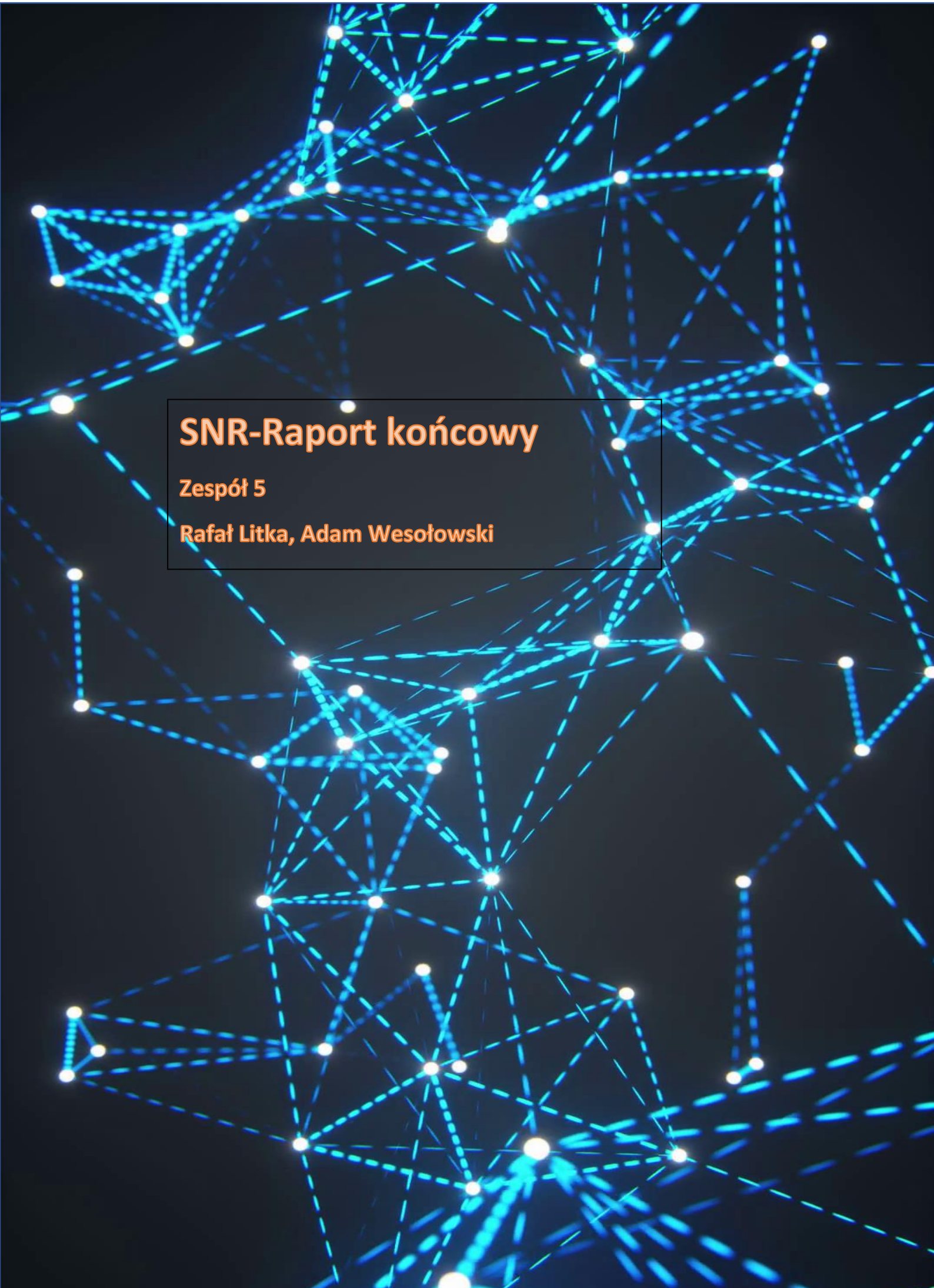


SNR-Raport końcowy

Zespół 5

Rafał Litka, Adam Wesołowski

Polecenie

Wykorzystując obrazy zawarte w katalogach „accept” oraz „reject” wytrenować głęboki klasyfikator neuronowy klasyfikujący deski z katalogu „unlabeled” na egzemplarze akceptowalne i te przeznaczone do odrzucenia. Na podstawie wyniku działania klasyfikatora testowanego na zbiorze obrazów z katalogu „unlabeled” określić (pisemnie) reguły odrzucania desek.

