Project Report for Eklavya at SRA VJTI

ENTER-DARKNET

Sarrah Bastawala | Utkarsh Gandhi

Acknowledgement

We would like to express our gratitude towards our very helpful and insightful seniors i.e the present members of Society of Robotics and Automation (SRA), VJTI for their kind cooperation, helpful resources and timely encouragement along with the proper guidance and help which helped us greatly in completion of this project. Although it seemed almost impossible, putting together our collective effort to reach this milestone and looking forward to many more to come is all thanks to you guys ! We would like to express our special gratitude and thanks to SRA for giving us this opportunity.

Special Thanks to our Superb Mentors:

- Aman Chhaparia
- Prathamesh Tagore

TEAM :

SARRAH BASTAWALA

sarrah.basta@gmail.com

+917208091446

UTKARSH GANDHI

utkarshvg2401@gmail.com

+91 9004446212

TABLE OF CONTENTS:

1. PROJECT OVERVIEW:

1.1 BRIEF IDEA:

The main aim of the project is to familiarise ourselves with the darknet framework and after installing the setup, use it to train a multilayer convolutional neural network that can effectively classify images via deep learning. Thus we use a dataset of many images that is pre labelled and pass it through our network to teach it how to correctly classify an image.

Next, once our model is trained with the huge dataset, it is now ready to be able to correctly classify an image based on the classes that it has been trained on the data set. It shows us the percentage it predicts an image to be among each of its classes.

1.2 DESCRIPTION OF USE CASE AND PROJECT:

Project: Developing a general understanding of the darknet framework and its uses in deep learning and artificial intelligence , thereby applying our knowledge to the use of object classification using a CNN model.

Object classification and object detection has applications like:

1. Helping AI understand and use image data to analyse images For tasks such as facial recognition, motion detection etc that already surpass human-level accuracy.
2. Real life applications such as medical imaging, object identification in satellite images, traffic control systems, brake light detection.

2. INTRODUCTION:

2.1 GENERAL:

Darknet: Open Source Neural Networks in C

Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. Firstly we need to download the dark net framework in our machines and then we can use it to train the convolutional neural network model with specific parameters and hyperparameters suited for our task that we will be using to train a data set for a number of required classes. Once the model is effectively trained using darknet, we will be able to generate a weights file. And after applying the specific weights to the layers in the model we will be able to classify any image according to the classes in our model!

2.2 BASIC PROJECT DOMAINS:

- Deep Learning and Convolutional Neural Networks.
- Image Classification and Object Detection.
- Image Processing.
- Computer Vision.

2.3 THEORY:

We help a device analyse an image in the form of pixels by considering the image as an array of mattresses with the size of the matrix will have the resolution using the open CV library. The statistical data that the computer now has is analysed using algorithms by automatically grouping pixels into specified categories i.e so called classes. These algorithms segregate the image into a series of its most prominent features lowering the workload on the final classifier and give it an idea of what the image represents and what class it might be considered into.

This classification is usually reliant on the data fed to the algorithm and a well optimised classification data set works great in comparison to a bad data set with data imbalance based on class and poor quality of images and annotations . Thus, we leverage the potential of algorithms and convolutional neural networks to learn hidden knowledge from a data set of organised and unorganised samples by using deep learning where a lot of hidden layers are used in a model. Deep learning requires manual data labelling to interpret good and bad samples known as image annotation. The process of creating such labelled data to train a model needs tedious human work . However we have large data sets with millions of high-resolution labelled data of thousands of categories such as ImageNet, LabelMe, Google OID, or MS COCO.

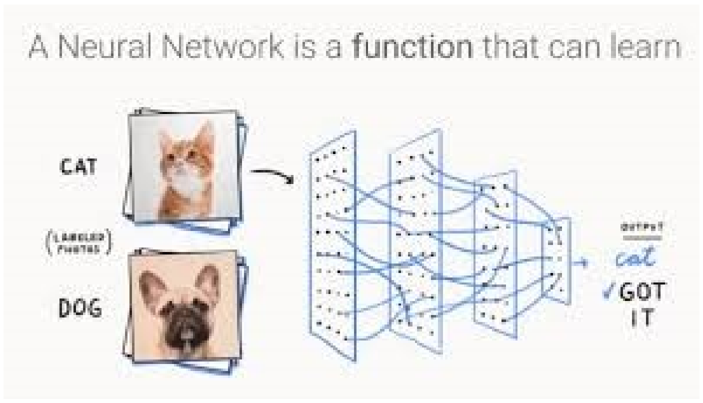
3. TECHNOLOGIES USED:

3.1 OpenCV Library

Opencv is an open source library which is very useful for computer vision applications such as video analysis, CCTV footage analysis and image analysis. OpenCV is written by C++ and has more than 2,500 optimized algorithms. When we create applications for computer vision that we don't want to build from scratch we can use this library to start focusing on real world problems. OpenCV has inbuilt support for Darknet Architecture.

