Automatic Number-Plate Recognition

吳易倫

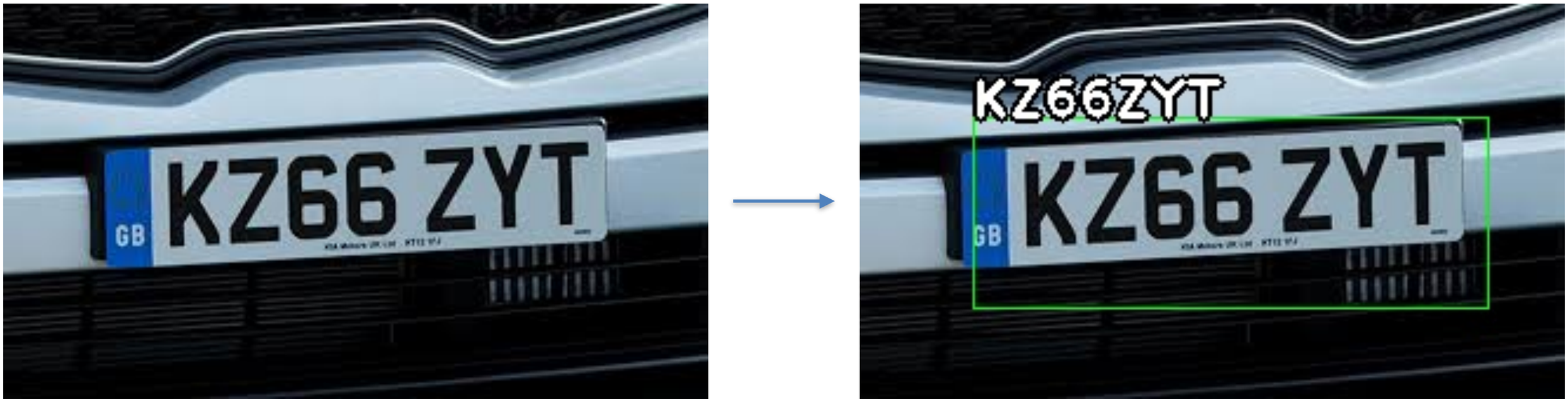
Original project: http://matthewearl.github.io/2016/05/06/cnn-anpr/

Outline

- Problem
- Dataset
- Package Introduction
- Tasks
  - Data Preprocessing
  - Build Deep Neural Network
  - Run the Operation
  - Visualization

