Practice Assignment

September 14, 2020

0.1 Task 1: Introduction

Welcome to this project on how to avoid overfitting with regularization. We will take a look at two types of regularization techniques: weight regularization and dropout regularization.

Overfitting

0.2 Task 2: Importing the Data

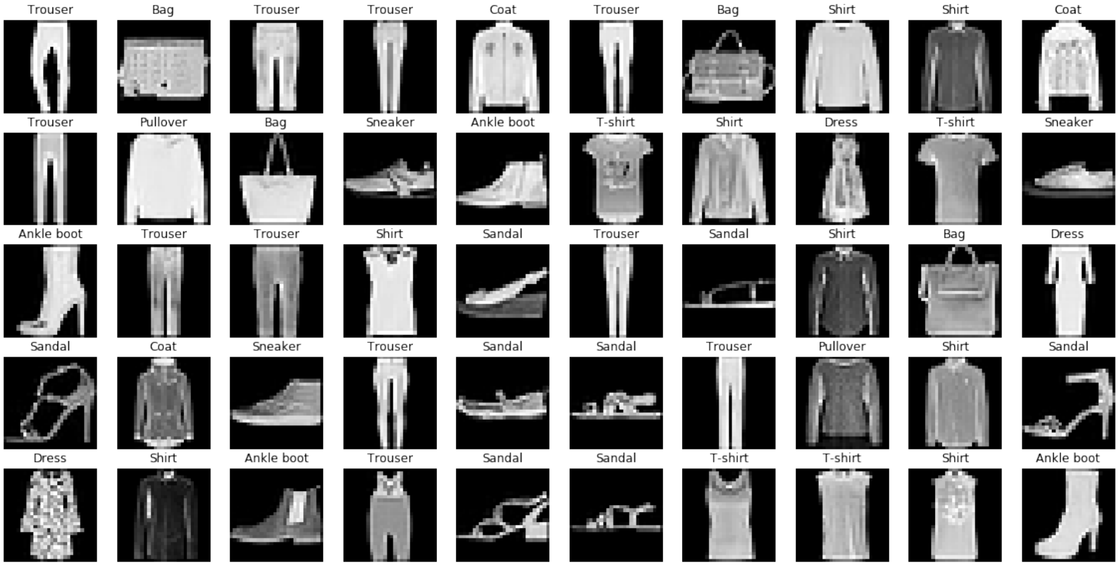
Note: If you are starting the notebook from this task, you can run cells from all previous tasks in the kernel by going to the top menu and then selecting Kernel > Restart and Run All ___

[1]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.regularizers import l2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *
from matplotlib import pyplot as plt
from tensorflow.keras.datasets import fashion_mnist
```

[2]:
```
(X_train, y_train), (X_val, y_val) = fashion_mnist.load_data()
print(f"There are {X_train.shape[0]} images for training and {X_val.shape[0]}␣
 ↪images for testing.")
```

```
There are 60000 images for training and 10000 images for testing.
```

[3]:
```
categories = ['T-shirt', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal',␣
 ↪'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
n_rows, n_cols = 5, 10

fig, axes = plt.subplots(n_rows, n_cols, figsize=(20, 10))
axes = axes.ravel()

for index in range(n_rows * n_cols):

    random_index = np.random.randint(0, len(X_train))
    axes[index].imshow(X_train[random_index], cmap='gray')
    axes[index].set_title(categories[y_train[random_index]])
    axes[index].axis('off')
```



0.3 Task 3: Processing the Data

Note: If you are starting the notebook from this task, you can run cells from all previous tasks in the kernel by going to the top menu and then selecting Kernel > Restart and Run All ___ Original Label: [5] is converted to -> One Hot Encoded Label: [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

[4]:
```
from tensorflow.keras.utils import to_categorical
print(f"Before: {y_train[random_index]}")
y_train = to_categorical(y_train, num_classes=10)
y_val = to_categorical(y_val, num_classes=10)
print(f"After: {y_train[random_index]}")
```

```
Before: 9
After: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

[5]:
```
X_train = X_train.reshape(-1, 28*28)/255.
X_val = X_val.reshape(-1, 28*28)/255.
print(f"Training set shape: {X_train.shape}")
print(f"Test set shape: {X_val.shape}")
```

```
Training set shape: (60000, 784)
Test set shape: (10000, 784)
```

