

Instytut Techniki Cieplnej



na kierunku Mechanika i projektowanie maszyn

w specjalności Lotnictwo

Wykrywanie defektów materiałów z wykorzystaniem algorytmów sztucznej inteligencji

Maciej Tarwacki

Numer albumu 324142

promotor

Dr inż. Mateusz Żbikowski

Warszawa, 2024

**Streszczenie**

Celem pracy jest stworzenie modeli uczenia maszynowego pozwalającego wykrywać i klasyfikować defekty materiałów. W pracy użyto danych ze zbioru *Defects location for metal surface the GC10-DET dataset* ze strony kaggle.com. Dataset składa się z 2306 czarno-białych fotografii. Do zdjęć dołączone są informacje na temat rodzaju defektu oraz miejsca, w którym dany defekt się znajduje. W pracy do stworzenia modelu uczenia maszynowego wykorzystano język programowania Python wraz z biblioteką TensorFlow.

**Summary**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

**Oświadczenie autora (autorów) pracy**

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

* niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
* niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
* niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
* wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
* znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

........................................ .........................................

data podpis autora (autorów) pracy

**Oświadczenie**

Wyrażam zgodę / nie wyrażam zgody\*1 na udostępnianie osobom zainteresowanym mojej pracy dyplomowej. Praca może być udostępniana w pomieszczeniach biblioteki wydziałowej. Zgoda na udostępnienie pracy dyplomowej nie oznacza wyrażenia zgody na jej kopiowanie w całości lub w części.

Brak zgody nie oznacza ograniczenia dostępu do pracy dyplomowej osób:

- reprezentujących władze Politechniki Warszawskiej,

- członków Komisji Akredytacyjnych,

- funkcjonariuszy służb państwowych i innych osób uprawnionych, na mocy odpowiednich przepisów prawnych obowiązujących na terenie Rzeczypospolitej Polskiej, do swobodnego dostępu do materiałów chronionych międzynarodowymi przepisami o prawach autorskich. Brak zgody nie wyklucza także kontroli tekstu pracy dyplomowej w systemie antyplagiatowym.

........................................ .........................................

data podpis autora (autorów) pracy

\*1 - niepotrzebne skreślić

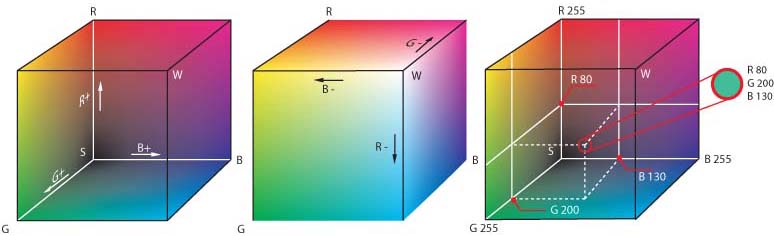
**SPIS TREŚCI**

1. **Wstęp teoretyczny**

Sztuczna inteligencja (ang. Artificial Inteligence w skrócie AI) od wielu lat jest obiektem zainteresowania naukowców, inżynierów i szeroko rozumianego biznesu. Wydaje się, jednak że dopiero na przełomie 2022 i 2023 roku trafiła ona do szerokiej grupy odbiorców. Wszystko za sprawą ChatuGPT, który potrzebował zaledwie 2 miesięcy by zdobyć 100 milionów użytkowników. Sztuczna inteligencja jednak jest używana przez większość z nas na co dzień, mimo że często nie zdajemy sobie z tego sprawy. AI można obecnie znaleźć niemalże wszędzie w internecie: korzystając z wyszukiwarki internetowej, używając systemy rozpoznawania mowy lub twarzy, oglądając spersonalizowane reklamy online, nie wspominają już o autonomicznych samochodach. Krótko mówiąc sztuczna inteligencja jest wokół nas. W niniejszej pracy przedstawię jak różne algorytmy sztucznej inteligencji mogą być przydatne w przemyśle, a konkretniej podczas wykrywania defektów materiałów.

* 1. **Podstawy przetwarzania obrazów**

Obraz cyfrowy można opisać przez dwuwymiarową funkcję g(x, y), której argumentami są dyskretne piksele. Wartościami tej funkcji są kolory w przestrzeni RGB, która jest kombinacja trzech liczb odpowiadającym kolejno kolorom czerwonemu, zielonemu i niebieskiemu. Każdej ze składowych przypisany jest jeden bajt pamięci (8 bitów), dlatego każdy z kolorów podstawowych może przyjąć 256 wartości. Oznacza to, że jest możliwe ponad 16,7 mln możliwości kolorów dla jednego piksela. Przyjmuje się że wartości (0, 0, 0) odpowiada kolorowi czarnemu, natomiast (255, 255, 255) kolorowi białemu. Na poniższym rysunku 1. przedstawiono sześcian kolorów RGB. [1][2][3]



Rys. 1. Sześcian kolorów RGB. [4]

Obok przestrzeni RGB innym popularnym modelem reprezentacji barw jest model CMY. Opisują go barwy podstawowe: żółto-niebieski (Cyan), purpurowy (Magenta) i żółty (Yellow). Jednak ze względów praktycznych (trudność uzyskania koloru czarnego z barw CMY) do kolorów dodaje się dodatkowo czarny – wtedy uzyskujemy przestrzeń barw CMYK [5].

Oprócz wyżej opisanego formatu kolorowego można wyróżnić prostsze (zajmujące mniej pamięci) reprezentację pojedynczego punktu. Wyróżniamy również formaty [1][5]:

- binarny (czarno-biały), będący najprostszym formatem. Jeden bajt zapisuje informacje o ośmiu punktach obrazu. Wartościom 0 przypisywany jest kolor czarny, a 1 biały. Obrazy binarne wykorzystuje się do pomiarów wielkości geometrycznych oraz wykorzystywane są do wielu algorytmów przetwarzających obrazy [7].

- monochromatyczny (odcień szarości), wyrażającą względną jasność punktu. Jeden bajt przechowuje informacje o jednym punkcie. Format ten pozwala na zakodowanie 256 stopni szarości. Poziom jasności równy 255 to kolor biały, a 0 to czerń. Obrazy szare, podobnie jak czarno-białe, są często używane w algorytmach przetwarzania obrazów.

Na poniższym rysunku 2. przedstawiono różnice między formatem kolorowy, binarnym oraz monochromatycznym.