Darknet Architecture is a pre-trained model for classifying 80 different classes. Our goal now is that we will use Darknet(YOLOv3) in OpenCV to classify objects using the Python language.

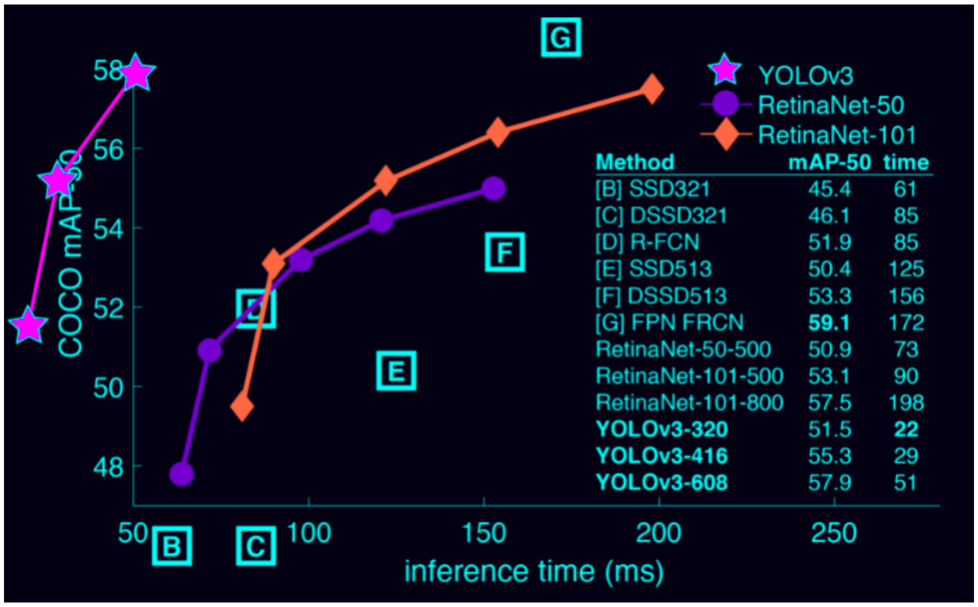
3.2 Convolutional Neural Networks



We use a totally different approach than normal neural networks. We apply a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. CNN incorporates changes in its mode of operations. The inputs of CNN are not fed with the complete numerical values of the image. Instead, the complete image is divided into a number of small sets with each set itself acting as an image. A small size of filter divides the complete image into small sections. Each set of neurons is connected to a small section of the image. This process repeats until the complete image in bits size is shared with the system. The result is a large Matrix, representing different patterns the system has captured from the input image which is again downsampled (reduced in size) with a method known as Max-Pooling. It extracts maximum values from each sub-matrix and results in a matrix of much smaller size.

3.3 YOLO: Real-Time Object Detection

You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev. Prior detection systems repurpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered detections.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

3.4 Numpy and Python

NumPy library is an important foundational tool for studying Machine Learning. Many of its functions are very useful for performing any mathematical or scientific calculation. In some ways, NumPy arrays are like Python's built-in list type, but NumPy arrays provide much more efficient storage and data operations as the arrays grow larger in size.An image is essentially a standard NumPy array containing pixels of data points. Therefore, by using basic NumPy operations, such as slicing, masking, and fancy indexing, you can modify the pixel values of an image.

4. WORKFLOW:

4.1 SETTING UP DARKNET:

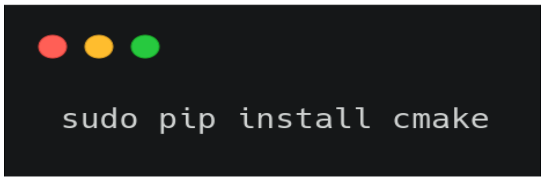
For Linux:

Prerequisites:

Install python3 (it's preinstalled on ubuntu).

