

Wydział Elektryczny

Politechnika Warszawska

*Faculty of Electrical Engineering
Warsaw University of Technology*

Projekt indywidualny

Zastosowanie modeli uczenia maszynowego opartych o transformery do zadań klasyfikacji obiektów oraz zbadanie różnic między nimi

Opiekun: dr inż. Zuzanna Krawczyk

Paweł Kapela
Informatyka Stosowana, studia dzienne, 4. semestr
nr indeksu 325482

Rozdział 1

Wstęp

1.1 Zakres pracy

W niniejszej pracy zamierzam skupić się na zaimplementowaniu oraz porównaniu dokładności i szybkości działania modeli widzenia komputerowego opartych o splotowe sieci neuronowe (CNN) oraz transformery.

Porównania dokonam w kontekście zadania klasyfikowania znaków drogowych. Wybrałem do tego następujące architektury:

1. splotowe sieci neuronowe:
 - a. architektura ResNet-34,
 - b. architektura ResNet-50.
2. transformer wizji (Vision Transformer, ViT).¹

Moim celem jest również bliższe przyjrzenie się sposobowi działania wykorzystanych modeli oraz stojącymi za nimi pomysłami. Postaram się więc również zwizualizować ich działanie w różnych etapach wnioskowania.

1.2 Wykorzystane zestawy danych

1.2.1 Kryteria doboru zestawów danych

Ze względu charakterystykę zadania klasyfikacji, zdecydowałem się na wykorzystanie dostosowanego do niego zestawu danych. Oznacza to wybranie zestawu danych ze zdjęciami obejmującymi możliwie tylko dany znak drogowy.

Wybierając zestaw danych do testów kierowałem się następującymi kryteriami:

- możliwie jak największa liczba różnych typów znaków,
- rozsądna (wystarczająca do wytrenowania/dotrenowania modelu, niezbyt duża, aby ograniczyć koszt obliczeniowy) rozdzielcość próbek,
- zawartość zdjęć wykonanych pod różnym kątem,

- zawartość zdjęć wykonanych w różnych warunkach oświetleniowych (brak/nadmierne oświetlenie).

1.2.2 Wybrane zestawy danych

Zdecydowałem się na wykorzystanie zbioru niemieckich znaków drogowych opracowanych przez Instytut Neuroinformatyki Uniwersytetu Ruhry w Bochum — GTSRB (German Traffic Sign Recognition Benchmark).⁵

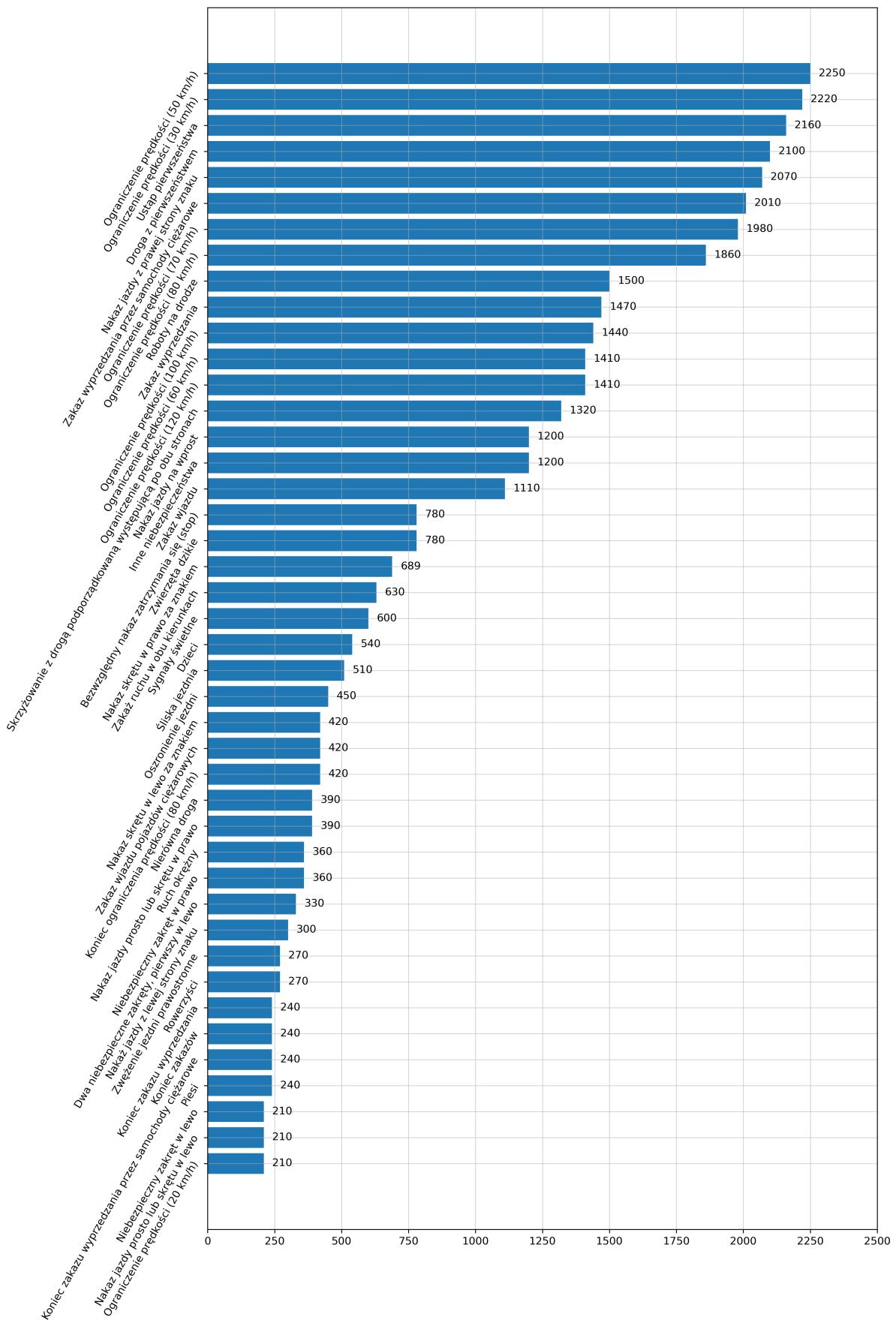
GTSRB

Zestaw ten zawiera łącznie 51840 próbek - 39209 w zestawie uczącym i 12630 w zestawie testowym. Próbki reprezentują 43 różne znaki drogowe, przykład dla każdego z nich jest przedstawiony poniżej.

