Project 2

Deep Learning Project 2: A Comparison of Architectures

Feature Extraction and Classification Performance

Fully Connected ANNs vs CNN using MNIST and fashionMNIST [composed with Keras]

Code Part:

https://drive.google.com/drive/folders/1xakJXFcenA4YaFunJSlzvDoUqaYwljeL?usp=sharing

Video Part:

https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=9b47d83c-0bd5-4b51-9e6a-b21600444e33

Discussion:

1. While the MNIST dataset is perhaps the most frequently utilized dataset in ML courses, FashionMNIST is considered much more challenging. Explain why the samples in the dataset you used in this exercise seem harder to classify than the numerical identification tasks.

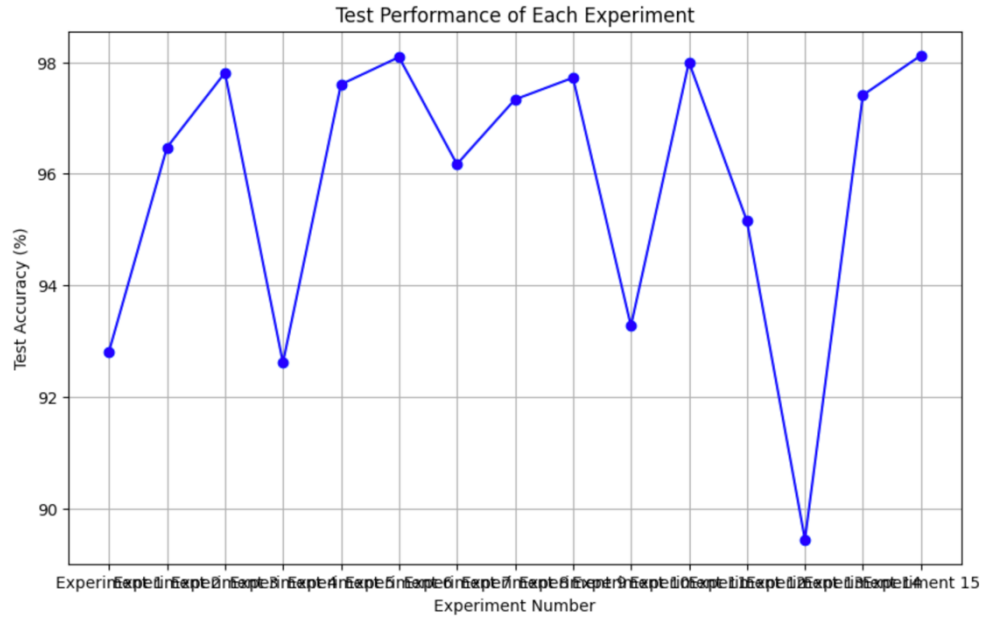
The images in FashionMNIST contain a variety of different types of clothing, such as shoes, tops, bags, etc., and these categories have many subtle differences in visual characteristics. The MNIST numbers (0-9) are quite different in shape and are relatively easier to distinguish.
The differences between some categories in the FashionMNIST dataset are very small, such as shirts and T-shirts, sandals and sneakers, etc. These similar categories may make it difficult for the model to capture salient features to distinguish them during the learning process.
FashionMNIST images often lack clear outlines and boundaries, and the shapes and edges of many clothing items are not very obvious in the images.
Clothing images in FashionMNIST may contain complex textures and details that are difficult to effectively capture with simple models.

