Machine Learning Engineer Nanodegree

Capstone Proposal

Domain Background

I believe everyone hates the CAPTCHA, the text images mixed with text, we have to type in before accessing webs or logining accounts. The purpose of CAPTCHA is make sure we are real person and prevent the computer from automatcally logining in. My inspiration is from the MNINST and SVHN datasets.The concepts are the same.

Although more and more websites change their defend system form CAPTCHA to others, lots of websites,including government's sites, still use the CAPTHA. By this capstone, I want to reveal that even beginner as I can easily solve CAPTCHA. it is not only easy to machine to solve, but it annoy human users a lot. Baes on the study :"How Good are Humans at Solving CAPTCHAs? A Large Scale Evaluation". it revealed that it is more effective for an attacker touse Mechanical Turk to solve captchas than an underground service. chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://web.stanford.edu/~jurafsky/burszstein_2010_captcha.pdf (https://web.stanford.edu/~jurafsky/burszstein_2010_captcha.pdf)

Problem Statement

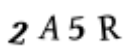
The problem is a classification of letters by computer vision. Use amounts of CAPTCHAs with various letters' fonts as input data. Becasuse the names of all image files are already labeled, the output results will be the files' names. For example, if input data is "2A2X", the correct output results will be the file's name "2A2X". Furthermore, for computation efficiency, I may split the input images into numbers(0~9) and letters(A~Z), then put each letter into the files with the same name.

Datasets and Inputs

To break the CAPTCHAs, we need to tons of images to train our model to manually solve them. Fortunately, there are CAPTCHA samples from the https://captcha.com/captcha-examples.html (https://captcha.com/captcha-examples.html) I download almost 9,000 images as my datatsets. The samples I used are all formated with 72x24 pixels and the backgrounds are all white, which save me lots of time at preprocessing. The postion of numbers and letters are almost at the center of images. Although the vertical position of each number or letter is not the same, the max height and max width of each number or letter are all the same. This characteristic is important if there are glue with each other, like "XWJE" as below. With the glue of both "W" and "J" letters, I can use the characteristic mentioned above to recognize and split the letters.

After I browsed whole datasets, I can sort images into five classes, all letters, all numbers, one number three letters, two numbers two letters and three numbers one letter. The amount of five classes images are almost random and all letters are capital. Except for all numbers and all letters, the sequence of letters and numbers are also random and there are no sign of dominant class; therefore, I can just randomly split the datasets into training/validation/testing.

In [3]:
```
from IPython.display import Image
display(
Image(filename = 'C:\\Users\\MLUSER\\Documents\\GitHub\\Udacity\\2A5R.PNG'),
Image(filename = 'C:\\Users\\MLUSER\\Documents\\GitHub\\Udacity\\XWJE.PNG')
)
```

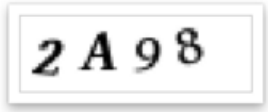


2A2X.png

2A5R.png

2A5Z.png

2A9N.png

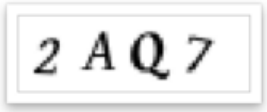


2A98.png

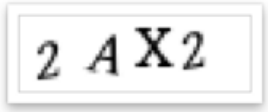
2AD9.png

2AEF.png

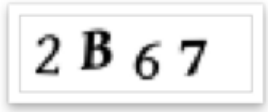
2APC.png



2AQ7.png



2AX2.png



2B67.png

2BF6.png

2BK3.png

2BKT.png

2BLG.png

2BN9.png

2BTE.png

2BTX.png

Solution Statement

For this kind of image recogition problem, the method flashed into my mind is Convolutional Neural Networks(CNN). Based on the properity of input datasets as mentioned above, at preprocessing step, I will firstly enchance the pixels of letters and numbers images pixels by openCV package, then split each number and letters into same pixels grouping and save them into files named "number(0~9)" or "letter(A~Z)". Secondly, I will map the labels into one-hot encoding by sklearn package. After preprocessing, I will build my CNN model by Keras package with Tensorflow backend. Because the simple complexity of my input datasets, I may just use two convolutional layers to enchance the computaional efficiency.

Benchmark Model

I plan to use Multiply Layers Perceptron(MLP) as bechmark models to compare the CNN. The idea is from the MLND projects "cifar10_mlp" and "cifar10_cnn". The CAPTCHAs datasets are well labeled like cifar datasets and even much simplier. I will also use Keras as tool to construct the MLP Model.

Evaluation Metrics

Fisrt, I may use accuracy to evaluate the CNN and MLP model. Because, in real scenario, there is no tolerance of false judgement rate. Furthermore, The typical CAPTCHAs images, consisted of 0~9 and A~Z, consider the upper and lower case as the same. Second, the ROC curve I will definitely use as reviewer advice. For MLP and CNN Models, I will test each model by diferent validation folds to verify the rate of true positive versus false positive.

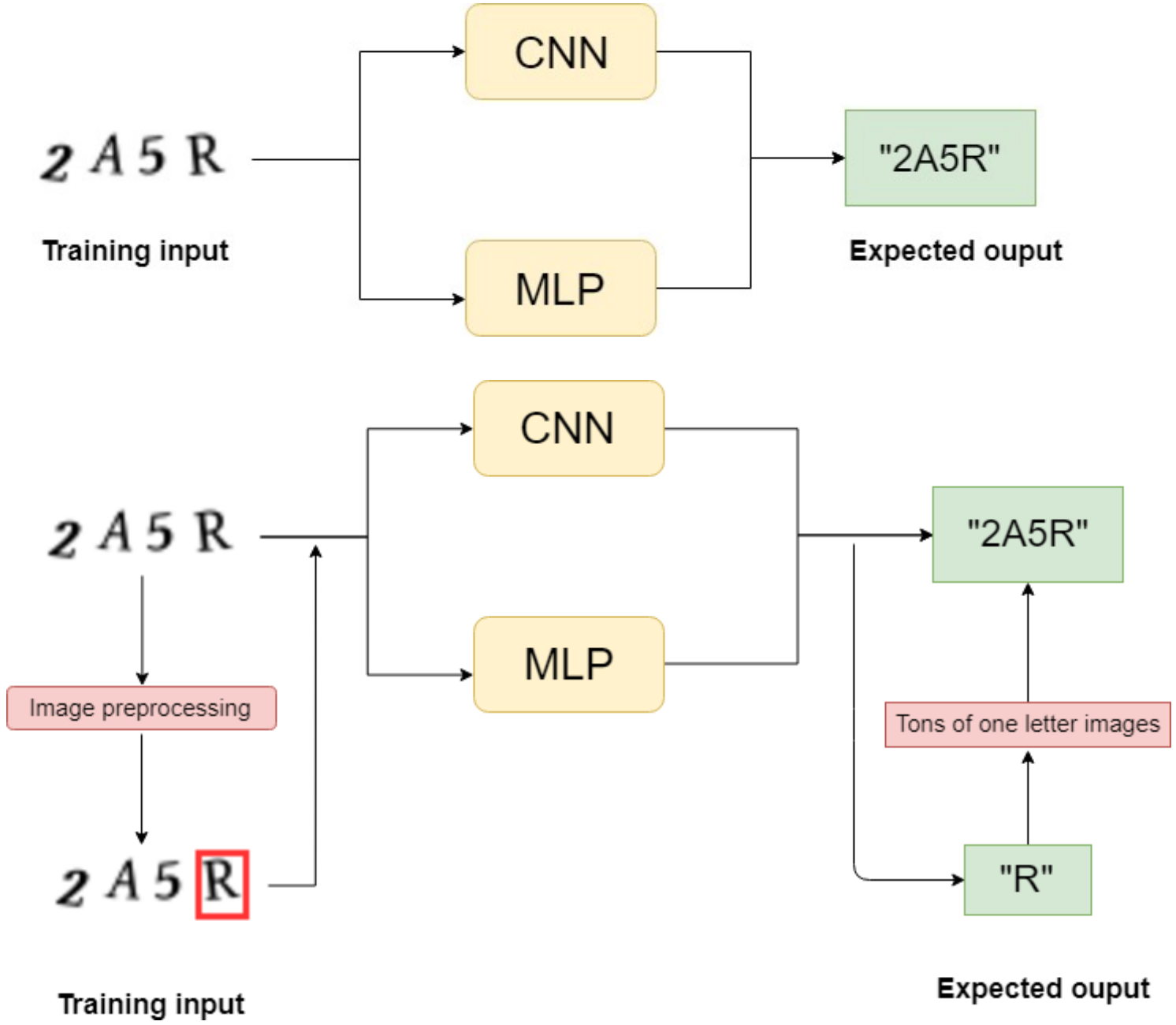
Project Design

My traing data as below are images with random four letters, numbers and characters. the images files are named as their contents. However, we can find that images with four latters can be split to one letter, then we can use one letter image to train our model. it will reduce our computational time. Therefore, In image preprocessing, I use findContours() of OpenCV function to split the boundaries of each letter in each image. Then I extract individual letters from images I have and save each letter in it's own folder. So with images preprocessing, our working flow is modified as below.

The strategy I used is two Convolutional layers with two MaxPooling layers and two fully-connected layers(Dense), one is hidden layer and the others is output layer. I used the Multiply layer perceptron(MLP) as the benchmark model. the architecture as below is two fully-connected layers with two drop out layers.

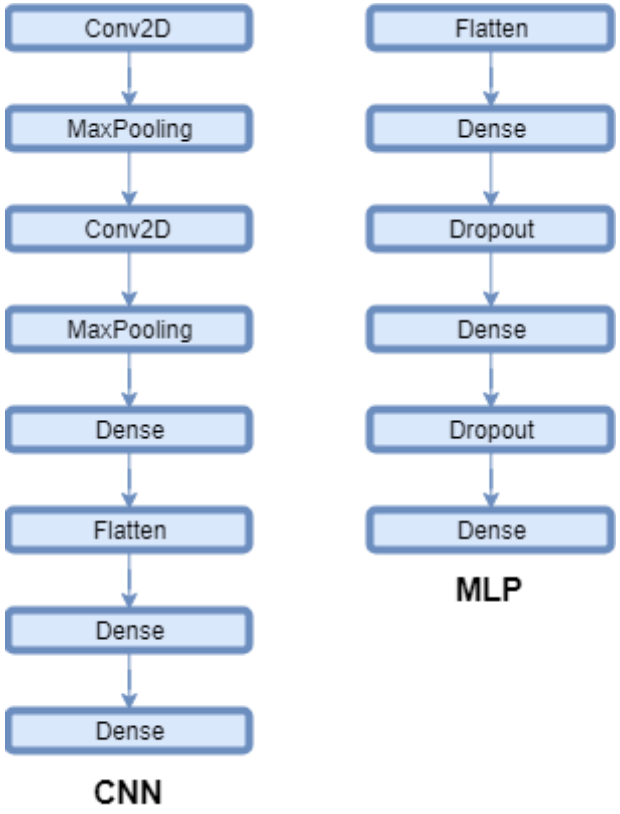
In [34]:

```
from IPython.display import Image
import nbconvert
display (
Image(filename = 'C:\\Users\\MLUSER\\Documents\\GitHub\\Udacity\\flow diagram.jpg'),
Image(filename = 'C:\\Users\\MLUSER\\Documents\\GitHub\\Udacity\\flow diagram (9).png')
)
```



In [29]:
```
Image(filename = 'C:\\Users\\MLUSER\\Documents\\GitHub\\Udacity\\flow diagram (6).png')
```

Out[29]:



Before submitting your proposal, ask yourself. . .

- Does the proposal you have written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Solution Statement** and **Project Design**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your proposal?
- Have you properly proofread your proposal to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?

```
In [ ]:
```