Politechnika Wrocławska

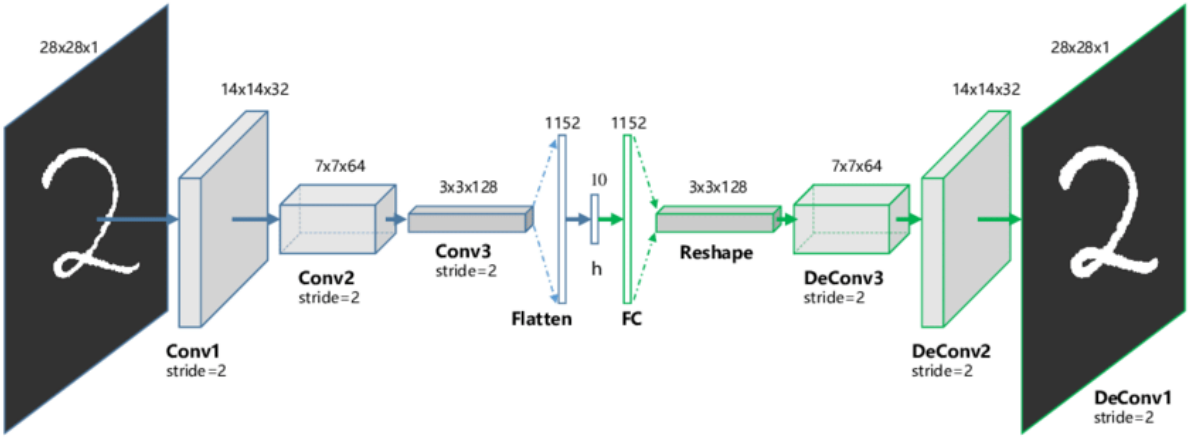
Dokument końcowy
Projektowanie i zastosowanie sieci neuronowych

Spis treści

Wojciech Papis, Piotr Malanowski, Kacper Mirecki
Nr albumu: 264211, 264183, 264256

1 Opis zrealizowanego projektu

Celem projektu było zasotoswanie sieci neuronowej do przetwarzania obrazów techniką super-resolution, która zwiększa ich rozdzielczość i poprawia jakość. Opracowaliśmy konwolucyjny autoenkoder (Convolutional Autoencoder (CAE)), który przekształca obrazy o niskiej rozdzielczości na obraz o wysokiej rozdzielczości.
Model został zaprojektowany z myślą o datasecie Unsplash, który uprzednio podzieliliśmy na odpowiednie zestawy danych – treningowe i walidacyjne. Autoenkoder, który został użyty do tego projektu ma następującą strukturę:



Rysunek 1: Struktura autoenkodera

- Wejście autoenkodera:
  1. Na wejściu przyjmujemy obrazy RGB o rozmiarze 800x1200 pikseli
- Kodowanie:
  1. Dwie warstwy konwolucyjne z 64 filtrami o rozmiarze 3x3.
  2. Warstwa MaxPooling2D, która zmniejsza rozmiar danych zachowując przy tym wypełnienie (używamy jej, aby zmniejszyć złożoność obliczeniową i kontrolować nadmierne dopasowanie (overfitting)).
  3. Warstwa Dropout z 30% prawdopodobieństwem zerowania neuronów, aby zapobiec przeuczeniu.
  4. Kolejne dwie warstwy konwolucyjne z 128 filtrami również o rozmiarze 3x3 i funkcją aktywacji ReLU.
  5. Kolejna warstwa MaxPooling2D, która zmniejsza rozmiar danych.
- Środkowa warstwa kodera:
  1. Warstwa konwolucyjna z 256 filtrami, która tworzy skompresowaną reprezentację obrazu.
- Dekodowanie:
  1. Warstwa UpSampling2D, która zwiększa rozmiar danych.
  2. Dwie warstwy konwolucyjne z 128 filtrami.
  3. Dodanie warstwy 5 i 10 (obie konwolucyjne z taką samą ilościa filtrów) w celu połączenia informacji z różnych etapów kodera i dekodera (skip connections).

4. Kolejna warstwa UpSampling2D zwiększająca rozmiar danych.
5. Dwie warstwy konwolucyjne z 64 filtrami.
6. Dodanie warstw 14 i 2, również obie konwolucyjne.
7. Ostatnia warstwa konwolucyjna – wyjściowa z 3 filtrami (RGB) i funkcją aktywacji ReLU generująca oczekiwany obraz.

