Autor (imię, nazwisko, grupa)

Generowanie nowych danych na przykładzie zbioru MNIST

Wstęp

Parę słów ogólnie o generowaniu danych

Przedstawienie jakie modele zostaną użyte (np. autoenkoder – krótko opisać co to jest)

Co jest celem projektu

Opis wybranego zbioru danych

(źródło skąd zbiór pochodzi, ile zawiera obserwacji, jakiego wymiaru są obrazy, czy są kolorowe czy czarno-białe, przedstawienie kilku przykładowych obrazów)

Przykładowo:

Do pokazania zastosowań modeli generatywnych został wybrany zbiór MNIST (https://keras.io/api/datasets/mnist/) . Zawiera on czarno-białe obrazy ręcznie pisanych cyfr od 0 do 9. Obrazy mają wymiar 28x28. Zbiór uczący składa się z 60 tysięcy obserwacji, a testowy z 10 tysięcy. Wartości każdego piksela mieszczą się w zakresie od 0 do 255. Do każdej obserwacji jest przypisana etykieta mówiąca o tym, jaka cyfra znajduje się na obrazie. Na rysunku 1 widać przykładowe dziesięć obserwacji ze zbioru MNIST.

Rysunek 1 Przykładowe obserwacje ze zbioru danych MNIST

Ładowanie danych jest przedstawione na poniższym listingu.

Listing 1. Załadowanie danych

```
1. from keras.datasets import mnist
2. (x_train, y_train), (x_test, y_test) = mnist.load_data()
3. x_train = np.expand_dims(x_train, axis=-1)
4. x_train_scaled = (x_train/255).copy()
```

Zastosowanie autoenkodera do generowania danych

Opis struktury autoenkodera,

Enkoder składa się z następujących warstw: (nazwy warstw, ich parametry)

Dekoder składa się z następujących warstw….

(można przedstawić schemat na rysunku)

Model autoenkodera jest trenowany przez X epok, z optymalizatorem… współczynnikiem uczenia…

Kod pozwalający na utworzenie i wytrenowanie autoenkodera znajduje się na listingu nr...

(listing)

(warto też wykonać i opisać wykres funkcji straty dla autoenkodera)

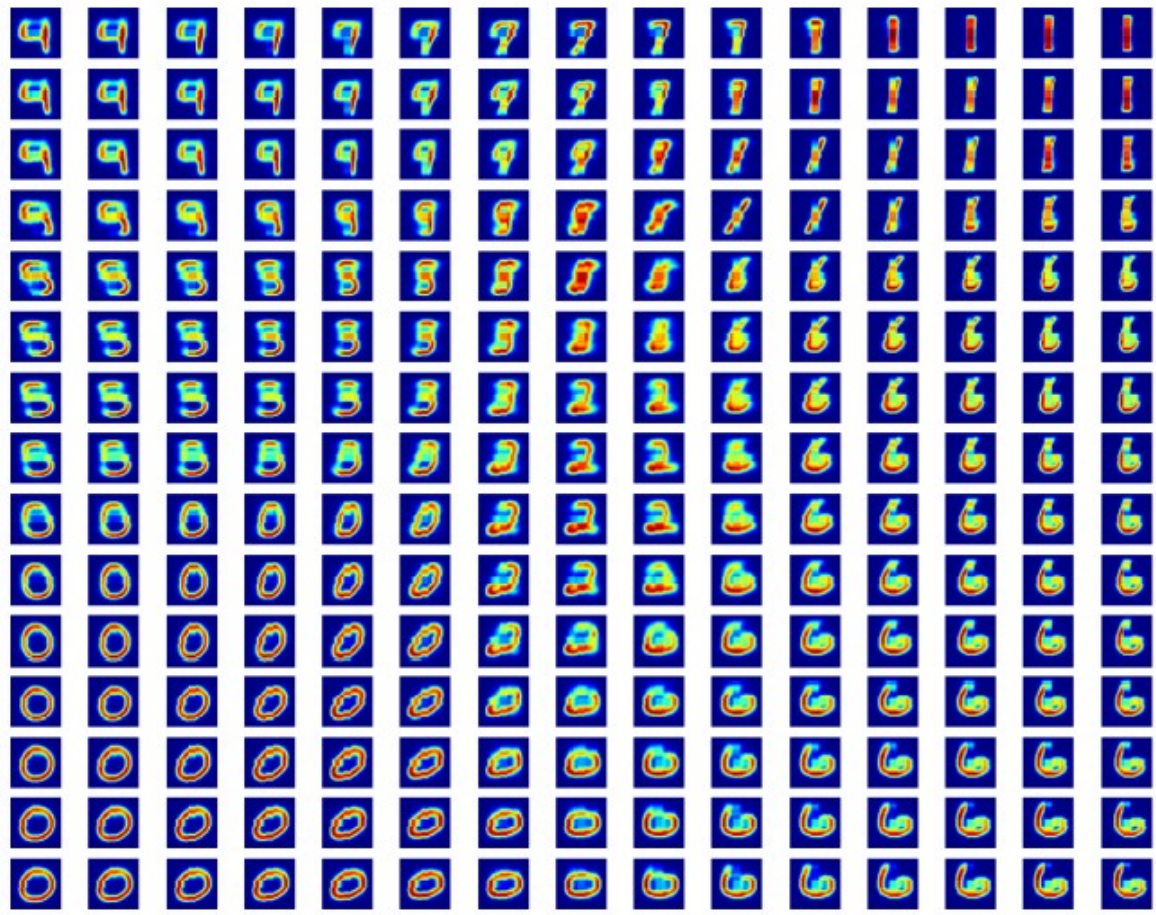
Opis procesu generowania danych przy użyciu autoenkodera

Kod służący do ilustracji wygenerowanych danych znajduje się na listingu nr...

