**Convolutional Neural Network And Dog Breeds**

Zheran (Rachel) Wang

Department of Math & Statistics

# Abstract

Convolutional Neural Networks (CNN) is a class of artificial neural networks that mostly use on image analysis. In this research, CNN technology will be used to apply onto the dog image data and determine the breed of the dog from the image, also test how accurate the prediction.

Though the analysis, the last iteration obtained a 100% of training accuracy on the training data, and 47% highest testing accuracy on the test data.

**Keywords**: Convolutional Neural Networks, Dog breeds, Train data, Test data.

# Introduction

Nowadays, images and videos are more common on everywhere. On the well-known video social network YouTube, when you upload a video, if you are using somebody else's music, the system will determine who this music belongs to and automatically send the viewing income to the person who the music belongs to. But in term of the video clip, the system cannot determine it. This always ends up with the issue of copyright. If the system can recognize the video clip that a video using and do the same income method as the music pieces, then there will be less issue and argument happen on the internet. But before jump directly into the video, we first need to worried about the image recognition since the video is made from a lot of images.

There are several methods to determine the images, the well-known method is called Convolutional Neural Network (CNN). Convolutional Neural Networks (CNN) is a machine learning method in a class of artificial neural networks that mostly use on image analysis. Similar to the artificial neural networks, CNN also has an input layer, hidden layers, and output layer. Inside the hidden layers, the model usually includes the convolutional layer, pooling layer, and fully connected layer.

The convolutional layer does the operation on the input image, then send the output into the second layer called pooling layer. The purpose of the pooling layer is to down-sample, which will decrease the sample to avoid overfitting. The pooling layer has different types which are max pooling layer and average pooling layer. Max pooling layer selected the maximum value for each cluster from the previous output and send to next layer. Average pooling layer that gets an average value for each cluster from the previous output and sends to the next layer. The fully connected layer simply connect all one layer to every neuron on another layer and make the final decision based on the previous activation.

Purpose of the analysis

This is study, I am going to use CNN method to determine the dog breeds from the dog images. And test how accurate the prediction was.

# Data

Data source:

Dog Breed Identification from Kaggle Competition website.

<https://www.kaggle.com/c/dog-breed-identification#description>

# Methods and analysis

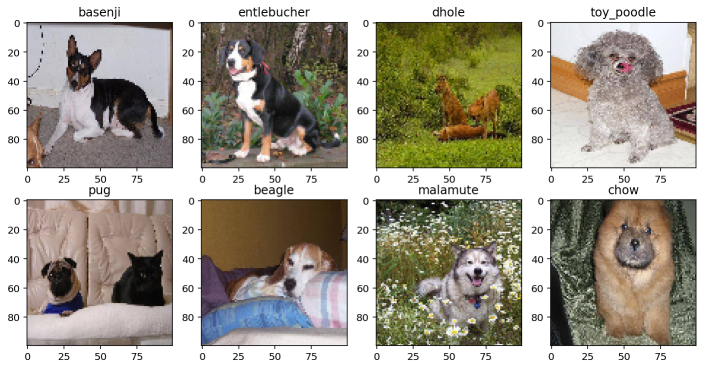
Since CNN is the method of machine learning, which mean the computer (machine) will take care of most the work, but there still have something else that needs to be done before leaving everything to the computer.

## Step 1: import data

The data is called Dog Breed Identification which comes from the Kaggle competition website. The main purpose of the competition is to use 10222 different dog images which include 120 different breeds of dogs to determine the breed of a dog from the image.

The data comes with only the image with ID and the breed label with ID, to import data, not only need to import the image and the label also needs to merge them so that the breed label and the image match. Also since the images are coming with different sizes, I changed the size of all the images into 100x100. In order to match the image and label and change the images size, the method that I use is called OpenCV. This method in python is common to deal with images. After import the image, I checked if the image and the breeds are correctly matched by using plot image.

The sample images with breeds are showing below.



## Step 2: decrease the sample size

My personal purpose is instead of testing the whole 120 different breeds, just focus on the most repeated 5 different breeds and test how accurate the result was. This step can be done by define the frequency function and choose the 5 most repeated breeds and put into a new dataset. My new dataset after this step now has 588 images in it.

The five most repeated breeds are Maltese dog, a Scottish deerhound, Entlebucher, Bernese mountain dog, and Afghan hound.

## Step 3: Apply CNN

First Convolutional layer

The technique that using to apply CNN is Tensorflow with one thousand iterations and 50 as the interval between the iteration. The dimension of the image will change from R­2 to R1 by multiplying the weight and the height together. Then I define the first convolutional layer by setting the strides into 1 so that the filter moving 1 pixel along x-axis and y-axis. The set the padding as same so that each output will end up with same image size. Then add a bias-value to each filter-channel, which in this case, the bias is the number of filters. In this layer, I have 5x5 pixel across x and y-axis and 8 filters. This step can be done by using the default function in python called tensorflow.conv2d.

Max-Pooling layer

Then I define the pooling layer, I choose max-pooling as the pooling layer with filter size as 2x2 so that the machine will pull out the maximum value from each 4 value in a window, and with 4 different windows, will form a 2x2 matrix will the largest value in each window. This step can be done by using the default function called tensorflow.max-pooling.

ReLU layer

After the output from pooling layer, the sample will move to the ReLU layer, which will add non-saturating [activation function](https://en.wikipedia.org/wiki/Activation_function) f(x)=max(0,x), this step will increase the nonlinearity to the decision function and the overall network, which allow us to learn more complicated functions. This step can be done by using the default function called tensorflow.relu.

Flatten layer

Before letting the result goes to the second convolutional lay, I did some adjustment to the output, which was done in the flatten layer. This layer changing the shape of the output image into the new dimension, in this case, the size of the new dimension is the number of features -1. The number of features can be calculated by multiply number of image, image height, image weight, and a number of channels together. This step can be done by using default function called tensorflow.reshap.

Second convolutional layer

Then the data moving to second convolutional layer to increase the accuracy. This time, since the resize already been done in the previous layer, I just choose the same pixel size as the first convolutional layer, which is 5x5 across x-axis and y-axis. But this time, the number of filter change into 16 filters.

Fully connected layer

After all the previous layer, all the data will go into the fully connected layer, this layer is connected all the activation in the previous layer and also this layer is where the decision making. The decision can be made by computing the matrix multiplication followed by a bias offset. In this case, the bias is the number of the output. Also, this layer can add ReLU layer to help the analysis. The fully connected size is 128.

Loss layer

After the decision made from the previous layer, the loss layer usually the layer that testing the accuracy of the training data with the true labels. This layer usually is the last layer of CNN. In this case, the cross-entropy loss is used to determine the accuracy. The learning rate is 10-4.

## Step 4: Test result

After getting the accuracy of the training data, I also test the test data. Since I only pick the most repeated 5 breeds as my training dataset, there is no reason that I use the original test dataset from the website. Therefore, I created my own test dataset by getting 25 images as my test batch size for each iteration. Then test the accuracy by using the decision that made from the previous step.

# Result and Conclusion

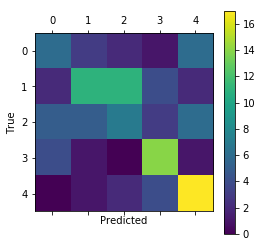
## Result

The following table shows the result of 1000 iterations in 50 intervals.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iteration: | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 | 450 | 500 |
| Training Accuracy: | 28.00% | 48.00% | 32.00% | 60.00% | 68.00% | 72.00% | 68.00% | 76.00% | 72.00% | 84.00% |
| Test Accuracy: | 30.50% | 29.70% | 36.40% | 36.40% | 36.40% | 38.10% | 44.90% | 44.90% | 43.20% | 44.90% |
| Iteration: | 550 | 600 | 650 | 700 | 750 | 800 | 850 | 900 | 950 | 1000 |
| Training Accuracy: | 80.00% | 88.00% | 96.00% | 92.00% | 84.00% | 92.00% | 96.00% | 100.00% | 100.00% | 100.00% |
| Test Accuracy: | 44.10% | 41.50% | 47.50% | 44.90% | 47.50% | 49.20% | 47.50% | 43.20% | 47.50% | 46.60% |

By looking at the result table shown above. we can see that the accuracy of training data increase to 100% after 900 iterations. But looking at the test accuracy of the output, the accuracy remains almost same after 500 integration which is around 45%. This concludes a low-test accuracy.

The plot of true value and the predicted result are showing below.



This plot shows me that for group 4 and 3, the accuracy is better than other 3 groups. The reason that cause might be the breeds of group 4 and 3 are easier to identify the other 3 groups.



From the plot shown above, it is easy to tell one of the reasons for low test accuracy. For example, on the second column last row, the image not only contains dog but also contain some human being. And instead of test only the dog image, the machine might end up testing the person’s hair.

## conclusion

Base on all the result, we can conclude that the CNN model is overfitting due to the low testing accuracy. One reason that causes this result is the sample size of the data is only 588. Usually, for image testing, the sample size should greater than 10 thousand. The size of the pixel and the fully connected size can also influence the overfitting.

The other reason for the low testing accuracy is the breeds of dog in the data, the group of Entlebucher and Bernese mountain dog is looking similar to each other in terms of color and size.

The third reason for the low test accuracy may also due to the dog's position, by looking at the images closely, there are a lot of dogs are not sit upright when taking the photo. Some dogs are swimming, some dogs are hiding the body underneath the blanket, some dogs are cover with mud. All of them will affect getting the decision function of each breed.

The fourth reason is the environment of the dog in the image. As I pointed out previously, there are some images are not only contain the dog but also contain some other elements such as human being or another type of animal (cat in the first lot) that will affect the decision made in the end.

# Further analysis

For further analysis of this project, I can increase the image number by flip the image over and add some noise to it. Or the technic called Generative adversarial networks can also be used to increase the sample size. So that the sample size will increase and improve the testing accuracy.

Instead, only test most repeated 5 breeds, I can come up with two different decision function that one of them is testing the breeds that are totally unlike, and the other one can test when the breeds are similar to each other such as Husky and Alaska.

For the environment of the image, I can get the advantage of some image editing program such as Photoshop, that can cut out only the dog from the image. So that the image in the end that only contains dog to test.

# References

1. *Simple Convolutional N. Network with Tensorflow | Kaggle*, [www.kaggle.com/kaggleslayer/simple-convolutional-n-network-with-tensorflow](http://www.kaggle.com/kaggleslayer/simple-convolutional-n-network-with-tensorflow).
2. “Convolutional Neural Network.” *Wikipedia*, Wikimedia Foundation, 9 Dec. 2017, en.wikipedia.org/wiki/Convolutional\_neural\_network#Building\_blocks.
3. Zhang, Jey. “TensorFlow学习笔记2：构建CNN模型.” *Jey Zhang*, 5 July 2017, www.jeyzhang.com/tensorflow-learning-notes-2.