Ex9

June 15, 2022

1 Task 1: Classification

- Create a Neural Network model
- Define optimization procedure
- Train Classifier
- Evaluate model on test set

1.1 Dataset

[1]:
```
!nvidia-smi
```

```
Tue Jun 14 13:23:07 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   69C    P8    11W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

[2]:
```
USE_GPU = 0
```

[3]:
```
# Import TensorFlow
import tensorflow as tf

# Print the installed TensorFlow version
```

```
print(f'TensorFlow version: {tf.__version__}\n')

# Get all GPU devices on this server
gpu_devices = tf.config.list_physical_devices('GPU')

# Print the name and the type of all GPU devices
print('Available GPU Devices:')
for gpu in gpu_devices:
    print(' ', gpu.name, gpu.device_type)

# Set only the GPU specified as USE_GPU to be visible
tf.config.set_visible_devices(gpu_devices[USE_GPU], 'GPU')

# Get all visible GPU  devices on this server
visible_devices = tf.config.get_visible_devices('GPU')

# Print the name and the type of all visible GPU devices
print('\nVisible GPU Devices:')
for gpu in visible_devices:
    print(' ', gpu.name, gpu.device_type)

# Set the visible device(s) to not allocate all available memory at once,
# but rather let the memory grow whenever needed
for gpu in visible_devices:
    tf.config.experimental.set_memory_growth(gpu, True)
```

```
TensorFlow version: 2.8.2

Available GPU Devices:
  /physical_device:GPU:0 GPU

Visible GPU Devices:
  /physical_device:GPU:0 GPU
```

[4]:
```
fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.
 ↪load_data()
```

[5]:
```
# Add color channel
train_images = train_images.reshape((train_images.shape[0], 28, 28, 1))
test_images = test_images.reshape((test_images.shape[0], 28, 28, 1))
```

[6]:
```
# Normalization
train_images = train_images / 255.0
test_images = test_images / 255.0
```

[7]:
```
from sklearn.model_selection import train_test_split

# Get training and validation data
X_train, X_val, y_train, y_val = train_test_split(train_images, train_labels,␣
 ↪test_size=0.1)
```

[8]:
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

[9]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i, :, :, 0], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

[10]:
```
X_train[0].shape
```

[10]:
```
(28, 28, 1)
```

[11]:
```
input = tf.keras.Input(shape=(28, 28, 1))
X = tf.keras.layers.Conv2D(16, (3,3), strides=(2,2), padding='same')(input)
X = tf.keras.layers.Dropout(0.5)(X)
# X = tf.keras.layers.BatchNormalization()(X)
X = tf.keras.layers.MaxPool2D((2,2), padding='same')(X)

X = tf.keras.layers.Conv2D(32, (3,3), strides=(2,2), padding='same')(X)
X = tf.keras.layers.Dropout(0.5)(X)
# X = tf.keras.layers.BatchNormalization()(X)
```

```
X = tf.keras.layers.MaxPool2D((2,2), padding='same')(X)

X = tf.keras.layers.Conv2D(64, (3,3), strides=(2,2), padding='same')(X)
X = tf.keras.layers.Dropout(0.5)(X)
# X = tf.keras.layers.BatchNormalization()(X)
X = tf.keras.layers.MaxPool2D((2,2), padding='same')(X)

X = tf.keras.layers.Flatten()(X)
# X = tf.keras.layers.Dense(100, activation="relu")(X)
X = tf.keras.layers.Dense(64, activation="relu")(X)
X = tf.keras.layers.Dense(10, activation="softmax")(X)
model = tf.keras.Model(input, X)
```

[12]:
```
# Decay learning rate according to #epoch
def scheduler(epoch, lr=0.01):
  if epoch < 5:
    return lr
  else:
    return lr * tf.math.exp(-0.1)
```

[13]:
```
# Monitor the validation loss and schedule lr automatically
callbacks = [tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5),
                    tf.keras.callbacks.LearningRateScheduler(scheduler,␣
 ↪verbose=1)]
```

[14]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(),␣
 ↪loss="sparse_categorical_crossentropy", metrics=["accuracy"])
```