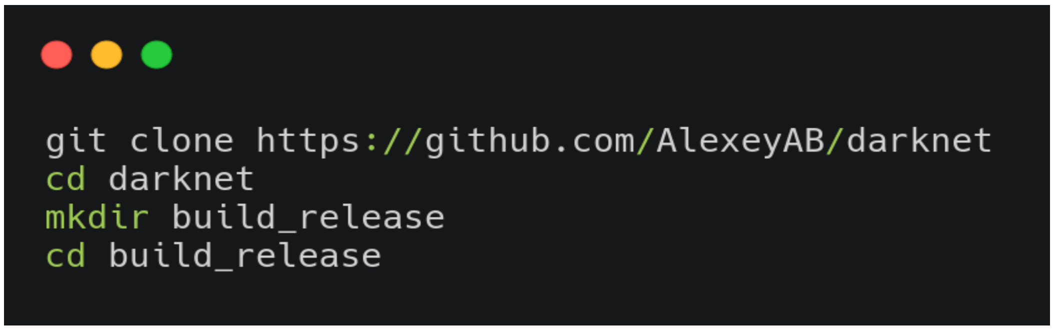
Install pip.

Install cmake using:



```
sudo pip install cmake
```

Commands:



```
git clone https://github.com/AlexeyAB/darknet
cd darknet
mkdir build_release
cd build_release
```

Now if you are doing it via GPU (CUDA):

Before make, you can set such options in the `Makefile (present in the root darknet directory of your cloned repository)`: link

- `GPU=1` to build with CUDA to accelerate by using GPU (CUDA should be in `/usr/local/cuda`)
- `CUDNN=1` to build with cuDNN v5-v7 to accelerate training by using GPU (cuDNN should be in `/usr/local/cudnn`)
- `CUDNN_HALF=1` to build for Tensor Cores (on Titan V / Tesla V100 / DGX-2 and later) speedup Detection 3x, Training 2x

- `OPENCV=1` to build with OpenCV 4.x/3.x/2.4.x - allows to detect on video files and video streams from network cameras or web-cams
- `DEBUG=1` to build debug version of Yolo
- `LIBSO=1` to build a library `darknet.so` and binary runnable file `uselib` that uses this library. Or you can try to run so `LD_LIBRARY_PATH=./:$LD_LIBRARY_PATH ./uselib test.mp4` How to use this SO-library from your own code - you can look at C++ example: https://github.com/AlexeyAB/darknet/blob/master/src/yolo_console_dll.cpp or use in such a way: `LD_LIBRARY_PATH=./:$LD_LIBRARY_PATH ./uselib data/coco.names cfg/yolov4.cfg yolov4.weights test.mp4`

If it's via CPU:

Install openCV first using:

```
sudo apt update
sudo apt install libopencv-dev python3-opencv
```

Then:

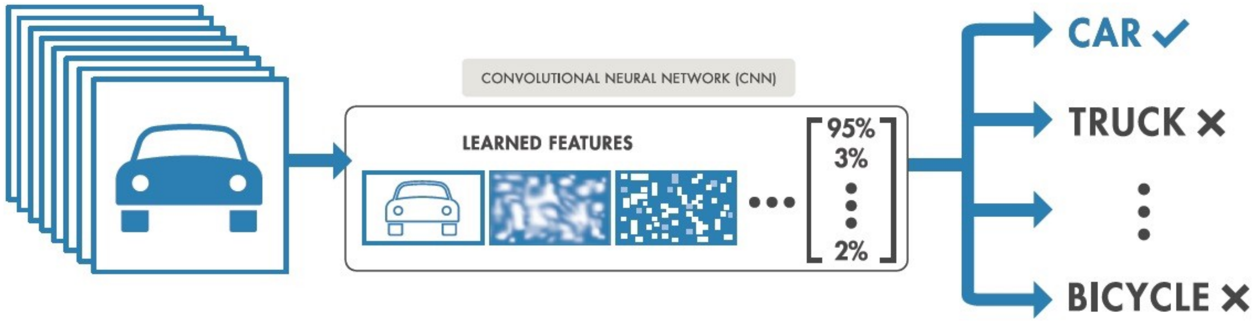
```
cmake ..-DENABLE_CUDA=OFF
```

And the final command is:

```
make -j4
```

4.2 APPROACH:

The approach of the project is as described before. First of all it's aim is to effectively classify an image among the specified classes according to the provided data set. The proposed project consists of a model that is well researched and comprises various layers such as convolutional pooling and softmax.