Rys. 2. Porówanie formatu kolorowego, monochromatycznego i binarnego [6]

Obraz cyfrowy, który opisywany jest przez funkcje g(x,y) posiada początek układu współrzędnych w lewym górnym rogu obrazu, co zostało przedstawione na rysunku 3.

Obraz zawierający tekst, diagram, Prostokąt, linia

Opis wygenerowany automatycznie

Rys. 2. Przedstawienie obrazu cyfrowego przez funkcję dyskrętną g(x, y). [2]

Często stosowanym przedstawieniem obrazu cyfrowego jest przestawienie go w postaci następującej macierzy. [8]

Wartości macierzy reprezentują piksele obrazu cyfrowego. Liczby M i N określające liczbę kolumn i wierszy, determinuje nam również rozdzielność obrazu, która wynosi M x N. Natomiast iloczyn M i N określają liczbę pikseli w obrazie.

* + 1. **Przekształcenie kontekstowe**

Przekształcenie kontekstowe to grupa metod przetwarzania obrazów. Jak sama nazwa wskazuje przekształcenie jest wykonywane w pewnym kontekście (otoczeniu) pojedynczego piksela. Wartość piksela po przekształceniu zależy zarówno od wartości tego piksela, jak również od najbliższych pikseli. Jako otoczenie lokalne najczęściej wykorzystuje się macierz 3x3, 5x5, 7x7 (czasem większe), gdzie środkiem macierzy jest interesujący nas piksel. Taką macierz nazywamy maską filtru. Natomiast przekształcenia tego typu noszą również nazwę filtracją przestrzenną obrazów. Podczas filtracji maska filtru przechodzi przez każdy piksel obrazu i na tej podstawie obliczane są wartości nowego piksela. W zależności jaki efekt końcowy chcemy uzyskać stosuje się maski filtru z różnymi współczynnikami wagowymi. Wyróżnia się dwie grupy filtrów: dolnoprzepustowe (wygładzanie, redukcja szumów), górnoprzepustowe (wyostrzanie, detekcja konturów). [15]

Przykładowe zastosowania przekształceń kontekstowych [16]:

- pozbycie się z obrazu szumu,

- wzmocnienie konkretnych elementów na obrazie,

- usunięcie określonych wad z obrazu,

- poprawa jakości obrazu,

- odtworzenie częściowo zniszczonego obrazu.

* 1. **Konwolucyjne sieci neuronowe**

Konwolucyjne (inaczej splotowe) sieci neuronowe (ang. convolutional neural networks – CNN) są najpopularniejszymi modelami do przetwarzana obrazu [10]. Sieci te potrafią wyostrzyć ważne cechy, co jest pożądane w rozpoznawaniu lub klasyfikacji wzorców [12]. Splotowe sieci złożone są zazwyczaj z:

- obrazu (obrazów) wejściowych

- kilku warstw splotowych (ang. convolution layer) naprzemiennie występującymi z warstwami ReLU,

- warstw łączącej (ang. pooling layer),

- sieci neuronowej, która zawiera kilka w pełni połączonych warstw (ang. fully connected).

Na rysunku x pokazano jak przedstawia się typowa architektura konwolucyjnych sieci neuronowych.

Obraz zawierający tekst, diagram, zrzut ekranu

Opis wygenerowany automatycznie

Rys. x. Przykładowa architektura splotowych sieci neuronowych [13].

* + 1. **Warstwa splotowa**

Splotem (inaczej konwolucją od ang. convolution) nazywamy matematyczne działanie na dwóch funkcjach f i g:

Splot jest również wykorzystywany w przekształceniach kontekstowym. Wówczas mamy do czynienia ze splotem dyskretnym (cyfrowym), który opisywany jest wzorem:

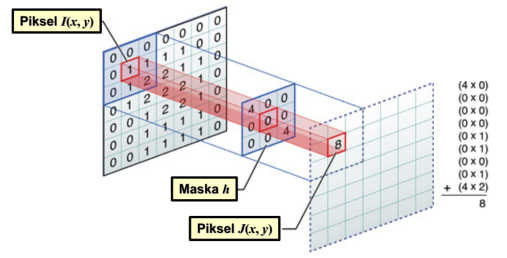
Gdzie:

J – obraz po przekształceniu,

I - obraz, który ulega przekształceniu,

h – maska filtru (inaczej filtr).

Wartość nowego piksela jest obliczana przez wymnożenie wartości pikseli przez odpowiednie wagi w masce filtru, a następnie dokonywane jest sumowanie otrzymanych ilorazów. Pojedynczą sekwencje operacji filtracji z użyciem splotu przedstawia rysunek x.



Rys. x. Schemat obliczenia nowego piksela w operacji filtracji. [x]

Podczas obliczania splotu obrazu występuje tak zwany problem brzegu. Polega on na tym, że brzegowe wartości nie posiadają wszystkich „sąsiadów”. Istnieją różne rozwiązania tego problemu. Najczęstszymi z nich są [17]:

- przyjęcie jakiejś wartości poza obszarem obrazu (najczęściej przyjmuje się 0),

- odbicie lustrzane obrazu poza obszarem,

- powielenie brzegowych wartości poza obszarem obrazu,

- obcięcie maski filtru by na brzegach nie wychodziła poza obszar.

W konwolucyjnych sieciach neuronowych stosuje się różnego rodzaju filtry, po użyciu których uzyskane obrazy noszą nazwę mapy cech. Sieci CNN podczas fazy uczenia dobierają takie filtry, które są najprzydatniejsze do powierzonego im zadania [9]. Konwolucja, jak sama nazwa wskazuje, jest najbardziej istotną operacją w konwolucyjnych sieciach neuronowych. Piksele obliczone w warstwie splotowej można traktować jako neurony, jednak nie są one połączone z każdym punktem obrazu, a połączone są tylko z pikselami znajdującymi się w obszarze filtru. W CNN stosuje się zazwyczaj kilka warstw splotowych, przez co w pierwszej z nich sieć skupia się na cechach ogólnych, a w kolejnych warstwach łączy te cechy w bardziej złożone schematy. Podobnie jak w klasycznych sieciach neuronowych, wartość cechy można przekazać do funkcji aktywacji.

Wyżej wspomniana funkcja aktywacji jest to funkcja nieliniowa, do której trafiają sygnały wejściowe przemnożone przez odpowiednie wagi. Innymi słowy do neuronu wartością wejściową jest średnia ważona sygnałów wejściowych, a wyjściową wartością jest wartość funkcji aktywacji dla argumentu równemu średniej ważonej. Najpopularniejszymi funkcjami aktywacji są [11]:

- tangens hiperboliczny:

- funkcja ReLu:

- funkcja logistyczna: f

- funkcja skokowa Heaviside’a

Na rysunku x przedstawiono jak wyglądają wyżej wymienione funkcje i ich pochodne.