Wykorzystane środowisko:

- Python 3.7
- Numpy
- Tensorflow/Keras
- Scikit-learn
- Matplotlib

Sposób rozwiązania

1. Pierwszym krokiem było podzielenie zbioru danych na uczący i testujący. Nie tworzyliśmy zbioru walidacyjnego, ponieważ chcieliśmy przetestować działanie sieci tylko dla określonych liczb epok, a danych uczących było niewiele, a stworzenie kolejnego zbioru by tą liczbę jeszcze bardziej ograniczyło. Stworzyliśmy własny skrypt, który dane etykietowane dzielił we wskazanym stosunku na zbiór uczący i testujący i tworzył odpowiednią strukturę folderów, które mogły być wykorzystane przez generatory kerasa.
2. Do tworzenia zbiorów dla modelu wykorzystaliśmy generatory kerasa. Pozwalają one w bardzo łatwy sposób podzielić zbiory na kategorie – na podstawie folderów, w których się znajdują. Rozwiązanie to pozwala też ładować zbiory danych mniejszymi częściami, co jest znacznie efektywniejsze od ładowania całego zbioru na raz. Dane są też bardzo duże, gdyż są zdjęciami RGB w dosyć wysokiej rozdzielczości i mogłyby się nie zmieścić w pamięci gdyby zostały załadowane na raz. Generatory te pozwalają wybrać rozmiar *„Batcha”*, którym będziemy uczyć model, co pozwala uczyć model na zbiorze podzielonym na małe kawałki. Im większy był taki zbiór, tym model szybciej się uczył, jednakże stosowanie dużych zbiorów wykorzystywało dużo pamięci i nasz program czasami przestawał przez to działać. Generatory pozwalają też generować nowe dane na podstawie otrzymanych transformując obrazy. My wykorzystaliśmy przewrócenie zdjęcia w pionie, ponieważ zdjęcia są niemalże symetryczne w tej osi – z wyjątkiem samej deski.
3. Następnym krokiem było stworzenie modeli sieci neuronowych wykorzystywanych do klasyfikacji obrazów. Wykorzystaliśmy model Sequential z kerasa, który pozwala na tworzenie modelu szeregowo, warstwa po warstwie. Aby przetestować różne architektury stworzyliśmy 3 sieci o różnych rozmiarach warstw i głębokości. Sieci te budowaliśmy z następujących warstw:
    - Warstwa konwolucyjna/splotowa
      Warstwa ta bazuje na operacji splotu z bankiem liniowych filtrów dwuwymiarowych (zazwyczaj 3 x 3 lub 5 x 5) w celu określenia pewnych cech charakterystycznych danego obrazu 2D.

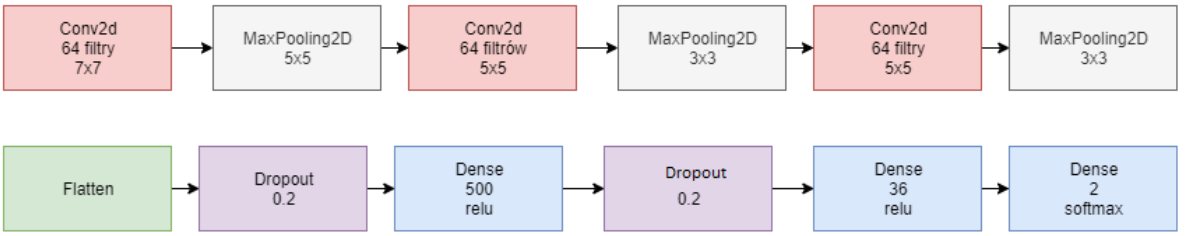
    - Warstwa MaxPool
      Jest to funkcja zmniejszająca wymiar obrazu poprzez wybieranie największej wartości z zadanego obszaru np. 2 x 2, 3 x 3. Doskonale nadaje się do zmniejszania wymiarowości obrazu i badania różnych poziomów uszczegółowienia.

- Warstwa spłaszczająca
  Warstwa ta zmienia kształt danych - zamienia wielowymiarowy zbiór na wektor.
- Warstwa Perceptronowa z aktywacją relu
- Warstwa Perceptronowa z aktywacją Softmax
  Ma rozmiar równy liczbie klasyfikowanych klas. Wyznacza prawdopobieństwo, że obraz należy do danej klasy. Suma wszystkich jej wyjść jest zawsze równa 1.

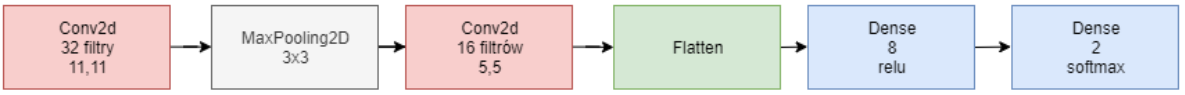
Generalna struktura naszych sieci prezentuje się następująco:
W pierwszej części odpowiadającej za wykrywanie cech na obrazie zastosowaliśmy naprzemiennie warstwy konwolucyjne i agregacyjne. Pozwala to wykrywać elementy zdjęć o różnych rozmiarach.
Następnie zastosowane jest spłaszczenie wyniku, i zwykła sieć neuronowa, której celem jest nauczenie się klasyfikowania danych na podstawie cech znalezionych przez warstwy konwolucyjne. Sieć zakończona jest warstwą softmax wyznaczającą wynik klasyfikacji.

