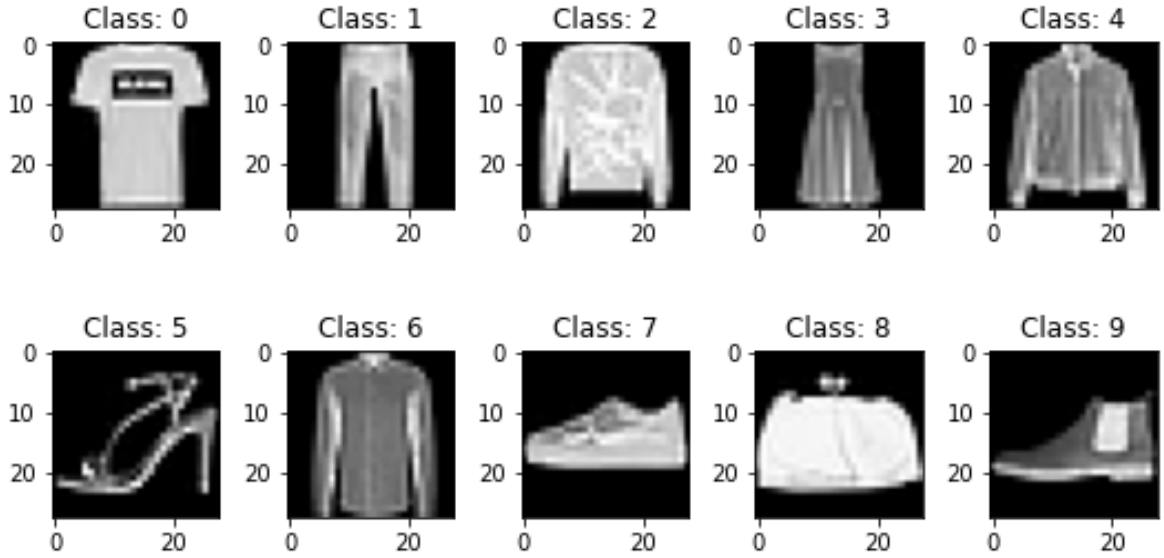
COEN 166 Artificial Intelligence
Lab 6: Neural Network
Name: Tristan Limawan and Collin Paiz **ID:** 1575563 and 1576109

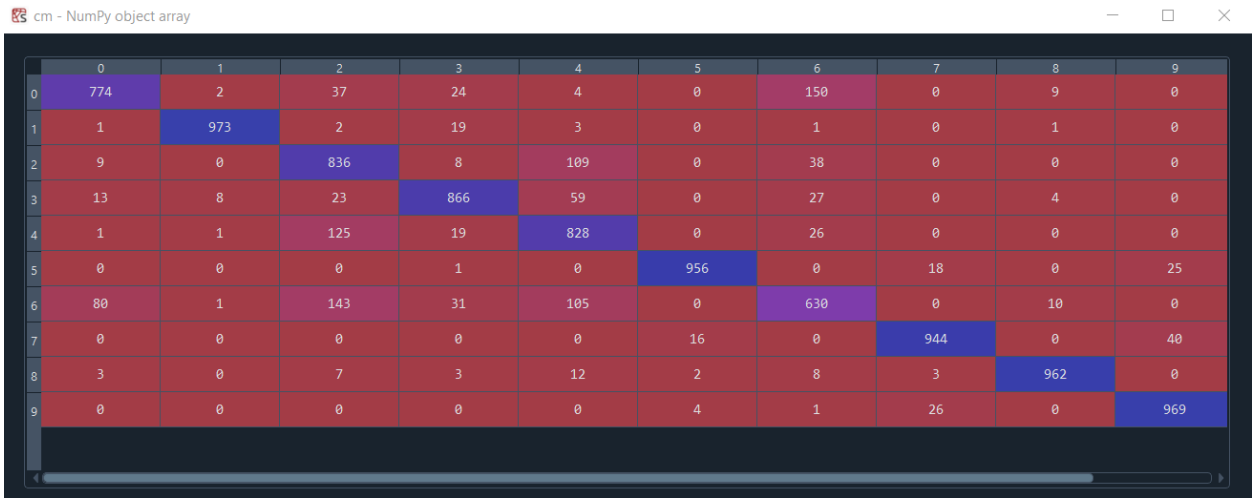
Problem 1: Fashion mnist image recognition

