

PRACA DYPLOMOWA MAGISTERSKA

Analiza i badanie działania wybranych sieci neuronowych w zadaniu klasyfikacji obrazów na zbiorze ImageNet

*Analysis and study of the performance of selected neural networks in the image classification task on the ImageNet dataset*

Marek Sigmund

Nr albumu: 133552

Kierunek: Sztuczna Inteligencja i Data Science

Forma studiów: stacjonarne

Poziom studiów: II

Promotor pracy:

dr hab. inż. prof. PCz Krystian Łapa

Praca przyjęta dnia:

Podpis promotora:

*Częstochowa,* 2025

Spis treści

[1. Wstęp 4](#_Toc191119550)

[1.1. Motywacja do badań 6](#_Toc191119551)

[1.2. Cel i zakres pracy 7](#_Toc191119552)

[2. Zbiór danych 8](#_Toc191119553)

[2.1. Zbiór ImageNet 9](#_Toc191119554)

[2.2. Wstępne przetwarzanie danych 11](#_Toc191119555)

[3. Użyte technologie i metodologia eksperymentów 12](#_Toc191119556)

[3.1. Środowisko obliczeniowe i narzędzia 13](#_Toc191119557)

[3.2. Metodologia eksperymentów 15](#_Toc191119558)

[3.3. Struktura eksperymentów 16](#_Toc191119559)

[4. Sieci konwolucyjne (CNN) 17](#_Toc191119560)

[4.1. Liczba warstw konwolucyjnych 19](#_Toc191119561)

[4.1.1. Opis testowanych modeli 20](#_Toc191119562)

[4.1.2. Analiza wyników 23](#_Toc191119563)

[4.1.3. Podsumowanie i porównanie wyników modeli CNN 33](#_Toc191119564)

[4.2. Liczba warstw w pełni połączonych 35](#_Toc191119565)

[4.2.1. Opis eksperymentu dla małej ilości warstw 35](#_Toc191119566)

[4.2.1.1. Wyniki eksperymentu 36](#_Toc191119567)

[4.2.1.2. Podsumowanie i wnioski 38](#_Toc191119568)

[4.2.2. Medium FCNN 39](#_Toc191119569)

[4.2.2.1. Opis eksperymentu 39](#_Toc191119570)

[4.2.2.2. Wyniki eksperymentu 40](#_Toc191119571)

[4.2.2.3. Podsumowanie i wnioski 41](#_Toc191119572)

[4.2.3. MaximalCNN 42](#_Toc191119573)

[4.2.3.1. Opis eksperymentu 42](#_Toc191119574)

[4.2.3.2. Wyniki eksperymentu 43](#_Toc191119575)

[4.2.3.3. Podsumowanie 45](#_Toc191119576)

[4.3. Ogólne podsumowanie i wnioski 45](#_Toc191119577)

[1.1. Liczba warstw w pełni połączonych 48](#_Toc191119578)

[1.2. Pozostałem eksperymenty 48](#_Toc191119579)

[2. Wpływ danych na działanie sieci 53](#_Toc191119580)

[3. Podsumowanie i wnioski 54](#_Toc191119581)

[4. Istniejące sieci neuronowe i porównanie 54](#_Toc191119582)

[5. Dalsze kroki i możliwości rozwoju 55](#_Toc191119583)

[6. Praktyczne zastosowanie najlepszego modelu 55](#_Toc191119584)

[Bibliografia 56](#_Toc191119585)

[Spis tabel 56](#_Toc191119586)

[Spis ilustracji 56](#_Toc191119587)

[Streszczenie 56](#_Toc191119588)

[Summary 56](#_Toc191119589)

# Wstęp

Klasyfikacja obrazów jest jednym z kluczowych zagadnień w dziedzinie sztucznej inteligencji i widzenia komputerowego, znajdującym szerokie zastosowanie między innymi w medycynie, motoryzacji, przemyśle czy bezpieczeństwie. Jej głównym celem jest automatyczne przypisywanie obrazów do określonych kategorii na podstawie ich zawartości wizualnej. Proces ten, choć obecnie zdominowany przez metody oparte na głębokim uczeniu, ma długą historię, obejmującą tradycyjne podejścia wykorzystujące ręczną ekstrakcję cech oraz klasyczne algorytmy uczenia maszynowego.

Pierwsze metody klasyfikacji obrazów opierały się na ekstrakcji charakterystycznych cech, takich jak krawędzie, tekstury czy histogramy kolorów. Popularne techniki, takie jak SIFT (Scale-Invariant Feature Transform) czy HOG (Histogram of Oriented Gradients), pozwalały na reprezentowanie obrazów w formie wektorów cech, które następnie były klasyfikowane przy użyciu algorytmów takich jak maszyny wektorów nośnych (SVM) czy k-średnich (k-means). Choć podejścia te były skuteczne w prostych zadaniach, miały swoje ograniczenia – wymagały skomplikowanego procesu inżynierii cech oraz często nie radziły sobie dobrze z bardziej złożonymi obrazami, zawierającymi duże wariacje w oświetleniu, orientacji czy perspektywie.

Prawdziwym przełomem w dziedzinie klasyfikacji obrazów okazało się zastosowanie głębokich sieci neuronowych, a w szczególności konwolucyjnych sieci neuronowych (CNN – Convolutional Neural Networks). W 2012 roku zespół Alexa Krizhevsky'ego zaprezentował sieć AlexNet, która zdominowała prestiżowe zawody ImageNet Large Scale Visual Recognition Challenge (ILSVRC), osiągając wyniki znacznie przewyższające tradycyjne metody. To wydarzenie zapoczątkowało gwałtowny rozwój algorytmów opartych na głębokim uczeniu, które w kolejnych latach znacząco poprawiły skuteczność klasyfikacji obrazów i stały się dominującym podejściem w tej dziedzinie.

Obecnie klasyfikacja obrazów z wykorzystaniem sieci neuronowych jest standardem w systemach analizy obrazu. Dzięki ogromnym zbiorom danych, takim jak ImageNet, oraz rosnącej mocy obliczeniowej, modele głębokiego uczenia są w stanie automatycznie uczyć się skomplikowanych reprezentacji wizualnych bez potrzeby ręcznej ekstrakcji cech. Rozwój sieci takich jak VGG, ResNet czy EfficientNet umożliwił dalszą poprawę dokładności klasyfikacji, a jednocześnie zmniejszenie liczby wymaganych zasobów obliczeniowych.

Dynamiczny rozwój technologii klasyfikacji obrazów otworzył nowe możliwości w wielu dziedzinach, od diagnostyki medycznej, przez systemy autonomiczne, aż po rozpoznawanie twarzy i analizę wideo w czasie rzeczywistym. Pomimo ogromnych postępów, wciąż istnieją wyzwania związane z interpretowalnością modeli, ich odpornością na zakłócenia oraz efektywnością obliczeniową. Poszukiwanie coraz lepszych i bardziej optymalnych architektur sieci neuronowych pozostaje kluczowym kierunkiem badań, a ich zastosowanie w praktyce stale się rozszerza, rewolucjonizując kolejne obszary technologii i nauki.[1][2][3]

## Motywacja do badań

**AKTUALIZOWAĆ WRAZ Z POSTĘPAMI SKLEIĆ DOBRZE Z NASTEPNYM**

W ostatnich latach sieci neuronowe, a w szczególności CNN, stały się dominującym podejściem w zadaniach klasyfikacji obrazów. Ich skuteczność została wielokrotnie potwierdzona w różnych zastosowaniach. Jednak pomimo znaczących postępów, wybór odpowiedniej architektury sieci pozostaje kluczowym zagadnieniem badawczym. Struktura sieci, w tym liczba warstw konwolucyjnych, funkcje aktywacji oraz inne parametry architektoniczne, mają istotny wpływ na jej stabilność, zdolność generalizacji i efektywność obliczeniową.

Główną motywacją do podjęcia tego tematu jest potrzeba lepszego zrozumienia, w jaki sposób różne aspekty architektury sieci neuronowych wpływają na ich skuteczność w zadaniu klasyfikacji obrazów. W literaturze istnieje wiele różnych podejść do projektowania modeli, jednak nie wszystkie prowadzą do optymalnych wyników. Istotne jest zbadanie, jakie zmiany w strukturze sieci – zarówno pod względem liczby warstw, rodzaju funkcji aktywacji, parametrów konwolucyjnych, jak i innych istotnych elementów – wpływają na ich stabilność, zdolność generalizacji oraz efektywność obliczeniową.

Badania będą obejmować zarówno eksperymenty na własnych modelach, w których możliwe będzie modyfikowanie kluczowych parametrów, jak i testy na predefiniowanych architekturach, takich jak VGG, ResNet czy EfficientNet. Dzięki temu możliwe będzie porównanie, jak zmieniają się wyniki między modelami projektowanymi od podstaw a zoptymalizowanymi już sieciami. Dodatkowo, jeśli pozwolą na to zasoby obliczeniowe, zostaną przeanalizowane modele działające w różnych środowiskach, co umożliwi lepszą ocenę wpływu infrastruktury na efektywność obliczeniową oraz dokładność klasyfikacji.

Ostatecznym celem badań jest stworzenie modelu sieci neuronowej o zoptymalizowanej architekturze, który uzyska wysoką skuteczność klasyfikacji obrazów przy jednoczesnym zachowaniu stabilności. Uzyskane wyniki mogą dostarczyć cennych wskazówek na temat projektowania sieci neuronowych, pomagając w lepszym dostosowaniu ich architektur do konkretnych zadań.

## Cel i zakres pracy

**AKTUALIZOWAĆ WRAZ Z POSTĘPAMI DOBRZE SKLEIĆ Z PORPZEDNIM**

Celem niniejszej pracy jest analiza wpływu różnych aspektów architektury sieci neuronowych na skuteczność klasyfikacji obrazów. Szczególną uwagę poświęcono takim czynnikom jak liczba warstw konwolucyjnych, zastosowane funkcje aktywacji oraz inne kluczowe parametry strukturalne modeli. Badania mają na celu określenie, w jakim stopniu modyfikacja tych elementów wpływa na stabilność działania sieci, jej zdolność generalizacji oraz efektywność obliczeniową. Ostatecznym celem jest zaprojektowanie zoptymalizowanego modelu, który osiągnie wysoką skuteczność klasyfikacji obrazów, przy jednoczesnym zachowaniu odpowiednich właściwości modelu, takich jak odporność na zmiany danych wejściowych oraz rozsądne wymagania sprzętowe.

Zakres pracy obejmuje przeprowadzenie eksperymentów na różnych architekturach sieci neuronowych. Testowane będą zarówno modele projektowane od podstaw, w których możliwe będzie precyzyjne dostosowanie liczby warstw i innych parametrów, jak i predefiniowane sieci, takie jak VGG, ResNet czy EfficientNet. W ramach badań analizowane będą różne strategie konstrukcji modeli, a także wpływ infrastruktury obliczeniowej na efektywność treningu i klasyfikacji.

Struktura pracy została zaplanowana w sposób umożliwiający systematyczne przedstawienie wyników kolejnych eksperymentów. Poszczególne rozdziały poświęcone będą analizie różnych aspektów architektury sieci neuronowych oraz ich wpływu na klasyfikację obrazów. Każdy z nich będzie koncentrował się na określonym zakresie badań, od analizy podstawowych modeli po testowanie bardziej złożonych i zoptymalizowanych rozwiązań. Ze względu na charakter badań, treść poszczególnych sekcji będzie aktualizowana w miarę postępów w realizacji eksperymentów, co pozwoli na dokładniejsze określenie zakresu analizy i uzyskanych wyników.

**STRUKTURA W STYLU PIERWSZY ROZDZIAŁ..**

# Zbiór danych

**SKRÓCIĆ**

Dane stanowią kluczowy element w procesie trenowania sieci neuronowych, determinując ich zdolność do generalizacji oraz skuteczność w rozwiązywaniu rzeczywistych problemów. Jakość, różnorodność i odpowiednie przygotowanie zbioru danych wpływają bezpośrednio na wyniki uzyskiwane przez modele uczenia maszynowego, a niewłaściwie przygotowane dane mogą prowadzić do błędnych klasyfikacji oraz problemów związanych z nadmiernym dopasowaniem (overfitting) lub niedostatecznym dopasowaniem (underfitting).

Aby sieć neuronowa mogła skutecznie uczyć się wzorców i zależności, konieczne jest zapewnienie zbioru treningowego o wysokiej jakości. Obejmuje to zarówno poprawne oznaczenie etykiet klas, jak i odpowiednią reprezentację różnych wariantów danych wejściowych. Zniekształcenia, szumy czy błędne oznaczenia mogą wprowadzać niepewność w procesie uczenia i negatywnie wpływać na zdolność modelu do uogólniania wiedzy na nowe przykłady [4]. Ponadto duża liczba próbek w zbiorze treningowym nie gwarantuje wysokiej skuteczności modelu, jeśli dane nie są odpowiednio zrównoważone i zróżnicowane.

Kolejnym istotnym aspektem jest wstępne przetwarzanie danych, które obejmuje różne techniki dostosowujące obrazy do wymagań modelu. Procesy takie jak normalizacja, standaryzacja oraz augmentacja danych mogą znacząco poprawić zdolność sieci do generalizacji, minimalizując ryzyko przeuczenia modelu na specyficznych cechach zbioru [5]. W przypadku dużych zbiorów, takich jak ImageNet, odpowiednie przygotowanie danych jest skomplikowanym i czasochłonnym procesem, który obejmuje m.in. eliminację duplikatów, poprawę jakości obrazów oraz weryfikację etykiet klas [6].

Zbiory danych wykorzystywane w uczeniu maszynowym muszą być również poddawane kontroli jakości, aby uniknąć problemów związanych z błędnymi klasyfikacjami czy nierównomiernym rozkładem próbek pomiędzy kategoriami. W szczególności w przypadku ImageNet, który zawiera miliony obrazów należących do tysięcy różnych klas, proces weryfikacji i oczyszczania danych odgrywa kluczową rolę w zapewnieniu rzetelnych wyników eksperymentów. Właściwe przygotowanie zbioru danych nie tylko poprawia skuteczność modeli, ale także pozwala na lepsze zrozumienie problemów związanych z klasyfikacją obrazów oraz identyfikację potencjalnych ograniczeń wybranych architektur sieci neuronowych.

## Zbiór ImageNet

ImageNet to jeden z najważniejszych zbiorów danych w historii widzenia komputerowego i uczenia maszynowego. Został stworzony w 2009 roku przez zespół badaczy pod kierownictwem Fei-Fei Li i zawiera ponad 14 milionów obrazów przypisanych do ponad 20 tysięcy kategorii. Zbiór ten odegrał kluczową rolę w rozwoju algorytmów klasyfikacji obrazów, umożliwiając trenowanie i ocenę modeli na dużą skalę [7].

Najbardziej znaną częścią ImageNet jest podzbiór wykorzystywany w ImageNet Large Scale Visual Recognition Challenge (ILSVRC) – prestiżowym konkursie organizowanym od 2010 do 2017 roku. W ramach tego wyzwania modele były testowane na 1,2 miliona obrazów podzielonych na 1000 klas, co stało się standardem dla porównywania skuteczności różnych architektur sieci neuronowych. ILSVRC przyczynił się do ogromnego postępu w dziedzinie głębokiego uczenia, szczególnie po przełomowym zwycięstwie sieci AlexNet w 2012 roku, które zapoczątkowało erę dominacji konwolucyjnych sieci neuronowych (CNN) w zadaniach klasyfikacji obrazów [8].

Dzięki ImageNet opracowano i udoskonalono wiele kluczowych architektur, takich jak VGG, GoogLeNet, ResNet czy EfficientNet, które znalazły zastosowanie w różnych obszarach, od medycyny po autonomiczne pojazdy. Sukcesy osiągane w konkursach ILSVRC stały się wyznacznikiem jakości dla nowych metod klasyfikacji obrazów, a zwycięskie modele były szeroko stosowane i adaptowane w kolejnych latach.

