Object Detection and Segmentation for Fine-Grained Recognition Using Convolutional Neural Networks in TensorFlow

Abstract- *This project addresses the problem of fine-grained recognition: recognizing subordinate categories for classifying images across 10 categories:airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Based on the insight that images with similar poses can be automatically discovered for fine-grained classes in the same domain. The appearance descriptors are learned using a deep learning using convolutional neural network. Our approach requires only image level class labels, without any use of part annotations or segmentation masks, which may be costly to obtain. TensorFlow is using as the tools for the efficient implementation. Detailed explenations are provided of implemetation including TensorBoard aspects.*

Outline

- 1.Introduction
- 2.Background
  - 2.1.Neural Network
  - 2.2.convolutional neural networks
  - 2.2.1.Local receptive fields
  - 2.2.2.Shared weights and biases
  - 2.2.3.Pooling Layers
- 3.Data Set and the Goal
  - 3.1.Description:
- 4.Architecture and Method
  - 1.Python Code
- 5.Next Steps

1. Introduction

Neural Network is well-suited to problem in which the training data corresponds to noisy, complex sensor data, such as input from cameras and microphones. In addition, it is applicable to problem for which more symbolic representations are often used. In general, it is appropriate for problems with the following

charectristics:

- *Instances are represented by many attribute-value pairs.*
- *The target function output may be descrete-valued, real-valued, or a vectored, or a vector of several real- or discrete- valued attributes.*
- *The training examples may contain errors.*
- *Fast evaluation of the learned target function may be required.*
- *The ability of humans to understand the learned target function is not important.*

Deep learning (deep structured learning, hierarchical learning or deep machine learning) based on set of algorithms attempts attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations.

Choossing Neural Network as an algorithm in deep learning often much harder to train; That's unfortunate, since we have good reason to believe that if we could train deep nets they'd be much more powerful. Furthermore, upon reflection, it's strange to use networks with fully-connected layers to classify images. The reason is that such a network architecture does not take into account the spatial structure of the images. For instance, it treats input pixels which are far apart and close together on exactly the same footing.

Convolution Neaural Network (CNN) presents a special architecture which is particularly well-adapted to classify images. It was inspired by biological processes and is variations of multilayer perceptrons designed to use minimal amounts of preprocessing. Using this architecture makes convolutional networks fast to train. This, in turns, helps us train deep, many-layer networks for classifying images.

TensorFlow is used to accompolish the mathematical computation with a directed graph of nodes & edges:

- Nodes: typically implement mathematical operations, but can also represent endpoints to feed in data, push out results, or read/write persistent variables.
- Edges: describe the input/output relationships between nodes. These data edges carry dynamically-sized multidimensional data arrays, or tensors. This totorial can be consider as a frame work for implimenting a larger and more sophisticated models. The orgin of the Code is provided by http://www.tensorflow.org (http://www.tensorflow.org). The wirter manipulate the code based on the desire to implement the more complex convolutional networks, illustrating more details, and visualizing the ongoing process in the network architecture, training and evaluation

Two datasets are going to investigate in this network:

