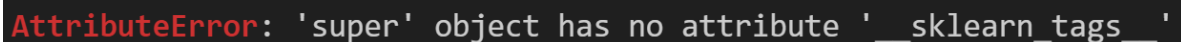


HW3 Report Deep Learning

For this assignment I chose to use the Fashion-MNIST dataset. A CNN was implemented and trained on this dataset. The model includes two convolutional layers, max-pooling layers, dropout and FC layers. The model was trained using the Adam optimizer with different learning rates, dropout rates and training epochs. Instead of automated grid search, the hyperparameter tuning was performed manually by iterating over combinations of learning rates, epochs, dropout rates and optimizers.

I had a bug while trying to tune hyperparameters using the GridSearchCV. So I had to try and do that manually. Adding a screenshot of the error:



```
AttributeError: 'super' object has no attribute '__sklearn_tags__'
```

This error also occurred to me when trying to run the example notebook that you added to the Moodle.

Evaluation

The model performed well on the test set with an accuracy of **92%**. Most classes, like **Sandal**, **Trouser**, and **Bag**, had excellent precision, recall, and F1-scores close to **0.99**, showing the model's strong performance.

However, the **Shirt** class struggled, with an F1-score of **0.75**, indicating some confusion with other classes. Overall, the model is reliable and consistent but improving the performance for the **Shirt** class could make it even better.

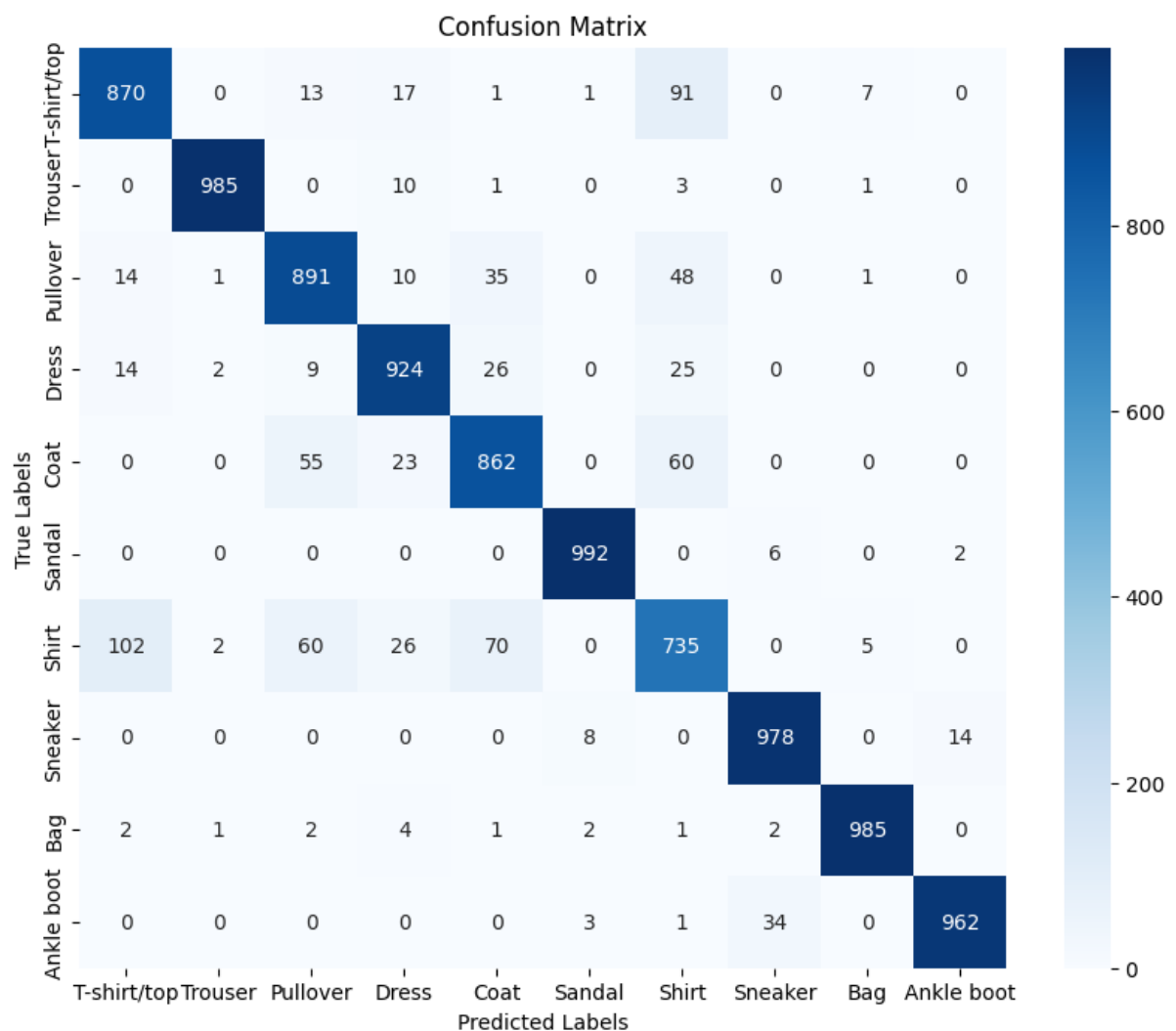
```
Accuracy on the test set: 91.84%

Classification Report:
              precision    recall  f1-score   support

T-shirt/top      0.87      0.87      0.87     1000
  Trouser        0.99      0.98      0.99     1000
  Pullover       0.87      0.89      0.88     1000
    Dress       0.91      0.92      0.92     1000
    Coat        0.87      0.86      0.86     1000
  Sandal         0.99      0.99      0.99     1000
    Shirt        0.76      0.73      0.75     1000
  Sneaker        0.96      0.98      0.97     1000
    Bag          0.99      0.98      0.99     1000
  Ankle boot     0.98      0.96      0.97     1000

 accuracy              0.92      10000
 macro avg           0.92      0.92      0.92     10000
weighted avg           0.92      0.92      0.92     10000
```

