ECSE 551 Group 17 Mini-Project 3

Hongtao Xu 260773785
Homayoun Sohrabpour Shafti homayoun.sohrabpourshafti@mail.mcgill.ca
Norman Kong 260923365

Abstract

To classify images effectively, the preferred machine learning model is a convolutional neural network. The modified Fashion-MNIST dataset was used for image classification for this particular project. Multiple network architectures and optimization hyperparameters were evaluated to ensure the best performance. The model was submitted to the Kaggle competition, and a preliminary accuracy of 77.9% was achieved.

1. Introduction

Due to its ability to tackle complex problems that are otherwise difficult to solve, machine learning has become a widely applicable tool in the modern world. Computer vision is one significant area of study in machine learning, and Convolutional Neural Networks (CNNs) is a popular vision algorithm used to perform recognition tasks on visual data, with the goal of gaining a deeper understanding of real-world scenes.

The objective of this mini-project is to utilize machine learning for image recognition. The project revolves around a 10-class classification problem using the Fashion-MNIST dataset and the Japanese Integers, developed for this mini-project. The dataset consists of images that contain clothing articles and Japanese digits, with the goal of recognizing the Japanese digit present in each image. This is a supervised classification task, which means that each image is associated with a label and the objective is to predict this label. We are required to design and validate a supervised classification model for this task, and we are free to use any model we wish, provided it is written in Python using PyTorch. To accomplish this, multiple CNN models were constructed and compared using different performance benchmarks to determine the best architecture. Additionally, the hyper-parameters of each model were tuned to maximize their efficiency and accuracy on the test set. Finally, the CNNs were trained and evaluated on a modified version of the Fashion-MNIST dataset.

2. Dataset

There are two different datasets that we work with them in this project. Train and test.
The training dataset is a collection of grayscale clothing articles and Japanese digits images. This dataset contains 60,000 samples in which each sample has a fixed size of 28*28 pixels. Also, the label of the training dataset is available in a separate file.
On the other hand, the test dataset contains 10,000 samples. However, the size of the image is the same as the train data.

3. Proposed Method

The method proposed in this report divides into two steps including preprocessing and building a CNN model. In this section, each of these steps will be discussed.

3.1 Preprocessing

In this step, a class is defined in order to read the data from the train set and train label and transform data inside the training dataset. In the context of deep learning and computer vision, "transforms" typically refer to a set of predefined image transformations that are commonly used for data augmentation. Data augmentation is a technique used to artificially increase the size of a training dataset by generating additional training examples through various modifications of the original images. This can help improve the performance and robustness of deep learning models by exposing them to a wider range of variations in the training data. After that, data was split to train and test set in order to prepare the data for the training step. 80% of the training set was used for training the model and 20% was used for validation of the model during the training phase.

3.2 CNN Model

The CNN takes in 28x28 grayscale images and outputs a prediction of which of the 10 classes the image belongs to. The network consists of two convolutional layers (conv1 and conv2) which apply filters to the input image to extract features, followed by two fully connected layers (fc1 and fc2) which perform classification on the extracted features. After each convolutional layer, there is a max pooling layer which reduces the spatial size of the feature maps. The ReLU activation function is applied after each layer except the output layer which uses the softmax activation function. The network has a total of 353,410 parameters that need to be learned during training.

3.2.1 CNN Hyper Parameters

The batch size is set to 64 for both the training and testing data loaders, and the data is shuffled at the beginning of each epoch. The convolutional layers use a kernel size of 5. The learning rate is set to 0.1 for the first 35 epochs and then reduced to 0.01 for the following 15 epochs. The momentum parameter of the optimizer is set to 0.9 for all epochs, and L2 regularization is applied with a weight decay of 0.001 for all epochs. These parameters are important in determining the performance and convergence of the neural network model during training.

3.2.2 Training and Testing

PyTorch code for training and testing a neural network using cross-entropy loss. The cross-entropy loss is computed between the predicted output and the actual target.

3.2.2.1 Training

The train function is called once for each epoch during training. It sets the network to training mode, iterates through the batches in the training dataset using the train_loader, computes the loss, and backpropagates the gradients to update the model parameters using the optimizer. It also saves the current state of the model and optimizer to files. Every 20 batches, it prints out the training progress, including the epoch number, batch index, total number of batches, and loss. The output of the train function is the training progress which includes the epoch number, batch index, total number of batches, and the loss.

3.2.2.2 Testing

The test function is called after training to evaluate the performance of the trained model on the test dataset. It sets the network to evaluation mode, iterates through the batches in the test dataset using the test_loader, computes the loss and accuracy of the predictions, and saves the results in the test_losses and scores lists. It prints out the average loss and accuracy of the model on the test set. The output of the test function is the average loss and accuracy of the model on the test set.

In the next section, the result of these functions will be discussed.

4. Results

The results of train functions were measured for 35 epochs. After each epoch, the average loss function was calculated using the test function. Table 1 is showing the results of average loss and accuracy for different epochs. As it shows, after epoch 10 there was not a significant change in the accuracy and average loss.

Table 1. Result of average loss and accuracy for different epochs

	Average loss	Accuracy
Epoch 1	1.36	52%
Epoch 10	0.49	84%
Epoch 15	0.48	84%
Epoch 25	0.42	87%
Epoch 34	0.42	87%

Figure 1 plots the cross entropy loss over the number of training examples seen during training. The x-axis represents the number of training examples seen, and the y-axis represents the cross entropy loss. The plot shows the trend of the training loss over time.

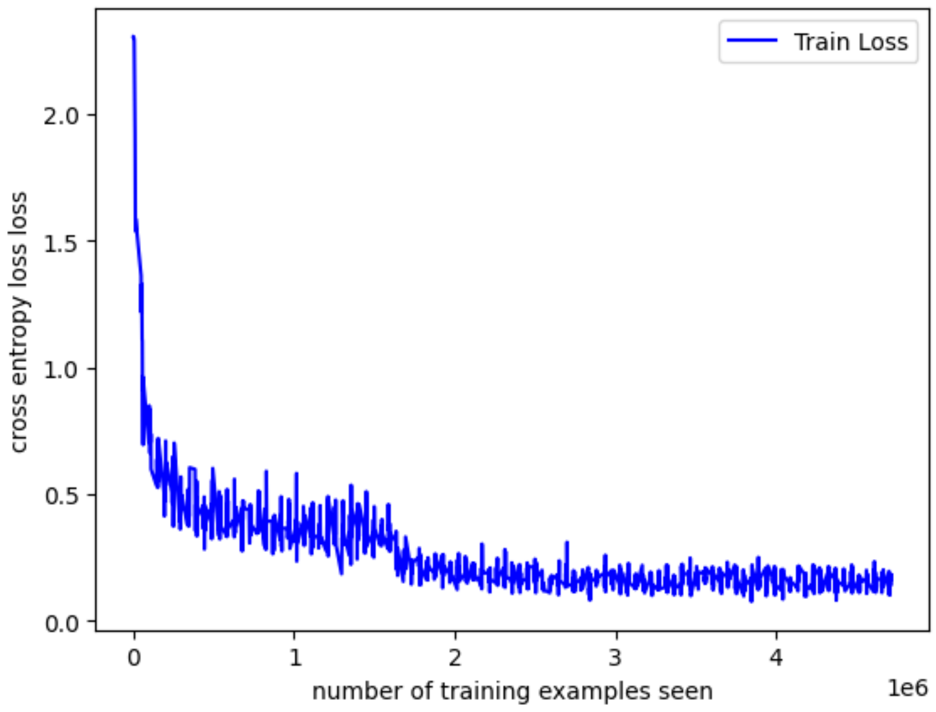


Figure 1. The trend of training loss over time

Figure 2 illustrates the accuracy score of the neural network on the test data as training progresses. The x-axis shows the number of training examples seen, while the y-axis shows the accuracy score as a percentage. The figure shows how the accuracy score improves with training, and how the rate of improvement decreases over time.