Listing... Wyświetlenie wygenerowanych danych

```
num = 15
limit = 0.6
step = limit*2/num
fig, ax = plt.subplots(num, num, figsize = (20,16))
X_vals = np.arange(-limit, limit, step)
Y_vals = np.arange(-limit, limit, step)
for i, x in enumerate(X_vals):
 for j, y in enumerate(Y_vals):
     test_in = np.array([[x,y]])
     output = decoder.predict(x=test_in)
     output = np.squeeze(output)
     ax[-j-1,i].imshow(output, cmap = 'jet')
     ax[-j-1,i].axis('off')
```

Efekt zastosowania opisanego powyżej autoenkodera w zadaniu generowania danych jest przedstawiony na rysunku XX



Rysunek XX Dane wygenerowane przy użyciu autoenkodera

Zastosowanie rywalizującej sieci generatywnej GAN

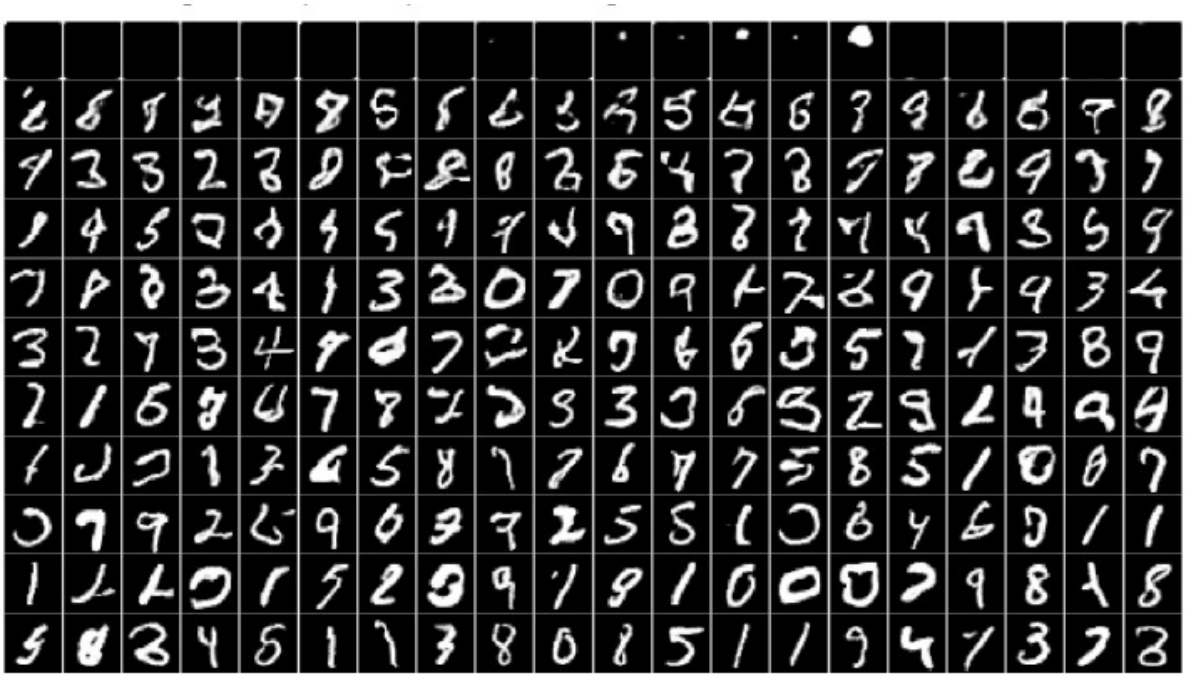
Sieć GAN będzie się składała z dwóch modeli: generatora i deskryptora.

Opis struktury deskryptora, listing z kodem. Opis kompilacji deskryptora (parametry optymalizatora!). Czy deskryptor jest wstępnie trenowany na danych ze zbioru rozszerzonego o szum z rozkładu normalnego?

Opis struktury generatora, listing z kodem.

Tworzenie modelu GAN – opis i listing

Uczenie modelu GAN – opis i listing



Rysunek YY Obrazy generowane przez model GAN (przedstawiona jest co piąta epoka przy 100 epokach uczenia)

Podsumowanie i wnioski

Czy oba modele udało się skutecznie zastosować do wybranego zbioru?

Jakie problemy zostały napotkane podczas realizacji (jeśli jakieś były)

Który model pozwala na generację lepszych danych

Co można poprawić w modelach lub jakiego innego modelu można użyć do generacji (np. poczytać o autoenkoderach warianyjnych – czym się różnią od „zwykłych” autoenkoderów)

Załączniki

Do projektu załączony jest plik/pliki z kodem w formacie .ipynb lub .py