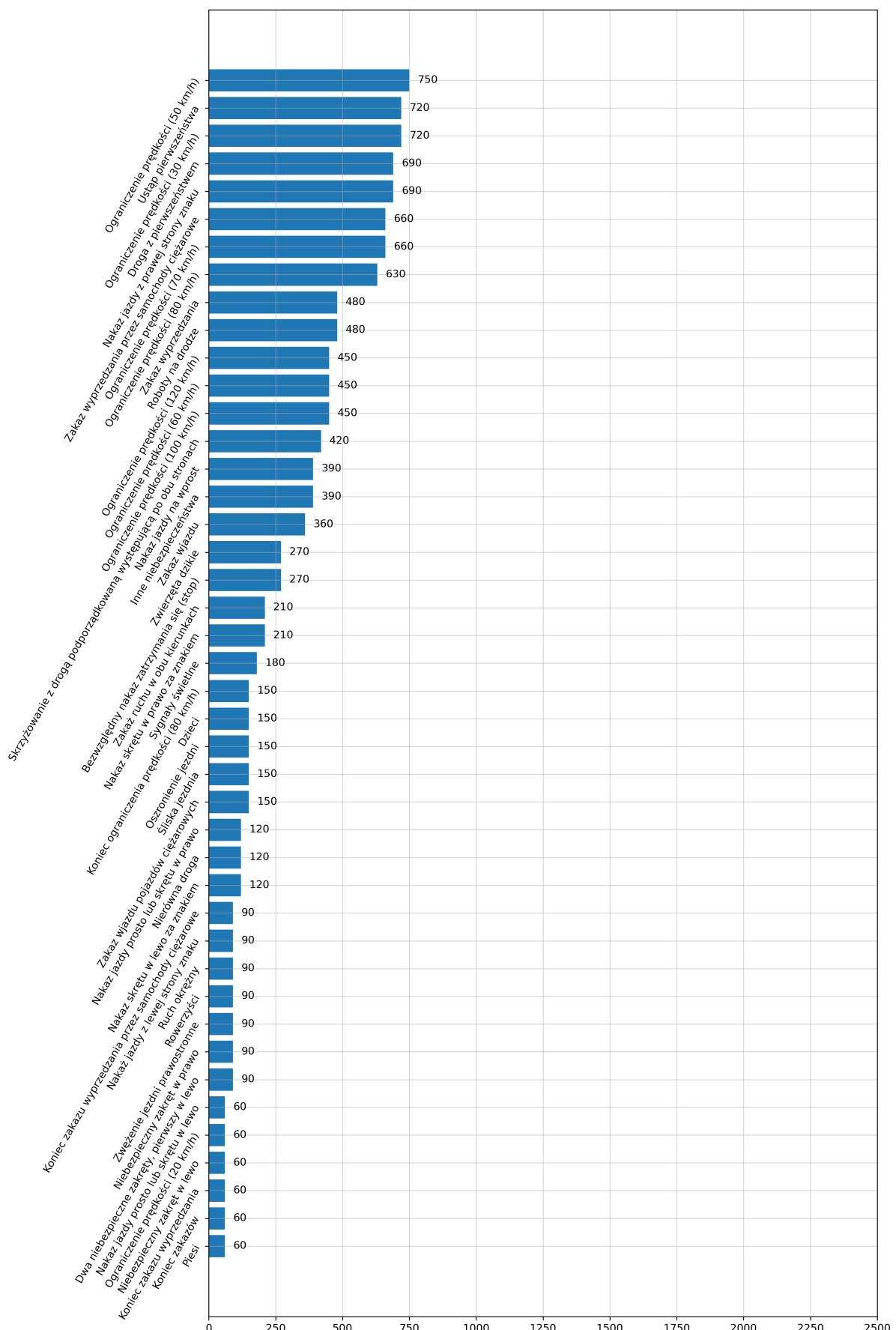


Rysunek 1.1: Przykładowe próbki ze zbioru GTSRB

Jak widać powyżej, zestaw ten zawiera również “poruszone” zdjęcia, co również jest użytecznym przypadkiem do trenowania/testowania.

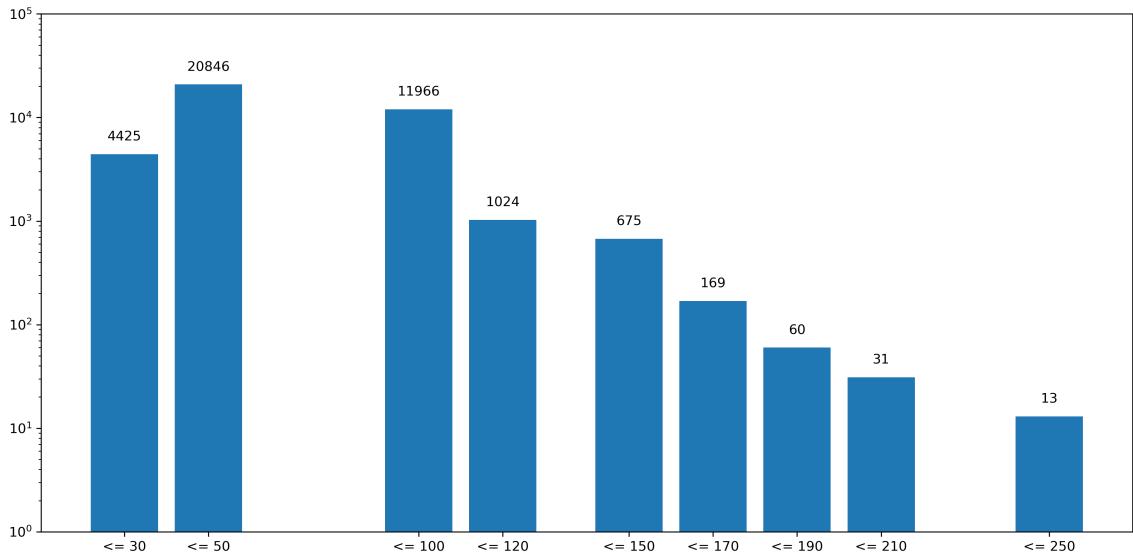


Rysunek 1.2: Liczebność próbek w poszczególnych klasach zbioru uczącego GTSRB

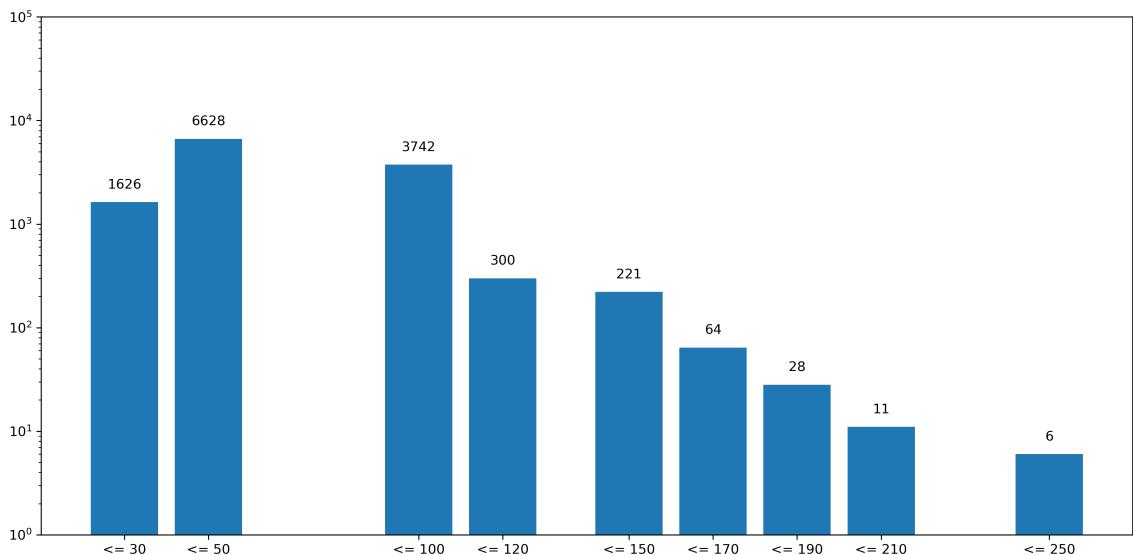


Rysunek 1.3: Liczebność próbek w poszczególnych klasach zbioru testowego

Rozkłady klas w obydwu zbiorach są do siebie bardzo zbliżone, dzięki czemu sprawdzenie dokładności modelu na zbiorze testowym będzie wiarygodne. Widać również dużą różnicę pomiędzy liczebnościami poszczególnych klas.