2 Lista wykonawców

1. Wojciech Papis – 264211
2. Piotr Malanowski – 264183
3. Kacper Mirecki – 264256

3 Sposób realizacji projektu

Realizacja projektu przebiegała w następujących krokach:

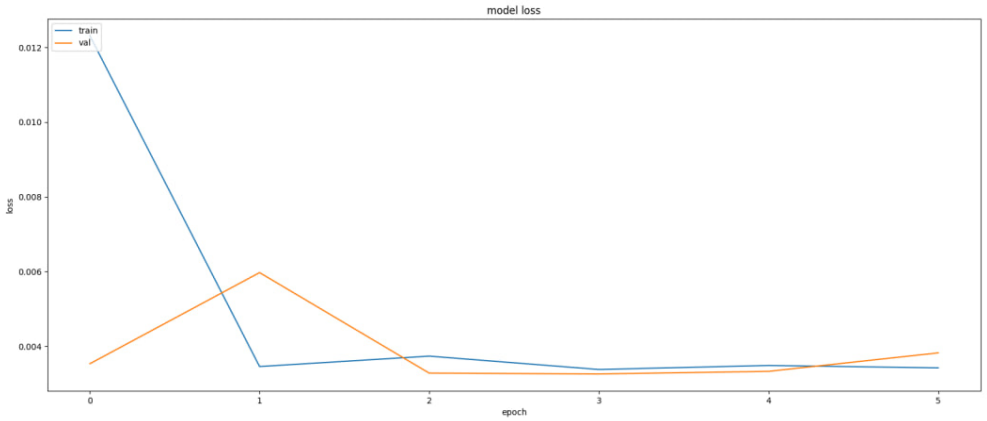
1. Przygotowanie zbioru danych pobranego z kaggle i podzielenie go na odpowiednie podzbiory treningowe i walidacyjne w stosunku 85%/15%.
2. Zaprojektowanie i implementacja przy użyciu TensorFlow i Keras odpowiedniego modelu dopasowanego do naszego tematu, zdecydowaliśmy się na konwolucyjny autoenkoder (CAE).
3. Trening modelu i dopasowanie odpowiednich parametrów treningowych takich jak liczba epok, czy też batch size.
4. Analiza i porównanie wyników z poprzednimi.
5. Dalsza optymalizacja i dostrajanie parametrów np:
    - Zmniejszanie i zwiększanie rozmiarów filtrów w warstwach konwolucyjnych,
    - Dodanie/usuwanie warstwy Dropout w celu osiągnięcia lepszej wydajności, i zmniejszenia przeuczeniu,
    - Dodanie/usuwanie regularyzacji L1 w celu zmniejszenia złożoności i nadmiernemu dopasowaniu modelu.

4 Opis wykorzystanych technologii

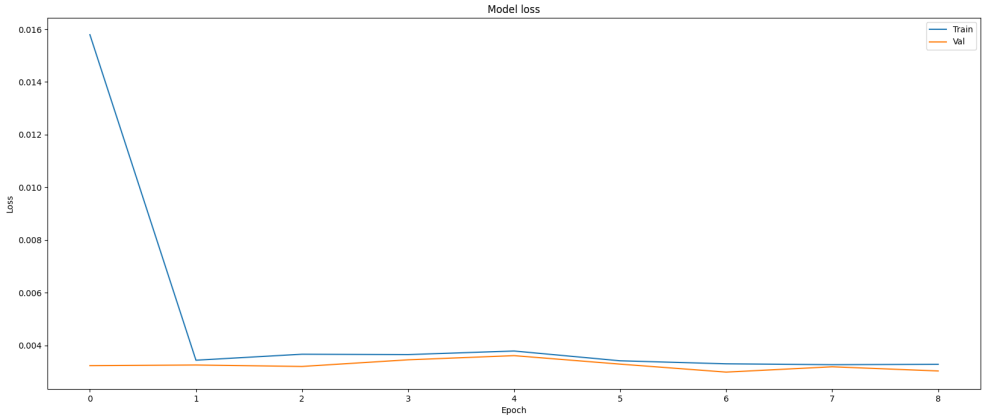
- TensorFlow / Keras – główny framework dzięki, któremu zbudowaliśmy i trenowaliśmy model sieci neuronowych,
- pandas – biblioteka, która pomogła nam w przygotowaniu zestawu danych do trenowania sieci i podzieleniu jej na właściwe zbiory do trenowania i walidacji,
- matplotlib – dzięki tej bibliotece wyświetlamy stosowne obrazy przedstawiajace róznice między oryginalnymi obrazami, a zrekonstruowanymi, i także wyświetlamy wykres pokazujący postępy w uczeniu się sieci,
- numpy – biblioteka została wykorzystana do obróbki i przygotowania danych.