Obraz zawierający linia, Wykres, diagram, tekst

Opis wygenerowany automatycznie

Rys. x. Wykresy funkcji aktywacji i ich pochodnych. [11]

* + 1. **Warstwa łącząca**

Celem warstwy łączącej (agregującej) jest zmniejszenie obrazu wejściowego, aby obniżyć obciążenie obliczeniowe i pamięciowe. Zadaniem tej warstwy jest również zredukowanie liczby parametrów, co powoduje zmniejszenie ryzyka występowania zjawiska przetrenowania. Idea działania warstwy łączącej jest podobna do tej z warstwy splotowej. W tej operacji filtr (nazywanym również jądrem łączącym) jest przesuwany przez całą macierz obrazu i obliczana jest wartość nowego piksela. Podczas stosowania filtru 2x2 obraz ulega zmniejsza się 4-krotnie – jeden piksel reprezentuje kwadrat 2x2. Do obliczenia nowego piksela stosuje się tak zwaną funkcję agregacji. Do najczęściej używanych funkcji agregacji możemy zaliczyć [9][11]:

- funkcję maksymalizującą (ang. max pooling), która jest najczęściej stosowana,

- funkcję uśredniającą (ang. average pooling).

Na rysunku x przedstawiono techniki max pooling i average pooling.

Obraz zawierający tekst, diagram, zrzut ekranu, linia

Opis wygenerowany automatycznie

Rys. x. Schemat działania agregacji z użyciem maksimum i z użyciem średniej. [9]

Pomimo tego, że warstwy łączące są obecne w konwolucyjnych sieciach neuronowych, to wiele osób zajmujących się sztuczną inteligencją podważa jej użyteczność [9]. Wadą warstwy pooling jest oczywiście zmniejszenie ilości informacji w stosunku do obrazu wejściowego, po operacji agregacji zostajemy tylko z 25% danych. W związku z tym część dostępnych architektur sieci konwolucyjnych nie stosuje warstwy łączącej lub używa jej w znacznie ograniczony sposób. Przykładem są sieci Residual Network (ResNet).

* + 1. **Sieć neuronowa**

Inspiracją do powstania sztucznych sieci neuronowych jest oczywiście ludzki mózg. Swoimi cechami nie dorównuje swojemu pierwowzorowi, ale w dzisiejszych czasach jest szeroko stosowana do rozwiązywania różnorakich problemów. Sztuczne neurony są traktowane jako jednostki obliczeniowe. Zasady działania neuronów została opisana w podrozdziale 1.2.1. Sieci neuronowe są złożone z trzech warstw: wejściową, ukrytą i wyjściową.

Obraz zawierający diagram, krąg, linia, Czcionka

Opis wygenerowany automatycznie

Rys. x. Schemat sieci neuronowej. [14]

W konwolucyjnych sieciach przed przekazaniem danych do sieci neuronowej trzeba jeszcze dokonać operację Flatten. Po ostatniej warstwie łączącej danymi wyjściowymi jest tablice trójwymiarowa, która należy przekształcić w wektory. Każda taka tablica posiada 3 wymiary: kanały, wysokość, szerokość obrazu. Jak operacja Flatten przekształca tablice w wektor dobrze obrazuje rysunek x. [9]

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie

Rys. x. Schemat sieci neuronowej. [18]

1. **Wstęp teoretyczny**

Wykrywanie i klasyfikacja defektów różnego typu materiałów za pomocą sztucznej inteligencji jest zagadnieniem, w którym prowadzone są różnego typu badania. Temat jest rozwijany również przez firmy na rynku, gdyż automatyczne znajdowanie defektów może być wykorzystywane w kontroli jakości lub sprawdzeniu czy dany komponent wciąż może być użytkowany.

W literaturze używa się różnych rodzajów modeli w celu detekcji defektu materiałów. Często modele bazują na popularnych algorytmach, które są ulepszane, modyfikowane i dostosowywane do danych. Do przykładowych modeli i algorytmów na których bazują modele w literaturze można zaliczyć:

- ResNet50 [19] [21] [22]

- SVM – Support vector machine (Metoda wektorów nośnych) [20]

- Faster R-CNN [21] [22]

- Konwolucyjne sieci neuronowe CNN [23]

- CNN wraz nauczaniem transferowym TL (Transfer Learning) [24]

- ResNet34 [25]

- MobileNet v3 [26]

W tym rozdziale opisane zostało kilka artykułów lub prac, które przedstawiają modele do detekcji i klasyfikacji defektów na powierzchni stali.

* 1. **Intelligent detection and classification of surface defects on cold-rolled galvanized steel strips using a data-driven faulty model with attention mechanism**

W pracy pod tytułem „Intelligent detection and classification of surface defects on cold-rolled galvanized steel strips using a data-driven faulty model with attention mechanism” zaproponowano model głębokiego uczenia nazwany SSFDANet (steel surface faulty detection attention net) [25]. Model automatycznie znajduje i klasyfikuje 20 rodzajów defektów wykrytych w taśmie stalowej. SSFDANet jest oparty o konwolucyjne sieci neuronowe z tak zwanym mechanizmem uwagi, który powoduje poprawę działania sieci neuronowej. Działa on poprzez dobieranie odpowiednich charakterystyk wejściowych, co sprawia, że sieć koncentruje się bardziej na najistotniejszych cechach a tłumi mniej znaczące. Główną strukturą modelu jest ResNet, którego schemat został przedstawiony na rysunku x.

A diagram of a computer

Description automatically generated

Rys. x. Schemat sieci rezydualnej (ResNet) [11]

Zastosowany model SSFDANet składa się z dwóch elementów:

- RoINet,

- ClassificationNet.

Pierwszy z części ma za zadanie rozpoznanie tak zwanego Region of Interests (ROI), czyli wydzielenie obszaru w których znajduje się potencjalny defekt materiału. Drugi z części klasyfikuje jaki typ defektu znajduję się w ROI. Dla sieci RoINet danymi wejściowymi są zdjęcia stalowych taśm z defektami. Dla sieci ClassificationNet danymi wejściowymi są dane wejściowe z RoINet, a danymi wyjściowymi odpowiednio zaklasyfikowane defekty. Architekturę poszczególnych elementów modelu SSFDANet przedstawiono na rysunkach x i x.

