2602191301 - Allin Setiawan

Import Dataset

```
!unzip '/content/t10k-images-idx3-ubyte.zip'
!unzip '/content/train-images-idx3-ubyte.zip'
!unzip '/content/train-labels-idx1-ubyte.zip'
!unzip '/content/t10k-labels-idx1-ubyte.zip'
```

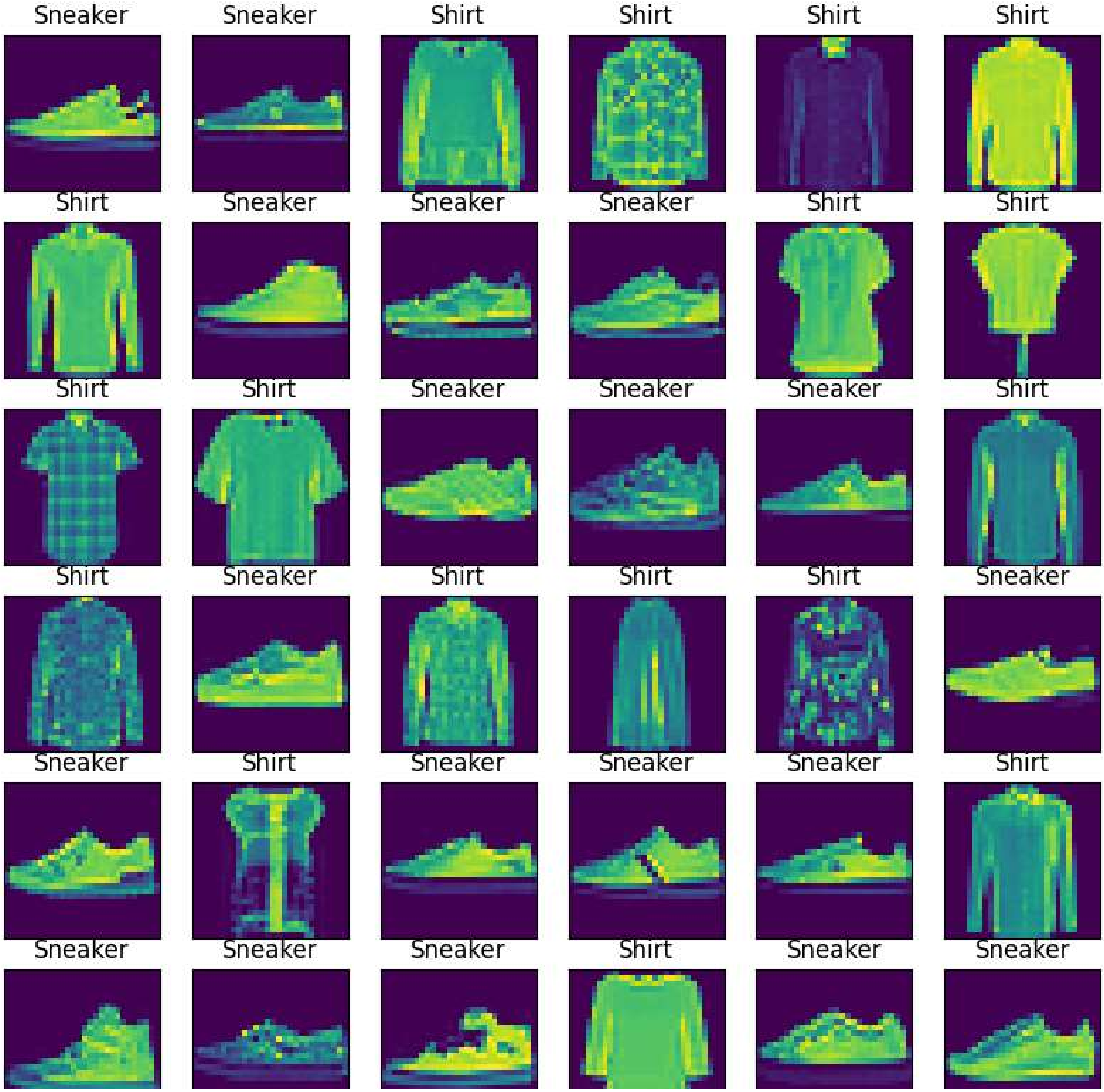
```
Archive:  /content/t10k-images-idx3-ubyte.zip
  inflating: t10k-images-idx3-ubyte
Archive:  /content/train-images-idx3-ubyte.zip
  inflating: train-images-idx3-ubyte
Archive:  /content/train-labels-idx1-ubyte.zip
  inflating: train-labels-idx1-ubyte
Archive:  /content/t10k-labels-idx1-ubyte.zip
  inflating: t10k-labels-idx1-ubyte
```

```
!pip install idx2numpy
import idx2numpy

x_train = idx2numpy.convert_from_file('train-images-idx3-ubyte')
y_train = idx2numpy.convert_from_file('train-labels-idx1-ubyte')

x_test = idx2numpy.convert_from_file('t10k-images-idx3-ubyte')
y_test = idx2numpy.convert_from_file('t10k-labels-idx1-ubyte')
```

```
Collecting idx2numpy
  Downloading idx2numpy-1.2.3.tar.gz (6.8 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from ic
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from idx2
Building wheels for collected packages: idx2numpy
  Building wheel for idx2numpy (setup.py) ... done
  Created wheel for idx2numpy: filename=idx2numpy-1.2.3-py3-none-any.whl size=7904 sha25
  Stored in directory: /root/.cache/pip/wheels/e0/f4/e7/643fc5f932ec2ff92997f43f007660fe
Successfully built idx2numpy
Installing collected packages: idx2numpy
Successfully installed idx2numpy-1.2.3
```