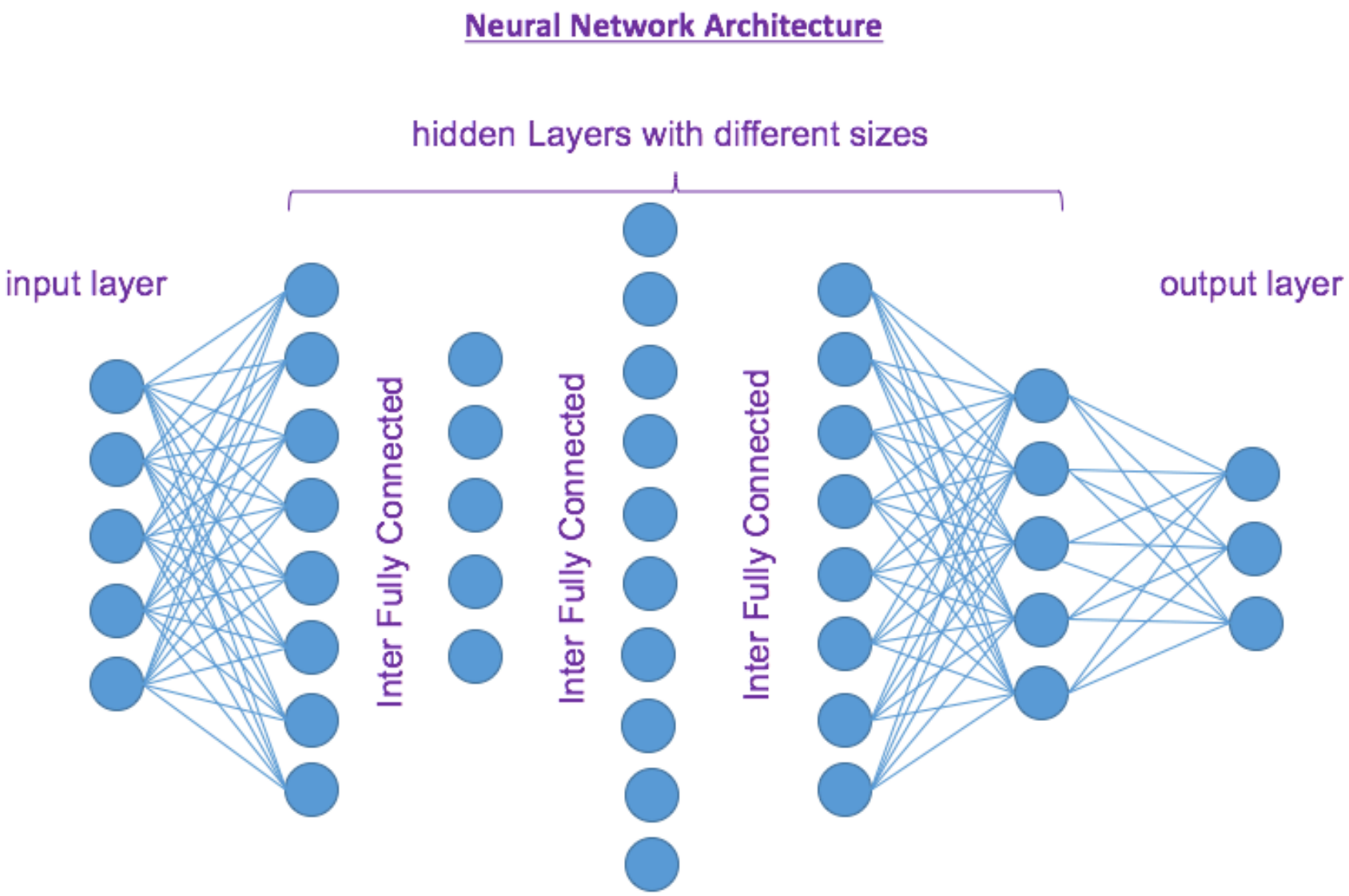
- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. http://www.cs.toronto.edu/%7Ekriz/cifar.html (http://www.cs.toronto.edu/%7Ekriz/cifar.html).
- Yahoo! Shopping Shoes Image Content, version 1.0. This dataset contains 107 folders, each corresponding to a type and brand of shoe. for instances: classes/aerosoles_sandals, classes/aetrex_sandals, ... Each folder contains some number of images of shoes of the respective type and brand. All together 5357 images. This dataset should be polished to extract training images and test images. http://webscope.sandbox.yahoo.com (http://webscope.sandbox.yahoo.com).

The report is currently in beta. I welcome notification of typos, bugs, minor errors, and major misconceptions. Please drop me a line at Ali.Miraftab@utsa.edu if you spot such an error.

2. Background

2.1. Neural Network

We already know Neural Network is using networks in which adjacent network layers are fully connected to one another. That is, every neuron in the network is connected to every neuron in adjacent layers:



In particular, for each pixel in the input image, we encoded the pixel's intensity as the value for a corresponding neuron in the input layer. As an instance, for the 640×480 pixel images we've been using, this means our network has 307200 input neurons. We then trained the network's weights and biases so that the network's output would - we hope! - correctly identify the input image.

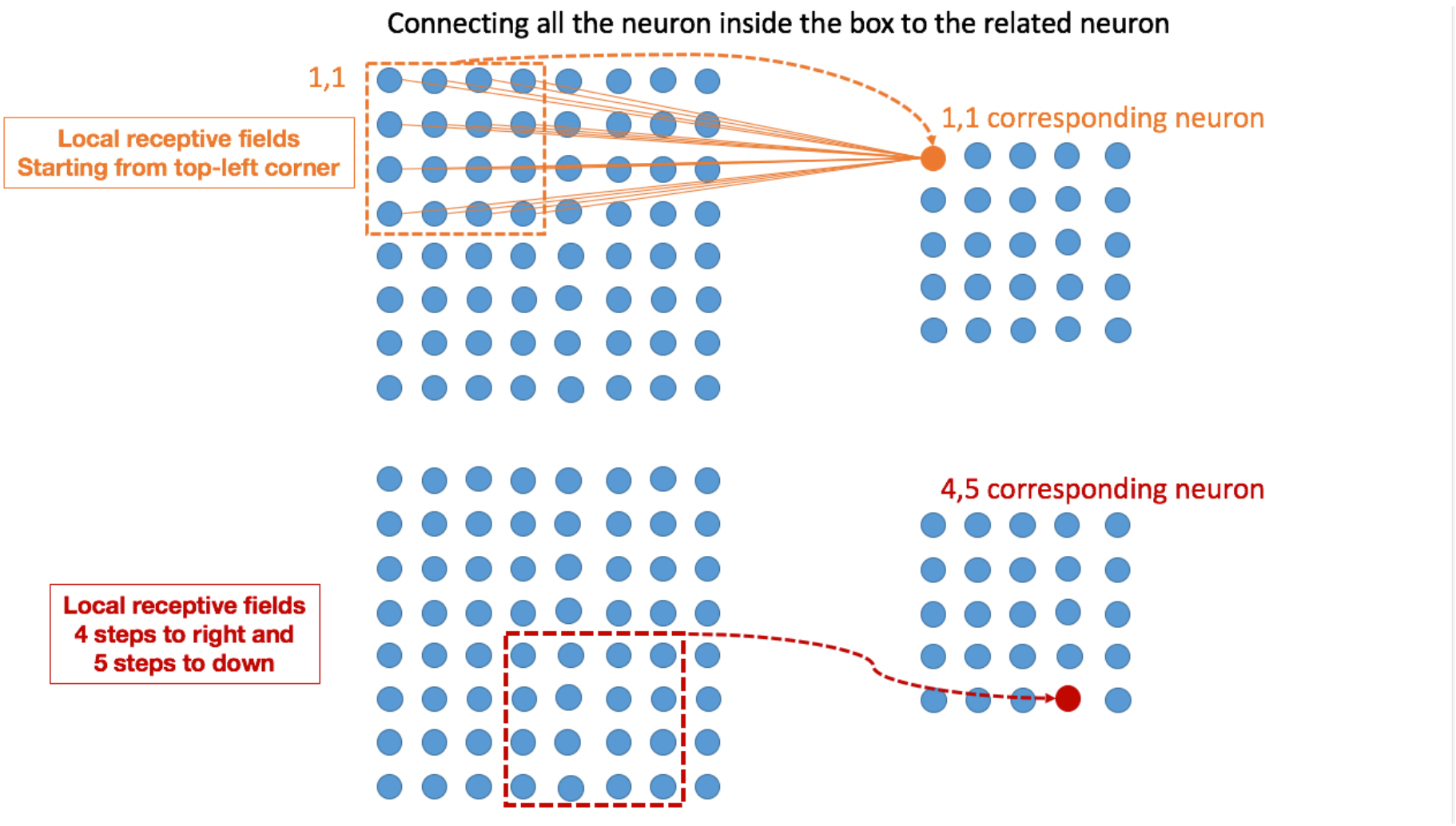
2.2. convolutional neural networks

To implement fast and more intelligent multilayer nural network, the architecture on CNN use three basic ideas:

- local receptive fields,
- shared weights,
- and pooling.

- Local receptive fields:

The window on the input pixels in the input image is called the local receptive field for the hidden neuron.
It is shown the local receptive field being moved by one pixel at a time. In fact, sometimes a different length is used. **This movement step is called Stride.** Usually, it is set as one. However, different stride lengths can be investigate by using validation data for the best performance.
And so on, building up the first hidden layer.



2.2.2. Shared weights and biases:

Each hidden neuron has a bias and weights connected to its local receptive field. CNN uses the same weights and bias for each of the **Local receptive fields** and correspondly hidden neurons. In other words, for the j,k the hidden neuron, the output is:

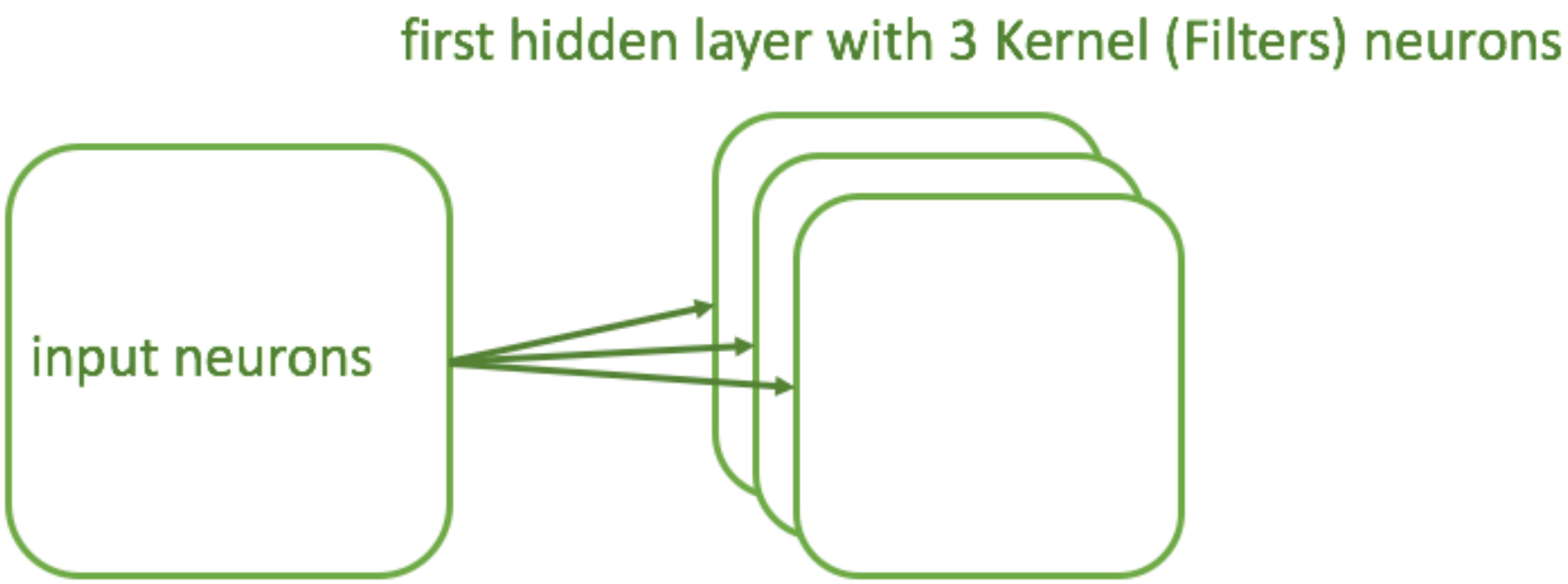
$$\sigma\left(b + \sum_{l=0}^{x}\sum_{m=0}^{x} w_{l,m} a_{j+l,k+m}\right). \tag{1}$$

Here, σ is the neural activation function - perhaps the sigmoid function we used in earlier chapters. b is the shared value for the bias. $w_{l,m}$ is a 5×5 array of shared weights. And, finally, we use $a_{x,y}$ to denote the input activation at position x,y.

This implies that all the neurons in the first hidden layer detect exactly the same feature. As a matter of fact, CNN are well adapted to the translation invariance of images: move a picture of a cat (say) a little ways, and it's still an image of a cat.

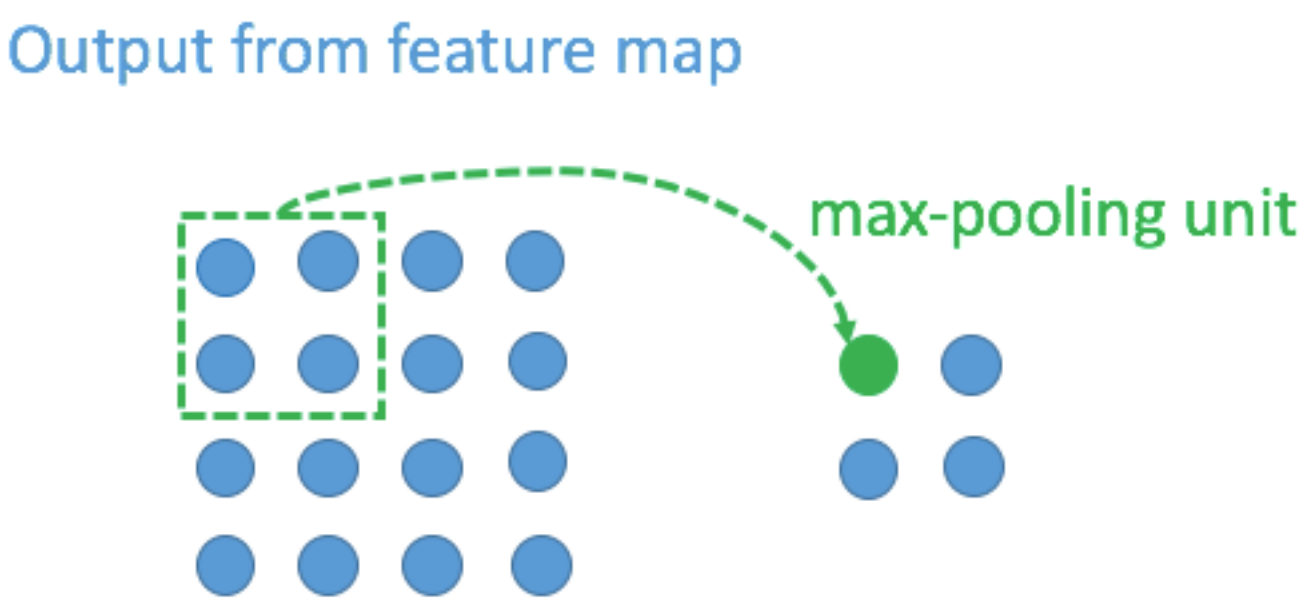
For this reason, we sometimes call the map from the input layer to the hidden layer a feature map. We call the weights defining the feature map the shared weights. And we call the bias defining the feature map in this way the shared bias. The shared weights and bias are often said to define a kernel or filter. In the literature, people sometimes use these terms in slightly different ways.
A big advantage of sharing weights and biases is that it greatly reduces the number of parameters involved in a convolutional network.

The network structure I've described so far can detect just a single kind of localized feature. To do image recognition we'll need more than one feature map. And so a complete convolutional layer consists of several different feature maps:



2.2.3. Pooling Layers:

In addition to the convolutional layers just described, convolutional neural networks also contain pooling layers. Pooling layers are usually used immediately after convolutional layers. What the pooling layers do is simplify the information in the output from the convolutional layer.



In detail, a pooling layer takes each feature map output from the convolutional layer and prepares a condensed feature map. For instance, each unit in the pooling layer may summarize a region of (say) 2×2 neurons in the previous layer.

- max-pooling
- L2 pooling

3. TensorFlow for the CIFAR-10 dataset

The proposed mode for CIFAR-10 by Alex Krizhevsky consists alternating convolutions and nonlinearities which is followed by fully connected layers and at the end softmax classifier. The implemented model by TensorFlow webpage consists of two CNN and two fully connected network layers with with a peak performance of about 86% accuracy within a few hours of training time on a GPU. Here it is tried to implement the three CNN layers and improve the performance. In parallel, the sophisticated framework for implementing CNN will be study in details. the majority of the materials can be found in the TensorFlow webpage. However, this report tries to put the explanations in a same place and modify the framework by it's own purpose.
Note: The lack of GPU in the current machines causes spending much more time.

3.1. Code Navigation:

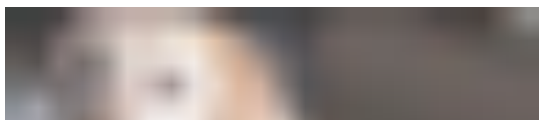
- **Read the input : cifar10_input.py**
  - inputs( ) : read the data
  - distorted_inputs( ) : preprocess the data
- **Make the Model : cifar10.py**
  - inference( ) : classification, defining the layers
- **Trains a model : cifar10_train.py**
  - loss( )
  - train( )
- **Evaluate&check : cifar10_eval.py**

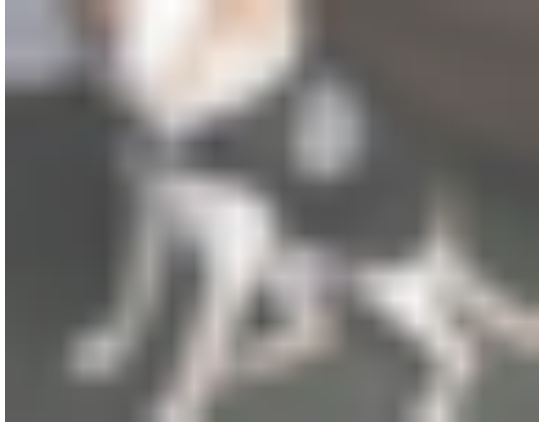
3.1.1 Preprocess of the data:

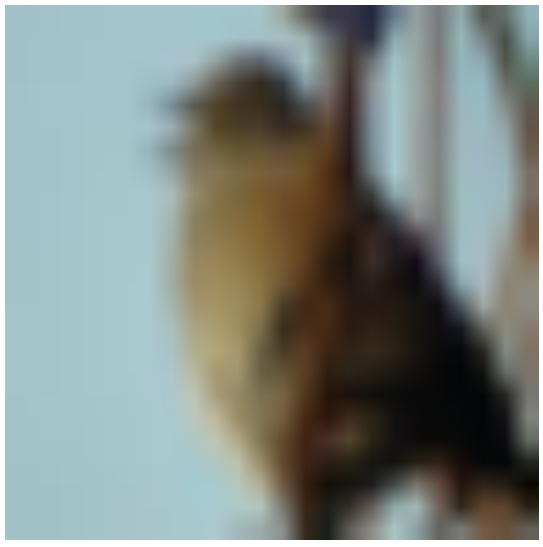
TensorFlow provides a variety of distorions which can be used for training step. For instance: Randomly flip, image brightness, image contrast, ...

In [ ]:

```
distorted_image = tf.image.random_brightness(distorted_image, max_delta=63)
distorted_image = tf.image.random_contrast(distorted_image, lower=0.2, upper=1.8)
```

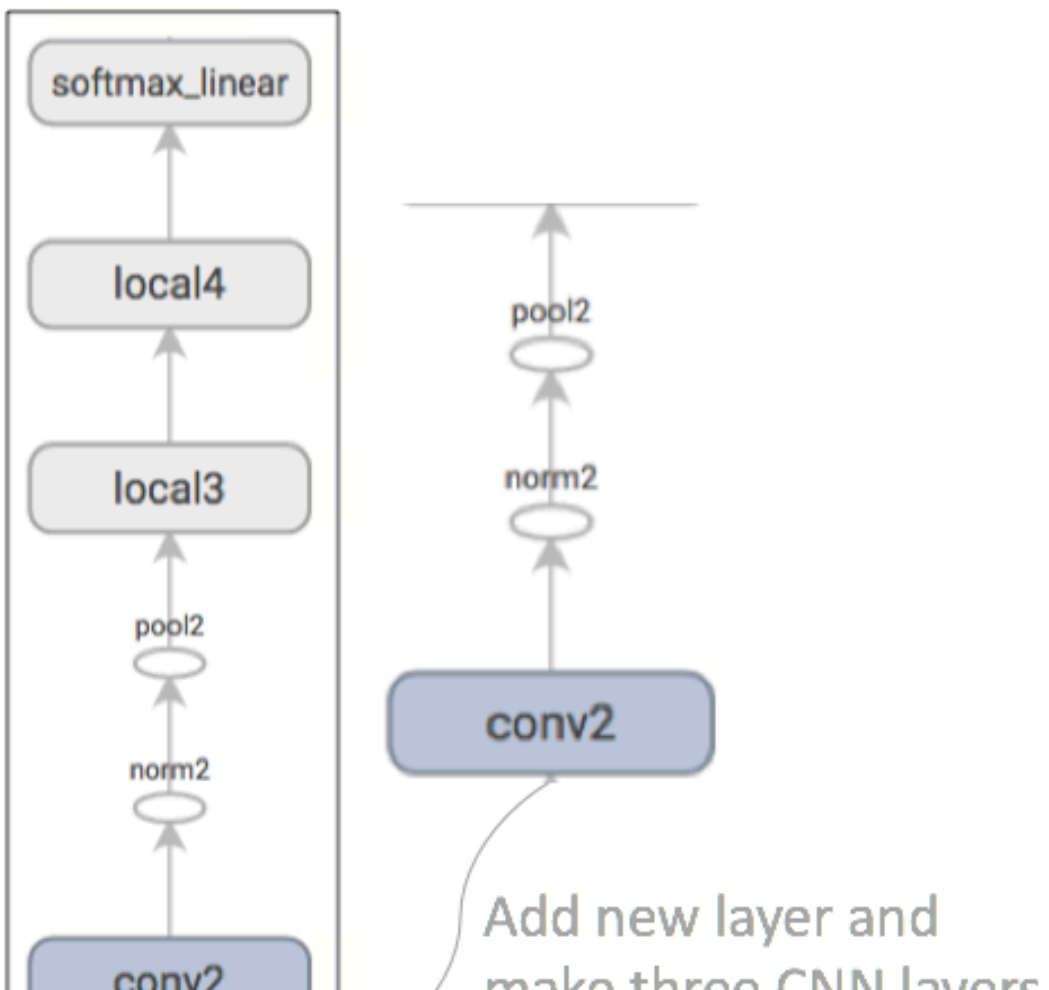


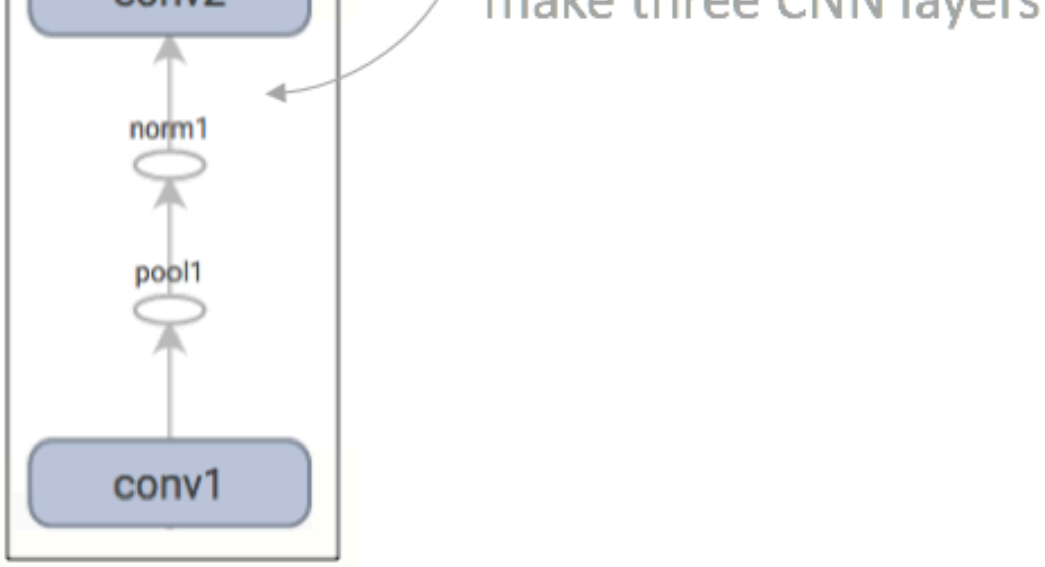




3.1.1 Studying Model:

As the aforementioned, inference() contains the model constructed. the graph that is produced by TensorBoard describes the inference operation. The 3rd CNN graph which is going to add to the model is illustrated separately.





4. Architecture and Method

4.1. Pytheon Code

In [7]:

```
from __future__ import print_function
%matplotlib inline
print("Hello")
```

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/IPython/core/ultratb.py
", line 1118, in get_records
    return _fixed_getinnerframes(etb, number_of_lines_of_context, tb_o
ffset)
  File "/usr/local/lib/python2.7/dist-packages/IPython/core/ultratb.py
", line 300, in wrapped
    return f(*args, **kwargs)
  File "/usr/local/lib/python2.7/dist-packages/IPython/core/ultratb.py
", line 345, in _fixed_getinnerframes
    records = fix_frame_records_filenames(inspect.getinnerframes(etb,
context))
  File "/usr/lib/python2.7/inspect.py", line 1044, in getinnerframes
    framelist.append((tb.tb_frame,) + getframeinfo(tb, context))
  File "/usr/lib/python2.7/inspect.py", line 1004, in getframeinfo
    filename = getsourcefile(frame) or getfile(frame)
  File "/usr/lib/python2.7/inspect.py", line 454, in getsourcefile
    if hasattr(getmodule(object, filename), '__loader__'):
  File "/usr/lib/python2.7/inspect.py", line 483, in getmodule
    file = getabsfile(object, _filename)
  File "/usr/lib/python2.7/inspect.py", line 467, in getabsfile
    return os.path.normcase(os.path.abspath(_filename))
  File "/usr/lib/python2.7/posixpath.py", line 371, in abspath
    cwd = os.getcwd()
OSError: [Errno 2] No such file or directory

ERROR: Internal Python error in the inspect module.
Below is the traceback from this internal error.
```

```
Unfortunately, your original traceback can not be constructed.


---------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
/usr/local/lib/python2.7/dist-packages/IPython/core/interactiveshell.p
yc in run_code(self, code_obj, result)
   2900             if result is not None:
   2901                 result.error_in_exec = sys.exc_info()[1]
-> 2902             self.showtraceback()
   2903         else:
   2904             outflag = 0

/usr/local/lib/python2.7/dist-packages/IPython/core/interactiveshell.p
yc in showtraceback(self, exc_tuple, filename, tb_offset, exception_on
ly)
   1828                     except Exception:
   1829                         stb = self.InteractiveTB.structured_tr
aceback(etype,
-> 1830                                         value, tb, tb_offs
et=tb_offset)
   1831
   1832                     self._showtraceback(etype, value, stb)

/usr/local/lib/python2.7/dist-packages/IPython/core/ultratb.pyc in str
uctured_traceback(self, etype, value, tb, tb_offset, number_of_lines_o
f_context)
   1390         self.tb = tb
   1391         return FormattedTB.structured_traceback(
-> 1392             self, etype, value, tb, tb_offset, number_of_lines
_of_context)
   1393
   1394

/usr/local/lib/python2.7/dist-packages/IPython/core/ultratb.pyc in str
uctured_traceback(self, etype, value, tb, tb_offset, number_of_lines_o
f_context)
   1298             # Verbose modes need a full traceback
   1299             return VerboseTB.structured_traceback(
-> 1300                 self, etype, value, tb, tb_offset,
number_of_lines_of_context
   1301             )
   1302         else:

/usr/local/lib/python2.7/dist-packages/IPython/core/ultratb.pyc in str
uctured_traceback(self, etype, evalue, etb, tb_offset, number_of_lines
_of_context)
   1182                 structured_traceback_parts +=
formatted_exception
   1183         else:
-> 1184             structured_traceback_parts += formatted_exception[
```

```
0]
   1185
   1186         return structured_traceback_parts

IndexError: string index out of range
```

In [ ]:

```
shoe = sio.loadmat('shoe_annos.mat')
```

In [ ]:

```
shoe
```

In [ ]:

```
shoe['annotations']
```

In [ ]:

```
shoe_annos = shoe['annotations']
```

In [ ]:

```
shoe_annos
```

In [ ]:

```
shoe_annos.shape
```

In [ ]:

```
sio.whosmat('shoe_annos.mat')
```

In [ ]:

```
shoe_annos[0,0]
```

In [ ]:

```
shoe_annos[0,0:30]
```

In [2]:

```
from IPython.display import Image, display
display(Image(filename='classes/lowa_hiking/757550203_640.jpg'))
```

In [ ]:

```
from pygame import surfarray, image, display
import pygame
import numpy #important to import


image_file = '/files/Webscope/ShoesImage/Webscope_I2/ydata-yshoes-image-content-v2_(
pygame.init()
image = image.load("classes/lowa_hiking/761326860_640.jpg") #surface to render
resolution = (image.get_width(),image.get_height())
print resolution
screen = display.set_mode(resolution) #create space for display
```

In [ ]:

```
import cv2
import numpy as np
image_file = '/files/Webscope/ShoesImage/Webscope_I2/ydata-yshoes-image-content-v2_(
img = cv2.imread(image_file)
```

In [ ]:

```
from PIL import Image
image_file = 'classes/lowa_hiking/757550203_640.jpg'
im = Image.open(image_file)
#rgb_im = im.convert('RGB')
#r, g, b = rgb_im.getpixel((1, 1))
r, g, b = im.getpixel((1, 1))
print r, g, b
```

In [ ]:

```
from PIL import Image
im = Image.open("classes/lowa_hiking/757550203_640.jpg") #Can be many different forn
pix = im.load()
print im.size #Get the width and hight of the image for iterating over
### print pix[x,y] #Get the RGBA Value of the a pixel of an image
```

In [4]:

```
import PIL
import Image
FILENAME='classes/lowa_hiking/757550203_640.jpg' #image can be in gif jpeg or png fo
im=Image.open(FILENAME).convert('RGB')
pix=im.load()
w=im.size[0]
h=im.size[1]
print (w, h)
```

```
640 480
```

In [6]:

```
i_start = 0
j_start = 0

while i_start == 0:
    for i in range(w):
        for j in range(h):
            if pix[i,j] != (255, 255, 255):
                i_start = i
                j_start = j
```

In [8]:

```
print (i_start, j_start)
```

```
539 449
```

In [16]:

```
pix[200,100]
```

Out[16]:

```
(161, 153, 130)
```

In [9]:

```
import matplotlib.cm as cm
imgplot = plt.imshow(pix[i_start:i_start+100, j_start:j_start+30], cmap = cm.Greys_r
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-9-3e5d2ee7373b> in <module>()
      1 import matplotlib.cm as cm
----> 2 imgplot = plt.imshow(pix[i_start:i_start+100, j_start:j_start+
30], cmap = cm.Greys_r)

TypeError: an integer is required
```

In [ ]:

```

```