A diagram of a diagram of a block diagram

Description automatically generated with medium confidence

Rys. x. Schemat RoINet

A diagram of a diagram

Description automatically generated

Rys. x. Schemat ClassificationNet.

Model SSFDANet posłużył do klasyfikacji 20 kategorii (19 defektów i taśma bez defektu), a zestaw danych posiadał 20 tys. zdjęć. Każde ze zdjęć było rozdzielczości 1024x128. Dane zostały podzielone na grupę uczącą (60%), grupę walidacyjną (30%) i grupę testową (10%). Przykładowe zdjęcia przedstawiające defekty przedstawiono na rysunku x. W doświadczeniu użyto biblioteki Tensoflow2.

A collage of different types of lines

Description automatically generated

Rys. x. Przykłady rodzajów defektów w taśmie stalowej oraz taśma bez defektu.

Algorytm RoINet okazał się przewidywać większe RoI niż w rzeczywistości. Powodem tego było znaczna czułość modelu na zmianę na powierzchni w pobliżu defektu. Jednak ta cecha okazuje się przydatna w dalszej klasyfikacji defektów, dlatego nie została ona poprawiona w modelu.

Wyniki uzyskane w pracy porównano do 3 popularnych modeli: drzewo decyzyjne, VGG19, ResNet. Ponadto ClassificationNet (CNet) był również używany w wersji bez mechanizmu uwagi. Każdy model był testowany w dwóch konfiguracjach – z i bez określania RoI defektu. Do porównania modeli użyto dokładności, a wyniki zostały przedstawione w tabeli x i x.

Tab. x. Całkowita dokładność modeli.

| **Model** | **Without RoI** | **With RoI** |
| --- | --- | --- |
| Decision Tree | 66.94% | 76.91% |
| VGG19 | 74.46% | 82.35% |
| ResNet | 80.13% | 87.34% |
| CNet | 84.24% | 89.28% |
| CNet without A | 80.92% | 87.18% |

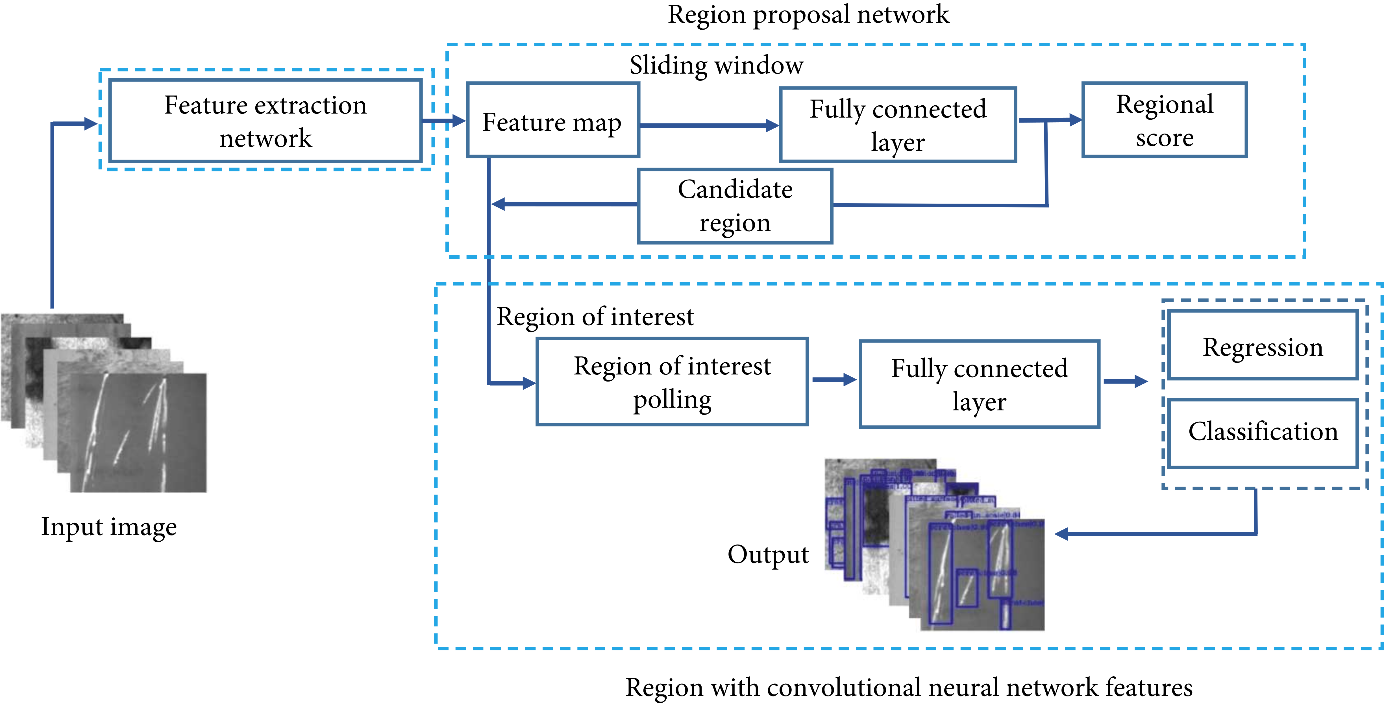
Tab. x. Całkowita dokładność i dla kilku defektu.

| **Model** | **Overall** | **Material**  **Lose** | **Dark**  **Spots** | **Dark**  **Lines** | **Scratch** | **Slag** | **Hole** | **Warped**  **Skin** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decision Tree | 76.91% | 59.12% | 82.11% | 72.78% | 68.93% | 48.39% | 64.26% | 68.79% |
| VGG19 | 82.35% | 76.27% | 85.38% | 74.35% | 68.68% | 87.47% | 78.18% | 74.55% |
| ResNet | 87.34% | 75.74% | 90.56% | 78.03% | 77.25% | 69.56% | 77.35% | 87.27% |
| SSFDANet | 89.28% | 87.15% | 93.69% | 85.36% | 87.23% | 85.50% | 89.84% | 87.46% |

