

**Vivasvan Patel**  
**EntryNo-2016CSJ0005**  
**Institute – IIT Jammu**

This submission includes 2 implementations AlexNet and CIFAR-10 neural networks

Requirments for using both implementation – Python3, tensorflow, numpy, matplotlib and unpickle.

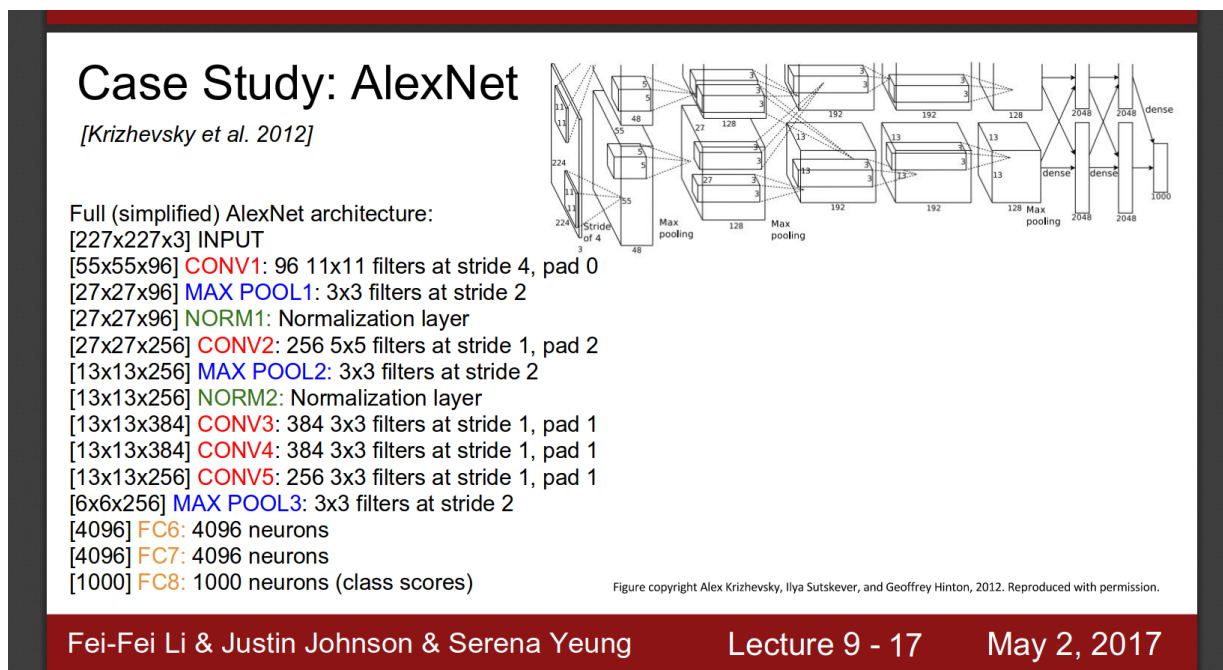
# 1

**AlexNet** :- This implementation has eight layers; the first five were convolutional layers, and the last three were fully connected layers .

It can predict upto 1000 classes.

DataSet consists of images of 3 dog breeds namely :- beagle,greatdane,bulldog.  
To add more data refer to the README inside AlexNet folder.

Archietecture :-



Gradient Descent is then applied to minimize the loss function if in Train mode.

After that we apply ArgMax to find the class closest to it the new image and SoftMax to find the probabilities of new image being the new class in Predict mode.

- Accuracy Report :- Training did not complete(due to slow only CPU computations).
- Advantages
  1. This model has total of 1000 class output so a single model can predict wide range of objects.
  2. Image quality is 256\*256 px so many details of an image can be captured in it. Giving a more accurate results than CIFAR-10.
- Disadvantage
  1. Training took too long on using CPU.

Future Plans for this implementation :-

1. Some kind of preprocessing input data to improve computation speed.
2. Code for training part to use multiple GPUs if available to the system.
3. Complete training and generate accuracy.

## 2

**CIFAR-10** :- 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.

Data download link :- <https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz>

DataSet is divided in 3 batches of train data and 1 batch of test data.

First we unpickle these batchfiles to produce a dict with keys b'label' containing the labels and b'data' containing the image data.(there are other labels as well like filenames and dirnames and spaceinformation). Then converting it into numpy arrays it is input into the CNN\_model\_func.

Archietecture :-

(conv1,conv2 and dense have Relu as activation function.)

- INPUTLAYER takes - [batch\_size,32\*32\*3] shaped tensor - outputs [batch\_size,32,32,3] tensor.
- CONV1 takes - [batch\_size,32(image\_width),32(image\_height),3(channels)] tensor – outputs [batch\_size,32,32,16 (number of filters used)]. Kernel\_Size = [5,5] and striding = (2,2)
- POOL1 take – output of INPUTLAYER – outputs [batch\_size,16,16,16] shaped tensor.KernalSize = (2,2) and stride = (2,2)
- CONV2 takes – output of POOL1 – outputs [batch\_size,16,16,32(again number of filters used in conv2)].kernalSize=(3,3) and stride = (1,1)
- POOL2 takes – output of CONV2 – outputs [batch\_size,8,8,32] shaped tensor.kernal and stride as pool1
- POOL2\_FLAT takes – output od POOL2 and returns [batch\_size,8\*8\*32] shaped tensor.
- DENSE takes output of POOL2\_FLAT and outputs [batch\_size,8\*8\*32] shaped tensor.
- DROPOUT takes output of DENSE and outputs [batch\_size,10] shaped tensor.

Gradient Descent is then applied to minimize the loss function if in Train mode.

After that we apply ArgMax to find the class closest to it the new image and SoftMax to find the probabilities of new image being the new class in Predict mode.

- Accuracy Report :- on these 3 batches training took about 20 minutes and obtained 7% error on the test batch.
- Advantages
  1. This model is exeremly fast in training( using CPU computations ) so most Emmbeded Devices can this model.
  2. Even with absolutly no preprocessing of the input, model gives high accuracy again lower the proccessing required for emmbeded systems.
- Disadvantage
  1. Also many details of the image is missed cause of small image size(i.e. 32\*32\*3 pixels for each image)
  2. Outputs only 10 classes so cannot classify a wide range of things with single model.

Future Plans for this implementation :-

Adding script for genetating right sized data from google images links.

Expand this model to CIFAR-100