asignment10

April 15, 2021

Necessary Libraries

[1]:
```
import tensorflow
from tensorflow import keras
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Conv2DTranspose
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Reshape
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras import backend as K
import numpy as np
from tensorflow.keras.layers import Dense, Input, Conv2D, LSTM, MaxPool2D,␣
↪UpSampling2D
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import to_categorical
from numpy import argmax, array_equal
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.models import Model
from random import randint
import pandas as pd
import numpy as np
from tensorflow.keras import layers
from PIL import Image
from tensorflow.keras import regularizers
from tensorflow.keras import backend
from tensorflow.keras.callbacks import TensorBoard
import tensorflow as tf
```

1 Loading and normalizing

[2]:
```
input_shape = (28, 28, 1)

# the data, split between train and test sets
(x_train_valid, y_train_valid), (x_test, y_test) = keras.datasets.fashion_mnist.
 ↪load_data()
x_train, x_validation, y_train, y_validation = train_test_split(x_train_valid,␣
 ↪y_train_valid, test_size=0.2, random_state=0)

# Scale images to the [0, 1] range
x_train = x_train.astype("float32") / 255
x_validation = x_validation.astype("float32") / 255
x_test = x_test.astype("float32") / 255
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-labels-idx1-ubyte.gz
32768/29515 [==============================] - 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 10s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 2s 0us/step
```

[3]:
```
x_train.shape, x_validation.shape, x_test.shape
```

[3]:
```
((48000, 28, 28), (12000, 28, 28), (10000, 28, 28))
```

2 Flattening

[4]:
```
# Reshape the images into flat ANN layers
x_train = x_train.reshape(-1, 784)
x_validation = x_validation.reshape(-1, 784)
x_test = x_test.reshape(-1, 784)

x_train.shape, x_validation.shape, x_test.shape
```

[4]:
```
((48000, 784), (12000, 784), (10000, 784))
```

3 Task 1

[5]:
```
## input layer
input_layer = Input(shape=(784,))    # 28*28

## encoding architecture
encode_layer1 = Dense(128, activation='relu')(input_layer)
encode_layer2 = Dense(64, activation='relu')(encode_layer1)
encode_layer3 = Dense(32, activation='relu')(encode_layer2)


## decoding architecture
decode_layer1 = Dense(64, activation='relu')(encode_layer3)
decode_layer2 = Dense(128, activation='relu')(decode_layer1)

## output layer
output_layer  = Dense(784)(decode_layer2)

model = Model(input_layer, output_layer)

model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 784)]             0
_________________________________________________________________
dense (Dense)                (None, 128)               100480
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_4 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_5 (Dense)              (None, 784)               101136
=================================================================
Total params: 222,384
Trainable params: 222,384
Non-trainable params: 0
_________________________________________________________________
```

[6]:
```
model.compile(optimizer='adam', loss='mse')
```

```
early_stopping = EarlyStopping(monitor='val_loss', min_delta=0, patience=10,␣
↪verbose=1, mode='auto')
tboard_callback = tf.keras.callbacks.TensorBoard(log_dir = "/tmp/autoencoder",␣
↪histogram_freq = 1, profile_batch = '500,520')


model.fit(x_train, x_train, epochs=30, batch_size=2048,␣
↪validation_data=(x_validation, x_validation), callbacks=[early_stopping,␣
↪tboard_callback])
```

```
Epoch 1/30
24/24 [==============================] - 2s 45ms/step - loss: 0.1579 - val_loss:
0.0638
Epoch 2/30
24/24 [==============================] - 1s 21ms/step - loss: 0.0583 - val_loss:
0.0480
Epoch 3/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0460 - val_loss:
0.0396
Epoch 4/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0377 - val_loss:
0.0333
Epoch 5/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0318 - val_loss:
0.0287
Epoch 6/30
24/24 [==============================] - 1s 21ms/step - loss: 0.0280 - val_loss:
0.0267
Epoch 7/30
24/24 [==============================] - 1s 21ms/step - loss: 0.0264 - val_loss:
0.0256
Epoch 8/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0252 - val_loss:
0.0247
Epoch 9/30
24/24 [==============================] - 1s 21ms/step - loss: 0.0244 - val_loss:
0.0239
Epoch 10/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0236 - val_loss:
0.0232
Epoch 11/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0230 - val_loss:
0.0226
Epoch 12/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0223 - val_loss:
0.0220
```