Confusion matrix



Looking at the confusion matrix, it's clear that the model performs quite well for most classes, especially for trousers and sandals, which have very few misclassifications. These items likely stand out because of their distinct shapes and features. However, there are some noticeable challenges. For example, shirts are often confused with T-shirts and coats, which makes sense because they share similar features like general shape and texture. I also noticed some misclassifications between sneakers and ankle boots, possibly due to overlapping visual traits. Overall, the model does a good job, but these areas of confusion suggest that it might need more diverse data or additional training to better distinguish similar items.

Hyperparameter Tuning

After 40 minutes of running, the hyperparameter tuning showed that the best configuration was a learning rate of 0.001, 10 epochs, a dropout of 0.5, and the Adam optimizer, achieving an accuracy of 70.26%. Adam consistently outperformed SGD across all settings. Lower learning rates provided more stable and accurate results, while higher dropout rates like 0.5 helped prevent overfitting. This process highlights how careful tuning can significantly improve model performance.

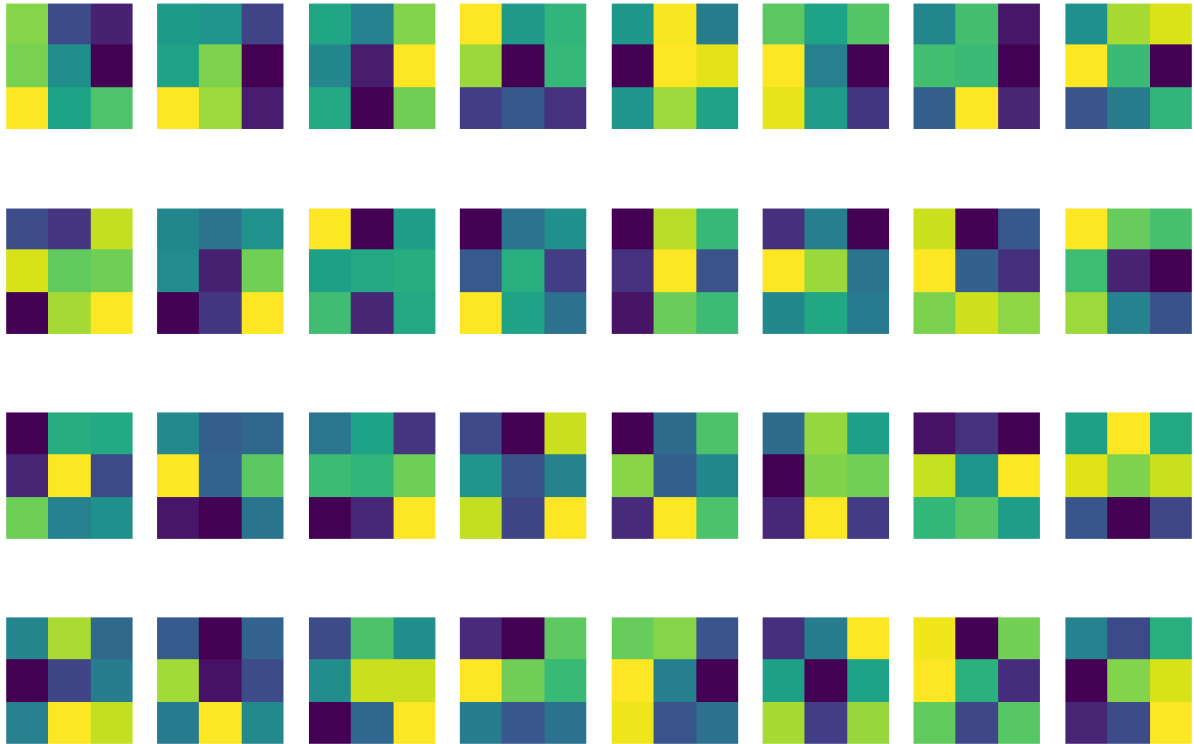
```
Testing configuration: lr=0.01, epochs=5, dropout=0.3, optimizer=Adam