By using convolutional neural networks we make an effective trainable model using these layers and then supply it to the darknet along with our data set and label files where we can train the model using deep learning. Once our model is effectively trained, it will be able to identify, distinguish and correctly label any image fed to it in its data.

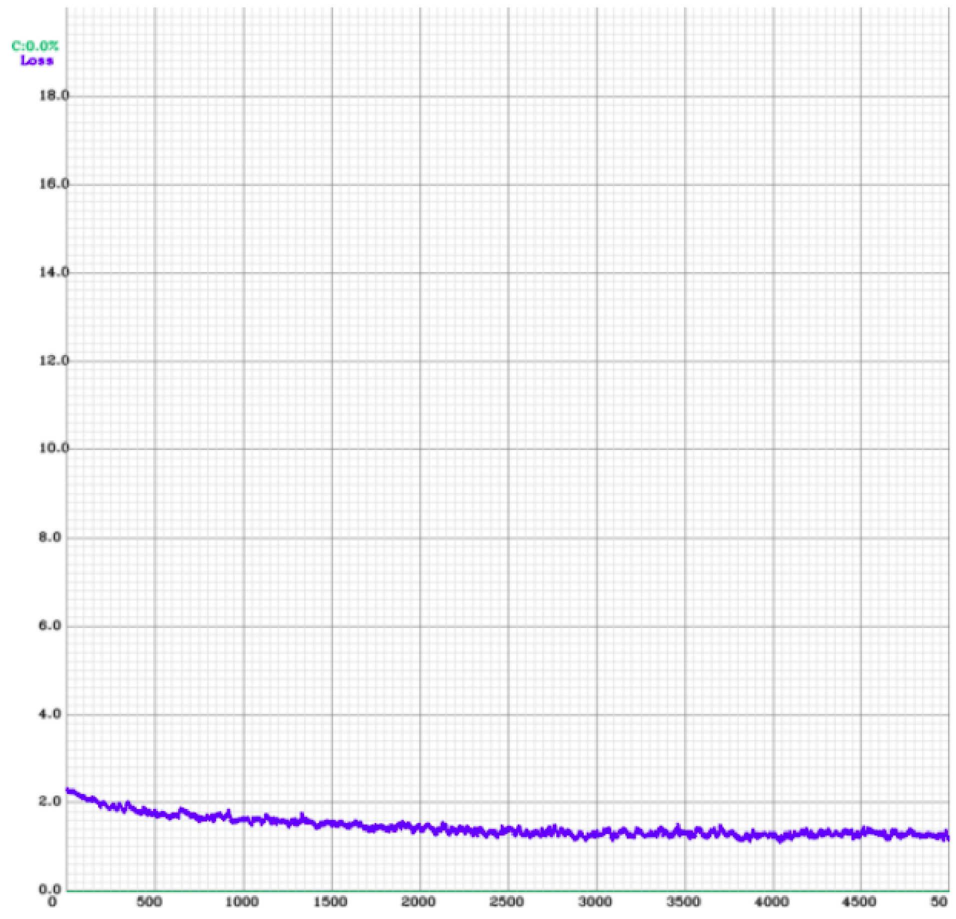
To train the model we pass a config file, a data file including the paths of our data set and labels to a darknet train command. We then use it to generate a weights file. Which is passed along with the configured data file during the testing of any image and applies the correct values in the model. Once the model is trained and an image is passed on to it for testing, it classifies the image according to the output of its final softmax layer and prints the probability or percentage of the image being each of the given labels.

