Konwolucyjne sieci neuronowe w klasyfikacji emocji

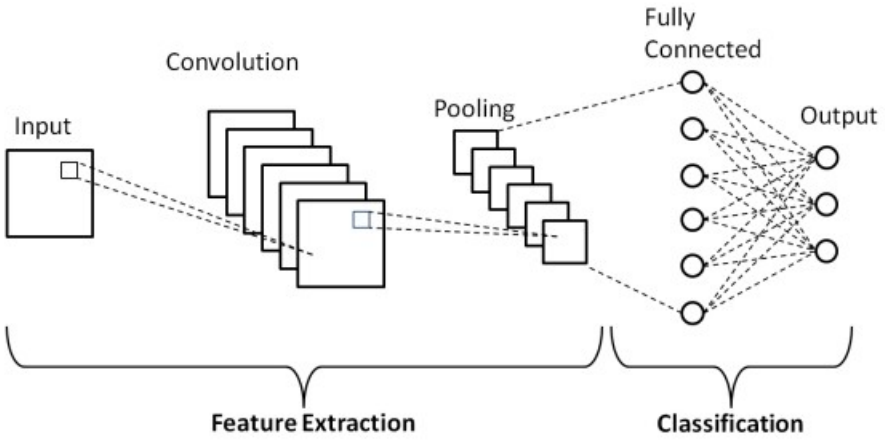
Jakub Bednarz
Krzysztof Madajczak
Mateusz Strzelecki

1. Opis problemu

Przedmiotem projektu była implementacja i analiza konwolucyjnej sieci neuronowej klasyfikującej ludzkie emocje na podstawie obrazów, a także porównanie jej skuteczności z alternatywnymi podejściami, takimi jak inne architektury sieci neuronowych oraz rozwiązania typu transfer learning. Za punkt wyjścia przyjęto uczenie nadzorowane na podstawie wybranego zbioru treningowego FER2013 [1], przedstawiającego zbliżenia na twarze. Zadaniem modelu była klasyfikacja przedstawianych emocji do 7 klas: złości, obrzydzenia, strachu, szczęścia, smutku, zaskoczenia, neutralności. Na tej podstawie stworzony został graficzny interfejs użytkownika umożliwiający wykrywanie twarzy na załadowanych zdjęciach oraz klasyfikację emocji każdej osoby na zdjęciu.

2. Teoretyczny opis użytych metod

Konwolucyjne sieci neuronowe (CNN) są rodzajem sztucznych sieci neuronowych uczenia głębokiego, które zostały zaprojektowane do przetwarzania danych przestrzennych, takich jak obrazy. Podstawą działania CNN są specjalne operacje matematyczne zwane konwolucjami, efektywnie ekstrahujące cechy z obrazów. Wykorzystują one małe macierze (tzw. filtry) do przesuwania się po danych wejściowych. Operacja konwolucji polega na mnożeniu elementów macierzy filtra z odpowiednimi elementami macierzy danych wejściowych, a następnie sumowaniu wyników. Proces ten jest powtarzany dla wszystkich lokalnych obszarów danych wejściowych, tworząc tzw. mapę cech.
Na podstawie odpowiednio złożonych map cech (uzyskanych po wielu warstwach) możliwa jest klasyfikacja obrazów, realizowana przez gęste warstwy w pełni połączone sieci neuronowej.



Rys. 1: Schematyczna architekura klasyfikacji CNN [2]

Kluczowym elementem w konstrukcji CNN jest hierarchiczność następujących po sobie warstw konwolucyjnych, z których każda ma swoje zadanie. Pierwsze warstwy

