Computer Vision Final Project Report

ZHANG CHENG (張　程) 5119F058

1. Task Description

My project is 2019 PRMU challenge on old Japanese character Recognition. In other word, Recognizing successive three characters in old Japanese documents, and output Unicode of a set of three characters. See pictures below to check details.
Categories: about 50 of KANA (Japanese alphabets), called "Kuzushiji". Not including KANJI (Chinese characters).

Fig 1. Kuzushi sample

2. Work before presentation

First, looking at dataset is very necessary. There are about 400,000 kana images, 120.000 3-character images and 16,000 test images. And the labels are Unicode.
Text recognition contains two parts, text localization and text classification. But actually there are no location information.
So the most intuitive method is split the image into three characters and send them to single character recognition system.

2.1 Split method

2.1.1 split images

Here I just use horizontal projection to split images. From Fig.2, we count the black pixels.

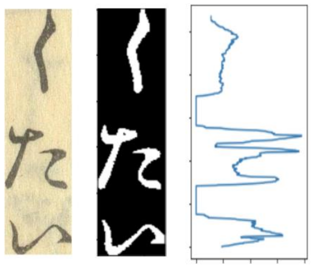


Fig 2. Horizontal Projection

Firstly, we preprocess the images such like binarization and Dilation. We check out the parts that the number of black pixel is less that 5, and find the middle point of this range. This is the basic idea. Let's see the experiments.

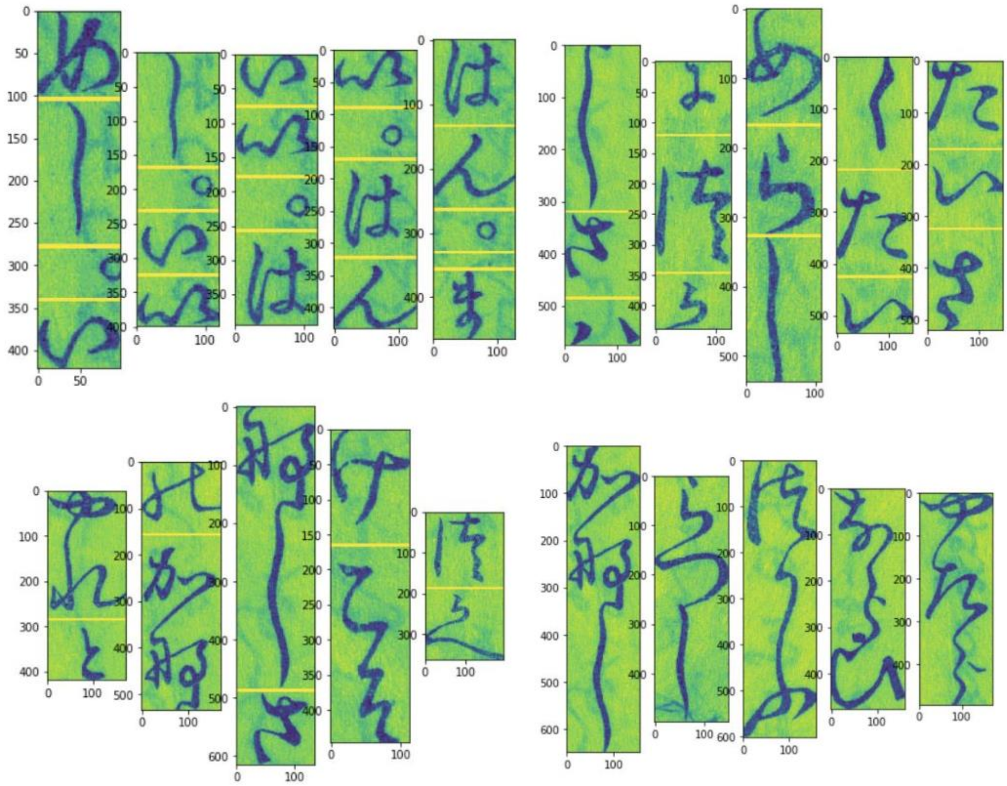


Fig 3. Split experiments

There are four parts. 0 line, 1 line, 2 lines, 3 more lines. For the 0 line images, I just use 1-third line and 2-third line. For the 1 line images, I split the remaining part into the same length. 2 lines images are good. So now I must deal with 3 more lines images.

See Fig 4 to check the distribution of 2,000 images.

seps3more 23
seps2 193
seps1 1210
seps0 574

Fig 4. Split distribution

For the 3 more lines images, firstly combine the close lines. Then ensure that there is at most one line in up part and down part. Like Fig 5.