4.3 CNN MODELS

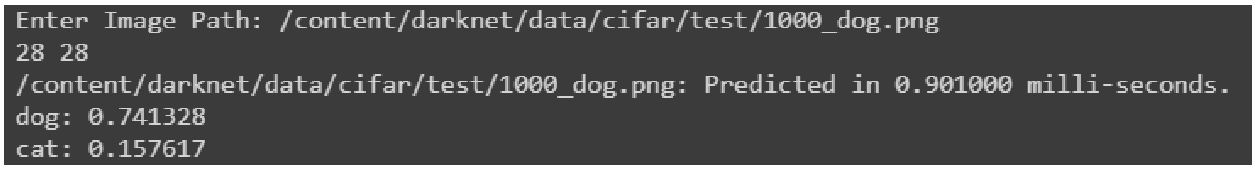
4.3.1 Pre-existing Model:

First we train our model using a pre-existing CNN image classification model using the CIFAR-10 dataset. 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. We will be training our model with 5000 images.

Training Graph:



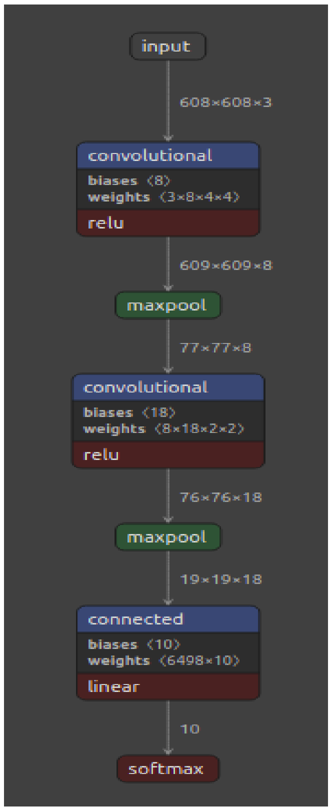
Output:



```
Enter Image Path: /content/darknet/data/cifar/test/1000_dog.png
28 28
/content/darknet/data/cifar/test/1000_dog.png: Predicted in 0.901000 milli-seconds.
dog: 0.741328
cat: 0.157617
```

4.3.2 Creating our own model

Now, we create our own model for the CIFAR-10 dataset and use a config (.cfg) file of the model to train the dataset. The image below shows our model.



This model is inspired by the week 2 assignment in the course Convolutional Neural Networks by Andrew NG on Coursera.

It is a very small and simple model for image classification.

It consists of 2 convolutional layers and 2 max pooling layers and a connected layer. The last connected layer has 10 filters because we have 10 classes. This will give us our 10 predictions. We use a softmax to convert the predictions into a probability distribution.

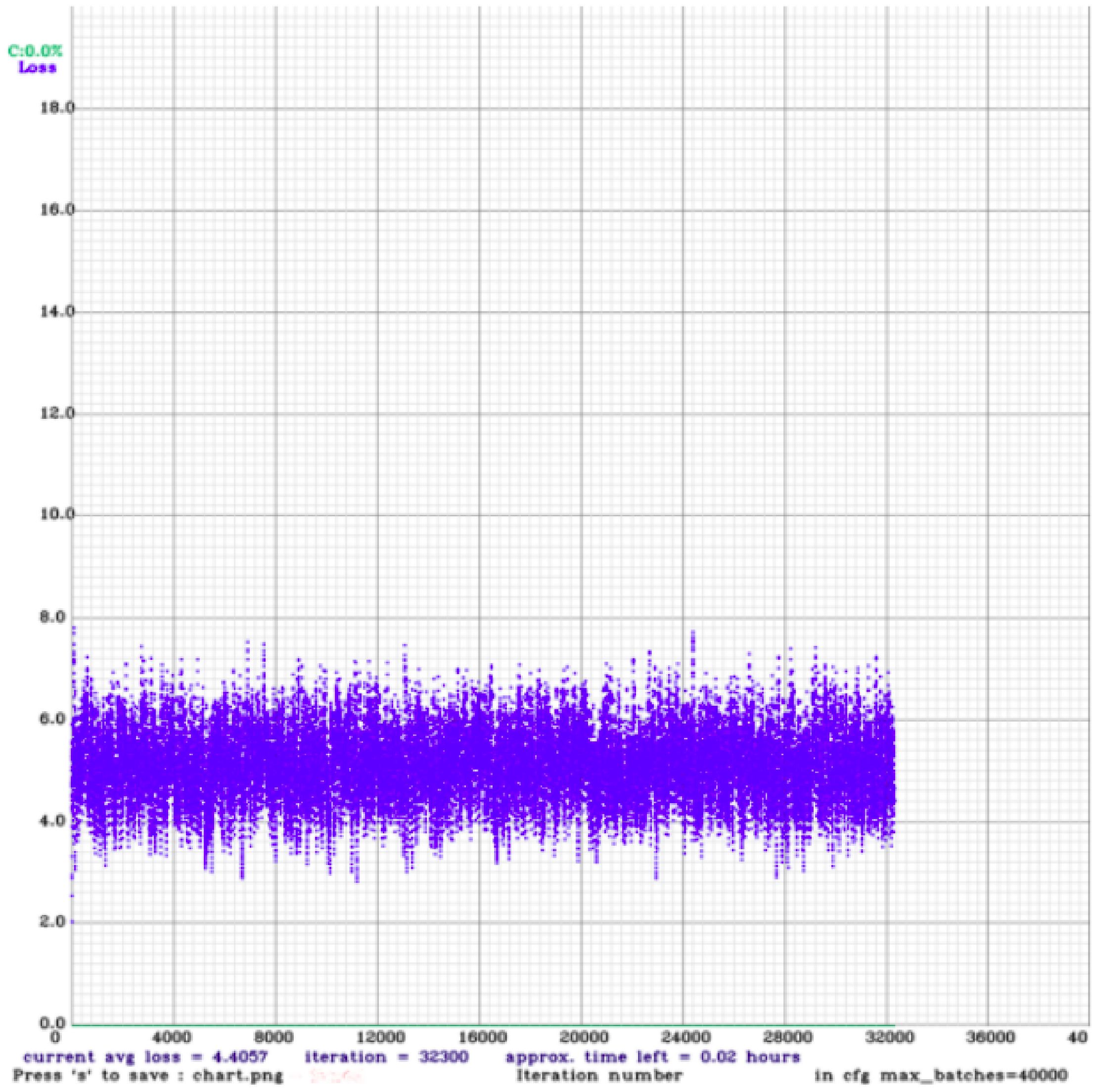
The accuracy of the model for classification of images is not that good as it is not complex or deep.