wykorzystują filtry do wykrywania podstawowych cech, takich jak krawędzie, tekstury i podstawowe wzorce. Kolejne warstwy koncentrują się na coraz bardziej skomplikowanych i abstrakcyjnych cechach, korzystając z wyników poprzednich warstw. Ważne dla efektywnego działania CNN są warstwy poolingowe, które redukują rozmiar map cech, wybierając najważniejsze informacje (najskuteczniejsze są w tym informacje o maksimum cechy).
Konwolucyjne sieci neuronowe są skuteczniejsze w pracy z obrazami niż inne architektury sieci neuronowych ze względu na kilka czynników. Po pierwsze, wykorzystują one lokalne zależności przestrzenne, a więc różne cechy mogą być poprawnie wykrywane niezależnie od siebie. Dzięki temu sieci konwolucyjne są odporne na translacje, co znacznie ułatwia uczenie. Ponadto, wykorzystywanie tego samego filtra na różnych obszarach obrazu znacznie zmniejsza liczbę parametrów do nauki, co sprawia, że sieci są bardziej efektywne obliczeniowo. Występujące po warstwach konwolucyjnych warstwy poolingowe zmniejszają rozmiar danych, zachowując jednocześnie najważniejsze informacje, co pomaga w redukcji overfittingu i poprawie ogólnej wydajności sieci.
Takie podejście pozwala na automatyczne budowanie hierarchii często trudnych do przewidzenia cech obrazu. Wykorzystując wiele warstw, sieci konwolucyjne są w stanie nauczyć się coraz bardziej abstrakcyjnych reprezentacji obrazu, co prowadzi do lepszych wyników w zadaniach takich jak detekcja czy klasyfikacja obrazów.

Jedną z możliwych metod doboru hiperparametrów modelu jest metoda symulowanego wyżarzania. Polega ona na losowych perturbacjach przestrzeni hiperparametrów i akceptowaniu tych które poprawiają działanie modelu. Możliwe jest także akceptowanie parametrów które nie poprawiają funkcji kosztu, w celu uniknięcia utknięcia w lokalnym optimum i szukania lepszych rozwiązań globalnych. Prawdopodobieństwa takiej akceptacji (szerokiego przeszukiwania przestrzeni hiperparametrów) jest określane przez malejącą z czasem temperaturę, na wzór wyżarzania w metalurgii. W dalszych iteracjach bardziej prawdopodobne jest zatem stopniowe doprecyzowanie wartości w ograniczonych przedziałach. Metoda ta nie daje gwarancji znalezienia optymalnego rozwiązania, ale w przeciwieństwie np. do metody Grid Search (sprawdzającej wszystkie kolejne kombinacje) nie wymaga tak ogromnych zasobów obliczeniowych.

Bardzo efektywną metodą uzyskania skutecznego modelu jest tzw. transfer learning, polegający na wykorzystaniu dostępnych bardzo zaawansowanych modeli, trenowanych do rozwiązywania zbliżonego problemu. Istniejące modele, przeważnie bardzo długo uczone na ogromnych zbiorach danych, są adaptowane do nowego zadania poprzez dostosowanie ich wag podczas dodatkowego treningu na docelowym zbiorze. Transfer learning pozwala osiągać świetne wyniki nawet w przypadku nielicznych danych treningowych oraz przyspiesza proces uczenia dzięki wykorzystaniu istniejącej wiedzy.

3. Opis realizacji zadania

Podstawowych narzędziem użytym w projekcie była biblioteka pytorch, pozwalająca na łatwą implementację sieci neuronowych. W celu uzyskania adekwatnych możliwości obliczeniowych przydatne były narzędzia cuda oraz platforma kaggle, umożliwiające wykorzystanie kart graficznych oraz obliczeń w chmurze do przyspieszenia obliczeń. Wykorzystywane zostały również pakiety numpy, torchvision, sklearn, opencv oraz matplotlib.

3.1. Implementacja architektury CNN

Pierwszym etapem było stworzenie konwolucyjnej sieci neuronowej dla analizowanego

problemu klasyfikacji wieloklasowej. Przyjęta została następująca architektura:

- łącznie 6 warstw konwolucyjnych, hierarchicznie wykrywających coraz większe struktury na zdjęciu
- po każdej warstwie konwolucyjnej następuje warstwa aktywacji ReLu, która wprowadza nieliniowość do modelu
- warstwy normalizacji wsadowej stabilizujące model po warstwie aktywacji poprzez redukcje wewnętrznych zmian kowariancji
- co dwa bloki warstw konwolucyjnych stosowana jest warstwa max pooling okna 2x2, która zmniejsza wymiary przestrzenne danych wejściowych, wybierając maksymalną wartość w obrębie okna poolingowego
- warstwy dropout, losowo ustawiające część jednostek wejściowych na zero podczas treningu, co pomaga zapobiegać przeuczeniu przez redukcję współzależności pomiędzy neuronami
- warstwy w pełni połączone zastosowane w celu klasyfikacji emocji na podstawie wykrytych cech

Danymi dla sieci były poddane podstawowej obróbce zdjęcia z podstawowego zbioru FER2013 (około 28000 zdjęć treningowych i 7000 testowych), z wydzielonymi 20% danych treningowych na dane walidacyjne wykorzystywane do oceny modelu podczas nauki.

