Project Design Documentation

Author: Liu, Zhenyan

1. Motivation

Deep Learning, a kind of Artificial Intelligence, is everywhere, like face recognition when checking passports or voice recognition when entering some places. Neural Network is the core of deep learning, and plenty of cutting-edge research are using deep neural networks. Therefore, I choose deep learning models as my project topic.

Classification is major task of deep learning. Some classification models can even reach higher accuracy than human beings. The task of classification can help us to better understand how deep learning works.

2. Datasets

There are three datasets in this project, belonging to three different types:

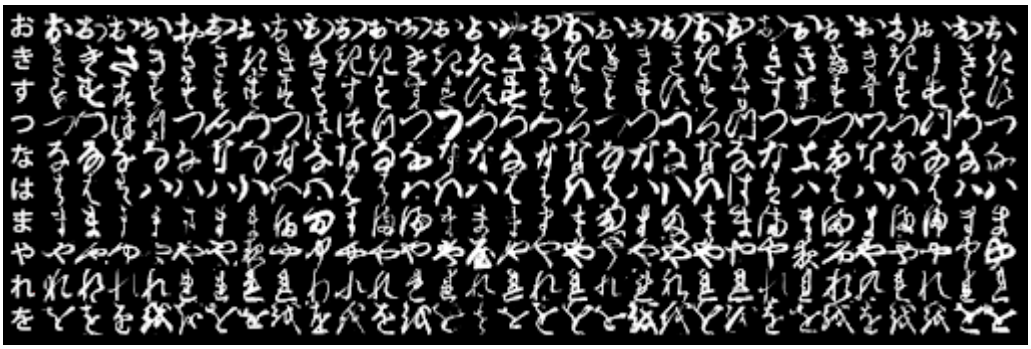
a. MNIST

A dataset recognizing as “digit”. Each image is a handwritten digit, and the value of that digit is between 0 and 9.

b. KMNIST

A dataset recognizing as “character”. Each image is a handwritten Japanese character. The classes in order are:

"o", "ki", "su", "tsu", "na", "ha", "ma", "ya", "re", "wo"



c. FashionMNIST

A dataset recognizing as “fashion”. Each image contains a fashion stuff. The classes are: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

All of the three datasets have the same image size: 28*28 and 10 categories, so that we can apply the same model without modifying any shape values.

3. Implementation

a. Project Source Code Organization

```
|----data (Storing three datasets, automatically downloaded)
|     |----FashionMNIST
|     |----KMNIST
|     |----MNIST
|----static (Storing images uploaded by users)
|     |----images (Directory storing images)
|----templates (Web app)
|     |----upload.html (home web page)
|     |----upload_ok.html (web page after submitting an image)
|----network.py (Three types of neural networks)
|----img_transer.py (Transform the input image into specific size and grey color)
|----train_test.py (train and test the model, save model parameters)
|----demo.py (launch a web application used for demonstration)
|----[*.pkl] (stored model parameters used in demonstration part)
```

Network.py contains three classes: Lin() for linear function, Full() for fully connected network, Conv() for convolutional neural network. Each time we choose one network and run the program. Details explained in the next section.

[*.pkl] are:
digit_lin_params.pkl, digit_full_params.pkl, digit_conv_params.pkl
character_lin_params.pkl, character_full_params.pkl, character_conv_params.pkl
fashion_lin_params.pkl, fashion_full_params.pkl, fashion_conv_params.pkl
These nine pickles save model parameters corresponding to nine models: three datasets and their three types of models, respectively.

b. Parameters:
Batch_size = 64, learning_rate = 0.01, momentum = 0.5, epochs = 10

4. Models

There are three classes representing three types of networks in network.py

a. Lin()
A linear function followed by log_softmax:

$$y = \text{log\_softmax}(W^T x + b)$$

b. Full()
Two fully connected tanh layers followed by log_softmax:

$$y^0 = \tanh\big(\text{fc1}(x)\big)$$
$$y = \text{log\_softmax}(\text{fc2}(y^0))$$

c. Conv()
Two convolutional layers and one fully connected layer, all activation functions using relu, followed by log_softmax.

$$y^0 = \text{conv2}\big(\text{conv1}(x)\big)$$
$$y = \text{log\_softmax}(\text{fc}(y^0))$$

5. Ablation

I conduct ablation studies on three types of models, based on KMNIST dataset.

a. For Linear function, try to change batch_size to 16:
Test set: Average loss: 1.0733, Accuracy: 6909/10000 (69%)
Whether the batch_size is 16 or 64 has no big difference for this task.

b. For Fully connected layers, there is a hyperparameter named hidden_size, which is the number of nodes between the first layer and the second layer.
Hidden_size = 50: *Test set: Average loss: 0.5354, Accuracy: 8369/10000 (84%)*
Hidden_size = 100: *Test set: Average loss: 0.5340, Accuracy: 8341/10000 (83%)*
Hidden_size = 300: *Test set: Average loss: 0.4903, Accuracy: 8453/10000 (85%)*
Hidden_size = 500: *Test set: Average loss: 0.4905, Accuracy: 8379/10000 (85%)*
The value achieving high accuracy on test set should be large, like 300 or 500. Introducing 500 more hidden nodes is not necessary, so I choose 300 as the number of hidden nodes.

c. For Convolutional neural network, I remove the second convolutional layer.
Test set: Average loss: 0.3696, Accuracy: 8981/10000 (90%)
The accuracy becomes lower than convolutional network. It is natural as the network is simpler than before, so it may not express images as well as two convolutional layers.