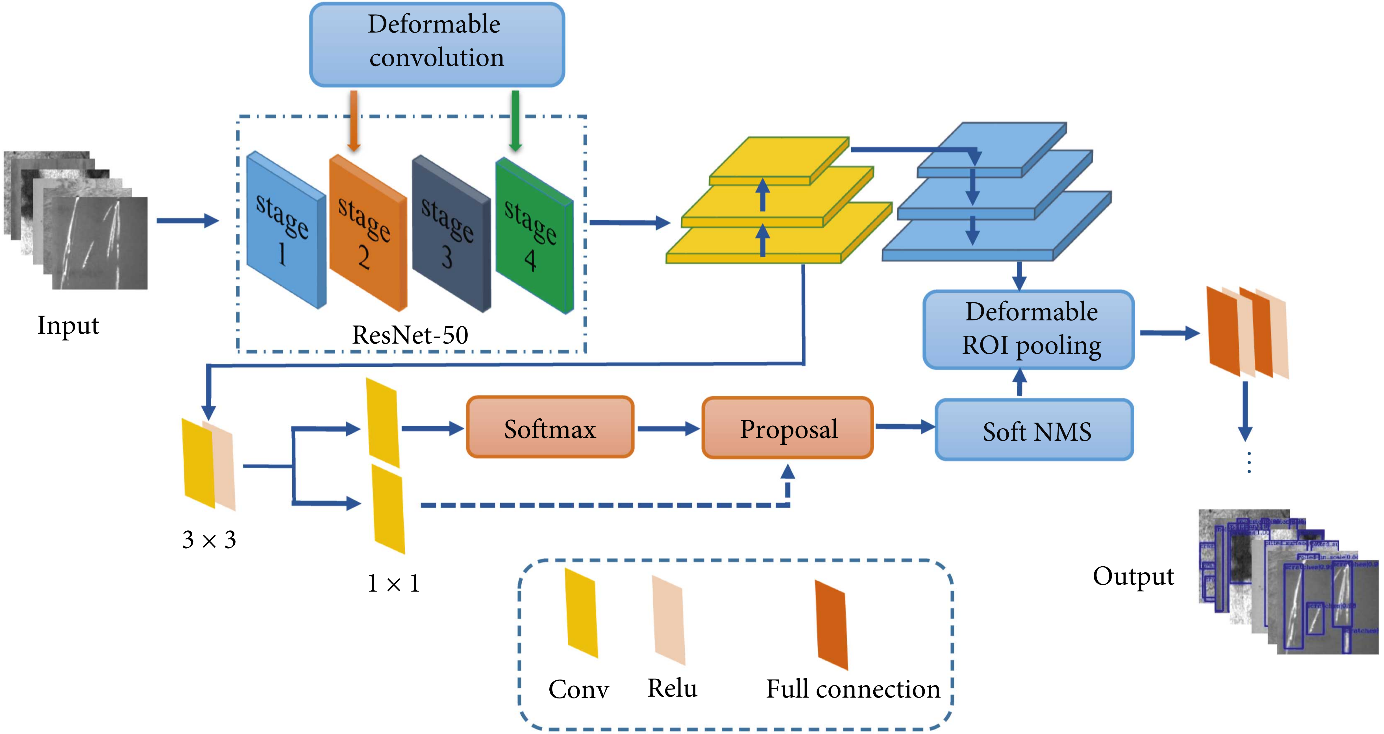
Jak można zauważyć, dodanie RoI poprawia dokładność w znaczący sposób dla każdego z modeli (w niektórych przypadkach nawet o 10 punktów procentowych). Zastosowanie modelu SSFDANet z mechanizmem uwagi pozwala osiągnąć 89,28% dokładności, co jest najlepszym wynikiem dla przetestowanych modeli. Tabela x wskazuje duże odchylenia precyzji dla niektórych defektów np. slag czy material lose dla decision tree. Takie znaczące odchylenia nie występują dla modelu zaproponowanego w tej pracy. SSFDANet sprawdza się zatem lepiej do klasyfikacji defektów niż popularnie wykorzystywane modele takie jak VGG19 czy ResNet.

* 1. **A new steel defect detection algorithm based on deep learning**

W artykule o nazwie „A new steel defect detection algorithm based on deep learning”, w którym dla danych NEU-DET zaproponowano model bazujący na zmodyfikowanej szybkiej sieci R-CNN (Fast R-CNN) [22]. Schemat klasycznej sieci szybkiej R-CNN przedstawiono na rysunku x, a schemat modelu zaproponowanego w artykule przedstawiono na rysunku x.



Rys. x. Schemat modelu Faster R-CNN.



Rys. x. Schemat ulepszonego modelu Faster R-CNN.

W zaproponowanym algorytmie za wydobywanie cech odpowiedzialna jest sieć ResNet-50. Z racji tego, że w klasycznym CNN część informacji jest tracona, używana jest tutaj sieć ResNet. Cechą tej sieci jest fakt, że przesyła ona dane wejściowe bezpośrednio do warstwy wyjściowej, co zapobiega traceniu części danych. W modelu wykorzystano tak zwany deformable convolution. Klasyczna konwolucja używa filtra kwadratowego, który może działać niewystarczająco dobrze dla kształtów o nieregularnej budowie. Deformable convolution może dopasować kształt filtra, żeby lepiej analizować cechy na obrazie. Filtr dynamicznie dostosowuje się do danych wejściowych. Porównanie obliczeń dokonywanych przez klasyczną konwolucję i przed deformable convolution została przedstawiona na rysunku x. Widać lepsze adaptowanie się do defektów to nieregularnym kształcie.

A group of images of a person's brain

Description automatically generated

Rys. x. Klasyczna konwolucja (a) i defermable convolution.

Innymi ulepszeniami modelu Faster R-CNN, które użyto w modelu są:

- Feature pyramid network FPN,

- Deformable pooling of ROI,

- Soft non-maximum suppression (soft NMS).

FPN jest to metoda pozwalająca na detekcję obiektów mających różne rozmiary na obrazie. Feature pyramid network w tym przypadku jest zastosowana, że model był zdolny do detekcji defektów o małej powierzchni.

Deformable pooling of ROI, podobnie jak deformable convolution, jest przydatne, jeśli kształty defektów są nieregularne. Ten moduł zwiększa zdolność detekcji modelu oraz powoduje, że proponowane obszary są większe – w celu lepszego identyfikowania złożonych defektów w stali.

Soft NMS używany jest w celu zapobieżenia niewykrycia defektu oraz zmniejsza prawdopodobieństwo fałszywej detekcji. Według autorów Soft NMS potrafi zwiększyć dokładność modelu, a sama metoda jest bardzo łatwy w zaimplementowaniu i nie wymaga dodatkowego trenowania modelu. [27]

Twórcy artykułu stworzyli 5 schematów modeli, żeby sprawdzić jak każda modyfikacja powoduje ulepszenie modelu. W tabeli x przedstawiono, które ulepszenia zastosowano w modelach. Schemat 1 odpowiada oryginalnemu Faster R-CNN modelowi, natomiast ostatni schemat posiada wszystkie przedstawione ulepszenia.