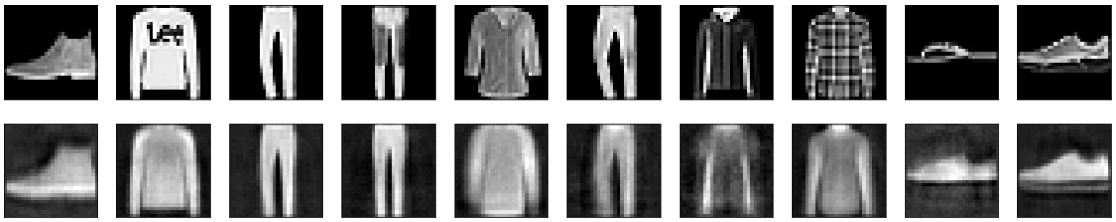
```
Epoch 13/30
24/24 [==============================] - 1s 21ms/step - loss: 0.0217 - val_loss:
0.0216
Epoch 14/30
24/24 [==============================] - 1s 23ms/step - loss: 0.0213 - val_loss:
0.0210
Epoch 15/30
24/24 [==============================] - 1s 23ms/step - loss: 0.0208 - val_loss:
0.0207
Epoch 16/30
24/24 [==============================] - 1s 23ms/step - loss: 0.0204 - val_loss:
0.0203
Epoch 17/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0201 - val_loss:
0.0201
Epoch 18/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0199 - val_loss:
0.0198
Epoch 19/30
24/24 [==============================] - 1s 23ms/step - loss: 0.0196 - val_loss:
0.0197
Epoch 20/30
24/24 [==============================] - 1s 23ms/step - loss: 0.0195 - val_loss:
0.0194
Epoch 21/30
24/24 [==============================] - 1s 27ms/step - loss: 0.0192 - val_loss:
0.0193
Epoch 22/30
24/24 [==============================] - 1s 40ms/step - loss: 0.0191 - val_loss:
0.0190
Epoch 23/30
24/24 [==============================] - 1s 23ms/step - loss: 0.0188 - val_loss:
0.0189
Epoch 24/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0187 - val_loss:
0.0186
Epoch 25/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0185 - val_loss:
0.0185
Epoch 26/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0182 - val_loss:
0.0184
Epoch 27/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0182 - val_loss:
0.0183
Epoch 28/30
24/24 [==============================] - 1s 23ms/step - loss: 0.0180 - val_loss:
0.0181
```

```
Epoch 29/30
24/24 [==============================] - 1s 22ms/step - loss: 0.0180 - val_loss:
0.0180
Epoch 30/30
24/24 [==============================] - 1s 21ms/step - loss: 0.0178 - val_loss:
0.0178
```

[6]:
```
<tensorflow.python.keras.callbacks.History at 0x7f765f295130>
```

[7]:
```
predictions = model.predict(x_test)
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # reconstruction
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(predictions[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```



[8]:
```
# Load the TensorBoard notebook extension.
%load_ext tensorboard
%tensorboard --logdir=/tmp/autoencoder

# can also run this in the terminal window:
# tensorboard --logdir=/tmp/autoencoder
# and go to http://localhost:6006/#scalars&run=train
```

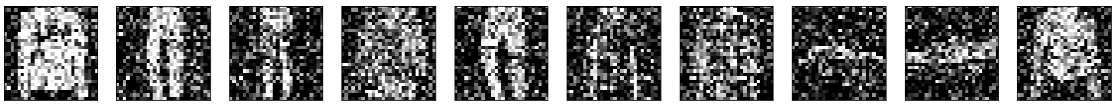
```
<IPython.core.display.HTML object>
```

4 Task 2

[9]:
```
noise_factor = 0.4
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0, scale=1.0,␣
 ↪size=x_train.shape)
x_validation_noisy = x_validation + noise_factor * np.random.normal(loc=0.0,␣
 ↪scale=1.0, size=x_validation.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0,␣
 ↪size=x_test.shape)

x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_validation_noisy = np.clip(x_validation_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)
```

[10]:
```
n = 10
plt.figure(figsize=(20, 2))
for i in range(1, n + 1):
    ax = plt.subplot(1, n, i)
    plt.imshow(x_test_noisy[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```



[11]:
```
x_train_noisy=x_train_noisy.reshape(-1, 28, 28, 1)
x_validation_noisy=x_validation_noisy.reshape(-1, 28, 28, 1)


x_train=x_train.reshape(-1, 28, 28, 1)
x_validation=x_validation.reshape(-1, 28, 28, 1)
```

[12]:
```
x_train_noisy.shape
```

[12]:
```
(48000, 28, 28, 1)
```

[13]:
```
input_layer = keras.Input(shape=(28, 28, 1))

x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(input_layer)
x = layers.MaxPooling2D((2, 2), padding='same')(x)
x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(x)
```

```
encoded = layers.MaxPooling2D((2, 2), padding='same')(x)

# At this point the representation is (7, 7, 32)

x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)
x = layers.UpSampling2D((2, 2))(x)
x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = layers.UpSampling2D((2, 2))(x)
decoded = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = keras.Model(input_layer, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

[14]:
```
autoencoder.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 32)          9248
_________________________________________________________________
up_sampling2d (UpSampling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 32)        9248
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 28, 28, 1)         289
=================================================================
Total params: 28,353
Trainable params: 28,353
Non-trainable params: 0
_________________________________________________________________
```

[15]:
```
autoencoder.fit(x_train_noisy, x_train,
                epochs=30,
                batch_size=2048,
```