Oto stworzone przez nas modele:

- Sieć 1
  Rozpoczyna się trzema naprzemiennymi warstwami konwolucyjnymi i agregującymi. Po spłaszczeniu zastosowane zostały naprzemiennie warstwy perceptronowe i odrzucające wagi. Warstwy odrzucające mają na celu ograniczenie przeuczenia sieci. Sieć jest dosyć głęboka, ale umożliwia wykrywanie cech o różnych rozmiarach.



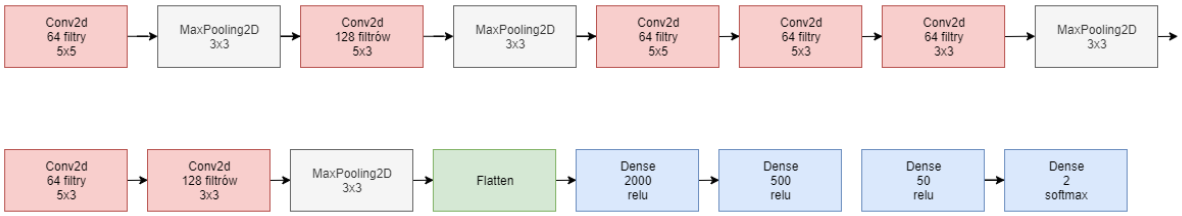
- Sieć 2
  Jako drugą sieć postanowiliśmy zastosować sieć dużo płytszą i prostszą niż pozostałe. Składa się z jedynie dwóch warstw konwolucyjnych i jednej agregującej. Warstwa perceptronowa jest też bardzo mała, gdyż składa się z jedynie 8 neuronów. Sieć ta ma bardzo niewiele wag, więc powinna uczyć się bardzo szybko i dobrze generalizować.



- Sieć 3

Ponieważ dla pierwszej sieci, większej od drugiej otrzymaliśmy lepsze wyniki, to stworzyliśmy jeszcze większą sieć. W stosunku do pierwszej sieci zwiększyliśmy ilość warstw konwolucyjnych, aby umożliwić wykrywanie większej ilości cech obrazu o różnych rozmiarach. Zauważyliśmy też, że cechy desek są bardziej rozciągnięte w

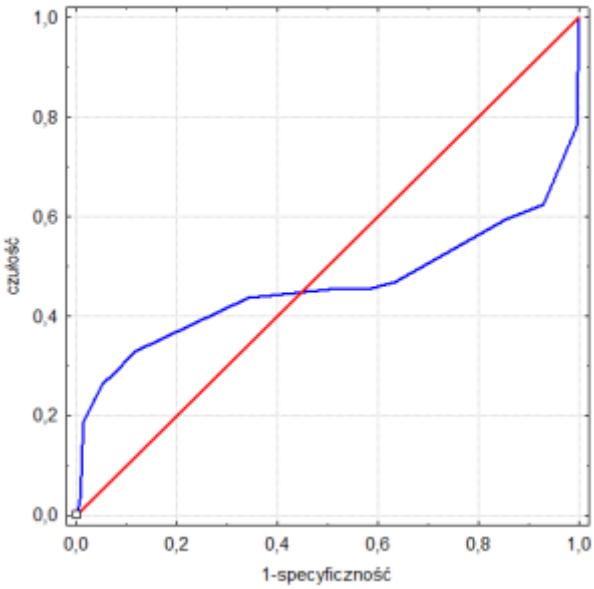
poziomie, więc zastosowaliśmy filtry które nie są kwadratowe, tylko prostokątne. Zrezygnowaliśmy też z warstw odrzucających wagi, aby zobaczyć jak sieć poradzi sobie bez nich. Zwiększyliśmy też rozmiary sieci perceptronowej, aby umożliwić realizację bardziej skomplikowanych warunków.



4. Wyznaczanie metryk i generacja wykresów.
   Każdy wytrenowany model przetestowaliśmy na zbiorze testującym. Na podstawie pewności sieci i rzeczywistych klas, korzystając z „scikit-learn" wyznaczyliśmy krzywą ROC i pole pod nią czyli AUCROC. Na podstawie klasyfikacji i rzeczywistych klas danych wygenerowaliśmy też macierze błędu („confusion matrix"), które w jasny sposób prezentują wyniki klasyfikacji.