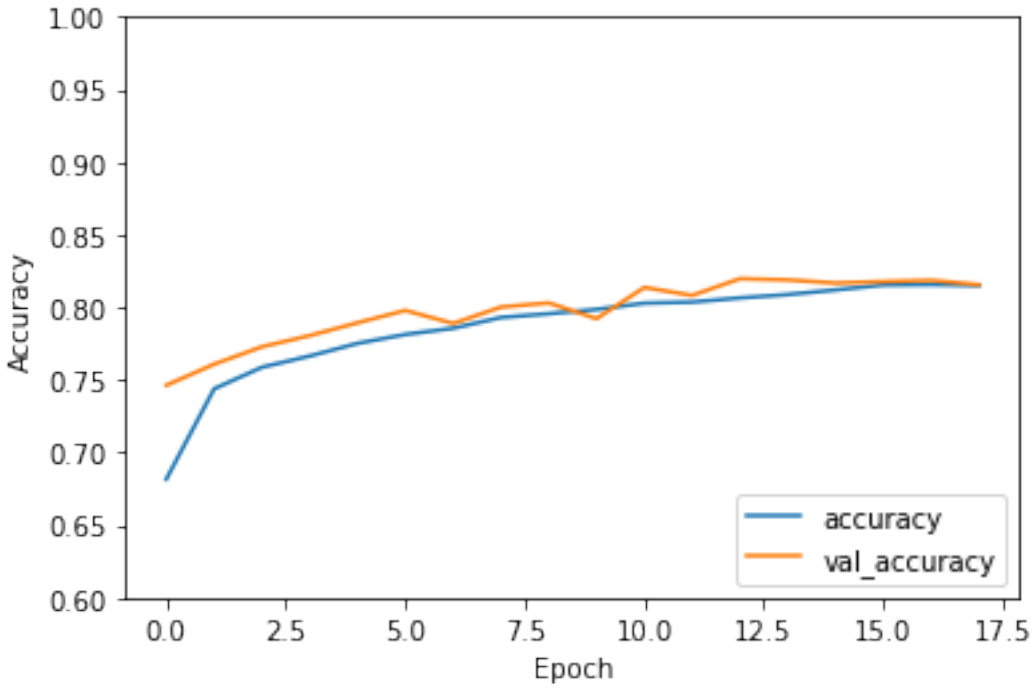
[15]:
```
history = model.fit(X_train, y_train, epochs=50, batch_size=8,␣
 ↪validation_data=(X_val,y_val), callbacks=callbacks)
```