Tab. x. Modele zaprezentowane w pracy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | ResNet-50 | Deformable networks | FPN | Soft NMS |
| 1 | × | × | × | × |
| 2 | √ | × | × | × |
| 3 | √ | √ | × | × |
| 4 | √ | √ | √ | × |
| **5** | **√** | **√** | **√** | **√** |
|  |  |  |  |  |

Tab. x. Porównanie precyzji dla każdego z modeli

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | AP | | | | | | mAP | mAP increasing |
| *a* | *b* | *c* | *d* | *e* | *f* |
| 1 | 0.250 | 0.652 | 0.752 | 0.735 | 0.545 | 0.811 | 0.624 | − |
| 2 | 0.252 | 0.677 | 0.772 | 0.789 | 0.555 | 0.816 | 0.643 | +0.019 |
| 3 | 0.458 | 0.738 | 0.841 | 0.782 | 0.583 | 0.881 | 0.714 | +0.090 |
| 4 | 0.516 | 0.754 | 0.768 | 0.854 | 0.610 | 0.902 | 0.745 | +0.121 |
| **5** | **0.501** | **0.791** | **0.792** | **0.874** | **0.649** | **0.905** | **0.752** | **+ 0.128** |

W tabeli x przedstawiono wyniki dla każdego z modeli. Przedstawiono precyzję (AP) dla poszczególnych klas (od a do f) oraz dokładność dla całego modelu. Każda kolejna modyfikacja powoduje wzrost dokładności całego modelu. Największy wzrost dokładności widać po zastosowaniu deformable network czyli opisanego wcześniej deformable convolution.

W pracy przedstawiono również porównanie zaproponowanego algorytmu do popularnych algorytmów stosowanych do klasyfikacji. Wyniki przedstawiono w tabeli x.

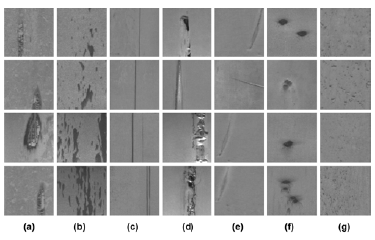
Tab. x. Porównanie precyzji dla każdego z modeli

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithms | AP | | | | | | mAP | mAP increasing |
| *a* | *b* | *c* | *d* | *e* | *f* |
|  | | | | | | | | |
| SSD | 0.302 | 0.522 | 0.620 | 0.390 | 0.709 | 0.515 | 0.510 | +0.242 |
| RetinaNet | 0.391 | 0.676 | 0.772 | 0.711 | 0.546 | 0.542 | 0.606 | +0146 |
| Cascade R-CNN | 0.321 | 0601 | 0.794 | 0.723 | 0.509 | 0.805 | 0.626 | +0.126 |
| This paper | **0.501** | **0.791** | **0.792** | **0.874** | **0.649** | **0.905** | **0.752** | − |

Z tabeli x można wywnioskować, że zaproponowany algorytm jest lepszy niż algorytm SDD, RetinaNet i Cascade R-NN. Widać znaczący wzrost precyzji, szczególnie dla klas b, d i f.

* 1. **Classification of strip steel surface defects based on data augmentation combined with MobileNet**

Kolejną interesującą pracą w zakresie klasyfikacji defektów powierzchni stalowych jest „Classification of strip steel surface defects based on data augmentation combined with MobileNet”[27]. W pracy zaproponowano model bazujący na sieci neuronowej MobileNet v3 oraz na metodzie augmentacji danych. Model uzyskał dokładność ponad 94% na zestawie danych składającym się 1360 zdjęć z 7 rodzajami defektów. Dane mają nazwę X-SDD i pochodzą z walcowania na gorąco stali. Każde zdjęcie jest wymiaru 128x128 pikseli. Rysunek x przedstawia przykłady zdjęć z X-SDD.



Rys. x. Przykłady zdjęć z datasetu.

Przestawione na rysunku x przykłady zdjęć odpowiadają następującym defektom stali:

- (a) – oxide scale of plate system,

- (b) – red iron sheet,

- (c) – surface scratches,

- (d) – slag inclusions,

- (e) – finishing roll printing,

- (f) – iron sheet ash,

- (g) – oxide scale of temperature system.

Augmentacja danych (inaczej dogenerowanie danych) polega na powiększeniu próbki danych poprzez sztuczne wytworzenie przy pomocy pewnych przekształceń, których rezultatem są nowe obrazy. Ta metoda pozwala ograniczyć przetrenowanie modelu – zmniejsza ryzyko zbytniego dopasowania modelu. Augmentacja pozwala zwiększyć dokładność modelu do nawet 82% [10]. Dogenerowanie danych polega na operacjach na zdjęciach takich jak: przesuwanie, obracanie, zmienianie rozmiaru, odbijanie lustrzane czy zmienianie kontrastu zdjęć itp. W cytowanej w tym podrozdziale pracy augmentacja opierała się na dodaniu do zdjęcia jednokolorowego (o jednej wartości pikseli) obszaru o tych samych rozmiarach, co oryginalne zdjęcie. Rysunek x przedstawia jak wyglądają dane po tej operacji.

A close-up of a black and white image

Description automatically generated

Rys. x. Zdjęcie po augmentacji. Wymiary zdjęć to 128x256 lub 256x128

Jak można zauważyć na rysunku x, po dodaniu jednokolorowego obszaru liczba zdjęć zwiększyła się 4-krotnie. Ponadto augmentacje przeprowadzono również poprzez dodanie 3 jednokolorowych obszarów, wówczas otrzymuje się zdjęcia o rozmiarze 256x256. Tego typu zdjęcia przedstawiono na rysunku x.

A close-up of a black and white photo

Description automatically generated

Rys. x. Zdjęcie po augmentacji. Wymiary zdjęć to 256x256

Po wykonaniu augmentacji zbiór danych zwiększył się 8-krotnie. Ten zabieg nie tylko spowodował wzrost danych do wykorzystania, ale również spowodował, że defekty są w innych miejscach na zdjęciu, co jest korzystne dla modelu. Z racji tego, że dane wejściowe do sieci neuronowe zastosowanej w pracy powinny mieć rozmiar 224x224. Wszystkie dane przeskalowano do tego rozmiaru. Augmentacji użyto na każdym zdjęciu z zestawy szkoleniowego zdjęć (70%), reszta zdjęć została wykorzystana do testowania modelu.