3.2. Poprawa skuteczności modelu: augmentacja danych, dobór hiperparametrów

Dla poprawy skuteczności modelu kluczowe było przede wszystkim rozszerzenie zbioru treningowego poprzez augmentację danych. Pozwala ona na generalizację modelu, przez co może on sobie lepiej radzić z nieznanymi danymi testowymi. W tym celu zbiór został rozszerzony o odbicia lustrzane oraz zdjęcia poddane rotacji. Zbiór testowy pozostał niezmieniony. W ten sposób liczba dostępnych danych treningowych została zwiększona z około 28 tysięcy do ponad 114 tysięcy obrazów.

W celu poprawy doboru stałej uczenia, liczby epok oraz rozmiaru wsadu zastosowana została metoda symulowanego wyżarzania. Dla pierwszych dwóch parametrów określone zostały przedziały poszukiwań oraz wielkość możliwego kroku. Rozmiar wsadu był losowany z zbioru możliwych wartości. Ze względu na możliwości obliczeniowe, liczba iteracji została określona na 30.

3.3. Porównawczy model feedforaward

3.4. Zastosowanie tranfer learningu

3.5. Stworzenie interfejsu użytkownika

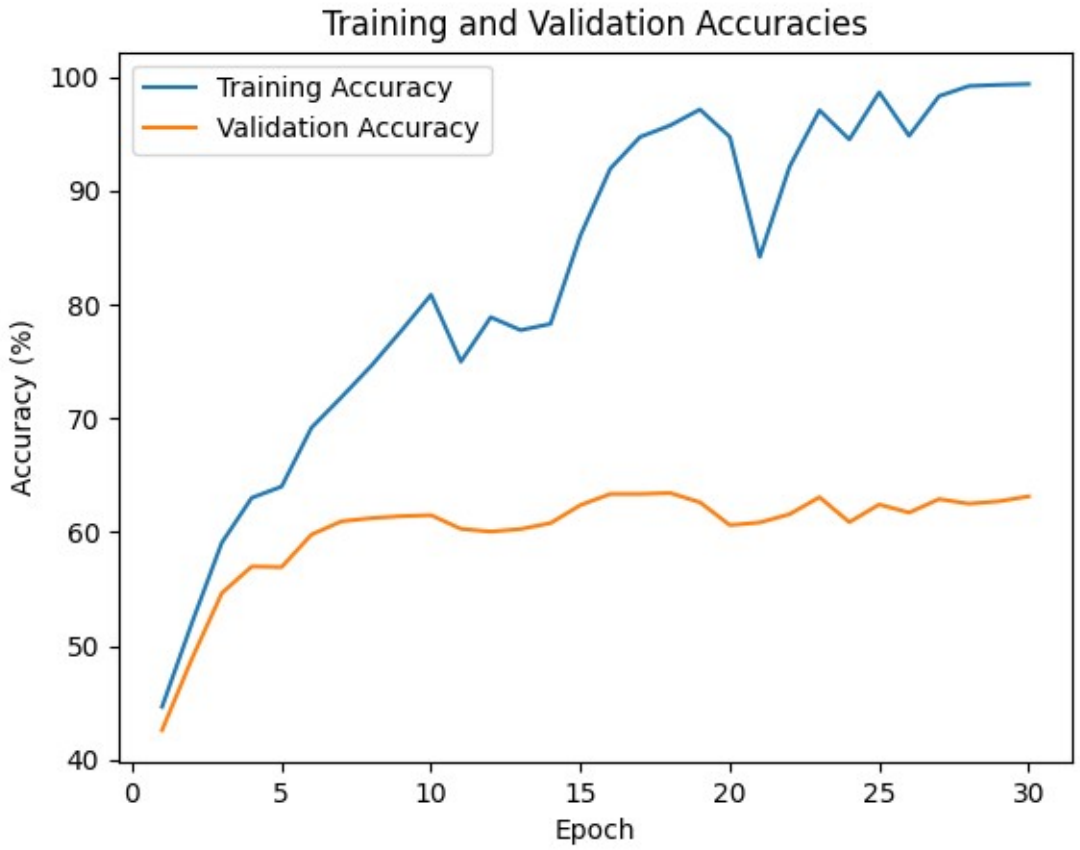
// opencv w obrobce grafiki
// implementacja kaskady haara
// algorytm wykrywania twarzy (twarz + oczy)