Rysunek 1.4: Wykres kubełkowy szerokości próbek w zbiorze uczącym



Rysunek 1.5: Wykres kubełkowy szerokości próbek w zbiorze testowym

GTSRB zawiera duży przekrój rozdzielczości i proporcji wymiarów zdjęć, nie są one jednak duże. Konieczne będzie więc skalowanie i przycinanie zdjęć do konkretnego wymiaru, aby dostosować je do używanego klasyfikatora. Bardzo niska rozdzielczość części próbek w zestawie danych nie stanowi problemu, ponieważ wiele rzeczywistych zdjęć do sklasyfikowania może powstać w niekorzystnych warunkach. W związku z tym oraz wymiarami obrazów akceptowanych przez modele wstępnie wytrenowane na zestawie zdjęć ImageNet (np. model ViT od Google²), zdecydowałem się na przeskalowanie i przycięcie wszystkich próbek do wymiarów 224 na 224 piksele.

Rozdział 2

Zadanie klasyfikacji obiektów

2.1 Przygotowanie zestawu danych

Pierwszym zadaniem była implementacja wczytywania i wstępnego przetwarzania przykładow. Próbki zbioru uczącego (w przeciwieństwie do próbek zbioru testowego) są poszgregowane w katalogach odpowiadającym poszczególnym klasom. Każdy katalog ze zdjęciami znaków zawiera plik CSV, który wiąże m. in. klasę, do której faktycznie dany przykład należy z nazwą pliku.

Zanim przystąpiłem do pisania odpowiedzialnego za to kodu w bibliotekach TensorFlow oraz PyTorch, usunąłem niepotrzebne poprzedzające zera z nazw katalogów (np. 00037 → 37) oraz przekonwertowałem zdjęcia z formatu PPM na PNG za pomocą prostej pętli w powłoce Fish wywołującej program ImageMagick. Konwersja ta była konieczna z powodu braku obsługi tego formatu przez funkcje wczytywania danych w TensorFlow.

Po wczytaniu zdjęć najpierw przeskalowałem zdjęcia tak, aby ich mniejszy bok miał długość 224 pikseli. Następnie przyciąłem zdjęcie do wymiarów 224 na 224 piksele, zachowując środek zdjęcia przed przycięcia. Chcę w ten sposób zapobiec utracie proporcji przy skalowaniu próbek.

Ostatnim etapem była normalizacja wartości RGB odpowiadających poszczególnym pikselom. W przypadku sieci splotowej ograniczyło się do zwykłego przeskalowania do zakresu od 0 do 1. Natomiast w przypadku transformera wizji konieczna okazała się normalizacja z wykorzystaniem wartości średniej oraz odchylenia standardowego użytych do przetwarzania wstępnego przy uczeniu na danych z ImageNet.

Największym wyzwaniem na tym etapie okazało się wydzielenie zbioru walidacyjnego tak, aby zachować identyczny stosunek pomiędzy liczebnością poszczególnych klas. W przypadku obu wykorzystywanych bibliotek sprowadziło się to do ręcznego (dodatkowa pętla) przeliczenia liczebności klas oraz połączenia odpowiednio podzielonych zbiorów.

Widoczną zaletą biblioteki TensorFlow pod tym względem jest zawarcie całej funkcjonalności wczytywania i przetwarzania próbek tak, aby napisany kod mógł zostać wykorzystany również na innych platformach przez nią wspieranych. Z kolei PyTorch jest bardziej przejrzysty i intuicyjny.

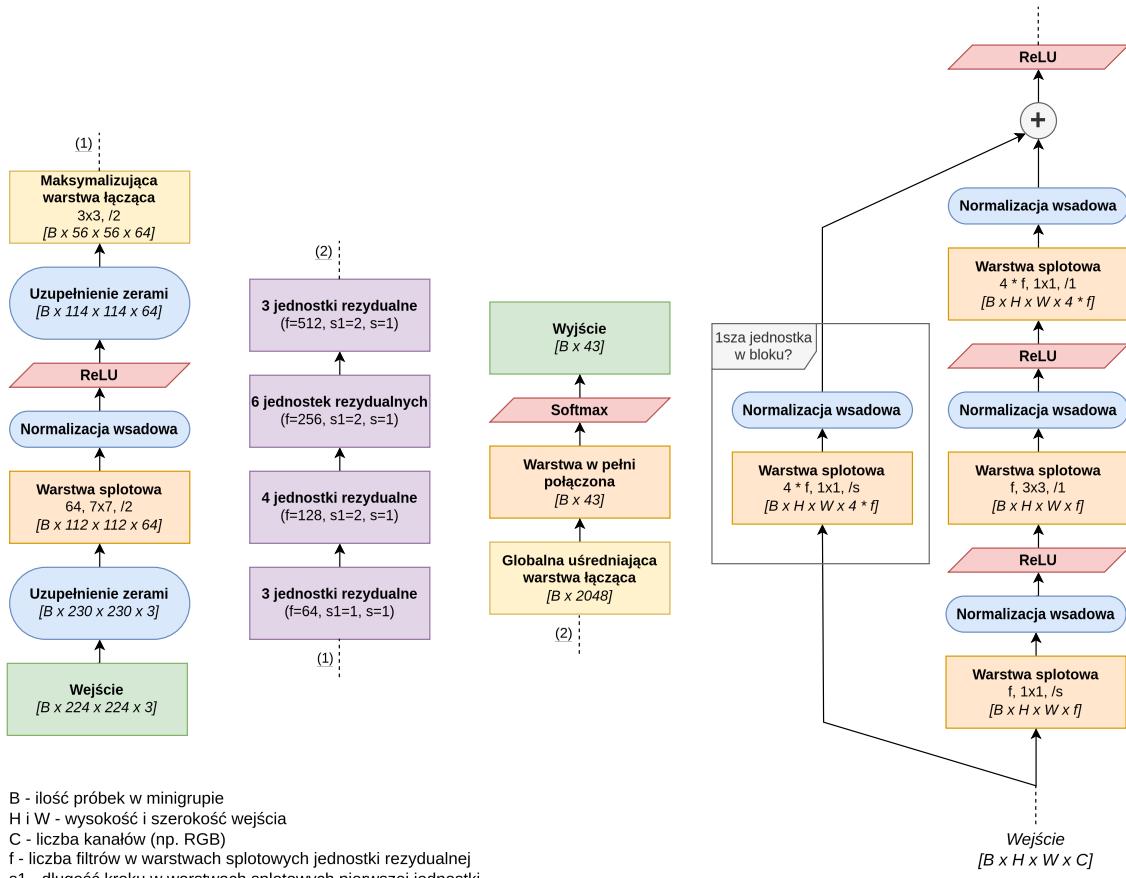
2.2 Splotowa sieć neuronowa

Model CNN do testów wybrałem w oparciu o zestawienie poszczególnych architektur i wiele zawartą w rozdziale 14. książki Auréliena Gérona³ oraz zestawienie modeli, które uzyskały pierwsze miejsca w konkursie ILSVRC w latach 2012-2017.⁴