```
                shuffle=True,
                validation_data=(x_validation_noisy, x_validation))
```

```
Epoch 1/30
24/24 [==============================] - 120s 2s/step - loss: 0.6161 - val_loss:
0.3933
Epoch 2/30
24/24 [==============================] - 41s 2s/step - loss: 0.3763 - val_loss:
0.3415
Epoch 3/30
24/24 [==============================] - 41s 2s/step - loss: 0.3368 - val_loss:
0.3240
Epoch 4/30
24/24 [==============================] - 42s 2s/step - loss: 0.3229 - val_loss:
0.3175
Epoch 5/30
24/24 [==============================] - 41s 2s/step - loss: 0.3176 - val_loss:
0.3139
Epoch 6/30
24/24 [==============================] - 42s 2s/step - loss: 0.3130 - val_loss:
0.3110
Epoch 7/30
24/24 [==============================] - 41s 2s/step - loss: 0.3106 - val_loss:
0.3085
Epoch 8/30
24/24 [==============================] - 41s 2s/step - loss: 0.3082 - val_loss:
0.3065
Epoch 9/30
24/24 [==============================] - 41s 2s/step - loss: 0.3062 - val_loss:
0.3048
Epoch 10/30
24/24 [==============================] - 41s 2s/step - loss: 0.3045 - val_loss:
0.3041
Epoch 11/30
24/24 [==============================] - 41s 2s/step - loss: 0.3046 - val_loss:
0.3029
Epoch 12/30
24/24 [==============================] - 41s 2s/step - loss: 0.3023 - val_loss:
0.3021
Epoch 13/30
24/24 [==============================] - 41s 2s/step - loss: 0.3024 - val_loss:
0.3010
Epoch 14/30
24/24 [==============================] - 44s 2s/step - loss: 0.3014 - val_loss:
0.3004
Epoch 15/30
24/24 [==============================] - 45s 2s/step - loss: 0.3006 - val_loss:
```

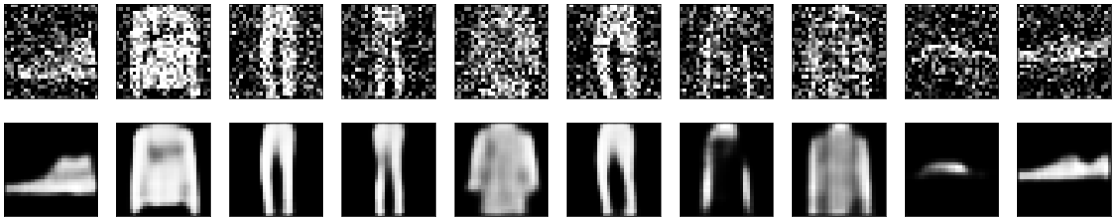
```
0.2999
Epoch 16/30
24/24 [==============================] - 43s 2s/step - loss: 0.2991 - val_loss:
0.2991
Epoch 17/30
24/24 [==============================] - 44s 2s/step - loss: 0.2989 - val_loss:
0.2989
Epoch 18/30
24/24 [==============================] - 44s 2s/step - loss: 0.2986 - val_loss:
0.2980
Epoch 19/30
24/24 [==============================] - 43s 2s/step - loss: 0.2984 - val_loss:
0.2978
Epoch 20/30
24/24 [==============================] - 42s 2s/step - loss: 0.2981 - val_loss:
0.2974
Epoch 21/30
24/24 [==============================] - 43s 2s/step - loss: 0.2968 - val_loss:
0.2968
Epoch 22/30
24/24 [==============================] - 42s 2s/step - loss: 0.2970 - val_loss:
0.2969
Epoch 23/30
24/24 [==============================] - 43s 2s/step - loss: 0.2970 - val_loss:
0.2967
Epoch 24/30
24/24 [==============================] - 43s 2s/step - loss: 0.2962 - val_loss:
0.2958
Epoch 25/30
24/24 [==============================] - 42s 2s/step - loss: 0.2959 - val_loss:
0.2954
Epoch 26/30
24/24 [==============================] - 43s 2s/step - loss: 0.2960 - val_loss:
0.2952
Epoch 27/30
24/24 [==============================] - 43s 2s/step - loss: 0.2951 - val_loss:
0.2955
Epoch 28/30
24/24 [==============================] - 42s 2s/step - loss: 0.2950 - val_loss:
0.2947
Epoch 29/30
24/24 [==============================] - 42s 2s/step - loss: 0.2945 - val_loss:
0.2951
Epoch 30/30
24/24 [==============================] - 41s 2s/step - loss: 0.2947 - val_loss:
0.2945
```

[15]:
```
<tensorflow.python.keras.callbacks.History at 0x7f7603c765b0>
```

[16]:
```
predictions = autoencoder.predict(x_test.reshape((-1, 28, 28, 1)))

n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # noisy
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test_noisy[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # reconstruction
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(predictions[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```



[ ]: