Topics in Machine Learning: Classification of Fashion-MNIST Using Lightweight CNNs

Tzu-Liang Hsu

Manchester ID: 14268736

Email: tzu-liang.hsu@postgrad.manchester.ac.uk

1 Introduction

Fashion MNIST is a dataset of 60,000 grayscale images [1]. Each image has 28×28 pixels and is associated with one of 10 labels. The pixel value ranges from 0 to 255.

Convolutional Neural Networks (CNNs) excel at feature extraction for image classification [2,3]. For relatively simple datasets, lightweight CNN can achieve competitive performance while requiring only small number of parameters. In the following, we study the design and training of an efficient CNN model for classifying Fashion-MNIST images.

2 Methods

The methods consist of two main parts: the first part focuses on the data preprocessing, including train-validation split and data transformation. The second part details the CNN architecture, including training configuration, and the training procedure.

Dataset

The Fashion-MNIST dataset is split into 48,000 for training and 12,000 for validation, corresponding to an 80/20 split. Since k-fold cross validation is used later in training and the pixel values range from 0 to 255, min-max scaling is applied to normalise all images. In PyTorch, this can be achieved using ToTensor function. It converts Python Imaging Library or ndarray to tensor in the range $[0.0, 1.0]$.

k-Fold Cross Validation

k-fold cross validation is a robust method for accessing model performance for a fixed set of hyperparameters. Unlike single train-validation split, cross validation resamples the datasets. It allows the model to be evaluated using different portions of data. In this project, the dateset is split into 5 folds, satisfying 80/20 training-validation split.

Model Architecture

Performing 2-dimensional convolution over an grayscale input 28×28 image requires reshaping the input shape as $28 \times 28 \times 1$. The output feature dimensions for both convolutional and max-pooling layers can be determined by

$$N_{\text{out}} = \left\lfloor \frac{N_{\text{in}} + 2P - K}{S} \right\rfloor + 1, \tag{1}$$

where N_{in} is the number of input features, P is the padding size, K is the kernel size, and S is the stride size. The number of parameters in a convolutional layer is

$$\text{Params} = (K \times K \times C_{\text{in}} \times C_{\text{out}}) \; + \; C_{\text{out}}, \tag{2}$$

where C_{in} and C_{out} are numbers of input and output channels, respectively.

To build an efficient CNN model, the architecture begins with two consecutive blocks, followed by a fully connected dense layer [3]. Figure 1 shows the overall model architecture. In the first block, the input has 1 channel, and the convolution outputs 16 channels. To keep the spatial dimension, we choose $P = 1$, $K = 3$, and $S = 1$, leading to the output feature shape $28 \times 28 \times 16$ based on Eq.(1). From Eq. (2), the first block has 160 parameters. Subsequently, a ReLU activation with He-initialisation is applied [4], followed by a max-pooling layer with $P = 0$, $K = 2$, and $S = 2$. The spatial dimension is then shrank to $14 \times 14 \times 16$.

The second block has the same structure but outputs 32 channels with size $7 \times 7 \times 32$. The total number of parameters thus is 4,640. After the two consecutive blocks, the output image is flattened into a vector with length 1,568, sent into a fully connected dense layer with 10 outputs (number of classes). As a result, this layer adds 15,690 parameters, resulting in a number of 20,490 parameters for the CNN model.

Training Configuration

The final dense layer uses a Softmax activation to compute the class probabilities. Cross entropy is then computed as the loss function, and the Adam optimiser is used to minimise the loss. The whole training is performed using MacBook M4 GPU.

The values of hyperparmeters are determined through a grid search method. The possible candidate learning rates and batch sizes are shown in Table. 1. Each combination is evaluated using 5-fold cross validation with 10 epochs per fold.

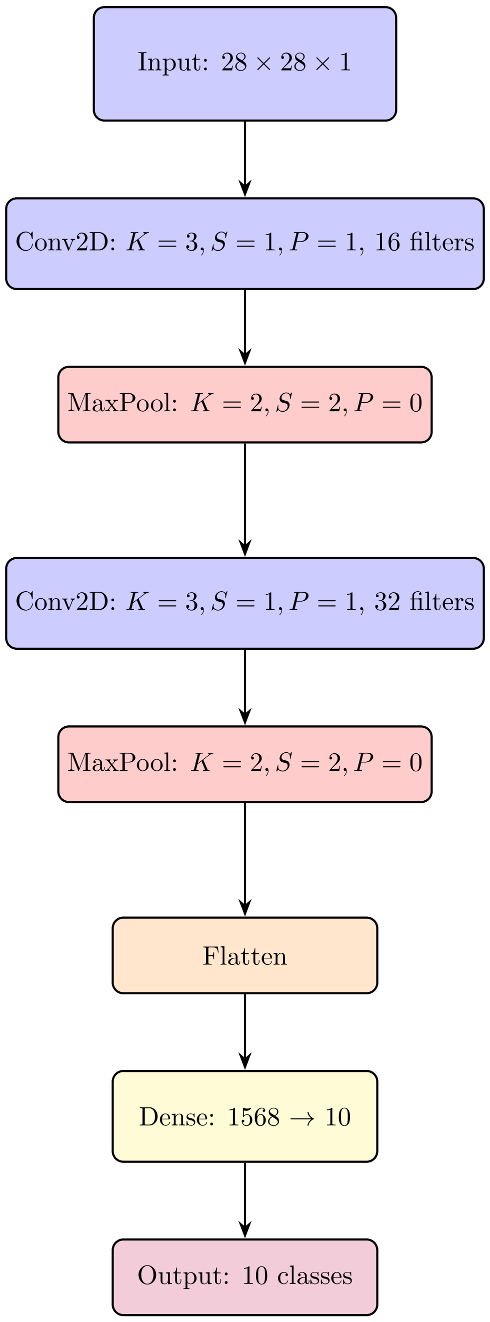
Input: $28 \times 28 \times 1$
Conv2D: $K = 3, S = 1, P = 1$, 16 filters
MaxPool: $K = 2, S = 2, P = 0$
Conv2D: $K = 3, S = 1, P = 1$, 32 filters
MaxPool: $K = 2, S = 2, P = 0$
Flatten
Dense: $1568 \rightarrow 10$
Output: 10 classes