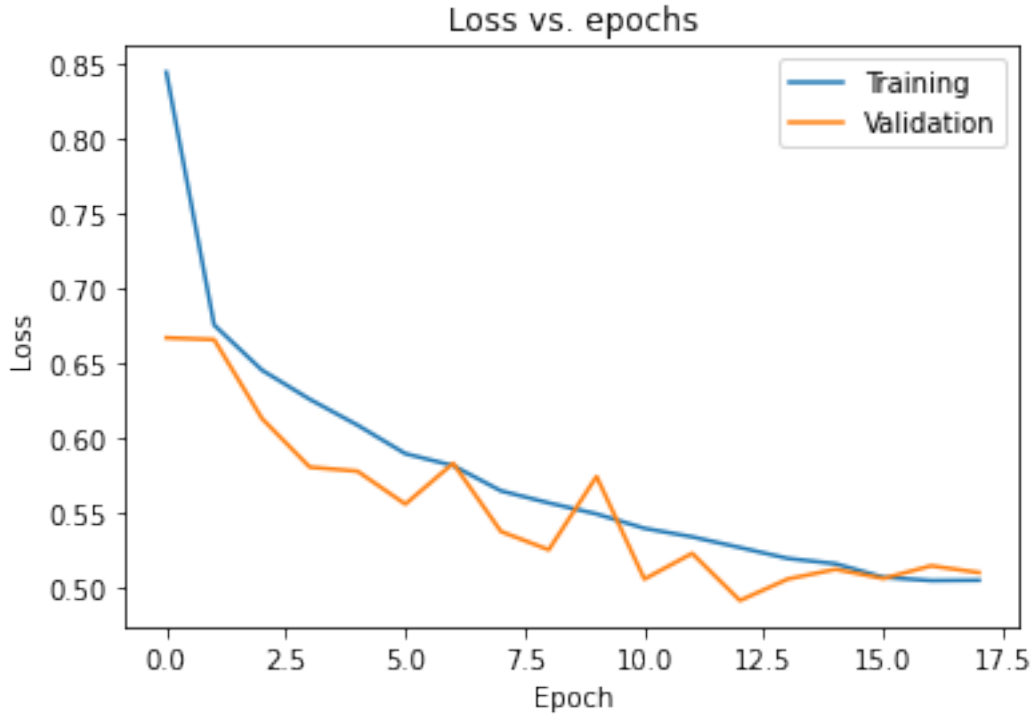
```
Epoch 1: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 1/50
6750/6750 [==============================] - 33s 4ms/step - loss: 0.8440 -
accuracy: 0.6818 - val_loss: 0.6671 - val_accuracy: 0.7463 - lr: 0.0010

Epoch 2: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 2/50
6750/6750 [==============================] - 22s 3ms/step - loss: 0.6756 -
accuracy: 0.7439 - val_loss: 0.6659 - val_accuracy: 0.7608 - lr: 0.0010

Epoch 3: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 3/50
6750/6750 [==============================] - 22s 3ms/step - loss: 0.6455 -
accuracy: 0.7587 - val_loss: 0.6133 - val_accuracy: 0.7728 - lr: 0.0010
```

```
Epoch 4: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 4/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.6261 -
accuracy: 0.7665 - val_loss: 0.5809 - val_accuracy: 0.7805 - lr: 0.0010

Epoch 5: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 5/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.6089 -
accuracy: 0.7751 - val_loss: 0.5782 - val_accuracy: 0.7893 - lr: 0.0010

Epoch 6: LearningRateScheduler setting learning rate to 0.0009048373904079199.
Epoch 6/50
6750/6750 [==============================] - 22s 3ms/step - loss: 0.5899 -
accuracy: 0.7813 - val_loss: 0.5562 - val_accuracy: 0.7978 - lr: 9.0484e-04

Epoch 7: LearningRateScheduler setting learning rate to 0.0008187306812033057.
Epoch 7/50
6750/6750 [==============================] - 22s 3ms/step - loss: 0.5819 -
accuracy: 0.7856 - val_loss: 0.5834 - val_accuracy: 0.7888 - lr: 8.1873e-04

Epoch 8: LearningRateScheduler setting learning rate to 0.0007404818097256124.
Epoch 8/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.5651 -
accuracy: 0.7930 - val_loss: 0.5380 - val_accuracy: 0.8002 - lr: 7.4082e-04

Epoch 9: LearningRateScheduler setting learning rate to 0.0006703198888740778.
Epoch 9/50
6750/6750 [==============================] - 22s 3ms/step - loss: 0.5570 -
accuracy: 0.7956 - val_loss: 0.5258 - val_accuracy: 0.8030 - lr: 6.7032e-04

Epoch 10: LearningRateScheduler setting learning rate to 0.0006065304623916745.
Epoch 10/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.5497 -
accuracy: 0.7985 - val_loss: 0.5746 - val_accuracy: 0.7922 - lr: 6.0653e-04

Epoch 11: LearningRateScheduler setting learning rate to 0.0005488114547915757.
Epoch 11/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.5401 -
accuracy: 0.8028 - val_loss: 0.5063 - val_accuracy: 0.8137 - lr: 5.4881e-04

Epoch 12: LearningRateScheduler setting learning rate to 0.0004965850966982543.
Epoch 12/50
6750/6750 [==============================] - 22s 3ms/step - loss: 0.5344 -
accuracy: 0.8036 - val_loss: 0.5234 - val_accuracy: 0.8082 - lr: 4.9659e-04

Epoch 13: LearningRateScheduler setting learning rate to 0.0004493287415243685.
Epoch 13/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.5273 -
```

```
accuracy: 0.8064 - val_loss: 0.4920 - val_accuracy: 0.8197 - lr: 4.4933e-04

Epoch 14: LearningRateScheduler setting learning rate to 0.0004065694229211658.
Epoch 14/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.5201 -
accuracy: 0.8088 - val_loss: 0.5063 - val_accuracy: 0.8188 - lr: 4.0657e-04

Epoch 15: LearningRateScheduler setting learning rate to 0.00036787919816561043.
Epoch 15/50
6750/6750 [==============================] - 24s 4ms/step - loss: 0.5165 -
accuracy: 0.8119 - val_loss: 0.5128 - val_accuracy: 0.8167 - lr: 3.6788e-04

Epoch 16: LearningRateScheduler setting learning rate to 0.0003332870840793475 5.
Epoch 16/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.5075 -
accuracy: 0.8154 - val_loss: 0.5067 - val_accuracy: 0.8177 - lr: 3.3287e-04

Epoch 17: LearningRateScheduler setting learning rate to 0.00030119396978989244.
Epoch 17/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.5053 -
accuracy: 0.8156 - val_loss: 0.5151 - val_accuracy: 0.8185 - lr: 3.0119e-04

Epoch 18: LearningRateScheduler setting learning rate to 0.0002725315571291 29696.
Epoch 18/50
6750/6750 [==============================] - 23s 3ms/step - loss: 0.5055 -
accuracy: 0.8151 - val_loss: 0.5106 - val_accuracy: 0.8155 - lr: 2.7253e-04
```

