Classifying Fashion-MNIST with an MLP-Mixer Architecture

Tasks 1-3

Please see sections 1-3 of the Jupyter Notebook.

Tasks 4 and 5 – Best Accuracy

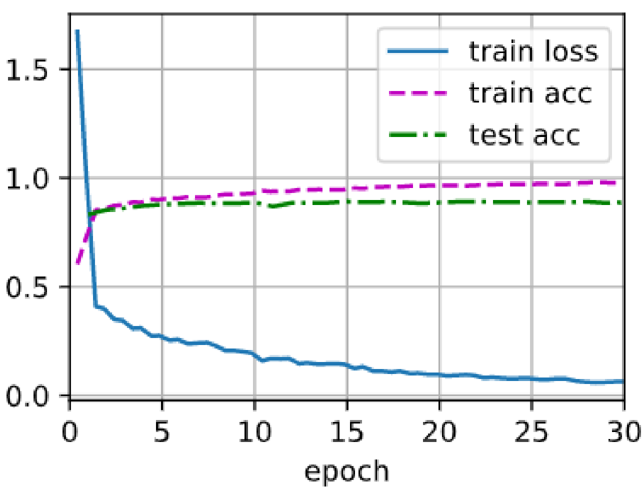
Development

Model Development

1 – Dropout and Data Augmentation

- After 30 epochs of training the model would often overfit the data. Introducing dropout and using training data augmentation helped prevent overfitting.
- I introduced a single dropout layer (p=0.25) immediately before the classifier.
- I also used torchvision.transforms.RandomResizedCrop(28, (0.9, 1)) to augment the training data in my_utils1.py (within load_data_fashion_mnist function).

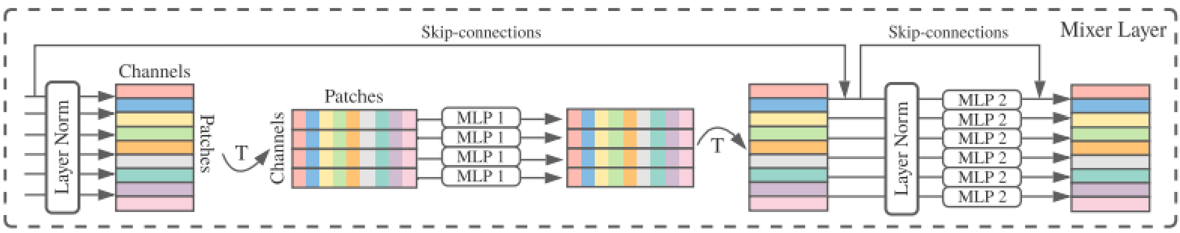
Figure 1 – Example of Observed Overfitting



2 - Layer normalisation before and after MLP 1 in each backbone block

- Tolstikhin et al.'s (2021) MLP-mixer architecture (which resembles the fundamental architecture used for this project and shown below) inspired my use of layer normalisation

Figure 2 – Mixer Layer from Tolstikhin et al.'s (2021) architecture



- Layer normalisation refers to normalising over features (shown as channels above)
- I also tried skip connections but did not use them in the final model

3 – Decreasing width

- I found a model which became narrower in the feature dimension with depth performed better than isotropic models (of constant width)
- This differs from Tolstikhin et al.'s (2021) architecture

Development of the Training Pipeline

- Used Xavier Normal initialisation of weights
- Lowered initial learning rate to 0.001 (from 0.1)
- Used ADAM optimisation (instead of SGD) – faster convergence but similar final accuracy
- Introduced data augmentation using random resized cropping
- Introduced an exponential learning rate scheduler

Changes that did not help

- Random vertical/horizontal flips, random rotations, or random erasing of training images
- Using weight decay
- Using 4 vs 16 patches
- Using more than 512 features per patch
- Using more than 2 blocks
- Training for more than 20 epochs
- Making hidden layers wider than input layers in the MLPs

Final Model

Final Training Parameters

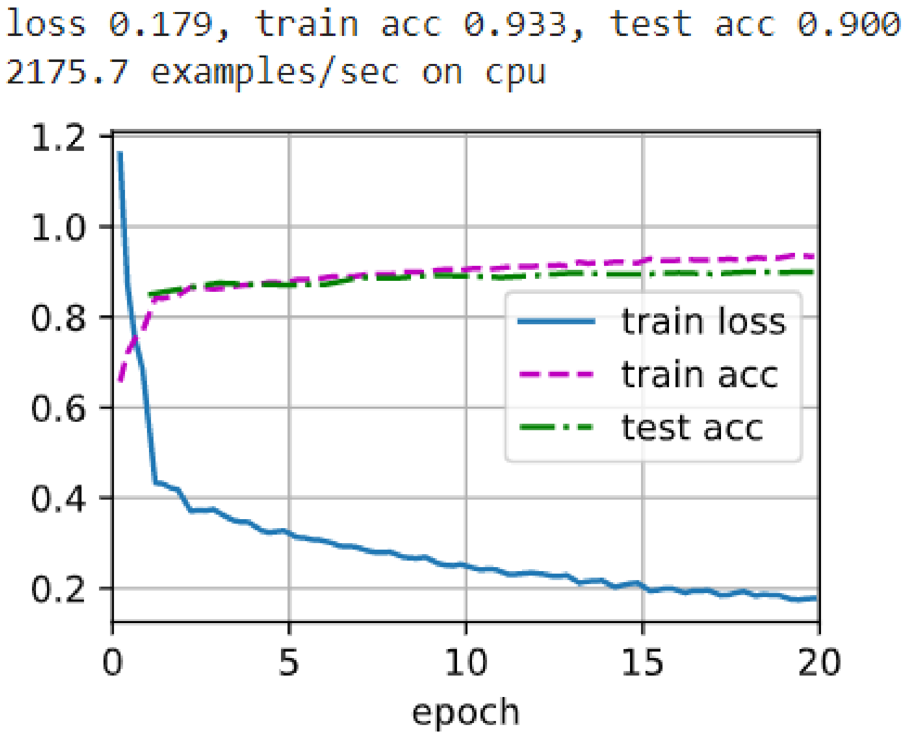
- Batch size 256
- 20 epochs
- Initial learning rate = 0.001
- Scheduler – gamma = 0.9

Final Model Parameters

- Patch size = 14x14, Number of patches = 4
- Number of features extracted per patch in stem = 512
- Number of blocks = 2
- Number of units in each layer (input, hidden, output):
    - Blocks 1
        - MLP 1: (16, 16, 16)
        - MLP 2: (512, 512, 256)
    - Block 2
        - MLP 1: (16, 16, 16)
        - MLP 2: (256, 128, 64)
- Number of outputs = 10
- Dropout rate = 0.25

Final Model Evolution Curves and Metrics

Figure 3 – Final Model Evolution Curves and Metrics



References

- my_utils from the lab session was used (and adapted) for reading the dataset, data loaders and training
- Code from other labs was used for model initialisation, loss and optimisation and training using a GPU
- Layer normalisation was inspired by Tolstikhin et al. (2021)
    - Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J. and Lucic, M., 2021. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems, 34.*