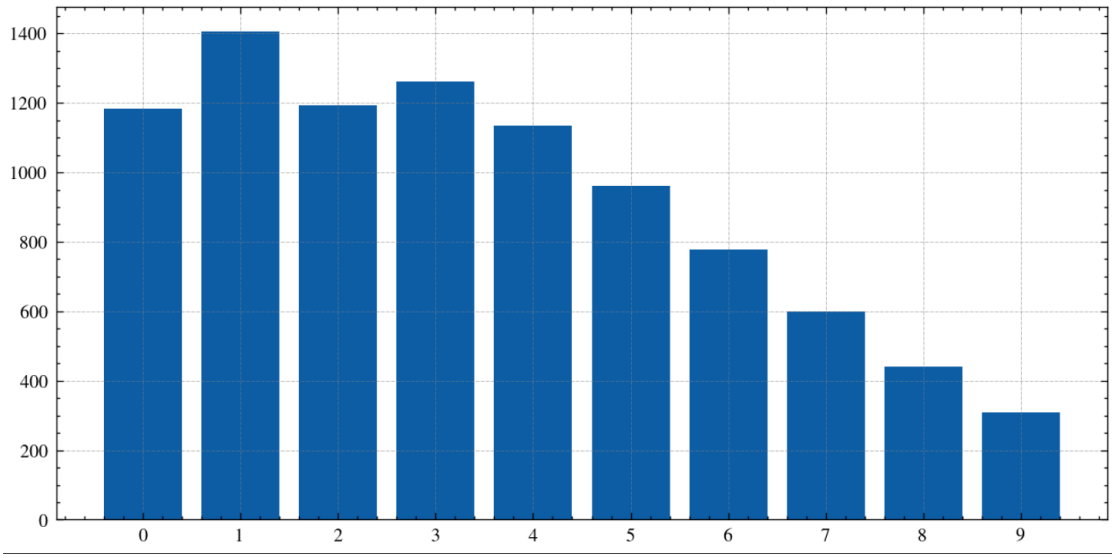
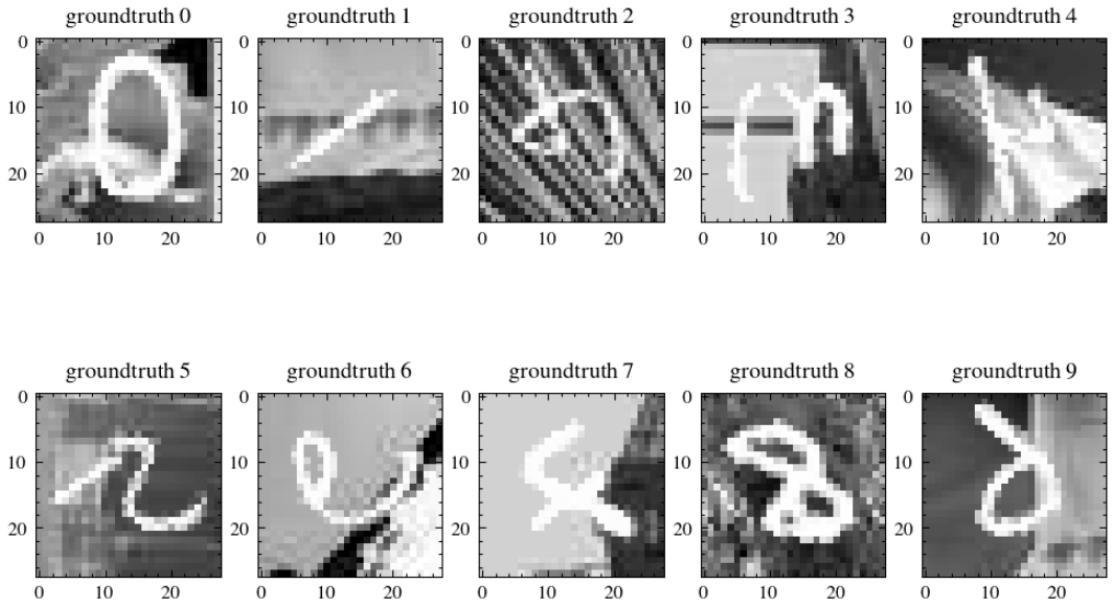
Task1

(1) Label distribution



The images below demonstrate the number of training examples belong to each label. It is clear that the distribution among labels is unbalanced. The label “1” counts the most and the label “9” counts the less. The number of “1” is 4 times larger than the number of “9”.

(2) Difference with MNIST



The figure above shows the example of each digit. It is clear that they are unclear, distorted, and even difficult to classify using human eyes. And the background is messy, unlike MNIST, which are pure black and white.

Task2

1. Data Processing

First, due to unbalance described in Task1, it is necessary to fix it. For each label except “1”, I resampled examples for each label and append them into the original dataset until the number of digit is equal to the number of “1”. Thus, they are balanced.

Then, to validate the model, I shuffle the dataset and split the training set and validation set using 8:2 Proportion.

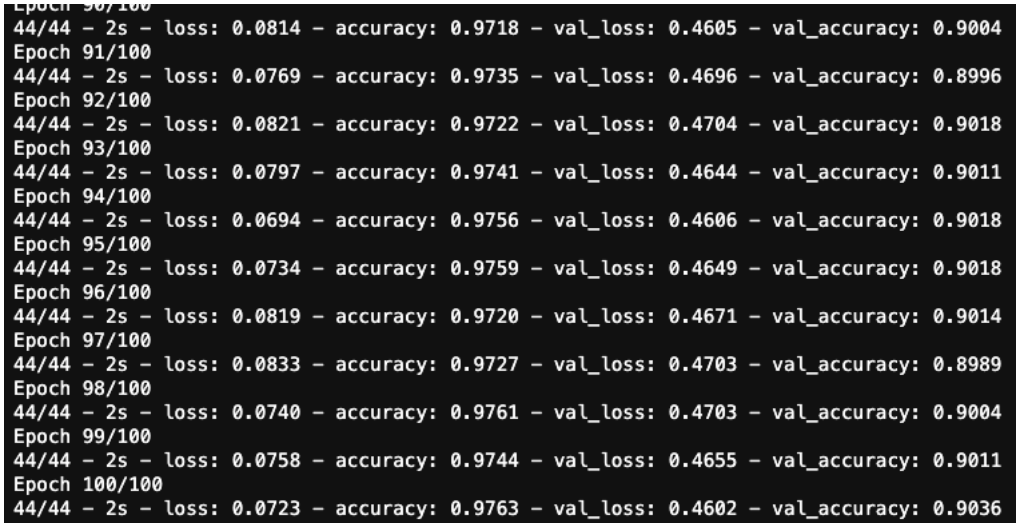
2. Model

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
batch_normalization_5 (Batch (None, 13, 13, 32)        128
_________________________________________________________________
dropout_4 (Dropout)          (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 11, 11, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 32)          0
_________________________________________________________________
batch_normalization_6 (Batch (None, 5, 5, 32)          128
_________________________________________________________________
dropout_5 (Dropout)          (None, 5, 5, 32)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 5, 5, 64)          51264
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
batch_normalization_7 (Batch (None, 2, 2, 64)          256
_________________________________________________________________
dropout_6 (Dropout)          (None, 2, 2, 64)          0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               8320
_________________________________________________________________
batch_normalization_8 (Batch (None, 128)               512
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
batch_normalization_9 (Batch (None, 64)                256
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                650
=================================================================
Total params: 79,338
Trainable params: 78,698
Non-trainable params: 640
```

The figure demonstrates the structure of my model. It is quite similar to LENET-5 but the layers are slightly different. There are three Convolution 2D layers with kernel size 32,32, and 64, respectively. And for each convolutional 2D layer, a Maxpooling2D layer, a batch normalization layer, and a dropout layer with probability 0.15 are followed. Then, a global average pooling 2D layer flatten the feature map, followed by two dense layers with unit size 128 and 64. In the end, the dense 10 layer is used as the output layer.

RMSProp is used as optimizer. And the model is trained on the training set with 100 epochs and 256 batch size. And a learning rate scheduler decrease the learning rate to its 0.94 every two epochs.

The final accuracy on the validation set is around 90%.



Final, I train the model using the same parameters on the full training set again and output the model to local file.