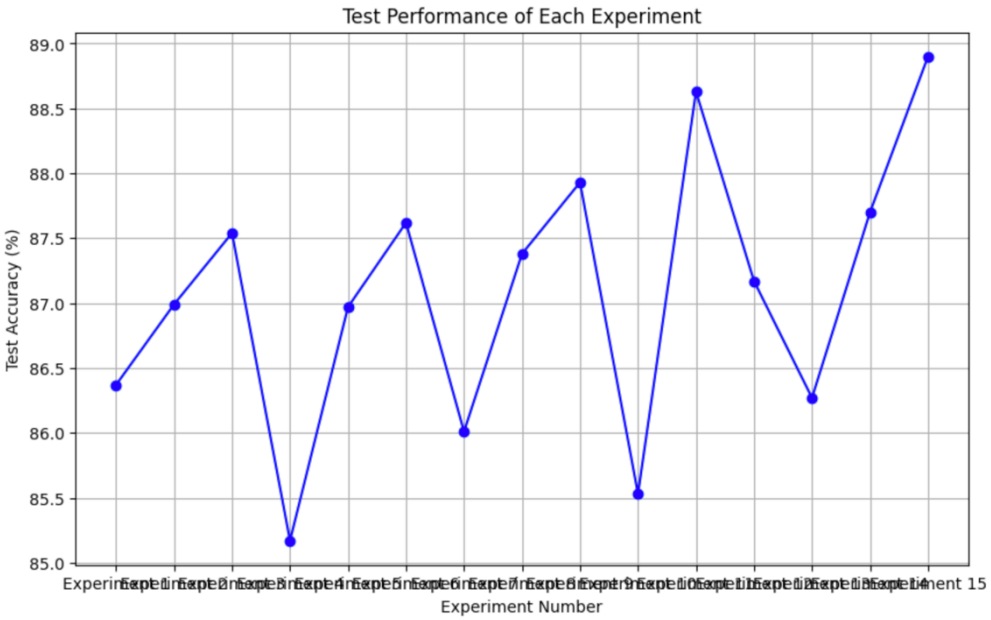
2. Utilizing the template attached, provide the calculations for map dimensions, number of weights and number of bias terms for your top performing CNN model.

Layer	Activation Map Dimensions	Number of Weights	Number of Biases
Input	28x28x1	0	0
CONV2D (dims)	26x26x64	576	64

POOL-2	13x13x64	0	0
CONV2D (dims)	11x11x128	73728	128
POOL-3	5x5x128	0	0
FC-10	10	32000	10

3. Compare the results of your experiments for Part 1, Part 2 and Part 3 – use the values from your recorded model performance to generate at least (3) meaningful figures related to your results. a. Display the results of the test performance for each experiment in a single graph (preferred). b. Provide a table or plot showing how complexity of the model contributed to challenges when training both the FC and CNN implementation. (Did you overfit or stop learning?)





Experiment	Layers	Hidden Layer Sizes	Training Data Division	Batch Size	Learning Rate	Optimizer	Accuracy
Experiment 1	2	32, 16	60,000 Training / 10,000 Testing	64	0.002	SGD	92.81
Experiment 2	2	64, 32	60,000 Training / 10,000 Testing	64	0.001	Adam	96.47
Experiment 3	2	128, 64	60,000 Training / 10,000 Testing	64	0.0005	RMSprop	97.81
Experiment 4	3	64, 32, 16	60,000 Training / 10,000 Testing	64	0.001	SGD	92.61
Experiment 5	3	128, 64, 32	60,000 Training / 10,000 Testing	64	0.0002	Adam	97.6
Experiment 6	2	256, 128	60,000 Training / 10,000 Testing	64	0.0001	RMSprop	98.09
Experiment 7	2	32, 32	60,000 Training / 10,000 Testing	64	0.01	SGD	96.17
Experiment 8	2	64, 64	60,000 Training / 10,000 Testing	64	0.001	Adam	97.33
Experiment 9	2	128, 128	60,000 Training / 10,000 Testing	64	0.0005	RMSprop	97.72
Experiment 10	3	256, 128, 64	60,000 Training / 10,000 Testing	64	0.001	SGD	93.27
Experiment 11	2	512, 256	60,000 Training / 10,000 Testing	64	0.0001	Adam	98
Experiment 12	3	32, 16, 8	60,000 Training / 10,000 Testing	64	0.002	RMSprop	95.15
Experiment 13	3	64, 32, 16	60,000 Training / 10,000 Testing	64	0.0004	SGD	89.44
Experiment 14	3	128, 64, 32	60,000 Training / 10,000 Testing	64	0.001	Adam	97.41
Experiment 15	3	256, 128, 64	60,000 Training / 10,000 Testing	64	0.0002	RMSprop	98.12