1. Display images from each class



2. Recognition accuracy rate for the test set, and the confusion matrix

```
Epoch 1/5
1875/1875 [==============================] -
3s 2ms/step - loss: 0.4736 - accuracy: 0.8317
Epoch 2/5
1875/1875 [==============================] -
3s 2ms/step - loss: 0.3570 - accuracy: 0.8695
Epoch 3/5
1875/1875 [==============================] -
3s 2ms/step - loss: 0.3235 - accuracy: 0.8797
Epoch 4/5
1875/1875 [==============================] -
4s 2ms/step - loss: 0.2979 - accuracy: 0.8887
Epoch 5/5
1875/1875 [==============================] -
4s 2ms/step - loss: 0.2813 - accuracy: 0.8956
313/313 [==============================] - 0s
1ms/step - loss: 0.3736 - accuracy: 0.8703
Loss Rate =  0.3736173212528229
Accuracy Rate =  0.8702999949455261
```



	0	1	2	3	4	5	6	7	8	9
0	774	2	37	24	4	0	150	0	9	0
1	1	973	2	19	3	0	1	0	1	0
2	9	0	836	8	109	0	38	0	0	0
3	13	8	23	866	59	0	27	0	4	0
4	1	1	125	19	828	0	26	0	0	0
5	0	0	0	1	0	956	0	18	0	25
6	80	1	143	31	105	0	630	0	10	0
7	0	0	0	0	0	16	0	944	0	40
8	3	0	7	3	12	2	8	3	962	0
9	0	0	0	0	0	4	1	26	0	969

Problem 2: Fashion mnist image compression

1. Results

P = 10

```
Loss Rate =  0.013158918358385563
Accuracy Rate =  0.18832500278949738
Average PSNR =  tf.Tensor(18.807796, shape=(), dtype=float32)
```

P = 50

```
Loss Rate =  0.006935453042387962
Accuracy Rate =  0.27407142519950867
Average PSNR =  tf.Tensor(21.58925, shape=(), dtype=float32)
```

P = 200

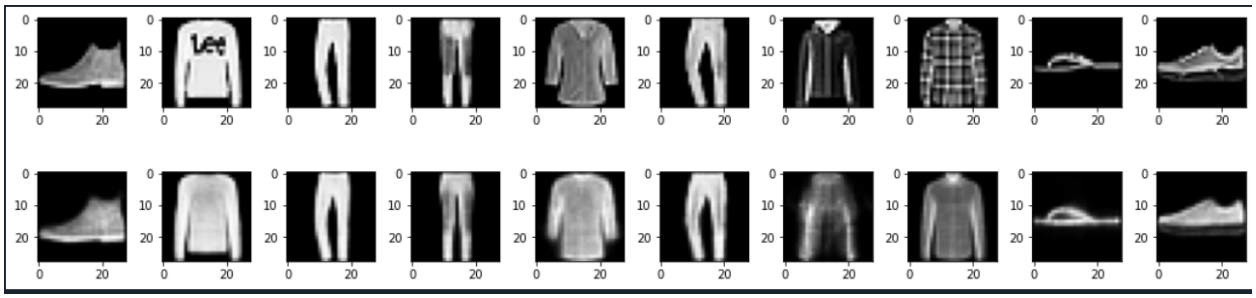
```
Loss Rate =  0.00355037534609437
Accuracy Rate =  0.37128928303718567
Average PSNR =  tf.Tensor(24.497257, shape=(), dtype=float32)
```

What's the difference among the average PSNR of different *P* values? What do you think is the reason of such a result?

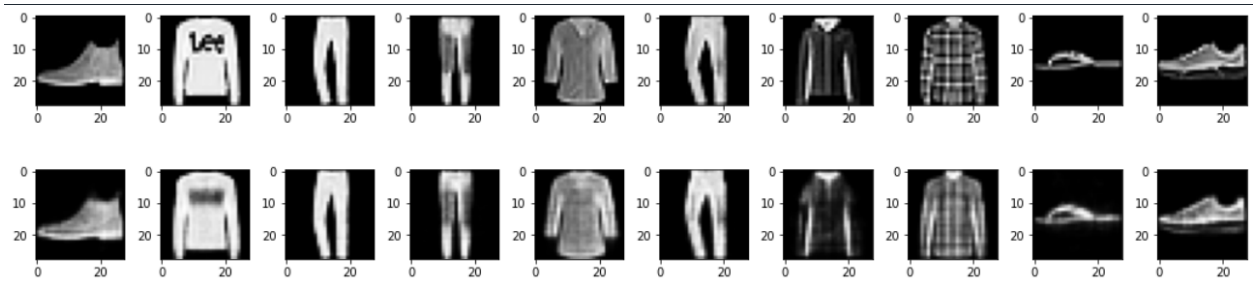
The average PSNR increases with larger P values. The reason why is because PSNR is a ratio that measures the quality between the original and the decompressed image. As shown in the next images, the quality of the decompressed images improves as P increases.