4. Prezentacja osiągniętych wyników

4.1. Implementacja architektury CNN

Podczas treningu sieci w każdej epoce mierzona była skuteczność aktualnego modelu na

zbiorze treningowym i walidacyjnym. Przebieg tych wartości przedstawia Wyk.1:

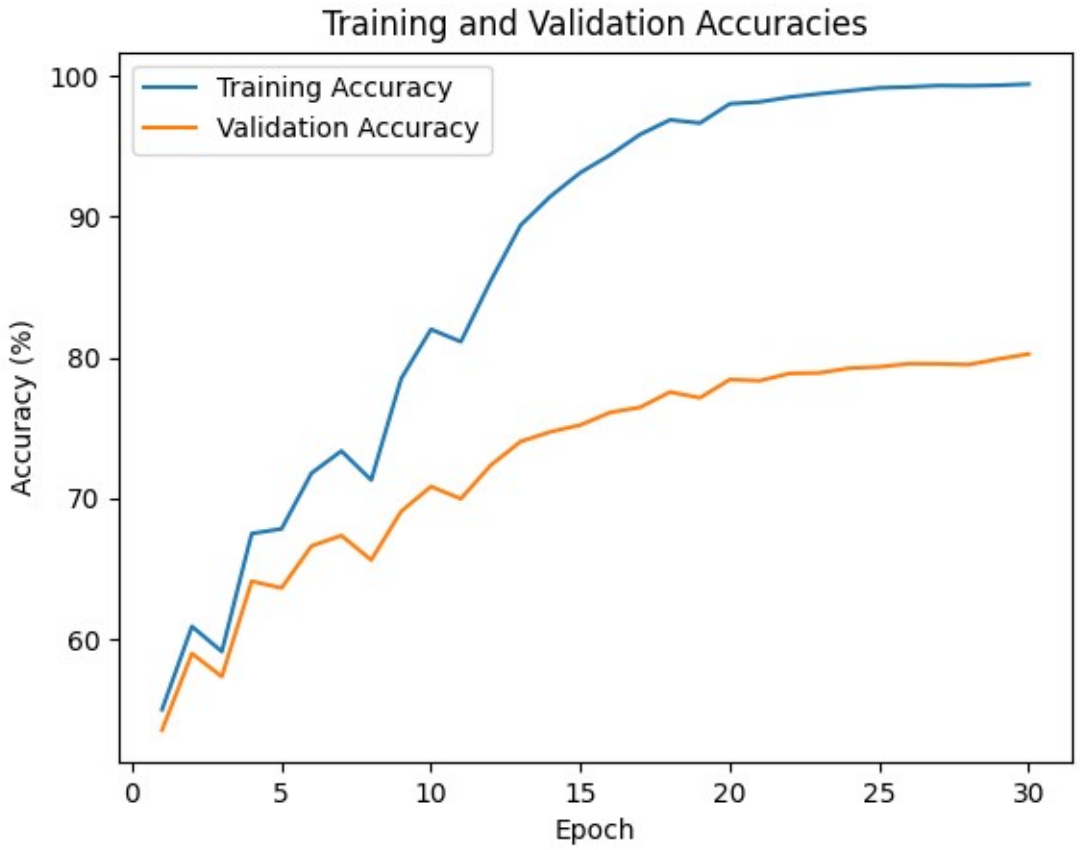


Wyk. 1: Wykres dokładności treningowej i walidacyjnej podstawowego modelu

Uzyskany model uzyskał skuteczność testową na poziomie 63,78%.
Obliczone zostały także podstawowe metryki przewidywań modelu dla każdej klasy: macierz konfuzji, precyzja, odzysk oraz wynik F1, jak również ich średnie ważone. Należy odnotować, że wszystkie średnie ważone okazały się być bardzo zbliżone [6].