0.4 Task 4: Regularization and Dropout

Note: If you are starting the notebook from this task, you can run cells from all previous tasks in the kernel by going to the top menu and then selecting Kernel > Restart and Run All ___

Neural Network

Dropouts

Dropouts:

0.5 Task 5: Creating the Experiment Part 1

Note: If you are starting the notebook from this task, you can run cells from all previous tasks in the kernel by going to the top menu and then selecting Kernel > Restart and Run All ___

[6]:
```
def create_model(weight_reg = False, dropout_reg = False):

    model = Sequential()

    if weight_reg:
        model.add(Dense(100, activation='relu', kernel_regularizer=l2(l=0.001),␣
 ↪input_shape=(784,)))
        model.add(Dense(200, activation='relu', kernel_regularizer=l2(l=0.001)))
    else:
        model.add(Dense(100, activation='relu', input_shape=(784,)))
        model.add(Dense(200, activation='relu'))
    if dropout_reg:
        model.add(Dropout(rate=0.2))
    model.add(Dense(10, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
    model.summary()

    return model
```

0.6 Task 6: Creating the Experiment Part 2

Note: If you are starting the notebook from this task, you can run cells from all previous tasks in the kernel by going to the top menu and then selecting Kernel > Restart and Run All ___

[7]:
```
def show_plots(results, epochs):

    plt.plot(range(epochs), results.history['accuracy'], label='Training␣
 ↪accuracy')
    plt.plot(range(epochs), results.history['val_accuracy'], label='Testing␣
 ↪accuracy')
    plt.legend()
    plt.show()
```

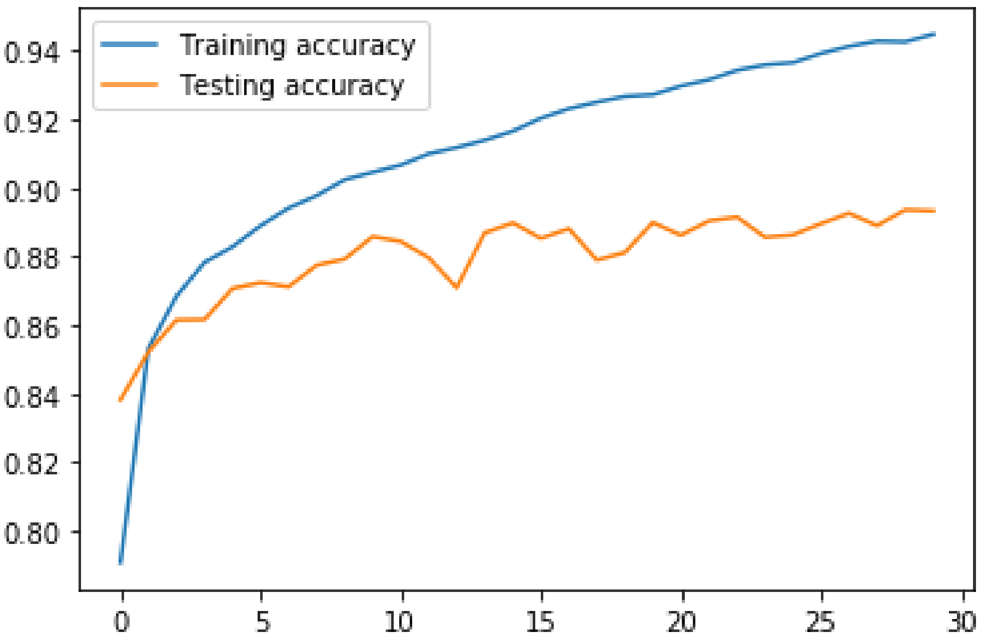
[8]:
```
from tensorflow.keras.callbacks import LambdaCallback

simple_log = LambdaCallback(on_epoch_end = lambda e, l: print(f"Epoch: {e+1}",␣
 ↪end='. '))

def run_experiment(epochs=30, weight_reg = False, dropout_reg=False):

    model = create_model(weight_reg, dropout_reg)
    results = model.fit(X_train, y_train, validation_data=(X_val, y_val),␣
 ↪batch_size=256,
                        epochs=epochs, callbacks=[simple_log], verbose=False)
    show_plots(results, epochs)
```

0.7 Task 7: Results

Note: If you are starting the notebook from this task, you can run cells from all previous tasks in the kernel by going to the top menu and then selecting Kernel > Restart and Run All ___

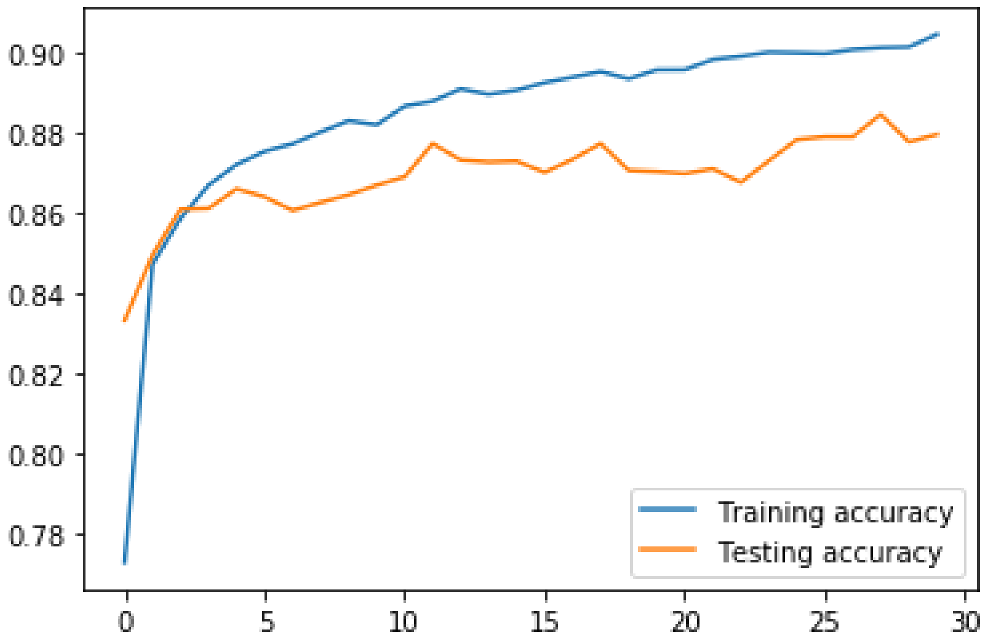
[9]:
```
run_experiment()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               78500
_________________________________________________________________
dense_1 (Dense)              (None, 200)               20200
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2010
=================================================================
Total params: 100,710
Trainable params: 100,710
Non-trainable params: 0
_________________________________________________________________
Epoch: 1. Epoch: 2. Epoch: 3. Epoch: 4. Epoch: 5. Epoch: 6. Epoch: 7. Epoch: 8.
Epoch: 9. Epoch: 10. Epoch: 11. Epoch: 12. Epoch: 13. Epoch: 14. Epoch: 15.
Epoch: 16. Epoch: 17. Epoch: 18. Epoch: 19. Epoch: 20. Epoch: 21. Epoch: 22.
Epoch: 23. Epoch: 24. Epoch: 25. Epoch: 26. Epoch: 27. Epoch: 28. Epoch: 29.
```



[10]:
```
run_experiment(weight_reg=True, dropout_reg=True)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 100)               78500
_________________________________________________________________
dense_4 (Dense)              (None, 200)               20200
_________________________________________________________________
dropout (Dropout)            (None, 200)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                2010
=================================================================
Total params: 100,710
Trainable params: 100,710
Non-trainable params: 0
_________________________________________________________________
Epoch: 1. Epoch: 2. Epoch: 3. Epoch: 4. Epoch: 5. Epoch: 6. Epoch: 7. Epoch: 8.
Epoch: 9. Epoch: 10. Epoch: 11. Epoch: 12. Epoch: 13. Epoch: 14. Epoch: 15.
Epoch: 16. Epoch: 17. Epoch: 18. Epoch: 19. Epoch: 20. Epoch: 21. Epoch: 22.
Epoch: 23. Epoch: 24. Epoch: 25. Epoch: 26. Epoch: 27. Epoch: 28. Epoch: 29.
```



That’s it for this project! Thank you for following along!