Preprocessing and Scaling Dataset

```
import matplotlib.pyplot as plt
import numpy as np

#Sesuai dengan soal mengambil label 'shirt' dan 'sneaker' untuk dataset ini
classes_of_interest = ['Shirt', 'Sneaker']
original_labels = [6, 7] #shirt = 6, sneaker = 7

mask = np.isin(y_train, original_labels)

# Filter pada training data
x_train_filtered = x_train[mask]
y_train_filtered = y_train[mask]

# Melakukan Mapping dari original labels ke new labels (0 for 'Shirt', 1 for 'Sneaker')
mapping = {6: 0, 7: 1}
y_train_filtered = np.array([mapping[label] for label in y_train_filtered])

class_names = ['Shirt', 'Sneaker']

plt.figure(figsize=(10, 10))
for i in range(min(36, len(x_train_filtered))):
    plt.subplot(6, 6, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train_filtered[i].reshape((28,28)))
    label_index = int(y_train_filtered[i])
    plt.title(class_names[label_index])
plt.show()
```



Saya membuat class baru untuk model autoencoder ini dikarenakan pada soal kelas yang diberikan hanya 'shirt' dan 'sneakers'

```
#Scaling Dataset
x_train_filtered = x_train_filtered.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split

# Melakukan split data training 80% dan remaining untuk (validation dan test)
x_train, x_remaining, y_train, y_remaining = train_test_split(
    x_train_filtered, y_train_filtered, test_size=0.2, random_state=42
)

# Melakukan split data remaining dibagi menjadi ke test dan validation
x_val, x_test, y_val, y_test = train_test_split(
    x_remaining, y_remaining, test_size=0.5, random_state=42
)


print("Train shapes:", x_train.shape, y_train.shape)
print("Validation shapes:", x_val.shape, y_val.shape)
print("Test shapes:", x_test.shape, y_test.shape)
```

```
Train shapes: (9600, 28, 28) (9600,)
Validation shapes: (1200, 28, 28) (1200,)
Test shapes: (1200, 28, 28) (1200,)
```

Model Autoencoder

```
!pip install scikit-image

import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from skimage.metrics import structural_similarity as ssim

input_dim = 784
encoding_dim = 128

input_img = Input(shape=(input_dim,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(input_dim, activation='sigmoid')(encoded)

autoencoder = Model(input_img, decoded)

autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

```
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: numpy>=1.17.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in /usr/local/lib/p
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-package
```

```
autoencoder.fit(x_train_filtered.reshape(-1, input_dim), x_train_filtered.reshape(-1, input_
                epochs=20,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test.reshape(-1, input_dim), x_test.reshape(-1, input_dim
```