Experiment	Layers	Hidden Layer Sizes	Batch Size	Learning Rate	Optimizer	Accuracy	Training Data Division
Experiment 1	2	64, 32	64	0.003	SGD	86.37	60,000 Training / 10,000 Testing
Experiment 2	2	64, 32	64	0.001	Adam	86.99	60,000 Training / 10,000 Testing
Experiment 3	2	128, 64	64	0.0005	RMSprop	87.54	60,000 Training / 10,000 Testing
Experiment 4	3	64, 32, 16	64	0.002	SGD	85.17	60,000 Training / 10,000 Testing
Experiment 5	3	128, 64, 32	64	0.0002	Adam	86.97	60,000 Training / 10,000 Testing
Experiment 6	2	256, 128	64	0.0001	RMSprop	87.62	60,000 Training / 10,000 Testing
Experiment 7	2	32, 32	64	0.01	SGD	86.01	60,000 Training / 10,000 Testing
Experiment 8	2	64, 64	64	0.001	Adam	87.38	60,000 Training / 10,000 Testing
Experiment 9	2	128, 128	64	0.0005	RMSprop	87.93	60,000 Training / 10,000 Testing
Experiment 10	3	256, 128, 64	64	0.005	SGD	85.53	60,000 Training / 10,000 Testing
Experiment 11	2	512, 256	64	0.0001	Adam	88.63	60,000 Training / 10,000 Testing
Experiment 12	3	32, 16, 8	64	0.002	RMSprop	87.17	60,000 Training / 10,000 Testing
Experiment 13	3	64, 32, 16	64	0.003	SGD	86.27	60,000 Training / 10,000 Testing
Experiment 14	3	128, 64, 32	64	0.001	Adam	87.7	60,000 Training / 10,000 Testing
Experiment 15	3	256, 128, 64	64	0.0002	RMSprop	88.9	60,000 Training / 10,000 Testing

For the Task3, sorry for not mention here. I will update the file in comment.
I have some trouble these days to see the doctor.

```
Running Experiment 2
Building and compiling the model...
Compiling model with optimizer and learning rate...
Starting training...
Epoch 1/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.7013 - loss: 0.8010 - val_accuracy: 0.8658 - val_loss: 0.3690
Epoch 2/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.8728 - loss: 0.3435 - val_accuracy: 0.8849 - val_loss: 0.3146
Epoch 3/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 15ms/step - accuracy: 0.8953 - loss: 0.2776 - val_accuracy: 0.8948 - val_loss: 0.2881
Epoch 4/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.9116 - loss: 0.2384 - val_accuracy: 0.9075 - val_loss: 0.2591
Epoch 5/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step - accuracy: 0.9229 - loss: 0.2102 - val_accuracy: 0.9097 - val_loss: 0.2494
Epoch 6/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.9317 - loss: 0.1809 - val_accuracy: 0.9112 - val_loss: 0.2454
Epoch 7/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.9406 - loss: 0.1600 - val_accuracy: 0.9027 - val_loss: 0.2753
Epoch 8/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step - accuracy: 0.9493 - loss: 0.1369 - val_accuracy: 0.9140 - val_loss: 0.2561
Epoch 9/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.9567 - loss: 0.1169 - val_accuracy: 0.9169 - val_loss: 0.2596
Epoch 10/15
750/750 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step - accuracy: 0.9617 - loss: 0.1014 - val_accuracy: 0.9072 - val_loss: 0.2941
Epoch 11/15
...
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.9820 - loss: 0.0473 - val_accuracy: 0.9121 - val_loss: 0.3781
Training complete, evaluating model...
Experiment 2 - Test Accuracy: 91.25%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
```

During training, it is easy to overfit, so much so that I need to keep pausing the training to adjust the training properties to ensure that overfitting disappears, but this is normal in CNN.

4. Discuss in a few sentences the results of your best and worst performing model. a. Were larger networks (structures with more hidden nodes) worth the trade off in training time? b. While the performance between FC and CNN can be large, does the training time and complexity of the CNN seem necessary for this task? Under what circumstances might this change? What if you applied augmentations? Does complexity of a dataset matter?

The best performing model is the one using CNN, as it is more effective in capturing the complex features of clothing in the FashionMNIST dataset. The worst performing model is a simple fully connected network, which is not able to extract spatial information in the image well, resulting in poor performance on the classification task.
If the task requires high accuracy and there are ample computational resources, then a larger network is worthwhile; however, if resources are limited, a trade-off may need to be made between accuracy and training time.
For datasets like FashionMNIST that have complex image features, the complexity of CNNs is necessary because CNNs are able to extract local features of the image that are difficult to capture with fully connected networks.
If data augmentation (such as rotation, flipping, scaling, etc.) is applied, the complexity of CNN may become more necessary, because data augmentation can generate more diverse training samples, help the model generalize, and reduce overfitting. For complex data sets, the structure and complexity of CNN help better capture the characteristics of the data; but for simple data sets, complex models may lead to overfitting.