EE 562 project - Fashion-MNIST classification

by Yilin Shen, Jeffery Xu, Kai-Jen Zheng

1 Goal

We aim to work on modified version of project idea No.1, MNIST: Handwritten digit recognition. Instead of using original MNIST dataset, we will use Fashion MNIST dataset, which is:

- direct drop-in replacement of MNIST
- not that easy as MNIST and not overused
- able to present modern CV tasks

This dataset consists of 10 classes of fashion items, including T-shirts, coats and bags, etc. There are 60000 images for training and 10000 images for testing. All images are 28x28 grayscale image, associated with a label from 10 classes. The goal is to create three different classifiers, get the best accuracy of fashion item classification within the Fashion-MNIST dataset.

2 Methods

We will adopt deep learning with 3 different neural network structures to perform the classification, including VGG, ResNet and generic neural network. We will try different hyper-parameters and data preprocessing methods to explore best accuracy accross three classifiers.

2.1 ResNet

For ResNet classifier, ResNet-18 from `torchvision.models` is adopted. Since Fashion-MNIST's image size is as small as 28x28, there are not much of information contained, using a large neural network will easily to overfit, and take longer to train. There are also two ResNet's constraints, so the original image should be resized to 224x224, and each pixel should be normalized between 0 and 1. Thus, extra modification to first layer of ResNet-18 and input pre-processing is performed.

3 Experiments

3.1 ResNet

While adopting ResNet, several presets are still needed to make classifier start to train. These are:

- data transform sequences, such as crop, pad, flip, rotation
- loss function, such as mean absolute error, mean squared error, cross-entropy loss
- optimizer, such as Adam, SGD, RMSprop
- learning rate

- epoch

We will start from most adopted ones, along with basic parameters preset, to get a baseline result.

data transform	loss function	optimizer	learning rate	epoch
normalize	cross entropy	Adam	3e-4	5

Table 1. Presets for Resnet-18

Then we will try different options by changing only one at each time, to compare the performance difference with baseline. Modifications are shown below:

	data transform	loss function	optimizer	learning rate	epoch
Original	normalize	cross entropy	Adam	3e-4	5
Try #1					10, 20
Try #2			SGD, AdamW	1e-3 (for SGD)	
Try #3	ramdom flip + normalize				
Try #4		MSE, NLL			

Table 2. Combinations tried to optimize Resnet-18 training

Finally we will choose all the options that are beneficial to classification accuracy, apply them to baseline preset in a single run, to get the finalized version of trained classifier.

4 Results

4.1 ResNet

4.1.1 Baseline

Baseline classifier's accuracy is 92.28%.

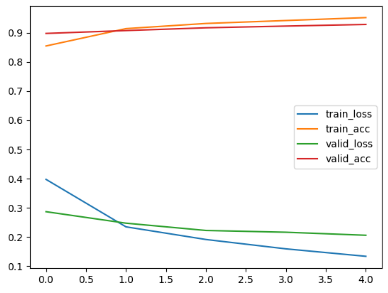

Figure 1. Epoch vs accuracy and loss

4.1.2 Epoch fine-tuning

We tried to make epoch more than 5, with 10 and 20. Accuacies are 92.69%, 92.34% respectively. Validation accuracy comes to a constant after epoch 4, also validation loss increases after epoch 4. This can be regarded as over-fitting.

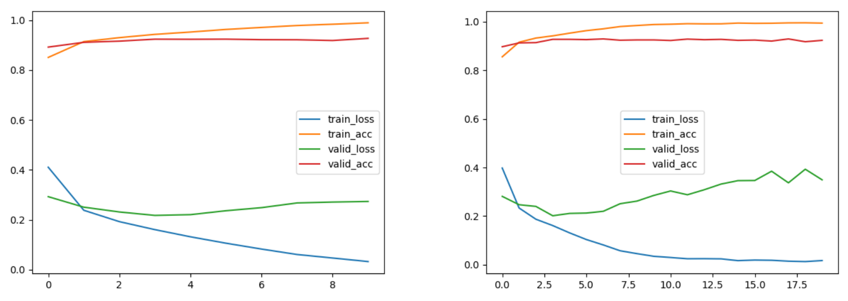

Figure 2. Epoch vs accuracy and loss

4.1.3 Data transform fine-tuning

Besides image normalization, random flopping is also added to image pre-process. This does not lead to much difference, because most of the image in dataset is self-symmetrical, and not distorted, not rotated.

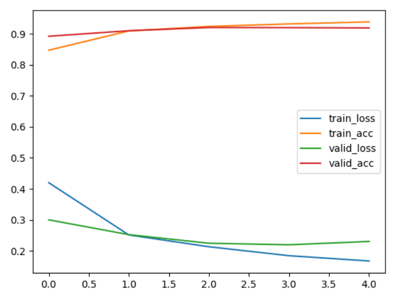

Figure 3. Epoch vs accuracy and loss

4.1.4 Loss function fine-tuning

For loss function, we also tried NLL and MSE. NLL stands for negative log-likelihood loss function, it is applied only on models with softmax, and it would not only care about correct prediction, but also prediction with high score. MSE stands for mean squared error, also called L2 loss. It would produce larger errors than smaller ones, punishing the model for making big mistakes. These two leads model accuracy to 92.25% and 92.5%.

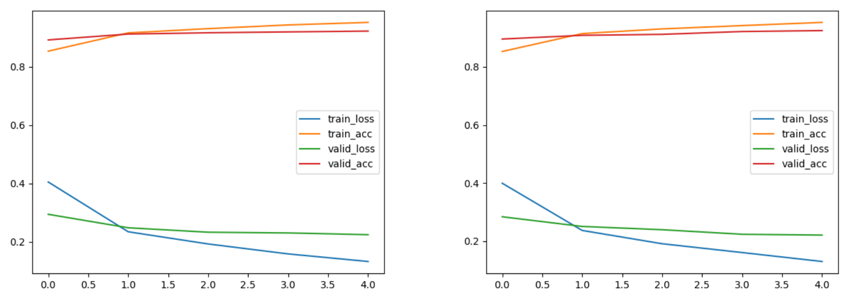

Figure 4. Epoch vs accuracy and loss

4.1.5 Optimizer fine-tuning

For optimizer, besides original Adam, we tried SGD and AdamW. SGD stands for stochastic gradient descent, and AdamW is an improved version of Adam class, which enhanced adam with weight decay. Accuracies are 88.27% and 92.48% respectively.

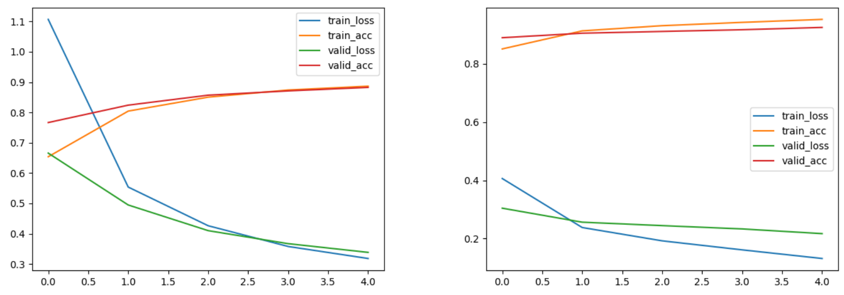

Figure 5. Epoch vs accuracy and loss

4.1.6 Combine all benificial options

We carefully identified these options' benefits and decided to apply random flip, AdamW and MSE to baseline and got a final accuracy of 92.86%.

5 Conclusions