4.2. Poprawa skuteczności modelu: augmentacja danych, dobór hiperparametrów

Analogiczne działania dla poprawionego modelu pokazują postęp modelu:



Wyk. 2: Wykres dokładności treningowej i walidacyjnej modelu ulepszonego

Poprawiony model uzyskał skuteczność testową na poziomie 68,61%.
Po wykonaniu dodatkowych porównawczych, liczba epok trenowania modelu została podniesiona względem wartości wskazywanej przez wyżarzanie.
Wszystkie uwzględnione metryki wykazały podobne zachowanie jak dla modelu podstawowego, z odpowiednio wyższymi wartościami [10].

4.3. Porównawczy model feedforaward

4.4. Zastosowanie tranfer learningu

4.5. Stworzenie interfejsu użytkownika

5. Dyskusja

Uzyskiwane przez nasze trzy modele skuteczności są zgodne z tym czego należało się spodziewać na podstawie obecnego stanu wiedzy i osiągnięć [5] odnośnie przedstawionego problemu.
Podstawowy wynik konwolucyjnej sieci neuronowej na poziomie nieznacznie przekraczającym 63% jest znacznie lepszy niż wynik uzyskanego przykładową inną siecią neuronową, która była tutaj sieć typu feedforaward. Dobrze ilustruje to przewagi sieci konwolucyjnych nad innymi architekturami, opisane w części teoretycznej. Głównym czynnikiem, który był w stanie wydatnie poprawić skuteczność sieci CNN była

augmentacja danych treningowych. Pozwoliła ona na lepszą generalizację obserwacji cech, co poprawiło wyniki na nieznanych danych. Postęp modelu dobrze widać po o wiele dłuższym utrzymywaniu rosnącej tendencji przez krzywą skuteczności na zbiorze walidacyjnym niż miało to miejsce dla podstawowego modelu.
Największą trudnością w dalszej poprawie skuteczności sieci były ograniczone możliwości obliczeniowe. Najbardziej widoczne było to w przypadku problemu doboru hiperparametrów. Ze względu na ogromną złożoność obliczeń bezpośredniego przeszukiwania przestrzeni możliwości (Grid Search), zdecydowaliśmy się na implementację metody symulowanego wyżarzania. Również tutaj jednak, ze względu na ograniczenie ilości iiteracji, pula dopasowywanych hiperparametrów była dość mała. W związku z powyższym, wiele czynników (ilość filtrów poszczególnych warstw, rozmiary jądra, dropouty, metoda aktywacji) musiało być dobranych heurystycznie, na podstawie typowo stosowanych wartości [3][7].
Niemniej jednak, osiągnięty wynik na poziomie 68,61% nie jest odległy od najlepszych osiąganych przez CNN na tym zbiorze rezultatów [3][8][9], mieszczących się w przedziale 71-75%.
Warto tutaj odnotować, że jest to dość wymagający zbiór danych, a ludzka zdolność do klasyfikacji emocji z obrazów FER2013 jest szacowana jedynie na 65±5% [8][9].

Lepsze rezultaty są możliwe do osiągnięcia korzystając z metod transfer learningu.
// dyskusja do transfer learningu
// dyskusja do wykrywania twarzy i zastowania modeli w interfejsie

Bibliografia:

[1] dane na podstawie: https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/data
[2] źródło schematu: https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909
[3] Deep Learning Praca z językiem Python i biblioteką Keras, François Chollet, Helion 2019
[4] https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/
[5] https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/overview
[6] obliczone metryki znajdują się w pliku logs/metrics/basic_cnn_metrics.txt
[7] https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
[8] https://arxiv.org/ftp/arxiv/papers/2105/2105.03588.pdf
[9] http://cs230.stanford.edu/projects_winter_2020/reports/32610274.pdf
[10] obliczone metryki znajdują się w pliku logs/metrics/final_cnn_metrics.txt