We made this model to help us understand the working of a config file and to build our basics to make a better and more complex model.

Understanding a config file:

In computing, configuration files (commonly known simply as config files) are files used to configure the parameters and initial settings for some computer programs. They are used for user applications, server processes and operating system settings.

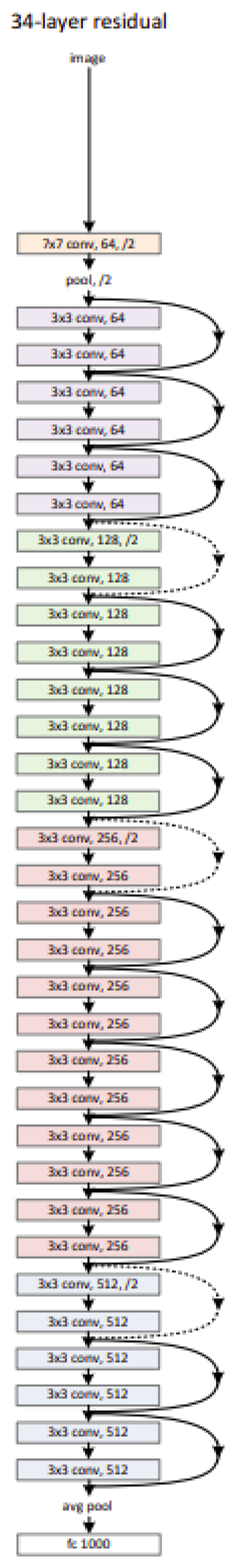
Training Graph:



Output:

```
Enter Image Path: /content/darknet/data/cifar/test/101_dog.png
28 28
/content/darknet/data/cifar/test/101_dog.png: Predicted in 0.609000 milli-seconds.
dog: 0.103659
deer: 0.103467
```

4.3.3 Making a 34-layer residual network



This image shows the 34-layer residual network we will be using for our model for image classification on the CIFAR-10 dataset.

It is inspired by the 'Deep Residual Learning for Image Recognition' research paper written by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun of Microsoft Research.

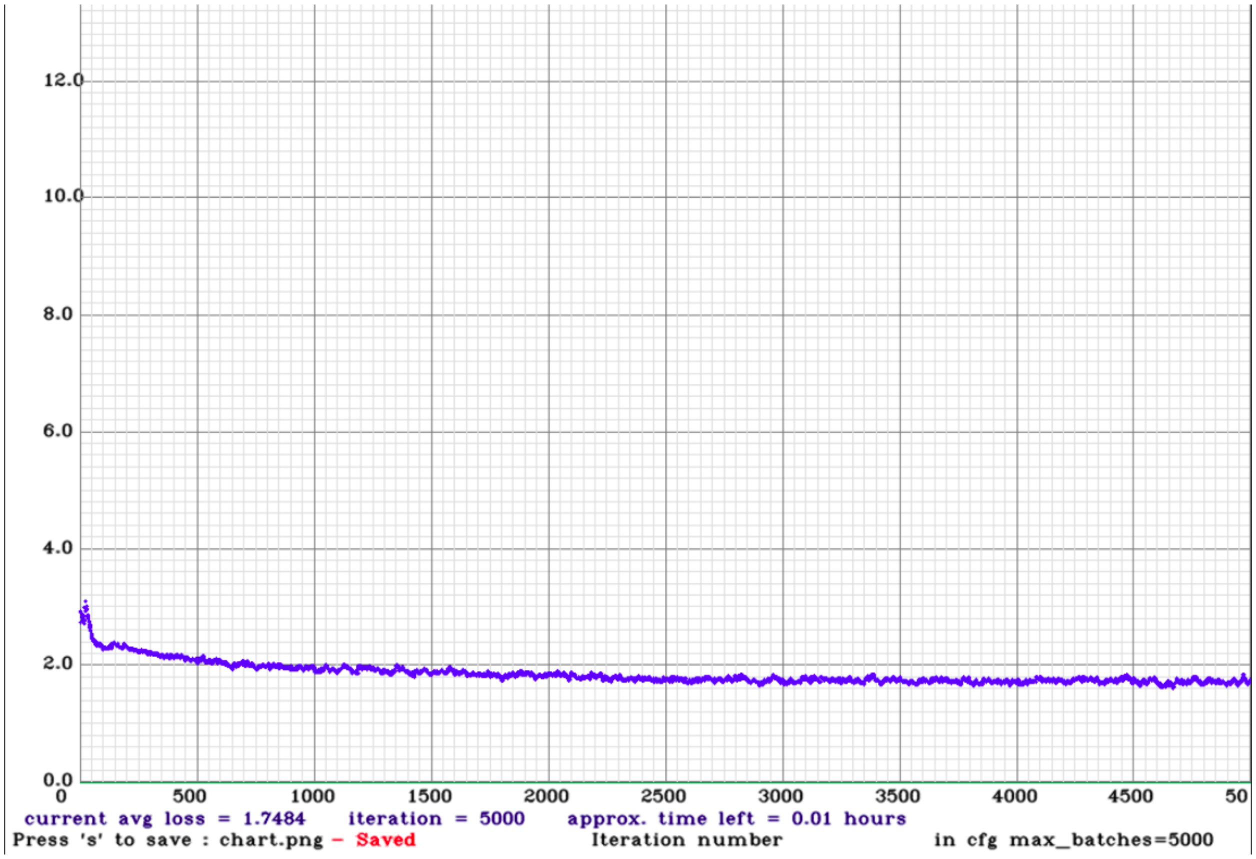
Deep convolutional neural networks have achieved the human level image classification result. When the deeper network starts to converge, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Such degradation is not caused by overfitting or by adding more layers to a deep network leads to higher training error. The deterioration of training accuracy shows that not all systems are easy to optimize.

To overcome this problem, Microsoft introduced a deep residual learning framework. Instead of hoping every few stacked layers directly fit a desired underlying mapping, they explicitly let these layers fit a residual mapping. Shortcut connections are those skipping one or more layers. The shortcut connections perform identity mapping, and their outputs are added to the outputs of the stacked layers. By using the residual network, there are many problems which can be solved such as:

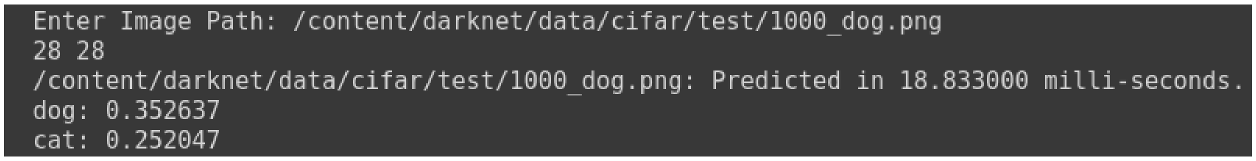
ResNets are easy to optimize, but the "plain" networks (that simply stack layers) show higher training error when the depth increases. ResNets can easily gain accuracy from greatly increased depth, producing results which are better than previous networks.

