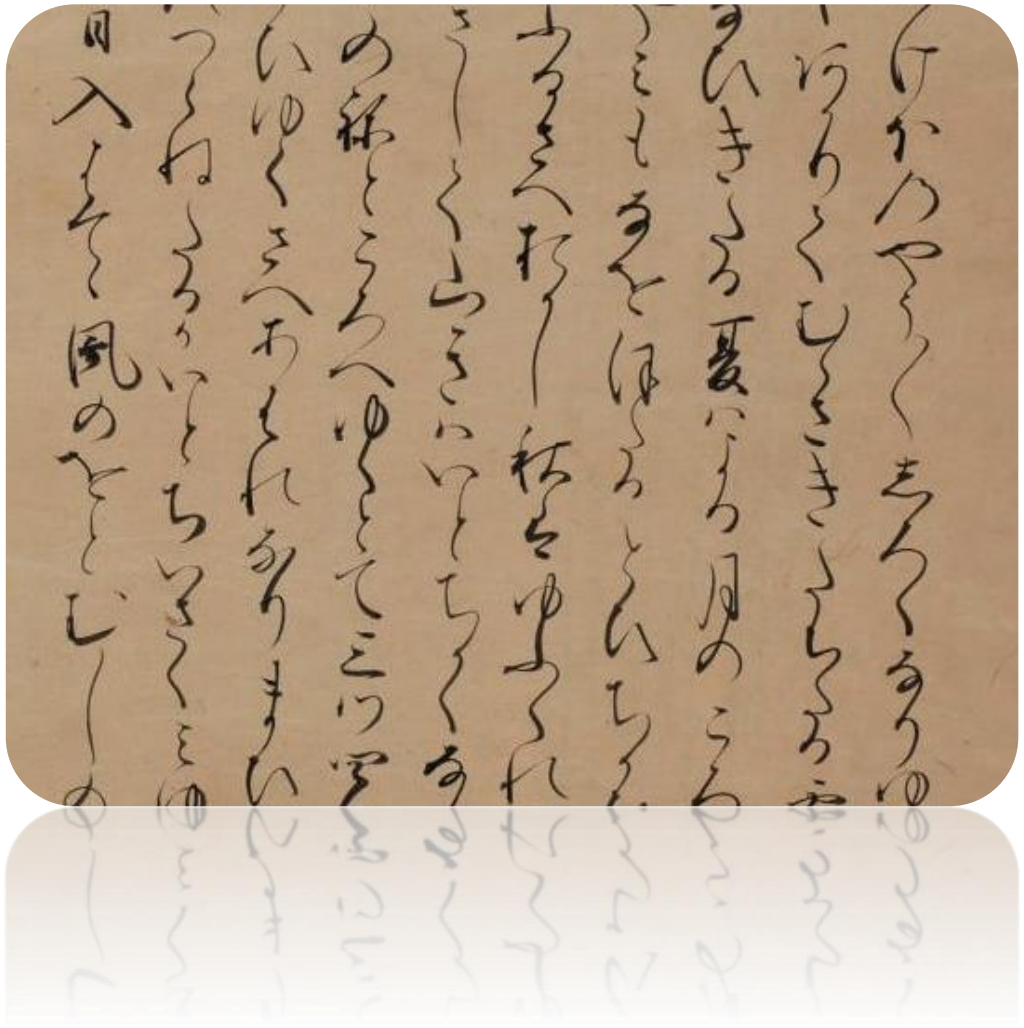
Kuzushi Recognition

ZHANG CHENG 5119F058

2019/07/10



Contents

1. Background
2. Intuitive Method (split)
3. Intuitive Result
4. End to End Method
5. Yolov3 Result
6. Future Work

1. Background

Text recognition

1. Text localization(detection)
2. Text classification

What we have:

1. Train images (3 characters) and labels
2. 1 character and labels

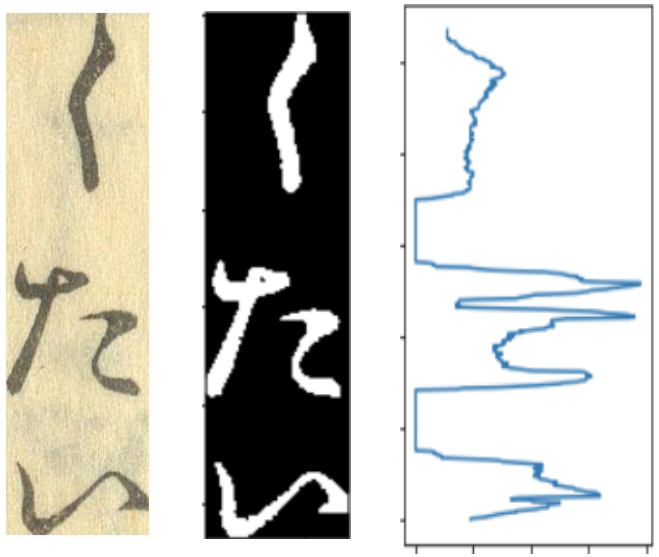
No location data!!!