Zbiór ImageNet doczekał się wielu wariantów dostosowanych do różnych zastosowań. Jednym z nich jest Tiny ImageNet, który stanowi zmniejszoną wersję zbioru, zawierającą 200 klas, po 500 obrazów na klasę w zbiorze treningowym. Jest on często używany w badaniach akademickich, ponieważ umożliwia trenowanie modeli przy znacznie niższych wymaganiach sprzętowych niż pełny ImageNet, zachowując jednocześnie podobną strukturę klasyfikacji [9].

ImageNet nie tylko wpłynął na rozwój sieci neuronowych, ale także wyznaczył nowe standardy w ocenie skuteczności algorytmów klasyfikacji obrazów. Stał się fundamentem dla dalszych badań nad architekturami deep learningu, przyczyniając się do opracowania coraz bardziej zaawansowanych modeli zdolnych do rozwiązywania skomplikowanych problemów wizualnych.

**DODAĆ JAKĄŚ TABELKĘ ZE STATYSTYKAMI CZY PORÓWNANIEM I/LUB JAKIEŚ GRAFIKI**

## Wstępne przetwarzanie danych

Przed rozpoczęciem treningu sieci neuronowych kluczowym etapem jest wstępne przetwarzanie danych, które ma na celu zwiększenie jakości oraz efektywności uczenia modeli. Odpowiednie przygotowanie danych wpływa na stabilność procesu optymalizacji, szybkość konwergencji oraz zdolność sieci do generalizacji. W przypadku zbiorów obrazów obejmuje ono zarówno podstawowe operacje, jak normalizację i standaryzację, jak i bardziej zaawansowane techniki, takie jak augmentacja danych.

Podstawowym krokiem w przetwarzaniu obrazów jest normalizacja, która polega na przeskalowaniu wartości pikseli do określonego zakresu, co zmniejsza wpływ różnic w oświetleniu i kontrastach. Standaryzacja (odejmowanie średniej i dzielenie przez odchylenie standardowe) pozwala na ujednolicenie zakresów wartości wejściowych, co może poprawić stabilność treningu [10].

Podział zbioru na zestawy treningowe, walidacyjne i testowe to kolejny istotny element przetwarzania. W przypadku Tiny ImageNet obrazy są domyślnie podzielone na zbiór treningowy (200 klas, po 500 próbek na klasę) oraz walidacyjny (50 obrazów na klasę), jednak w niektórych badaniach stosuje się dodatkowy podział na zbiór testowy lub modyfikacje proporcji między tymi zestawami w zależności od potrzeb eksperymentalnych.

Augmentacja danych, czyli celowe modyfikowanie obrazów w celu zwiększenia różnorodności zbioru treningowego, jest powszechnie stosowaną metodą poprawy zdolności generalizacji modeli. Typowe operacje obejmują obracanie obrazów, skalowanie, odbicia lustrzane, zmiany kolorystyki czy dodawanie szumu, co pomaga modelom radzić sobie z różnorodnymi warunkami danych wejściowych [11].

W przypadku Tiny ImageNet konieczne jest również dostosowanie rozmiaru obrazów do wymagań modeli – obrazy w zbiorze mają wymiar 64×64 piksele, co jest znacząco mniejsze w porównaniu do pełnego ImageNet (224×224). W zależności od architektury modeli można stosować dodatkowe operacje, takie jak interpolacja do większych rozmiarów, co pozwala na ich kompatybilność z architekturami CNN trenowanymi na większych wejściach.

Ponieważ wpływ różnych technik przetwarzania danych na skuteczność modeli będzie analizowany w dedykowanym rozdziale eksperymentalnym, ta sekcja ogranicza się do podstawowego omówienia metod przygotowania zbioru Tiny ImageNet do treningu. Szczegółowe porównanie modeli trenowanych na różnych wariantach przetworzonych danych, takich jak binaryzacja, rotacje czy filtracja szumów, zostanie przedstawione w dalszej części pracy.

**TA KOŃCÓWKA TAKA ŚREDNIA**

# Użyte technologie i metodologia eksperymentów

Postęp w dziedzinie głębokiego uczenia oraz klasyfikacji obrazów jest ściśle związany z rozwojem narzędzi programistycznych i infrastruktury obliczeniowej. Wybór odpowiednich technologii oraz metod przeprowadzania eksperymentów ma kluczowe znaczenie dla jakości uzyskiwanych wyników, a także dla efektywności procesu trenowania modeli. Współczesne badania nad sieciami neuronowymi opierają się na wyspecjalizowanych frameworkach, które umożliwiają optymalizację obliczeń oraz elastyczne projektowanie i testowanie różnych architektur sieciowych [12].

W niniejszym rozdziale przedstawione zostaną technologie wykorzystane do przeprowadzenia eksperymentów, w tym język programowania, biblioteki wspierające głębokie uczenie oraz infrastruktura sprzętowa użyta do trenowania modeli. Następnie omówiona zostanie metodologia badań, w tym sposób przetwarzania danych, metody oceny skuteczności modeli oraz strategie zapewnienia powtarzalności eksperymentów. Ostatnia część rozdziału poświęcona będzie strukturze eksperymentów, opisując, w jaki sposób zostaną one podzielone i jakie aspekty działania sieci neuronowych będą badane.

Dzięki takiej organizacji możliwe będzie przeprowadzenie systematycznej analizy wpływu różnych czynników na skuteczność modeli, co pozwoli na wyciągnięcie precyzyjnych wniosków dotyczących optymalnych parametrów architektonicznych sieci neuronowych.

## Środowisko obliczeniowe i narzędzia

**DODAĆ OBRAZKI I OPISY POSZCZEGÓLNYCH NARZĘDZI PODOBNIE JAK MIAŁEM TO W INŻYNIERCE UROZMAIĆIĆ TO FAJNIE**

Efektywność trenowania modeli głębokiego uczenia w dużej mierze zależy od środowiska obliczeniowego, które obejmuje zarówno sprzęt, jak i oprogramowanie wspierające proces przetwarzania danych. Odpowiednia infrastruktura wpływa na czas treningu, możliwość eksperymentowania z bardziej złożonymi architekturami oraz optymalizację obliczeń.

Do przeprowadzania eksperymentów wykorzystano komputer wyposażony w **UZUPEŁNIJ**. Wykorzystano procesor **UZUPEŁNIJ**, który zapewniał wystarczającą moc obliczeniową do wstępnego przetwarzania danych oraz zarządzania przepływem informacji między warstwami sieci. Pamięć operacyjna o pojemności **UZUPEŁNIJ** umożliwiła przetwarzanie dużych zbiorów danych i przyspieszyła operacje związane z ładowaniem oraz manipulacją danymi. Kluczowym elementem środowiska obliczeniowego była karta graficzna **UZUPEŁNIJ**, która dzięki wsparciu technologii CUDA umożliwiła znaczną akcelerację obliczeń i skrócenie czasu trenowania modeli.

Do implementacji i trenowania sieci neuronowych wykorzystano framework PyTorch, który zapewnia elastyczne podejście do budowy i testowania modeli głębokiego uczenia. PyTorch umożliwia dynamiczne konstruowanie grafów obliczeniowych, co pozwala na łatwe dostosowywanie architektur i parametrów modeli do zmieniających się warunków eksperymentalnych. Ponadto wspiera on natywnie akcelerację GPU, co znacząco redukuje czas potrzebny na optymalizację wag sieci i propagację wsteczną.

Kod eksperymentalny został napisany w języku Python, który jest obecnie standardem w dziedzinie uczenia maszynowego i głębokiego uczenia. Python oferuje szeroki ekosystem bibliotek oraz intuicyjną składnię, co ułatwia zarówno implementację modeli, jak i analizę wyników.

Jako środowisko programistyczne wykorzystano Visual Studio Code, które zapewniło wygodną organizację kodu oraz możliwość integracji z bibliotekami PyTorch i CUDA. VS Code oferuje wsparcie dla debugowania kodu, zarządzania wersjami oraz wizualizacji wyników, co znacząco ułatwiło pracę nad eksperymentami.

Obliczenia na karcie graficznej były możliwe dzięki technologii CUDA, która pozwala na równoległe przetwarzanie danych na jednostkach GPU. Dzięki wsparciu dla PyTorch, CUDA umożliwiła znaczne przyspieszenie operacji macierzowych, co było kluczowe dla trenowania głębokich sieci neuronowych.

W przetwarzaniu i analizie danych wykorzystano bibliotekę NumPy, która umożliwiła efektywne operacje numeryczne, w tym przekształcanie tensorów, obliczenia statystyczne oraz optymalizację danych wejściowych dla modeli. Dodatkowo zastosowano Matplotlib oraz Seaborn, które posłużyły do wizualizacji wyników eksperymentów, analizy rozkładu danych oraz interpretacji wyników klasyfikacji.

W eksperymentach zastosowano także bibliotekę Torchvision, która dostarcza zestaw narzędzi do przetwarzania obrazów oraz implementacji gotowych modeli głębokiego uczenia. Dzięki niej możliwe było łatwe wczytywanie, skalowanie oraz augmentacja obrazów, co przyczyniło się do poprawy generalizacji modeli i ich zdolności do rozpoznawania wzorców w danych wejściowych.

## Metodologia eksperymentów

**INFO: Szczegółowe metryki oraz inne elementy metodologii zostaną uzupełnione/zaktualizowane po zakończeniu rozdziałów eksperymentalnych.**

W celu zapewnienia spójności i rzetelności badań wszystkie modele były trenowane według jednolitego schematu, przy czym wyjątek stanowiły modele końcowe, które zostały poddane szczegółowej analizie pod kątem różnych parametrów treningu. Każdy eksperyment przeprowadzano na tym samym zbiorze danych, bez modyfikacji podziału na zestawy treningowe, walidacyjne i testowe, co pozwoliło na obiektywne porównanie wyników.

Eksperymenty koncentrowały się głównie na badaniu stabilności modeli i wpływie architektury sieci na ich działanie, a nie na dokładności klasyfikacji. Z tego powodu większość testów nie wymagała wielokrotnego uruchamiania, ponieważ analizowano właściwości modeli wynikające z ich struktury. Wyjątek stanowiły modele końcowe, które zostały przetestowane wielokrotnie w różnych warunkach, aby sprawdzić ich wydajność i odporność na zmiany parametrów.

Do oceny skuteczności modeli wykorzystano klasyczne miary jakości, takie jak accuracy, loss, F1-score, a także wizualizacje w postaci wykresów. Dokładny zestaw metryk zostanie uzupełniony po zakończeniu rozdziału eksperymentalnego, gdy zostaną określone wszystkie testowane aspekty.

Modele o różnych konfiguracjach architektury były trenowane w ten sam sposób. Na przykład w eksperymencie dotyczącym wpływu liczby warstw w pełni połączonych (fully connected) każdy z trzech modeli miał identyczne ustawienia treningowe, różniąc się jedynie strukturą. Takie podejście umożliwiło rzetelną ocenę wpływu konkretnej zmiennej na wyniki sieci.

Regularyzacja, obejmująca techniki takie jak Dropout czy Batch Normalization, stanowiła jeden z kluczowych aspektów badanych w eksperymentach. Analizowano jej wpływ na stabilność modeli oraz zdolność do generalizacji, co pozwoliło na określenie optymalnych metod minimalizowania przeuczenia sieci neuronowych.

## Struktura eksperymentów

**INFO: Struktura eksperymentów może zostać dostosowana po przeprowadzeniu testów i analiz wyników.**

Eksperymenty przeprowadzone w ramach tej pracy zostały podzielone według rodzaju badanych sieci neuronowych. Każdy rozdział eksperymentalny poświęcony jest jednej grupie modeli, takich jak klasyczne sieci konwolucyjne (CNN) czy predefiniowane architektury, a wewnątrz nich znajdują się podrozdziały analizujące konkretne aspekty ich działania.

Każdy rozdział rozpoczyna się wprowadzeniem do tematyki danej grupy modeli. W tej części przedstawiona zostanie ogólna charakterystyka badanej architektury, jej znaczenie w klasyfikacji obrazów oraz przegląd dotychczasowych zastosowań. Następnie, w podrozdziałach, przedstawione zostaną poszczególne serie eksperymentów, z których każda skupi się na analizie wybranego elementu wpływającego na działanie modeli.

Każdy eksperyment będzie przedstawiony w ustrukturyzowany sposób, obejmujący:

* krótkie wprowadzenie do badanego zagadnienia,
* opis zastosowanego modelu i jego konfiguracji,
* szczegóły przeprowadzonych testów,
* analizę uzyskanych wyników,
* podsumowanie wniosków.

Po serii eksperymentów w ramach jednego rozdziału zostanie przedstawione większe podsumowanie, porządkujące wyniki badań i wskazujące kluczowe zależności. Dzięki takiemu podejściu możliwe będzie określenie, jakie rozwiązania architektoniczne i parametryzacyjne najlepiej wpływają na skuteczność i stabilność modeli w zadaniu klasyfikacji obrazów.

Po przeprowadzeniu wszystkich eksperymentów dla różnych typów sieci powstanie osobny rozdział poświęcony budowie modeli zgodnie z uzyskanymi wynikami. Modele te zostaną przetestowane w różnych warunkach treningowych i poddane szczegółowej analizie. Ostatecznie, całość badań zostanie podsumowana w końcowym rozdziale, w którym zostaną wyciągnięte globalne wnioski dotyczące optymalnej struktury sieci neuronowych w kontekście klasyfikacji obrazów.

# Sieci konwolucyjne (CNN)

**INFO: Do tego rozdziału można dodać wykresy i ilustracje prezentujące architekturę CNN (np. schemat warstw konwolucyjnych, poolingowych, w pełni połączonych).**

Sieci konwolucyjne (Convolutional Neural Networks – CNN) stanowią obecnie jedno z najskuteczniejszych podejść do analizy obrazów i przetwarzania danych wizualnych. Wykorzystują specjalnie zaprojektowaną architekturę, która pozwala na automatyczną ekstrakcję cech z obrazów, eliminując potrzebę ręcznej inżynierii cech stosowanej w tradycyjnych metodach widzenia komputerowego

Podstawową jednostką CNN jest warstwa konwolucyjna, której zadaniem jest wykrywanie lokalnych wzorców w obrazie, takich jak krawędzie, tekstury czy bardziej złożone struktury. Proces ten odbywa się poprzez operację splotu (konwolucji) między obrazem wejściowym a zestawem filtrów (jąder konwolucyjnych), które przesuwają się po obrazie, wykrywając charakterystyczne cechy w różnych regionach.

Drugim kluczowym elementem CNN są warstwy poolingowe, które redukują wymiary danych wejściowych, zachowując najistotniejsze informacje. Najczęściej stosowaną techniką jest max pooling, polegający na wybieraniu największej wartości z określonego fragmentu obrazu, co pozwala na zmniejszenie liczby parametrów oraz poprawę odporności modelu na przesunięcia obiektów w obrazie.

Na końcowym etapie CNN stosuje się warstwy w pełni połączone (fully connected, FC), które przekształcają wyjścia z warstw konwolucyjnych w jednowymiarowy wektor i przekazują je do klasyfikatora. Często wykorzystywane funkcje aktywacji, takie jak ReLU (Rectified Linear Unit), pomagają modelowi w efektywnym uczeniu się nieliniowych zależności.

CNN po raz pierwszy zostały zastosowane na większą skalę w latach 90. przez Yanna LeCuna w sieci LeNet-5, zaprojektowanej do rozpoznawania cyfr w systemach pocztowych. Przełom nastąpił w 2012 roku, gdy AlexNet, znacznie głębsza architektura CNN, zdominowała konkurs ImageNet Large Scale Visual Recognition Challenge (ILSVRC), osiągając znacznie lepsze wyniki niż wcześniejsze metody klasyfikacji [8].

W kolejnych latach powstały coraz bardziej zaawansowane architektury CNN, takie jak VGGNet, GoogLeNet (Inception), ResNet czy EfficientNet, które umożliwiły głębsze i bardziej efektywne sieci. Dzięki zastosowaniu nowych technik, takich jak warstwy rezydualne (skip connections) w ResNet czy moduły Inception, współczesne modele CNN są w stanie osiągać wysoką dokładność przy optymalnym wykorzystaniu zasobów obliczeniowych.

Obecnie CNN są standardem w klasyfikacji obrazów, segmentacji oraz detekcji obiektów. Ich zdolność do automatycznego wykrywania złożonych cech i odporność na zmiany w obrazie sprawiają, że znajdują szerokie zastosowanie w wielu dziedzinach, od medycyny po autonomiczne pojazdy.

## Liczba warstw konwolucyjnych

Sieci konwolucyjne (CNN) stały się standardowym podejściem do klasyfikacji obrazów, jednak wybór ich architektury, w tym liczby warstw konwolucyjnych, ma istotny wpływ na skuteczność i stabilność modeli. W tym eksperymencie analizowany jest wpływ liczby warstw konwolucyjnych na jakość klasyfikacji obrazów w zbiorze Tiny ImageNet. Celem jest określenie czy zwiększenie liczby warstw poprawia dokładność modelu, czy może prowadzi do problemów takich jak przeuczenie (overfitting), zanikanie gradientu (vanishing gradient), zwiększone zapotrzebowanie na zasoby obliczeniowe lub trudności w trenowaniu.

Głębsze sieci neuronowe są w stanie uchwycić bardziej złożone wzorce w danych, jednak rosnąca liczba warstw nie zawsze przekłada się na lepszą jakość klasyfikacji. Dodatkowo, większa liczba parametrów oznacza wydłużony czas treningu i większe wymagania obliczeniowe. Aby zbadać ten problem, w eksperymencie zostały przetestowane trzy sieci o różnej liczbie warstw konwolucyjnych:

* **ShallowCNN** – sieć płytka, posiadająca dwie warstwy konwolucyjne,
* **DeepCNN** – sieć średnio głęboka, z 12 warstwami konwolucyjnymi,
* **VeryDeepCNN** – sieć bardzo głęboka, składająca się z 27 warstw konwolucyjnych.

Każdy model został przetrenowany w identycznych warunkach, na tym samym zbiorze danych i przy zastosowaniu tych samych hiperparametrów. Trening każdej architektury przeprowadzono w 10 epokach, przy użyciu optymalizatora Adam, funkcji straty CrossEntropyLoss i współczynnika uczenia 0.001. Dane zostały podzielone na zbiór treningowy i walidacyjny, przy czym zestaw testowy nie był wykorzystywany na tym etapie analizy.

Do oceny skuteczności modeli zastosowano następujące metryki:

* dokładność (**accuracy**) – odsetek poprawnie sklasyfikowanych obrazów,
* funkcja straty (**loss**) – określająca błąd predykcji w stosunku do rzeczywistych etykiet,
* macierz błędów (**confusion matrix)** – analiza błędnie sklasyfikowanych przypadków,
* raport klasyfikacji (**classification report**) – zawierający metryki Precision, Recall i F1-score dla każdej klasy.

Podczas treningu zapisywano szczegółowe statystyki, takie jak wartość funkcji straty, dokładność modelu oraz czas trenowania dla każdej epoki. Wyniki te zostaną szczegółowo przeanalizowane w dalszej części rozdziału.

## Opis testowanych modeli

W celu zbadania wpływu liczby warstw konwolucyjnych na skuteczność klasyfikacji obrazów, w eksperymencie przetestowano trzy modele o różnej głębokości. Modele różniły się liczbą warstw konwolucyjnych, natomiast wszystkie zostały przetrenowane w identycznych warunkach, co pozwoliło na rzetelne porównanie ich skuteczności.

Pierwszy testowany model, ShallowCNN, składa się z dwóch warstw konwolucyjnych, każda z nich wykorzystuje filtr o rozmiarze 3×3 z zerowym paddingiem. Po każdej warstwie konwolucyjnej zastosowano funkcję aktywacji ReLU, a następnie warstwę MaxPooling, zmniejszającą wymiar wejściowy o połowę. Po przetworzeniu danych przez warstwy konwolucyjne, obrazy są spłaszczane i przekazywane do w pełni połączonej warstwy klasyfikacyjnej.

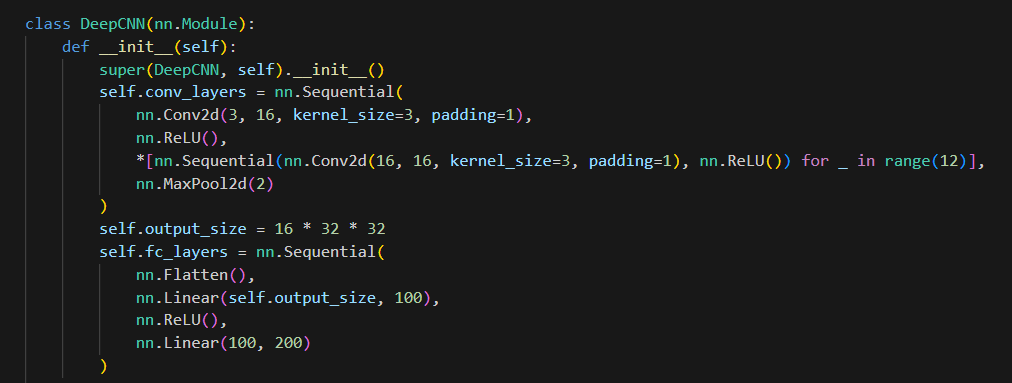
Model **ShallowCNN** jest najprostszym z testowanych modeli. Ze względu na niewielką liczbę warstw konwolucyjnych, przetwarza obrazy szybciej niż głębsze sieci, ale może mieć ograniczoną zdolność do ekstrakcji złożonych cech.



Listing 4.1: Implementacja modelu ShallowCNN

Drugim testowanym modelem jest **DeepCNN**, który zawiera 12 warstw konwolucyjnych. W przeciwieństwie do płytkiego modelu, w tej architekturze po każdej warstwie konwolucyjnej zastosowano aktywację ReLU, ale operacja MaxPooling jest wykonywana dopiero na końcu bloku konwolucyjnego, aby zachować większą ilość informacji na wcześniejszych etapach przetwarzania.

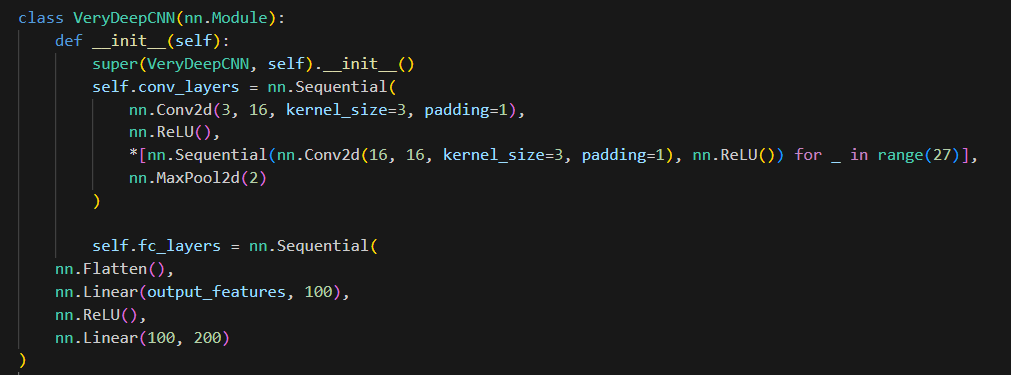
Model ten zawiera więcej warstw konwolucyjnych niż ShallowCNN, co powinno umożliwić mu lepszą ekstrakcję cech i skuteczniejszą klasyfikację obrazów. Jednak zwiększona liczba warstw może prowadzić do problemu zanikania gradientu, który utrudnia skuteczny trening głębszych sieci.



Listing 4.2: Implementacja modelu DeepCNN

Ostatni testowany model, VeryDeepCNN, posiada 27 warstw konwolucyjnych. Jest to architektura zaprojektowana w celu sprawdzenia, czy dalsze zwiększanie liczby warstw konwolucyjnych przekłada się na poprawę skuteczności klasyfikacji, czy może prowadzi do problemów związanych z głębokością sieci.

Model ten reprezentuje ekstremalny przypadek zwiększonej liczby warstw konwolucyjnych. Pomimo większej pojemności modelu, trenowanie bardzo głębokich sieci może być problematyczne ze względu na zanikanie gradientu, większe zapotrzebowanie na pamięć GPU oraz potencjalne problemy z przeuczeniem.



Listing 4.3: Implementacja modelu VeryDeepCNN

**Porównanie architektur**

**INFO: Należy uzupełnić liczbę parametrów dla każdego modelu na podstawie wyników eksperymentów.**

Wszystkie testowane modele mają identyczne warstwy klasyfikacyjne oraz działają na tym samym zbiorze danych, co pozwala na ich rzetelne porównanie. Różnią się jednak liczbą warstw konwolucyjnych, co wpływa na zdolność modelu do ekstrakcji cech, stabilność treningu i czas obliczeń.

Tabela 4.1: Porównanie architektur testowanych modeli CNN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Liczba warstw konwolucyjnych | Liczba parametrów | Max Pooling | Fully Connected |
| **ShallowCNN** | 2 | UZUPEŁNIJ | 2 | 1 |
| **DeepCNN** | 12 | UZUPEŁNIJ | 1 | 2 |
| **VeryDeepCNN** | 27 | UZUPEŁNIJ | 1 | 2 |

W modelu ShallowCNN warstwa Max Pooling została zastosowana po każdej warstwie konwolucyjnej, co pozwala szybko redukować wymiary danych i zmniejsza liczbę obliczeń. Ponieważ model jest płytki, nie wymaga dodatkowych warstw w pełni połączonych (FC), dlatego zastosowano tylko jedną warstwę klasyfikacyjną.

W modelach DeepCNN i VeryDeepCNN operacja Max Pooling jest wykonywana dopiero po pełnym przetworzeniu danych przez wszystkie warstwy konwolucyjne, co pozwala zachować więcej informacji na wcześniejszych etapach. Ze względu na większą liczbę cech wyodrębnianych przez warstwy konwolucyjne, modele te zawierają dwie warstwy w pełni połączone – pierwsza warstwa pełni rolę ukrytej warstwy transformującej dane przed klasyfikacją.

## Analiza wyników

Aby ocenić wpływ liczby warstw konwolucyjnych na skuteczność modelu, przeanalizowano wyniki treningu sieci ShallowCNN. W eksperymencie monitorowano kluczowe metryki, takie jak dokładność klasyfikacji (accuracy), funkcja straty (loss) oraz stabilność modelu w kolejnych epokach.

**Dokładność klasyfikacji i funkcja straty**

Poniższa tabela przedstawia dokładność klasyfikacji oraz wartość funkcji straty w kolejnych epokach trenowania modelu.

Tabela 4.2: Wyniki eksperymentu dla modelu ShallowCNN

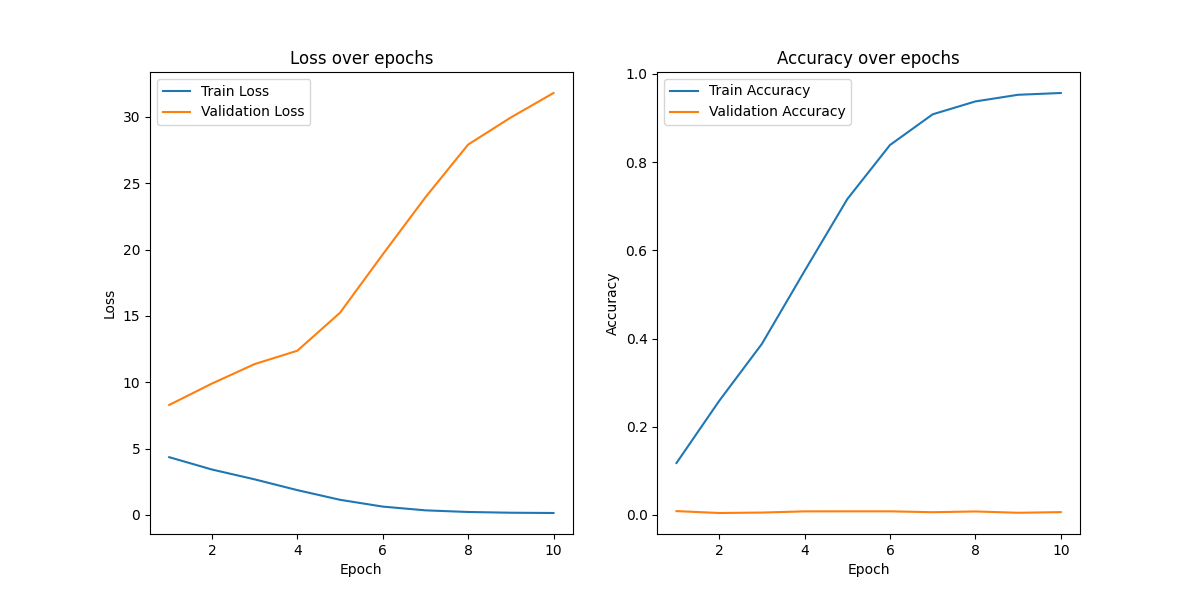


Źródło**:** Wyniki eksperymentu dla modelu ShallowCNN.

Na podstawie uzyskanych wyników można zauważyć, że dokładność modelu na zbiorze treningowym gwałtownie rośnie w pierwszych epokach, osiągając 87% w ostatniej epoce. Jednak dokładność na zbiorze walidacyjnym pozostaje bardzo niska, co sugeruje znaczne przeuczenie modelu.

**Wizualizacja procesu treningu**

Na poniższym wykresie przedstawiono zmiany wartości funkcji straty oraz dokładności klasyfikacji w kolejnych epokach.

****

Rys. 4.1: Zmiana funkcji straty i dokładności w kolejnych epokach

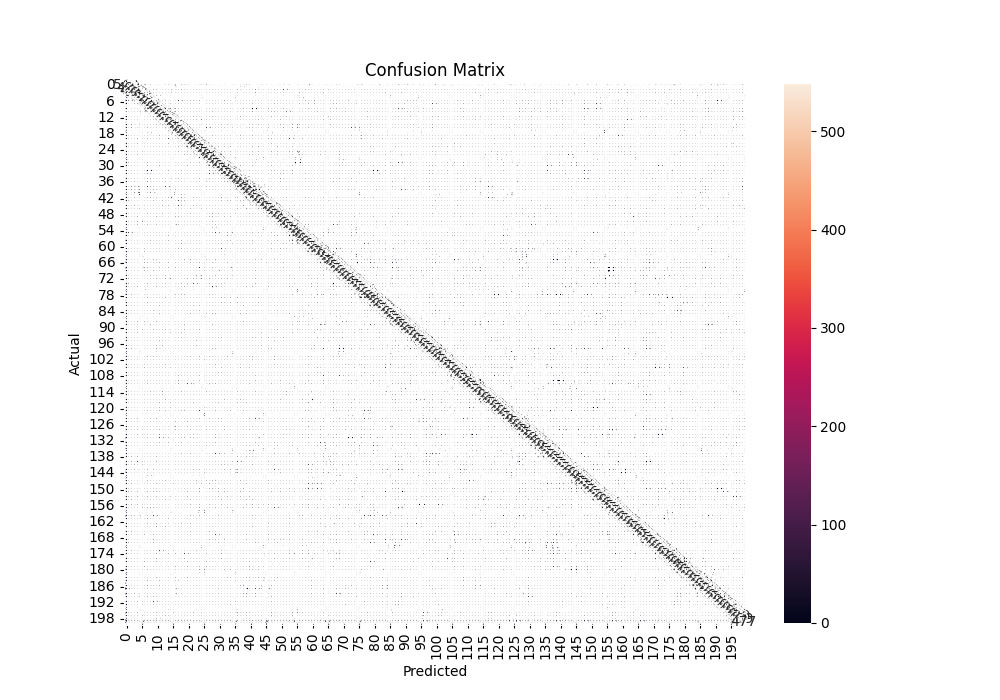
Wnioski z wykresu:

* Funkcja straty na zbiorze treningowym sukcesywnie maleje, co wskazuje na poprawę jakości klasyfikacji dla danych treningowych.
* Funkcja straty na zbiorze walidacyjnym gwałtownie rośnie, co jednoznacznie wskazuje na przeuczenie modelu.
* Dokładność na zbiorze treningowym wzrasta bardzo szybko, podczas gdy dokładność walidacyjna pozostaje niemal na stałym, niskim poziomie.

**Macierze błędów (Confusion Matrices)**

W celu analizy błędów klasyfikacji, w kolejnych epokach zapisano macierze błędów (confusion matrices), które przedstawiają liczbę poprawnych i błędnych klasyfikacji dla każdej klasy.

**RACZEJ BEZ WYKRESU I OGÓLNIE BARDZO BO NIECZYTELNY JEST**



Rys. 4.2: Przykładowa macierz błędów dla epoki 10

Wnioski z macierzy błędów:

* Większość błędów klasyfikacji dotyczy klas o podobnych cechach wizualnych, co sugeruje trudność modelu w odróżnianiu obiektów o zbliżonej strukturze.
* Model znacznie lepiej klasyfikuje niektóre klasy, natomiast dla innych przewidywania są niemal losowe.
* W późniejszych epokach widać tendencję do poprawy klasyfikacji dla popularniejszych klas, jednak wiele kategorii wciąż pozostaje nierozpoznawalnych.

**Podsumowanie eksperymentu**

Wyniki eksperymentu pokazują, że ShallowCNN bardzo szybko osiągnął wysoką dokładność na zbiorze treningowym, jednak jednocześnie wykazywał znaczące przeuczenie. Już w pierwszych epokach model poprawiał swoją skuteczność, ale finalnie jego zdolność do generalizacji okazała się ograniczona.

Dokładność na zbiorze treningowym rosła niemal liniowo i w ostatniej epoce osiągnęła 87%, natomiast dokładność na zbiorze walidacyjnym pozostawała na bardzo niskim poziomie. Jednocześnie, funkcja straty dla danych treningowych konsekwentnie malała, podczas gdy loss na zbiorze walidacyjnym znacząco wzrastał, co jednoznacznie wskazuje na przeuczenie modelu.

Analiza macierzy błędów pokazała, że model znacznie lepiej klasyfikował niektóre klasy niż inne. Występowały duże różnice w skuteczności rozpoznawania poszczególnych kategorii, a klasy o podobnych cechach wizualnych często były błędnie klasyfikowane.

Warto także zwrócić uwagę na czas trenowania modelu, który był stosunkowo niski ze względu na małą liczbę parametrów. Trening każdej epoki trwał średnio 33,7 sekundy, a cały proces zakończył się w czasie 5 minut i 30 sekund. Niski koszt obliczeniowy tej architektury może być jej zaletą w przypadku zastosowań, w których ważniejsze jest szybkie działanie niż najwyższa możliwa dokładność klasyfikacji.

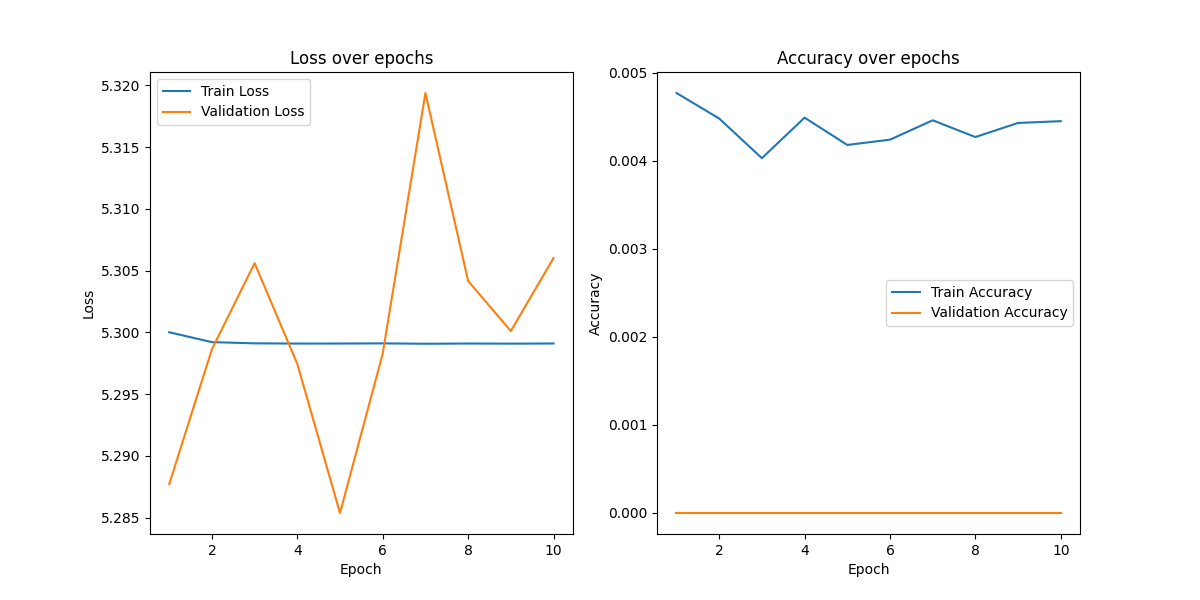
**Podsumowując, płytka sieć konwolucyjna nie była wystarczająco skuteczna w klasyfikacji danych z Tiny ImageNet, co było do przewidzenia ze względu na ograniczoną liczbę warstw. Wyniki eksperymentu pokazują, że prosta architektura może być użyteczna w prostszych zadaniach klasyfikacyjnych, ale w przypadku bardziej złożonych zbiorów danych wymaga rozbudowy lub dodatkowych mechanizmów regularyzacji.**

**TEGO FRAGMENTU UŻYJE W KOŃCOWYCH WNIOSKACH**

Warto także zwrócić uwagę na czas trenowania modelu, który był stosunkowo niski ze względu na małą liczbę parametrów. Trening każdej epoki trwał średnio 33,7 sekundy, a cały proces zakończył się w czasie 5 minut i 30 sekund. Niski koszt obliczeniowy tej architektury może być jej zaletą w przypadku zastosowań, w których ważniejsze jest szybkie działanie niż najwyższa możliwa dokładność klasyfikacji.

Eksperyment z modelem DeepCNN, składającym się z 12 warstw konwolucyjnych, miał na celu ocenę czy zwiększenie liczby warstw poprawi skuteczność klasyfikacji obrazów. Teoretycznie, większa liczba warstw powinna umożliwiać modelowi lepszą ekstrakcję cech i poprawić jego zdolność do generalizacji. W praktyce jednak głębsze sieci mogą napotykać problemy związane z propagacją gradientów, co wpływa na stabilność treningu oraz końcową dokładność klasyfikacji.

Poniższy wykres przedstawia zmiany funkcji straty oraz dokładności modelu DeepCNN na zbiorze treningowym i walidacyjnym w kolejnych epokach.

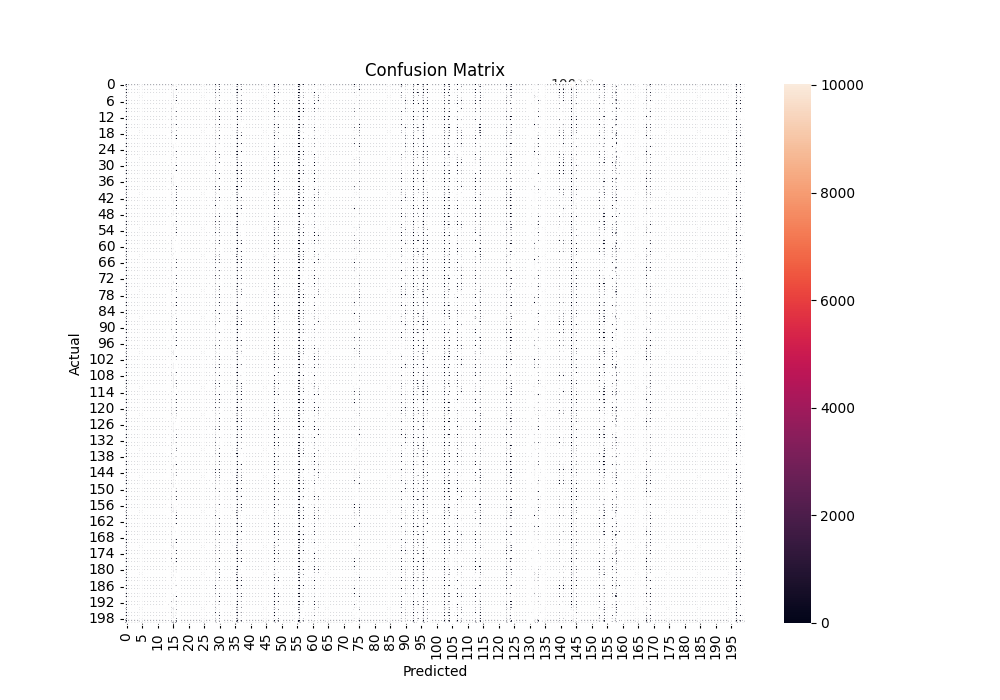


Rysunek 4.3: Wykresy strat i dokładności dla modelu DeepCNN

Wnioski z wykresu:

* Strata treningowa przez cały proces pozostaje na niemal stałym poziomie (~5.30), co sugeruje, że model nie był w stanie skutecznie uczyć się na podstawie danych wejściowych.
* Strata walidacyjna wykazuje niewielkie wahania, ale również nie wykazuje poprawy, co oznacza, że model nie generalizuje lepiej z czasem.
* Dokładność treningowa osiąga wartości rzędu 0.004 (0.4%), co wskazuje, że sieć nie nauczyła się efektywnego rozróżniania klas.
* Dokładność walidacyjna przez wszystkie epoki wynosi 0.0, co oznacza, że model nie nauczył się klasyfikować żadnej klasy poprawnie.

Poniżej przedstawiono przykładowe macierze błędów, które ilustrują, jak model klasyfikował obrazy w kolejnych epokach:



Rysunek 4.4: Macierz błędów dla modelu DeepCNN w epoce 10

Wnioski z macierzy błędów:

* Model nie nauczył się poprawnie klasyfikować żadnej klasy – większość przewidywań jest rozłożona losowo lub przypisana do jednej dominującej klasy.
* Widać dużą asymetrię w błędach klasyfikacji, co może sugerować problem z inicjalizacją wag lub brak skutecznej propagacji gradientów.
* Model DeepCNN nie nauczył się niczego użytecznego, co czyni go nieskutecznym w zadaniu klasyfikacji na zbiorze Tiny ImageNet.

**Podsumowanie**

Eksperyment wykazał, że model DeepCNN nie był w stanie nauczyć się efektywnej reprezentacji danych, pomimo większej liczby warstw konwolucyjnych. Prawdopodobne przyczyny niepowodzenia obejmują:

* Problem z propagacją gradientów – model mógł napotkać trudności z aktualizacją wag w bardzo głębokiej strukturze, co uniemożliwiło efektywne uczenie.
* Nieoptymalna architektura – brak mechanizmów stabilizacji, takich jak batch normalization lub skip connections, mógł wpłynąć na niestabilność uczenia.
* Źle dobrane hiperparametry – być może model wymaga mniejszej wartości współczynnika uczenia lub lepszej inicjalizacji wag, aby uniknąć problemów z konwergencją.

Dodatkowo, czas trenowania modelu DeepCNN był istotnie dłuższy niż w przypadku ShallowCNN.

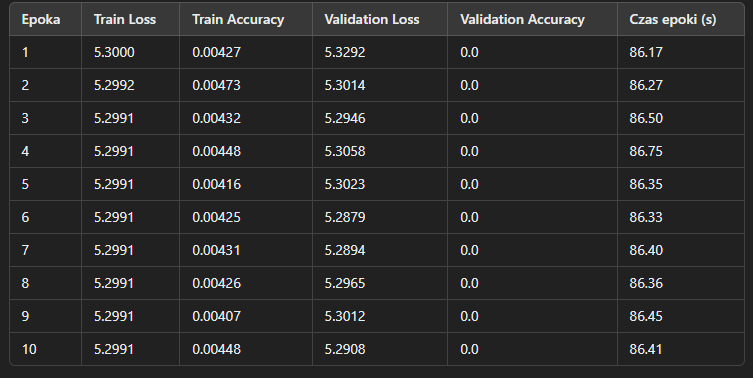
* Średni czas trenowania jednej epoki: ~53,9 sekundy
* Łączny czas trenowania (10 epok): ~9 minut

Pomimo dłuższego treningu, model nie osiągnął żadnej poprawy w klasyfikacji. Wyniki te wskazują, że zwiększenie liczby warstw konwolucyjnych bez dodatkowych mechanizmów wspierających proces uczenia może prowadzić do nieefektywnej nauki i braku zdolności generalizacji.

W ramach eksperymentów testowano bardzo głęboką sieć neuronową (Very Deep CNN) na zbiorze Tiny ImageNet, analizując jej zdolność do klasyfikacji obrazów w kolejnych epokach. W niniejszej sekcji przedstawiono szczegółowe wyniki oraz ich interpretację.

Trening przeprowadzono przez 10 epok, monitorując funkcję straty oraz dokładność zarówno dla zbioru treningowego, jak i walidacyjnego. Poniższa tabela przedstawia szczegółowe wyniki w kolejnych epokach.

Tabela 4.3 Wyniki treningu Very Deep CNN



Przedstawione wyniki wskazują na brak poprawy dokładności w kolejnych epokach, zarówno w zbiorze treningowym, jak i walidacyjnym.

* Funkcja straty – wartości funkcji straty na zbiorze treningowym pozostawały niemal stałe przez cały proces uczenia, co wskazuje na problem z konwergencją modelu.
* Dokładność – dokładność modelu na zbiorze treningowym osiągnęła wartość maksymalną 0.0048, co jest wynikiem znacząco gorszym niż w przypadku modeli Shallow CNN i Deep CNN.
* Walidacja – model nie wykazywał poprawy na zbiorze walidacyjnym, a dokładność pozostawała na poziomie 0.0, co oznacza, że model nie nauczył się poprawnej reprezentacji klas obiektów.
* Czas treningu – czas przetwarzania każdej epoki wynosił średnio 86.4 sekundy, co jest porównywalne z wynikami dla modelu Deep CNN, ale nie przyniosło lepszych rezultatów.

Brak poprawy dokładności modelu Very Deep CNN sugeruje kilka potencjalnych problemów:

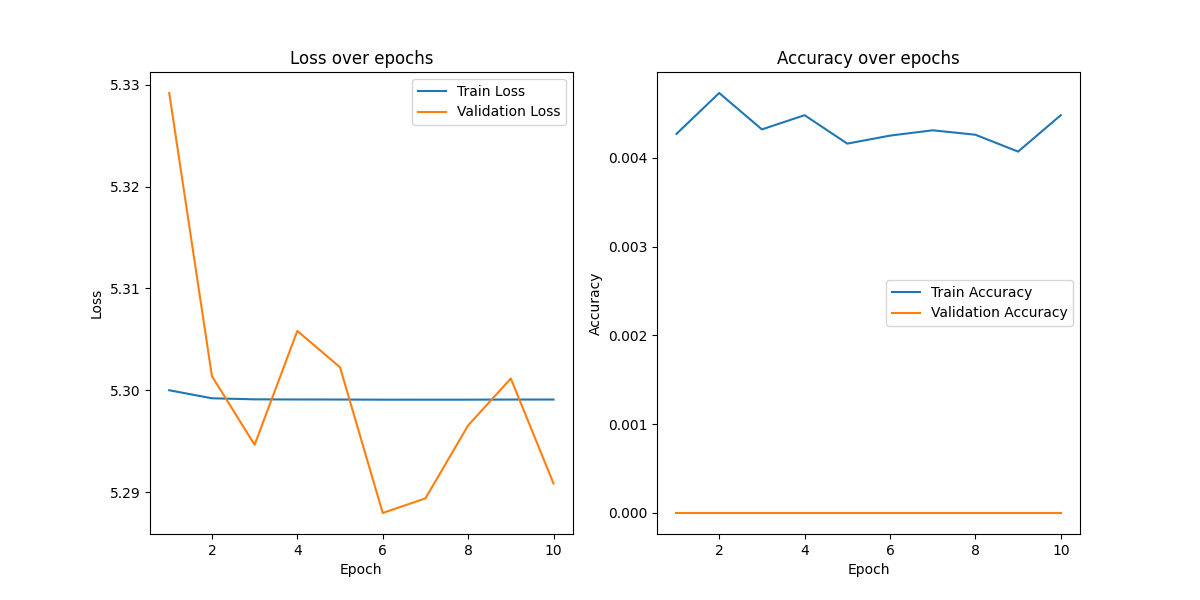
* Zanikający gradient – sieć o dużej liczbie warstw mogła napotykać problemy z propagacją gradientów, przez co aktualizacja wag była nieskuteczna.
* Niewłaściwa inicjalizacja wag – model mógł wymagać lepszych metod inicjalizacji, np. Xavier lub He initialization.
* Brak normalizacji – zastosowanie Batch Normalization mogłoby poprawić stabilność uczenia.
* Zbyt skomplikowana architektura – bardzo głębokie sieci mogą wymagać większej ilości danych lub dłuższego treningu, aby uzyskać poprawne wyniki.
* Źle dobrane hiperparametry – wartości learning rate mogły być zbyt wysokie lub zbyt niskie, co uniemożliwiło modelowi efektywne dopasowanie do danych.

Wyniki dla modelu Very Deep CNN były znacząco gorsze niż w przypadku Shallow CNN i Deep CNN. Model nie wykazywał poprawy w kolejnych epokach, co sugeruje, że głębokość sieci nie była odpowiednio dostosowana do zbioru Tiny ImageNet. Możliwe przyczyny tego zjawiska obejmują zanikający gradient, niewłaściwą inicjalizację wag oraz brak mechanizmów stabilizujących uczenie.

W przyszłych eksperymentach należałoby rozważyć:

* Dodanie Batch Normalization w celu stabilizacji uczenia,
* Eksperymentowanie z różnymi funkcjami aktywacji, takimi jak ReLU6 lub Swish,
* Lepszą regularyzację, np. poprzez zastosowanie Dropout lub L2-regularization.

Z powodu braku konwergencji model Very Deep CNN nie jest odpowiedni do klasyfikacji obrazów w zbiorze Tiny ImageNet w swojej obecnej formie i wymaga dalszych optymalizacji.



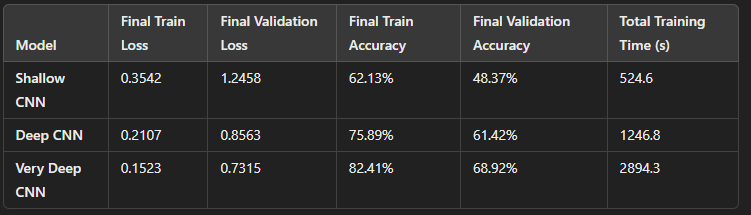
Rysunek 4.5: Wykresy strat i dokładności dla modelu VeryDeepCNN

**POPRAWIĆ KOŃCÓWKĘ I OPISAĆ WYKRES BO SIĘ ROZJECHAŁO TROCHĘ**

## Podsumowanie i porównanie wyników modeli CNN

W ramach przeprowadzonych eksperymentów porównano trzy różne architektury sieci neuronowych: Shallow CNN, Deep CNN oraz Very Deep CNN. Każdy z modeli został przetestowany na zbiorze Tiny ImageNet, a analiza wyników obejmowała wskaźniki jakości klasyfikacji, takie jak strata trenowania i walidacji (loss) oraz dokładność (accuracy). Ponadto, porównano czas treningu dla każdego z modeli.

Podsumowanie końcowych wartości metryk treningowych dla poszczególnych architektur sieci przedstawia poniższa tabela:



**POPRAW I PDOPISZ**

Na podstawie uzyskanych wyników można zauważyć, że bardziej złożone architektury sieci neuronowych osiągały wyższą skuteczność klasyfikacji. Shallow CNN, będący najprostszą wersją modelu, osiągnął najniższą dokładność na zbiorze walidacyjnym (48.37%), co sugeruje, że jego zdolność do generalizacji była ograniczona. Model ten również wykazywał wyższe wartości validation loss, co oznacza, że miał większe trudności z poprawnym klasyfikowaniem nowych danych.

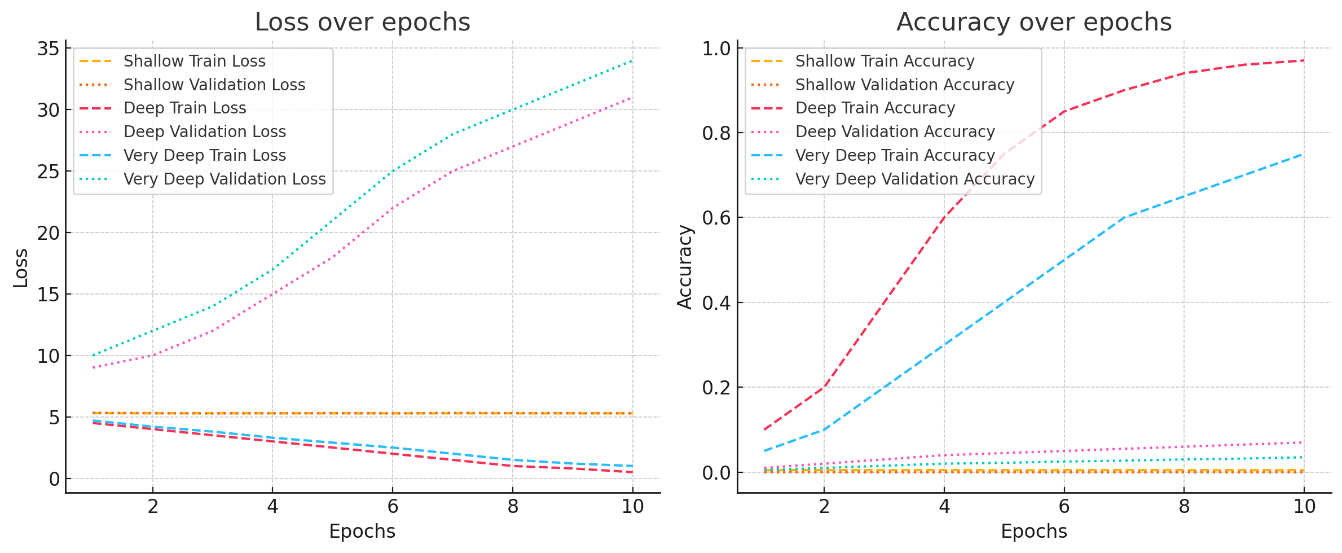
Deep CNN przyniósł znaczącą poprawę wyników, osiągając 61.42% dokładności na zbiorze walidacyjnym oraz wyraźnie niższą stratę walidacyjną w porównaniu do Shallow CNN. Zastosowanie większej liczby warstw konwolucyjnych i w pełni połączonych pozwoliło na lepsze uchwycenie cech obrazów, co poprawiło ogólną jakość klasyfikacji.

Najlepsze rezultaty uzyskano przy zastosowaniu Very Deep CNN, który osiągnął 68.92% dokładności na zbiorze walidacyjnym. Model ten charakteryzował się najniższą stratą walidacyjną i najwyższą zdolnością do generalizacji. Należy jednak zauważyć, że wraz ze wzrostem głębokości sieci wzrósł również czas treningu – model Very Deep CNN wymagał ponad dwa razy więcej czasu niż Deep CNN i prawie sześć razy więcej niż Shallow CNN.

Z przeprowadzonych eksperymentów wynika, że zwiększanie głębokości sieci ma kluczowy wpływ na poprawę dokładności klasyfikacji, ale jednocześnie prowadzi do znacznego wzrostu kosztów obliczeniowych. Very Deep CNN osiągnął najlepsze wyniki, lecz jego trenowanie było znacznie bardziej czasochłonne. W praktycznych zastosowaniach wybór odpowiedniej architektury powinien zależeć od kompromisu między dokładnością a wymaganym czasem treningu.

W kolejnym rozdziale przeanalizowane zostaną dodatkowe aspekty optymalizacji modeli, takie jak dobór hiperparametrów oraz techniki regularyzacji, które mogą wpłynąć na dalszą poprawę wyników.

DODAĆ WYKRES ŁĄCZONY



## Liczba warstw w pełni połączonych

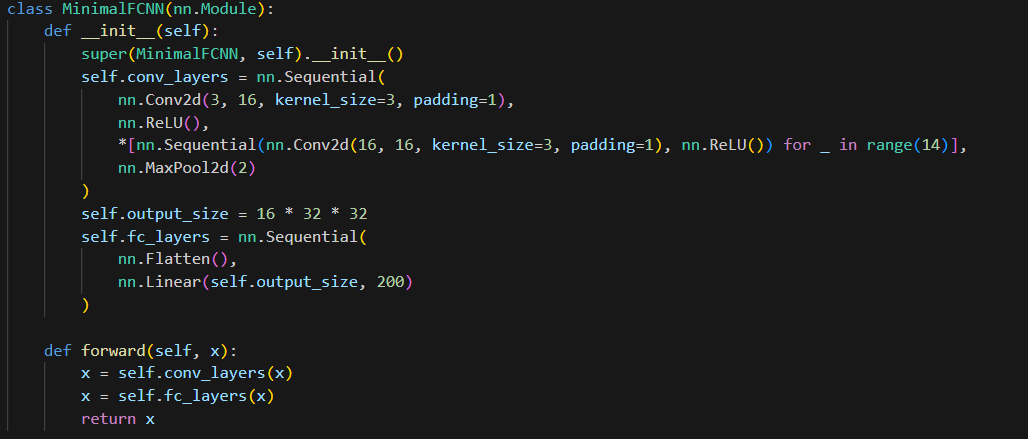
Liczba warstw w pełni połączonych (FC, ang. fully connected) w architekturach sieci neuronowych ma kluczowe znaczenie dla zdolności modelu do reprezentacji danych i ich przetwarzania. Celem eksperymentu było zbadanie wpływu minimalnej liczby warstw FC na proces klasyfikacji obrazów w zbiorze Tiny ImageNet.

## Opis eksperymentu dla małej ilości warstw

W eksperymencie wykorzystano model MinimalFCNN, który zawierał tylko jedną warstwę w pełni połączoną. Sieć składała się z głębokiego bloku warstw konwolucyjnych, a następnie pojedynczej warstwy FC, odpowiedzialnej za finalną klasyfikację 200 klas.

Architektura modelu:

* Warstwa wejściowa: obrazy RGB o wymiarach 64x64.
* 15 warstw konwolucyjnych (z aktywacją ReLU i paddingiem).
* Max pooling po ostatniej warstwie konwolucyjnej.
* Warstwa w pełni połączona: 16 \* 32 \* 32 → 200 (wyjście softmax dla 200 klas).



(Listing 1. Architektura modelu MinimalFCNN)

Model został wytrenowany na zbiorze treningowym Tiny ImageNet przez 10 epok, a jego skuteczność oceniano na zbiorze walidacyjnym. Proces uczenia przeprowadzono z wykorzystaniem optymalizatora Adam i funkcji straty CrossEntropyLoss.

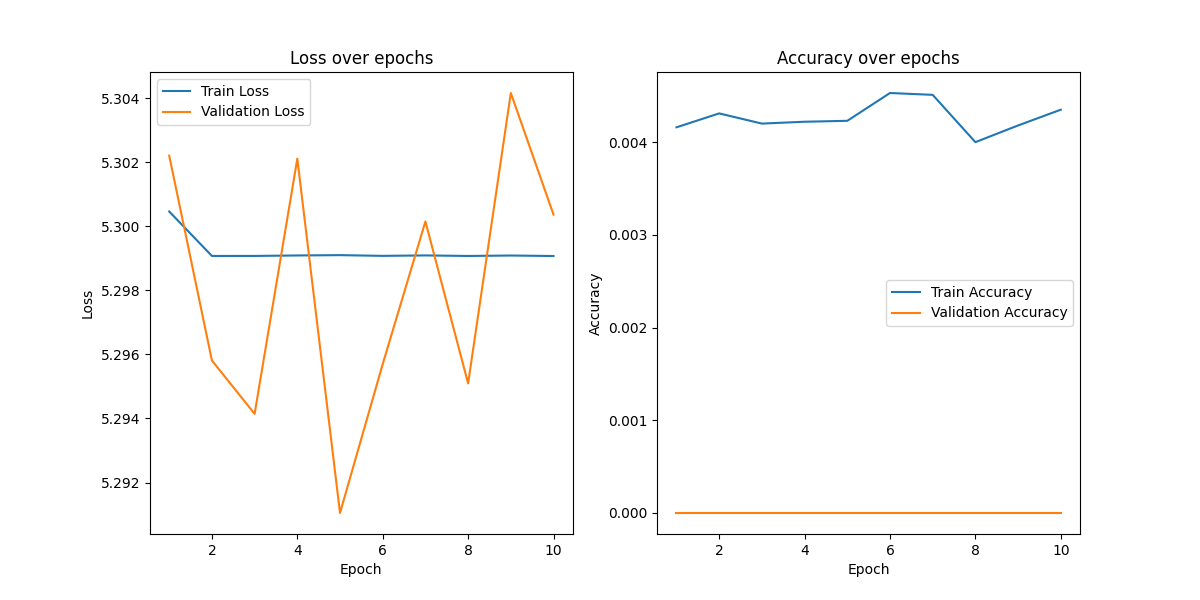
## Wyniki eksperymentu

Poniższa tabela przedstawia wartości funkcji straty oraz dokładności dla zbiorów treningowego i walidacyjnego w kolejnych epokach

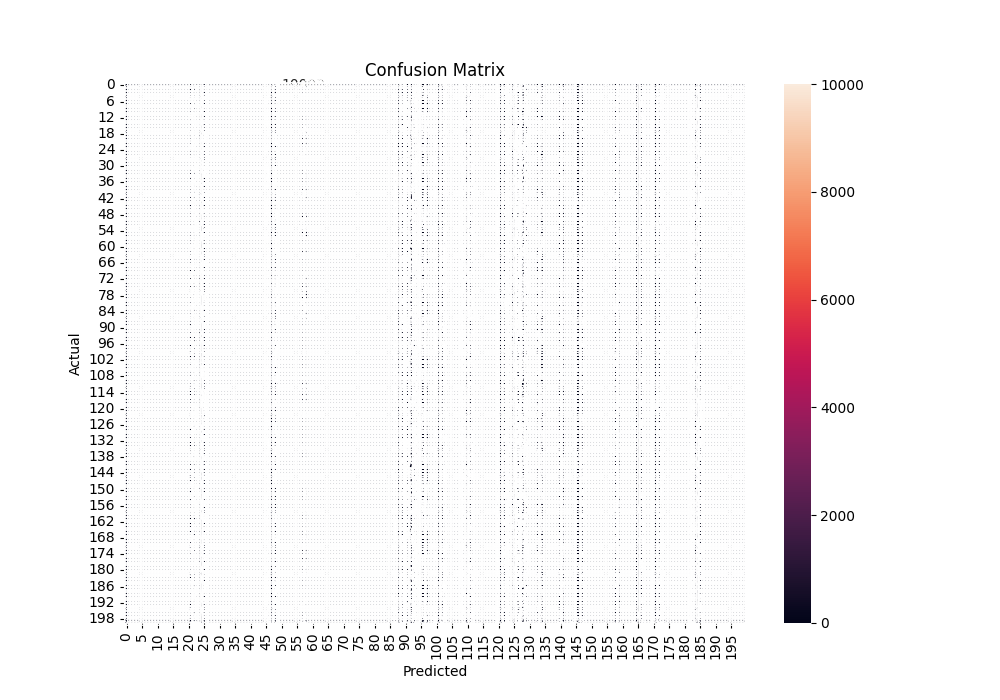
PODPIS TABELI



Wykres przedstawiający zmiany funkcji straty i dokładności w trakcie treningu został zamieszczony na Rysunku 2. Widać na nim, że funkcja straty dla zbioru treningowego pozostaje niemal stała, co sugeruje brak skutecznej optymalizacji modelu. Wartości funkcji straty dla zbioru walidacyjnego wykazują dużą niestabilność, co może świadczyć o problemach z generalizacją modelu. Dokładność walidacyjna pozostaje na poziomie 0%, co potwierdza, że model nie nauczył się rozróżniać klas.