Our model has 34 convolutional layers with one max pooling layer and multiple shortcuts.

Training Graph:



Output:



```
Enter Image Path: /content/darknet/data/cifar/test/1000_dog.png
28 28
/content/darknet/data/cifar/test/1000_dog.png: Predicted in 18.833000 milli-seconds.
dog: 0.352637
cat: 0.252047
```

5.PROJECT CONCLUSION AND FUTURE WORK:

5.1 What did we achieve so far

As explained in the name of our project 'enter-darknet', our main aim was to explore and understand the darknet framework, and be able to use it in order to train any custom data set via machine learning. Right from the basics of installations in different systems, to exploring prewritten codes for object detection in images and videos, trying in testing different models to train our chosen data set to achieve the best accuracy, we have successfully been able to make the best use of this to take our understanding of machine learning to a deeper level.

As shown above, after comparing the various accuracies that our predictions gave us, we conclude that choosing the 34 layer resnet model is the best and most optimum choice for training on the Cifar - 10 dataset via darknet.

5.2 Future Prospects

Even though we have chosen an extremely small dataset and a simple training model keeping in mind our limited resources and computational power, while training from scratch, Darknet in itself is a high performance open source framework for the implementation of neural networks. Written in C and CUDA, it can be integrated with CPUs and GPUs,and show exemplary performance at both when used correctly.

Advanced implementations of deep neural networks can be done using Darknet. These implementations include You Only Look Once (YOLO) for real-time object detection, ImageNet classification, recurrent neural networks (RNNs), and many others.

However, as being able to work without wasting a lot of computational power is an ultimate aim, when we apart from the most obvious path of choosing deeper,larger neural networks to train huge data sets, is correct hyperparameter tuning that results in the best accuracy and lowest training time possible. New year resorts in the and machine learning comes for everyday and testing these newly proposed models using our framework so as to advance this field in the community would be a constant aim.

6. References:

Drive:

https://drive.google.com/drive/u/0/folders/1RWdPRY5_vRV82a5bEDPjy3V8FqGq6T5Z

Github :

Link to Repo : https://github.com/Utkarsh2401/Enter_Darknet
Reference : https://guides.github.com/features/issues/

Courses:

1. 3b1b NN playlist (https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi)
2. 3b1b LA playlist (https://www.3blue1brown.com/topics/linear-algebra)
3. Neural Networks and Deep Learning by DeepLearning.AI (Andrew NG) (https://www.coursera.org/learn/neural-networks-deep-learning/home/welcome)
4. Convolutional Neural Networks by DeepLearning.ai (Andrew NG): https://www.coursera.org/learn/convolutional-neural-networks/home/welcome

Additional Resources Referred:

Research Paper: https://arxiv.org/pdf/1512.03385.pdf

Dual Boot: https://www.youtube.com/watch?v=-iSAyiicyQY
Darknet: https://github.com/AlexeyAB/darknet
https://analyticsindiamag.com/guide-to-darknet-installation-on-windows-10-cpu-version/
Deep Learning and Neural Networks:
https://medium.com/@safrin1128/weight-initialization-in-neural-network-inspired-by-andrew-ng-e0066dc4a566

Layers in the model:
https://suelan.github.io/2019/05/08/The-Implementation-of-Convolutional-and-MaxPool-layer/

Config files:
https://github.com/AlexeyAB/darknet/wiki/CFG-Parameters-in-the-different-layers
https://github.com/cvjena/darknet/blob/master/cfg/yolo.cfg (explanation of cfg file using comments)

Tutorials to train custom models:
https://pjreddie.com/darknet/train-cifar/
https://colab.research.google.com/github/luxonis/depthai-ml-training/blob/master/colab-notebooks/Easy_TinyYolov3_Object_Detector_Training_on_Custom_Data.ipynb

EKLAVYA MENTORSHIP PROGRAMME

At

SOCIETY OF ROBOTICS AND AUTOMATION,
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
MUMBAI

SEPTEMBER 2021