2. Intuitive Method

Steps:

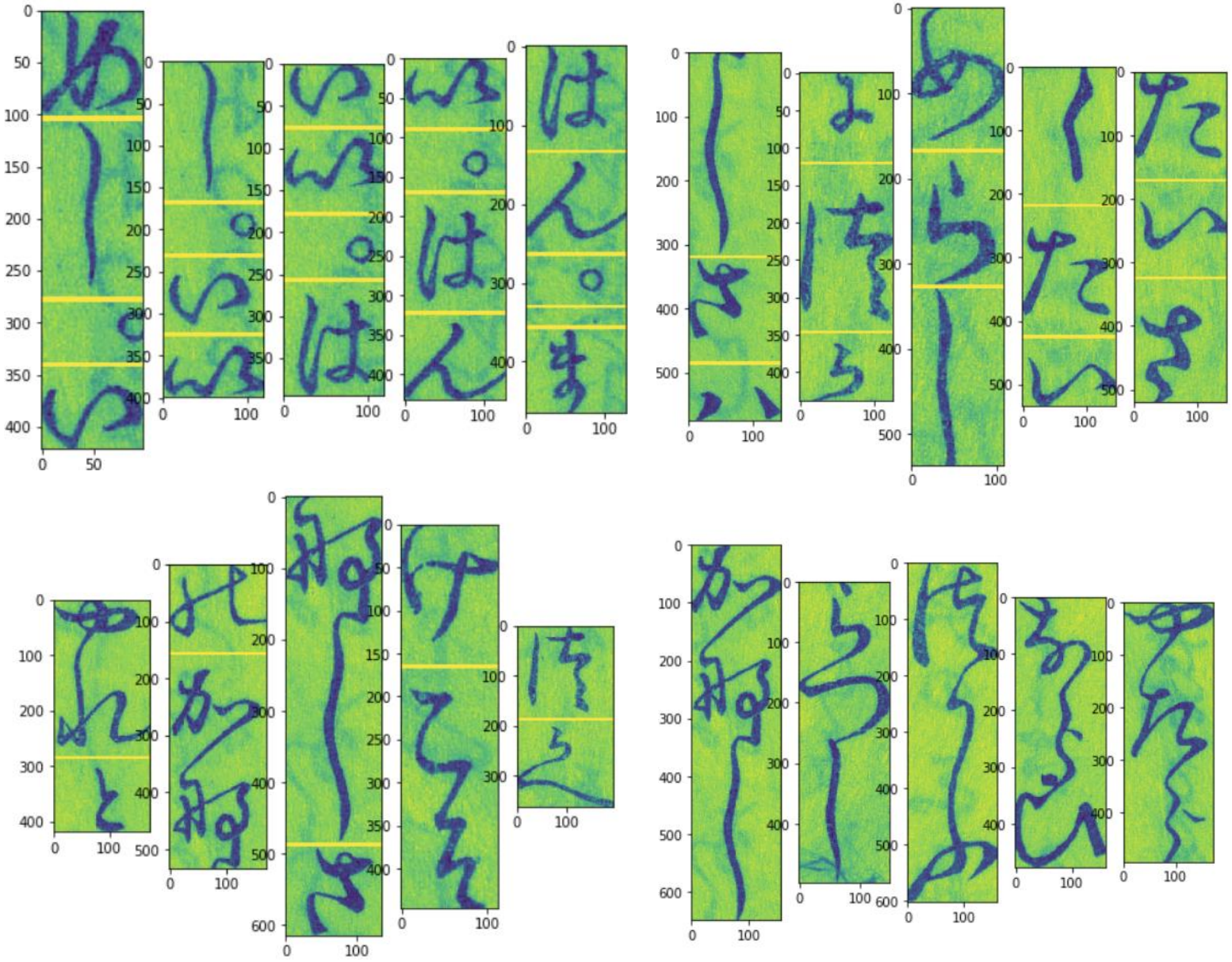
1. Split train images into 3 characters
2. Train a classification Model
3. Recognize them separately
4. Adjust result by ngram and probability analysis

2.1 Split images

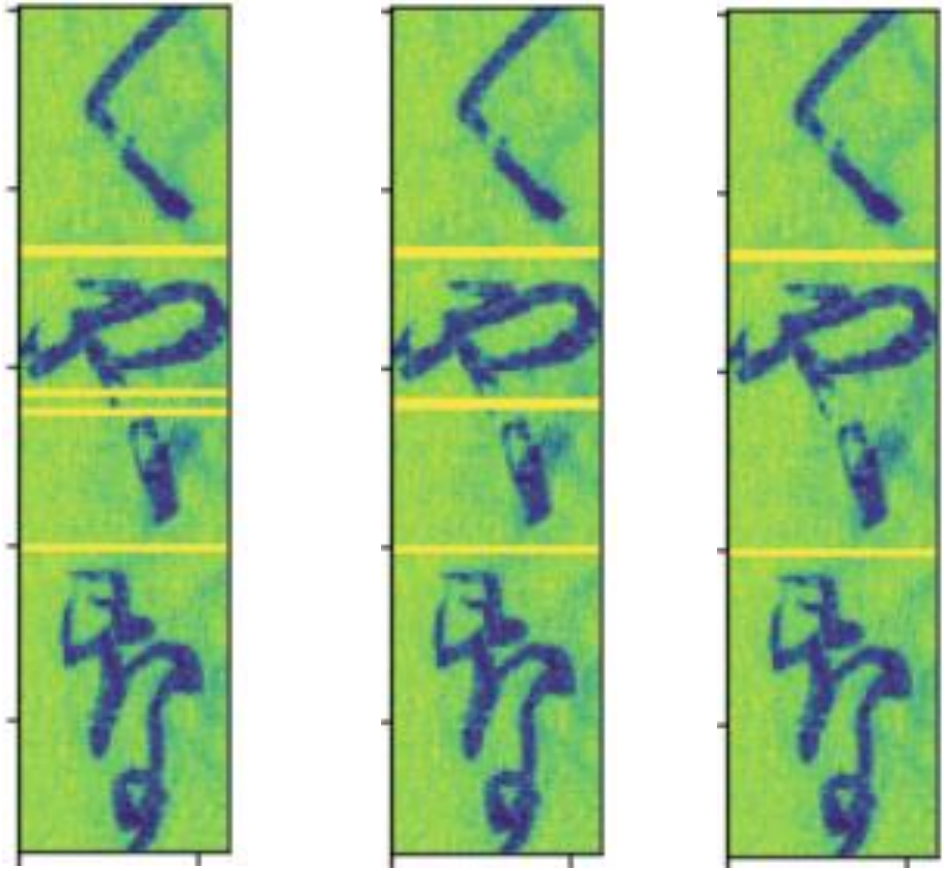


Binary, Dilation

seps3more 23
seps2 193
seps1 1210
seps0 574



2.1 Split images



Environment:

Pytorch 1.1.0

Server: google colab

2.2 Train a Model(CNN)

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

```
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)
```

Epochs = 5
Final loss = 0.1
Accuracy = 95~97%

2.3 Adjust result by ngram frequency analysis

```
{
'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,
……
}
```

Top 3 possible results and probabilities

↓

If top 1 prob < 0.3

↓

Max(5*bigram_prob + 0.3*prob + 1*unigram_prob)

3. Intuitive Results

Single character CNN model accuracy: 95%

3 characters accuracy on train datasets: 28%

1 character accuracy on train datasets: 63%

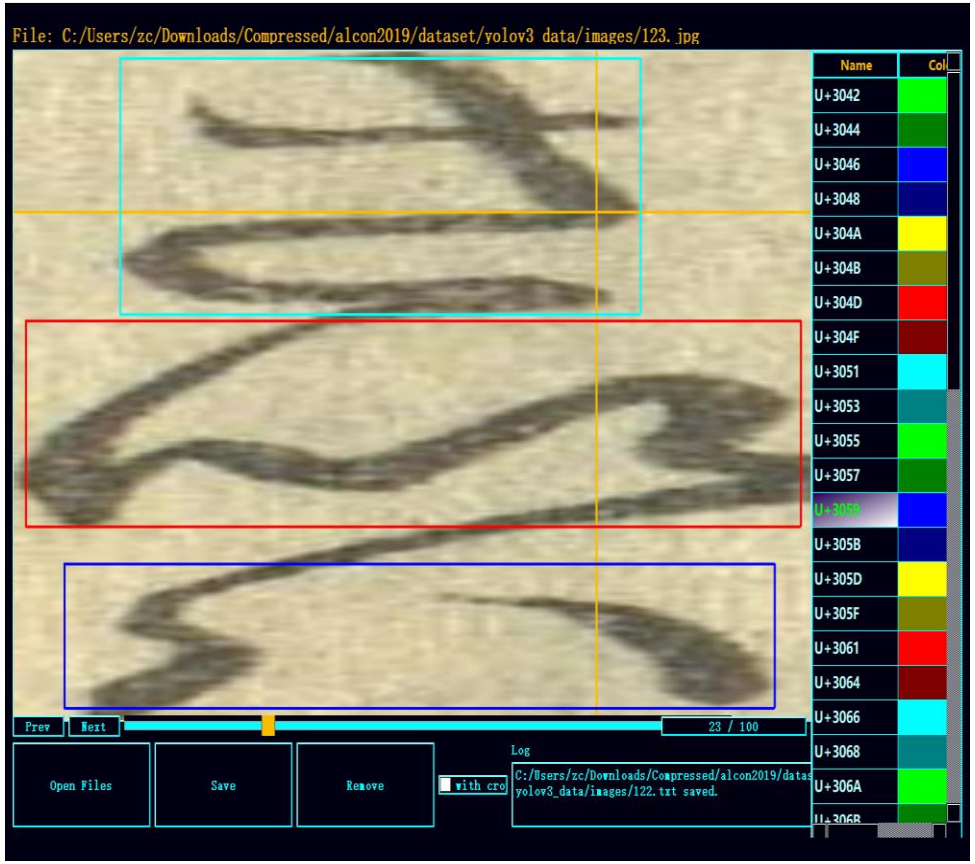
0.63 * 0.63 * 0.63 = 0.25

Split error: 95 – 63 = 32%

Score on CodaLab(3000): 0.101

4. End to End Method

Create location data by hand[1]!

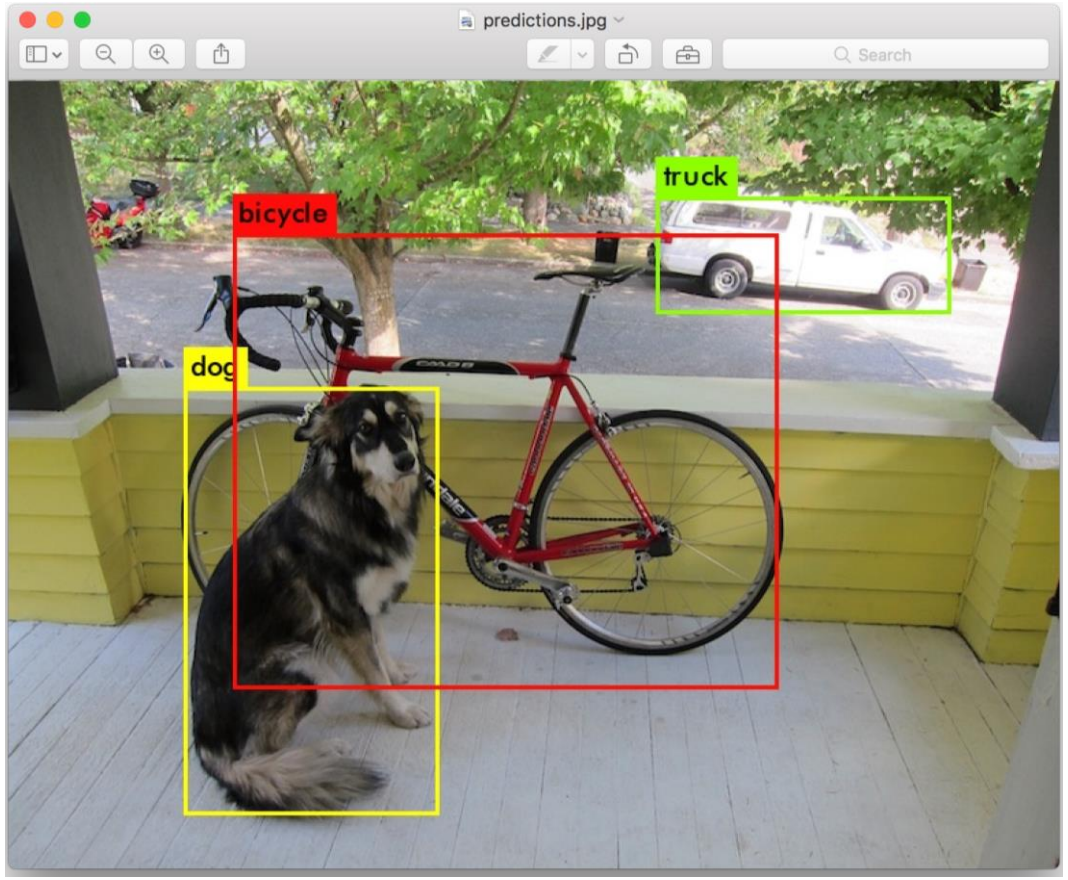


Class	x_center	y_center	width	height
40	0.602734	0.160811	0.422656	0.308108
25	0.510938	0.511486	0.909375	0.228378
17	0.498437	0.843243	0.995313	0.300000

Here I just labeled 100 samples.

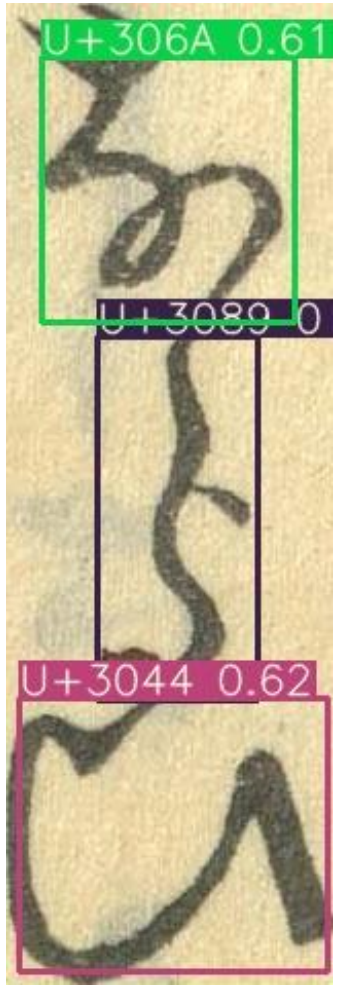
4. End to End Method

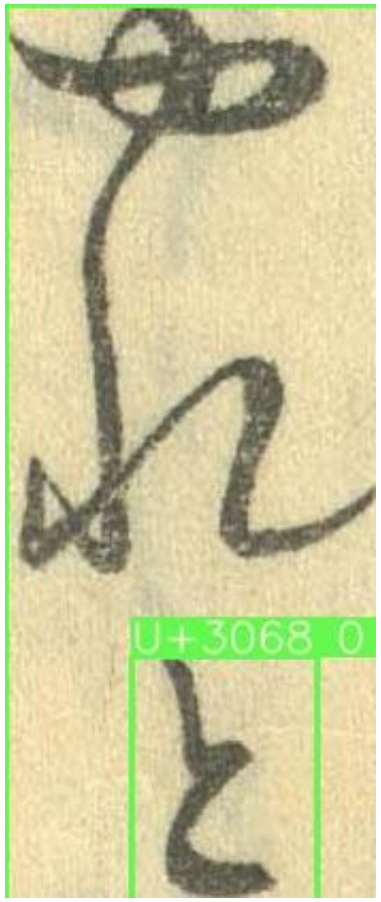
Yolov3 is a powerful object detection model, especially for real-time[2].



Here I use a pytorch version[3] of it, since I have to use Colab to train it.

5. Yolov3 Result





Large dataset

Good result ?

6. Future Work

Try SVHN[4]

4. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks

Reference

1. Yolov3 label tool, https://github.com/developer0hye/Yolo_Label, 2019/07/07
2. Yolov3, https://pjreddie.com/darknet/yolo/, 2019/07/07
3. Yolov3 pytorch version, https://github.com/ultralytics/yolov3, 2019/07/07
4. SVHN paper, http://arxiv.org/pdf/1312.6082.pdf, 2019/07/07
5. SVHN understanding(Chinese version), http://nladuo.github.io/2018/10/20/%E8%AF%BB%E8%AE%BA%E6%96%87Multi-digit-Number-Recognition-from-Street-View-Imagery-using-Deep-Convolutional-Neural-Networks/, 2019/07/07