Does processing in the canine vision spectrum retain performance?

Team members

- Aaryan Gupta (19D070001)
- Eeshaan Jain (19D070022)
- Vipin Singh (19D070069)

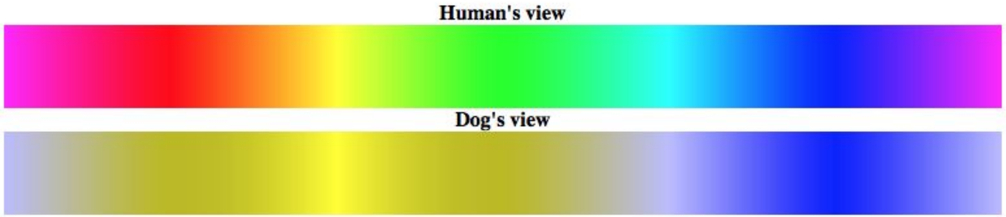
Summary

Introduction

One of the fundamental tasks of computer vision can be stated to be image classification. For now, the approaches to image-related tasks for such robots are done in the RGB spectrum, but we explore the possibilities of doing the same in the canine vision spectrum. We aim to review popular classification models and test the performance of the models in the canine vision spectrum. We found out that DenseNet performs marginally better in the canine vision spectrum in both - low parameter and high parameter settings. In ResNet and EfficientNet_b1, the RGB data performed slightly better than the transformed data.

CANINE VISION SPECTRUM

- Dogs have a dichromatic vision similar to deuteranopes vision
- Deuteranopia is red-green color blindness characterized by the inability to distinguish red and green pigments.



Human and Dog perception of color Credits: UCSB

Data Transformation

- Converting an image from a normal person's vision to a canine (deuteranope) vision involves linear intensity transformation and residual clipping.
- In LMS color space, modeling color deficiency is an easy task, so we first convert the color space from RGB to LMS.

RGB color space to XYZ color space to LMS color space

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3756 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The matrix transformation given left converts an Image from RGB color space into XYZ color space.

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.1551 & 0.5431 & -0.03286 \\ -0.1551 & 0.4568 & 0.03286 \\ 0 & 0 & 0.01608 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The matrix transformation given left converts an Image from XYZ color space into LMS color space.

Normal Vision to Canine vision in LMS color space to RGB color space

$$\begin{bmatrix} L_d \\ M_d \\ S_d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0.49 & 0 & 1.25 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

The transformation given left is used to convert the image from the human perceived color spectrum to canine perceived color spectrum in LMS color space.

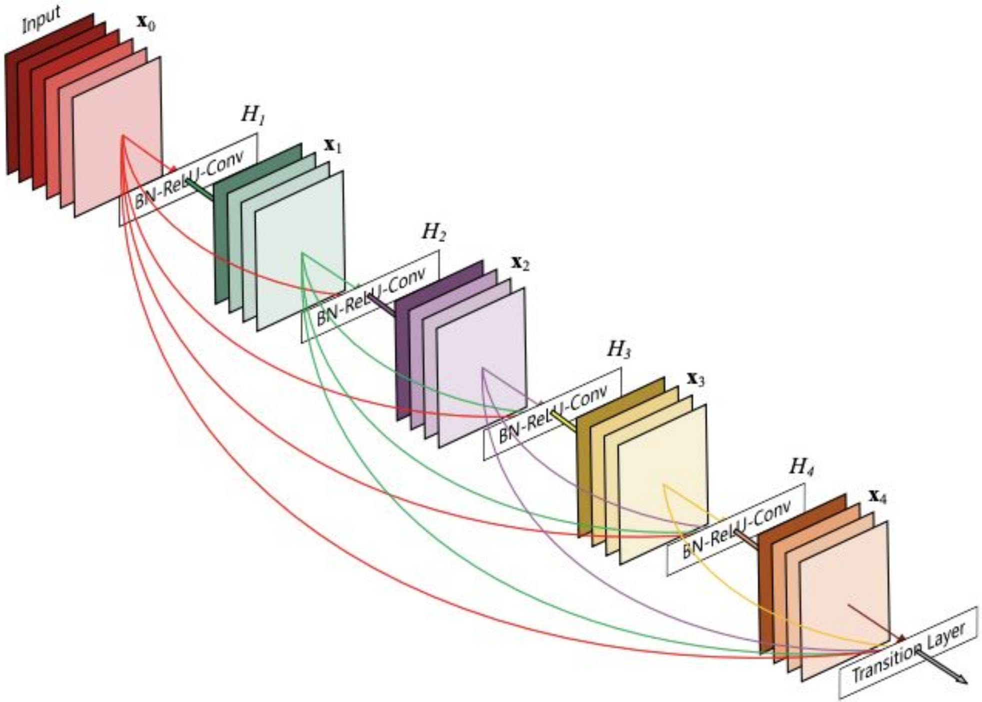
$$\begin{bmatrix} R_d \\ G_d \\ B_d \end{bmatrix} = \begin{bmatrix} 8.0655 & -13.1980 & 12.2918 \\ -0.9953 & 5.3625 & -10.3650 \\ -0.0380 & -0.4059 & 66.4792 \end{bmatrix} \begin{bmatrix} L_d \\ M_d \\ S_d \end{bmatrix}$$

The matrix transformation given above converts the image in LMS color space to RGB color space. Now applying residual clipping to the intensity level (0-255), the image perceived by a dog is obtained.

Deep Learning Models

DenseNet-121

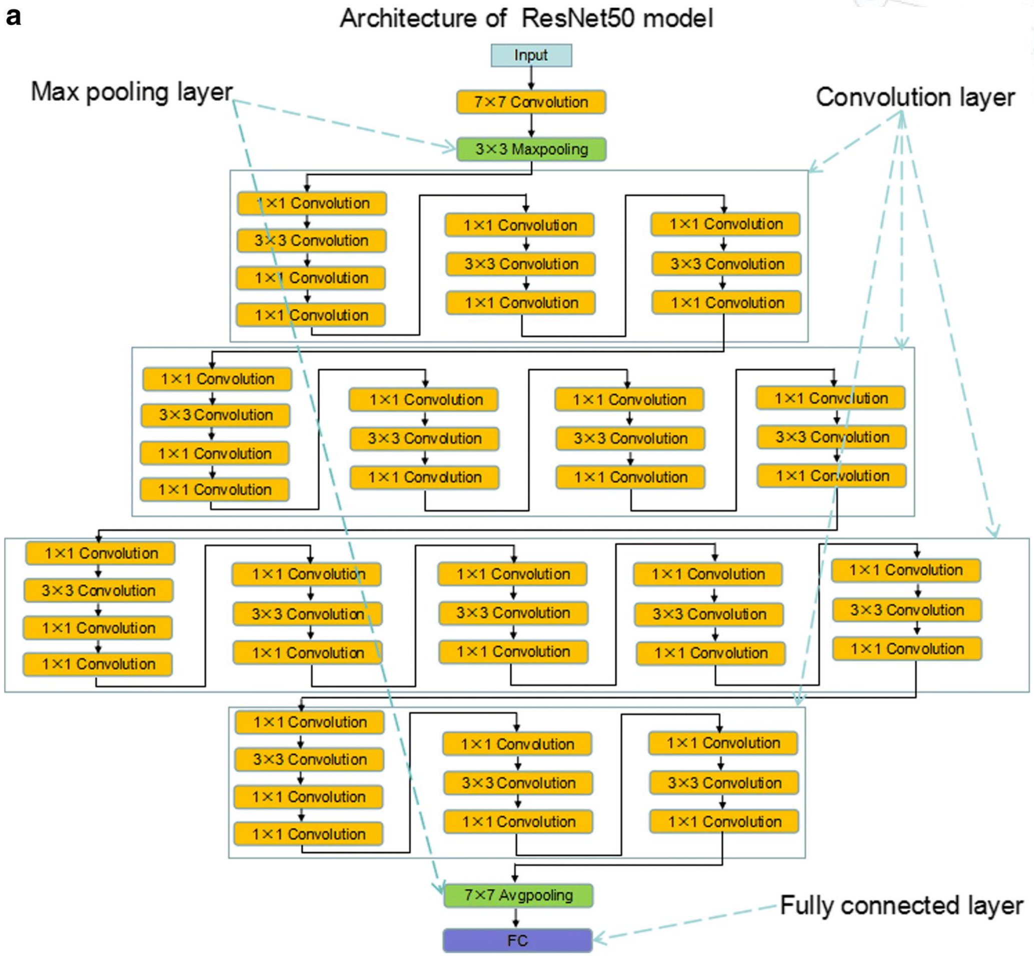
- Each layer in the model receives input from all preceding layers, and the layer passes on its feature maps to all subsequent layers.
- DenseNets are divided into Dense Blocks where the size of feature maps does not change within the block, but the number of filters between them changes.



DenseNet-121 Architecture Model, Credits: Pytorch

ResNet-50

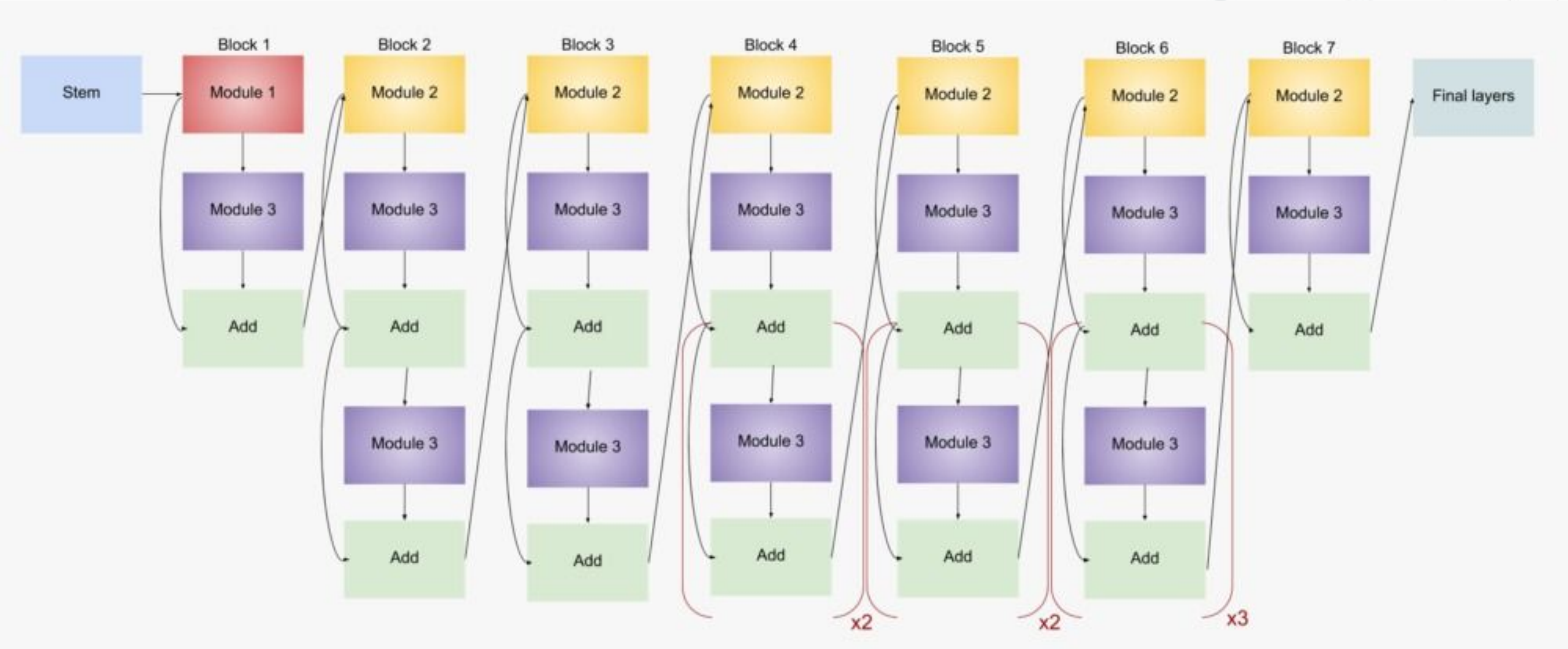
- It is an artificial neural network that builds a network by stacking residual blocks on top of one another.
- It uses residual blocks and skip-connections.
- ResNet-50 is a deep learning model that is 50 layers deep.
- Architecture is similar to ResNet-34. The significant difference is that the building block is modified into a bottleneck design in ResNet-50 model.