   Krzywa ROC jest graficzną reprezentacją poprawności decyzji o sklasyfikowaniu danych przez zadany model. To czy model poprawnie klasyfikuje dane określa przede wszystkim położenie wykresu względem przekątnej y = x, gdyż odpowiada ona za losowy wybór klasy dla danego obiektu. Model klasyfikujący na podstawie np. wyuczenia będzie reprezentowany przez krzywą ponad przekątną, a jego maximum i stu procentową poprawność określa punkt (0 ; 1) na wykresie. Lepszą sprawność modelu określa krzywizna bliżej położona względem tego punktu.

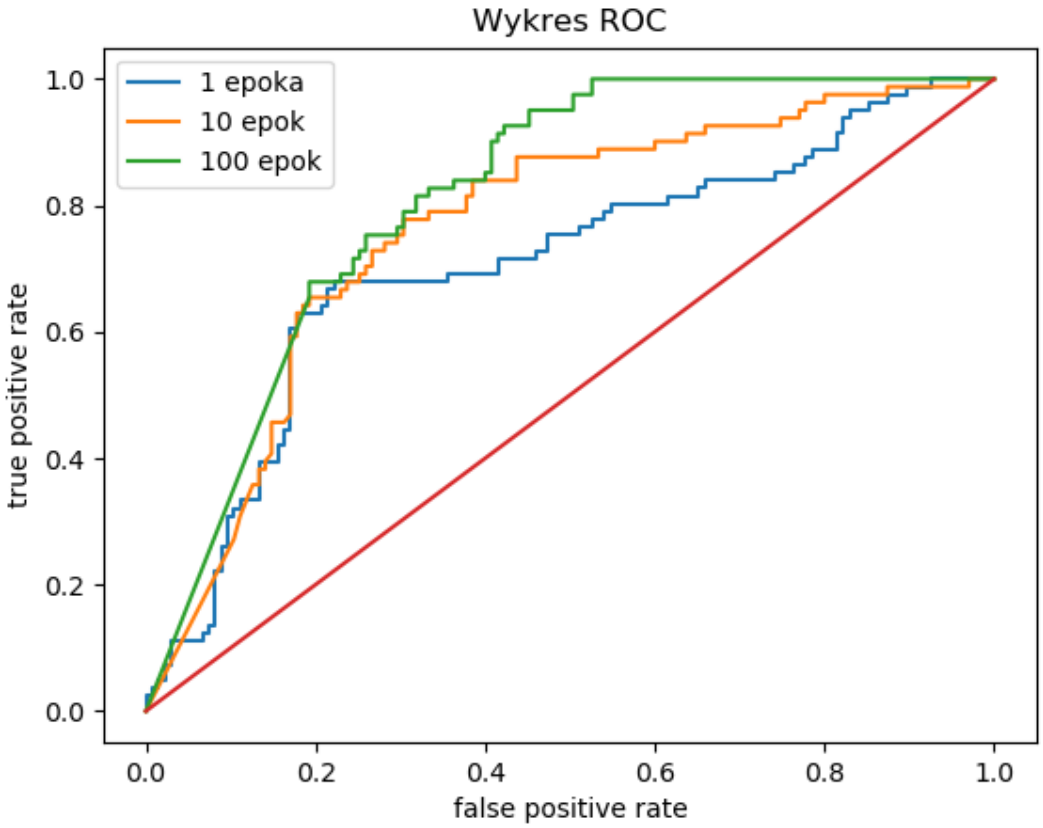
   Parametr AUCROC (Area Under Curve ROC) jest wyliczonym polem pod wykresem krzywej ROC, który wykorzystuje się jako miarę trafności danego modelu klasyfikacyjnego. Przez konstrukcję samej krzywej parametr ten przyjmuje wartości z przedziału [0 ; 1], a im wyższa jest jego wartość tym model uznaje się za lepszy w dziedzinie klasyfikacji danych. Trzeba jednak pamiętać, że samo pole pod krzywą nie jest daną wystarczającą do klasyfikacji jakości modelu i porównywania go z innymi, gdyż mogą zdarzyć się sytuacje gdzie mimo wysokiej wartości tego parametru klasyfikacja może być niepoprawna dla pewnej części zbioru.



Otrzymane rezultaty

Sieć 1.

