# CAPTCHA Identification Using CNN

*Jiamin Shang, Zhixin Wang*

## 1.Abstract

This study attempts to identify different forms of captcha. Our project. Specifically this project covers 4000 captcha pictures, which includes digital captcha and image captcha. The ultimate goal is to identify the content correctly when given a certain type of captcha. Convolutional neural network is used on this prediction and it has an adequate feedback. Grey process, size normalization will be conducted on the raw data. After the preprocessed data is used to generate training model, our test data indicated that our model has a good performance, the accuracy is over 95%.

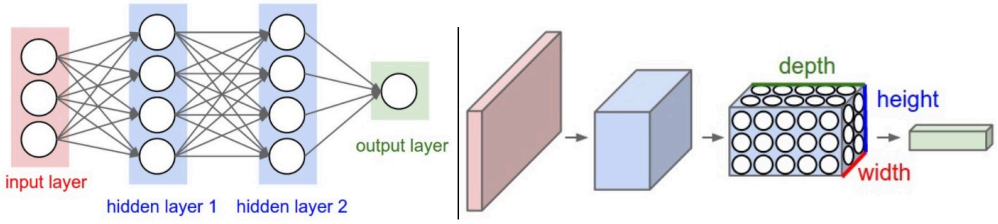
## 2.Introduction

### 2.1Motivation & Background

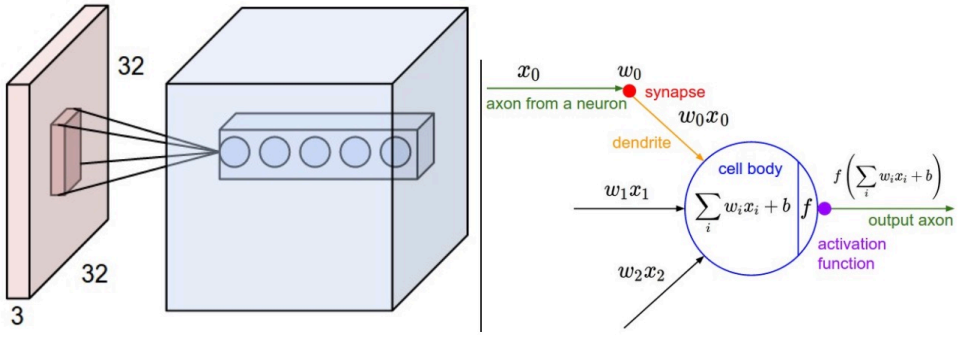
Image processing techniques have been applied increasingly is recent years. Self driving would be the representative one, autos can deal with different scenarios because they have already been training after learning different patterns, what is a signal, how to judge if another vehicle is too close to hit you. So we want to know the performance of image recognition, how accurate it can be, how sensitive it could be to different images. In addition, our project can let us understand how machine learning techniques help build the architecture of image recognition.

### 2.2 Algorithm

Convolutional neural network is very similar to the ordinary Deep Neural Network, they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, and produces non-linearity output. And they still have a loss function on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.




The whole network is different at ConvNet and MaxPoolNet, which is designed for input that have too many dimensions,30*30 image would have 30*30*3 dimensions input. ConvNet would use a small local receptive field to scan the input which is be able to reduce the dimensions to save time and resource to train our model. And MaxPoolNet is like to calculate the value from the feature map that we got from last layer, the method maybe the maximum value of ConvNet output or average value of it.




## 3. Code with document

## 3.1 Load Dataset and Data preprocess

The first step is to load dataset, our dataset contains digital captcha and picture captcha, which would have a little bit different in data preprocessing. In the digital captcha part, we would segment the four-digital captcha into four parts and the label would 0~9, for the picture captcha, we would just label them with the category they belong to.

In [4]:
```
def read_path(path_name):
    images = []
    labels = []
    for dir_item in os.listdir(path_name):

        full_path = os.path.abspath(os.path.join(path_name, dir_item))

        if os.path.isdir(full_path):
            read_path(full_path)
        else:
            if dir_item.endswith('.jpg'):
                image = cv2.imread(full_path)
                image = cv2.resize(image, (IMAGE_SIZE, IMAGE_SIZE), interpolation=cv2.INTER_NEAREST)

                image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY );

                imagex = []
                for img in image:
                    imagex.extend(img)
                images.append(imagex)
                labels.append(path_name)

    return images,labels
```

In [6]:
```
def load_dataset(path_name):
    images,labels = read_path(path_name)
    result = []
    #labels = [[0,1] if label.endswith('camera') else [1,0] for label in labels]
    for label in labels:
        if label.endswith('0'):
            result.append([1,0,0,0,0,0,0,0,0,0])
        elif label.endswith('1'):
            result.append([0,1,0,0,0,0,0,0,0,0])
        elif label.endswith('2'):
            result.append([0,0,1,0,0,0,0,0,0,0])
        elif label.endswith('3'):
            result.append([0,0,0,1,0,0,0,0,0,0])
        elif label.endswith('4'):
            result.append([0,0,0,0,1,0,0,0,0,0])
        elif label.endswith('5'):
            result.append([0,0,0,0,0,1,0,0,0,0])
        elif label.endswith('6'):
            result.append([0,0,0,0,0,0,1,0,0,0])
        elif label.endswith('7'):
            result.append([0,0,0,0,0,0,0,1,0,0])
        elif label.endswith('8'):
            result.append([0,0,0,0,0,0,0,0,1,0])
        elif label.endswith('9'):
            result.append([0,0,0,0,0,0,0,0,0,1])

    return images, result
```

In [5]:
```
def load_dataset(path_name):
    images,labels = read_path(path_name)
    result = []

    for label in labels:
        if label.endswith('camera'):
            result.append([1,0,0,0,0,0,])
        elif label.endswith('lotus'):
            result.append([0,1,0,0,0,0])
        elif label.endswith('watch'):
            result.append([0,0,1,0,0,0])
        elif label.endswith('umbrella'):
            result.append([0,0,0,1,0,0])
        elif label.endswith('sunflower'):
            result.append([0,0,0,0,1,0])
        elif label.endswith('scissors'):
            result.append([0,0,0,0,0,1])


    return images, result
```

## 3.2 Architecture of Model and Functions

Then we defined the parameters that used for the training of neural network, and the ConvNet and MaxpoolNet function, we will use a 2*2 receptive field to scan the whole input and the stride is 1,  then we take the maximum value of receptive field before we send it to the activation function.

In [9]:
```
training_epochs = 61
learning_rate = 0.0005
decay_rate = 0.9
momentum=0.001
```

In [10]:
```
def weight_variable(shape,stddev):
    initial = tf.truncated_normal(shape, stddev=stddev)
    return tf.Variable(initial)

def bias_variable(shape,stddev):
    initial = tf.constant(stddev, shape=shape)
    return tf.Variable(initial)

def conv2d(x, W):

    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):

    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                          strides=[1, 2, 2, 1], padding='SAME')
```

The third part of our work is to build the architecture of our neural network. We implemented three hidden layers in the network, for example, in the first hidden layer, we used 8*8 patch to calculate 32 features, 1 means the input channel and 32 means the output channel, 0.01 is the parameter for the initialization of weights. In addition, we set the dropout variable to avoid the overfitting situation.

In [11]:
```
with tf.name_scope('data'):
    X = tf.placeholder("float", [None, 1600])
    Y = tf.placeholder("float", [None, 6])

    x_image = tf.reshape(X, [-1,40,40,1])

with tf.name_scope('parameters'):
    with tf.name_scope('inputlayer'):
        W_conv1 = weight_variable([8, 8, 1, 32],0.01)
        b_conv1 = bias_variable([32],0.01)
        h_conv1 = tf.nn.elu(conv2d(x_image, W_conv1) + b_conv1)
        h_pool1 = max_pool_2x2(h_conv1)

        tf.summary.histogram('W_conv1',W_conv1)
        tf.summary.histogram('b_conv1',b_conv1)

    with tf.name_scope('hiddenlayer1'):
        W_conv2 = weight_variable([8, 8, 32, 64],0.01)
        b_conv2 = bias_variable([64],0.01)
        h_conv2 = tf.nn.elu(conv2d(h_pool1, W_conv2) + b_conv2)
        h_pool2 = max_pool_2x2(h_conv2)

        tf.summary.histogram('W_conv2',W_conv2)
        tf.summary.histogram('b_conv2',b_conv2)

    with tf.name_scope('hiddenlayer1_1'):
        W_conv3 = weight_variable([8, 8, 64, 64],0.01)
        b_conv3 = bias_variable([64],0.01)
        h_conv3 = tf.nn.elu(conv2d(h_pool2, W_conv3) + b_conv3)
        h_pool3 = max_pool_2x2(h_conv2)

        tf.summary.histogram('W_conv3',W_conv3)
        tf.summary.histogram('b_conv3',b_conv3)
```

```
    with tf.name_scope('hiddenlayer2'):
        W_fc1 = weight_variable([10 * 10 * 64, 1024],0.01)
        b_fc1 = bias_variable([1024],0.01)
        h_pool2_flat = tf.reshape(h_pool3, [-1, 10*10*64])
        h_fc1 = tf.nn.elu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

        tf.summary.histogram('W_fc1',W_fc1)
        tf.summary.histogram('b_fc1',b_fc1)

    with tf.name_scope('dropout'):
        keep_prob = tf.placeholder(tf.float32)
        h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

        tf.summary.scalar('dropout_keep_probability', keep_prob)

    with tf.name_scope('outputlayer'):
        W_fc2 = weight_variable([1024, 6],0.01)
        b_fc2 = bias_variable([6],0.01)

        tf.summary.histogram('W_fc2',W_fc2)
        tf.summary.histogram('b_fc2',b_fc2)
```

In the fourth part, we used Softmax as the classification activation function, cross entropy as the loss function , AdmaOptimizer as the gradient estimation function and built a function to record the accuracy of our training model.

```
with tf.name_scope('activations'):
    y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

In [16]:
```
with tf.name_scope('cross_entropy'):
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(y_conv), reduction_indices=[1]))
    tf.summary.scalar('cross_entropy', cross_entropy)

with tf.name_scope('train'):
    train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)

with tf.name_scope('accuracy'):
    with tf.name_scope('correct_prediction'):
        correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(Y,1))
    with tf.name_scope('accuracy'):
        accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
        tf.summary.scalar('accuracy', accuracy)

with tf.name_scope('prediction'):
    prediction = tf.argmax(y_conv,1)
```

## 3.2 Model Result Test

Finally , the main function is to launch the project work and get the output of our program, predict fucntion reflects the fact that when digital captcha and picture captcha go through the training model, whether our model can show the content of captcha correctly.

In [13]:
```
sess = tf.Session()
merged = tf.summary.merge_all()
writer = tf.summary.FileWriter("./data/logs/", sess.graph)

init = tf.global_variables_initializer()
init.run(session=sess)

for i in range(training_epochs):
    for start, end in zip(range(0, len(train_images), 30), range(30, len(train_images), 30)):
        train_step.run(session=sess,feed_dict={X: train_images[start:end], Y: train_labels[start:end], keep_prob: 0.5})
        rs=sess.run(merged,feed_dict={X: train_images[start:end], Y: train_labels[start:end], keep_prob: 0.5})
        writer.add_summary(rs, i)

    if (i)%5 == 0:
        train_accuracy = accuracy.eval(session=sess,feed_dict={X: valid_images, Y: valid_labels, keep_prob: 1.0})
        print("step %d, training accuracy %g"%(i, train_accuracy))
```

In [33]:
```
imaget, labels = load_dataset("./data/preimg")
imaget = np.array(imaget)
```

In [34]:
```
predict = sess.run(prediction,feed_dict={X: imaget,keep_prob: 1.0})
print(outputres(predict))
```

In [21]:
```
predict = sess.run(prediction,feed_dict={X: targetimg,keep_prob: 1.0})
print(predict)
```

# 4.Result

*For digital captcha, the accuracy record as follow:*

| Training epochs | Accuracy |
|---|---|
| 0 | 0.313253 |
| 5 | 0.493976 |
| 10 | 0.86747 |
| 15 | 0.987952 |
| 20 | 1 |

*Input*

*Output*

```
predict = sess.run(prediction,feed_dict={X: targetimg,keep_prob: 1.0})
print(predict)
```

```
[9 1 4 1]
```

*For picture capthca, the accuracy record as follw:*

| Training epochs | Accuracy |
|---|---|
| Step 0 | 0.446429 |
| Step 5 | 0.785714 |
| Step 10 | 0.767857 |
| Step 15 | 0.821429 |
| Step 20 | 0.75 |
| Step 25 | 0.75 |
| Step 30 | 0.767857 |
| Step 35 | 0.75 |
| Step 40 | 0.75 |
| Step 45 | 0.75 |

| | |
|---|---|
| Step 50 | 0.75 |
| Step 55 | 0.75 |
| Step 60 | 0.75 |

*Input*

*Output*

```
In [34]: predict = sess.run(prediction,feed_dict={X: imaget,keep_prob: 1.0})
         print(outputres(predict))

         ['watch', 'camera', 'sunflower', 'watch', 'sunflower', 'camera']
```

## 5.Discussion

The whole process of this machine learning project is truly interesting. It help us go through the neural network model and many details. Convolutional neural network is suitable for the case that input contains too many dimensions, after the implement of convolution layer and max pool layer, the model would become more concise and readable.

During our implementation of this project, we put a lot of effort into the improvement of model accuracy. We tried to reshape image in different size, change the network architecture, loss function, activation function, gradient estimation and parameter initialization, as a result, 4-5 hidden layers, cross entropy and Relu function, AdamOptimizer, Gaussian initialization serves as the best combinations in our test.

In the end, the data we collected are some pictures and they are highly similar to each other, which made our model fit for some certain styles picture, in the future scope, we want to collect more different pictures as input to make our model more reliable.

## 6.Reference:


[1] Convolutional Neural Network for Visual Recognition: http://cs231n.github.io/convolutional-networks/

[2] Image Segmentation: http://www.cnblogs.com/subconscious/p/4660952.html

[3] Stanford University CS231n: Convolutional Neural Networks for Visual Recognition: http://cs231n.stanford.edu/project.html

[4] CIFAR-10 - Object Recognition in Images: https://www.kaggle.com/c/cifar-10

[5] ImageNet Classification with Deep Convolutional Neural Networks. Alex Krizhevsky. 2017