu stanu technicznego obiektu, ponadto ustawa o drogach publicznych [15] wśród zadań zarządcy drogi wymienia przeprowadzanie okresowych kontroli stanu drogowych obiektów inżynierskich. W związku z dynamicznym wzrostem ilości obiektów, pojawia się zadanie sprawdzenia i utrzymania odpowiedniego stanu konstrukcji.



Rysunek 1.1: Zmiana liczby obiektów mostowych w czasie. Źródło: [30]

W tradycyjnej ocenie stanu technicznego, oceniane są wszystkie elementy składające się na obiekt inżynierski, znaczna ich część a szczególnie te które są krytyczne dla bezpieczeństwa konstrukcji są elementami betonowymi. Ocena odbywa się w sposób wizualny w której sprawdzany jest szereg czynników składających się na wynik końcowy. Każdy z elementów klasyfikowany jest według sześciostopniowej skali punktowej od 0 do 5 przedstawionej w tabeli 1.1 która wskazuje na obecny stan elementu/obiektu. Wizualna ocena jest czasochłonna, narażona na subiektywną ocenę osoby przeprowadzającej sprawdzenie, jej kompetencje i spostrzegawczość. Aby zapewnić

jak najlepsze rezultaty w zadaniu oceny stanu technicznego rozwój uczenia maszynowego dostarcza narzędzia które pozwalają taki proces zautomatyzować, użycie autonomicznych urządzeń z systemem rozpoznawania pęknięć pozwala dostrzec uszkodzenia w miejscach ciężko dostępnych ze względu na usytuowanie obiektu, np.: na podporach pośrednich w długich obiektach mostowych.

Tabela 1.1: Skala i kryteria ocen elementów. Źródło: [4]

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 obniżające przydatność użytkową, ale możliwe do naprawy
1	przedawaryjny	wykazuje nieodwracalne uszkodzenia dyskwalifikujące przydatność użytkową
0	awaryjny	uległ zniszczeniu lub przestał istnieć

Tabela 1.2: Przykładowa karta oceny przyczółka. Źródło: [17]

Lp.	Rodzaj uszkodzeń	Zakres uszkodzeń [%]					Przykładowe kody uszkodzeń
		0	≤ 5	10	20	≥ 30	
1	Zanieczyszczenia, wegetacja roślin	5	4	3			NB, NC, NK, WB, WC, WK
2	Przeciek, zacieki	5	4	3			CB, CC, CK
3	Korozja materiału konstrukcyjnego:	a osady, wykwity	5	4	3		OB, OC, OK, KB, KC, KK
		b uszczerbienie, miejscowe zniszczenie struktury materiału	5	4	3	2	
4	Korozja zbrojenia	5	3	2	1	0	KZ
5	Rysy:	a skurczowe (powierzchniowe)	5	4	3	2	RB, RK, RC
		b wzdułz korodującego zbrojenia	5		3	2	
		c powstałe na skutek przeciążenia	5		2		
6	Pęknięcia w strefie decydującej o nośności podpory	5		1			RB, RK, RC
7	Ubytki materiału konstrukcyjnego	5	3	2	1	0	UB, UC, UK
8	Przemieszczenie (osiadanie i/lub obrót)	5	2	1	0	0	PB, PC, PK
9	Uszkodzenie ciosu podłożyskowego (korozja, ubytki betonu, rysy, pęknięcia)	5	3	2	1	0	UB, UK, UC, RB, RK, RC, KB, KK, KC
10	Podmycie fundamentu	5	2	2	1	0	UT
11	Odsłonięcie skrzydła zagrażające trwałości/stateczności konstrukcji	5	3	3	2	2	UT, PT

Tabela 1.2 przedstawia przykładową kartę oceny przyczółka mostowego, wiele z analizowanych kategorii związanych jest z uszkodzeniem betonu.

Pęknięcie/zarysowanie betonu jest jednym z pierwszych znaków sygnalizujących pogorszenie stanu konstrukcji, zlekceważenie wczesnych objawów może prowadzić do poważnych zagrożeń, których skutkiem mogą być wysokie koszty naprawy w zaawansowanym stopniu uszkodzenia, a nawet katastrofa budowlana. W pracach [52],[51],[35] przedstawiono metody próby wykrywania pęknięć za pomocą metod laserów podczerwieni, termicznych czy radiologicznych. Jednak wraz z rozwojem analizy obrazu i dziedziny uczenia maszynowego, coraz więcej badań w tym temacie wykorzystuje analizę obrazu. Automatyczne wykrywanie pęknięć w elementach betonowych może

być przeprowadzany analizując obrazy które pozwalają na dostarczenie informacji jak długość, szerokość oraz charakter uszkodzenia.

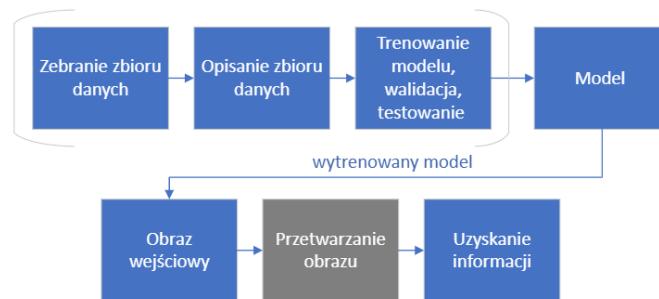
Metody pozyskiwania takich informacji można podzielić na dwa typy: przetwarzanie obrazu i uczenie maszynowe.

Metody przetwarzania obrazu bazują na operacjach morfologicznych i stosowaniu filtrów, w tym procesie obraz o wysokiej rozdzielcości poddawany jest opisany przekształceniom, usunięciu szumów, po czym możliwa jest detekcja np.: konturów pęknięcia, zaletą takiego podejścia jest brak konieczności zbierania dużej ilości danych aby wytrenować model co jest konieczne w przypadku podejścia z użyciem uczenia maszynowego. Jakość obrazu, kontrast z otoczeniem i warunki wykonywania zdjęć mają kluczowe znaczenie, ponieważ jedną z podstawowych operacji jest progowanie obrazu (ang. thresholding) - operacja przekształcania obrazu na binarny z wykorzystaniem punktu progowego według którego piksele są klasyfikowane na podstawie swojej intensywności. [28], [57], [39], [50]



Rysunek 1.2: Schemat metody opartej na przetwarzaniu obrazu. Źródło: Opracowanie własne

Metody oparte na uczeniu maszynowym wymagają zebrania odpowiedniego zbioru danych w celu trenowania i testowania modelu. Takie podejście może wymagać przetwarzania obrazu jednak jest ono wykorzystywane już po uzyskaniu obrazu z wyselekcjonowanym pęknięciem, które odbywa się automatycznie. [25], [60], [62]



Rysunek 1.3: Schemat metody opartej na uczeniu maszynowym. Źródło: Opracowanie własne

Zrealizowane badania z użyciem powyższych metod oraz ich porównanie przedstawiono w pracy [44].

Uczenie głębokie dla zadań klasyfikacji oraz segmentacji pozwala uzyskać optymalne wyniki [38]. W pracy [22] zastosowano metodę okna przesuwnego które przechodząc zdjęcie wycinek po wycinku pozwala zidentyfikować pęknięcie. Jednak zaproponowane podejście pozwalało jedynie na klasyfikację obrazu (pęknięty/nie pęknięty). Aby wykonać analizę na poziomie poszczególnych pikseli w pracy [60] przedstawiono metodę która pozwala na analizę wycinków oraz przypisanie poszczególnych pikseli do klas, jednak takie rozwiązanie było ubogie w globalne informacje przestrzenne co wpływało na rezultaty - poszczególne wycinki obrazu nie wiedziały o swoim istnieniu. Drugą wadą było zawyżanie rozmiaru pęknięcia. Praca [61] wykorzystuje CNN jako klasyfikatora, a do wyodrębniania pęknięć stosuje metody przetwarzania obrazu co czyni ją niezdolną do generalizacji - sieć CNN wymaga konkretnego rozmiaru obrazu wsadowego oraz parametry wykorzystywane w przetwarzaniu muszą być indywidualne dostosowane do każdego obrazu.

W najnowszych badaniach o analogicznej tematyce do zbierania danych powszechnie wykorzystywane są drony (UAV) ([37], [41]), które pozwalają na sprawne wykonywanie zdjęć z zachowaniem określonego kąta nachylenia w stosunku do powierzchni oraz stałej odległości od elementu, te informacje są kluczowe w określaniu realnego rozmiaru pęknięcia. Badania wykonane przez Uniwersytet w Zhejiangu [58], z wykorzystaniem architektury Swin Transformer [41] oraz danymi zebranymi z pomocą UAV na kampusie uczelni pozwoliły na wykonanie modelu na podstawie którego błąd względny między pęknięciem na obrazie przewidzianym a pęknięciem rzeczywistym wyniósł 5%.

Obszerne zestawienie metod i prac związanych z oceną stanu technicznego z wykorzystaniem podejścia opartego na danych (data-driven approach) oraz uczeniu głębokim można znaleźć w artykule [18].

## **1.1. Cel i zakres pracy**

Celem pracy jest stworzenie modelu który pozwala na wykrywanie pęknięć w elementach betonowych oraz algorytmu pozwalającego na określenie ich szerokości.

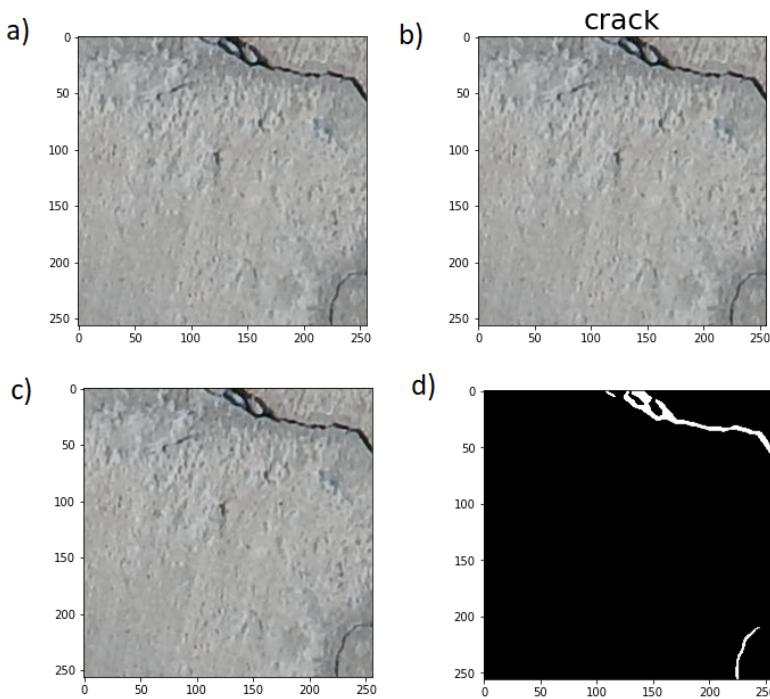
Zakres pracy obejmuje analizę oraz porównanie dostępnych architektur modeli uczenia głębokiego, rozszerzenie publicznie dostępnego zbioru danych przez opisanie zdjęć - tworząc tym samym wyodrębniony zbiór który może być wykorzystany w zadaniu wykrywania pęknięć.

Rozdział 2 poświęcony jest opisaniu problematyki zadania oraz części teoretycznej. Rozdział 3 i 4 obejmuje praktyczną implementację modeli opisanych w rozdziale 2, przedstawienie zbioru danych oraz propozycję algorytmu pomiaru pęknięć.

# Przedstawienie problemu

## 2.1. Segmentacja obrazu

Segmentacja obrazu jest jednym z kluczowych problemów w widzeniu komputerowym, ma na celu zapewnić informacje o analizowanym obrazie, używana jest powszechnie w analizie obrazów medycznych, autonomicznych pojazdach. W odróżnieniu od klasyfikacji obrazu, w wyniku której obraz zostaje przypisany do danej kategorii 2.1(b), segmentacja jest procesem grupowania w wyniku którego każdy piksel zostaje przypisany do odpowiedniej klasy, piksele należące do tej samej klasy mają tę samą wartość.



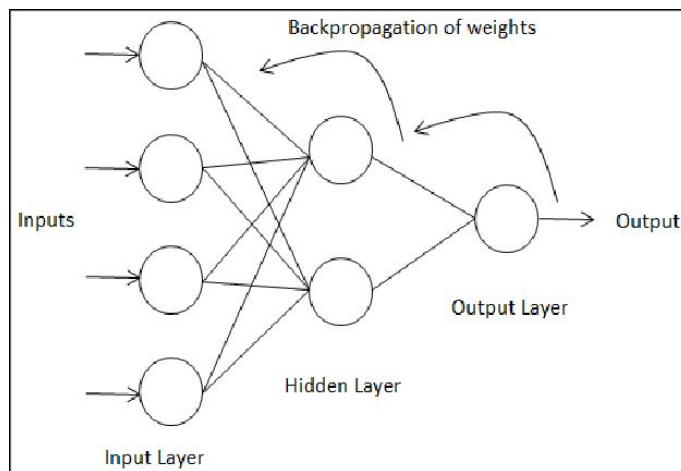
Rysunek 2.1: Dane wejściowe-wyjściowe w: klasyfikacji obrazu (a-b), segmentacji semantycznej (c-d). Źródło: opracowanie własne

Wynik segmentacji przedstawiono na obrazie 2.1(d) gdzie kolorem czarnym oznaczone są piksele związane z tłem, natomiast białym piksele przypisane do klasy pęknięcie. Metody segmentacji można podzielić na oparte na obszarach - bazujące na sąsiedztwie pod kątem jednorodności pikseli oraz na oparte na krawędziach - bazujące na wyznaczaniu granic obiektów na obrazie.

## 2.2. Głębokie sieci neuronowe

Jednokierunkowe sieci neuronowe (ang. feed-forward neural network) składają się z dyskretnych warstw połączonych ze sobą, zwykle w skład wchodzą: warstwa wejściowa - odbiera sygnał wejściowy i przekazuje go dalej bez zmian, warstw ukrytych które wykonują operacje na otrzymanych danych i przekazują je dalszym warstwom, warstwy wyjściowej która generuje ostateczne wartości. [31] Sieci neuronowe trenowane są na podstawie danych, powszechnie stosowany jest algorytm propagacji wstecznej.

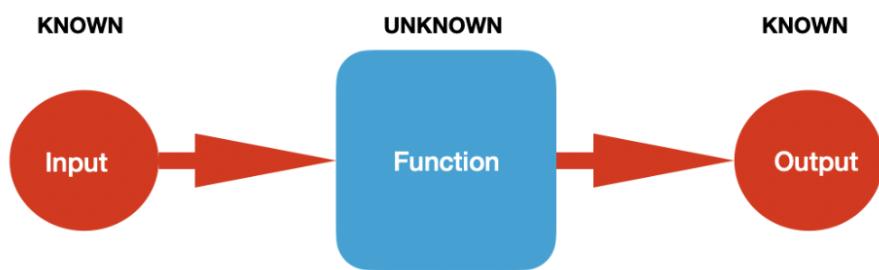
1. Do modelu wprowadzane są dane wejściowe, w celu uzyskania wartości wyjściowych neuronów
2. Zwracana jest różnica (błąd) między wyjściem a danymi rzeczywistymi
3. Obliczany jest gradient błędu w postaci funkcji wag neuronu, który następnie jest modyfikowany tak aby zminimalizować uzyskany błąd
4. Propagacja błędu wyjściowego aby oszacować błąd warstw ukrytych
5. Obliczane są gradienty błędów oraz modyfikowane są wagi neuronów warstw ukrytych



Rysunek 2.2: Schemat propagacji wstecznej. Źródło: [43]

Sieci głębokie [33] poprawiają wyniki uczenia się dzięki stopniowemu i hierarchicznemu procesowi ekstrakcji cech z danych wejściowych. Głębokie sieci konwolucyjne stopniowo filtryują części danych wejściowych, w procesie uczenia wyłapują cechy obrazu do rozpoznawania wzorców. Model CNN poszukuje cech różnicujących, pewnych, niezmiennych.