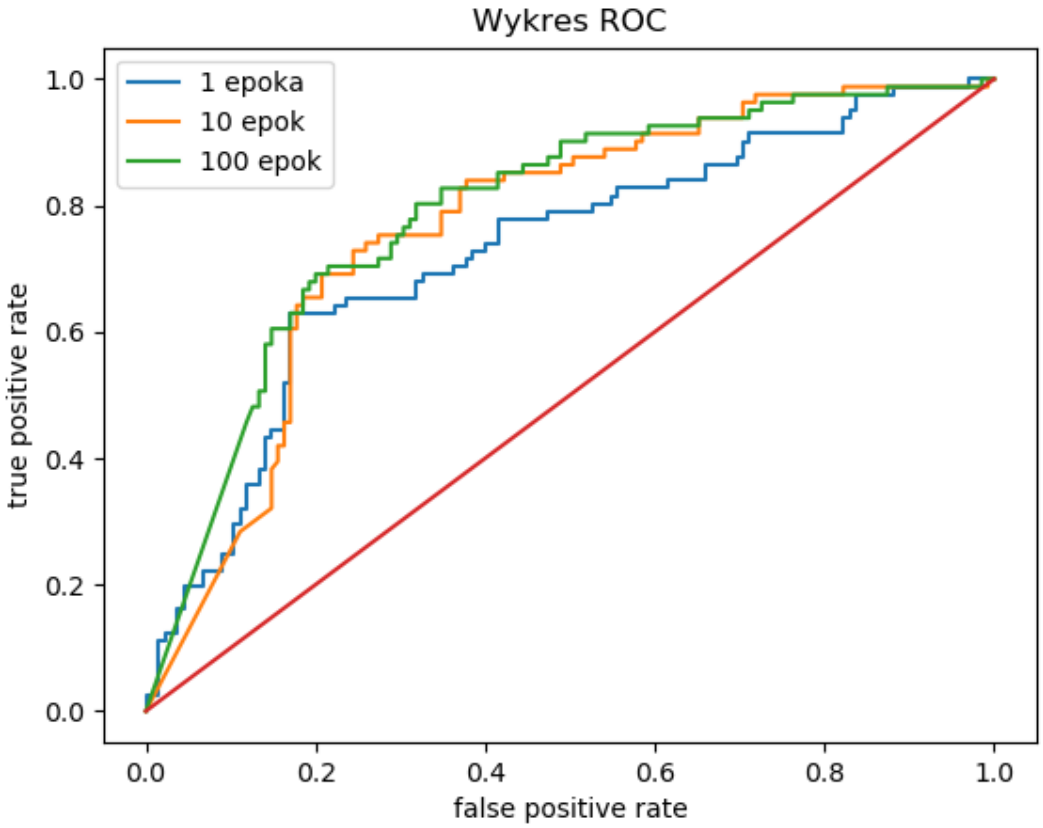
1 epoka		predicted	
AUCROC:	0.7058	accepted	rejected
actual	accepted	111	24
	rejected	32	49
10 epok		predicted	
AUCROC:	0.7641	accepted	rejected
actual	accepted	111	24
	rejected	30	51
100 epok		predicted	
AUCROC:	0.8143	accepted	rejected
actual	accepted	94	41
	rejected	18	63



Jak można zauważyć z wykresu krzywej ROC oraz wartości parametru AUCROC sieć polepsza swoją zdolność poprawnej klasyfikacji wraz ze zwiększaniem liczby epok uczących. Między kształtami krzywych i wartościami parametrów AUC występuję znaczne różnice, co może świadczyć o tym że dla 100 epok sieć wciąż nie osiągnęła swoich maksymalnych możliwości poprawnej klasyfikacji, jeżeli chodzi o tą metrykę. Tablica błędów natomiast pokazuje, że model sklasyfikował więcej przykładów źle dla okresu uczenia 100 epok. Możliwe jest, że model zaczął się zbytnio dostosowywać do danych treningowych (gdyż błąd na danych uczących nieustannie malał), i wystąpiło przeuczenie, a lepszy kształt krzywej ROC wynika z większej pewności decyzji klasyfikacyjnych.

Sieć 2.