[16]:
```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.6, 1])
plt.legend(loc='lower right')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss vs. epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='upper right')
plt.show()
```


1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
Accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
accuracy
val_accuracy


Loss vs. epochs
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
Loss
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
Training
Validation

[17]:
```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
313/313 - 1s - loss: 0.5082 - accuracy: 0.8091 - 891ms/epoch - 3ms/step

Test accuracy: 0.8090999722480774
```

[18]:
```
predictions = model.predict(test_images[:25])
```

[19]:
```
import numpy as np

predicted_labels = np.array(class_names)[np.argmax(predictions, axis=-1)]
```

[20]:
```
print(predicted_labels)
```

```
['Ankle boot' 'Pullover' 'Trouser' 'Trouser' 'Shirt' 'Trouser' 'Coat'
 'Shirt' 'Sneaker' 'Sneaker' 'Coat' 'Sneaker' 'Bag' 'Dress' 'Coat'
 'Trouser' 'Pullover' 'Shirt' 'Bag' 'T-shirt/top' 'Pullover' 'Sneaker'
 'Sneaker' 'Ankle boot' 'Trouser']
```

[21]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i, :, :, 0], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]] + " - " + predicted_labels[i])
plt.show()
```

2 Task 2: Train Autoencoder

- Create a Neural Network model
- Define optimization procedure
- Train Classifier
- Evaluate model on test set

[22]:
```
fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.
 ↪load_data()

train_images = train_images.reshape((train_images.shape[0], 28, 28, 1))
```

```
test_images = test_images.reshape((test_images.shape[0], 28, 28, 1))
```

[23]:
```
def addGaussianNoise(mean=0.0, dev=1.0):
  gaussian_noise = mean + dev * np.random.rand(28, 28, 1)
  return gaussian_noise
```

[24]:
```
gaussian_noise = addGaussianNoise(dev=50)
```

[25]:
```
train_image = train_images.copy() + gaussian_noise
test_image = test_images.copy() + gaussian_noise
```

[26]:
```
train_image = train_image / 255.0
test_image = test_image / 255.0

train_images = train_images / 255.0
test_images = test_images / 255.0
```

[27]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i, :, :, 0], cmap=plt.cm.gray)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

[28]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_image[i, :, :, 0], cmap=plt.cm.gray)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

[29]:
```
input_img = tf.keras.Input(shape=(28, 28, 1))

x = tf.keras.layers.Conv2D(filters = 16, kernel_size = (3, 3),␣
 ↪activation='relu', padding='same')(input_img)
x = tf.keras.layers.MaxPooling2D(pool_size = (2, 2), padding='same')(x)

x = tf.keras.layers.Conv2D(filters = 8, kernel_size = (3, 3),␣
 ↪activation='relu', padding='same')(x)
x = tf.keras.layers.Conv2D(filters = 8, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same')(x)
encoded = tf.keras.layers.MaxPooling2D(pool_size = (2, 2), padding='same')(x)
```