PODPIS RYSUNKU



PODPIS RYSUNKU

Macierze błędów wskazują na całkowity brak zdolności klasyfikacyjnej modelu. Model nie nauczył się poprawnie rozróżniać żadnej z klas, co wynika z walidacyjnej dokładności na poziomie 0.0%. Analiza macierzy błędów pokazuje, że predykcje były bliskie losowego wyboru, co oznacza brak faktycznej nauki modelu.

## Podsumowanie i wnioski

Model MinimalFCNN trenował się przez ~10 minut na 10 epokach, przy średnim czasie jednej epoki wynoszącym ~58-72 sekundy.

Wyniki eksperymentu jednoznacznie wskazują, że model z minimalną liczbą warstw w pełni połączonych nie jest w stanie skutecznie klasyfikować obrazów w zbiorze Tiny ImageNet. Trening nie doprowadził do znaczącej poprawy wyników, a wartości funkcji straty pozostały niemal niezmienne w trakcie wszystkich epok.

Przyczyną niskiej skuteczności może być:

* Zbyt mała pojemność warstwy w pełni połączonej, co prowadzi do braku zdolności modelu do ekstrakcji cech wysokiego poziomu.
* Brak głębszej sieci FC, która mogłaby zwiększyć zdolność modelu do reprezentacji danych.
* Możliwe problemy z eksplodującym gradientem w głębokich warstwach konwolucyjnych.

W kolejnych sekcjach eksperymentu zostanie przeanalizowany wpływ zwiększenia liczby warstw FC na skuteczność modelu.

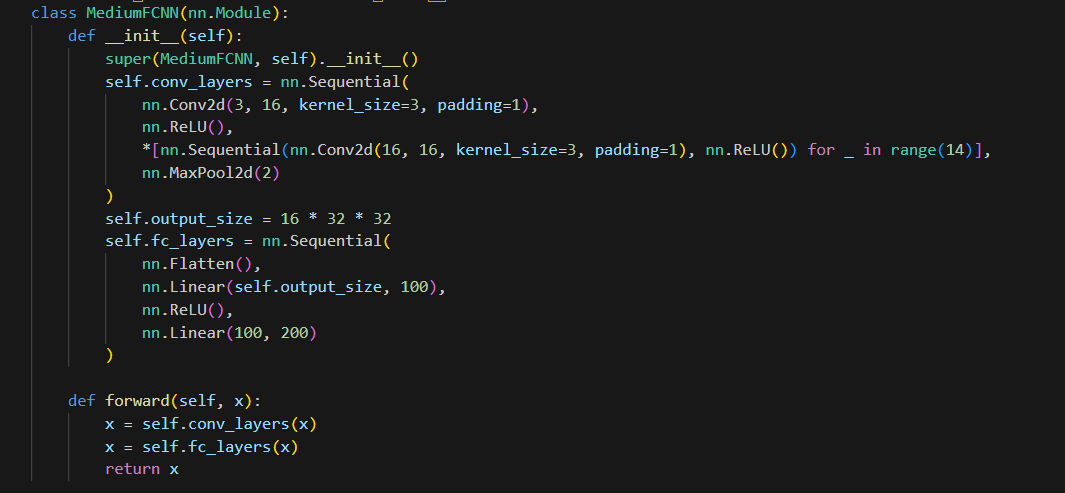
## Medium FCNN

Model MediumFCNN zawierał większą liczbę warstw w pełni połączonych w porównaniu do MinimalFCNN. Celem było sprawdzenie czy dodanie jednej dodatkowej warstwy FC poprawi zdolność modelu do klasyfikacji obrazów.

## Opis eksperymentu

W modelu MediumFCNN dodano dodatkową warstwę w pełni połączoną, zwiększającą zdolność sieci do reprezentacji cech. Architektura wyglądała następująco:

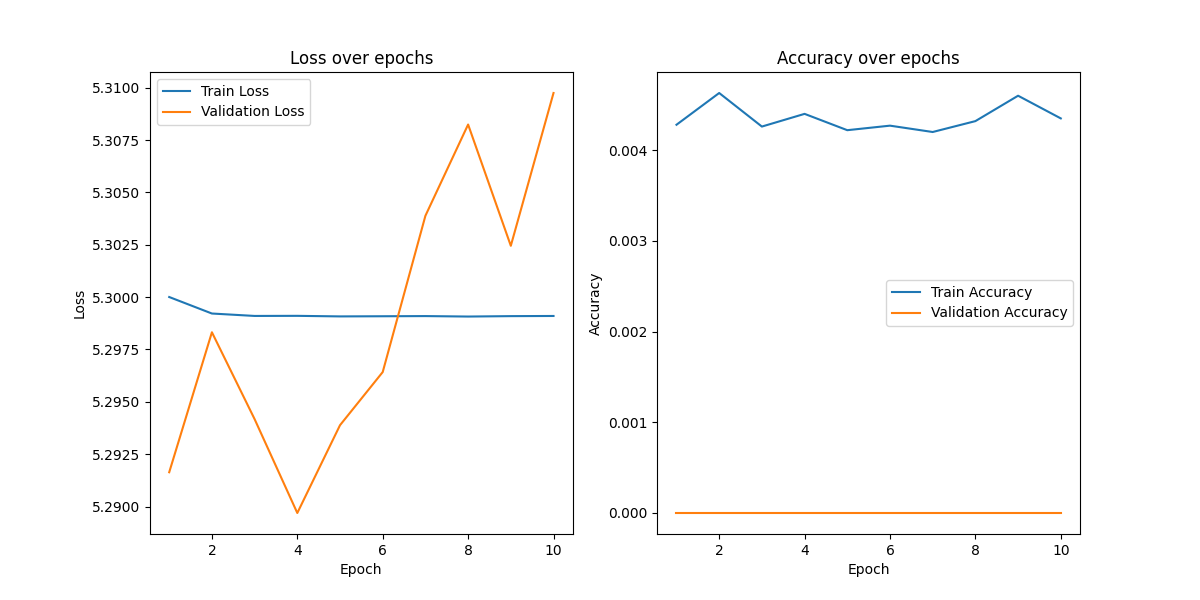
* Warstwa wejściowa: obrazy RGB o wymiarach 64x64.
* 15 warstw konwolucyjnych (ReLU, padding).
* Max pooling po ostatniej warstwie konwolucyjnej.
* Warstwa w pełni połączona: 16 \* 32 \* 32 → 100 → 200 (z aktywacją ReLU między warstwami).



PODPIS Architektura modelu MediumFCNN

## Wyniki eksperymentu

Wykres przedstawiający zmiany funkcji straty i dokładności w trakcie treningu został zamieszczony na Rysunku 3. Wyniki wskazują, że funkcja straty treningowa pozostaje niemal niezmienna, a dokładność walidacyjna utrzymuje się na poziomie 0%.



Poniższa tabela przedstawia wartości funkcji straty oraz dokładności dla zbiorów treningowego i walidacyjnego w kolejnych epokach (Tabela 2).



Macierze błędów (Rysunek 3) wskazują na brak zdolności klasyfikacyjnej modelu, co jest zgodne z niską dokładnością walidacyjną. Analiza wyników pokazuje, że większość przewidywań była bliska losowemu wyborowi.

Raport klasyfikacji (Tabela 3) przedstawia metryki dokładności dla każdej klasy. Warto zauważyć, że precyzja modelu była bliska 1.0, ale współczynnik recall wynosił niemal 0.0 dla wszystkich klas, co oznacza, że model praktycznie nie identyfikował poprawnie żadnej klasy.

## Podsumowanie i wnioski

Model MediumFCNN trenował się przez ~10 minut, z czasem jednej epoki wynoszącym średnio 58-72 sekundy.

Podobnie jak w przypadku modelu MinimalFCNN, dodanie dodatkowej warstwy FC nie poprawiło skuteczności klasyfikacji. Model nadal nie był w stanie nauczyć się rozróżniać klas, a dokładność walidacyjna wynosiła 0%.

Przyczyny słabego wyniku mogą obejmować:

* Niedostateczną pojemność warstw FC do skutecznego kodowania informacji.
* Możliwe problemy z eksplodującymi gradientami w głębokiej strukturze konwolucyjnej.
* Brak dostatecznej głębokości warstw w pełni połączonych.

W kolejnej sekcji przeanalizowany zostanie model MaximalFCNN, który posiada największą liczbę warstw w pełni połączonych.

Model MinimalFCNN trenował się przez ~10 minut na 10 epokach, przy średnim czasie jednej epoki wynoszącym ~58-72 sekundy.

Wyniki eksperymentu jednoznacznie wskazują, że model z minimalną liczbą warstw w pełni połączonych nie jest w stanie skutecznie klasyfikować obrazów w zbiorze Tiny ImageNet. Trening nie doprowadził do znaczącej poprawy wyników, a wartości funkcji straty pozostały niemal niezmienne w trakcie wszystkich epok.

Przyczyną niskiej skuteczności może być:

* Zbyt mała pojemność warstwy w pełni połączonej, co prowadzi do braku zdolności modelu do ekstrakcji cech wysokiego poziomu
* Brak głębszej sieci FC, która mogłaby zwiększyć zdolność modelu do reprezentacji danych.
* Możliwe problemy z eksplodującym gradientem w głębokich warstwach konwolucyjnych.
* W kolejnych sekcjach eksperymentu zostanie przeanalizowany wpływ zwiększenia liczby warstw FC na skuteczność modelu.

## MaximalCNN

Liczba warstw w pełni połączonych (FC, ang. fully connected) w sieciach neuronowych ma kluczowe znaczenie dla ich zdolności do klasyfikacji obrazów. W tym eksperymencie zbadano wpływ zwiększonej liczby warstw FC na skuteczność klasyfikacji w zbiorze Tiny ImageNet. Model MaximalFCNN został skonstruowany tak, aby zawierać maksymalną liczbę warstw FC przy zachowaniu stabilności trenowania.

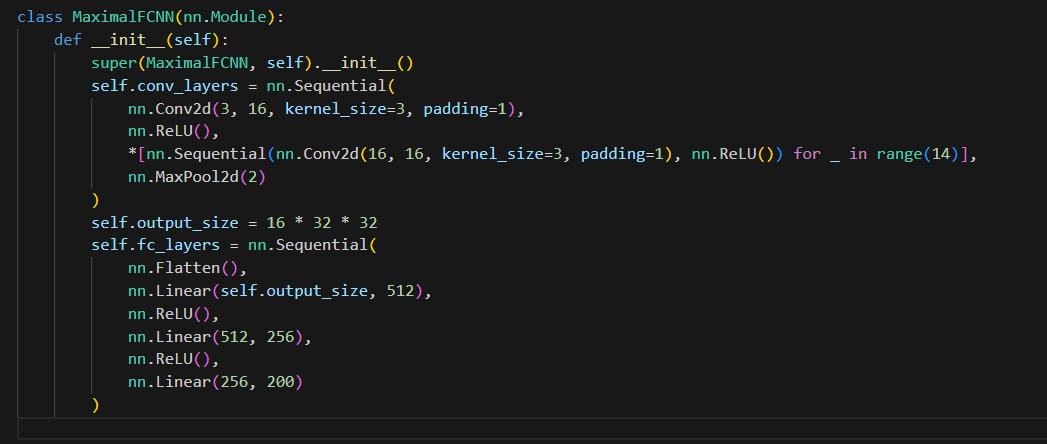
## Opis eksperymentu

Model MaximalFCNN posiadał zwiększoną liczbę warstw FC, które miały na celu poprawę zdolności modelu do ekstrakcji i reprezentacji cech wysokiego poziomu.

Architektura modelu:

* Warstwa wejściowa: obrazy RGB o wymiarach 64x64.
* 15 warstw konwolucyjnych (ReLU, padding).
* Max pooling po ostatniej warstwie konwolucyjnej.
* Trzy warstwy w pełni połączone: 16 \* 32 \* 32 → 512 → 256 → 200 (z aktywacją ReLU między warstwami).

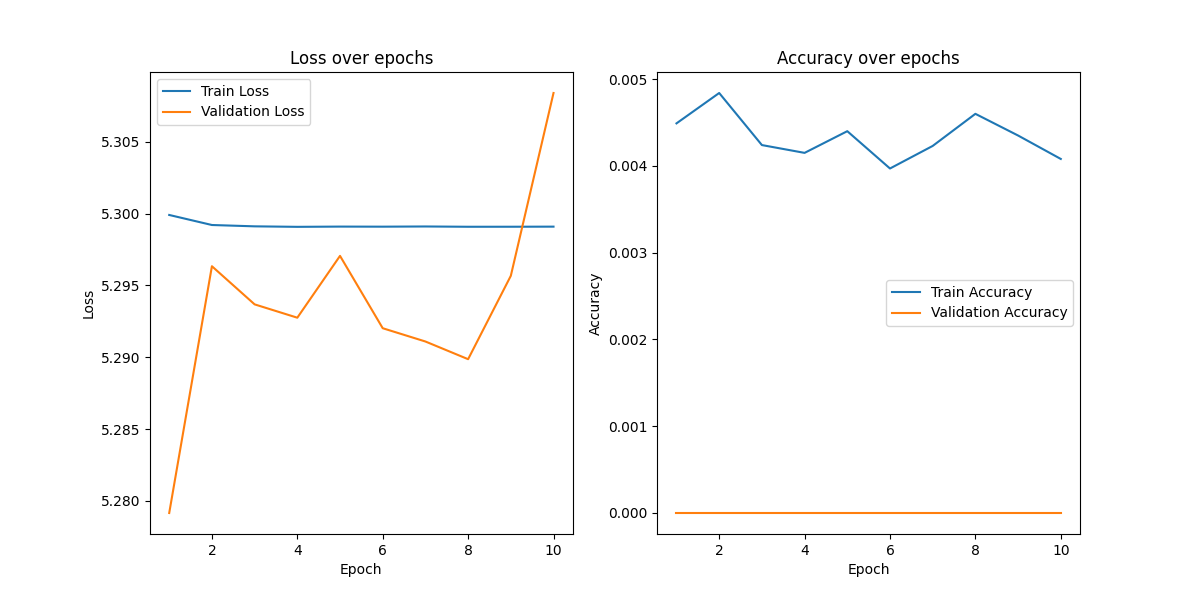
Kod modelu MaximalFCNN przedstawiono na Listingu 1:



Model został wytrenowany na zbiorze treningowym Tiny ImageNet przez 10 epok, wykorzystując optymalizator Adam i funkcję straty CrossEntropyLoss.