Należy pamiętać, że biorąc udział w tym konkursie modele są bardzo rozbudowane ze względu na obecność 1000 klas w jego zestawie danych oraz dostępność ponad miliona próbek w zestawie uczącym. Nie chcę wybierać zbytnio zaawansowanego modelu, aby ograniczyć ryzyko przetrenowania oraz zapotrzebowanie na moc obliczeniową.

Ostatecznie zdecydowałem się na **ResNet-50**.

2.2.1 Opis architektury



Rysunek 2.1: Schemat modelu ResNet-50 (zawarty w bibliotece Keras)

Jest to głęboka sieć splotowa zawierająca połączenia skrótowe, które dodają sygnał wejściowy do wyjścia danej jednostki. Rozwiążanie to usprawnia proces uczenia w dwojakim sposobie. Po pierwsze, zamiast początkowo generować wartości bliskie zeru, modeluje ona funkcję tożsamościową. Przyspiesza to znacznie uzyskanie zbieżności, jeśli funkcja docelowa dla modelu jest bardzo podobna do tożsamościowej. Po drugie, sygnał może o wiele

łatwiej rozprzestrzenić się po całej sieci. Jest to pomocne w sytuacjach, gdyby niektóre warstwy nie zdążyły do pewnego momentu rozpoczęć dostosowywania swoich wag i tym samym blokowały propagację wsteczną.³

W jednostce rezydualnej, w której następuje zmiana liczba map cech oraz dwukrotne zmniejszenie wymiarów tensora wejściowego, bezpośrednie połączenie pomijające zastąpione jest warstwą splotową. Służy to dopasowaniu wymiaru danych wejściowych do zsumowania z wyjściem jednostki rezydualnej.

Schemat widoczny na rysunku 2.1 został sporządzony w oparciu o analizę kodu źródłowego modułu `tensorflow.keras.applications.ResNet50`, schemat wygenerowany przez funkcję `tensorflow.keras.utils.plot_model` oraz informacje zawarte w książce.³

2.2.2 Uczenie modelu

Od zera

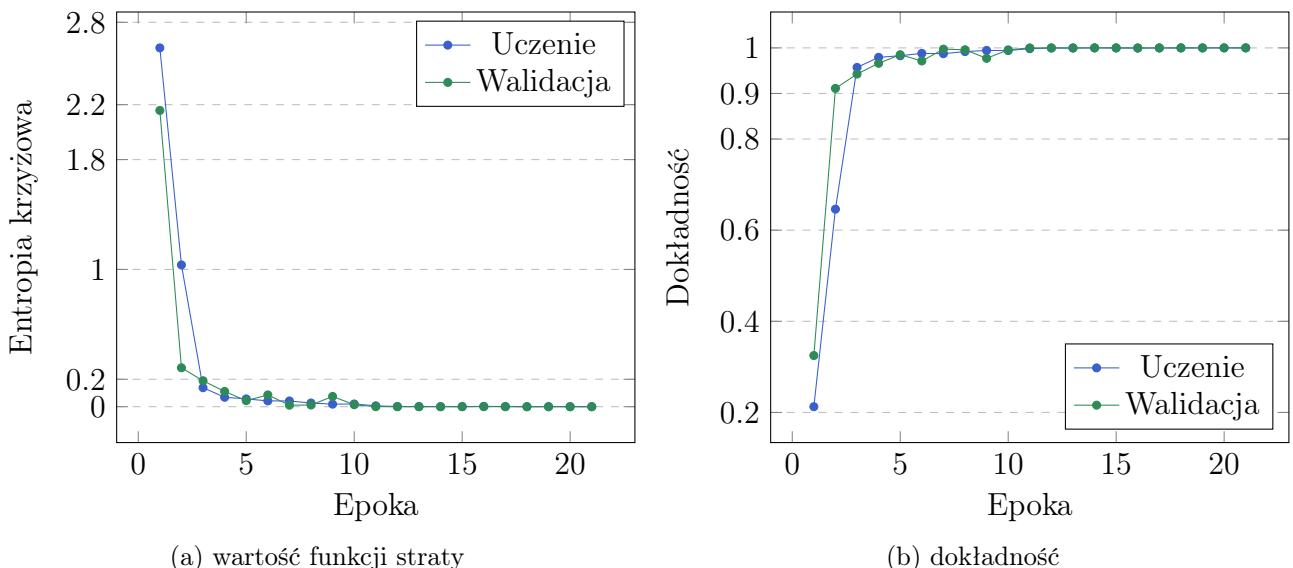


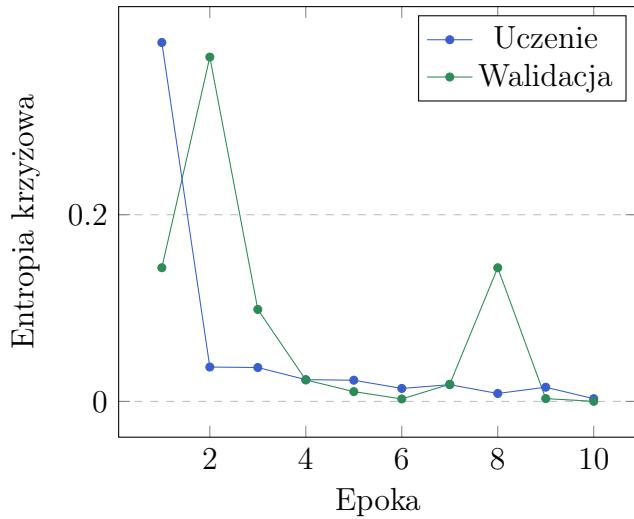
Tabela 2.1: Krzywe uczenia modelu

Model był uczyony optymalizatorem Adam (o domyślnych parametrach $\beta_1 = 0.9$, $\beta_2 = 0.999$), początkowy współczynnik uczenia wynosił 0.001, a funkcją straty była kategoryczna entropia krzyżowa. Dodatkowo wyliczana była metryka dokładności. Zastosowano zredukowanie współczynnika uczenia w przypadku braku postępów w minimalizowaniu funkcji straty walidacji. Uczenie modelu zostało zatrzymane po 21. epoce przez mechanizm wcześniego zatrzymywania (obserwował dokładność dla zbioru walidacyjnego, jego “cierpliwość” została ustawiona na 10 epok) i zostały przywrócone wagi z epoki, która miała najwyższą dokładność dla zbioru walidacyjnego.