```
x = tf.keras.layers.Conv2DTranspose(8, (3, 3), activation='relu',␣
 ↪padding='same')(encoded)
x = tf.keras.layers.UpSampling2D((2, 2))(x)

x = tf.keras.layers.Conv2DTranspose(8, (3, 3), activation='relu',␣
 ↪padding='same')(x)
x = tf.keras.layers.Conv2DTranspose(16, (3, 3), activation='relu',␣
 ↪padding='same')(x)
x = tf.keras.layers.UpSampling2D((2, 2))(x)

decoded = tf.keras.layers.Conv2D(1, (3, 3), activation='sigmoid',␣
 ↪padding='same')(x)
```

[30]:
```
autoencoder = tf.keras.Model(input_img, decoded)
```

[31]:
```
autoencoder.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 28, 28, 1)]       0

 conv2d_3 (Conv2D)           (None, 28, 28, 16)        160

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 16)       0
 2D)

 conv2d_4 (Conv2D)           (None, 14, 14, 8)         1160

 conv2d_5 (Conv2D)           (None, 14, 14, 8)         584

 max_pooling2d_4 (MaxPooling  (None, 7, 7, 8)          0
 2D)

 conv2d_transpose (Conv2DTra  (None, 7, 7, 8)          584
 nspose)

 up_sampling2d (UpSampling2D  (None, 14, 14, 8)        0
 )

 conv2d_transpose_1 (Conv2DT  (None, 14, 14, 8)        584
 ranspose)

 conv2d_transpose_2 (Conv2DT  (None, 14, 14, 16)       1168
 ranspose)
```

```
up_sampling2d_1 (UpSampling  (None, 28, 28, 16)       0
2D)

conv2d_6 (Conv2D)            (None, 28, 28, 1)         145

=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
_________________________________________________________________
```

[32]:
```
autoencoder.compile(optimizer='adam', loss='mse')
```

[33]:
```
callbacks = [tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=2),
                    tf.keras.callbacks.LearningRateScheduler(scheduler,␣
 ↪verbose=1)]
```

[34]:
```
history = autoencoder.fit(train_image, train_image,
                epochs=50,
                batch_size=8,
                shuffle=True,
                validation_data=(test_image, test_image),callbacks=callbacks
               )
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 1/50
7500/7500 [==============================] - 32s 4ms/step - loss: 0.0146 -
val_loss: 0.0110 - lr: 0.0010

Epoch 2: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 2/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0101 -
val_loss: 0.0095 - lr: 0.0010

Epoch 3: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 3/50
7500/7500 [==============================] - 30s 4ms/step - loss: 0.0091 -
val_loss: 0.0089 - lr: 0.0010

Epoch 4: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 4/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0086 -
val_loss: 0.0084 - lr: 0.0010

Epoch 5: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 5/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0083 -
```

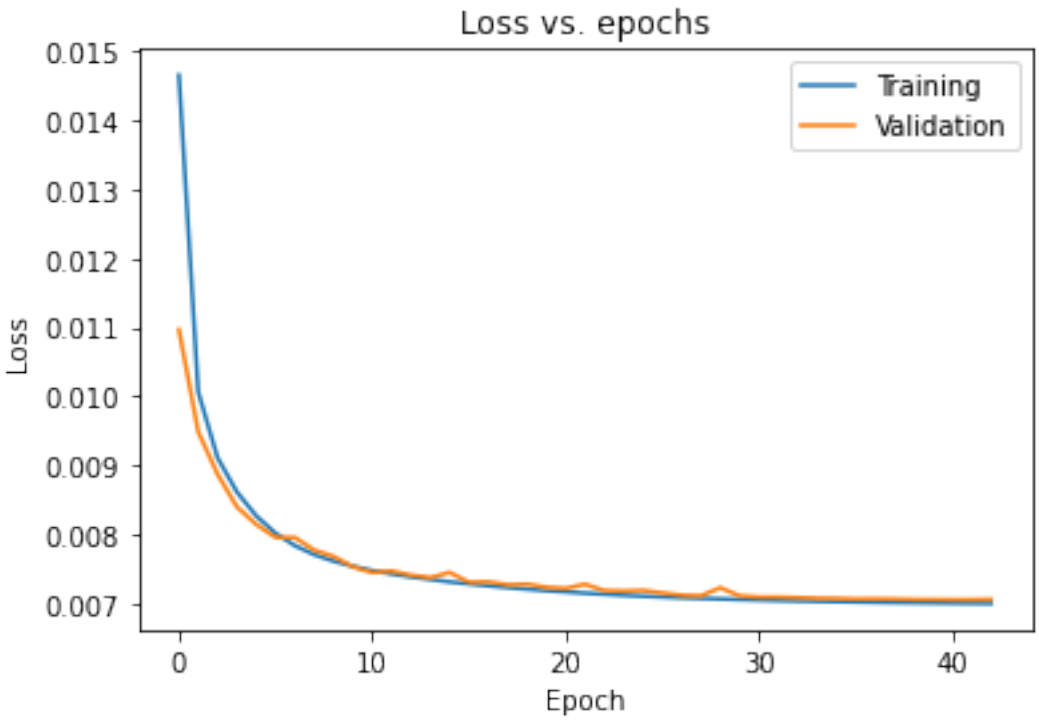
```
val_loss: 0.0081 - lr: 0.0010