Testing configuration: lr=0.01, epochs=5, dropout=0.3, optimizer=SGD
Testing configuration: lr=0.01, epochs=5, dropout=0.5, optimizer=Adam
Testing configuration: lr=0.01, epochs=5, dropout=0.5, optimizer=SGD
Testing configuration: lr=0.01, epochs=10, dropout=0.3, optimizer=Adam
Testing configuration: lr=0.01, epochs=10, dropout=0.3, optimizer=SGD
Testing configuration: lr=0.01, epochs=10, dropout=0.5, optimizer=Adam
Testing configuration: lr=0.01, epochs=10, dropout=0.5, optimizer=SGD
Testing configuration: lr=0.001, epochs=5, dropout=0.3, optimizer=Adam
Testing configuration: lr=0.001, epochs=5, dropout=0.3, optimizer=SGD
Testing configuration: lr=0.001, epochs=5, dropout=0.5, optimizer=Adam
Testing configuration: lr=0.001, epochs=5, dropout=0.5, optimizer=SGD
Testing configuration: lr=0.001, epochs=10, dropout=0.3, optimizer=Adam
Testing configuration: lr=0.001, epochs=10, dropout=0.3, optimizer=SGD
Testing configuration: lr=0.001, epochs=10, dropout=0.5, optimizer=Adam
Testing configuration: lr=0.001, epochs=10, dropout=0.5, optimizer=SGD
```

	lr	epochs	dropout	optimizer	accuracy
0	0.010	5	0.3	Adam	0.5124
1	0.010	5	0.3	SGD	0.1309
2	0.010	5	0.5	Adam	0.1980
3	0.010	5	0.5	SGD	0.1300
4	0.010	10	0.3	Adam	0.6249
5	0.010	10	0.3	SGD	0.2750
6	0.010	10	0.5	Adam	0.6194
7	0.010	10	0.5	SGD	0.0995
8	0.001	5	0.3	Adam	0.6520
9	0.001	5	0.3	SGD	0.1000
10	0.001	5	0.5	Adam	0.6182