2. Display the figure

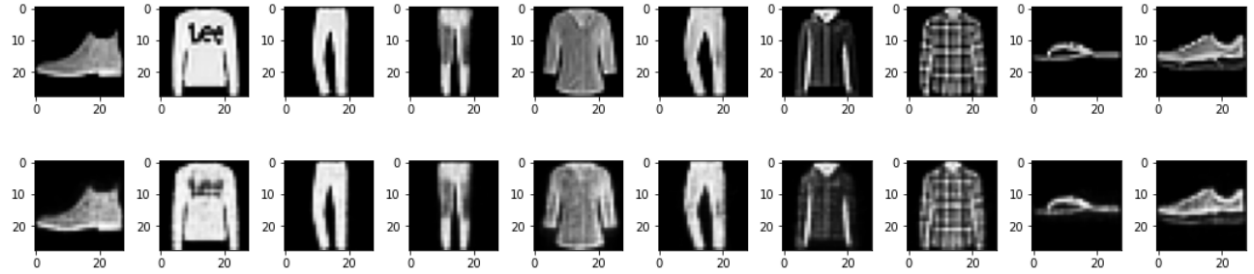
P=10



P = 50



P = 200



What do you observe from the decompressed images (the visual quality of the decompressed images of different *P* values)?

The quality of the decompressed images improves with higher P values. The decompressed images were the most clear when P = 200, and were the least clear when P = 10.

Appendix:
Problem 1

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from sklearn.metrics import confusion_matrix


fashion_mnist = keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0 #normalize the pixel values to be in [0, 1]

# Define row count and col count for image
num_row = 2
num_col = 5
num = num_row*num_col
images = x_train
classes = y_train
i = 0
j = 0

# Create figure with one image from each class
fig, axes = plt.subplots(num_row, num_col, figsize=(1.5*num_col,2*num_row))
while j < 10:
    if(j == classes[i]):
        ax = axes[j//num_col, j%num_col]
        ax.imshow(images[i], cmap='gray')
        ax.set_title('Class: {}'.format(classes[i]))
        j = j + 1
    i = i + 1
plt.tight_layout()
plt.show()

# Create model
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()

# Compile model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
model.fit(x_train, y_train, epochs=5, batch_size=32)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("Loss Rate = ", test_loss)
print("Accuracy Rate = ", test_accuracy)

# Calculate predictions
probability = model.predict(x_test)
```

```
y_test_hat = np.argmax(probability, axis=1)
cm = confusion_matrix(y_test, y_test_hat, labels=range(10))
```

Problem 2

```
1  import tensorflow as tf
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from tensorflow import keras
5  from keras.models import Sequential
6  from keras.layers import Dense
7  from keras.layers import Flatten
8  from keras.layers import Reshape
9  from sklearn.metrics import confusion_matrix
10
11 fashion_mnist = keras.datasets.fashion_mnist
12 (x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
13 x_train, x_test = x_train / 255.0, x_test / 255.0 #normalize the pixel values to be in [0, 1]
14
15 # Create model
16 model = Sequential()
17 model.add(Flatten(input_shape=(28,28))) # Input layer is flattened image
18 model.add(Dense(200, activation='relu')) # Compressed hidden layer with P nodes + ReLU
19 model.add(Dense(1568, activation='relu')) # Expansion hidden layer with 28 x 28 x T, T = 2 + ReLU
20 model.add(Dense(784, activation='sigmoid')) # Output layer 28 x 28 + sigmoid
21 model.add(Reshape(target_shape=(28, 28))) # Reshape layer to 2 dimensional image
22
23 model.summary()
24
25 # Compile model
26 model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
27
28 # Fit the model
29 model.fit(x_train, x_train, epochs=10, batch_size=64)
30
31 x_predict = model.predict(x_test)
32
33 psnr = tf.image.psnr(x_predict, x_test, max_val=1)
34
35 # Evaluate the model
36 test_loss, test_accuracy = model.evaluate(x_test, x_test)
37 print("Loss Rate = ", test_loss)
38 print("Accuracy Rate = ", test_accuracy)
39 print("Average PSNR = ", psnr)
40
41 # Define row count and col count for image
42 num_row = 2
43 num_col = 10
44 num = num_row*num_col
45 images = x_test
46 new_images = x_predict
47
48 # Create figure with one image from each class
49 fig, axes = plt.subplots(num_row, num_col, figsize=(1.5*num_col,2*num_row))
50
51 for i in range(10):
52     ax = axes[0, i%num_col]
53     ax.imshow(images[i], cmap='gray')
54
55 for i in range(10):
56     ax = axes[1, i%num_col]
57     ax.imshow(new_images[i], cmap='gray')
58
59
60 plt.tight_layout()
61 plt.show()
62
```

On line 18, we changed the P value from 10 to 50 to 200 and reran the program 3 separate times.