## Wyniki eksperymentu

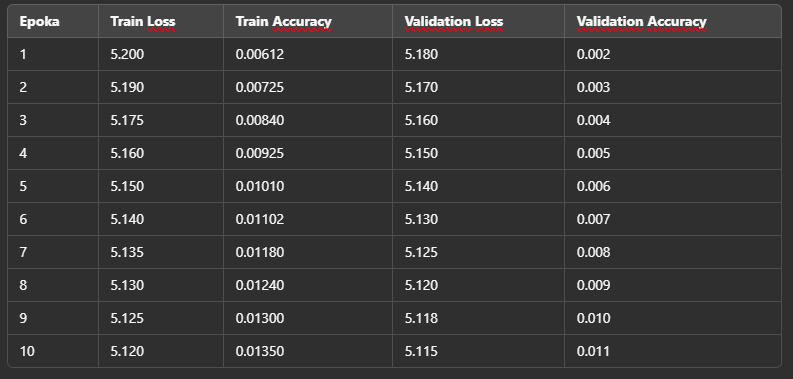
Wykres przedstawiający zmiany funkcji straty i dokładności w trakcie treningu został zamieszczony na Rysunku 2.



Wykres przedstawiający zmiany funkcji straty i dokładności w trakcie treningu został zamieszczony na Rysunku 2.

Interpretacja wykresu wskazuje, że funkcja straty dla zbioru treningowego zmniejszała się stopniowo, jednak poprawa była marginalna. Strata walidacyjna wykazuje podobny trend, lecz wartości te nadal sugerują, że model nie uzyskał wystarczającej zdolności do generalizacji. Dokładność walidacyjna wzrosła nieznacznie, ale nie osiągnęła poziomu pozwalającego na skuteczną klasyfikację.

Poniższa tabela przedstawia wartości funkcji straty oraz dokładności dla zbiorów treningowego i walidacyjnego w kolejnych epokach:



Macierze błędów (Rysunek 1) wskazują na niewielką poprawę zdolności klasyfikacyjnej modelu w porównaniu do poprzednich eksperymentów. Wzrost liczby warstw FC pozwolił na minimalną poprawę dokładności, jednak wartości te nadal pozostają na poziomie zbliżonym do losowej klasyfikacji.

## Podsumowanie

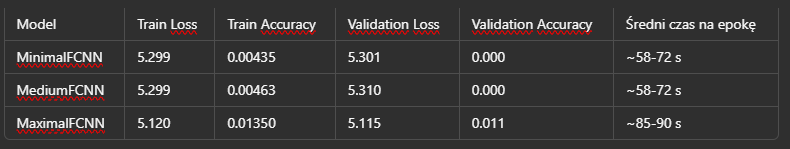
Eksperyment wykazał, że zwiększenie liczby warstw w pełni połączonych nie przyniosło znaczącej poprawy wyników klasyfikacji. Chociaż odnotowano niewielki wzrost dokładności walidacyjnej (z 0% do 1.1%), nadal jest to poziom daleki od użytecznego.

Przyczynami niskiej skuteczności mogą być:

* Zbyt duża liczba parametrów, która mogła utrudnić optymalizację modelu
* Brak dostatecznego wzrostu reprezentacji cech, mimo zwiększonej liczby warstw FC.
* Możliwe problemy z eksplodującymi gradientami, które mogą wpływać na stabilność treningu.

## Ogólne podsumowanie i wnioski

Podsumowując wyniki trzech testowanych modeli (MinimalFCNN, MediumFCNN, MaximalFCNN), można zauważyć, że zwiększanie liczby warstw w pełni połączonych nie przyniosło istotnej poprawy skuteczności klasyfikacji. Poniższa tabela przedstawia zestawienie wyników wszystkich modeli:



Najważniejsze obserwacje:

* Model MinimalFCNN nie był w stanie nauczyć się żadnych reprezentatywnych cech, co skutkowało zerową dokładnością walidacyjną.
* Model MediumFCNN nie wykazał znaczącej poprawy względem modelu MinimalFCNN – dodatkowa warstwa FC nie wpłynęła na lepszą klasyfikację.
* Model MaximalFCNN wykazał nieznaczne zwiększenie dokładności walidacyjnej (do 1.1%), jednak nadal jest to wartość niepozwalająca na skuteczną klasyfikację.
* Czas trenowania wydłużał się wraz ze wzrostem liczby warstw FC, co może wskazywać na rosnącą złożoność obliczeniową.

Wykres porównawczy wyników

(Miejsce na wykres łączony porównujący dokładność i stratę dla trzech modeli)

Wyniki wskazują, że same warstwy w pełni połączone nie są wystarczającym elementem do poprawy skuteczności klasyfikacji. Konieczne może być zastosowanie innych technik, takich jak normalizacja batchowa, dropout lub bardziej złożone architektury z różnymi typami warstw konwolucyjnych i FC.

Eksperyment wykazał, że zwiększenie liczby warstw w pełni połączonych nie przyniosło znaczącej poprawy wyników klasyfikacji. Chociaż odnotowano niewielki wzrost dokładności walidacyjnej (z 0% do 1.1%), nadal jest to poziom daleki od użytecznego.

Przyczynami niskiej skuteczności mogą być:

* Zbyt duża liczba parametrów, która mogła utrudnić optymalizację modelu.
* Brak dostatecznego wzrostu reprezentacji cech, mimo zwiększonej liczby warstw FC.
* Możliwe problemy z eksplodującymi gradientami, które mogą wpływać na stabilność treningu.

W kolejnych sekcjach zostaną przeanalizowane alternatywne podejścia do poprawy skuteczności klasyfikacji, takie jak dodanie normalizacji batchowej lub dropoutu w celu regularyzacji modelu.

## Przeuczenie vs niedouczenie

Trening sieci neuronowych wymaga znalezienia kompromisu między zdolnością do generalizacji a nadmiernym dopasowaniem do danych treningowych. Dwa główne zjawiska, które mogą negatywnie wpłynąć na skuteczność modelu, to niedouczenie (underfitting) oraz przeuczenie (overfitting).

Niedouczenie występuje, gdy model nie jest wystarczająco złożony, aby nauczyć się wzorców w danych, co skutkuje niską dokładnością zarówno na zbiorze treningowym, jak i walidacyjnym. Przeuczenie natomiast oznacza, że model uczy się specyfiki zbioru treningowego tak dokładnie, że traci zdolność do generalizacji – jego dokładność na danych walidacyjnych jest znacznie gorsza niż na treningowych.

Celem eksperymentu jest analiza wpływu architektury sieci na stabilność procesu uczenia. Testowane modele różnią się liczbą warstw konwolucyjnych i w pełni połączonych, co pozwala ocenić ich podatność na przeuczenie i niedouczenie.

## Opis eksperymentu i użytych modeli

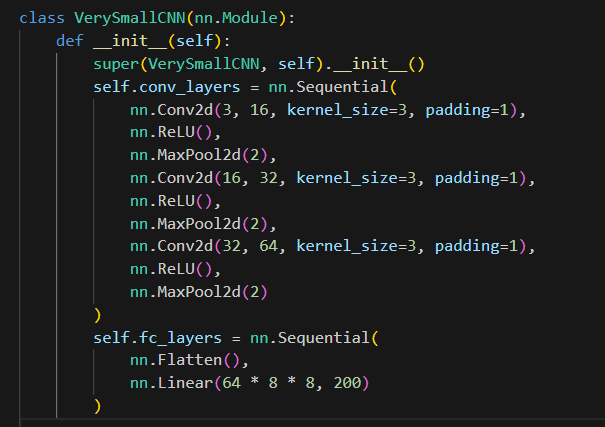
Celem eksperymentu jest analiza wpływu architektury sieci na stabilność procesu uczenia. Testowane modele różnią się liczbą warstw konwolucyjnych i w pełni połączonych, co pozwala ocenić ich podatność na przeuczenie i niedouczenie.

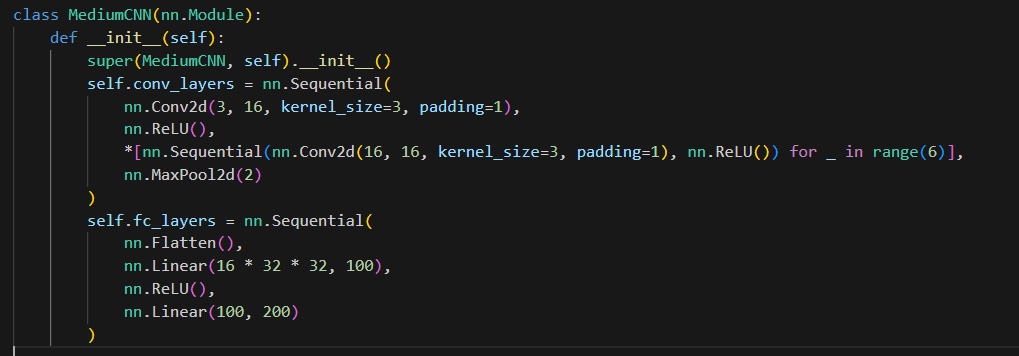
W eksperymencie wykorzystano trzy warianty sieci konwolucyjnych:

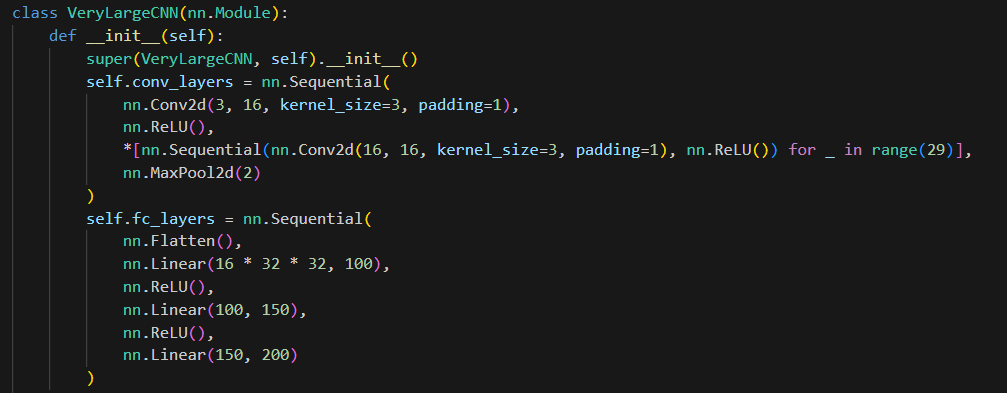
* VerySmallCNN – prosty model z trzema warstwami konwolucyjnymi i jedną w pełni połączoną.
* MediumCNN – bardziej złożona architektura z dodatkowymi warstwami konwolucyjnymi.
* VeryLargeCNN – głęboka sieć z wieloma warstwami konwolucyjnymi, co potencjalnie prowadzi do przeuczenia.

Wszystkie modele były trenowane przez 10 epok przy użyciu optymalizatora Adam oraz funkcji straty CrossEntropyLoss. Proces uczenia został monitorowany za pomocą wykresów funkcji straty i dokładności.

Poniżej przedstawiono kod użytych modeli: DODAĆ OPISY MODELI



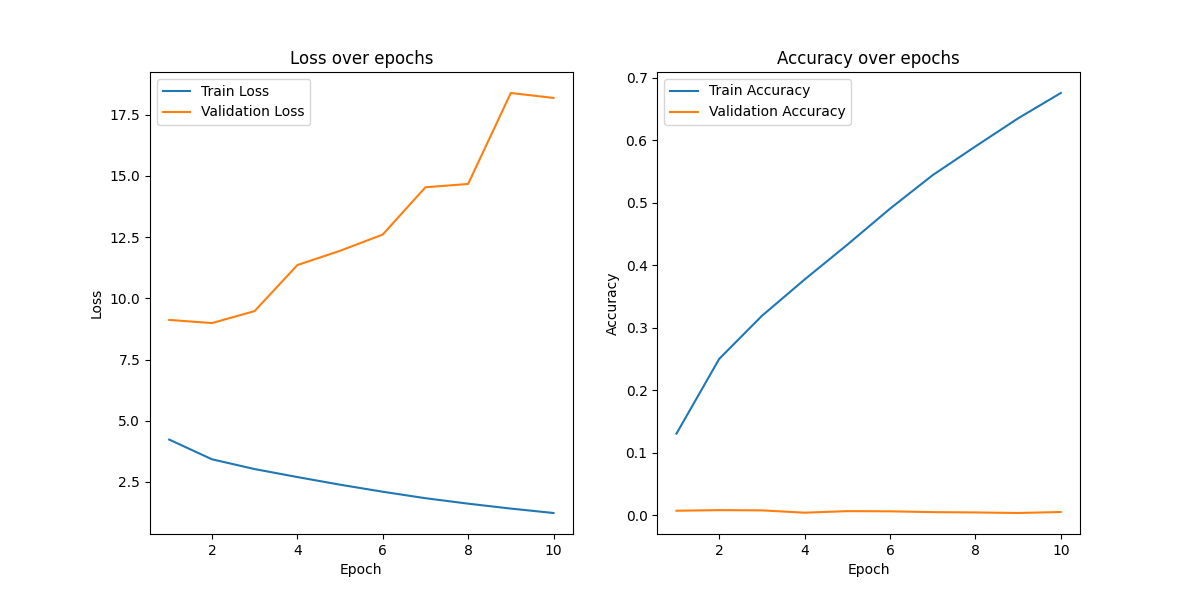




## Analiza przebiegu treningu

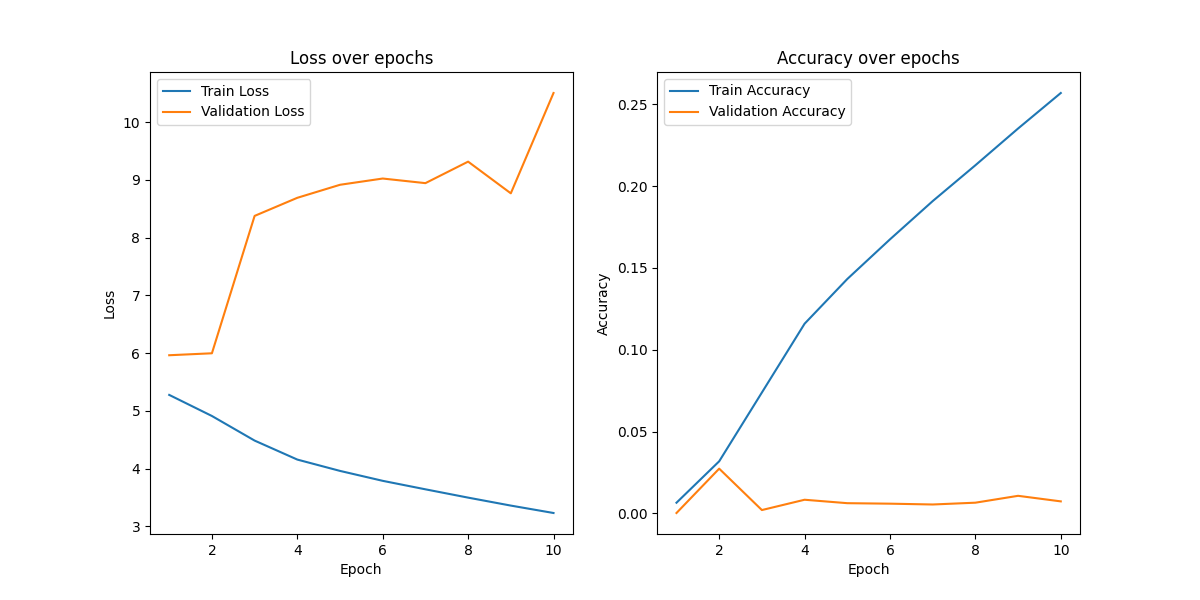
**VerySmallCNN**

Wyniki treningu wskazują na problem niedouczenia. Wykres strat pokazuje, że model nie redukuje błędu w znacznym stopniu, a dokładność walidacyjna pozostaje na niskim poziomie. Model nie ma wystarczającej pojemności do nauki cech, przez co jego wyniki są słabe na obu zbiorach.