Figure 1: **CNN Architecture.**

Table 1: **Hyperparameter search candidates.**

Hyperparameter	**Candidates**
Learning rate	$\{0.005, 0.001, 0.0005, 0.0001\}$
Batch size	$\{32, 64, 128\}$

3 Results

The following results present the validation performance obtained across different data folds. We also demonstrate the efficiency and effectiveness of the CNN model and the method used to avoid overfitting.

Hyperparameter Search

All possible combinations of the hyperparameters listed in Table 1 were evaluated using 5-fold cross validation with 10 epochs. Table 2 summarises the average validation loss and accuracy for each combination. The optimal hyperparameter setting is a learning rate of 0.001 and a batch size of 64, corresponding to an average validation loss of 0.2535 and an accuracy of 0.9112.

With the optima hyperparameter setting, a final training was conducted using an 80/20 split of the full dataset. The final model was trained for up to 50 epochs, with early stopping using a patience of 5. The training history of the final model is shown in Fig. 2. The resulting model has an accuracy of 0.904400 with 20,490 parameters.

Table 2: **Validation performance across hyperparameter settings.**

Learning rate	**Batch size**	**Avg Val Loss**	**Avg Val Accuracy**
0.005	32	0.3362	0.8956
0.005	64	0.2968	0.9034
0.005	128	0.2923	0.9048
0.001	32	0.2652	0.9092
0.001	64	0.2535	**0.9112**
0.001	128	0.2613	0.9069
0.0005	32	0.2563	0.9089
0.0005	64	0.2665	0.9055
0.0005	128	0.2872	0.8986
0.0001	32	0.3212	0.8879
0.0001	64	0.3369	0.8819
0.0001	128	0.3608	0.8742

4 Discussion

Table. 2 demonstrates that the learning rate has a stronger influence on model performance. When the learning rate is set to 0.001, the average validation accuracies obtained via 5-fold cross validation are around 0.91. In contrast, using a batch size of 128 generally yields poor performance across different learning rates, with the only exception occurring when the learning rate is 0.005.

Figure. 2 shows the training history for both the training and validation datasets. As expected,

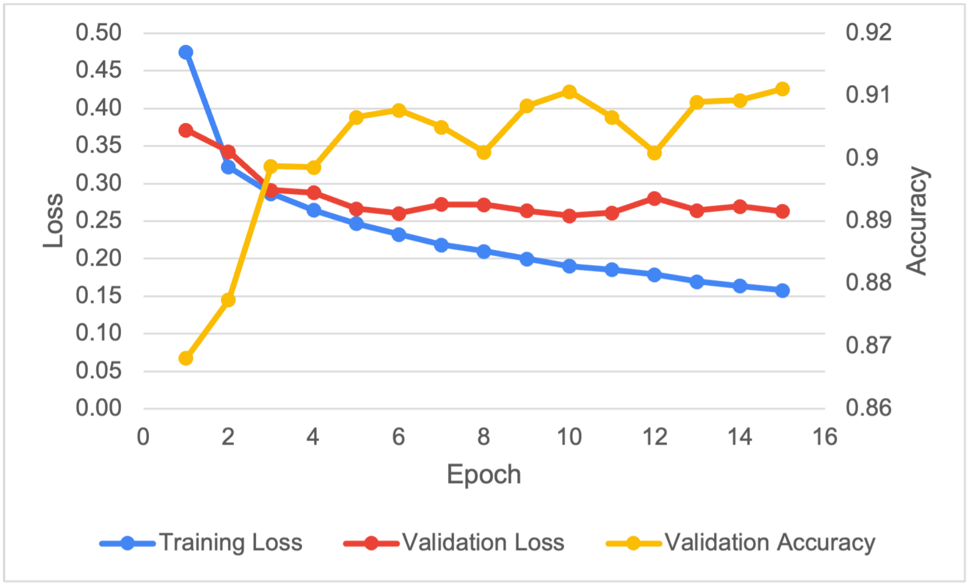

Figure 2: **Training History of the final model.**

as the number of epochs increases, both the training loss and validation loss generally decrease, while the validation accuracy increases. However, when epoch is around 6, the validation loss begins to rise slightly, showing the onset of overfitting. To address this, the training is halted when the validation loss fails to improve for 5 consecutive epochs. Hence, the early stopping effectively prevents further overfitting and ensures that the best-performing weights are retained.

5 Conclusion

To sum up, the efficient and lightweight Convolutional Neural network (CNN) has achieved an accuracy of 0.90 on the Fashion-MNIST dataset, with only around 20k parameters. The model consists of two consecutive convolutional-pooling blocks followed by a dense output. Cross validation approach was used to reliably determine the optimal hyperparameter setting. In the final training, early stopping with a patience of 5 was applied to prevent overfitting and ensure the best-performing weights were retained.

1. Xiao, H., Rasul, K. & Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

2. O'shea, K. & Nash, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).

3. Isong, N. Building efficient lightweight cnn models. *arXiv preprint arXiv:2501.15547* (2025).

4. He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034 (2015).