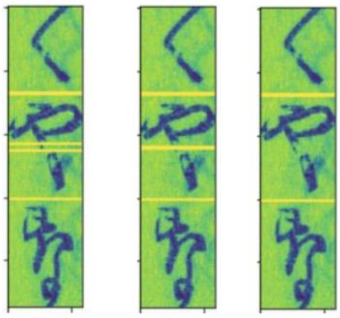


Fig 5. Solve 3 more lines

2.1.2 Train single character CNN models

The model is very simple. Because in my opinion, it's like a handwritten number recognition. This is the net part.

```
class Net(nn.Module):
        def __init__(self):
                super(Net, self).__init__()
                self.conv1 = nn.Conv2d(1, 6, 5)
                self.pool = nn.MaxPool2d(2, 2)
                self.conv2 = nn.Conv2d(6, 16, 5)
                self.fc1 = nn.Linear(16 * 13 * 13, 512)
                self.fc2 = nn.Linear(512, 256)
                self.fc3 = nn.Linear(256, 48)
```

And I do some transformation like grayscale and resize.

```
    kana_transform = tv.transforms.Compose([
        tv.transforms.Grayscale(),
        tv.transforms.Resize((64, 64)),
        tv.transforms.ToTensor(),
        tv.transforms.Normalize(*kana_norm)
    ])
```

The final Accuracy is about 95%.

2.1.3 Adjust results by frequency pattern

And we can adjust results by the word relevance. We get a frequency dictionary like this.

```
{
'し': 0.05546,
'と': 0.0472,
'か': 0.04635,
 'い': 0.04102,
 'に': 0.03922,
 'て': 0.03911,
'り': 0.03883,
'な': 0.03688,
'も': 0.03209,
……
}
```

```
{
('と', 'い'): 0.01532,
('い', 'ふ'): 0.01268,
('し', 'て'): 0.00885,
('に', 'て'): 0.00775,
('も', 'の'): 0.00693,
('な', 'り'): 0.00677,
('か', 'ら'): 0.0061,
('や', 'う'): 0.00599,
('に', 'し'): 0.00462,
……
}
```

Firstly we get top 3 results and their probabilities. If the top 1's probability is less than 0.3, then we decide final results based on the frequency and probabilities. The formula is like,

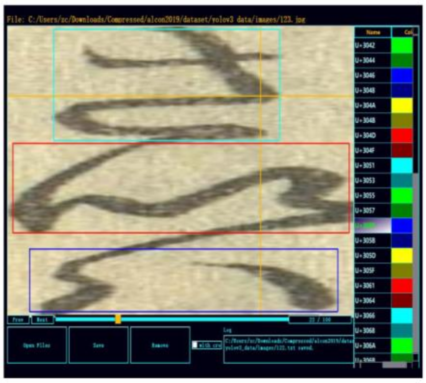
$$\mathrm{Max}(5 * \mathrm{bigram\_prob}\ +\ 0.3 * \mathrm{prob}\ +\ 1 * \mathrm{unigram\_prob})$$

2.1.4 Results analysis

The 3-character accuracy is about 28%. The 1-character accuracy is about 63%. So we can know that split error is about 32%. And the score on CodaLab is 0.101

2.2 Yolov3 method

We know that there is no location data. But we can label it by ourselves. And I labeled 100 samples here. See Fig 6.



Class	x_center	y_center	width	height
40	0.602734	0.160811	0.422656	0.308108
25	0.510938	0.511486	0.909375	0.228378
17	0.498437	0.843243	0.995313	0.300000

Here I just labeled 100 samples.

Fig 6. Manually labeled data

And I use Yolov3 model to train and recognize. There are some good results and bad results. We can infer that we will get good results by large datasets.

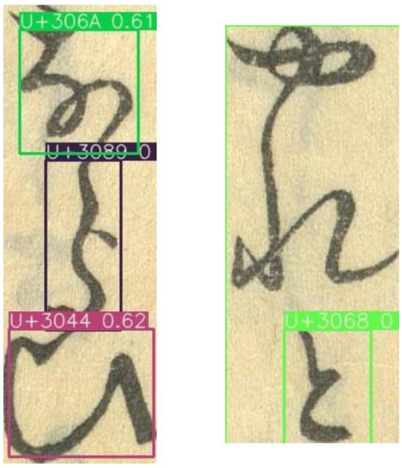


Fig 7. Yolov3 predict results

3. Work after presentation

I also tried two methods to deal with this challenge.

3.1 Multilabel classification

Actually we could thought it as a multilabel classification problem. We have 48 classes and we should predict 3 labels. So I read the paper. And I use a very simple method, that is, label ranking. The input is like [0,0,1, ···, 1,0,0,1]. In the last layer, we use sigmoid other than softmax. The reason is that we actually make a 48 binary classifier. So the loss is binary_crossentropy. And the number of train samples is 10,000. Fig 8 is the process.

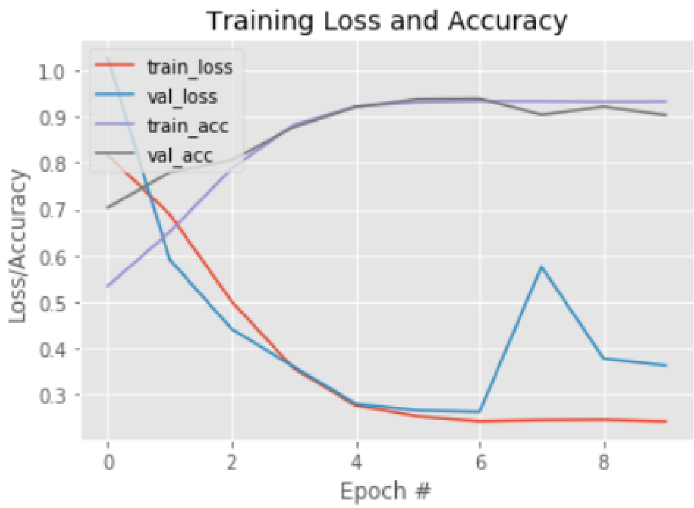


Fig 8. Train Loss and Accuracy

And we can get the top 3 probability labels. But the problem is, I can't decide their orders. For example, we can get something like Fig 9. But the order is not clear. So I have to consider other ideas.

```
U+3068: 79.83%
U+3064: 73.08%
U+3042: 30.63%
```

Fig 9. Multilabel Label Rank Results

3.2 Three classifiers

So I tried to make three classifiers for each character. I mean for the same input with different labels. Here I use Resnet50 model. And I trained them for about 9 hours. Each classifier has 95% accuracy. And finally I got 0.7007 score on CodaLab. That's a big progress compared with 0.101 (split method)!! Check the details below.

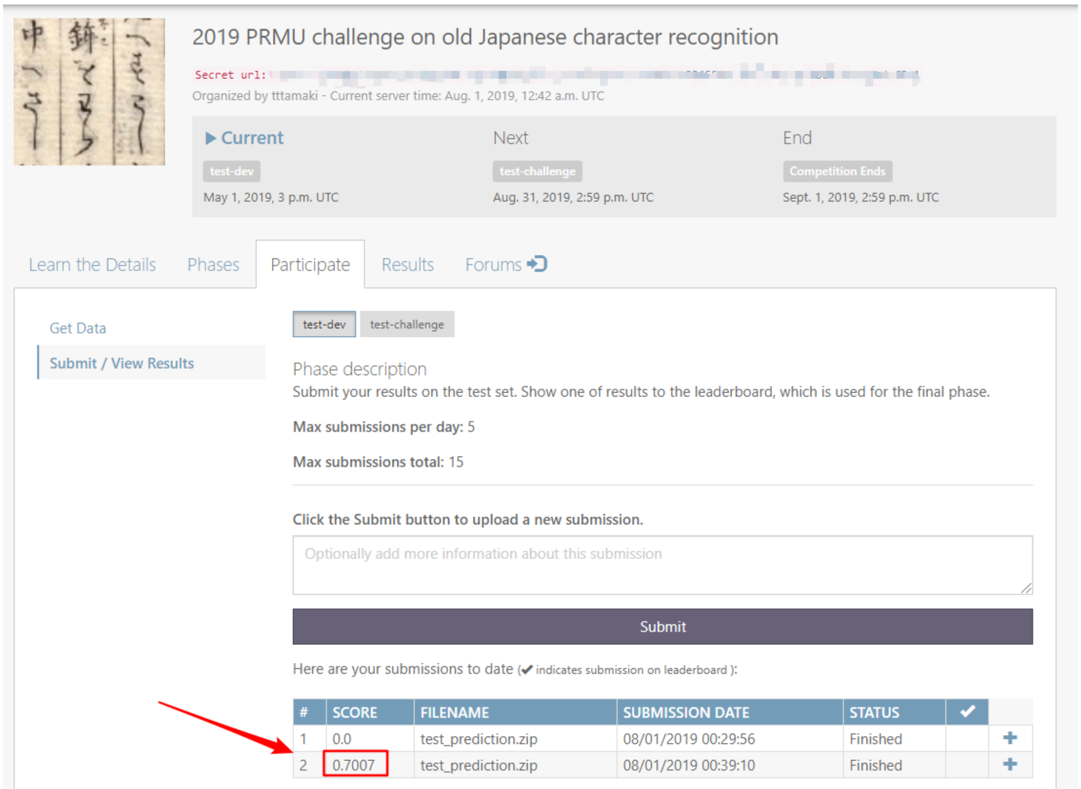


Fig 10. Three classifiers Results

4. Conclusion

I tried a lot of method to deal with it. And the split method is not good at this task. The three classifiers method has a great performance (0.7007). But it's also not a method to predict the whole 3 characters. So we should learn that it's better to treat it as a whole image. And if we have more location datasets, we can get higher accuracy by Yolov3.

I want to use Chen's method. But I am new to deep learning. I don't know how to make 3 classifiers in the last layer. This summer I will take a Deep Learning course on Coursera. After that, I will have a systematic understanding of deep learning. Then I will try to research it.