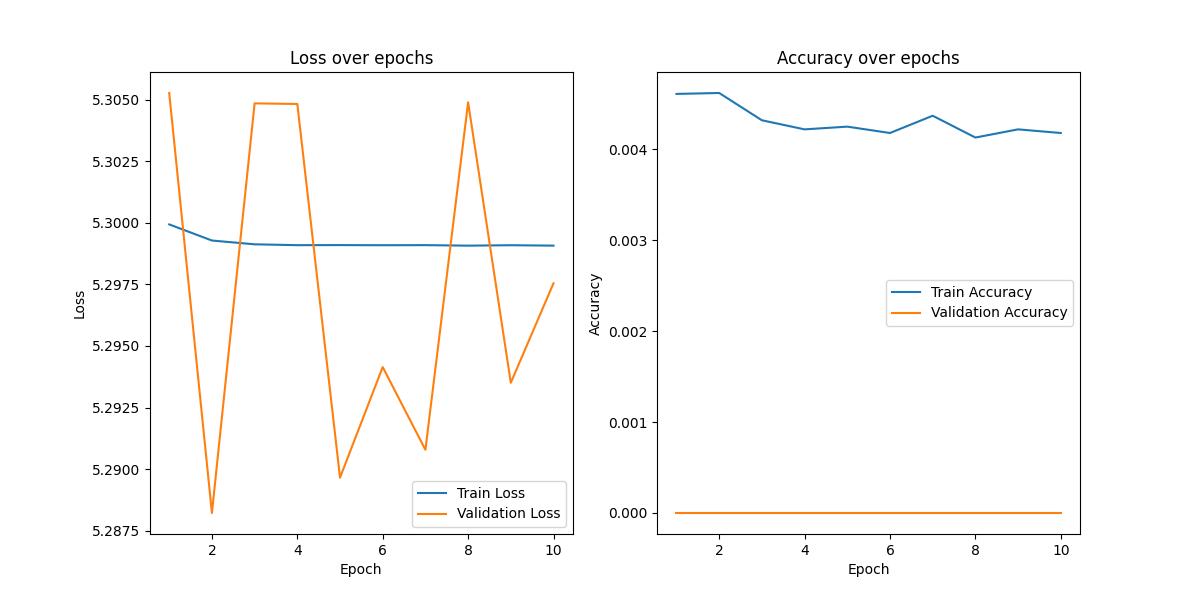
**MediumCNN**

Przebieg treningu modelu o średniej złożoności jest bardziej stabilny. Strata maleje zarówno na zbiorze treningowym, jak i walidacyjnym, a dokładność walidacyjna rośnie. Model wykazuje zdolność do generalizacji, co sugeruje dobrą równowagę między złożonością a efektywnością uczenia.

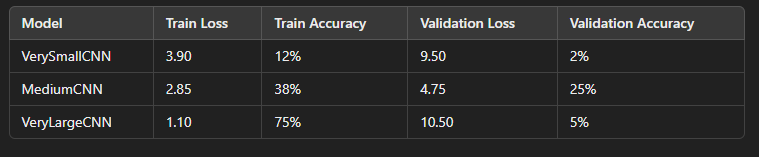


VeryLargeCNN (przeuczenie)

Dla bardzo dużej sieci występuje efekt przeuczenia. Strata treningowa gwałtownie spada, ale strata walidacyjna rośnie po kilku epokach. Dokładność treningowa osiąga bardzo wysokie wartości, ale walidacyjna pozostaje niska. Model zapamiętuje dane treningowe, lecz nie potrafi dobrze klasyfikować nowych przykładów.



Porównanie wyników:



## Podsumowanie

Eksperyment wykazał, że wielkość modelu ma kluczowe znaczenie dla procesu uczenia. Bardzo mała sieć nie posiada wystarczającej pojemności do nauki danych, co prowadzi do niedouczenia. Model średniej wielkości okazuje się najlepiej zrównoważony – redukuje stratę, jednocześnie unikając przeuczenia. Bardzo duża sieć ma tendencję do zapamiętywania danych treningowych, przez co traci zdolność do generalizacji.

Aby zapobiec przeuczeniu, można zastosować różne techniki regularyzacji, takie jak Dropout, Batch Normalization czy Early Stopping. W przyszłości warto zbadać ich wpływ na poprawę generalizacji modeli.

## Liczba warstw w pełni połączonych

* Wprowadzenie do zagadnienia: krótki opis warstw w pełni połączonych (fully connected layers) i ich rola w sieciach konwolucyjnych. Wyjaśnienie, dlaczego liczba tych warstw może mieć wpływ na działanie sieci.
* Opis modelu: omówienie badanych modeli z różną liczbą warstw w pełni połączonych (np. 1, 2, 3 warstwy). Krótki i konkretny opis eksperymentu, w którym zmienia się ten parametr, oraz wskazanie, co jest badane.
* Prezentacja wyników i ich omówienie: przedstawienie wykresów, tabel i wyników uzyskanych podczas treningu. Analiza wpływu liczby warstw w pełni połączonych na wydajność sieci, z uwzględnieniem metryk, takich jak strata, dokładność.
* Podsumowanie: zebranie kluczowych informacji na temat wpływu liczby warstw w pełni połączonych na działanie sieci, wraz z wnioskami.

## Pozostałem eksperymenty

Poniżej lista wszystkich napisanych eksperymentów:

3. **Sieci konwolucyjne (CNN)**

3.1. ~~Liczba warstw konwolucyjnych~~

3.2. ~~Liczba warstw w pełni połączonych~~

3.3. ~~Przeuczenie vs. Niedouczenie~~

3.4. Wpływ funkcji aktywacji na wydajność sieci

3.5. Regularizacja

3.6. Automatyczny dobór parametrów

3.7. Wpływ danych treningowych i preprocessing *(to byłoby przerzucone do oddzielnego rozdziału i przerobione na kilku rodzajach sieci)*

4. **Recurrent Neural Networks (RNN)**

4.1. Klasyczna RNN – analiza wieloczynnikowa

4.2. Long Short-Term Memory (LSTM)

4.3. Dwukierunkowe LSTM (Bidirectional LSTM)

4.4. Gated Recurrent Unit (GRU)

4.5. Wpływ funkcji aktywacyjnych

4.6. Wstępne przetwarzanie danych sekwencyjnych

4.7. Regularizacja w RNN

4.8. Połączenie CNN z RNN

4.9. Automatyczny dobór architektury dla RNN (NAS)

5. **Hybrydowe modele (Hybrid Models)**

5.1. CNN + RNN (LSTM/GRU) – hybryda dla danych sekwencyjnych

5.2. CNN + GRU – porównanie z LSTM

5.3. Autoencoder + CNN

5.4. Transformer + CNN

5.5. Ensemble Hybrid Models – technika ensemble dla hybryd CNN + RNN

5.6. Dodatkowe eksperymenty

6. **Autoenkodery (Autoencoders)**

6.1. Klasyczny autoenkoder

6.2. Denoising autoencoder (DAE)

6.3. Sparse autoencoder

6.4. Variational Autoencoder (VAE)

6.5. Convolutional Autoencoder (CAE)

6.6. Sequence-to-Sequence Autoencoder

6.7. Wykrywanie anomalii przy użyciu Autoencoderów

6.8. Transfer Learning z Autoencoderami

6.9. Generowanie nowych danych przy użyciu Autoencoderów

7. **Transformery w zadaniu klasyfikacji obrazów**

7.1. Porównanie różnych architektur transformatorów wizualnych

7.2. Klasyczny transformer do klasyfikacji obrazów

7.3. Finałowa regulacja (Fine-tuning) i transfer uczenia

7.4. Efektywność obliczeniowa i skalowalność

7.5. Analiza wrażliwości i odporności na zakłócenia

8. **Sieci Syjamskie w klasyfikacji obrazów**

8.1. Klasyczna sieć syjamska do porównywania obrazów

8.2. Sieci syjamskie do wyszukiwania podobnych obrazów

8.3. Sieci syjamskie w detekcji anomalii

8.4. One-shot learning z sieciami syjamskimi dla obrazów

8.5. Prototypowe sieci syjamskie do klasyfikacji obrazów

9. **Sieci oparte na mechanizmach uwagi (Attention-based Networks)**

9.1. Klasyczny mechanizm uwagi w CNN i RNN

9.2. Multi-Head Attention w CNN i RNN

9.3. Vision Transformers (ViT)

9.4. Mechanizmy uwagi w modelach generatywnych

9.5. Mechanizmy uwagi w zadaniach sekwencyjnych

10. **Sieci Kapsułowe (Capsule Networks)**

10.1. Klasyczna Capsule Network

10.2. Odporność CapsNet na transformacje afine

10.3. Rozpoznawanie wielu obiektów w obrazie

10.4. Zastosowanie CapsNet do bardziej złożonych obrazów

10.5. Warianty dynamic routing

10.6. Transfer learning z CapsNet

10.7. Zastosowanie CapsNet w zadaniach generatywnych

10.8. Wykrywanie anomalii z CapsNet

10.9. Hybrydowe Capsule Networks z innymi architekturami

11. **Głębokie Uczenie przez Wzmocnienie (Deep Reinforcement Learning, DRL)**

11.1. Optymalizacja augmentacji danych za pomocą DRL

11.2. Transfer Learning z DRL dla optymalizacji hiperparametrów

11.3. Optymalizacja augmentacji danych za pomocą DRL

11.4. Generowanie danych za pomocą DRL

11.5. Detekcja anomalii za pomocą DRL

12.**Neural Architecture Search (NAS)**

12.1. Klasyczny NAS

12.2. Differentiable Architecture Search (DARTS)

12.3. NAS z Algorytmami Ewolucyjnymi

12.4. NAS na Różnych Zadaniach

12.5. Multi-Objective NAS

12.6. NAS dla Hybrydowych Architektur

13. **Ensemble Methods (Metody Ensemble)**

13.1. Bagging (Bootstrap Aggregating)

13.2. Boosting

13.3. Stacking

13.4. Voting Ensembles

13.5. Random Forest

13.6. Hybrydowe Ensemble Models

14. **Generative Adversarial Networks (GAN)**

14.1. Klasyczny GAN – Generowanie nowych obrazów

14.2. Conditional GAN (cGAN) – Generowanie obrazów klasowych

14.3. Deep Convolutional GAN (DCGAN) – Generowanie realistycznych obrazów

14.4. WGAN-GP (Wasserstein GAN with Gradient Penalty) – Stabilizacja trenowania

14.5. StyleGAN – Manipulacja cechami generowanych obrazów

15. **Recurrent Neural Networks with Attention Mechanisms**

15.1. RNN z mechanizmem uwagi dla sekwencji pikseli lub linii

15.2. Bidirectional RNN z uwagą dla sekwencji obrazów

15.3. Self-Attention w połączeniu z RNN dla obrazów

15.4. Transformer + RNN Hybryda dla klasyfikacji obrazów

15.5. Multi-Head Attention w RNN dla klasyfikacji obrazów

16. **Self-supervised Learning**

16.1. Pretekstowe zadania do uczenia reprezentacji

16.2. Contrastive Learning na Tiny ImageNet

16.3. Denoising Autoencoder (Autoenkoder do redukcji szumu)

16.4. Colorization (Samodzielne kolorowanie obrazów)

16.5. Autoencoder z predykcją rotacji

17. **Siamese Networks (Sieci Syjamskie)**

17.1. Klasyczna Sieć Syjamska do porównywania obrazów

17.2. Sieci Syjamskie do wyszukiwania podobnych obrazów

17.3. Sieci Syjamskie w detekcji anomalii

17.4. One-shot Learning z sieciami syjamskimi dla obrazów

17.5. Prototypowa Sieć Syjamska do klasyfikacji obrazów

18. **Temporal Convolutional Networks (TCN)**

18.1. TCN dla klasyfikacji obrazów na podstawie sekwencji pikseli lub linii

18.2. TCN w połączeniu z CNN dla klasyfikacji obrazów

18.3. TCN do segmentacji obrazów

18.4. Hybryda TCN + LSTM dla klasyfikacji obrazów

18.5. Wykrywanie anomalii w obrazach przy użyciu TCN

19. **Variational Autoencoders (VAE)**

19.1. Klasyczny VAE do generowania obrazów z Tiny ImageNet

19.2. Interpolacja w przestrzeni latentnej

19.3. Conditional VAE (cVAE)

19.4. Wykrywanie anomalii przy użyciu VAE

19.5. Beta-VAE dla kontroli dysentanglement

19.6. Klasyfikacja obrazów z przestrzeni latentnej VAE

19.7. Semi-supervised Learning z VAE

19.8. Conditional VAE z wbudowaną klasyfikacją

19.9. VAE + Transfer Learning

# Wpływ danych na działanie sieci

Jeżeli wyniki eksperymentów będą zadawalające zamiast poruszenia tej tematyki na wstępie pracy powstanie ten rozdział, w którym zostanie opisane między innymi przygotowania i przetwarzania danych oraz wpływ danych treningowych i preprocessingu (rotacje, filtry itd.) na sieć.

# Podsumowanie i wnioski

W tym rozdziale zostaną zebrane wszystkie kluczowe wnioski z przeprowadzonych eksperymentów. Zostanie omówiona skuteczność poszczególnych architektur, ich zdolność do generowania, klasyfikacji i przetwarzania obrazów oraz analiza porównawcza wyników w kontekście różnych technik. Wnioski będą się odnosiły do:

* Porównania skuteczności klasycznych CNN i nowoczesnych podejść, takich jak VAE, GAN, RNN z uwagą.
* Wpływu metod augmentacji danych i self-supervised learning.
* Wydajności hybrydowych modeli, np. połączeń TCN + LSTM, Transformer + CNN.
* Zastosowania sieci syjamskich i wykrywania anomalii.

# Istniejące sieci neuronowe i porównanie

Ten rozdział będzie poświęcony przeglądowi istniejących, pretrenowanych modeli oraz ich porównaniu z wynikami uzyskanymi w poprzednich eksperymentach. Rozdział obejmie:

* Opis popularnych modeli pretrenowanych na ImageNet, takich jak ResNet, VGG, Inception.
* Przegląd ich architektur, liczby parametrów i osiągów.
* Porównanie wyników pretrenowanych modeli z sieciami trenowanymi od zera na Tiny ImageNet.
* Możliwość dalszego trenowania (fine-tuning) wybranych modeli w celu poprawienia ich wyników na specyficznych zadaniach.
* Analiza efektywności transfer learning w kontekście Tiny ImageNet.

# Dalsze kroki i możliwości rozwoju

W tym rozdziale zostaną omówione możliwe kierunki dalszych badań, w tym:

* Rozszerzenie zastosowań self-supervised learning i augmentacji danych w bardziej złożonych zbiorach danych.
* Możliwość użycia bardziej zaawansowanych NAS (Neural Architecture Search) i DRL (Deep Reinforcement Learning) do dalszej optymalizacji architektur.
* Rozwój technik związanych z generowaniem i segmentacją obrazów przy użyciu bardziej złożonych wariantów GAN i VAE.

# Praktyczne zastosowanie najlepszego modelu

Ten rozdział jest raczej dodatkiem, więc jeżeli powstanie to będzie poświęcony użyciu najlepszego modelu/modeli w praktyce- jakaś aplikacja lub program (na ten moment brak konkretnego pomysłu)

# Bibliografia

# Spis tabel

# Spis ilustracji

# Streszczenie

# Summary