Uczeniem nadzorowanym (2.3) nazywany jest proces, w którym do modelu dostarczane są opisane przez użytkownika dane, które należy podzielić na zbiór treningowy oraz walidacyjny zachowując odpowiednie proporcje. Wprowadzając do sieci informacje w parach (wejście-wyjście), możliwe jest uczenie modelu na zbiorze treningowym oraz kontrola uzyskiwanych wyników przez sprawdzanie jego działania na zbiorze walidacyjnym. Celem jest znalezienie funkcji odwzorowującej dane wejściowe na dane wyjściowe, dla nieznanych wcześniej przykładów. Ocena modelu odbywa się przez pomiar zdefiniowanych metryk (np. IoU 3.3 lub dokładność), które kalkulowane są w procesie uczenia po każdej wykonanej epoce.



Rysunek 2.3: Schemat uczenia nadzorowanego. Źródło: [46]

## 2.2.1. Warstwa konwolucyjna

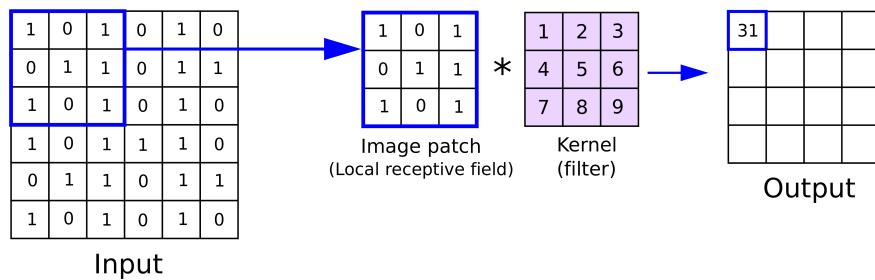
Warstwa konwolucyjna (ang. Convolutional Layer) [42] stanowi podstawową warstwę sieci neuronowych, zawiera wyuczone filtry (kernele), których zadaniem jest wyodrębnianie cech obrazu poprzez obliczanie iloczynu skalarnego. Operacja wykonywana jest używając okna przesuwnego o określonym rozmiarze (ang. kernel size) oraz kroku (ang. stride). Istotne parametry:

1. Kernel size - rozmiar filtra, stosując mniejszy rozmiar możliwe jest uzyskanie głębszej struktury co pozwala na lepsze wyodrębnianie cech analizowanego obrazu. Wybór rozmiaru filtra uzależniony jest od problematyki zadania.
2. Stride - określa krok o jaki okno przesuwa się po obrazie.
3. Padding - definiuje zachowanie na granicy obrazu, w segmentacji obrazu najczęściej przyjmowana jest wartość "same" oznaczająca lustrzane odbicie obrazu względem obramowania aby zachować kontekst zadania.

Rysunek 2.4 przedstawia działanie warstwy konwolucyjnej, otrzymana wartość opisana jest wzorem 2.1

$$output = \text{image value} \times \text{kernel value} \quad (2.1)$$

$$output = (1 \times 1) + (0 \times 2) + (1 \times 3) + (0 \times 4) + (1 \times 5) + (1 \times 6) + (1 \times 7) + (0 \times 8) + (1 \times 9) = 31$$



Rysunek 2.4: Operacja konwolucyjna. Źródło: [47]

## 2.2.2. Warstwa łącząca

Warstwy łączące (ang. Pooling Layer) [42] wykorzystywane są do stopniowego zmniejsza rozmiaru obrazu, co jednocześnie zmniejsza ilość parametrów do wytrenowania oraz upraszcza model. Warstwy łączące pozwalają wydobyć najważniejsze informacje. Przy zdefiniowanym rozmiarze filtra oraz kroku, okno przechodzi po obrazie mapując piksele do nowej wartości. Najważniejszym powodem stosowania warstw łączących jest uchronienie modelu przed przeuczeniem [33]. Najczęściej wykorzystywane są dwa rodzaje warstw łączących:

- Max-pooling - polega na wyciągnięciu maksymalnej wartości piksela z analizowanego okna.

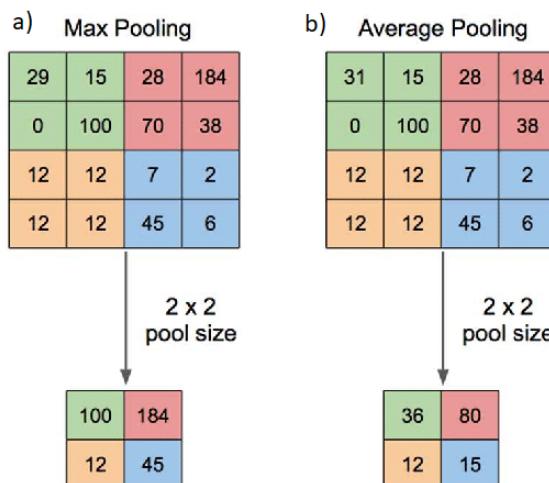
$$pool\ value = \max\{values\} \quad (2.2)$$

$$poolvalue = \max\{28; 184; 70; 38\} = 184 \quad (2.5a)$$

- Average-pooling - polega na wyciągnięciu średniej wartości z analizowanych pikseli w oknie.

$$pool\ value = \frac{1}{size^2} \sum_{i=i}^{size^2} values_i \quad (2.3)$$

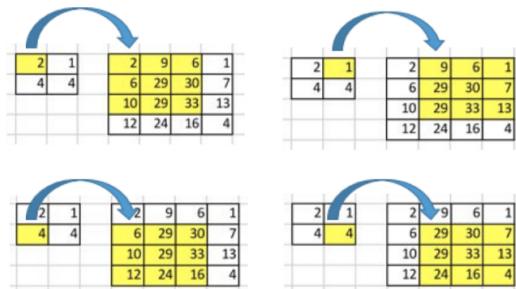
$$poolvalue = \frac{1}{2^2}(28 + 184 + 70 + 38) = \frac{28 + 184 + 70 + 38}{4} = 80 \quad (2.5b)$$



Rysunek 2.5: a) przedstawienie graficzne Max Pooling b) przedstawienie graficzne Average Pooling. Źródło: [59]

### 2.2.3. Odwrócona konwolucja

Odwrócona konwolucja (ang. Transposed convolution) [14] jest operacją która umożliwia odwrócenie "konwolucji", co pozwala przeprowadzić upsampling wraz z trenowalnymi parametrami (co wyklucza konieczność interpolacji wartości). Głównym założeniem jest mapowanie jednej wartości do wielu, dzięki czemu możliwe jest zwiększenie rozmiaru obrazu.

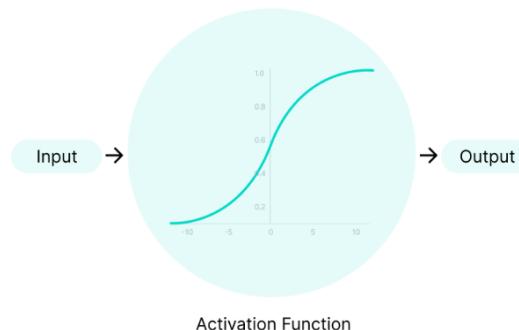


Rysunek 2.6: Odwrócona konwolucja. Źródło: [14]

### 2.2.4. Funkcje aktywacyjne

Funkcja aktywacji [19] służy do obliczania wartości wyjściowej neuronu. Podstawową rolą funkcji aktywacji jest przekształcenie zsumowanego ważonego wejścia z węzła na wartość wyjściową, która ma być przekazana do następnej warstwy ukrytej lub jako wyjście.

- ciągłe przejście pomiędzy swoją wartością maksymalną a minimalną (np. 0 - 1)
- łatwa do obliczenia i ciągła pochodna
- możliwość wprowadzenia do argumentu parametru ustalającego kształt krzywej

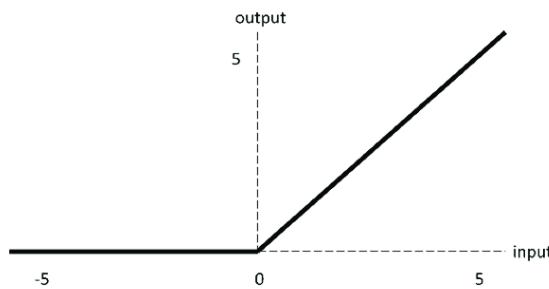


Rysunek 2.7: Aktywacja. Źródło: [19]

**ReLU** (ang. Rectified Linear Unit) [19] - opisana wzorem 2.4 Eliminuje problem

zaniekujących gradientów (pochodne zawsze znajdują się w zakresie  $[0, 1]$ , ich wieleokrotne mnożenie prowadzą do bardzo małych liczb powodujących znikome zmiany wag w warstwach neuronów). Strategia wykorzystująca ReLU opiera się na szkoleniu cech dzięki rzadszym aktywacjom jednostek, ponieważ gdy wartość funkcji jest równa 0, nie propagujemy sygnału do połączonych neuronów. Rezultatem używania funkcji ReLU jest to szybszy proces uczenia.

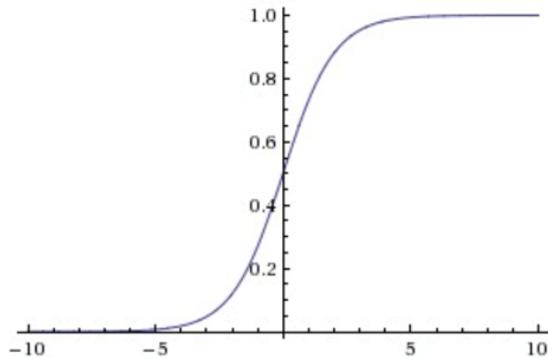
$$f(x) = \max(0, x) \quad (2.4)$$



Rysunek 2.8: ReLU. Źródło: [55]

**Sigmoid** [19] - opisana wzorem 2.5. Przyjmuje dowolną wartość rzeczywistą jako wejście, wartości wyjściowe zawierają się w zakresie od 0 do 1, jest różniczkowalna i zapewnia płynny gradient, tj. zapobiega skokom wartości wyjściowych. Im większe wejście, tym wartość wyjściowa będzie bliższa 1, natomiast im mniejsze wejście, tym wartość wyjściowa będzie bliższa 0. Ze względu na wartości wyjściowe jest powszechnie używana w modelach których zadaniem jest przewidzieć prawdopodobieństwo, ponieważ wartość prawdopodobieństwa zawiera się w zakresie 0-1.

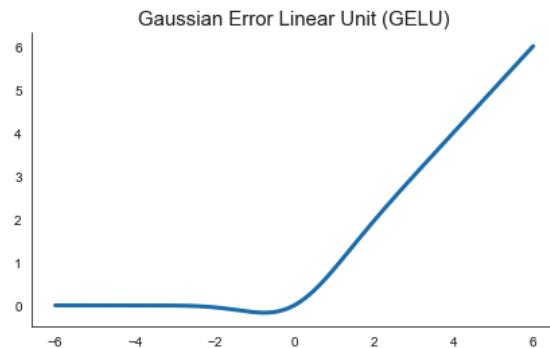
$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.5)$$



Rysunek 2.9: Sigmoid. Źródło: [1]

**GELU** (ang. Gaussian Error Linear Unit) [32] - opisana wzorem 2.6. Zaproponowana w 2016 roku, używana w złożonych modelach widzenia komputerowego oraz przetwarzania języka naturalnego np: transformers. GELU, w przeciwieństwie do funkcji ReLU, waży dane wejściowe według ich wartości, a nie znaku. W dziedzinie dodatniej ta niewypukła, niemonotoniczna funkcja nie jest liniowa - ma krzywiznę we wszystkich miejscach. W rezultacie GELU może być w stanie łatwiej aproksymować złożone funkcje. GELU poprawia wydajność we wszystkich zadaniach widzenia komputerowego, przetwarzania języka naturalnego i zadań głosowych.

$$f(x) = 0.5x(1 + \tanh(\sqrt{2/\pi}(x + 0.044715x^3))) \quad (2.6)$$

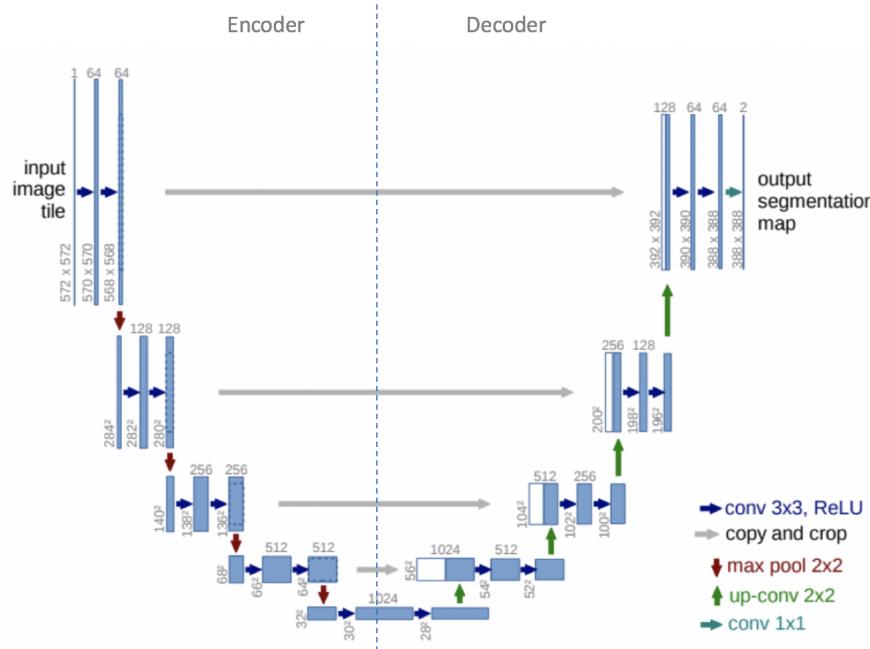


Rysunek 2.10: GELU. Źródło: [53]

## 2.3. U-Net

Sieć Unet [48] została opracowana w 2015 roku na Uniwersytecie we Fryburgu, z myślą o zastosowaniu w zadaniach przetwarzania obrazów medycznych. Do tej pory typowe sieci konwolucyjne zwracały głównie ogólną informację na temat obrazu - do jakiej klasy przynależy (klasyfikacja obrazu). Motywacją do powstania sieci Unet była poprawa wcześniejszej wykorzystywanej techniki segmentacji za pomocą okna przesuwnego (ang. sliding window), której głównymi wadami było wykorzystywanie dużych zasobów pamięci/mocy obliczeniowej oraz nie dość dobra precyzja lokalizacji obiektów, która w przetwarzaniu obrazów medycznych odgrywa kluczową rolę. Kolejnym powodem była niewielka ilość danych wejściowych, w oryginalnej dokumentacji podkreślono rolę jaką stanowi stosowanie rozszerzenia danych (ang. image augmentation).