5 Wyniki eksperymentów

Poniżej przedstawiamy wyniki eksperymentów i testów przeprowadzonych w ramach projektu:



Rysunek 2: Wykres strat treningowych i walidacyjnych w trakcie treningu modelu przed optymalizacją parametrów



Rysunek 3: Wykres strat treningowych i walidacyjnych w trakcie treningu modelu po optymalizacji parametrów

```
Epoch 8/8
533/533 [==============================] - ETA: 0s - loss: 0.0034 - accuracy: 0.8928
Epoch 8: val_loss did not improve from 0.00303
533/533 [==============================] - 3437s 6s/step - loss: 0.0034 - accuracy: 0.8928 - val_loss: 0.0032 - val_accuracy: 0.8158 - lr: 0.0010
```

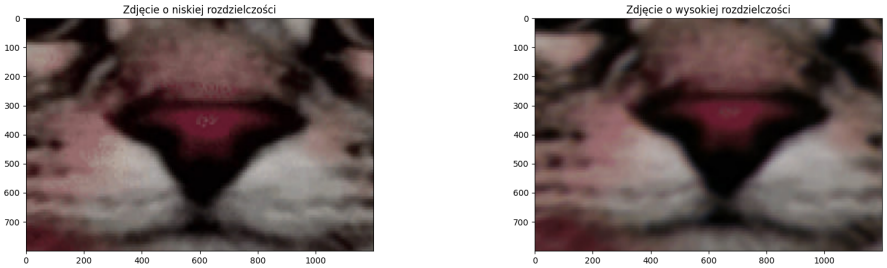
Rysunek 4: Przykładowe wyniki po wytrenowaniu modelu bez zoptymalizowanych parametrów

```
Epoch 9/9
533/533 [==============================] - ETA: 0s - loss: 0.0033 - accuracy: 0.9197
Epoch 9: val_loss did not improve from 0.00299
533/533 [==============================] - 3147s 6s/step - loss: 0.0033 - accuracy: 0.9197 - val_loss: 0.0030 - val_accuracy: 0.9120 - lr: 2.0000e-04
```

Rysunek 5: Wyniki po wytrenowaniu modelu z zoptymalizowanymi parametrami

- Użycie ostatecznej wersji wytrenowanego modelu

Rysunek 6: Trzy przykłady obrazów o niskiej rozdzielczości oraz odpowiadający mu oryginalny obraz o wysokiej rozdzielczości i obraz wygenerowany przez model.



Rysunek 7: Inny przykład porównania obrazów niskiej i przewidywanej wysokiej rozdzielczości (wygenerowanym przez model) tym razem z powiększeniem na szczegóły, gdzie bardzo łatwo zauważyć różnice.

6 Wnioski

Najważniejsze wnioski z projektu to:

- Opracowany model CAE osiągnął dokładność na poziomie 0.92 i niską wartość funkcji strat na zbiorze walidacyjnym.
- Po początkowych iteracjach wartość strat rekonstrukcji się stabilizuje.
- Dodanie warstwy Dropout, pomogło zapobiec przetrenowaniu modelu.
- Dodana regularyzacja L1 do warstw nie pomogła w osiągnięciu lepszej wydajności modelu, więc została usunięta podczas testów modelu.
- Optymalizacja parametrów treningowych i architektury modelu była kluczowa dla uzyskania jak najlepszych wyników.
- Użycie bardziej rozbudowanego zestawu danych zawierającego wiekszą liczbę obrazów oraz zdjęcia w pełnej rozdzielczości Full HD mogło by zwiększyć dokładność modelu.
- Zamiast sztucznego zaszumienia, model mógłby mieć lepsze zastosowanie w praktyce, gdyby wykorzystać rzeczywiste zaszumione obrazy zamiast sztucznej pikselizacji obrazu.
- Alternatywnym podejściem do tematu zamiast autoenkodera, może być generatywna sieć przeciwstawna (GAN).

7 Możliwość zastosowania

Opracowana sieć w ramach naszego projektu może znaleźć następujące zastosowania:

- Wyostrzenie i ogólna poprawa jakości zdjęć w fotografii,
- Zwiększenie czytelności obrazów medycznych, takich jak zdjęcia rentgenowskie czy rezonanse magnetyczne,
- Zwiększenie jakości filmów i obrazów istniejących cyfrowo.

8 Bibliografia

- Główne repozytorium
- `https://en.wikipedia.org/wiki/Super-resolution_imaging`
- `https://medium.com/analytics-vidhya/super-resolution-using-autoencoders-and-tf2-0-505215c1674`
- `https://ashrafur.medium.com/autoencoder-denoise-image-using-upsampling2d-and-conv2dtranspose-layers-part-1-8d88b9e81483`
- `https://onlinelibrary.wiley.com/doi/10.1002/ima.20007`
- `https://www.researchgate.net/publication/320658590_Deep_Clustering_with_Convolutional_Autoencoders`
- `https://www.kaggle.com/datasets/quadeer15sh/image-super-resolution-from-unsplash/data`
- Materiały na platformie YouTube