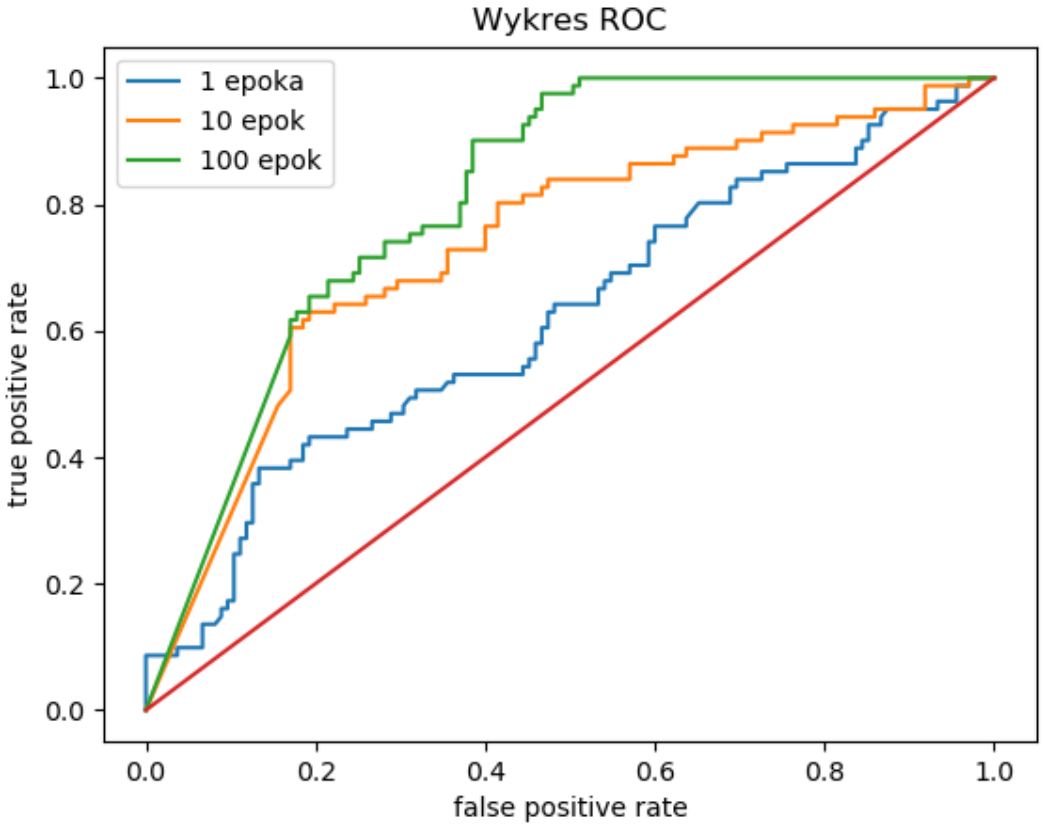
1 epoka		predicted	
AUCROC:	0.7358	accepted	rejected
actual	accepted	110	25
	rejected	30	51
10 epok		predicted	
AUCROC:	0.7666	accepted	rejected
actual	accepted	111	24
	rejected	30	51
100 epok		predicted	
AUCROC:	0.7902	accepted	rejected
actual	accepted	106	29
	rejected	24	57



Najmniejsza z trzech porównywanych sieci zaskakuje szybkością z jaką osiąga ona maksimum swoich możliwości. Wartości parametru AUCROC oraz kształty krzywych ROC dla wszystkich trzech liczb epok nie osiągają tak znaczących różnic jak chociażby w przypadku poprzedniego modelu. Świadczy to o dużej szybkości uczenia się sieci, co odbija się jednak kosztem gorszych wyników metryk ROC i AUCROC. Model ten za to nie przystosowuje się nadmiernie do danych, tak jak ma to miejsce w przypadku poprzedniego przypadku i ilość błędów klasyfikacji wyznaczona na podstawie tabeli błędu maleje wraz z czasem uczenia.

Sieć 3.

1 epoka		predicted	
AUCROC:	0.6198	accepted	rejected
actual	accepted	135	0
	rejected	81	0
10 epok		predicted	
AUCROC:	0.7359	accepted	rejected
actual	accepted	111	24
	rejected	32	49
100 epok		predicted	
AUCROC:	0.8113	accepted	rejected
actual	accepted	78	57
	rejected	8	73



Dla jednej epoki uczenia sieć ta często klasyfikowała wszystkie obrazy jako zaakceptowane, co zdażyło się też w tym przypadku. Świadczy to o niedouczeniu sieci. Dla największego z naszych modeli sieci krzywa ROC jak i parametr AUCROC mają największe dysproporcje dla wyników po procesie uczenia się przez 1 i 100 epok. Wynika to z faktu, że tak rozbudowana sieć po zaledwie jednej epoce jest zbyt losowa i jasno widać że potrzebuje ona dużo czasu by dawać zadowalające rezultaty, które jednak są jednak porównywalne z wynikami dla sieci pierwszej, co może świadczyć o osiągnięciu lub zbliżaniu się do maksimum osiągów dla danego zbioru uczącego. Jeżeli popatrzymy natomiast na tablicę błędu, to można dostrzec sytuację analogiczną jak w przypadku sieci 1 – błąd klasyfikacji dla 100 epok zwiększa się. Model jeszcze bardziej dostosowuje się do danych

trenujących, co może być spowodowane brakiem warstw odrzucających wagi, które pierwszy model posiada, i większym rozmiarem tego modelu.

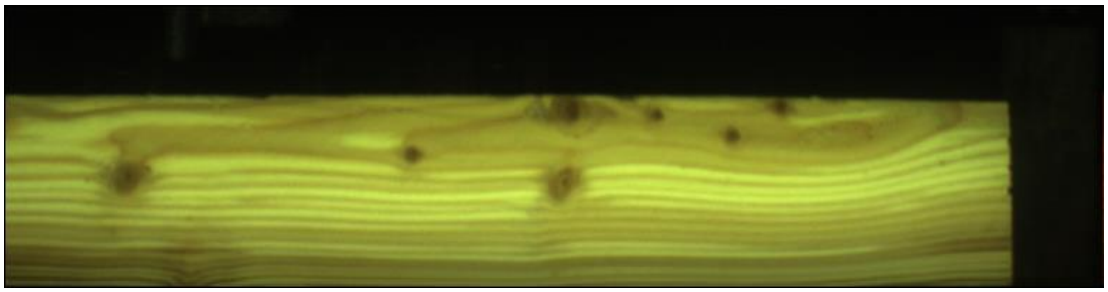
Podsumowanie.

