Assignment 4

Submitted By: Abhishek Ayachit (862188073)
Deep Learning: Assignment 4
Date: 06/06/2020

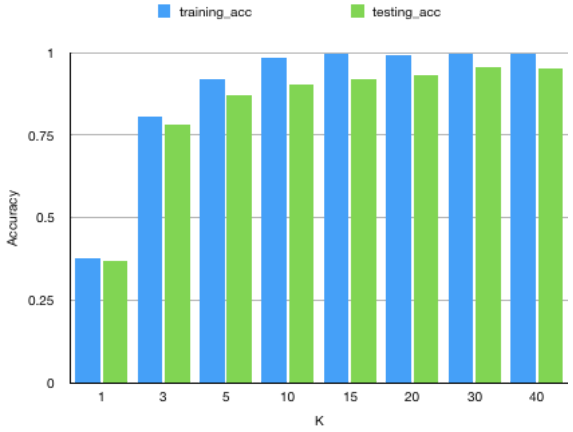
Q1. Setup your code so that you can run multiple MNIST models for varying choices of k and p automatically, Specifically, you need two for loops (one for k and one for p) and within the loop, you call TensorFlow/PyTorch.

```
K = [1, 3, 5, 10, 15, 20, 30, 40]
P = [0.1, 0.5, 1.0]
for p in P:
  for k in K:
    NN(x_train, y_train, x_test, y_test, epochs=epochs, batch_size=batch_size, "Q2", k=k, p=p)
```

```
train_acc = []
test_acc = []
test_loss = []
def NN(x_train, y_train, x_test, y_test, epochs=80, batch_size=10, filename, k, p):

    model = Sequential()

    model.add(Dense(k, input_shape=(784,), kernel_initializer='he_normal'))
    model.add(Activation('relu'))
    model.add(Dropout(p))

    model.add(Dense(k))
    model.add(Activation('relu'))
    model.add(Dropout(p))

    model.add(Dense(10))
    model.add(Activation('softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adam(),
                  metrics=['accuracy'])

    csv_logger = CSVLogger(filename+".log", append=True)
    plot = model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              verbose=1,
              validation_data=(x_test, y_test),
              callbacks=[csv_logger])



    train_acc.append((k, plot.history['accuracy']))
    test_acc.append((k, plot.history['val_accuracy']))
    test_loss.append((k, plot.history['loss']))
```

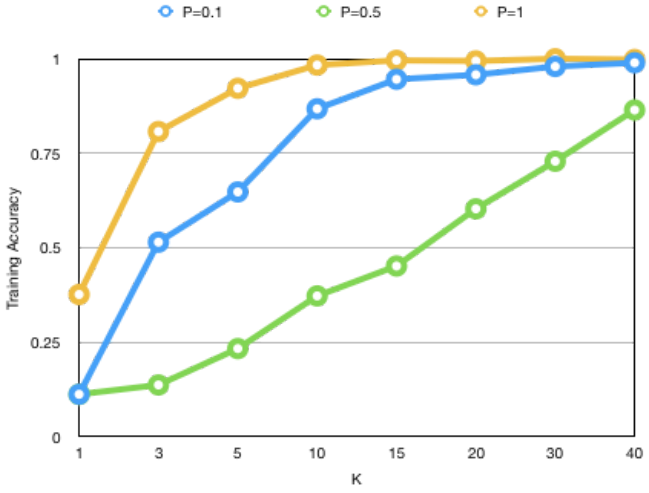
Q2. K = [1, 3, 5, 10, 15, 20, 30, 40] & P = [0.1, 0.5, 1.0]. Run MNIST models over these grids with Adam optimizer for 80 epochs.

- **Fix p = 1.0 which is the case of "no dropout regularization". Plot the test and training accuracy as a function of k. As k increases, does the performance improve? At what k, training accuracy becomes 100%?**



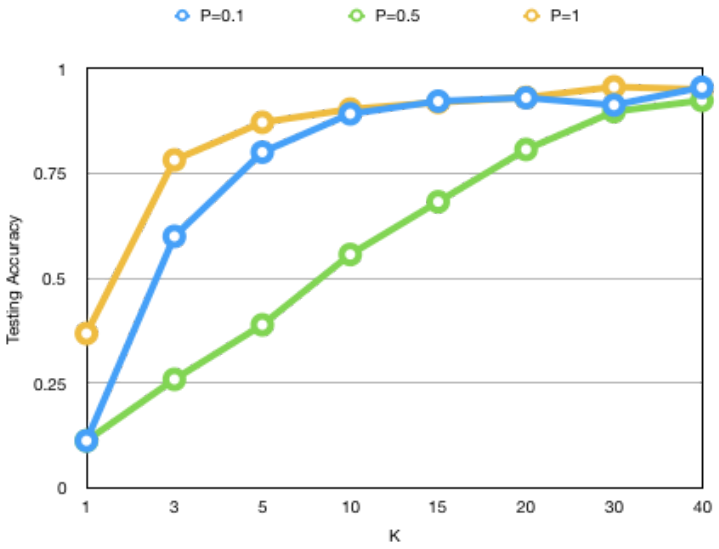
According to the above plot, it is clear that as the value of 'k' increases, the performance also increases with it.
At k=15, the training accuracy becomes 100%.

- **Plot the training accuracy as a function of k and for different p ∈ P on the same plot. What is the role of p on training accuracy? When p is smaller, is it easier to optimize or more difficult? For each choice of p, determine at what choice of k, training accuracy becomes 100%**



Role of P: it helps the model in preventing overfitting. As the value of P increases, the training accuracy seems to be decreasing. When there is no dropout, the accuracy is highest.
For P = 0.1, at k=40, the training accuracy becomes 100%
For P = 0.5, the training accuracy doesn't become 100% but it may become 100% at k slightly greater than 40.
For P = 1, the training accuracy becomes 100% at k=15.

- **Plot the test accuracy as a function of k and for different p ∈ P on the same plot. Does dropout help with the test accuracy? For which (k,p) configuration do you achieve the best test accuracy?**

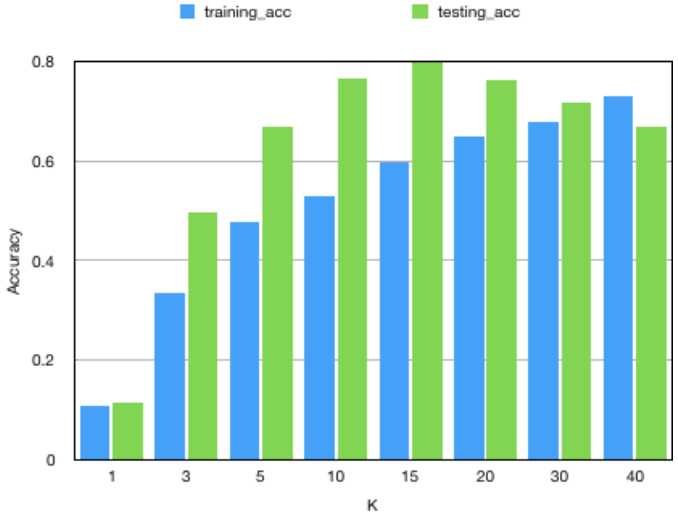


At P=0.1, the accuracy is much better than that at p=0.5. But the accuracy seems to be increasing as we increase the value of k. Therefore when there is more dropout, the accuracy starts to increase a little slower but with the higher value of k, it is similar to others.

For k=30 and P=1, the accuracy is highest which is closely followed by k=40 and p = 0.1

Q3. K = [1, 3, 5, 10, 15, 20, 30, 40] & P = [0.1, 0.5, 1.0]. Run MNIST models over these grids with Adam optimizer for 80 epochs. With Noisy Dataset

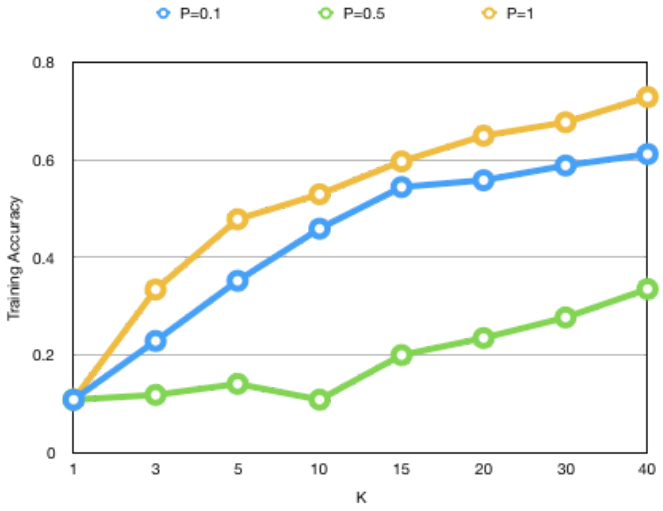
- **Fix p = 1.0 which is the case of "no dropout regularization". Plot the test and training accuracy as a function of k. As k increases, does the performance improve? At what k, training accuracy becomes 100%?**



Similar to the dataset without noise, the training accuracy increases with the value of 'k'. The testing accuracy on the other hand goes up and then comes down.

In the given number of epochs the training accuracy doesn't reach 100% due to the noisy dataset.

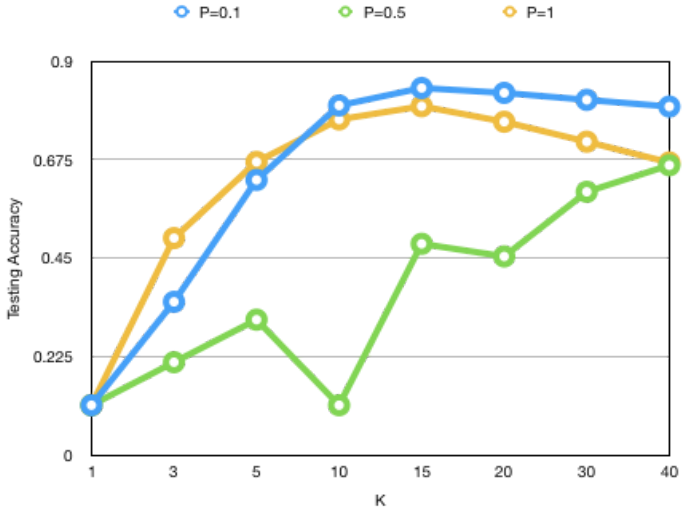
- **Plot the training accuracy as a function of k and for different p ∈ P on the same plot. What is the role of p on training accuracy? When p is smaller, is it easier to optimize or more difficult? For each choice of p, determine at what choice of k, training accuracy becomes 100%**



In the case of noisy dataset, when there is no dropout, the training accuracy is highest, while with the value of p = 0.5, the accuracy is least.

In this experiment, with epoch=80, in all the cases of P, the value doesn't reach 100% for any value of k. The value might reach 100% with k>40.

- **Plot the test accuracy as a function of k and for different p ∈ P on the same plot. Does dropout help with the test accuracy? For which (k,p) configuration do you achieve the best test accuracy?**



In this configuration, when there is no dropout, the testing accuracy first increases and then decreases. For p = 0.1, the accuracy is highest. Therefore, with more dropout rate, the accuracy seems to be low. With greater value of k, accuracy is quite similar.

For k=15 and p=0.1, the test accuracy is best.

Q4. Comment on the differences between Step 2 and Step 3. How does noise changes things? For which setup dropout is more useful?

When noise is added to the dataset, both the accuracies seem to be decreasing. In the case of P=0.1, the testing accuracy is higher than other values of p which is not the case for dataset without noise.

From the graph for testing accuracy of Q3, it seems that the dropout is more useful when there is high amount of dropout rate(P=0.1) and the noise is added to the data but also in the case of no noisy data, the high dropout rate(P=0.1) seem to be getting higher accuracy with higher value of k.