11	0.001	5	0.5	SGD	0.1000
12	0.001	10	0.3	Adam	0.6962
13	0.001	10	0.3	SGD	0.0810
14	0.001	10	0.5	Adam	0.7026
15	0.001	10	0.5	SGD	0.0640

Interpreting the visualizations of filters

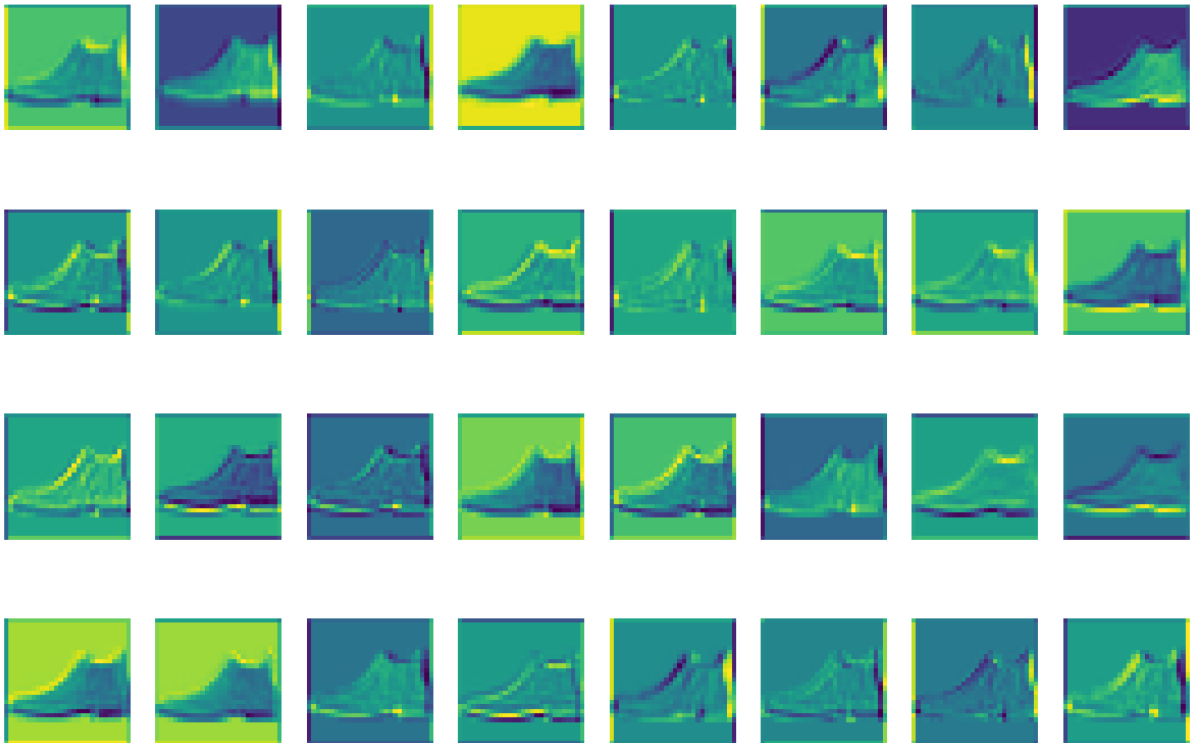
Filters in the First Convolutional Layer



In the first convolutional layer of the CNN, the filters learn to detect fundamental patterns like edges, gradients, and simple shapes from the input images. The color coding in the filters indicates the strength of the weights, with bright colors representing strong positive weights and dark colors indicating negative weights. The variety of filters demonstrates that the model is capturing different aspects of the images, such as vertical or horizontal edges, diagonal gradients, and texture patterns. For the Fashion-MNIST dataset, these filters are likely identifying key features of clothing items, such as edges, contours, and textures.

Interpreting the visualizations of feature maps

Feature Maps After First Convolutional Layer



The feature maps show what the filters detected in the image. Each map highlights specific parts of the image that a filter found important. For example, some feature maps focus on the outline of the shoe, while others pick up smaller details like texture. Together, these maps break the image into important pieces, helping the model understand it step by step.