Zastosowany w pracy MobileNet v3 large to konwolucyjna sieć neuronowa stworzona przez Google’a z myślą o zastosowaniu na urządzenia mobilne. Sieć posiada relatywnie mało parametrów, co czyni ją mało obciążającą dla procesora. MobileNet v3 został wybrany w pracy z powodu dużej wydajności oraz lepszej zdolności do klasyfikacji w porównaniu do sieci neuronowej podobnej klasy. MobileNet v3 można znaleźć w popularnych bibliotekach związanych z uczeniem maszynowym dla języka programowania Python (np. Pytorch czy Keras).

Do przetestowania modelu oraz porównania go do innych modeli użyto wskaźników takich jak: dokładność (accuracy), precyzja (precision) i czułość (recall). Korzystano z następujących wzorów

-

- ,

- ,

- ,

-

Gdzie:

- – liczba wszystkich zdjęć w zestawie testowym,

- – liczba wszystkich typów defektów, w tym wypadku 7,

- – liczba prawdziwych pozytywnych (ang. True Positives), klasyfikacji dla odpowiedniego defektu,

- – liczba fałszywych pozytywnych (ang. False Positives) klasyfikacji dla odpowiedniego defektu. Zdjęcia zaklasyfikowana jako ten defekt, w rzeczywistości będąca innym defektem,

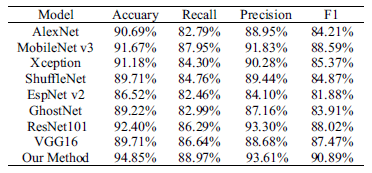
- – liczba fałszywych negatywnych (ang. False Negatives) klasyfikacji odpowiedniego defektu. Zdjęcia, które powinno być zaklasyfikowane jako danych defekt, ale są zaklasyfikowane jako inny defekt,

- – precyzja konkretnego defektu,

- – czułość konkretnego defektu.

Model z pracy porównano do: AlexNet, MobileNet v3, Xception, ShuffleNet, EspNet v2, GhostNet, ResNet101, VGG16. Porównanie przedstawiono w tabeli x.

Tab. x. Porównanie modelu z innymi popularnymi modelami.



Model uzyskał dokładność 94,85% co jest najlepszym wynikiem spośród wyżej wymienionych modeli. Dodanie augmentacji danych potrafiło zwiększyć dokładność o ponad 3 punkty procentowe. Wynik jest nawet lepszy od modeli, które znacznie bardziej obciążają procesor. Takimi modelami są tutaj ResNet101, VGG16 i Xception. Potrzebną moc obliczeniową i liczbę parametrów sieci pokazano w tabeli x.

Tab. x. Porównanie złożoności modeli.

A table of numbers and a model

Description automatically generated

Z tabeli można wywnioskować, że pomimo potrzeby małej mocy obliczeniowej oraz relatywnie małej liczby parametrów, model uzyskuje bardzo dobre wyniki w klasyfikowaniu defektów stalowych powierzchni.

1. **Model**
   1. **Konfiguracja oprogramowania i sprzęt**

Model do klasyfikacji defektów opisany w dalszej części pracy został napisany w języku programowania Python, używano edytora tekstu Jupyter Notebook. Korzystano z różnych bibliotek języka Python, w tym biblioteki związane z uczeniem maszynowy – scikit-learn, TensorFlow. Lista oprogramowania oraz ich wersji została przedstawiona w tabeli x.

Tab. x. Lista używanego oprogramowania i bibliotek wraz z wersjami.

|  |  |
| --- | --- |
| Oprogramowanie | Wersja |
| Windows 10 | 22H2 |
| Anaconda Navigator | 2.5.1 |
| Jupyter Notebook | 7.0.6 |
| Python | 3.10.13 |
| Pandas | 2.1.3 |
| Numpy | 1.26.2 |
| TensorFlow | 2.10.0 |
| sklearn | 1.3.2 |
| OpenCV | 4.8.1.78 |
| Matplotlib | 3.8.2 |
| CUDA | 11.2.2 |
| cuDNN | 8.10.77 |

Zainstalowana wersja TensorFlow pozwala na przeprowadzenie niektórych obliczeń na procesorze graficznym (GPU). Taka metoda obliczeń pozwala na szybsze ich wykonanie, porównując z tymi samymi obliczeniami wykonywanymi na procesorze (CPU). W tabeli x przedstawiono również parametry komputera na którym przeprowadzano obliczenia.

Tab. x. Parametry komputera.

|  |  |
| --- | --- |
| Podzespół | Nazwa/Parametry |
| Procesor | Intel(R) Core(TM) i5-10400F 2.90GHz |
| Karta graficzna | NVIDIA GeForce RTX 3060 |
| Pamięć RAM | 16GB |

* 1. **Zbiór danych – GC10-DET dataset**

Zbiór danych GC10-DET to zdjęcia przedstawiające 10 różnych defektów powierzchni blach stalowych, które powstały podczas walcowania na gorąco. Dataset został pobrany ze strony kaggle.com [28] oraz został przedstawiony w pracy pod tytułem "Deep Metallic Surface Defect Detection: the New Benchmark and Detection Network" [29]. Pobrany zbiór danych przedstawia 2294 czarno-białych o rozdzielczości 2048x1000 pikseli w formacie jpg. W danych znajdują się również 2280 pliki z rozszerzeniem xml, zawierające informacje na temat zdjęcia: nazwa zdjęcia, rodzaj defektu lub defektów występującego na zdjęciu, miejsce występowania defektu. 14 zdjęć nie miała przypisanego do siebie pliku z rozszerzeniem xml – te zdjęcia wykluczono z finalnego zbioru danych. Ponadto w zbiorze danych istnieją zdjęcia, na których znajduje się kilka różnych typów defektów. Założono, że defektem, który znajduje się na zdjęciu, jest defekt znajdujący się w nazwie folderu, w którym umieszczone jest zdjęcie. Tak stworzony zbiór danych przypisuje dane zdjęcie do jednego typu defektu. Defekty znajdujące się w bazie danych GC10-DET to:

- Silk spot (Ss) – dodać opisy

- Water spot (Ws) – są to ślady po wodzie powstające podczas suszenia,

- Welding line (Wl)

- Crescent gap (Cg)

- Punching hole (Pu)

- Inclusion (In)

- Oil spot (Os)

- Waist folding (Wf)

- Crease (Cr)

- Rolled pit (Rp)

Na zdjęciach x-y przedstawiono przykładowe zdjęcia przedstawiające każdy z defektów, zaznaczono miejsce występowania defektu.