Epoch 6: LearningRateScheduler setting learning rate to 0.0009048373904079199.
Epoch 6/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0080 -
val_loss: 0.0080 - lr: 9.0484e-04

Epoch 7: LearningRateScheduler setting learning rate to 0.0008187306812033057.
Epoch 7/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0078 -
val_loss: 0.0080 - lr: 8.1873e-04

Epoch 8: LearningRateScheduler setting learning rate to 0.0007404818097256124.
Epoch 8/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0077 -
val_loss: 0.0078 - lr: 7.4082e-04

Epoch 9: LearningRateScheduler setting learning rate to 0.0006703198887407778.
Epoch 9/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0076 -
val_loss: 0.0077 - lr: 6.7032e-04

Epoch 10: LearningRateScheduler setting learning rate to 0.0006065304623916745.
Epoch 10/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0075 -
val_loss: 0.0075 - lr: 6.0653e-04

Epoch 11: LearningRateScheduler setting learning rate to 0.0005488114547915757.
Epoch 11/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0075 -
val_loss: 0.0074 - lr: 5.4881e-04

Epoch 12: LearningRateScheduler setting learning rate to 0.0004965850966982543.
Epoch 12/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0074 -
val_loss: 0.0075 - lr: 4.9659e-04

Epoch 13: LearningRateScheduler setting learning rate to 0.0004493287415243685.
Epoch 13/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0074 -
val_loss: 0.0074 - lr: 4.4933e-04

Epoch 14: LearningRateScheduler setting learning rate to 0.0004065694229211658.
Epoch 14/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0073 -
val_loss: 0.0074 - lr: 4.0657e-04

Epoch 15: LearningRateScheduler setting learning rate to 0.00036787919816561043.
```

```
Epoch 15/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0073 -
val_loss: 0.0075 - lr: 3.6788e-04

Epoch 16: LearningRateScheduler setting learning rate to 0.0003328708407934755.
Epoch 16/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0073 -
val_loss: 0.0073 - lr: 3.3287e-04

Epoch 17: LearningRateScheduler setting learning rate to 0.00030119396978989244.
Epoch 17/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0073 -
val_loss: 0.0073 - lr: 3.0119e-04

Epoch 18: LearningRateScheduler setting learning rate to 0.0002725531557129696.
Epoch 18/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0072 -
val_loss: 0.0073 - lr: 2.7253e-04

Epoch 19: LearningRateScheduler setting learning rate to 0.0002465967263560742.
Epoch 19/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0072 -
val_loss: 0.0073 - lr: 2.4660e-04

Epoch 20: LearningRateScheduler setting learning rate to 0.0002231299295090139.
Epoch 20/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0072 -
val_loss: 0.0072 - lr: 2.2313e-04

Epoch 21: LearningRateScheduler setting learning rate to 0.0002018962986767292.
Epoch 21/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0072 -
val_loss: 0.0072 - lr: 2.0190e-04

Epoch 22: LearningRateScheduler setting learning rate to 0.00018268331768922508.
Epoch 22/50
7500/7500 [==============================] - 32s 4ms/step - loss: 0.0072 -
val_loss: 0.0073 - lr: 1.8268e-04

Epoch 23: LearningRateScheduler setting learning rate to 0.00016529869753867388.
Epoch 23/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0072 - lr: 1.6530e-04