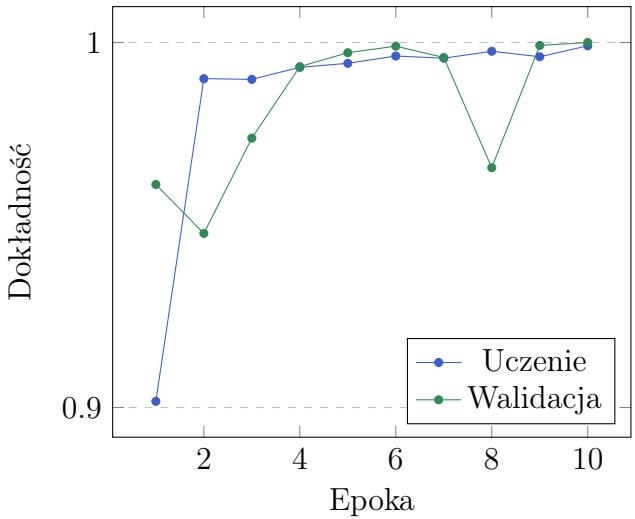
Uczenie transferowe

Jako bazy użyłem wag sieci wytrenowanej na zestawie danych ImageNet. Zaskakująco, nieporównywalnie lepiej poradził sobie model uczony od początku z wszystkimi warstwami

odblokowanymi (bez stopniowego “odmrażania” od strony wyjścia modelu), zatem rozpatrywany będzie tylko on.



(a) wartość funkcji straty



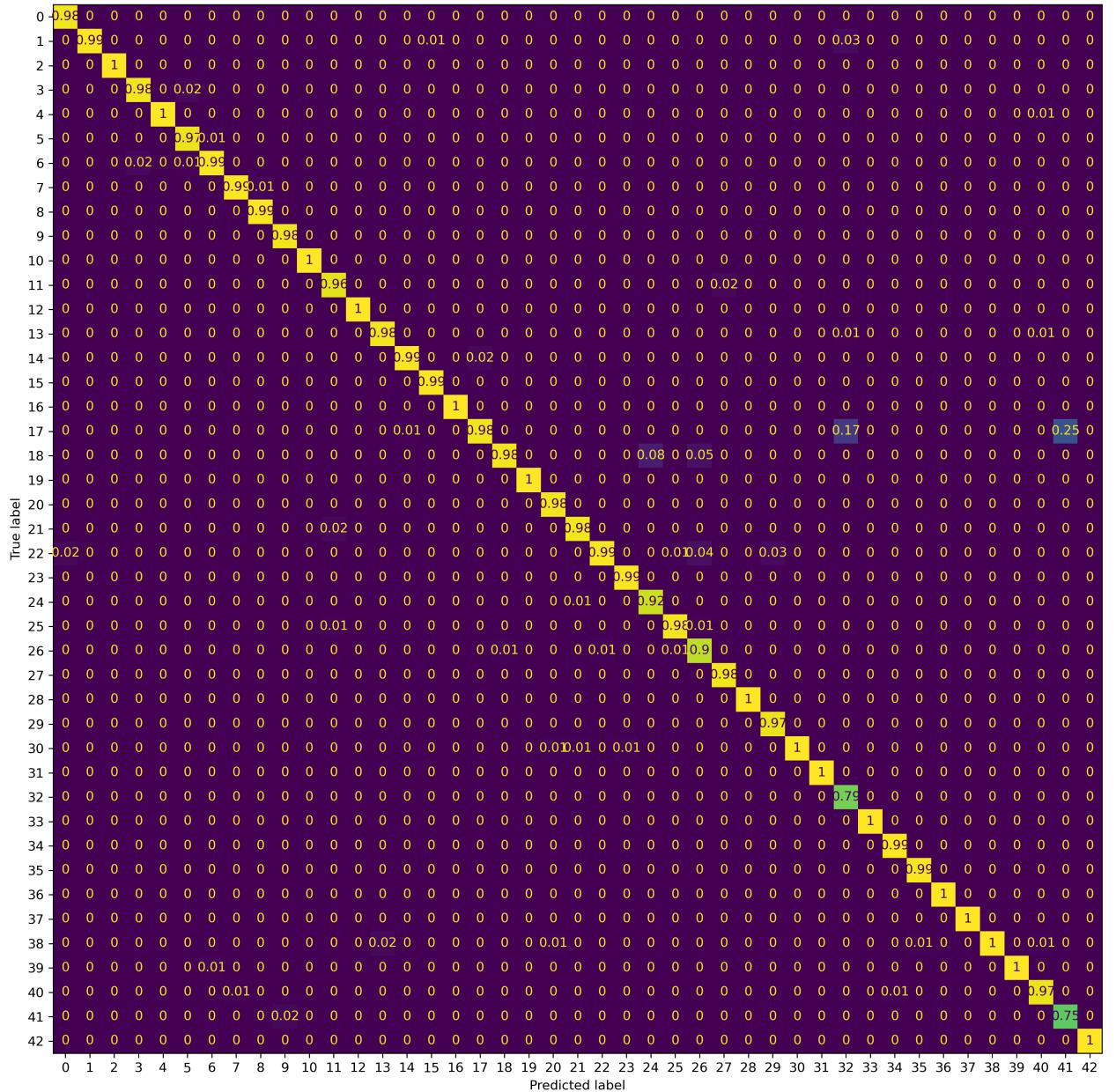
(b) dokładność

Tabela 2.2: Krzywe uczenia modelu

Jak widać, uczenie modelu osiągnęło zbieżność o 11 epok szybciej, a nawet ze względu na zbyt małą surowość mechanizmów wcześniego zatrzymywania - lekkie przetrenowanie.

2.2.3 Ocena działania modelu

Od zera

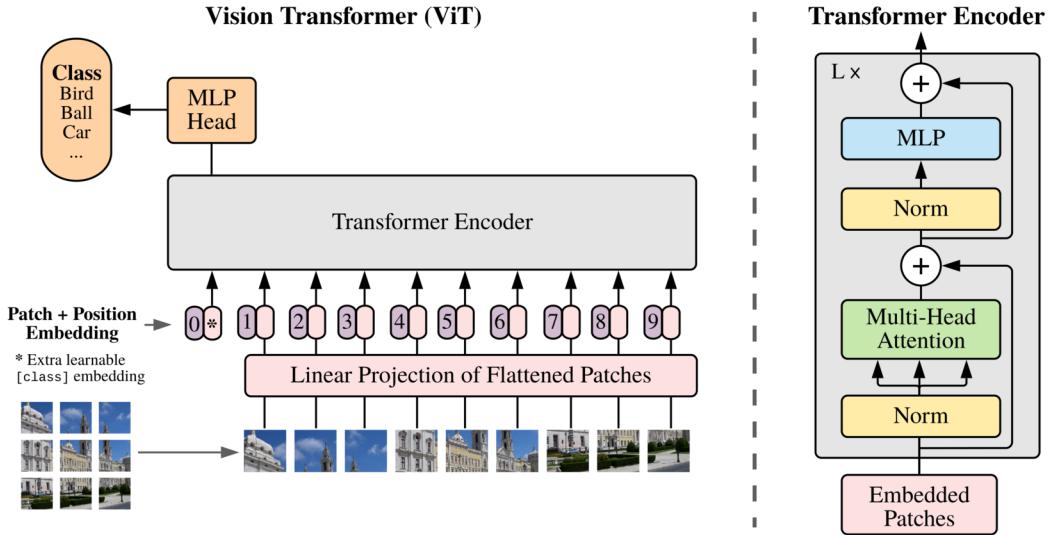


Rysunek 2.2: Macierz pomyłek

W powyższym zestawieniu widać, iż najczęściej mylona jest klasa 17-ta z klasami 32-gą i 41-szą. Po sprawdzeniu stojących za tymi klasami znakami okazuje się, że to znak zakazu wjazdu jest mylony ze znakami oznaczającymi końca obowiązywania wszystkich zakazów oraz końca zakazu wyprzedzania.

2.3 Transformer wizji (ViT)

2.3.1 Opis architektury



Rysunek 2.3: Schemat modelu transormera wizji¹

Transformer wizji (ViT) jest pierwszym udanym praktycznym zastosowaniem modelu opartego o mechanizm uwagi do zadania klasyfikacji zdjęć. Celem jego autorów było jak najwierniejsze przeniesienie pierwotnej architektury transformera (służącej do przetwarzania języka naturalnego), aby móc wykorzystać istniejące już jej implementacje.

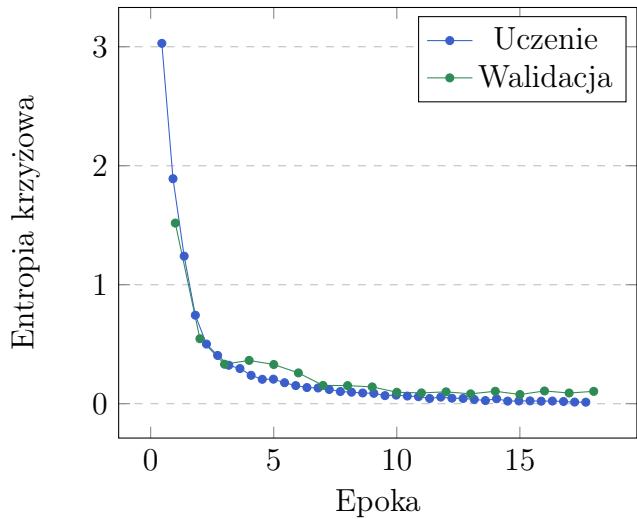
Najpierw rozpatrywany obraz jest dzielony na $N = \frac{HW}{P^2}$ kwadratowych fragmentów, gdzie H i W to odpowiednio wysokość i szerokość obrazu, a P to hiperparametr definiujący długość boku fragmentu. Następnie te “spłaszczone” fragmenty są przekazywane do warstwy przekształcenia liniowego (w pełni połączonej warstwy neuronów pozabawionych funkcji aktywacji), która dostosowuje dane do wymiaru D wektora reprezentacji ukrytej. Zostaje dołączone do nich również poznawane przez model w trakcie uczenia dodatkowe kodowanie, które stanowi wejście dla wielowarstwowego perceptronu rozpoznającego poszczególne klasy. Zakodowane fragmenty obrazu zostają zsumowane z kodowaniami pozycyjnymi (poznawanymi w trakcie uczenia) i przekazane do L następujących po sobie modułów enkodera.

Użyty przeze mnie model używa warstwy splotowej do przekształcenia liniowego i posiada $L = 12$ warstw.

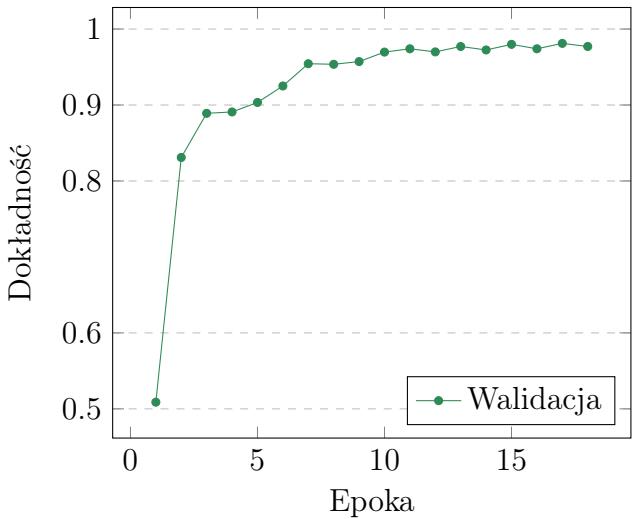
2.3.2 Uczenie modelu

Do uczenia transformera wizji wykorzystałem bibliotekę Transformers utrzymywianą przez firmę Hugging Face.⁶ Umożliwiło to łatwe wczytanie struktury modelu i jego wag oraz dostosowanie parametrów i przeprowadzenie uczenia.

Od zera



(a) wartość funkcji straty

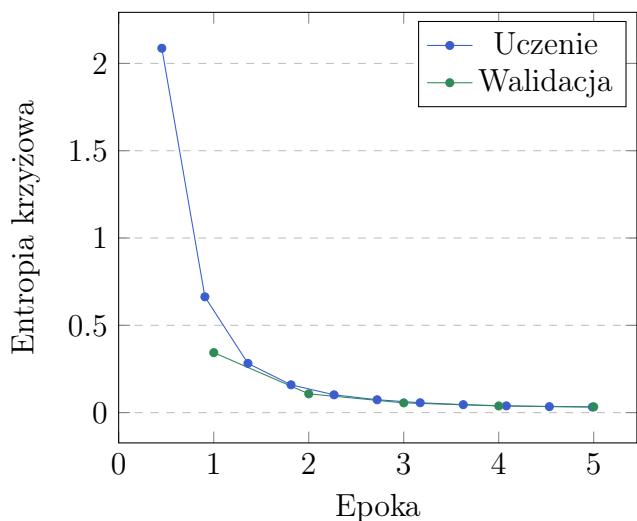


(b) dokładność

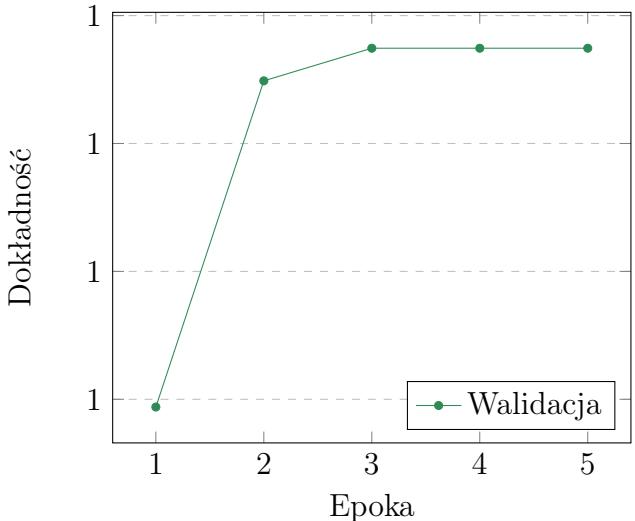
Tabela 2.3: Krzywe uczenia modelu

Uczenie transferowe

Jako bazy użyłem modelu wstępnie wytrenowanego przez Google na zestawie danych ImageNet-21k (zawierającego 14 milionów zdjęć) ([google/vit-base-patch16-224-in21k](#)).



(a) wartość funkcji straty

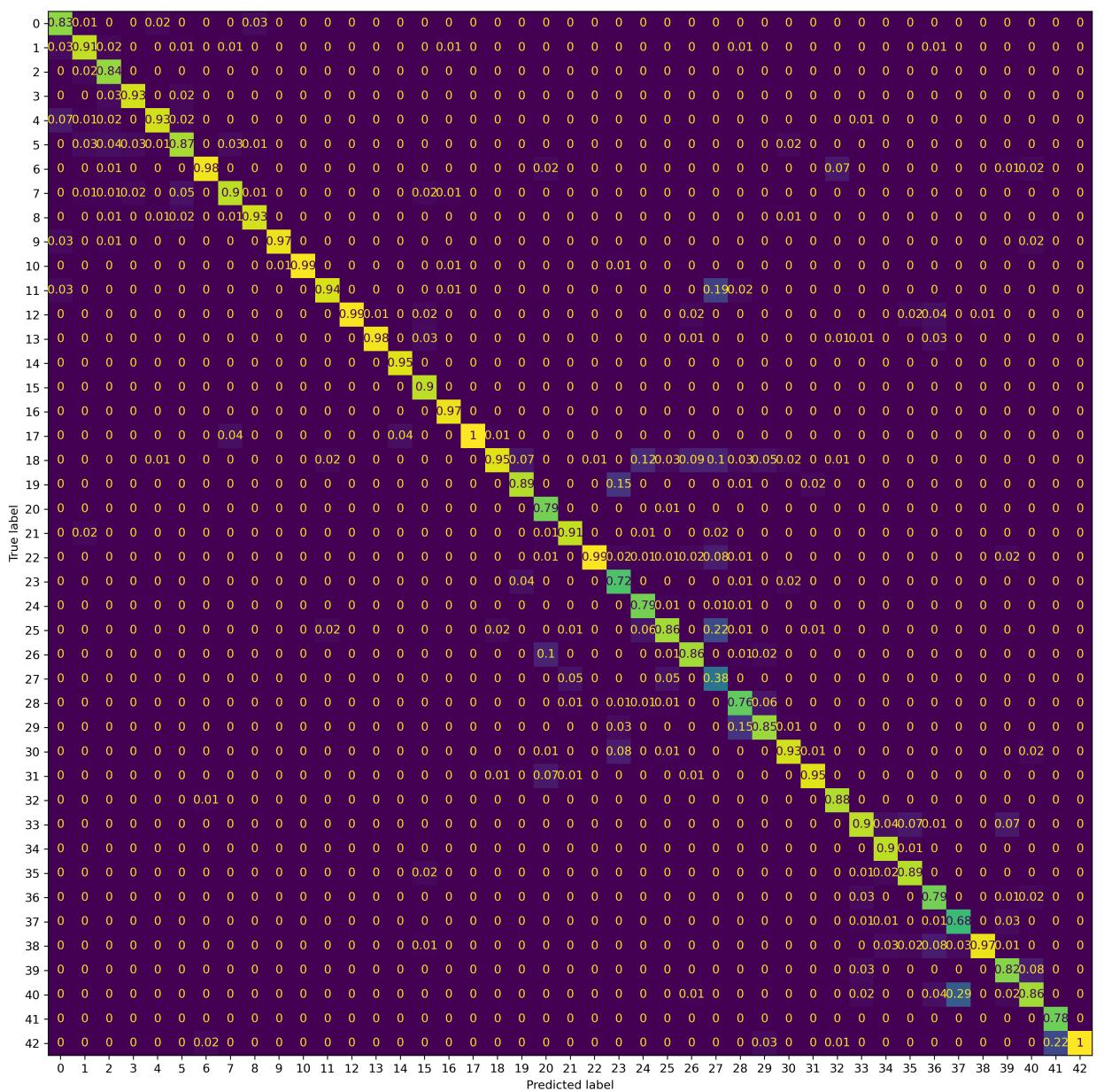


(b) dokładność

Tabela 2.4: Krzywe uczenia modelu

2.3.3 Ocena działania modelu

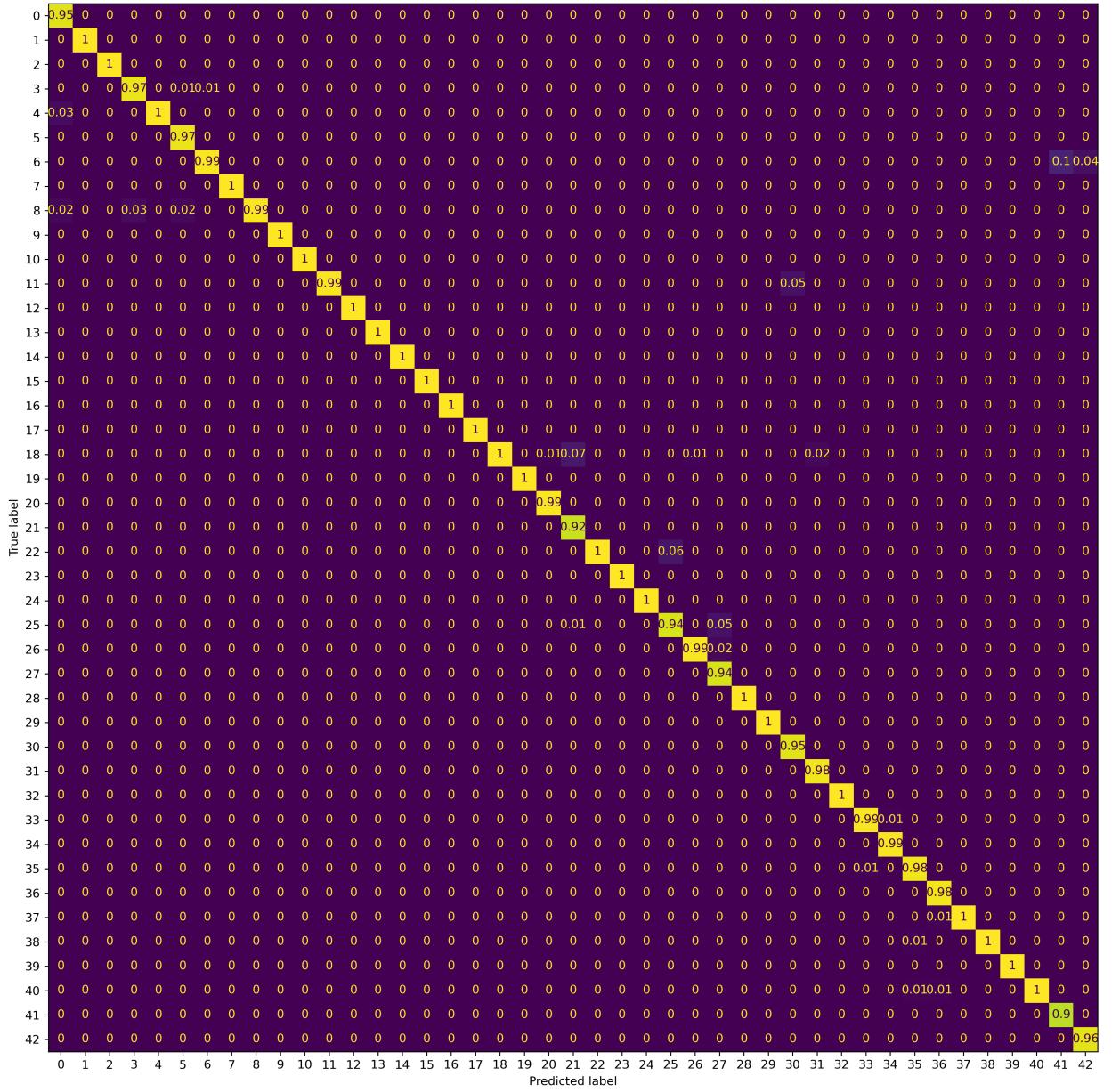
Od zera



Rysunek 2.4: Macierz pomyłek

Jak widać, transformer wizji uczyony od zera, nie osiągnął tak dobrych rezultatów, jak pozostałe modele.

Uczenie transferowe



Rysunek 2.5: Macierz pomyłek

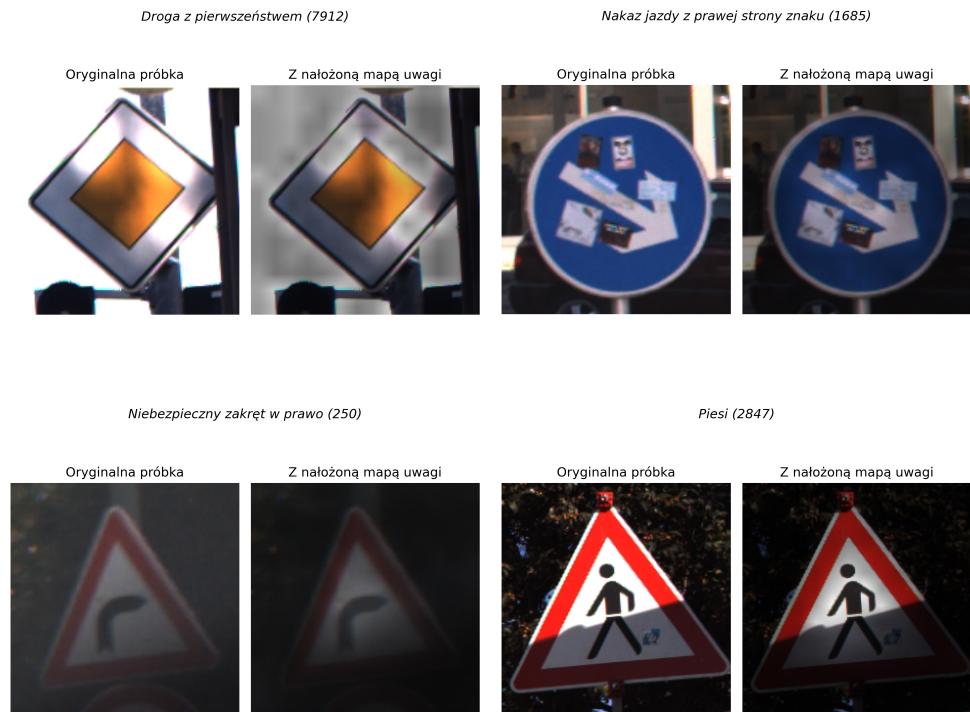
Model ViT, uprzednio wyuczony na bardzo dużym zestawie danych i dotrenowany na znacznie mniejszym zestawie danych docelowego zadania, znakomicie radzi sobie z zadaniem klasyfikacji. Nie wykazuje również przy tym zbytnio charakterystycznych pomyłek, jak w przypadku sieci ResNet-50.

2.3.4 Wizualizacja działania - mapa uwagi

Zaletą modeli opartych o mechanizm uwagi jest ich wyjaśnialność, czyli możliwość sprawdzenia w oparciu o jakie konkretne informacje z danego przypadku model podjął decyzję o klasyfikacji. Może to pomóc w diagnozowaniu miejsc, w których model jest niedotrenowany, bądź jego transparentnym użyciu³ (np. sprawdzenie, czy nie jest on dyskryminu-

jacy).

W tym konkretnym przypadku możliwe jest uzyskanie tzw. mapy uwagi. Proces przebiega intuicyjnie poprzez wymnożenie wag uwagi z poszczególnych warstw ukrytych transformera. Rezultatem jest obraz z nałożoną mapą, gdzie jaśniejsze miejsca pokazują, na których miejscach skupił się model podczas klasyfikowania próbki.



Rysunek 2.6: Mapy uwagi dla różnych przykładów z zestawu testowego

2.4 Porównanie modeli

Model		Dokładność (zbiór testowy)	Czas trenowania
ResNet-34	Od zera	97,89%	ok. 1,5 godziny
ResNet-50	Od zera	98,41%	ok. 2 godziny
	Uczenie transferowe	98,76%	ok. 1 godziny
ViT	Od zera	91,27%	ok. 8 godzin
	Uczenie transferowe	98,92%	ok. 2 godziny

Tabela 2.5: Zestawienie poszczególnych modeli

Rozdział 3

Podsumowanie

Pierwsze modele oparte o transformery wizji miały za zadanie przede wszystkim pokazać, że jest możliwe zastosowanie mechanizmów uwagi samych w sobie w dziedzinie widzenia komputerowego. W związku z tym są one bardzo zbliżone do modelu transformera stosowanego już z powodzeniem w przypadku zadań związanych z przetwarzaniem języka naturalnego (NLP). Oznacza to, że przetarły one szlak do dalszego rozwoju takich modeli, co już ma miejsce z powodzeniem.

Główne potencjalne korzyści z użycia transformerów w klasyfikacji i detekcji zdjęć leżą w uczeniu transferowym i wyjaśnialności. Obarczone jest to jednak większymi rozmiarami modeli, co spowalnia znaczowo uczenie i wnioskowanie. Większe modele są konieczne, jeśli model ViT ma być w stanie poznać cechy, nie dysponując przy tym pewną dозą niezmienniczości (tak jak CNN). W porównaniu z modelami CNN o podobnych rozmiarach ViT wypada jednak korzystniej obliczeniowo, a dalsze usprawnienia można uzyskać np. poprzez ograniczenia pola działania mechanizmu uwagi.

Bibliografia

- ¹ Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.
- ² Google. Video transformer (base-sized model). <https://web.archive.org/web/20240815110837/https://huggingface.co/google/vit-base-patch16-224>. Dostęp: 2024-08-17.
- ³ Aurélien Géron. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow. Wydanie II.* Helion, Gliwice, 2020.
- ⁴ Bulent Siyah. Imagenet winning cnn architectures (ilsvrc), 2020.
- ⁵ J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012.
- ⁶ Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.