Nazwa Unet pochodzi od charakterystycznego kształtu sieci, która formuje się w literę "U", architekturę można podzielić na dwie części - encoder (lewa strona) oraz decoder (prawa strona) (2.11).



Rysunek 2.11: Architektura sieci U-net. Źródło: [48]

Funkcją encodera jest analiza obrazu - odpowiedzenie na pytanie co znajduje się na obrazie, natomiast decoder ma za zadanie dostarczyć informację gdzie dokładnie dany obiekt się znajduje. Encoder i decoder połączone są tzw. "skip connections" (szare strzałki pokazane na obrazie 2.11) które umożliwiają połączenie informacji które do-

starcza encoder, w celu lokalizacji/klasyfikowania poszczególnych pikseli. Zaproponowana architektura nie zawiera warstw w pełni połączonych (ang. fully connected layers), jest to architektura w pełni konwolucyjna (ang. fully convolutional network).

$$conv\ 3 \times 3\ (ReLU) \rightarrow conv\ 3 \times 3\ (ReLU) \rightarrow max\ pool\ 2 \times 2 \quad (2.7)$$

Scieżka encodera 2.7 na każdym z poziomów składa się z dwóch operacji konwolucyjnych o rozmiarze  $3 \times 3$  i aktywacją z użyciem funkcji ReLU, a następnie operacji max-pooling o rozmiarze  $2 \times 2$  i kroku równym 2. Na każdym z poziomów rozmiar obrazu zmniejsza się dwukrotnie, a liczba warstw (ang. feature channels) się podwaja.

$$conv\ 3 \times 3\ (ReLU) \rightarrow conv\ 3 \times 3\ (ReLU) \rightarrow up\ pool\ 2 \times 2 \quad (2.8)$$

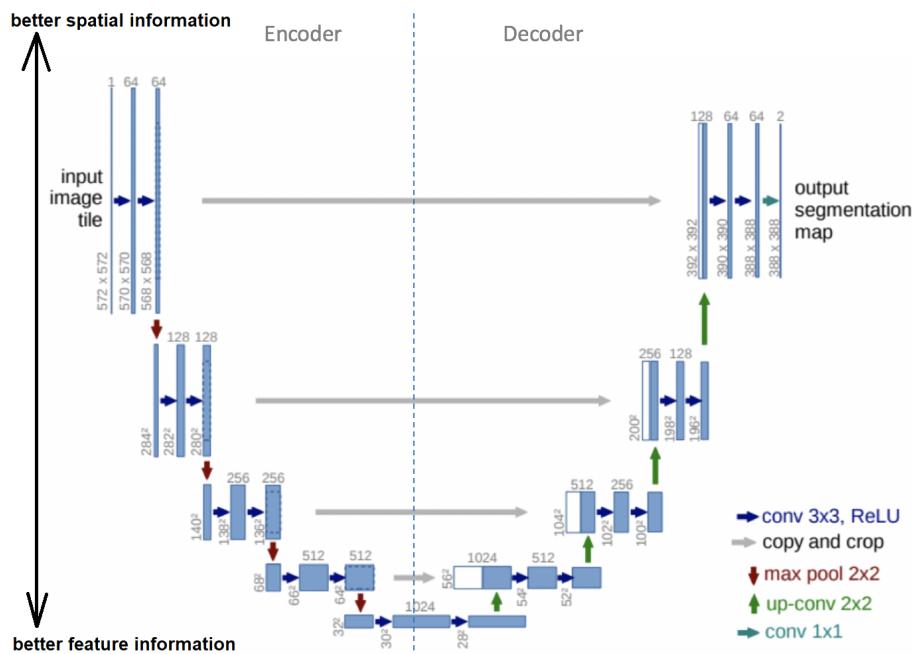
Ścieżka decodera 2.8 jest symetryczna do encodera, gdzie na każdym z poziomów wykonywane są dwie operacje konwolucyjne  $3 \times 3$ (ReLU) a następnie operacja up-conv  $2 \times 2$ , krok równy 2. Z każdym poziomem rozmiar obrazu zwiększa się dwukrotnie a liczba warstw (ang. feature channels) dwukrotnie zmiesza. Poszczególne poziomy powtarzane są aż do osiągnięcia rozmiaru obrazu wejściowego.

$$conv\ 3 \times 3\ (ReLU) \rightarrow conv\ 3 \times 3\ (ReLU) \rightarrow conv\ 1 \times 1\ (Sigmoid/Softmax) \quad (2.9)$$

Na ostatnim poziomie 2.9 wykonywane są dwie operacje konwolucyjne o rozmiarze  $3 \times 3$ , a następnie operacja konwolucyjna o rozmiarze  $1 \times 1$  (funkcja aktywacji zależna jest od rodzaju segmentacji, autorzy używają funkcji Softmax, lecz w przypadku segmentacji binarnej częściej używana jest funkcja Sigmoid), w celu zachowania pierwotnego rozmiaru, natomiast liczba warstw (ang. feature channels) odpowiada liczbie klas do której piksele mogą być klasyfikowane.

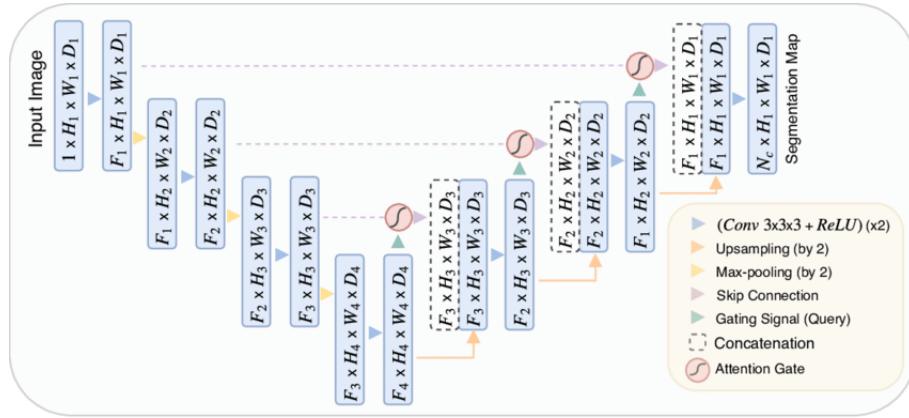
## 2.4. Attention U-Net

Sieć Attention Unet [45] została zaproponowana adresując wady, które występują w bazowej wersji Unet. Górnne warstwy sieci zawierają dokładniejsze informacje przestrzenne na temat obiektów, jednak brakuje im informacji odnośnie cech obrazu - sytuacja ulega odwróceniu wraz z kolejnymi poziomami (głębszymi warstwami) modelu 2.12.



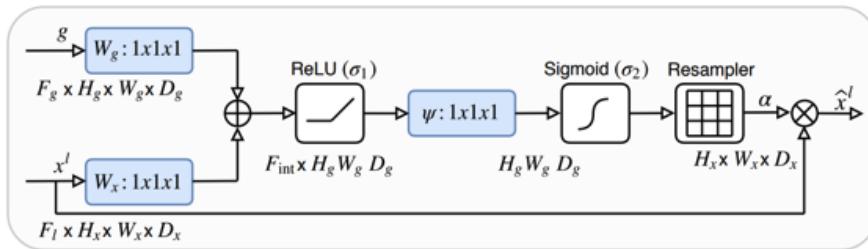
Rysunek 2.12: Zależności w strukturze Unet, Źródło: opracowanie własne

Autorzy zaproponowali użycie “attention gate” przy połączeniu skip connection 2.13. Attention gates wykorzystują “soft attention”: w ogólności je zadaniem jest nadawanie wag poszczególnym obszarom w obrazie (istotnie obszary posiadają większe wagi). Przez możliwą propagację wsteczną przypisane wagi są aktualizowane w procesie uczenia oraz optymalizowane aby podkreślać jedynie ważne elementy i im przypisywać jak najwyższe wagi, tłumiąc tym samym obszary nieistotne. W rezultacie każdy z pikseli w analizowanym obrazie posiada nadaną mu wagę która reprezentuje jego istotność w zadaniu segmentacji.

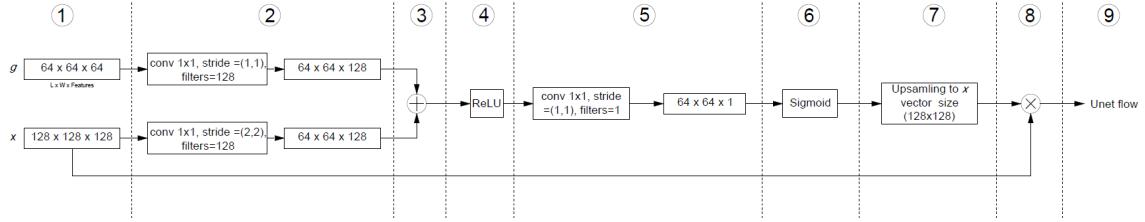


Rysunek 2.13: Architektura sieci Attention U-Net. Źródło: [45]

Działanie attention gate 2.14 przyjmując przykładowe dane wstępne można opisać w następujący sposób:



Rysunek 2.14: Attention gate schemat. Źródło: [45]



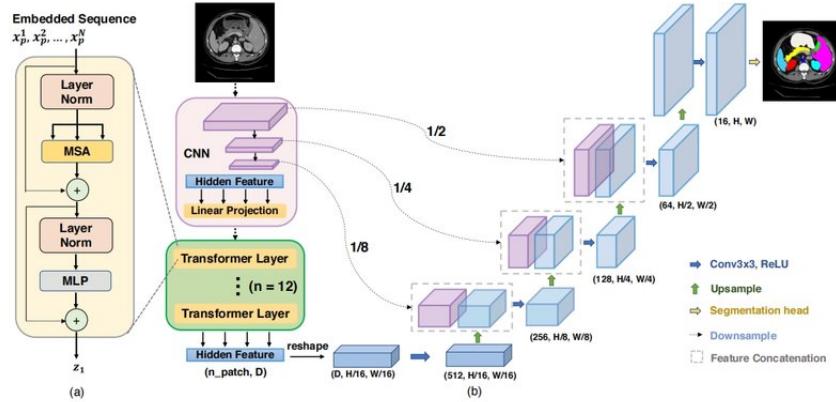
Rysunek 2.15: Attention gate schemat blokowy. Źródło: opracowanie własne

1. Dwa wektory wejściowe:  $g$  (gating signal) pochodzący z niższego poziomu sieci (zawiera lepsze informacje dotyczące cech) oraz  $x$  pochodzący z wyższego poziomu encodera - zawiera lepsze informacje przestrzenne.
2. Wykonywane są operacje konwolucyjne o rozmiarze  $1 \times 1$  na obu wektorach, przy czym operacja wektora  $g$  ma krok równy  $(1, 1)$ , natomiast wektora  $x$  krok  $(2, 2)$  - aby wyrównać rozmiary (ponieważ wektor  $g$  pochodzi z niższego poziomu, jego rozmiar jest dwa razy mniejszy niż rozmiar wektora  $x$ ).

3. Sumowanie po elementach wag obu wektorów w wyniku czego piksele o większej istotności mają przypisane większe wagi. Powstaje jeden wektor
4. Aktywacja funkcją *ReLU*
5. Operacja konwolucyjna  $1 \times 1$ , w wyniku której otrzymany jest wektor reprezentujący wagi (zakres  $0 - \infty$ )
6. Normalizacja wektora wag do zakresu  $(0 - 1)$  - aktywacja funkcją *Sigmoid*
7. Upsampling wektora wag to rozmiaru wektora  $x$
8. Mnożenie po elementach wektora  $x$  i tensora wag (*pixel value*  $\times$  *weight value*)
9. dalsze kroki Unet

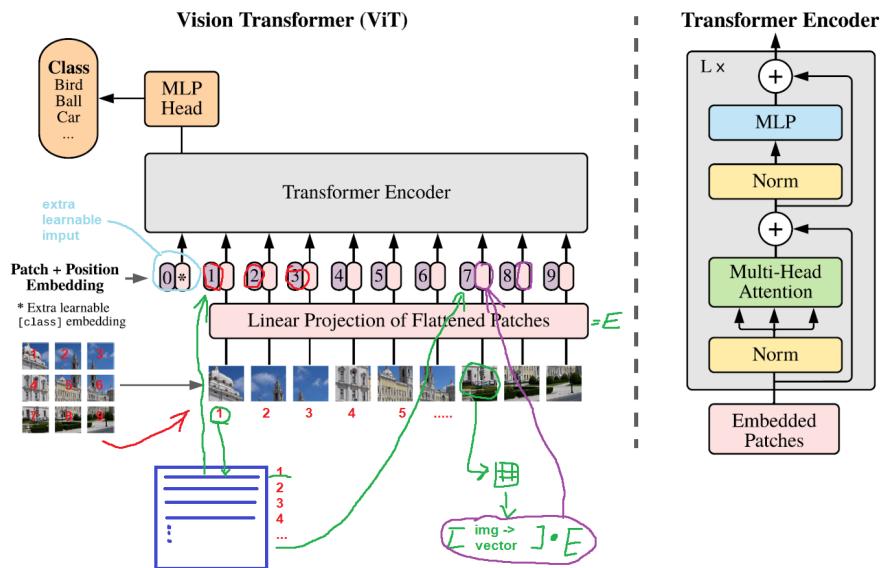
## 2.5. TransUnet

TransUnet [23] wykorzystuje w strukturze enkodera hybrydowe połączenie CNN oraz ViT (vision transformers). Architektura przedstawiona jest na rysunku 2.16.



Rysunek 2.16: Architektura sieci TransUnet. Źródło: [23]

ViT [27] jest modelem klasyfikacji obrazu, w którym obraz konwertowany jest na serie nie nachodzących na siebie części o rozmiarze, z których każda zawiera informacje o swoim położeniu (ang. patch embedding) zapewniając globalny kontekst dla innych części obrazu.



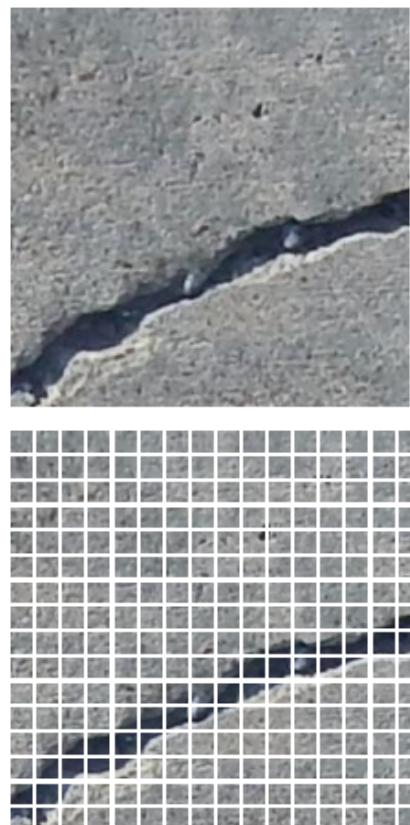
Rysunek 2.17: Architektura ViT. Źródło: [27]

1. Obraz dzielony jest na  $n$  części o rozmiarach  $(p, p, c)$  gdzie  $p$  jest definiowanym parametrem a  $c$  oznacza liczbę kanałów (3 dla obrazu kolorowego). Części numerowane są od góry do dołu.
2. Każda z części przekształcana jest do jednowymiarowego wektora, w wyniku powstaje  $n$  wektorów o wymiarach  $(1, p^2 \times c)$
3. Jednowymiarowe wektory mnożone są przez trenowalny tensor o wymiarach  $(p^2 \times c, d)$ , który liniowo rzutuje każdą z części obrazu na wymiar  $d$ . Wymiar  $d$  jest zdefiniowaną stałą. W rezultacie powstaje  $n$  części obrazu o wymiarach  $(1, d)$
4. Do wyniku wcześniejszej operacji dodawany jest trenowalny wektor  $(*)$  o wymiarach  $(1, d)$ .
5. W rezultacie otrzymany jest tensor o wymiarach  $(n + 1, d)$ , do którego dodawany jest tensor pozycyjny o wymiarach  $(n + 1, d)$

```

Image size: 256 X 256
Patch size: 16 X 16
Patches per image: 256
Elements per patch: 768

```



Rysunek 2.18: Wizualizacja podziału obrazu na części o rozmiarze 16x16px. Źródło: opracowanie własne

W kolejnym kroku używając transformer encoder który za wartość wejściową przyjmuje tensor embeded patches sieć uczy się odnajdywać pożądane cechy w wydzielonych częściach obrazu. Encoder zawiera warstwę normalizującą, MHA oraz MLP. MHA (Multi-Head Attention) wykonuje operacje  $N$  razy (num heads) zwracając wartości wag dla każdej z części obrazu. Wielokrotne wykonanie obliczeń pozwala osiągnąć lepsze wyniki uczonych cech. MLP (MultiLayer Perceptron) odpowiedzialny jest za klasyfikowanie obrazu w kontekście globalnym na podstawie otrzymanych wag. Zwracaną wartością jest przypisane prawdopodobieństwo z jakim każda klasa została przyporządkowana.

Dekoder - definiowany jest jako kaskadowy upsampler (CUP) składa się z operacji 2.10

$$\text{upsampling } 2 \times 2 \rightarrow \text{feature concentration} \rightarrow \text{conv } 3 \times 3 (\text{ReLU}) \quad (2.10)$$

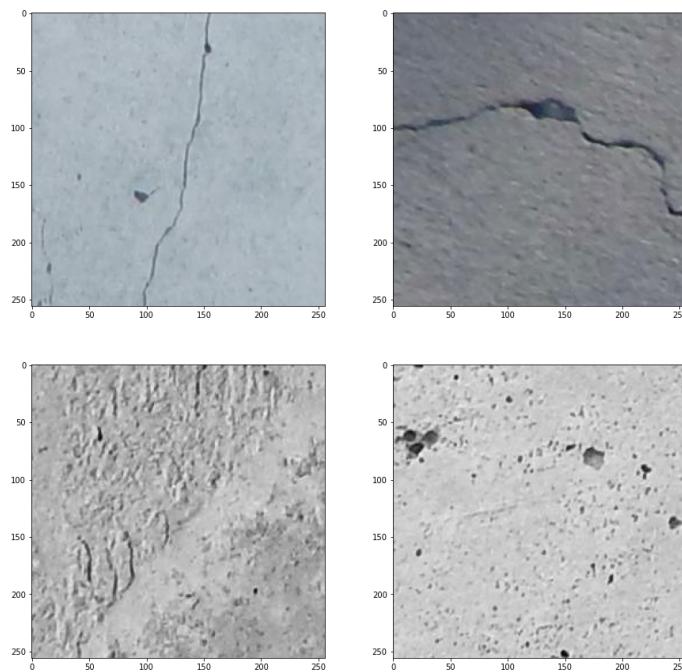
Encoder oraz decoder połączony jest za pomocą skip connections, jak w tradycyjnej sieci Unet, które używając opisanej wcześniej struktury pozwalają na dostarczenie globalnej informacji przestrzennej.

Transformer zachowuje się lepiej niż standardowy CNN ponieważ posiada globalny kontekst obrazu, jednak aby wyniki były zadowalające wymagane jest znacznie więcej danych wejściowych niż w przypadku CNN.

# Definiowanie modeli i implementacja algorytmu mierzenia pęknięć

## 3.1. Zbiór danych

SDNET2018 [26] jest zbiorem danych służącym do klasyfikacji obrazu ze względu na występujące/niewystępujące pęknięcie w elementach betonowych, zawiera pęknięcia w zakresie 0.06mm-25mm. Powstał z 230 zdjęć wysokiej rozdzielczości wykonanych aparatem 16 MP Nikon następnie podzielonych na 56092 podobrazów o rozmiarze 256x256px (3.1). Sporządzony przez pracowników Uniwersytetu Stanowego Utah oraz udostępniony do użytku publicznego. Inne zbiory danych które mogą być wykorzystane w zadaniu klasyfikacji/segmentacji obrazu: [40], [20], [24].



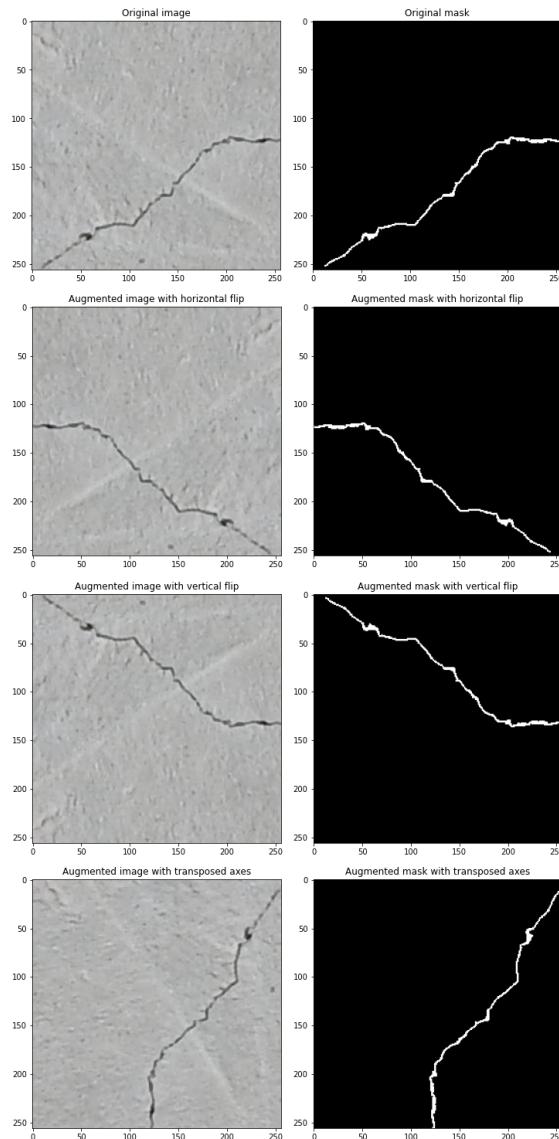
Rysunek 3.1: Przykładowe zdjęcia ze zbioru SDNET2018[26]. Źródło: [26]

Z powyższego zbioru (SDNET2018) wybrano 800 zdjęć które następnie ręcznie opisano, nadając każdemu z obrazów odpowiadającą maskę w której kolorem czarnym zaznaczono tło oraz kolorem białym pęknięcie. Adnotacje wykonano przy użyciu aplikacji *segments-ai* [12]. W celu otrzymania większej ilości danych wejściowych dla

każdego obrazu wykonano obrócenie horyzontalne, wertykalne oraz transpozycje osi X/Y, te same operacje zostały przeprowadzone dla korespondujących masek. W wyniku rozszerzenia danych otrzymano 3200 obrazów.

Przygotowany zbiór danych stworzony do celów realizacji pracy jest dostępny pod adresem:

<https://www.kaggle.com/datasets/jakubniemiec/concrete-crack-images>



Rysunek 3.2: Przykładowe zdjęcia otrzymywane w procesie rozszerzenia obrazu. Źródło: opracowanie własne

Zbiór danych wejściowych podzielono na podzbiór treningowy oraz walidacyjny w proporcji 80/20, w wyniku otrzymano 2560 obrazów z podzbiorze treningowym oraz 640 w podzbiorze walidacyjnym.

## 3.2. Funkcja kosztu

Funkcja kosztu ma na celu optymalizację uczonego modelu, najczęściej dąży się do jej zminimalizowania, w ogólności oznacza jak daleko przewidywane wyniki są od wyników pożądanych. W procesie uczenia użyto funkcję dice los [54] opisywaną równaniem 3.1 która pochodzi od metryki dice coefficient 3.3. Znajduje zastosowanie głównie w niezrównoważonych zbiorach danych gdzie klasy nie są równe (np. w segmentacji binarnej gdzie przewidywana klasa stanowi kilka procent całości obrazu, a pozostały obszar stanowi tło).

$$Dice\ loss = 1 - dice\ coefficient = 1 - \frac{2|A \cap B|}{|A| + |B|} \quad (3.1)$$

## 3.3. Metryki oceny zachowania modelu

Ewaluacja jest procesem mającym na celu sprawdzenie jakości uzyskanego rozwiązania. W procesie uczenia monitorowano dwa parametry:

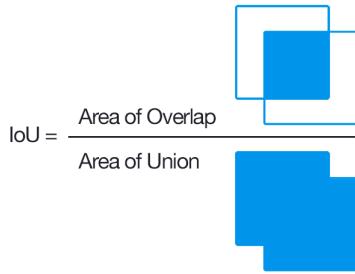
**Intersection Over Union (Jaccard index)** jest jedną z najczęściej stosowanych metryk w procesie segmentacji obrazu, pozwala ocenić jak dokładna jest przewidziana maska w odniesieniu do maski wejściowej. IoU opisuje równanie 3.2 - jest to obszar wspólny maski wejściowej oraz przewidzianej odniesiony do sumy tych obszarów. Wartości zwracane zawierają się w przedziale 0-1, gdzie wartości zbliżone do 0 oznaczają brak powiązania, natomiast wartości bliskie 1 oznaczają silne powiązanie, dobre zachowanie modelu. W przypadku segmentacji binarnej obliczany jest stosunek IoU dla każdej z klas (tło oraz pęknięcie) a następnie zwracana jest średnia wartość.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (3.2)$$

gdzie:

A - maska

B - przewidziana maska



Rysunek 3.3: Graficzne przedstawienie wzoru 3.2. Źródło: [49]

**Dice coefficient (Sørensen–Dice index, F1-score)** jest miarą służącą do określania podobieństwa dwóch zbiorów, opisywana wzorem 3.3. Tak samo jak w przypadku IoU zwracane wartości zawierają się w przedziale 0-1, gdzie wartości zbliżone do 0 oznaczają brak podobieństwa, a wartości zbliżone do 1 oznaczają bardzo duże podobieństwo.

$$DSC = \frac{2|A \cap B|}{|A| + |B|} \quad (3.3)$$

Recall określa ile poprawnych przewidzianych wyników zostało zwróconych. Można opisać jako iloraz poprawnie przypisanych pikseli do pęknienia i ilości pikseli przypisanych do pęknienia w pierwotnej masce [36].

$$recall = \frac{TP}{TP + FN} \quad (3.4)$$

Precision określa ile z pozytywnie przewidzianych wartości jest poprawnych. Można opisać jako iloraz poprawnie przypisanych pikseli do pęknienia i ilości wszystkich pikseli przypisanych do pęknienia w obrazie [36].

$$precision = \frac{TP}{TP + FP} \quad (3.5)$$

Wzór 3.3 można zapisać za pomocą wartości logicznych, stosowane oznaczenia pokazano na rysunku 3.5.

$$DSC = \frac{2 \times precision \times recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \quad (3.6)$$

gdzie:

TP (true positives) - piksele poprawnie przypisane do tej samej klasy w masce oraz w

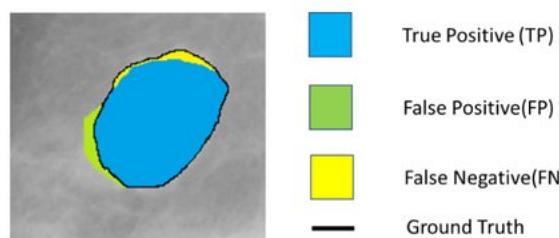
przewidzianym obrazie

FP (false positives) - piksele niepoprawnie przypisane do danej klasy w przewidzianym obrazie

FN (false negatives) - piksele należące do maski ale niepoprawnie przypisane w przewidzianym obrazie

		Actual (True) Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

Rysunek 3.4: Przedstawienie wartości logicznych na tablicy pomyłek. Źródło: [34]



Rysunek 3.5: Graficzne przedstawienie wartości logicznych. Źródło: [29]

### 3.4. Zestawienie wyników segmentacji obrazu

Wykorzystane biblioteki przedstawiono w tabeli 3.2, specyfikacje środowiska opisano w tabeli 3.1.

Tabela 3.1: Specyfikacja środowiska Google Colab Pro[3]

GPU	Tesla P100-PCIE-16GB
CPU	Intel® Xeon® E5-2630
RAM	27.3GB

Tabela 3.2: Wykorzystane biblioteki

Biblioteka	wersja
OpenCV [9]	4.1.2
Tensorflow [13]	2.8.0
Keras [5]	2.8.0
NumPy [8]	1.21.6
Matplotlib [6]	3.2.2
Scikit-image [10]	0.18.3
Scikit-learn [11]	1.0.2

Każdy z opisanych modeli (Unet 2.3, Attention Unet 2.4, Transunet2.5) wytrenowano używając tego samego zbioru danych opisanego w sekcji 3.1. Uczenie wykonano bez wcześniej trenowanych wag - każdy proces uczenia zaczynano od losowo dobranych wag początkowych. Warunkiem zakończenia uczenia było skończenie założonej ilości epok lub brak poprawy zachowania modelu oceniane na podstawie danych walidacyjnych [2], każdy model który poprawiał swoje zachowanie zapisywano w celu późniejszego wczytania obliczonych wag [7]. Jako algorytmu optymalizacji użyto funkcje Adam [21].

Tabela 3.3: Informacje o trenowanych modelach

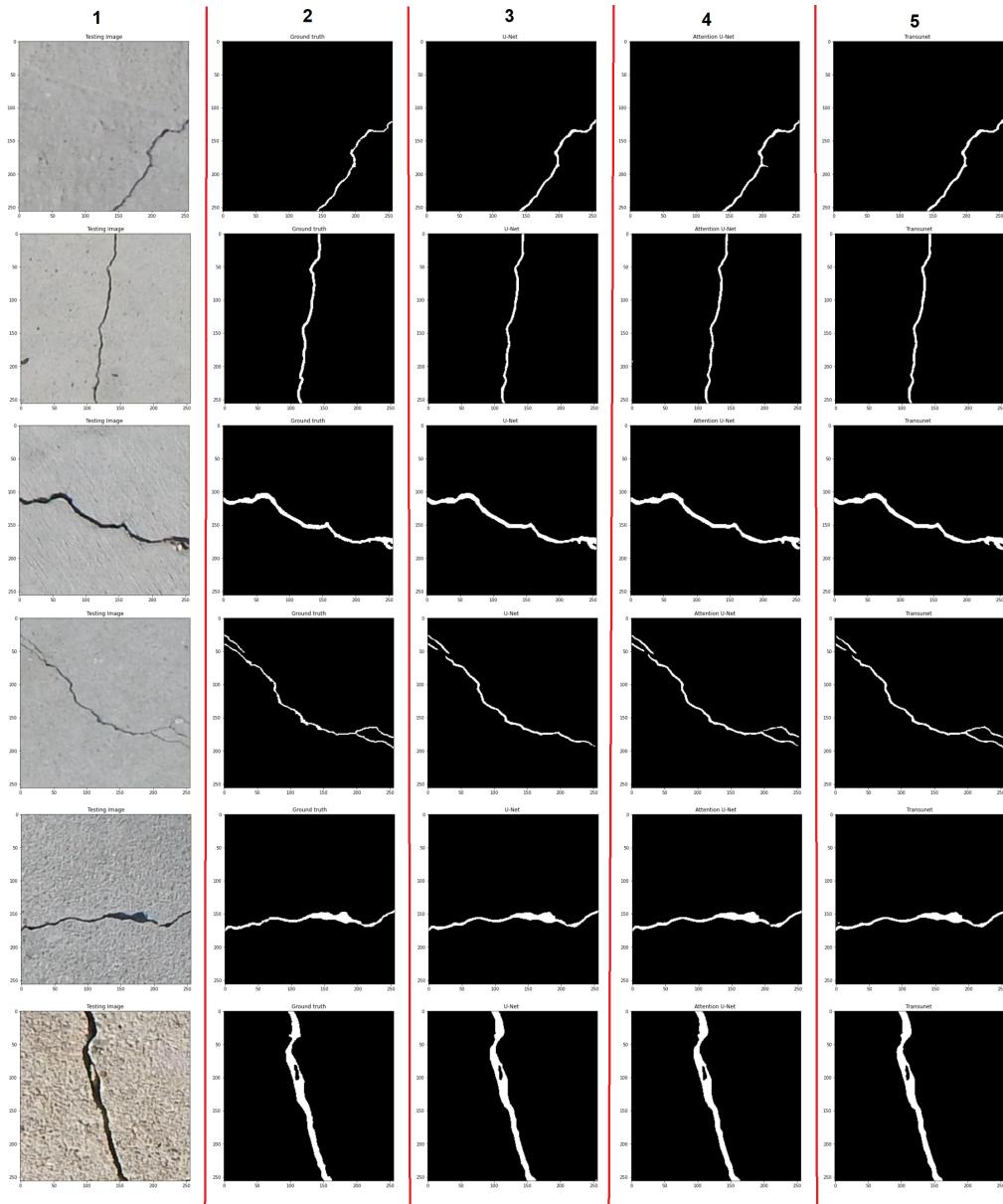
Model	liczba parametrów	czas trenowania
Unet	31.5M	1h31m
Attention-Unet	37.3M	1h58m
TransUnet	72.8M	2h42m

Wykonano ewaluacje modeli na zbiorze testowym, wyniki przedstawiono w tabeli 3.4. Zbiór testowy nie zawierał obrazów nie przedstawiających pęknięcia.

Tabela 3.4: Ewaluacja analizowanych modeli na zbiorze testowym

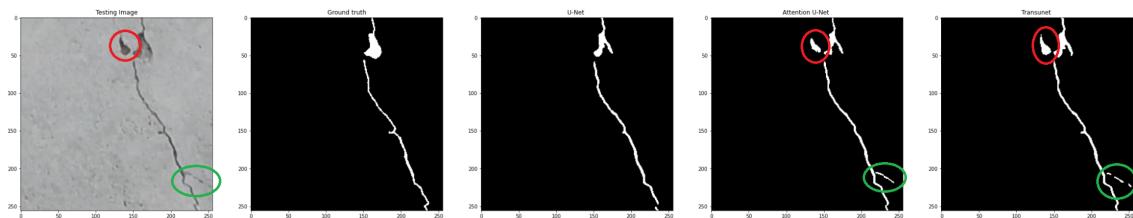
Model	mIoU[%]	F1[%]	Precision[%]	Recall[%]
Unet	87.35	85.33	84.45	87.16
Attention-Unet	87.59	85.82	<b>84.89</b>	87.71
<b>TransUnet</b>	<b>88.21</b>	<b>86.75</b>	83.40	<b>90.92</b>

Za pomocą każdego z modeli uzyskano zadowalające wyniki. Różnice pomiędzy modelami dla tego zadania są nieznaczne. Najlepiej jednak prezentuje się model Transunet dla którego uzyskano najlepsze wyniki, równocześnie jest on najbardziej kosztowny w procesie trenowania pod względem zasobów oraz potrzebnego czasu.



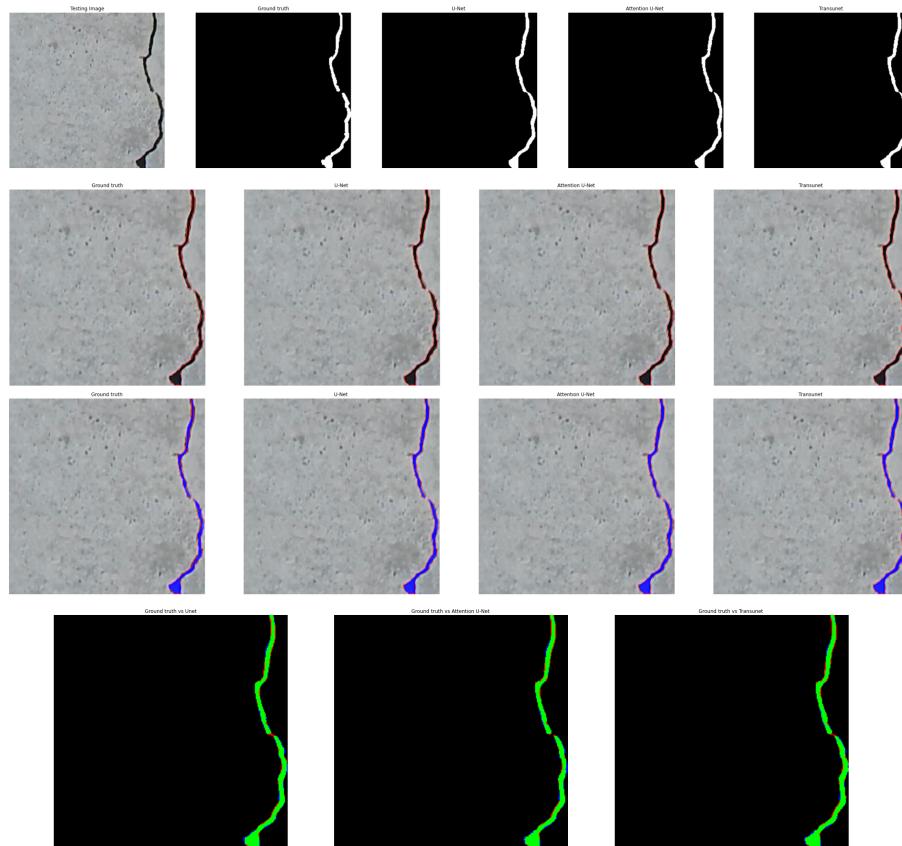
Rysunek 3.6: Przykładowe wyniki segmentacji dla wszystkich modeli  
 kolumna 1 - obraz wejściowy  
 kolumna 2 - Pierwotna maska  
 kolumna 3 - Unet - przewidziana maska  
 kolumna 4 - Attention Unet - przewidziana maska  
 kolumna 5 - Transunet - przewidziana maska.  
 Źródło: opracowanie własne

Istnieją przypadki w których model zachowuje się lepiej od pierwotnej maski. Na rysunku 3.7 można zauważyc że elementy zaznaczone kolorem czerwonym oraz zielonym nie występują w masce wejściowej (obraz drugi od lewej), jednak wynik modelu Attention Unet oraz Transunet zaklasyfikowały te obszary jako pęknięte. Podczas ewaluacji modelu te obszary zostały uznane jako błędnie przewidziane, jednak analizując manualnie obrazy można stwierdzić że te obszary są prawidłowe i mogą być uznane za pęknięte.



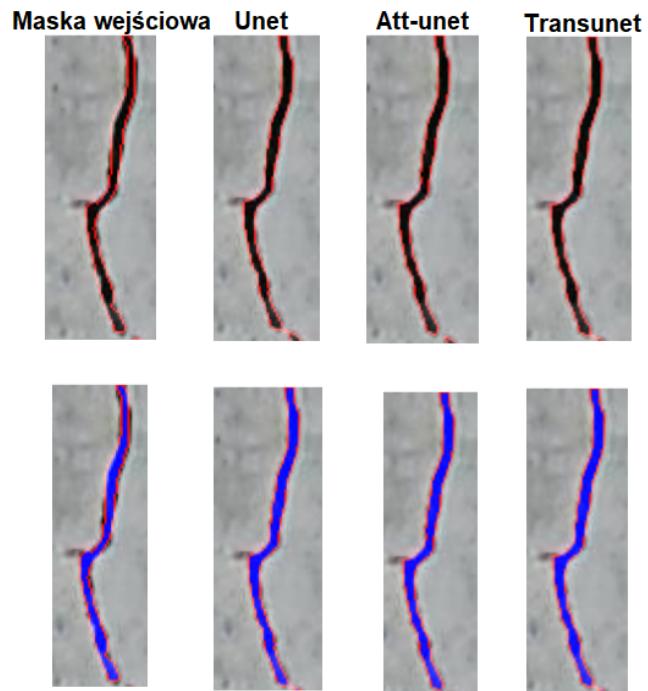
Rysunek 3.7: Przykładowe wyniki segmentacji dla wszystkich modeli. Źródło: opracowanie własne

### Analiza konturów i wizualizacja klasyfikacji pikseli 3.8



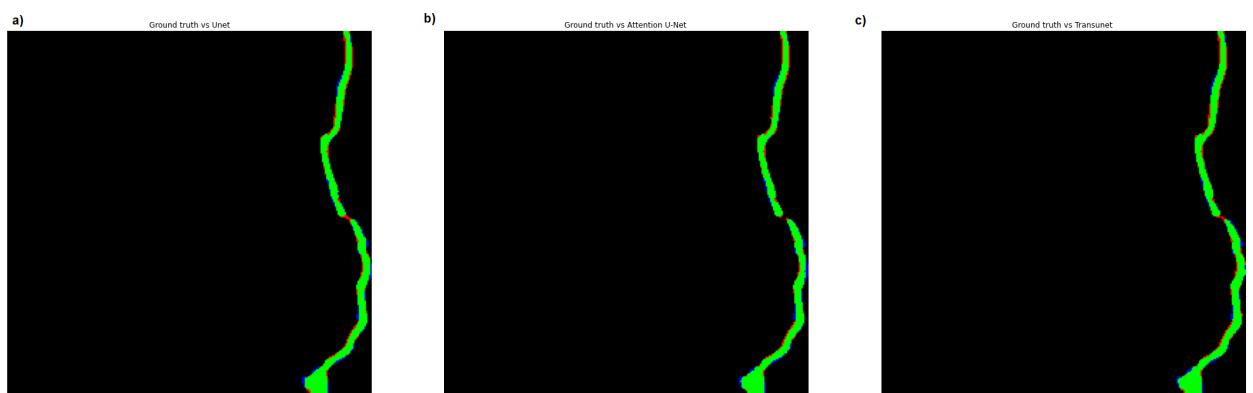
Rysunek 3.8: Analiza konturów i wizualizacja klasyfikacji pikseli. Źródło: opracowanie własne

Analizując kontur oraz wypełnienie można zobaczyć niedokładność maski, co wynika z ręcznego opisywania obrazów. Wykryty przez każdy z modeli kontur we wskażanym obrazie jest dokładniejszy niż w masce wejściowej.



Rysunek 3.9: Kontury i wypełnienie. Źródło: opracowanie własne

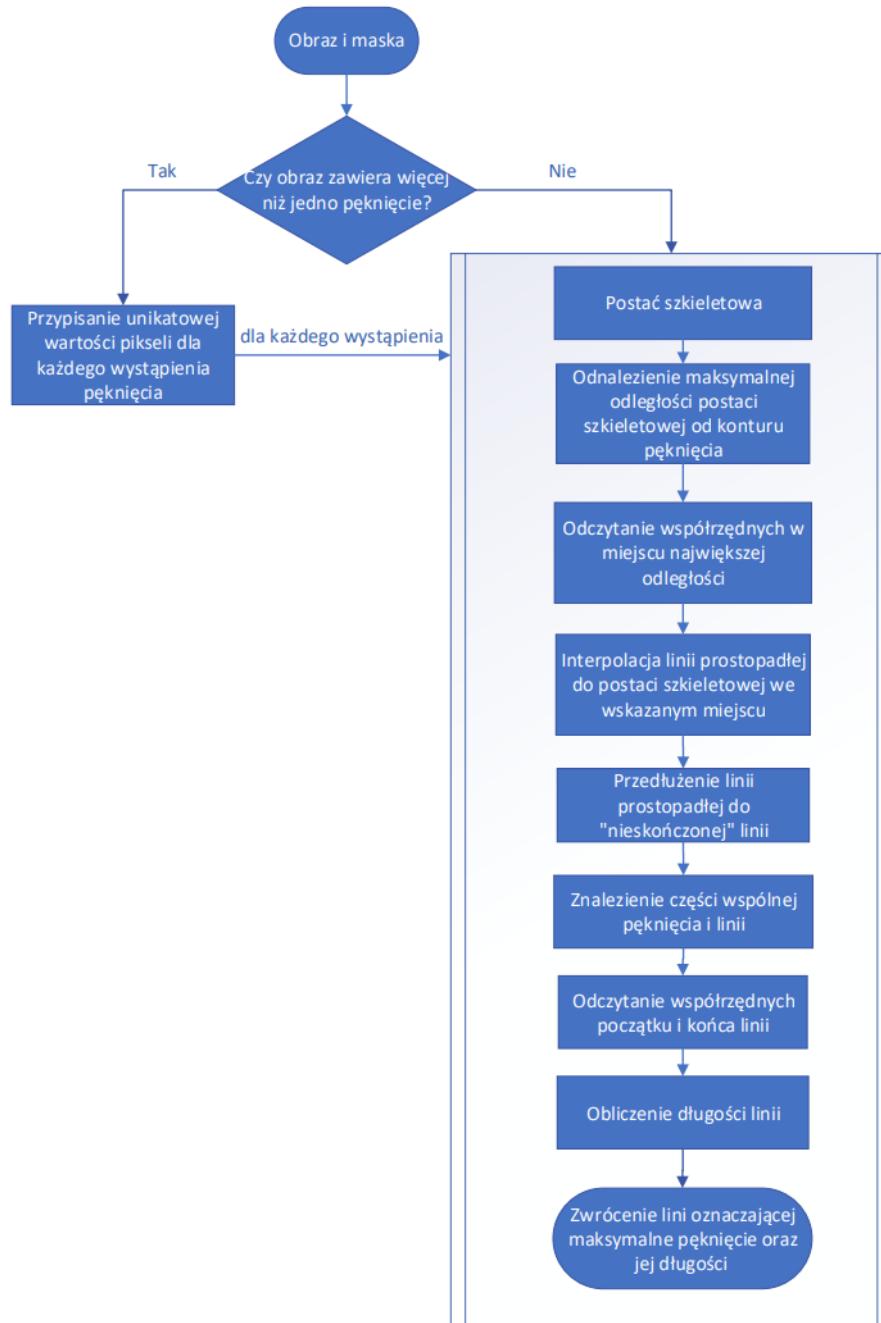
Kolorem zielonym oznaczono piksele wspólne dla maski wejściowej i rezultatu segmentacji dla każdego z modeli. Kolorem czerwonym oznaczono piksele przypisane pęknieniu przez model, które nie występują w masce wejściowej. Kolorem niebieskim - piksele należące do maski pierwotnej, których nie ma w przewidzianej masce.



Rysunek 3.10: Klasyfikacja pikseli - porównanie modelu z maską wejściową. Źródło: opracowanie własne

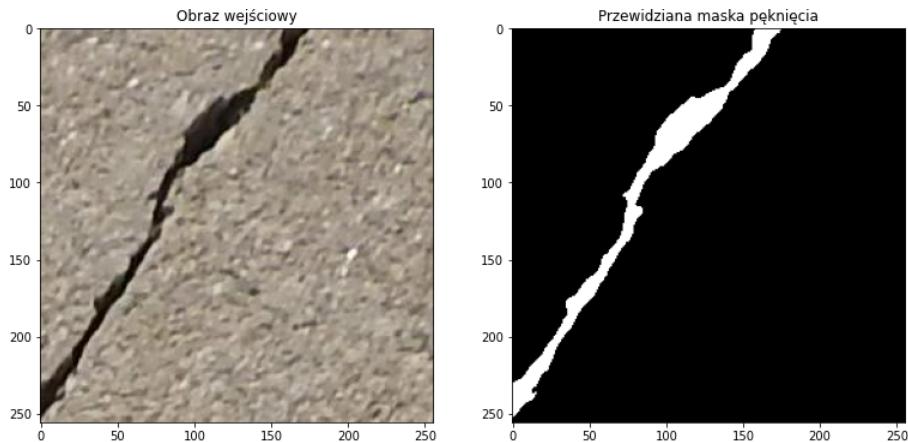
### 3.5. Algorytm mierzenia pęknięć

Algorytm mierzenia pęknięcia bazuje na obrazie maski otrzymanej w procesie segmentacji. Wykorzystując obraz wyjściowy za pomocą dodatkowej obróbki (image postprocessing) dokonuje się pomiaru, schemat blokowy przedstawiający poszczególne kroki postępowania przedstawiono na rysunku 3.11.



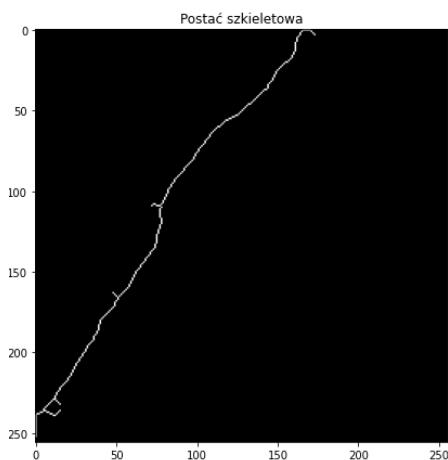
Rysunek 3.11: Schemat blokowy przedstawiający algorytm mierzenia pęknięcia. Źródło: opracowanie własne

### 1. Obraz wejściowy i przewidziana maska pęknięcia



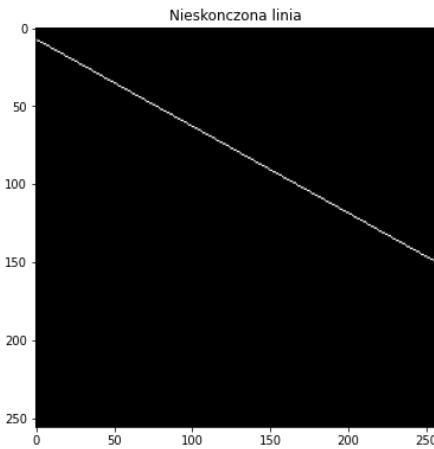
Rysunek 3.12: Obraz wejściowy do modelu i przewidziana maska. Źródło: opracowanie własne

### 2. Przekształcenie maski pęknięcia do postaci szkieletowej



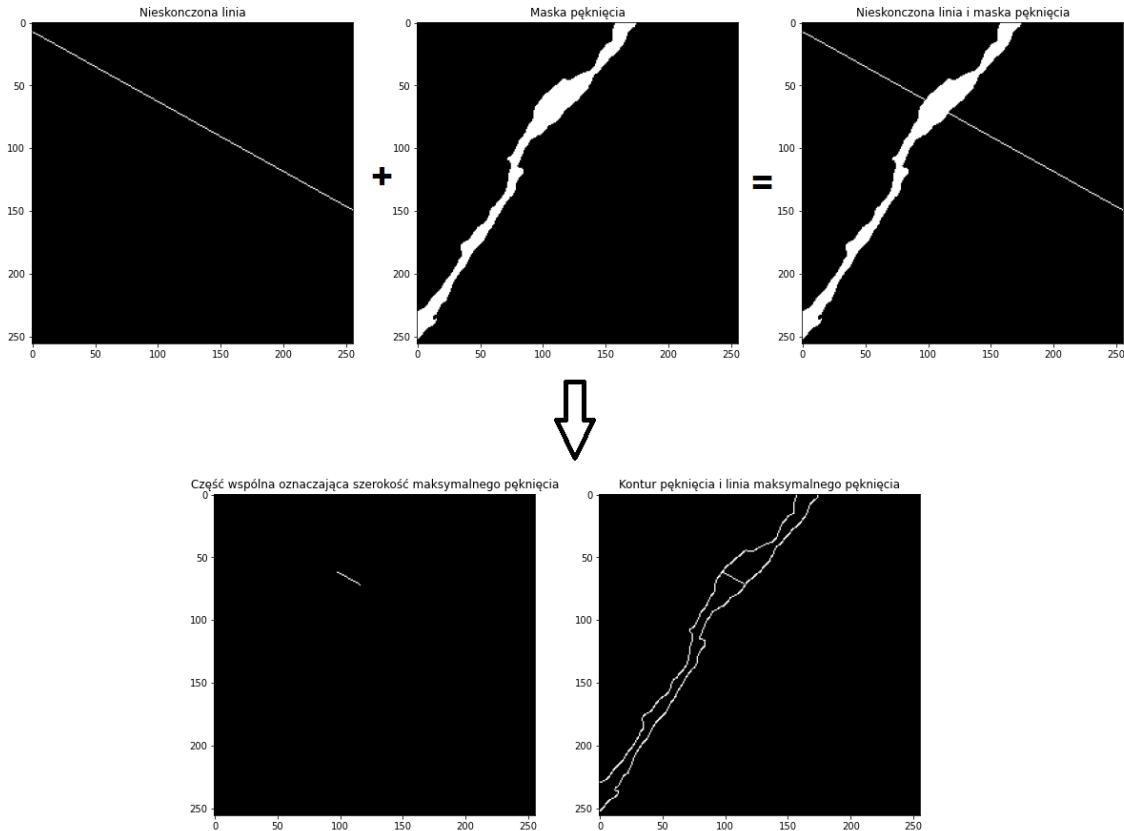
Rysunek 3.13: Postać szkieletowa. Źródło: opracowanie własne

3. Odszukanie maksymalnej odległości między postacią szkieletową a konturem pęknięcia
4. Interpolacja linii prostopadłej do szkieletu pęknięcia w miejscu maksymalnej odległości od konturu
5. Przedłużenie linii - utworzenie “nieskończonej” linii przechodzącej przez cały obraz



Rysunek 3.14: 'Nieskończona' linia pomocnicza. Źródło: opracowanie własne

## 6. Znalezienie wspólnego obszaru między utworzoną linią i pęknięciem



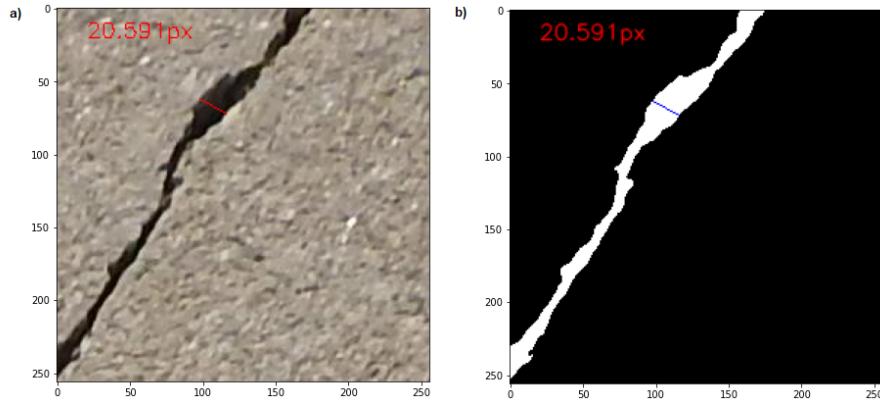
Rysunek 3.15: Proces wyznaczania linii przedstawiającej szerokość pęknięcia. Źródło: opracowanie własne

7. Odczytanie współrzędnych początku  $A = (x_1; y_1)$  oraz końca  $B = (x_2; y_2)$  linii reprezentującej rozwarcie

8. Obliczenie długości linii według wzoru 3.7

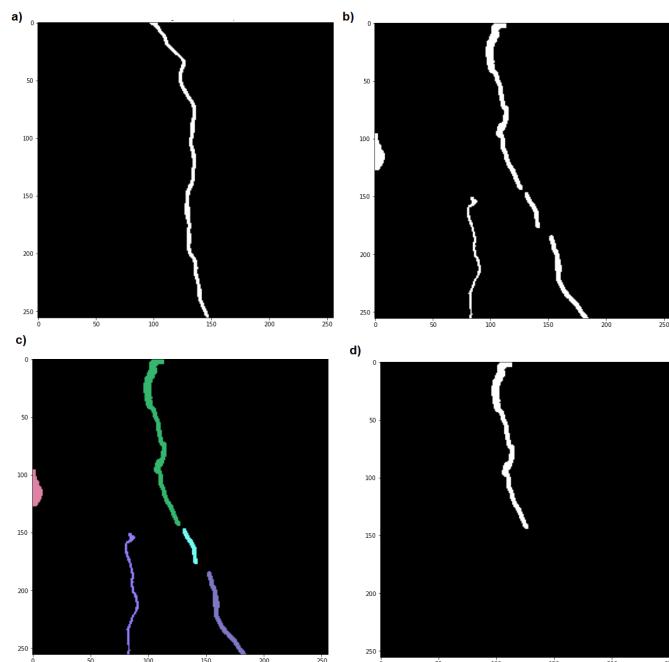
$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (3.7)$$

### 9. Przedstawienie wyniku na masce oraz obrazie wejściowym



Rysunek 3.16: Przedstawienie wyników pomiaru na obrazie wejściowym i masce. Źródło: opracowanie własne

Jeśli obraz posiada więcej niż jedno pęknięcie, każdemu wystąpieniu przypisana jest inna wartość 3.17, następnie filtrując kolejne elementy przeprowadzany jest powyższy algorytm.



Rysunek 3.17: a) obraz z pojedynczym pęknięciem b) obraz z wieloma pęknięciami c) Oznaczenia każdego wystąpienia pęknięcia inną wartością d) Wyselekcjonowane wystąpienie pęknięcia. Źródło: opracowanie własne

Dla każdego z modeli z wykorzystaniem powyższego algorytmu wykonano pomiar pęknięć przeprowadzony na zbiorze testowym. Średni względny błąd procentowy 3.8 między modelem a oryginalnymi maskami przedstawiono w tabeli 3.5.

$$MRPE = \frac{1}{n} \sum_{i=1}^n \frac{\hat{y} - y}{y} \times 100\% \quad (3.8)$$

gdzie:

$\hat{y}$  - wartość przewidziana

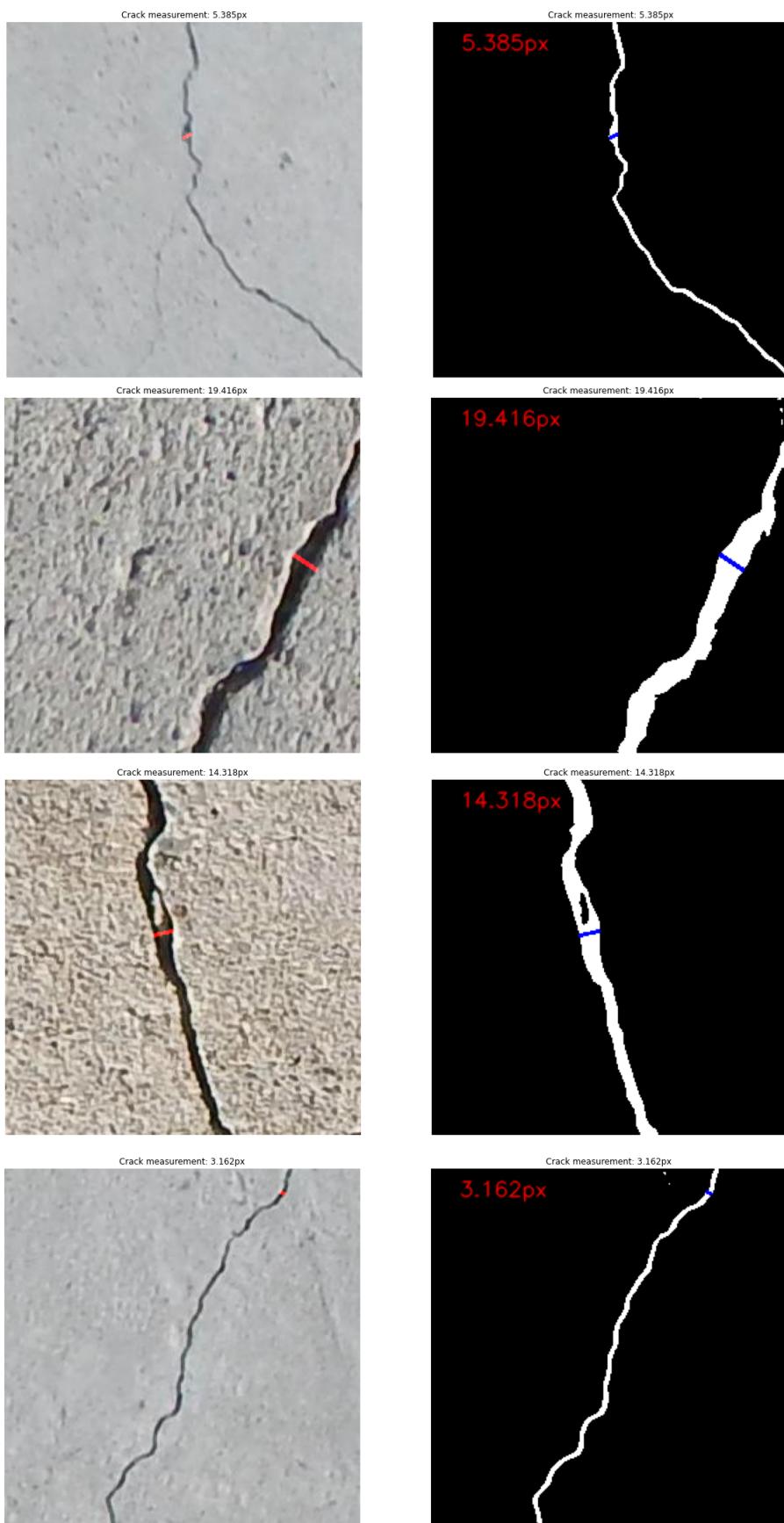
$y$  - wartość z maski oryginalnej

Tabela 3.5: Średni względny błąd procentowy dla poszczególnego modelu

Model	MRPE[%]
Unet	8.43
Attention-Unet	8.40
<b>TransUnet</b>	<b>8.32</b>

Tabela 3.6: Sumaryczne zestawienie wyników

Model	mIoU[%]	F1[%]	Precision[%]	Recall[%]	MRPE[%]
Unet	87.35	85.33	84.45	87.16	8.43
Attention-Unet	87.59	85.82	<b>84.89</b>	87.71	8.40
<b>TransUnet</b>	<b>88.21</b>	<b>86.75</b>	83.4	<b>90.92</b>	<b>8.32</b>

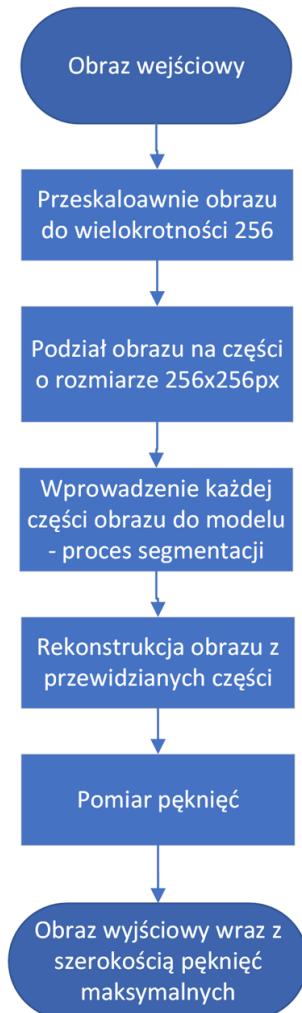


Rysunek 3.18: Przykładowe wyniki pomiaru pęknięcia

# Analiza obrazu laboratoryjnego

Wykonano pomiar pęknięć na zdjęciach wysokiej rozdzielczości (4000x6000px) pochodzących z pracy [56] w której pokazano wizyjny system pomiarowy do analizy pól deformacji elementów konstrukcyjnych wykonanych z różnych materiałów, który może być stosowany jako rozszerzenie tradycyjnych metod pomiarowych.

Obraz wejściowy został przeskalowany do wartości najbliższej wielokrotności 256 (ze względu na rozmiar wejściowy przyjmowany w modelu) - 3840x5888px następnie podzielony na nienachodzące na siebie części o rozmiarze 256x256px. Schemat postępowania przedstawia schemat 4.2.

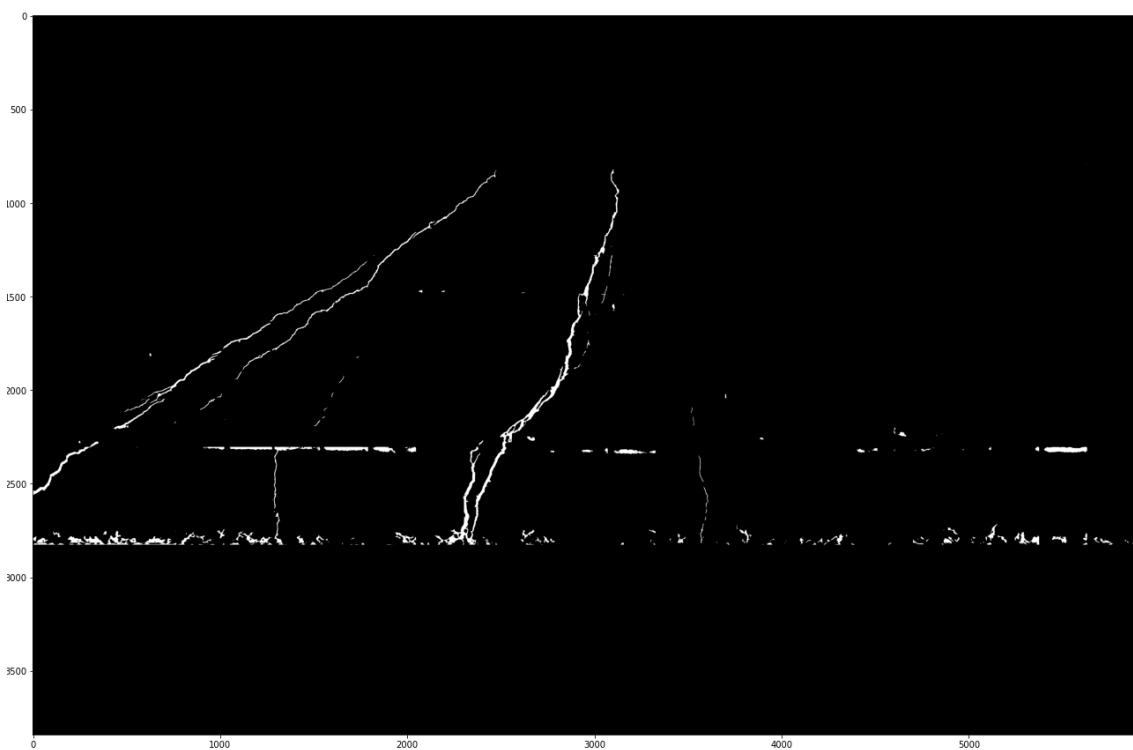


Rysunek 4.1: Schemat blokowy pomiaru pęknięć obrazu wysokiej rozdzielczości . Źródło: opracowanie własne



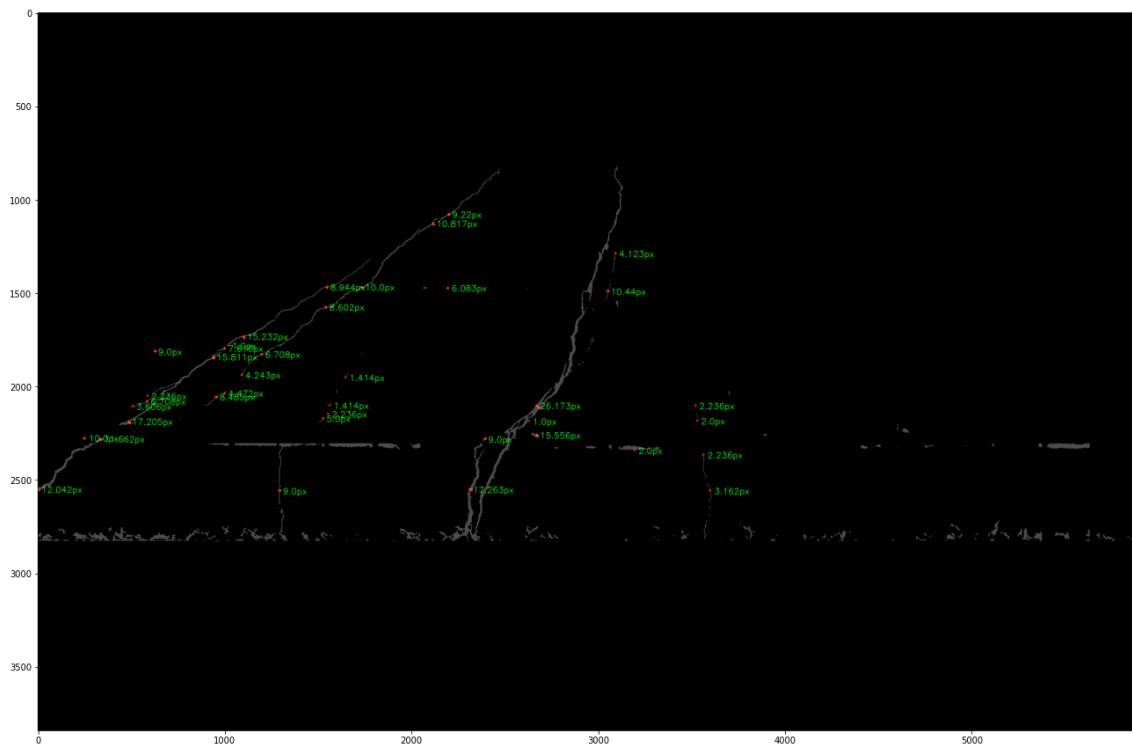
Rysunek 4.2: Obraz z badania laboratoryjnego . Źródło: opracowanie własne

Przy pomocy wytrenowanego modelu na zbiorze SDNET, wykonano segmentacje powyższego obrazu. Obraz 4.3 przedstawia wynik rekonstrukcji obrazu.

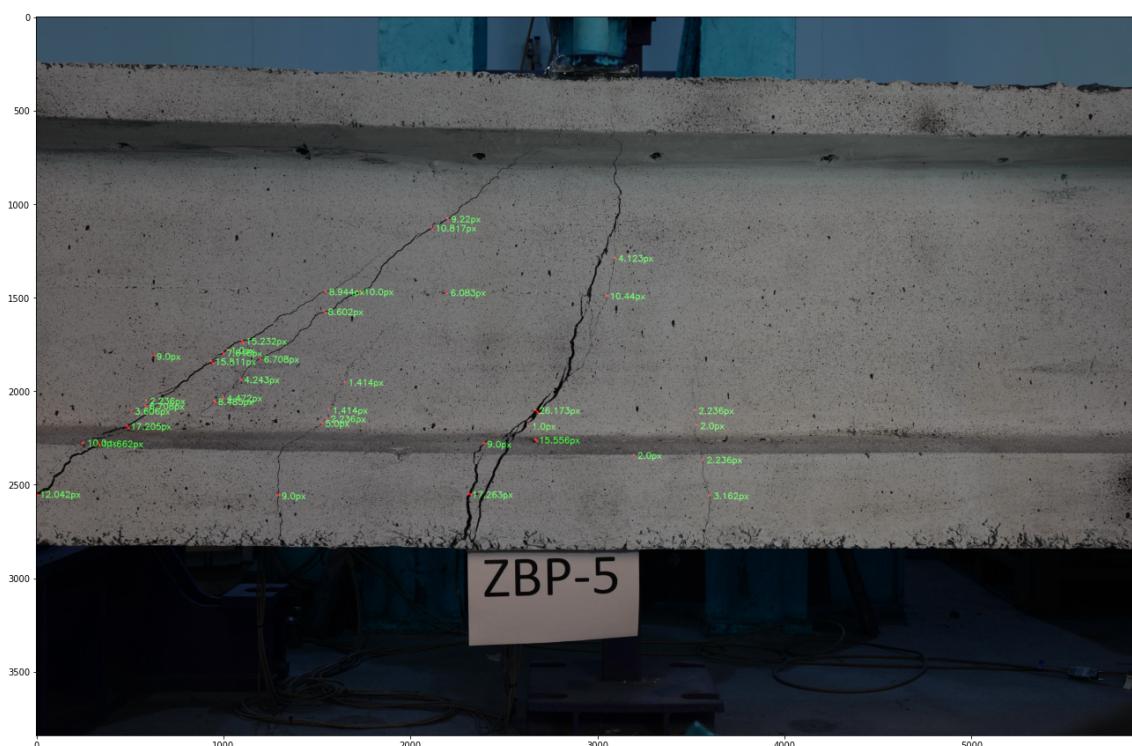


Rysunek 4.3: Wynik segmentacji po usunięciu szumów. Źródło: opracowanie własne

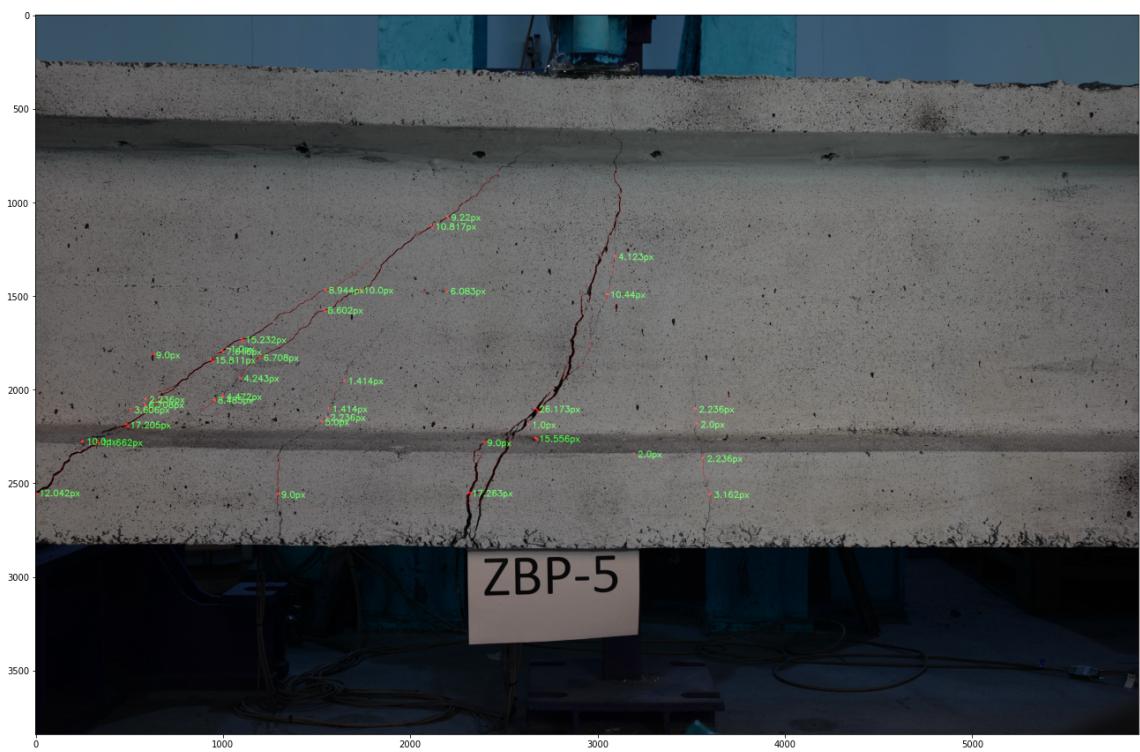
Dla przewidzianej maski z wykonano pomiar pęknięć.



Rysunek 4.4: Pomiar szerokości pęknięć. Źródło: opracowanie własne



Rysunek 4.5: Przedstawienie szerokości pęknięć na obrazie wejściowym. Źródło: opracowanie własne



Rysunek 4.6: Przedstawienie konturu pęknięcia i pomiaru na obrazie wejściowym. Źródło: opracowanie własne

# Podsumowanie i wnioski

W powyższej pracy przedstawiono sposób oceny stanu technicznego z wykorzystaniem głębokich sieci neuronowych, wykonano analizę i porównanie architektur modeli wykorzystywanych w segmentacji obrazu, stworzono zbiór dany który może być wykorzystany w zadaniach o powiązanej tematyce oraz zaproponowano sposób pomiaru pęknięć w elementach betonowych. Stosowanie opisanego podejścia pozwala zautomatyzować i przyspieszyć proces inspekcji obiektów budowlanych. Znając dokładne położenie urządzenia wykonującego zdjęcie możliwe jest określenie rzeczywistego wymiaru pęknięcia.

Tradycyjne sieci Unet pozwalają na uzyskanie satysfakcjonującego rozwiązania, jednak najnowsze podejście z użyciem architektury transformers umożliwia polepszenie rezultatów co potwierdzają wykonane badania - dla tego modelu uzyskano najlepsze wyniki.

Zaproponowany algorytm bazujący na operacjach przetwarzania obrazu jest w stanie z dużą precyzją określić szerokość pęknięć.

Aby usprawnić zachowanie trenowanych modeli w określonych warunkach można wykonać transfer przypisanych wag oraz dostrojenie jedynie kilku parametrów w celu otrzymania modelu sprecyzowanego do danego zadania. Takie podejście określone jest jako transfer learning gdzie wytrenowany wcześniej model jest ponownie wykorzystywany w podobnym zadaniu - pozwala to zaoszczędzić czas ponieważ model został już wcześniej wytrenowany i jest w stanie wyodrębniać pożądane cechy. W związku z powyższym w procesie dostrajania nie jest konieczne wprowadzenie dużego zestawu nowych danych.

# Spis rysunków

1.1	Zmiana liczby obiektów mostowych w czasie. Źródło: [30] . . . . .	4
1.2	Schemat metody opartej na przetwarzaniu obrazu. Źródło: Opracowanie własne . . . . .	6
1.3	Schemat metody opartej na uczeniu maszynowym. Źródło: Opracowanie własne . . . . .	6
2.1	Dane wejściowe-wyjściowe w: klasyfikacji obrazu (a-b), segmentacji semantycznej (c-d). Źródło: opracowanie własne . . . . .	9
2.2	Schemat propagacji wstecznej. Źródło: [43] . . . . .	10
2.3	Schemat uczenia nadzorowanego. Źródło: [46] . . . . .	11
2.4	Operacja konwolucyjna. Źródło: [47] . . . . .	12
2.5	a) przedstawienie graficzne Max Pooling b) przedstawienie graficzne Average Pooling. Źródło: [59] . . . . .	13
2.6	Odwrocona konwolucja. Źródło: [14] . . . . .	14
2.7	Aktywacja. Źródło: [19] . . . . .	14
2.8	ReLU. Źródło: [55] . . . . .	15
2.9	Sigmoid. Źródło: [1] . . . . .	16
2.10	GELU. Źródło: [53] . . . . .	16
2.11	Architektura sieci U-net. Źródło: [48] . . . . .	17
2.12	Zależności w strukturze Unet, Źródło: opracowanie własne . . . . .	19
2.13	Architektura sieci Attention U-Net. Źródło: [45] . . . . .	20
2.14	Attention gate schemat. Źródło: [45] . . . . .	20
2.15	Attention gate schemat blokowy. Źródło: opracowanie własne . . . . .	20
2.16	Architektura sieci TransUnet. Źródło: [23] . . . . .	22

2.17 Architektura ViT. Źródło: [27] . . . . .	22
2.18 Wizualizacja podziału obrazu na części o rozmiarze 16x16px. Źródło: opracowanie własne . . . . .	23
3.1 Przykładowe zdjęcia ze zbioru SDNET2018[26]. Źródło: [26] . . . . .	25
3.2 Przykładowe zdjęcia otrzymane w procesie rozszerzenia obrazu. Źródło: opracowanie własne . . . . .	26
3.3 Graficzne przedstawienie wzoru 3.2. Źródło: [49] . . . . .	28
3.4 Przedstawienie wartości logicznych na tablicy pomyłek. Źródło: [34] . . .	29
3.5 Graficzne przedstawienie wartości logicznych. Źródło: [29] . . . . .	29
3.6 Przykładowe wyniki segmentacji dla wszystkich modeli  kolumna 1 - obraz wejściowy  kolumna 2 - Pierwotna maska  kolumna 3 - Unet - przewidziana maska  kolumna 4 - Attention Unet - przewidziana maska  kolumna 5 - Transunet - przewidziana maska.  Źródło: opracowanie własne . . . . .	31
3.7 Przykładowe wyniki segmentacji dla wszystkich modeli. Źródło: opraco- wanie własne . . . . .	32
3.8 Analiza konturów i wizualizacja klasyfikacji pikseli. Źródło: opracowanie własne . . . . .	32
3.9 Kontury i wypełnienie. Źródło: opracowanie własne . . . . .	33
3.10 Klasyfikacja pikseli - porównanie modelu z maską wejściową. Źródło: opracowanie własne . . . . .	33
3.11 Schemat blokowy przedstawiający algorytm mierzenia pęknienia. Źródło: opracowanie własne . . . . .	34
3.12 Obraz wejściowy do modelu i przewidziana maska. Źródło: opracowanie własne . . . . .	35
3.13 Postać szkieletowa. Źródło: opracowanie własne . . . . .	35
3.14 'Nieskończona' linia pomocnicza. Źródło: opracowanie własne . . . . .	36

3.15 Proces wyznaczania linii przedstawiającej szerokość pęknięcia. Źródło: opracowanie własne . . . . .	36
3.16 Przedstawienie wyników pomiaru na obrazie wejściowym i masce. Źródło: opracowanie własne . . . . .	37
3.17 a) obraz z pojedynczym pęknięciem b) obraz z wieloma pęknięciami c) Oznaczenia każdego wystąpienia pęknięcia inną wartością d) Wyselekcjonowane wystąpienie pęknięcia. Źródło: opracowanie własne . . . . .	37
3.18 Przykładowe wyniki pomiaru pęknięcia . . . . .	39
4.1 Schemat blokowy pomiaru pęknięć obrazu wysokiej rozdzielczości . Źródło: opracowanie własne . . . . .	40
4.2 Obraz z badania laboratoryjnego . Źródło: opracowanie własne . . . . .	41
4.3 Wynik segmentacji po usunięciu szumów. Źródło: opracowanie własne . .	41
4.4 Pomiar szerokości pęknięć. Źródło: opracowanie własne . . . . .	42
4.5 Przedstawienie szerokości pęknięć na obrazie wejściowym. Źródło: opracowanie własne . . . . .	42
4.6 Przedstawienie konturu pęknięcia i pomiaru na obrazie wejściowym. Źródło: opracowanie własne . . . . .	43

## Spis tabel

1.1 Skala i kryteria ocen elementów. Źródło: [4] . . . . .	5
1.2 Przykładowa karta oceny przyczółka. Źródło: [17] . . . . .	5
3.1 Specyfikacja środowiska Google Colab Pro[3] . . . . .	30
3.2 Wykorzystane biblioteki . . . . .	30
3.3 Informacje o trenowanych modelach . . . . .	30
3.4 Ewaluacja analizowanych modeli na zbiorze testowym . . . . .	30

3.5 Średni względny błąd procentowy dla poszczególnego modelu . . . . .	38
3.6 Sumaryczne zestawienie wyników . . . . .	38

# Bibliografia

- [1] Advantages of relu vs tanh vs sigmoid activation function in deep neural networks. <https://androidkt.com/advantages-relu-tanh-sigmoid-activation-function-deep-neural-networks/>. dostęp: 24.06.2022.
- [2] Earlystopping. [https://keras.io/api/callbacks/early\\_stopping/](https://keras.io/api/callbacks/early_stopping/). dostęp: 24.06.2022.
- [3] Google colab. <https://colab.research.google.com/>. dostęp: 24.06.2022.
- [4] INSTRUKCJE PRZEPROWADZANIA PRZEGLĄDÓW DROGOWYCH OBIEKTÓW INŻYNIERSKICH. *Załącznik do Zarządzenia nr 35 Generalnego Dyrektora Dróg Krajowych i Autostrad z dnia 28 września 2020 roku.*
- [5] Keras. <https://keras.io/>. dostęp: 24.06.2022.
- [6] Matplotlib. <https://matplotlib.org/>. dostęp: 24.06.2022.
- [7] Modelcheckpoint. [https://keras.io/api/callbacks/model\\_checkpoint/](https://keras.io/api/callbacks/model_checkpoint/). dostęp: 24.06.2022.
- [8] Numpy. <https://numpy.org/doc/stable/index.html>. dostęp: 24.06.2022.
- [9] Opencv. <https://opencv.org/>. dostęp: 24.06.2022.
- [10] Scikit-image. <https://scikit-image.org/>. dostęp: 24.06.2022.
- [11] Scikit-learn. <https://scikit-learn.org/stable/>. dostęp: 24.06.2022.
- [12] Segments-ai. <http://segments.ai/>. dostęp: 24.06.2022.
- [13] Tensorflow. <https://www.tensorflow.org/>. dostęp: 24.06.2022.

- [14] Up-sampling with transposed convolution. <https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0/>. dostęp: 24.06.2022.
- [15] Ustawa z dnia 21 marca 1985 o drogach publicznych (dz.u. 2018 poz. 2068).
- [16] Ustawa z dnia 7 lipca 1994 r. prawo budowlane (dz.u. 2018 poz. 1202).
- [17] ZASADY STOSOWANIA SKALI OCEN PUNKTOWYCH STANU TECHNICZNEGO I PRZYDATNOŚCI DO UŻYTKOWANIA DROGOWYCH OBIEKTÓW INŻYNIERSKICH. Załącznik do Zarządzenia nr 64 Generalnego Dyrektora Dróg Krajowych i Autostrad z dnia 13 listopada 2008 roku.
- [18] M. Azimi, A. D. Eslamlou, and G. Pekcan. Data-driven structural health monitoring and damage detection through deep learning: State-of-the-art review. *Sensors*, 20(10):2778, 2020.
- [19] P. Baheti. Neural networks activation functions. <https://www.v7labs.com/blog/neural-networks-activation-functions/>, 2022. dostęp: 24.06.2022.
- [20] E. Bianchi and M. Hebdon. Concrete Crack Conglomerate Dataset. 10 2021.
- [21] J. Brownlee. Intuition of adam optimizer. <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>, 2021. dostęp: 24.06.2022.
- [22] Y.-J. Cha, W. Choi, and O. Büyüköztürk. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378, 2017.
- [23] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

- [24] Y. H.-J. Cho Hyunwoo. Elci: edge based labeled crack image, 2018.
- [25] A. Cubero-Fernandez, F. Rodriguez-Lozano, R. Villatoro, J. Olivares, J. M. Palomares, et al. Efficient pavement crack detection and classification. *EURASIP Journal on Image and Video Processing*, 2017(1):1–11, 2017.
- [26] S. Dorafshan, R. Thomas, and M. Maguire. SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. *Data in Brief*, 21, 11 2018.
- [27] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [28] Y. Fujita and Y. Hamamoto. A robust automatic crack detection method from noisy concrete surfaces. *Machine Vision and Applications*, 22(2):245–254, 2011.
- [29] S. J. S. Gardezi, A. Elazab, B. Lei, and T. Wang. Breast cancer detection and diagnosis using mammographic data: Systematic review. *Journal of Medical Internet Research*, 21(7):e14464, July 2019.
- [30] Generalna Dyrekcja Dróg Krajowych i Autostrad. Raport o stanie technicznym obiektów mostowych GDDKiA na koniec 2017 roku.
- [31] J. Grus. *Data science od podstaw: Analiza danych w pythonie*. Helion, 2018.
- [32] D. Hendrycks and K. Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.
- [33] A. Horzyk. Sztuczna inteligencja uczenie głębokie i głębokie sieci neuronowe deep learning, 2022. dostęp: 24.06.2022.
- [34] W. G. Hulleman and R. A. Vos. Modeling abiotic niches of crops and wild ancestors using deep learning: A generalized approach. *bioRxiv*, 2019.

- [35] P. K. Kankar, S. C. Sharma, and S. P. Harsha. Vibration-based fault diagnosis of a rotor bearing system using artificial neural network and support vector machine. *International Journal of Modelling, Identification and Control*, 15(3):185–198, 2012.
- [36] T. Kanstrén. A look at precision, recall, and f1-score. <https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec>, 2020. dostęp: 24.06.2022.
- [37] H. Kim, J. Lee, E. Ahn, S. Cho, M. Shin, and S.-H. Sim. Concrete crack identification using a uav incorporating hybrid image processing. *Sensors*, 17(9):2052, 2017.
- [38] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- [39] A. Landstrom and M. J. Thurley. Morphology-based crack detection for steel slabs. *IEEE Journal of selected topics in signal processing*, 6(7):866–875, 2012.
- [40] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li. Deepcrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, 338:139–153, 2019.
- [41] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021.
- [42] M. Mamczur. Jak działają konwolucyjne sieci neuronowe CNN. <https://mirosławmamczur.pl/jak-dzialaja-konwolucyjne-sieci-neuronowe-cnn/>, 2021. dostęp: 24.06.2022.
- [43] S. Mishra. Backpropagation algorithm. <https://smritimishrain.wordpress.com/2021/04/24/backpropagation-algorithm/>, 2021. dostęp: 24.06.2022.

- [44] H. S. Munawar, A. W. Hammad, A. Haddad, C. A. P. Soares, and S. T. Waller. Image-based crack detection methods: A review. *Infrastructures*, 6(8):115, 2021.
- [45] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.
- [46] P. D. Ravish Raj. Supervised, unsupervised, and semi-supervised learning with real-life usecase. <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>. dostęp: 24.06.2022.
- [47] A. H. Reynolds. Convolutional neural networks (CNNs). <https://anhreynolds.com/blogs/cnn.html>, 2019. dostęp: 24.06.2022.
- [48] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [49] A. Rosebrock. Intersection over union (IoU) for object detection. <https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>, 2020. dostęp: 24.06.2022.
- [50] Y. Sari, P. B. Prakoso, and A. R. Baskara. Road crack detection using support vector machine (SVM) and OTSU algorithm. In *2019 6th International Conference on Electric Vehicular Technology (ICEVT)*, pages 349–354. IEEE, 2019.
- [51] B. Shan, S. Zheng, and J. Ou. A stereovision-based crack width detection approach for concrete surface assessment. *KSCE Journal of Civil Engineering*, 20(2):803–812, 2016.
- [52] Q. Shan and R. Dewhurst. Surface-breaking fatigue crack detection using laser ultrasound. *Applied physics letters*, 62(21):2649–2651, 1993.

- [53] A. Shrivastav. Gaussian error linear unit (GELU). <https://iq.opengenus.org/gaussian-error-linear-unit>, 2022. dostęp: 24.06.2022.
- [54] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 240–248. Springer, 2017.
- [55] H. H. Sultan, N. M. Salem, and W. Al-Atabany. Multi-classification of brain tumor images using deep neural network. *IEEE access*, 7:69215–69225, 2019.
- [56] M. Tekieli. A vision-based measurement system for the analysis of structural element deformation fields, praca doktorska, politechnika krakowska. 2019.
- [57] F. Tian, Y. Zhao, X. Che, Y. Zhao, and D. Xin. Concrete crack identification and image mosaic based on image processing. *Applied Sciences*, 9(22):4826, 2019.
- [58] Wei Ding; Han Yang, Ke Yu, Jiangpeng Shu. Crack detection and quantification for concrete structures using UAV and Swin-Transformer. *Journal of Structural Engineering*.
- [59] M. Yani et al. Application of transfer learning using convolutional neural network method for early detection of terry’s nail. In *Journal of Physics: Conference Series*, volume 1201, page 012052. IOP Publishing, 2019.
- [60] A. Zhang, K. C. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li, and C. Chen. Automated pixel-level pavement crack detection on 3d asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering*, 32(10):805–819, 2017.
- [61] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu. Road crack detection using deep convolutional neural network. In *2016 IEEE international conference on image processing (ICIP)*, pages 3708–3712. IEEE, 2016.

- [62] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang. Deepcrack: Learning hierarchical convolutional features for crack detection. *IEEE Transactions on Image Processing*, 28(3):1498–1512, 2019.