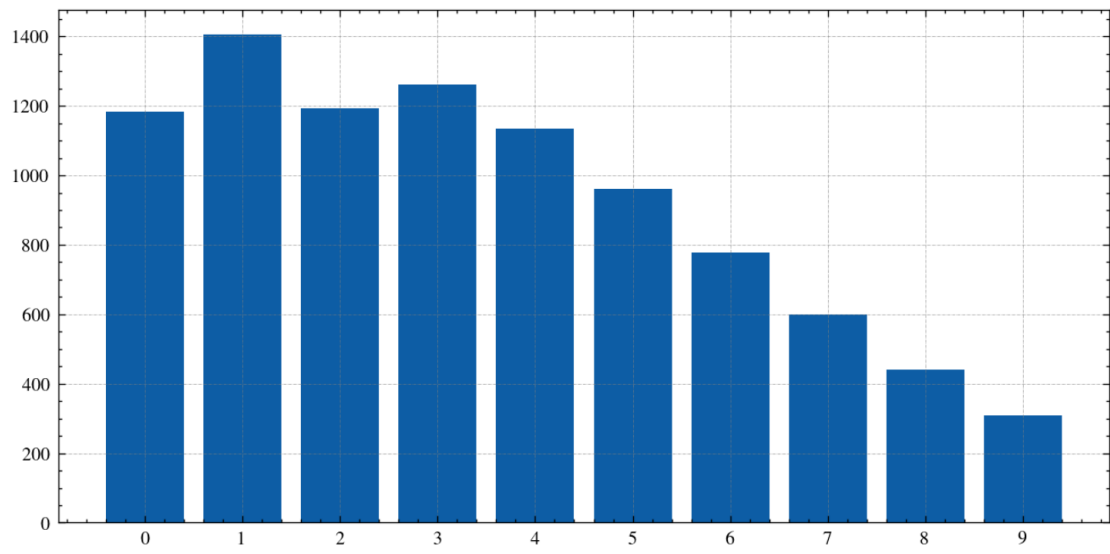


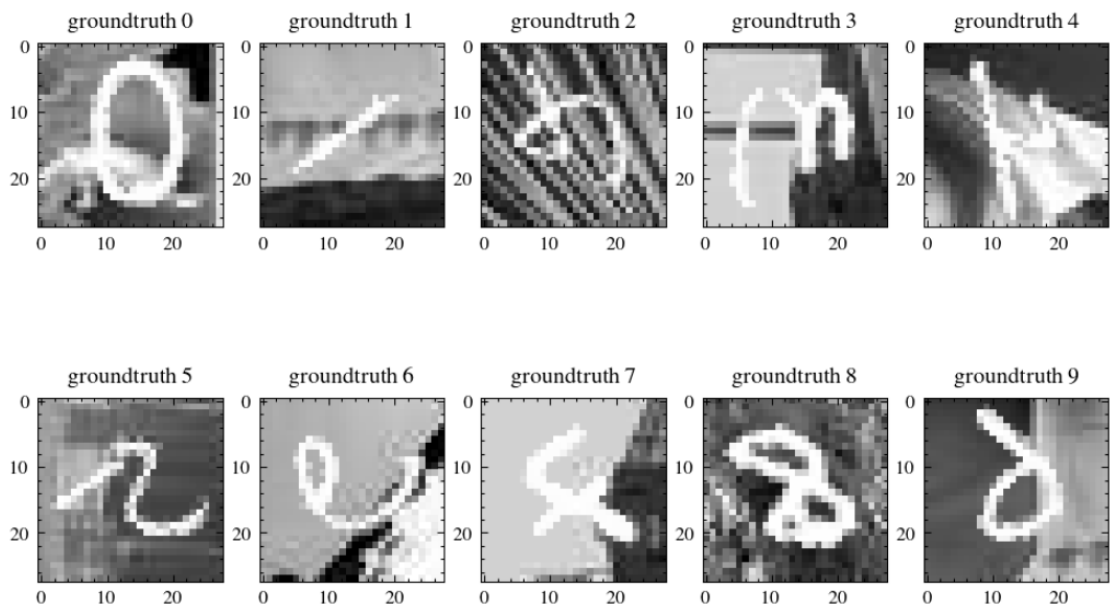
Task1

(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 (MaxPooling2D)	(None, 13, 13, 32)	0
batch_normalization_5 (Batch Normalization)	(None, 13, 13, 32)	128
dropout_4 (Dropout)	(None, 13, 13, 32)	0
conv2d_4 (Conv2D)	(None, 11, 11, 32)	9248
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 32)	0
batch_normalization_6 (Batch Normalization)	(None, 5, 5, 32)	128
dropout_5 (Dropout)	(None, 5, 5, 32)	0
conv2d_5 (Conv2D)	(None, 5, 5, 64)	51264
max_pooling2d_5 (MaxPooling2D)	(None, 2, 2, 64)	0
batch_normalization_7 (Batch Normalization)	(None, 2, 2, 64)	256
dropout_6 (Dropout)	(None, 2, 2, 64)	0
global_average_pooling2d_1 (Global Average Pooling2D)	(None, 64)	0
dense_3 (Dense)	(None, 128)	8320
batch_normalization_8 (Batch Normalization)	(None, 128)	512
dense_4 (Dense)	(None, 64)	8256
batch_normalization_9 (Batch Normalization)	(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%.

```
Epoch 90/100
44/44 - 2s - loss: 0.0814 - accuracy: 0.9718 - val_loss: 0.4605 - val_accuracy: 0.9004
Epoch 91/100
44/44 - 2s - loss: 0.0769 - accuracy: 0.9735 - val_loss: 0.4696 - val_accuracy: 0.8996
Epoch 92/100
44/44 - 2s - loss: 0.0821 - accuracy: 0.9722 - val_loss: 0.4704 - val_accuracy: 0.9018
Epoch 93/100
44/44 - 2s - loss: 0.0797 - accuracy: 0.9741 - val_loss: 0.4644 - val_accuracy: 0.9011
Epoch 94/100
44/44 - 2s - loss: 0.0694 - accuracy: 0.9756 - val_loss: 0.4606 - val_accuracy: 0.9018
Epoch 95/100
44/44 - 2s - loss: 0.0734 - accuracy: 0.9759 - val_loss: 0.4649 - val_accuracy: 0.9018
Epoch 96/100
44/44 - 2s - loss: 0.0819 - accuracy: 0.9720 - val_loss: 0.4671 - val_accuracy: 0.9014
Epoch 97/100
44/44 - 2s - loss: 0.0833 - accuracy: 0.9727 - val_loss: 0.4703 - val_accuracy: 0.8989
Epoch 98/100
44/44 - 2s - loss: 0.0740 - accuracy: 0.9761 - val_loss: 0.4703 - val_accuracy: 0.9004
Epoch 99/100
44/44 - 2s - loss: 0.0758 - accuracy: 0.9744 - val_loss: 0.4655 - val_accuracy: 0.9011
Epoch 100/100
44/44 - 2s - loss: 0.0723 - accuracy: 0.9763 - val_loss: 0.4602 - val_accuracy: 0.9036
```

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