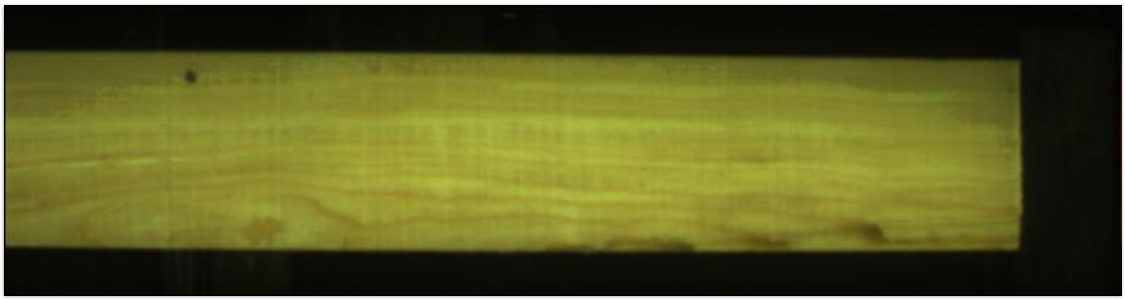
Najlepszy czas uczenia sieci wynosił 10 epok. Dla takiego okresu najwięcej danych zostało poprawnie sklasyfikowanych . Najlepsze wyniki osiągnęły sieci pierwsza i druga. Druga sieć była bardzo mała, więc według nas jest najlepsza.

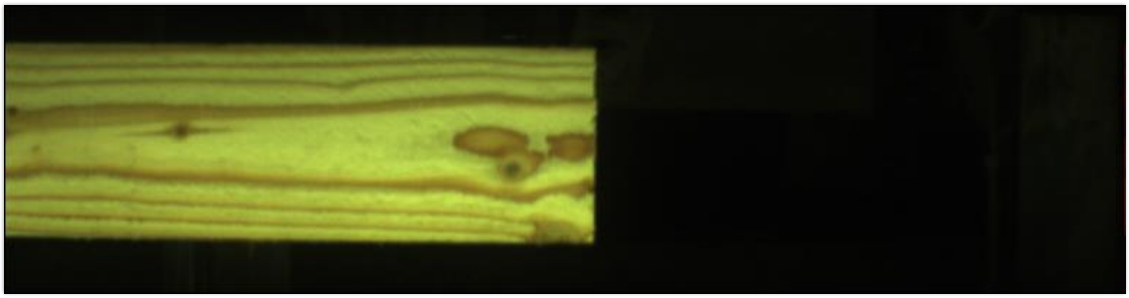
Klasyfikacja nieoznaczonych desek

Po stworzeniu i przetestowaniu modeli na oznaczonych danych, przeszliśmy do drugiej części zadania – klasyfikacji nieoznaczonych desek. Stworzyliśmy skrypt, który na podstawie danych w folderze unlabeled, klasyfikował je i przekopiowywał je do odpowiednich folderów odpowiadających kategoriom. Umożliwiało to łatwą prezentację danych i prostsze określenie zasad klasyfikacji sieci. Do klasyfikacji wykorzystaliśmy pierwszą sieć uczoną na 10 epokach, ponieważ ma stosunkowo mało parametrów, jednocześnie generując bardzo dobre wyniki.

Przykładowe deski oznaczone jako accepted:



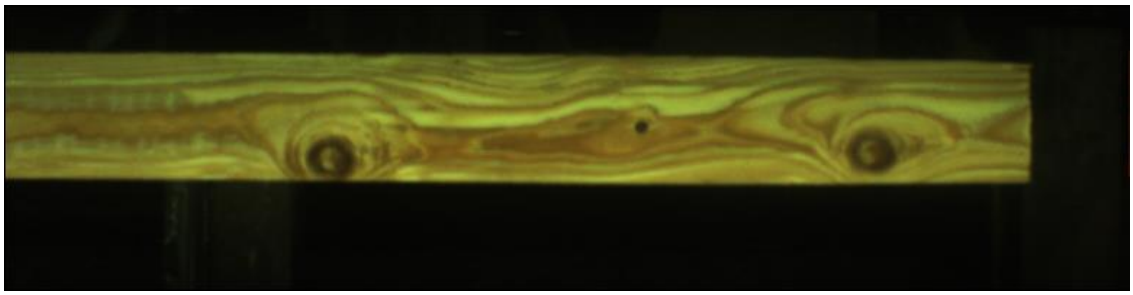


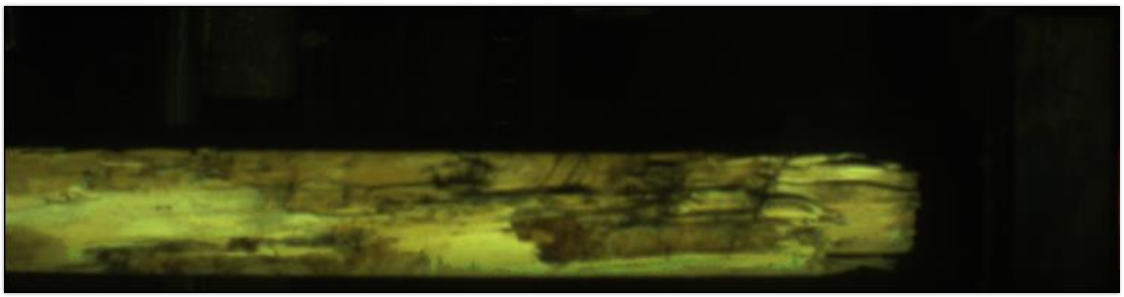


Klasyfikator jako accepted klasyfikował zazwyczaj deski, które:

- Były dłuższe niż połowa zdjęcia
- Były szerokie na co najmniej dwie trzecie zdjęcia
- Miały niedużo sęków
- Miały jasny, regularny kolor

Przykładowe deski oznaczone jako rejected:





Klasyfikator jako rejected klasyfikował zazwyczaj deski, które:

- Były krótsze niż połowa zdjęcia
- Były węższe niż dwie trzecie zdjęcia
- Miały bardzo dużo sęków
- Miały różne przebarwienia, koloru brązowego lub czarnego
- Były zaokrąglone, zamiast płaskie (Deska wycięta z brzegu pnia)
- Były uszkodzone, np. pęknięte, wyszczerbione, o nieregularnym kształcie

Wnioski i spostrzeżenia

1. Warto jest operować parametrem próby danych „batch size”, zwiększanie go pozwala znacznie szybciej trenować model, jednakże zbyt duża jego wartość może powodować błędy związane z ograniczoną ilością pamięci.
2. Mniejsze sieci dużo szybciej się uczą i lepiej generalizują, natomiast sieci bardziej rozbudowane są w stanie realizować bardziej skomplikowane warunki, ale muszą się dużo dłużej uczyć.
3. Większa sieć nie zawsze jest lepsza od mniejszej.
4. Nie wszystkie architektury sieci są dobre. Niektóre sieci, które trenowaliśmy nie potrafiły się nic nauczyć, albo utykały w lokalnych minimach funkcji celu.
5. Działanie nauczonego modelu w dużym stopniu zależy od danych trenujących.
6. Klasyfikacja dużych zdjęć RGB jest zadaniem bardzo trudnym nawet dla dużych i dobrych sieci.
7. Zbyt duże warstwy splotowe uniemożliwiały nam w niektórych przypadkach uczenie.
8. Otrzymany zbiór danych sklasyfikowanych był zbyt mało liczny dla potrzeb tego zadania.
9. Z tego, co zaobserwowaliśmy, dane w folderach accept i reject były do siebie bardzo podobne i osobiście niektóre deski przydzielilibyśmy do innych zbiorów. Sądzimy, że takie nakładanie się zbiorów danych może być dużą przeszkodą w klasyfikacji i uniemożliwiać otrzymanie lepszych wyników.
10. Warto wykorzystać wiele metryk do badania modelu. Pojedyncze niekoniecznie mówią o nim wszystko.
11. Zbyt długie uczenie sieci może skutkować przeuczeniem i większym błędem dla zbioru testującego.

Opinia o środowisku:

Wykorzystane środowisko było bardzo wygodne i znacznie ułatwiło nam pracę nad projektem, składało się z:

- Python 3.7 – Prosty, interpretowany język o wielkich możliwościach.
- Numpy - Bardzo wygodne operacje na wektorach i macierzach. Możliwość wyznaczania ważnych właściwości i wybierania elementów za pomocą bardzo prostego i intuicyjnego interfejsu.
- Tensorflow/Keras – wykorzystany do tworzenia modeli , ich uczenia i wykorzystania. Udostępnia gotowe warstwy i modele do klasyfikacji, co znacznie ułatwia pracę i umożliwia szybkie i bezproblemowe testowanie wielu różnych wersji modeli.
- Scikit-learn – wygodne tworzenie metryk na podstawie rezultatów.
- Matplotlib – umożliwił nam bardzo szybkie i proste tworzenie czytelnych i intuicyjnych wykresów.