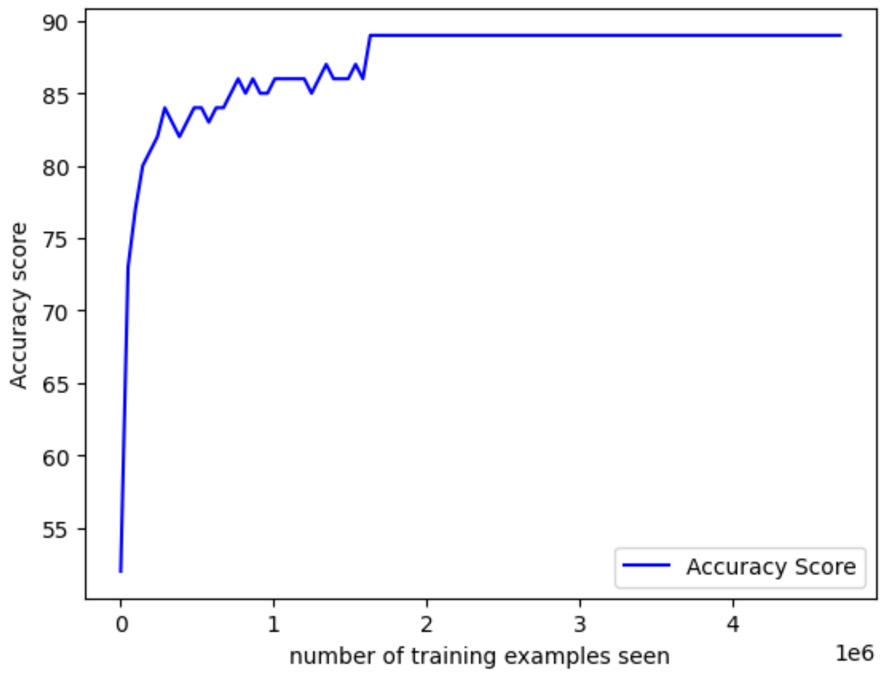


Figure 2. The accuracy score of the neural network

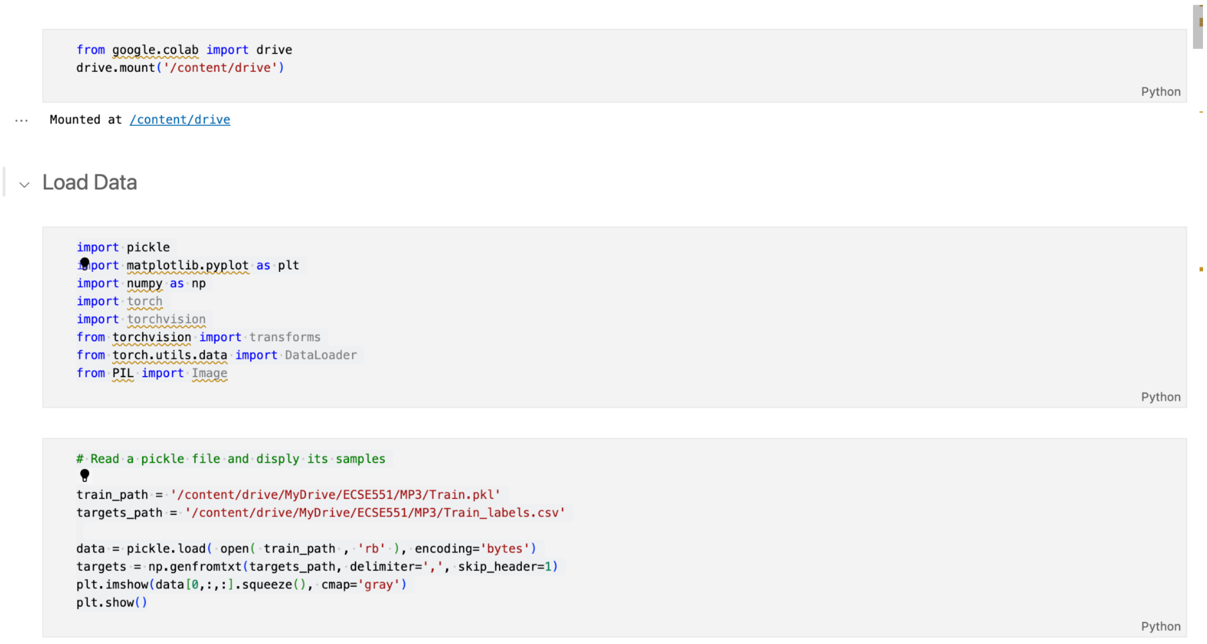
5. Discussion and Conlusion

A CNN was created in the report to classify a modified version of the Fashion-MNIST data set, and various experiments were conducted to explore the CNN architecture and the training method used. Through these experiments, trends were identified for learning rate, batch size, and dropout that led to the best final model. By tuning the additional parameters, using the Adam optimization algorithm to train the weights, providing the best outcome with 77.9% accuracy on the Kaggle validation set. Despite the strong performance of the final CNN, there are other possible avenues for future work, such as exploring weight decay regularization to enhance the model's generalization. Other popular CNN architectures, like ResNet and GoogLeNet, need to be tested, and other error metrics like loss, precision, recall, and F-score should be incorporated into the analysis to gain insights into the current model's weaknesses.

6. Statement of Contribution

1. Norman Kong: Data pre-processing, implementing of the algorithm, write-up contribution.
2. Homayoun Sohrabpour Shafti: Data pre-processing, implementing of the algorithm, write-up contribution.
3. Hongtao Xu: Data pre-processing, implementing of the algorithm, write-up contribution.

7. Appendix



```python
from google.colab import drive
drive.mount('/content/drive')
```
Python

```
Mounted at /content/drive
```

Load Data

```python
import pickle
import matplotlib.pyplot as plt
import numpy as np
import torch
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader
from PIL import Image
```
Python

```python
# Read a pickle file and disply its samples

train_path = '/content/drive/MyDrive/ECSE551/MP3/Train.pkl'
targets_path = '/content/drive/MyDrive/ECSE551/MP3/Train_labels.csv'

data = pickle.load( open( train_path , 'rb' ), encoding='bytes')
targets = np.genfromtxt(targets_path, delimiter=',', skip_header=1)
plt.imshow(data[0,:,:].squeeze(), cmap='gray')
plt.show()
```
Python