Epoch 24: LearningRateScheduler setting learning rate to 0.00014956844097469002.
Epoch 24/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0072 - lr: 1.4957e-04
```

```
Epoch 25: LearningRateScheduler setting learning rate to 0.0001353351108264178.
Epoch 25/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0072 - lr: 1.3534e-04

Epoch 26: LearningRateScheduler setting learning rate to 0.00012245627294760197.
Epoch 26/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0072 - lr: 1.2246e-04

Epoch 27: LearningRateScheduler setting learning rate to 0.00011080301192123443.
Epoch 27/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0071 - lr: 1.1080e-04

Epoch 28: LearningRateScheduler setting learning rate to 0.00010025870869867504.
Epoch 28/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0071 - lr: 1.0026e-04

Epoch 29: LearningRateScheduler setting learning rate to 9.071782551473007e-05.
Epoch 29/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0072 - lr: 9.0718e-05

Epoch 30: LearningRateScheduler setting learning rate to 8.208487997762859e-05.
Epoch 30/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0071 - lr: 8.2085e-05

Epoch 31: LearningRateScheduler setting learning rate to 7.427347009070218e-05.
Epoch 31/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0071 -
val_loss: 0.0071 - lr: 7.4273e-05

Epoch 32: LearningRateScheduler setting learning rate to 6.720540841342881e-05.
Epoch 32/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 6.7205e-05

Epoch 33: LearningRateScheduler setting learning rate to 6.080996536184102e-05.
Epoch 33/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 6.0810e-05

Epoch 34: LearningRateScheduler setting learning rate to 5.502313069882803e-05.
Epoch 34/50
```

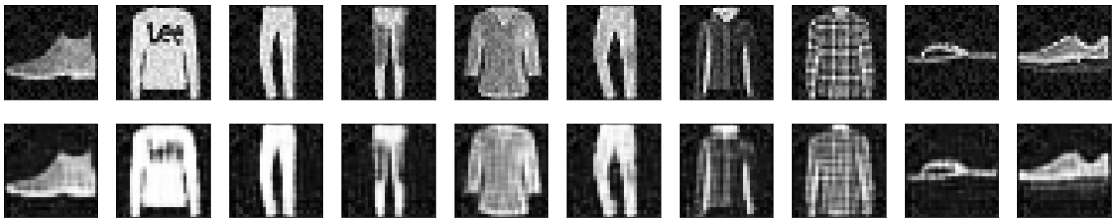
```
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 5.5023e-05

Epoch 35: LearningRateScheduler setting learning rate to 4.978698416380212e-05.
Epoch 35/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 4.9787e-05

Epoch 36: LearningRateScheduler setting learning rate to 4.5049124310025945e-05.
Epoch 36/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 4.5049e-05

Epoch 37: LearningRateScheduler setting learning rate to 4.076213008374907e-05.
Epoch 37/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 4.0762e-05

Epoch 38: LearningRateScheduler setting learning rate to 3.688309880089946e-05.
Epoch 38/50
7500/7500 [==============================] - 30s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 3.6883e-05

Epoch 39: LearningRateScheduler setting learning rate to 3.337320595164783e-05.
Epoch 39/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 3.3373e-05

Epoch 40: LearningRateScheduler setting learning rate to 3.019732321263291e-05.
Epoch 40/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 3.0197e-05

Epoch 41: LearningRateScheduler setting learning rate to 2.7323667382006533e-05.
Epoch 41/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 2.7324e-05

Epoch 42: LearningRateScheduler setting learning rate to 2.4723474780330434e-05.
Epoch 42/50
7500/7500 [==============================] - 31s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 2.4723e-05

Epoch 43: LearningRateScheduler setting learning rate to 2.237072476418689e-05.
Epoch 43/50
7500/7500 [==============================] - 32s 4ms/step - loss: 0.0070 -
val_loss: 0.0071 - lr: 2.2371e-05
```

[35]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss vs. epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='upper right')
plt.show()
```



[36]:
```
decoded_imgs = autoencoder.predict(test_image)

n = 10

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

    # display reconstruction
    ax = plt.subplot(2, n, i+1+n)
```

```
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```



[36]: