**CAFFE Deep learning framework**

**Dhruv Bajaj**

**About Caffe :-**

Caffe is one of the renowned deep learning frameworks which was developed by Berkeley AI Research ([BAIR](http://bair.berkeley.edu/)). It is an open source framework for deep learning models and its explicit use. Caffe is purely coded on C++ and Cuda library which makes it fast,compatible and easy to use. Caffe library is integrated with python and matlab interfaces for deep learning practinary .

It is well versed with use of GPU and CPU switching that makes code fast and interprets our deep learning models seamlessly. Caffe has an expressive architecture that encourages application and innovation. It is a seamlessly fast library and is among the fastest convolutional networking implementations available, widely used in the field of research and experiment. It has also been designed for the purposes of speed, open-source ML development, expressive architecture and seamless community support. These features make Caffe framework a popular choice for building Deep Learning models .

Caffe is a library made for machine vision or a forecasting application that lets you build your own deep nets with a sophisticated set of layers configuration options. You can even access premade nets.

Caffe was originally designed for machine vision tasks, so it's well suited for convolutional networks. However, recent versions of the library support speech , text ,reinforcement learning and recurrent nets for sequence processing. Since the library is written in C++ and cuda, the library allows seamless switching between CPU and GPU.

**HyperParameters**:-

With caffe you can build a deepnet by configuring its hyper parameters. In fact, the layer configuration options are very sophisticated. You can create a nete with many different types of layers, such as vision layers, a loss layer, an activation layer and a few others. So each layer can perform different functions or take down a different role. This flexibility allows you to develop extreme complex deepnets for your applications.

**Community and Nets:-**

Caffe supported by a large community where users can contribute their own deepnets as repositories known as “Model Zoo “ . AlexNet and GoogleNet are some popular user made nets.These models are learned and applied for problems ranging from simple regression, to large-scale visual classification, to Siamese networks for image similarity, to speech and robotics applications.

**Caffe in use:-**

Caffe is a revolutionary language in deep learning and is used by companies like Facebook, Samsung, Sony, Amazon, Intel, Adobe, Microsoft, Pinterest for face detection ,objectionable content detection, recommendation and AI features.

**Performance:-**

Caffe vectorizes input data through a special data representation called “ BLOB “. A blob is a type of array that speeds up data analysis and provides synchronization capabilities between CPU and a GPU. Caffe can process **over 60M images per day** with a single NVIDIA K40 GPU\*. That’s 1 ms/image for inference and 4 ms/image for learning and more recent library versions and hardware are faster still.

**Features and applications :-**

* [**Image Classification and Filter Visualization**](http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/00-classification.ipynb)**Instant recognition with a pre-trained model and a tour of the net interface for visualizing features and parameters layer-by-layer.**
* [**Learning LeNet**](http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/01-learning-lenet.ipynb)**Define, train, and test the classic LeNet with the Python interface.**
* [**Fine-tuning for Style Recognition**](http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/02-fine-tuning.ipynb)**Fine-tune the ImageNet-trained CaffeNet on new data.**
* [**Off-the-shelf SGD for classification**](http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/brewing-logreg.ipynb)**Use Caffe as a generic SGD optimizer to train logistic regression on non-image HDF5 data.**
* [**Multilabel Classification with Python Data Layer**](http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/pascal-multilabel-with-datalayer.ipynb)**Multilabel classification on PASCAL VOC using a Python data layer.**
* [**Editing model parameters**](http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/net_surgery.ipynb)**How to do net surgery and manually change model parameters for custom use.**
* [**R-CNN detection**](http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/detection.ipynb)**Run a pretrained model as a detector in Python.**
* [**Siamese network embedding**](http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/siamese/mnist_siamese.ipynb)**Extracting features and plotting the Siamese network embedding.**

**Basics of caffe library:**

* [**Nets, Layers, and Blobs**](https://caffe.berkeleyvision.org/tutorial/net_layer_blob.html)**: the anatomy of a Caffe model.**
* [**Forward / Backward**](https://caffe.berkeleyvision.org/tutorial/forward_backward.html)**: the essential computations of layered compositional models.**
* [**Loss**](https://caffe.berkeleyvision.org/tutorial/loss.html)**: the task to be learned is defined by the loss.**
* [**Solver**](https://caffe.berkeleyvision.org/tutorial/solver.html)**: the solver coordinates model optimization.**
* [**Layer Catalogue**](https://caffe.berkeleyvision.org/tutorial/layers.html)**: the layer is the fundamental unit of modeling and computation – Caffe’s catalogue includes layers for state-of-the-art models.**
* [**Interfaces**](https://caffe.berkeleyvision.org/tutorial/interfaces.html)**: command line, Python, and MATLAB Caffe.**
* [**Data**](https://caffe.berkeleyvision.org/tutorial/data.html)**: how to caffeinate data for model input.**

**Building a simple layer**

CNNs are still basically neural networks, which means they consist of multiple layers joined together. There are many different types of layers that can be used to build a CNN, convolution layer being one of them. Let’s go ahead and see how we can define a simple convolution layer in Caffe.

**Create a python file and add the following lines:**

import sys

import numpy as np

import matplotlib.pyplot as plt

sys.insert('/path/to/caffe/python')

import caffe

**Creating the neural network:**

name: "myconvolution"

input: "data"

input\_dim: 1

input\_dim: 1

input\_dim: 256

input\_dim: 256

layer {

name: "conv"

type: "Convolution"

bottom: "data"

top: "conv"

convolution\_param {

num\_output: 10

kernel\_size: 3

stride: 1

weight\_filler {

type: "gaussian"

std: 0.01

}

bias\_filler {

type: "constant"

value: 0

}

}

}

We just defined a single layer CNN consisting of 10 convolutional neurons (as specified by “num\_output”) with a kernel size of 3×3 (as specified by “kernel\_size”) and a stride of 1 (as specified by “stride”).Caffe handles data as “blobs”, which are basically memory abstraction objects. Our data is contained as an array in the field named ‘data’.

**Interacting with the neural network**

This “net” object contains two dictionaries — net.blobs and net.params.Basically, net.blobs is for data in the layers and net.params is for the weights and biases in the network

**created the “net” object to hold our convolutional neural network:**

**import sys**

**sys.path.insert(0, '/path/to/caffe/python')**

**import caffe**

**import cv2**

**import numpy as np**

**net = caffe.Net('myconvnet.prototxt', caffe.TEST)**

**print "\nnet.inputs =", net.inputs**

**print "\ndir(net.blobs) =", dir(net.blobs)**

**print "\ndir(net.params) =", dir(net.params)**

**print "\nconv shape = ", net.blobs['conv'].data.shape**

**Compute the output for Image:**

****

**img = cv2.imread('input\_image.jpg', 0)**

**img\_blobinp = img[np.newaxis, np.newaxis, :, :]**

**net.blobs['data'].reshape(\*img\_blobinp.shape)**

**net.blobs['data'].data[...] = img\_blobinp**

**net.forward()**

**for i in range(10):**

**cv2.imwrite('output\_image\_' + str(i) + '.jpg', 255\*net.blobs['conv'].data[0,i])**

****

Created a single layer network in Caffe. It is used extensively in deep learning applications built using Caffe.

**Classification of an Image**

Training a full network takes time, so we will use an existing trained model to classify an image for now. There are many models available [here](https://github.com/BVLC/caffe/wiki/Model-Zoo) for tasks such as flower classification, digit classification, scene recognition, and so on. We will be using the caffemodel file available [here](http://dl.caffe.berkeleyvision.org/bvlc_reference_caffenet.caffemodel).

import os

if os.path.isfile(caffe\_root + 'models/bvlc\_reference\_caffenet/bvlc\_reference\_caffenet.caffemodel'):

print 'CaffeNet found.'

else:

print 'Downloading pre-trained CaffeNet model...'

!../scripts/download\_model\_binary.py ../models/bvlc\_reference\_caffenet

### **Load net and set up input preprocessing**

### Set Caffe to CPU mode and load the net from disk.

caffe.set\_mode\_cpu()

model\_def = caffe\_root + 'models/bvlc\_reference\_caffenet/deploy.prototxt'

model\_weights = caffe\_root + 'models/bvlc\_reference\_caffenet/bvlc\_reference\_caffenet.caffemodel'

net = caffe.Net(model\_def,

model\_weights,

caffe.TEST)

**Caffe's caffe.io.Transformer**

**mu = np.load(caffe\_root + 'python/caffe/imagenet/ilsvrc\_2012\_mean.npy')**

**mu = mu.mean(1).mean(1) # average over pixels to obtain the mean (BGR) pixel values**

**print 'mean-subtracted values:', zip('BGR', mu)**

**# create transformer for the input called 'data'**

**transformer = caffe.io.Transformer({'data': net.blobs['data'].data.shape})**

**transformer.set\_transpose('data', (2,0,1)) # move image channels to outermost dimension**

**transformer.set\_mean('data', mu) # subtract the dataset-mean value in each channel**

**transformer.set\_raw\_scale('data', 255) # rescale from [0, 1] to [0, 255]**

**transformer.set\_channel\_swap('data', (2,1,0)) # swap channels from RGB to BGR**

Output:-

mean-subtracted values: [('B', 104.0069879317889), ('G', 116.66876761696767), ('R', 122.6789143406786)]

net.blobs['data'].reshape(50, 3, 227, 227)

image = caffe.io.load\_image(caffe\_root + 'examples/images/cat.jpg')

transformed\_image = transformer.preprocess('data', image)

plt.imshow(image)

Output



# copy the image data into the memory allocated for the net

net.blobs['data'].data[...] = transformed\_image

### perform classification

output = net.forward()

output\_prob = output['prob'][0] # the output probability vector for the first image in the batch

print 'predicted class is:', output\_prob.argmax()

# load ImageNet labels

labels\_file = caffe\_root + 'data/ilsvrc12/synset\_words.txt'

if not os.path.exists(labels\_file):

!../data/ilsvrc12/get\_ilsvrc\_aux.sh

labels = np.loadtxt(labels\_file, str, delimiter='\t')

print 'output label:', labels[output\_prob.argmax()]

Output:-

predicted class is: 281

output label: n02123045 tabby, tabby cat

# sort top five predictions from softmax output

top\_inds = output\_prob.argsort()[::-1][:5] # reverse sort and take five largest items

print 'probabilities and labels:'

zip(output\_prob[top\_inds], labels[top\_inds])

Output :

probabilities and labels:

[(0.31243637, 'n02123045 tabby, tabby cat'),

(0.2379719, 'n02123159 tiger cat'),

(0.12387239, 'n02124075 Egyptian cat'),

(0.10075711, 'n02119022 red fox, Vulpes vulpes'),

(0.070957087, 'n02127052 lynx, catamount'

**Conclusion**

Caffe is considered to be one of the most advanced frameworks for deep learning algorithms and is used widely by tech companies and coders to build a neural framework. Hope you enjoyed reading this blog and got an insight about the caffe framework, how it works, its application and its uses. Go ahead and build something amazing using it!

**References:-**

* <https://caffe.berkeleyvision.org/>
* <https://github.com/BVLC/caffe>
* <https://docs.google.com/presentation/d/1UeKXVgRvvxg9OUdh_UiC5G71UMscNPlvArsWER41PsU/edit#slide=id.gc2fcdcce7_216_0>
* <https://caffe.berkeleyvision.org/installation.html>
* <https://caffe.berkeleyvision.org/tutorial/>
* <https://github.com/DhruvBajaj01/MSRF_Internship/blob/main/Caffe_Work.ipynb>