# Architektura i trenowanie własnego modelu

Projektowany model to klasyczna sieć konwolucyjna (CNN) przeznaczona do rozpoznawania elementów na obrazach, składająca się z dwóch warstw konwolucyjnych, funkcji aktywacji ReLU, maxpoolingu oraz w pełni połączonych warstw.

1. **Warstwy Konwolucyjne – Convolution Layers:**
   * **Warstwa 1:** Conv2d(3, 32, kernel\_size=3, stride=1, padding=1) – Przyjmuje obrazy RGB o wymiarach 128x128 i tworzy 32 mapy cech (128x128). Kernel 3x3 służy do konwolucji.
   * **Aktywacja:** ReLU – Wprowadza nieliniowość, wspomagając model w uczeniu bardziej złożonych zależności.
   * **Pooling:** MaxPool2d(kernel\_size=2, stride=2) – Zmniejsza wymiary przestrzenne mapy cech do 64x64, co zmniejsza liczbę parametrów i zapobiega nadmiernemu dopasowaniu.
   * **Warstwa 2:** Conv2d(32, 64, kernel\_size=3, stride=1, padding=1) – Tworzy 64 mapy cech (64x64).
   * **Aktywacja:** ReLU
   * **Pooling:** MaxPool2d(kernel\_size=2, stride=2) – Zmniejsza wymiary do 32x32.
2. **Warstwy w pełni połączone – Fully Connected Layers:**
   * **Warstwa 1:** Linear(64 \* 32 \* 32, 128) – Spłaszcza dane wejściowe do wektora 128-wymiarowego.
   * **Aktywacja:** ReLU
   * **Warstwa 2:** Linear(128, 1) – Ostateczna warstwa klasyfikująca (klasyfikacja binarna).

Wybór tej architektury oparty jest na efektywności sieci konwolucyjnych w zadaniach rozpoznawania elementów w obrazach. Dwie warstwy konwolucyjne umożliwiają stopniową redukcję wymiarów obrazu oraz uchwycenie różnych poziomów cech.

Model został zaimplementowany w klasie **FaceIDModel**, dziedziczącej po PyTorch Lightning, co upraszcza proces trenowania i zarządzania cyklem życia modelu (np. wprowadzenie early stopping w celu zapobiegania przeuczeniu). Trenowanie odbywa się za pomocą optymalizatora **Adam**, wydajnego w przypadku dużych zbiorów danych, takich jak CelebA, oraz stosującego adaptacyjną strategię uczenia.

Funkcją celu jest **Binary Cross Entropy Loss**, użyta w wersji **nn.BCEWithLogitsLoss** dla lepszej efektywności obliczeniowej. Zastosowano także tensor **pos\_weight**, który pomaga radzić sobie z niezbalansowanym zbiorem danych.

Aby poprawić generalizację modelu, zastosowano augmentację danych w postaci **szumu Gaussa**, **zmiany koloru** oraz **rotacji zdjęć**, co pozwala na lepsze dostosowanie modelu do nieznanych danych, np. obrazów z kamery.

A graph with blue lines and red lines

Description automatically generatedA graph with a line graph

Description automatically generated

# Architektura i trenowanie wykorzystanego modelu z torchvision

A screenshot of a computer program

Description automatically generatedDo zadania klasyfikacji obrazu, polegającego na rozpoznawaniu atrybutu „Smiling”, wykorzystano model **ResNet-18** z uprzednio wytrenowanymi wagami na zbiorze **ImageNet1K\_V1**. Jest to sieć o 18 warstwach, która wykorzystuje **residual connections**, umożliwiające radzenie sobie z problemem zanikającego gradientu w głębokich sieciach.

**Struktura Modelu ResNet-18**

Model składa się z 4 bloków konwolucyjnych:

* **Blok 1:** 64 filtry
* **Blok 2:** 128 filtrów
* **Blok 3:** 256 filtrów
* **Blok 4:** 512 filtrów

Każdy blok odpowiada za ekstrakcję cech z obrazu, a **połączenia resztkowe** pomagają w efektywnym przetwarzaniu i nauce reprezentacji.

**Przygotowanie Danych**

Obrazy wejściowe musiały zostać przeskalowane do rozdzielczości **224x224 px**, zgodnie z wymaganiami modelu, który został wytrenowany na zbiorze **ImageNet**. Dodatkowo, obrazy normalizowane są według wartości średnich i odchyleń standardowych dla kanałów RGB:

* **Średnie:** (0.485, 0.456, 0.406)
* **Odchylenie standardowe:** (0.229, 0.224, 0.225)

**Warstwy W Pełni Połączone**

Do modelu ResNet-18 dodano dwie warstwy w pełni połączone:

* **Warstwa 1:** Linear(model.fc.in\_features, 128) – Spłaszcza dane do wektora o rozmiarze 128.
* **Warstwa 2:** Linear(128, 1) – Końcowa warstwa klasyfikująca z jedną jednostką, ponieważ jest to klasyfikacja binarna.

**Transfer Learning i Fine-Tuning**

Zastosowano **transfer learning** oraz **fine-tuning**, adaptując model ResNet-18 do nowego zadania. Po wstępnym załadowaniu wag z ImageNet, model przeszedł proces fine-tuningu, w którym zamrożono wagi w początkowych warstwach, a następnie dostosowano wyższe warstwy do specyfiki danych wejściowych, w tym warstwy w pełni połączone.

A graph with blue and orange lines

Description automatically generated**Optymalizacja**

Do trenowania modelu użyto **Adam** jako optymalizatora, zapewniającego adaptacyjne tempo uczenia, co było kluczowe w procesie fine-tuningu. Funkcją celu była **Binary Cross Entropy Loss**, odpowiednia dla klasyfikacji binarnej, co pozwoliło na efektywne rozróżnienie pomiędzy obrazami z uśmiechem a bez niego.

**Podsumowanie**

Model **ResNet-18** z pretrenowanymi wagami z **ImageNet** został skutecznie zaadoptowany do zadania klasyfikacji obrazu „Smiling” przy pomocy **transfer learning** i **fine-tuningu**. Zastosowane techniki pozwoliły na efektywne wykorzystanie wstępnie wytrenowanych wag oraz dostosowanie modelu do specyfiki nowych danych.

# Wyniki testów obydwu modeli na danych testowych dostępnych w CelebA

* Wyniki testów obu modeli dla danych testowych dostępnych w zbiorze CelebA (podsumowanie wyników w postaci macierzy pomyłek oraz przykładów dobrze i źle sklasyfikowanych obiektów, omówienie wyników).

A blue squares with white text

Description automatically generated A blue squares with white text

Description automatically generated

Accuracy dla naszego modelu to 97.57%. Accuracy modelu to 92.86%

# Wyniki testów obydwu modeli na danych ze zbioru WIDERFACE

 (opis przygotowania zbioru testowego oraz podsumowanie wyników jak w poprzednim punkcie, omówienie wyników).

A blue squares with white text

Description automatically generatedA blue squares with white text

Description automatically generated

\Test accract dla naszego modelu to 53.8% Accuracy dla modelu to 50.64%

# Wyniki testów modeli w programie obsługującym obraz z kamery

* Wyniki testów obu model w udostępnionym programie testowym (przykładowe wyniki dla różnych warunków akwizycji, omówienie wyników).

# Opis architektury modelu do detekcji twarzy

Do zadania detekcji twarzy wybrano model **Faster R-CNN ResNet-50 FPN**, oparty na architekturze **Feature Pyramid Network (FPN)**, ze względu na jego zaawansowaną strukturę, skuteczną w zadaniach detekcji obiektów. Model ten jest szeroko wykorzystywany ze względu na swoją efektywność w generowaniu regionów zainteresowania (ROIs) oraz precyzyjnej detekcji obiektów w obrazach.

Aby wykorzystać model w zadaniu detekcji twarzy, przeprowadzono następujące czynności na zbiorze treningowym: przygotowano dane wejściowe w postaci zdjęć oraz odpowiadających im adnotacji, zawierających informacje o pozycjach twarzy w obrazach (bounding boxy). Zbiór danych pochodził z pliku tekstowego, który wymagał odpowiedniego parsowania, aby przypisać dane do właściwych obrazów. W trakcie tego procesu napotkano problem związany z nadmiernie długą ścieżką do pliku, co prowadziło do błędów podczas treningu. Po rozwiązaniu tego problemu (pominięciu wadliwych z poziomu systemowego zdjęć) każde zdjęcie zostało przeskalowane do rozdzielczości **800x800 pikseli**, co jest wymaganym rozmiarem wejściowym modelu Faster R-CNN, który był trenowany na danych o tym rozmiarze.

Dodatkowo, zastosowano **normalizację danych**, zgodną z procesem pretrenowania modelu na zbiorze **ImageNet**:

* **Średnie:** (0.485, 0.456, 0.406)
* **Odchylenie standardowe:** (0.229, 0.224, 0.225)

Aby poprawić zdolność modelu do generalizacji oraz radzenia sobie z różnorodnymi danymi, zastosowano augmentację danych, w tym:

* Obrót zdjęć, co umożliwiło modelowi rozpoznawanie twarzy niezależnie od orientacji.
* Odbicie lustrzane, uwzględniające różne kierunki patrzenia na twarz.
* Delikatne modyfikacje kolorów oraz dodanie szumu, szczególnie w przypadku obrazów z kamerki, co miało na celu zwiększenie odporności modelu na zmienne warunki oświetleniowe oraz różną jakość obrazu.

Proces trenowania odbywał się na zbiorze danych **WIDER FACE**, który zawiera obrazy twarzy w różnych warunkach (np. zmienne oświetlenie, różne kąty). Podczas treningu zastosowano optymalizator **SGD z momentum**, a funkcję celu stanowiła suma strat związanych z klasyfikacją i regresją. Aby zapobiec przeuczeniu, wprowadzono mechanizm **early stopping** na podstawie monitorowania strat walidacyjnych oraz zastosowano harmonogram zmiany współczynnika uczenia (**ReduceLROnPlateau**).

Wybór danych walidacyjnych i treningowych polegał na losowym podziale zbioru danych na zestawy treningowe i walidacyjne w proporcji 80/20.

Całkowity czas treningu wyniósł 13201 sekund.

# Wyniki działania modelu detekcji twarzy

* Wyniki działania modelu detekcji twarzy w udostępnionym programie testowym (przykładowe wyniki umożliwiające ocenę działania detektora dla różnych warunków akwizycji i porównanie go z detektorem kaskadowym, omówienie wyników).