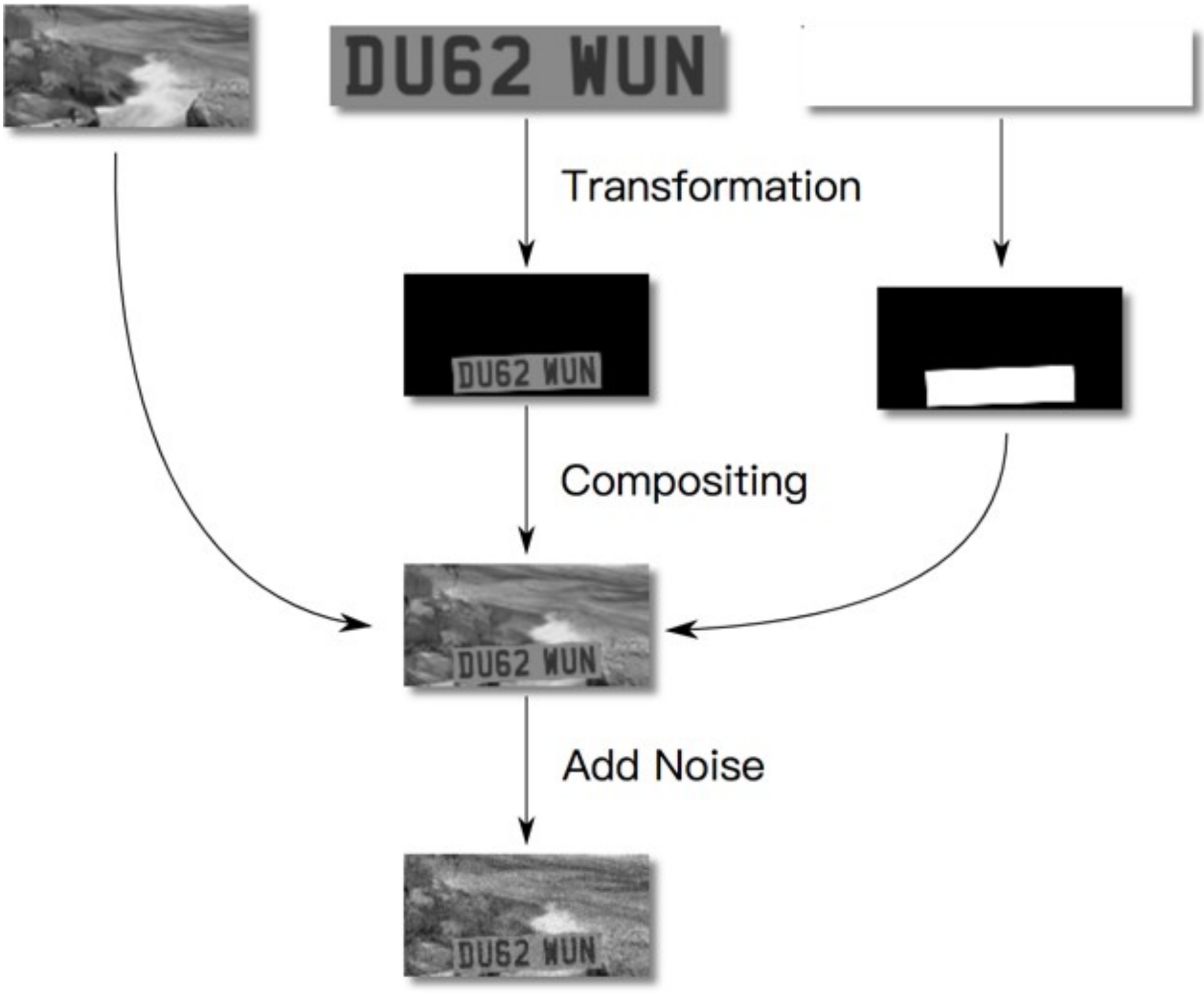
Problem

- Give an image. Find out the region of number plate and recognize characters in it.
- There are 7 characters in a number plate.



Dataset

Synthesizing images



Package Introduction

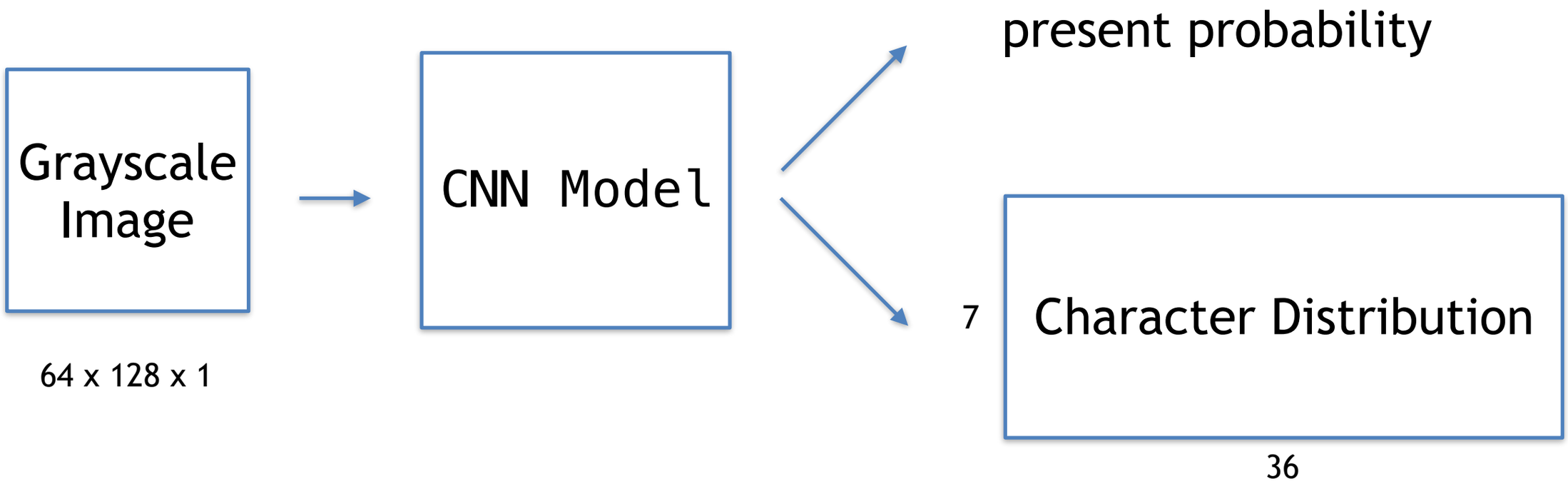
- TensorFlow 1.10.0
  - An end-to-end open source platform for machine learning
- tqdm
  - A fast, extensible **progress bar** for Python and CLI
- numpy
  - A powerful N-dimensional array object
  - Sophisticated functions
- OpenCV
  - OpenCV is an open source computer vision and machine learning software library.

How to Model Problem

How to Model Problem

How to Model Problem

- Input
  - Image size = 64 x 128
  - 7 characters in a number plate
  - 36 possible characters
- Output
  - The probability of a number plate presented in the input image.
  - The probability of the digit in each position, ie. for each of the 7 possible positions



How to Model Problem

Present

Too small

Too large

Not present

Truncated

Tasks

Get source code

- SSH into your server and find any good place

```
$ ssh nctuece@[your server ip]
```

- Clone the source code from GitHub

```
$ git clone https://github.com/w86763777/HCC-ML-LAB
$ cd HCC-ML-LAB/LAB3
```

Setup Environment

- Create virtual environment
  - Install pip (skip)

```
$ sudo apt-get install python3-pip
```

  - Install virtualenv using pip3 (skip)

```
$ sudo pip3 install virtualenv
```

- Create a virtual environment

  At the root of your project (i.e. HCC-ML-LAB/LAB3)

```
$ virtualenv -p python3 venv
...
done.
```

  - -p specify python interpreter
  - “venv” is your environment name

Setup Environment

- Activate virtual environment

```
$ source venv/bin/activate
(venv) $
```

- Install packages at a time

```
(venv) $ pip install -r requirements.txt
...
Successfully installed ...
```

- Leave virtual environment

```
(venv) $ deactivate
$
```

1-1 Data Preprocessing

- Dataset
    1. https://drive.google.com/open?id=1vkkLO49h6Gk_V4bVAAUJixURWjfxPm4e
    2. Use download script to facilitate the progress. Just run

        ```
        (venv) $ python download.py
        ```

    - You should see

```
(venv) $ ls
anpr  background.tar.gz  requirements.txt  test1.jpg  test2.jpg
uk_number_plate.zip  venv  weights.npz
```

1-1 Data Preprocessing

- Read image

```
# TODO: Checkpoint 1
# 1. Convert image from BGR to GRAY scale
# 2. Divide image by 255.0
image = cv2.imread(args.input_image)
image_gray = cv2.cvtColor(???, code=???)
image_gray = ???
```

cv2.cvtColor

Checkpoint1:

```
$ python -m anpr.cli.detect test1.jpg test1.out.jpg weights.npz
...
(168, 300) 0.0 1.0
...
```

Project Structure

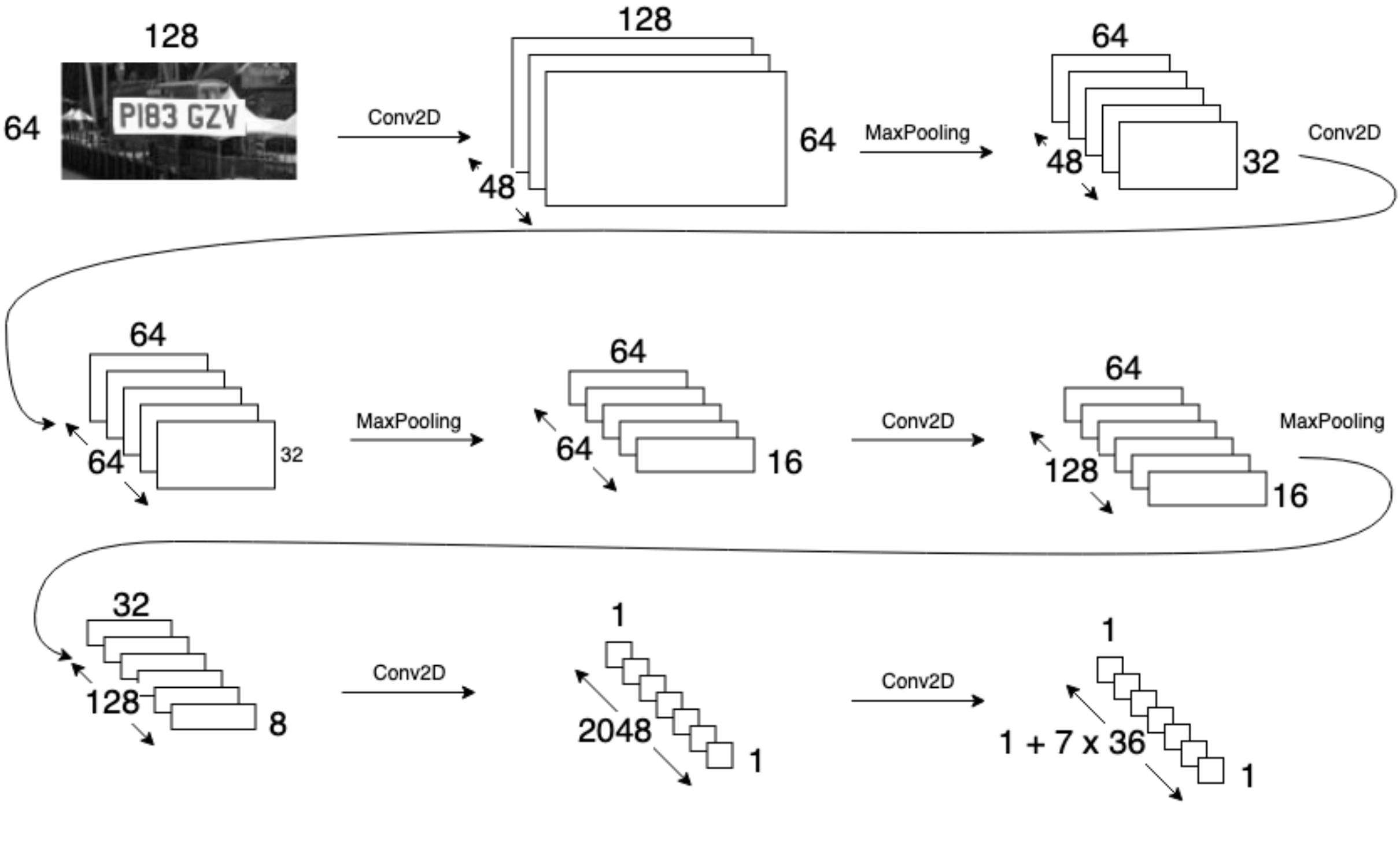
```
LAB3
├── anpr
│   ├── cli
│   │   ├── detect.py
│   │   ├── download.py
│   │   ├── gen.py
│   │   └── train.py
│   ├── data.py
│   ├── model.py
│   └── utils.py
├── test1.jpg
├── test2.jpg
...
```

- Files in cli (command-line interface)
  - Should not be imported
  - Include main statement
    `if __name__ == '__main__'`
- Files not in cli
  - Can be imported
  - Not include main statement

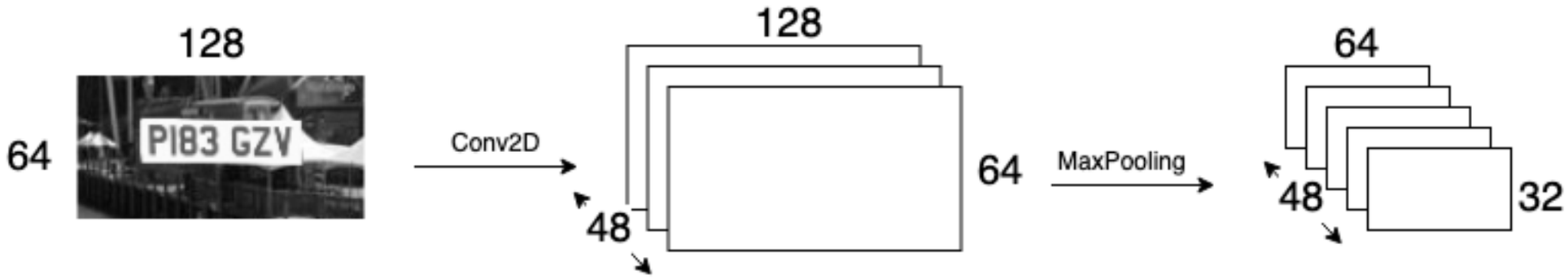
```
from anpr.utils import CHARS, IMAGE_SIZE, CODE_LENGTH
from anpr.model import create_valid_model, load_model
```

```
$ python -m anpr.cli.detect_ans test1.jpg test1.out.jpg weights.npz
```

1-2 Model


128
64
P183 GZV
Conv2D
128
48
64
MaxPooling
64
48
32
Conv2D
64
64
32
MaxPooling
64
64
16
Conv2D
64
128
16
MaxPooling
32
128
8
Conv2D
1
2048
1
Conv2D
1
1 + 7 x 36
1

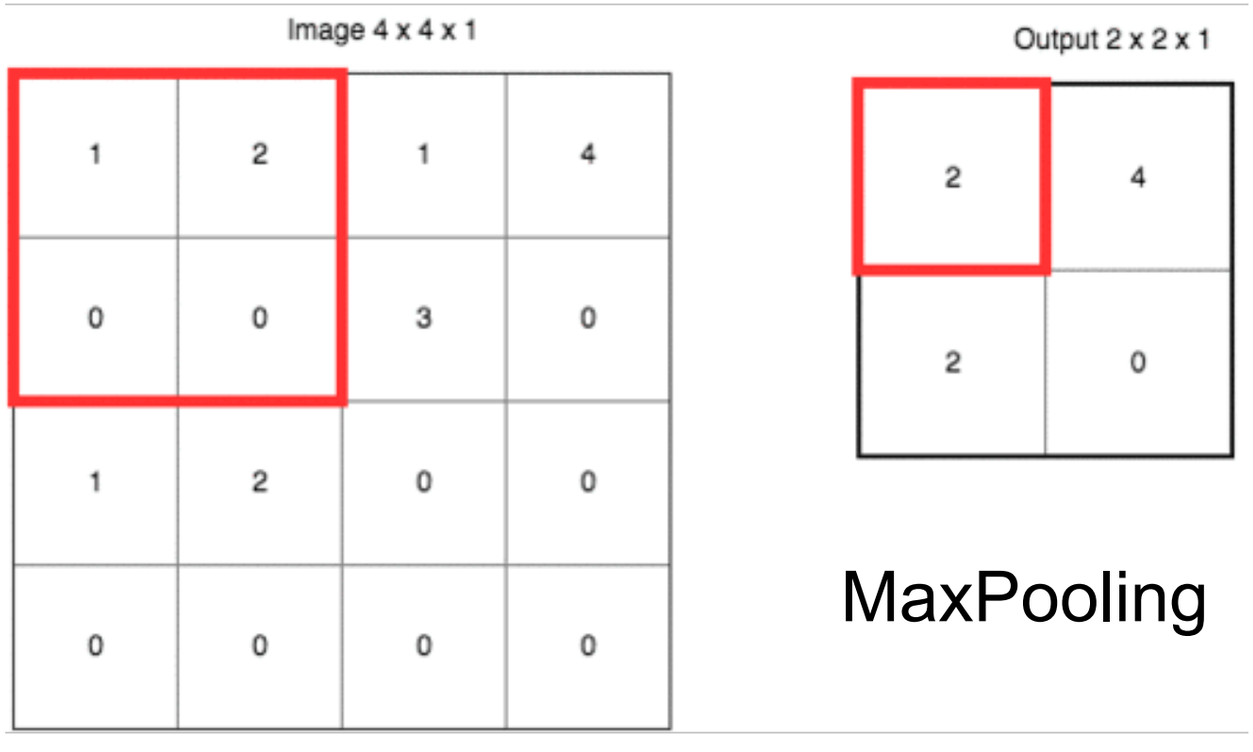
1-2 Model




```
inputs = tf.placeholder(tf.float32, [None, None, None])
...
x = Conv2D(filters=48, kernel_size=(5, 5), padding='same')(x)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='same')(x)
```

[tf.keras.layers.Conv2D]

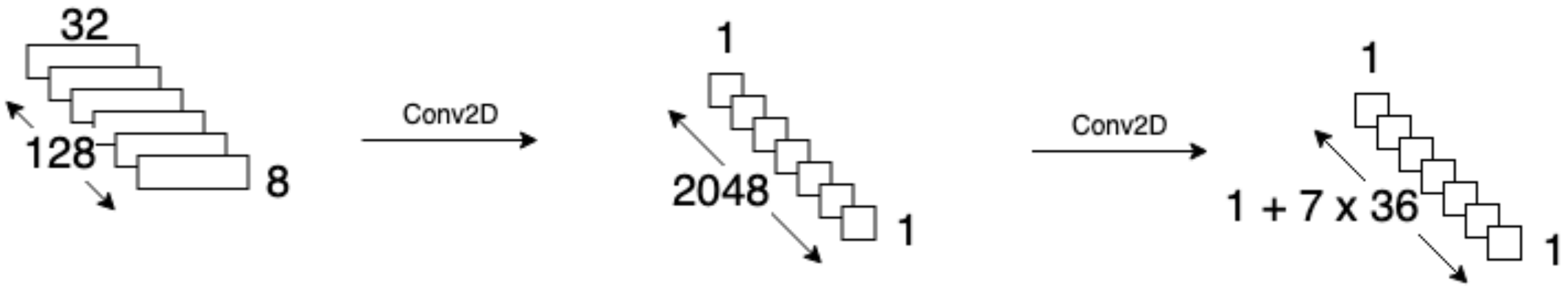
[tf.keras.layers.MaxPool2D]



MaxPooling

https://ithelp.ithome.com.tw/articles/10187424?sc=rss.qu

1-2 Model




```
# x.shape [8, 32, 128]
x = Conv2D(filters=2048, kernel_size=(8, 32), padding='valid')(x)
x = ReLU()(x)
labels = Conv2D(1 + CODE_LENGTH * len(CHARS), (1, 1), padding='same')(x)
```

1-2 Model

```
# TODO: Checkpoint2
# 1. Expand dimension of inputs to meet the specifications of Conv2D
# Hint: target shape is [height, width, channel]
inputs = tf.placeholder(tf.float32, [None, None, None])
# x = tf.expand_dims(inputs, axis=???)
# or
# x = tf.reshape(inputs, shape=[???])
```

- tf.expand_dims

  [128, 128, 3] -> [128, 128, 3, 1]

  [128, 128, 3] -> [128, 1, 128, 3]

- tf.reshape

  [32, 7, 36] -> [32, 252]

  [32, 128, 128] -> [32, 128, 128, 1]

1-2 Model

```
# TODO: Checkpoint2
# 1. Expand dimension of inputs to meet the specifications of Conv2D
# Hint: target shape is [height, width, channel]
inputs = tf.placeholder(tf.float32, [None, None, None])
# x = tf.expand_dims(inputs, axis=???)
# or
# x = tf.reshape(inputs, shape=[???])
```

Checkpoint2:

Show test1.out.jpg

```
(venv) $ python -m anpr.cli.detect_ans test1.jpg test1.out.jpg weights.npz
```

Pull files from remote device to local machine.

```
$ scp [username]@[ip]:~/HCC-ML-LAB/LAB3/test1.out.jpg ./
```

1-3 Train

```
(venv) $ mkdir data
(venv) $ tar -xf background.tar.gz -C data/
(venv) $ unzip uk_number_plate.zip -d fonts/
```

```
data/
└── background
    ├── Places365_val_00000001.jpg
    ├── Places365_val_00000002.jpg
    ├── Places365_val_00000003.jpg
    ├── Places365_val_00000004.jpg
    ├── Places365_val_00000005.jpg
    ├── ...
    ├── Places365_val_00000099.jpg
    └── Places365_val_00000100.jpg
fonts
└── UKNumberPlate.ttf
```

1-3 Train

```
(venv) $ python -m anpr.cli.gen 100
```

```
data/
├── background
│   ├── Places365_val_00000001.jpg
│   ├── Places365_val_00000002.jpg
│   ├── Places365_val_00000003.jpg
│   ├── ...
└── test
    ├── 00000000_PG55UCW_0.jpg
    ├── 00000001_MT14JLO_1.jpg
    ├── ...
```

1-3 Train

```
(venv) $ python -m anpr.cli.train
```

Checkpoint3

Show progress bar

```
(venv) nctuece@nctuece:~/HCC-ML-LAB/LAB3$ python -m anpr.cli.train
2019-04-20 16:10:28.374185: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that
this TensorFlow binary was not compiled to use: AVX2 FMA
2019-04-20 16:10:28.474236: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:897] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2019-04-20 16:10:28.474836: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1405] Found device 0 with properties:
name: GeForce GTX 1080 major: 6 minor: 1 memoryClockRate(GHz): 1.7335
pciBusID: 0000:01:00.0
totalMemory: 7.92GiB freeMemory: 7.73GiB
2019-04-20 16:10:28.474851: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1484] Adding visible gpu devices: 0
2019-04-20 16:10:28.671022: I tensorflow/core/common_runtime/gpu/gpu_device.cc:965] Device interconnect StreamExecutor
with strength 1 edge matrix:
2019-04-20 16:10:28.671057: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971]      0
2019-04-20 16:10:28.671063: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] 0:   N
2019-04-20 16:10:28.671258: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1097] Created TensorFlow device (/
job:localhost/replica:0/task:0/device:GPU:0 with 7458 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1080, pci
bus id: 0000:01:00.0, compute capability: 6.1)
  2%|▌                    | 195/10000 [00:42<34:11,  4.78it/s, digits_loss=1.5232, loss=2.2116, presence_loss=0.6883]
```