A close-up of a silk spot

Description automatically generated Rys. x. Defekt typu Silk spot (Ss).

A grey square with red lines

Description automatically generated with medium confidence Rys. x. Defekt typu Water spot (Ws).

A graph showing a welding line

Description automatically generated Rys. x. Defekt typu Welding line (Wl).

A black and white image of a cross section

Description automatically generated Rys. x. Defekt typu Crescent gap (Cg).

A white sheet with numbers

Description automatically generated with medium confidence Rys. x. Defekt typu Punching hole (Pu).

A black and white image of a column

Description automatically generated Rys. x. Defekt typu Inclusion (In).

A black and white image of a spot

Description automatically generated Rys. x. Defekt typu Oil spot (Os).

A rectangular red rectangle with numbers

Description automatically generated with medium confidence Rys. x. Defekt typu Waist folding (Wf).

A grey rectangular object with a red line

Description automatically generated with medium confidence Rys. x. Defekt typu Crease (Cr).

A grey rectangular object with a red square

Description automatically generated

Rys. x. Defekt typu Rolled pit (Rp).

Liczba defektów nie jest sobie równa, co powoduje trudność w tworzeniu odpowiedniego modelu do celów klasyfikacji defektów. Liczba występowania większości defektów mieści się w przedziale od 200 do 300 zdjęć. Istnieją jednak odstępstwa: silk spot występuje 649 razy, crease 51, a rolled pit 28. Mając na względnie tak rzadkie występowanie dwóch ostatnich defektów, trzeba pamiętać o odpowiednim podziale zbioru na zbiór uczący, testowy i walidacyjny. Dokonano podziału w stosunku 70% zbiór uczący, 20% walidacyjny, 10% testowy. Pamiętano, by ten procentowy stosunek, występował również wewnątrz każdej klasy defektu. Ponadto, żeby zwiększyć możliwości uczenia modelu zastosowano augmentację danych – co pozwala zwiększyć precyzję przypisywania rzadko występujących defektów. Na rysunku x przedstawiono jak często występują dane defekty w zbiorze danych.

A graph of different types of lines

Description automatically generated with medium confidence

Rys. x. Częstość występowania defektów w zbiorze danych.

* 1. **Augmentacja danych**

1. **Wyniki**
2. **Wnioski**

**Bibliografia**

[1] – Komputerowa analiza i przetwarzanie obrazów

[2] - <https://www.ibspan.waw.pl//~jaworska/st_dok0309.pdf>

[3] – Cyfrowe przetwarzanie obrazów

[4] - <https://commons.wikimedia.org/wiki/File:RGB_farbwuerfel.jpg>

[5] - <https://home.agh.edu.pl/~tarasiuk/dydaktyka/doc/GFK/S/01.pdf>

[6] - https://www.researchgate.net/figure/Figure2-a-RGB-image-b-Gray-Scale-image-c-Binary-image\_fig2\_344249310

[7] – Leszek Wojnar Analiza Obrazu [WojnarL\_AnalizaObrazu.pdf](file:///C:\Users\Maciek\Desktop\Studia%20politechnika\Praca%20magisterska\literatura\WojnarL_AnalizaObrazu.pdf)

[8] - Przetwarzanie i analiza obrazów w systemach przemysłowych. Wybrane zastosowania. Sankowski

[9] – Uczenie głębokie od zera. Podstawy implementacji w Pythonie – Seth Weidman. Helion 2020

[10] – Deep Learning. Praca z językiem Python i biblioteką Keras – François Chollet. Helion 2019

[11] – Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow - Aurélien Géron. Helion 2018

[12] - <https://home.agh.edu.pl/~horzyk/lectures/ai/SztucznaInteligencja-UczenieG%C5%82%C4%99bokichSieciNeuronowych.pdf>

[13] - <https://developersbreach.com/convolution-neural-network-deep-learning/>

[14] – Machine learning, Python, i data sciene – Andreas C. Mueller, Sarah Guido. Helion 2017

[15] - <https://www.ire.pw.edu.pl/~trubel/grk/files/Lab2.pdf>

[16] - <https://enauczanie.pg.edu.pl/moodle/pluginfile.php/199973/mod_resource/content/2/wk-w04.pdf>

[17] - <https://home.agh.edu.pl/~dwornik/2022_Gin_Obrazy/Gin_07_filtracja_2D.pdf>

[18] - <https://www.educative.io/answers/what-is-a-neural-network-flatten-layer>

[19] - Steel Surface Defect Classification Using Deep Residual Neural Network - Ihor Konovalenko , Pavlo Maruschak, Janette Brezinová , Ján Vináš, Jakub Brezina

[20] - Steel surface defect classification using multiple hyper-spheres support vector machine with additional information - Rongfen Gong, Chengdong Wu , Maoxiang Chu, 2018

[21] - Automatic Detection and Classification of Steel Surface Defect Using Deep Convolutional Neural Networks - Shuai Wang, Xiaojun Xia, Lanqing Ye, Binbin Yang, 2021

[22] - A New Steel Defect Detection Algorithm Based on Deep Learning - Weidong Zhao, Feng Chen , Hancheng Huang, Dan Li, Wei Cheng – 2021

[23] - Classification Of Surface Defects On Steel Sheet Using Convolutional Neural Networks - Shiyang Zhou, Youping Chen, Dailin Zhang, Jingming Xie, Yunfei Zho, 2015

[24] - Industrial Laser Welding Defect Detection and Image Defect Recognition Based on Deep Learning Model Developed - Honggui Deng, Yu Cheng, Yuxin Feng, Junjiang Xiang, 2021

[25] - Intelligent Detection and Classification of Surface Defects on Cold-Rolled Galvanized Steel Strips Using a Data-Driven Faulty Model With Attention Mechanism - Hao Chen, Zhenguo Nie, Qingfeng Xu, Jianghua Fei, Yaguan Li, Hongbin Lin, Wenhui Fan, Xin-Jun Liu, 2023

[26] - Classification of strip steel surface defects based on data augmentation combined with MobileNet – Xinglong Feng, Xianwen Gao, Ling Luo - 2022

[27] - Soft-NMS – Improving Object Detection With One Line of Code - Navaneeth Bodla, Bharat Singh, Rama Chellappa. Larry S. Davis

[28] - https://www.kaggle.com/datasets/zhangyunsheng/defects-class-and-location

[29] - Deep Metallic Surface Defect Detection: the New Benchmark and Detection Network – Xiaoming Lv, Fajie Duan, Jia-jia Jiang, Xiao Fu, Lin Gan - 2020