Tsinghua Deep Learning Summer School

Captcha Recognition

100 pts total.

Warm up: Learn about Keras.

Keras is a high-level deep learning interface running (mainly) on top of TensorFlow. It allows for easier and faster prototyping than core TensorFlow, and friendly for students to turn what they learn to code.

Keras has direct mapping from theory to code. Take the LeNet for an example:

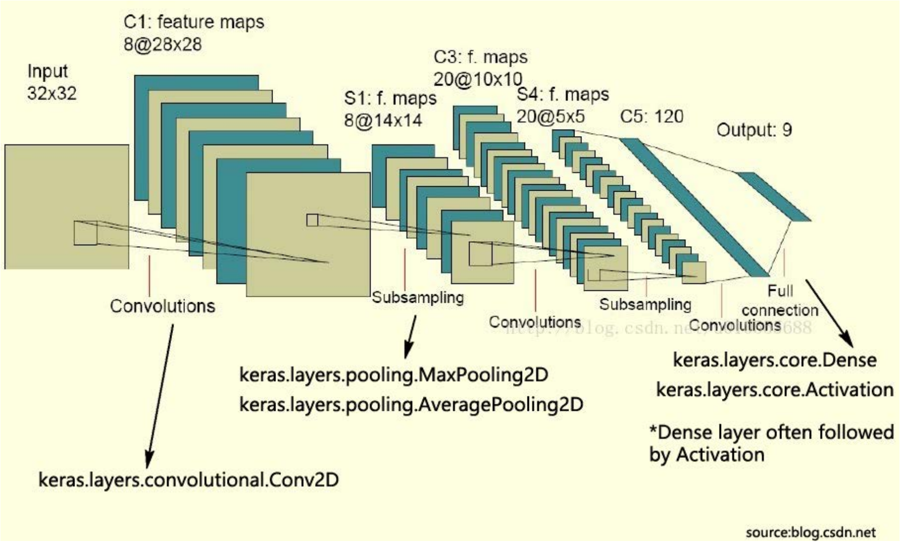

Figure 1: LeNet

(There are many types of activation you can find in the Keras Documents such as softmax, relu, tanh, etc.)

The picture above shows some layers provided by Keras, it's quite easy to understand what they mean via their name.

Keras has already been installed with tensorflow backend on the cloud machine. You can read the documentation on `https://keras.io/` to learn how to program using Keras.

Task 1: Code Completion (Captcha Recognition) (50 pts)

Captcha Recognition is similar to MNIST Recognition, it takes a captcha as its input, and tell what numbers it contains. To simplify the task, the captcha contains exactly 4 twisty numbers. (Check `captcha/data` folder to see the inputs, Figure 2 is an example which contains 1164.)

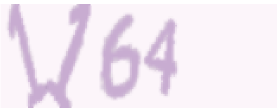

Figure 2: Captcha Example

You will get some Keras code (`captcha/lab-captcha.py`) with many blanks to fill in. And you need to complete the code to build the model according to the model image shown in Figure 3.

In each box, the left shows which layer to use and the right shows the input and output size, you need to calculate the configuration according to these sizes in order to complete the code.

Take the first Conv2D to explain what the input and output size means. For Input, 60 is height of image, 160 is width, 3 is channel. They are the original size of images we feed into the network. (You can ignore None, it is a placeholder of batch size.) The output size is 56 in height, 156 in width and 32 channels because we use 32 kernels/filters, each kernel size is 5 × 5. (You should know why the kernel size is 5 × 5? It was discussed in class, or you can refer to `http://cs231n.github.io/convolutional-networks/#conv` for explanation.)

Note: You will find the blanks under **TODO** in the file. You can add any code to help you debug, but you are **NOT** allowed to change the model or any training and prediction logic, especially the random seed.

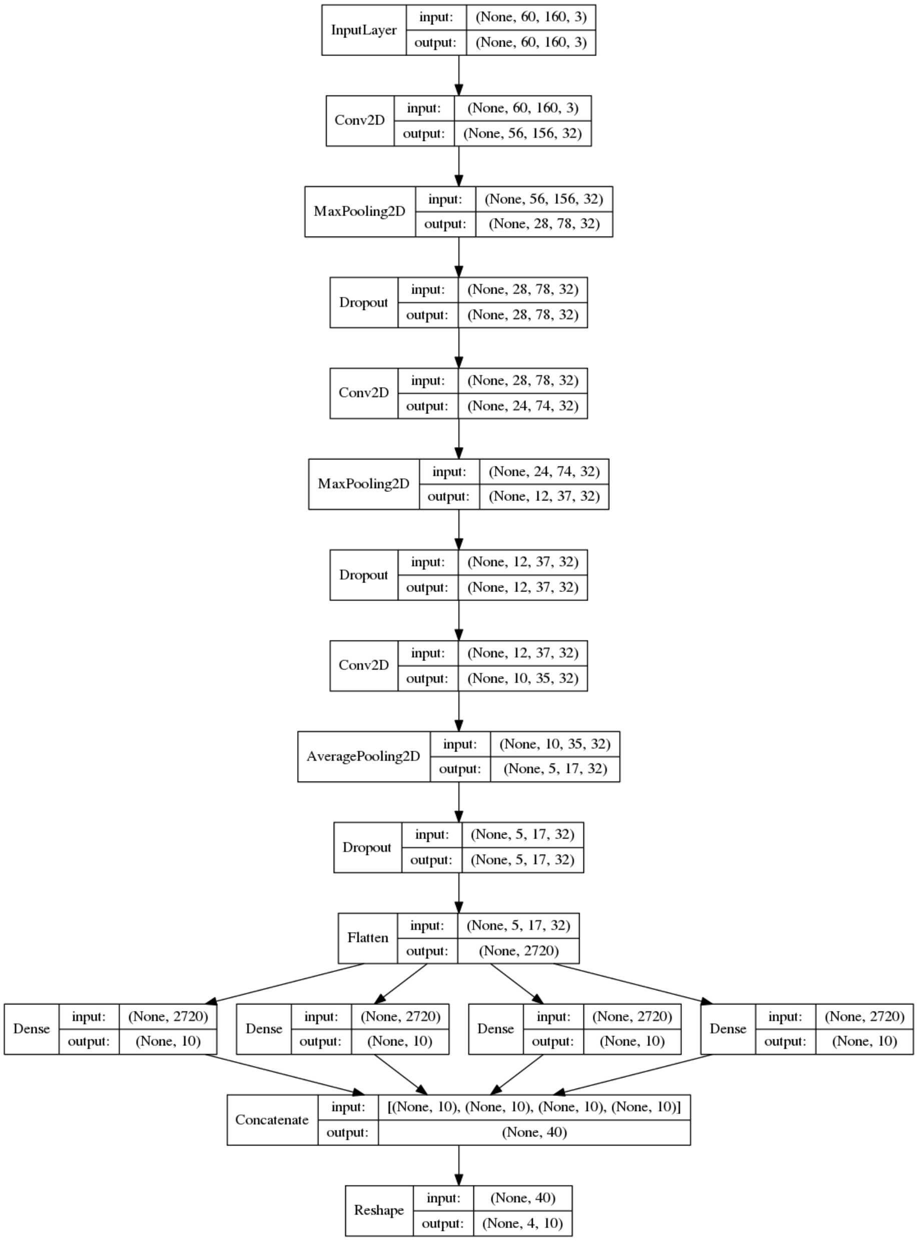
InputLayer
input: (None, 60, 160, 3)
output: (None, 60, 160, 3)
Conv2D
input: (None, 60, 160, 3)
output: (None, 56, 156, 32)
MaxPooling2D
input: (None, 56, 156, 32)
output: (None, 28, 78, 32)
Dropout
input: (None, 28, 78, 32)
output: (None, 28, 78, 32)
Conv2D
input: (None, 28, 78, 32)
output: (None, 24, 74, 32)
MaxPooling2D
input: (None, 24, 74, 32)
output: (None, 12, 37, 32)
Dropout
input: (None, 12, 37, 32)
output: (None, 12, 37, 32)
Conv2D
input: (None, 12, 37, 32)
output: (None, 10, 35, 32)
AveragePooling2D
input: (None, 10, 35, 32)
output: (None, 5, 17, 32)
Dropout
input: (None, 5, 17, 32)
output: (None, 5, 17, 32)
Flatten
input: (None, 5, 17, 32)
output: (None, 2720)
Dense
input: (None, 2720)
output: (None, 10)
Dense
input: (None, 2720)
output: (None, 10)
Dense
input: (None, 2720)
output: (None, 10)
Dense
input: (None, 2720)
output: (None, 10)
Concatenate
input: [(None, 10), (None, 10), (None, 10), (None, 10)]
output: (None, 40)
Reshape
input: (None, 40)
output: (None, 4, 10)

Figure 3: Model

Task 2: Get higher accuracy (50 pts)

In the previous task, you finish a basic convolutional neural network to recognize captcha, but it doesn't work well enough. In this task, you are going to design your own network and try different parameters to obtain as high (validation) accuracy as possible on the captcha recognition task.

Final score will be calculated using

$$\text{score} = \left(\frac{\text{Your acc} - \text{baseline}(88\%)}{\text{Max acc among all students} - \text{baseline}(88\%)} \right)^{1.5} \times 50$$

Submission

If you finish both the tasks above, please report your result to TA, you need to run the python file, show the accuracy output and we may ask what improvement you do.