```
Epoch 1/20
47/47 [==============================] - 3s 7ms/step - loss: 0.4598 - val_loss: 0.3673
Epoch 2/20
47/47 [==============================] - 0s 4ms/step - loss: 0.3443 - val_loss: 0.3308
Epoch 3/20
47/47 [==============================] - 0s 4ms/step - loss: 0.3201 - val_loss: 0.3155
Epoch 4/20
47/47 [==============================] - 0s 4ms/step - loss: 0.3081 - val_loss: 0.3064
Epoch 5/20
47/47 [==============================] - 0s 4ms/step - loss: 0.3004 - val_loss: 0.2998
Epoch 6/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2950 - val_loss: 0.2957
Epoch 7/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2911 - val_loss: 0.2920
Epoch 8/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2881 - val_loss: 0.2891
Epoch 9/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2853 - val_loss: 0.2866
Epoch 10/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2829 - val_loss: 0.2843
Epoch 11/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2808 - val_loss: 0.2827
Epoch 12/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2789 - val_loss: 0.2807
Epoch 13/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2774 - val_loss: 0.2793
Epoch 14/20
47/47 [==============================] - 0s 5ms/step - loss: 0.2759 - val_loss: 0.2783
Epoch 15/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2747 - val_loss: 0.2766
Epoch 16/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2735 - val_loss: 0.2759
Epoch 17/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2723 - val_loss: 0.2745
Epoch 18/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2714 - val_loss: 0.2736
Epoch 19/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2705 - val_loss: 0.2728
Epoch 20/20
47/47 [==============================] - 0s 4ms/step - loss: 0.2697 - val_loss: 0.2722
<keras.src.callbacks.History at 0x7c7a384ab160>
```

```
decoded_imgs = autoencoder.predict(x_test.reshape(-1, input_dim))
```

```
38/38 [==============================] - 0s 2ms/step
```

Evaluate With SSIM Scores

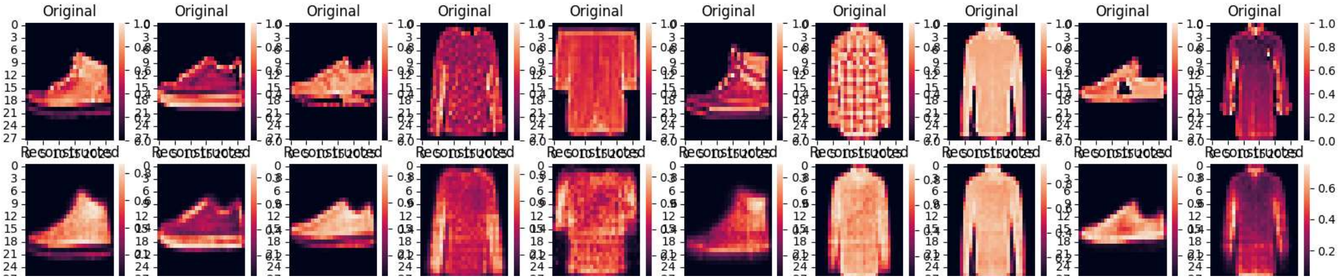
SSIM Scores -> Indeks SSIM yang dihasilkan adalah nilai desimal antara -1 dan 1, di mana 1 menunjukkan kesamaan sempurna, 0 menunjukkan tidak ada kesamaan, dan -1 menunjukkan anti-korelasi sempurna.

```
ssim_scores = []
for i in range(x_test.shape[0]):
    original_img = x_test[i].reshape(28, 28)
    img_predict = decoded_imgs[i].reshape(28, 28)

    ssim_score = ssim(original_img, img_predict, data_range=original_img.max() - original_im
    ssim_scores.append(ssim_score)
average_ssim = np.mean(ssim_scores)
print("Average SSIM on test data:", average_ssim)
```

```
Average SSIM on test data: 0.7657012151596361
```

Nilai rata-rata ssim pada test data untuk model ini adalah 0.76 yang mendekati nilai 1 berarti model ini dapat menunjukkan adanya kesamaan yang sempurna pada gambar yang digenerate dengan original gambarnya

```
import seaborn as sns
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
  ax = plt.subplot(2, n, i + 1)
  sns.heatmap(x_test[i].reshape(28, 28), ax=ax)
  ax.set_title("Original")
  ax.set_xlabel('')
  ax.set_ylabel('')

  ax = plt.subplot(2, n, i + 1 + n)
  sns.heatmap(decoded_imgs[i].reshape(28, 28), ax=ax)
  ax.set_title('Reconstructed')
  ax.set_xlabel('')
  ax.set_ylabel('')

plt.show()
```


Original
Original
Original
Original
Original
Original
Original
Original
Original
Original
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
0
3
6
9
12
15
18
21
24
27
1.0
0.8
0.6
0.4
0.2
0.0