6. Result Analysis

MNIST:
Lin: *Test set: Average loss: 0.2921, Accuracy: 9115/10000 (91%)*
Full: *Test set: Average loss: 0.1464, Accuracy: 9559/10000 (96%)*

Conv: *Test set: Average loss: 0.0296, Accuracy: 9887/10000 (99%)*

KMNIST:
Lin: *Test set: Average loss: 1.0220, Accuracy: 6918/10000 (69%)*
Full: *Test set: Average loss: 0.4912, Accuracy: 8453/10000 (85%)*
Conv: *Test set: Average loss: 0.2403, Accuracy: 9312/10000 (93%)*

FashionMNIST:
Lin: *Test set: Average loss: 0.4624, Accuracy: 8365/10000 (84%)*
Full: *Test set: Average loss: 0.3767, Accuracy: 8646/10000 (86%)*
Conv: *Test set: Average loss: 0.2699, Accuracy: 9019/10000 (90%)*

According to the performances of the three types of models on three different datasets, the accuracy of convolutional neural network is much better than linear function, and the performance of fully connected network (or simple forward neural network) is between them.

It seems like that the more complex the model is, the more accurate it is. This is because all the three models are relatively easy compared to really complicated models, so they are not likely to face overfitting problems. Therefore, a more complicated model, such as convolutional neural networks, can be more expressive for image information and suits quite well for image processing.

7. Demonstration

As mentioned above, the accuracy of convolutional model is highest among the three models, no matter on which dataset.

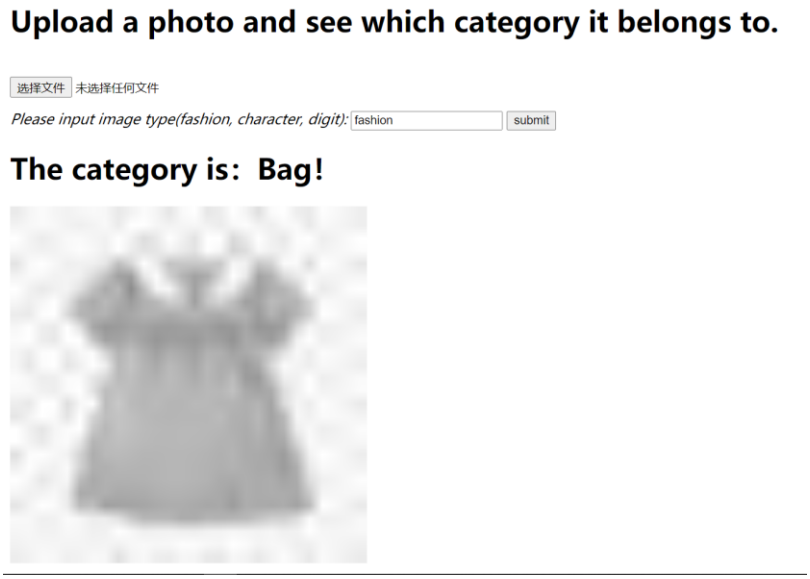
So, I designed this web page shown as below, which uses convolutional neural networks as the classification model for input images. It loads the parameters stored in training step, which means the model does not need to be trained again so that the response time can be very short.

Demonstrating steps:

a. Run *python demo.py* and visit http://127.0.0.1:5050/.
b. Upload an image and input which type it is. The type could only be one of “fashion”, “character”, “digit”.
c. Click submit, and the web page will output which category it belongs to after classification, and show the image processed by our program.

8. Future Work

Let’s see a failure case:



It fails because of the image preprocessing part. The images used in training, testing and verifying are of shape 28 * 28, grey-colored. However, if a user submits a colored large image, the processing section will first transform it into grey-colored, and then decrease the size into 28 * 28, no matter how large the original image is, even though it may be 1024 * 1024. We can see from the web page: the transformed image loses many characteristics and provides less information for models, and even ourselves cannot tell whether it is a bag or it is a skirt only based on the transformed image.

For future work, there are two solutions:

a. Change the image preprocessing part, so that it can maintain the most useful information while cutting the size of the original image.
b. Change the model itself, so that it can handle different sizes of input. Making use of pooling operation may be a possible approach.