ResNet-50 model Architecture, Credits - Springer

EfficientNet-B1

- CNN architecture and scaling method that uniformly scales all dimensions of network width, depth, and resolution with a fixed set of scaling coefficients.
  - For example, if we want to use 2^N more computational resources, then we can simply increase the network depth by α^N, width by β^N, and image size by γ^N where α, β, γ are constant coefficients determined by small grid search on the original small model.
- EfficientNet uses a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way.



EfficientNet-B1 Architecture

Adam Optimizer

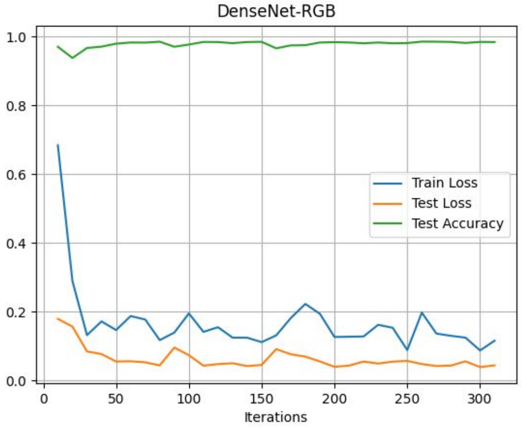
- First-order gradient-based technique for optimizing stochastic objective functions based on adaptive estimations of lower-order moments.
- simple to implement, computationally efficient, has low memory requirements, is insensitive to gradient re-scaling, and is well suited for vital data or parameters problems.

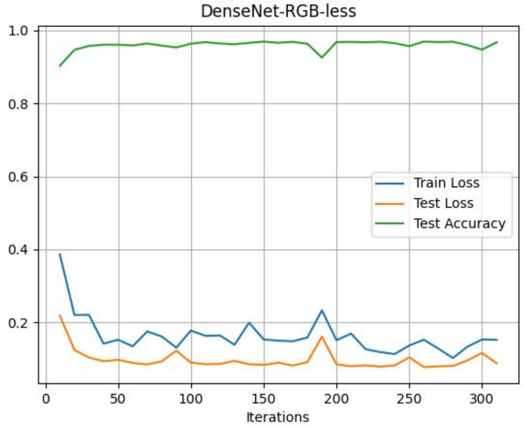
$$m_t = \beta_1 + (1 - \beta_1)\left[\frac{\delta L}{\delta w_t}\right]$$

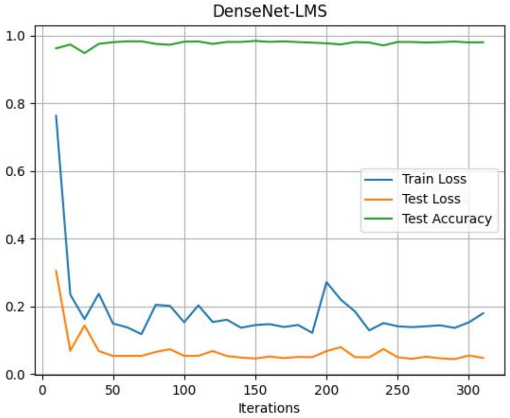
$$v_t = \beta_2 + (1 - \beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2$$

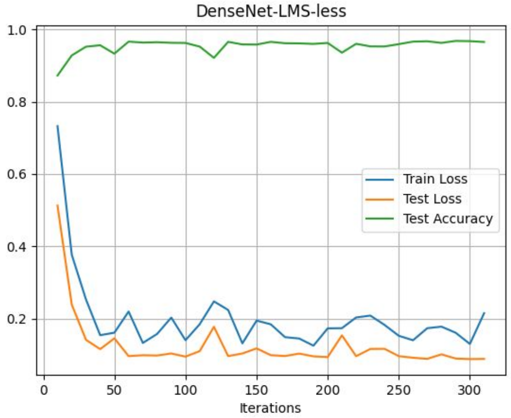
Experiment/Results

DenseNet-121

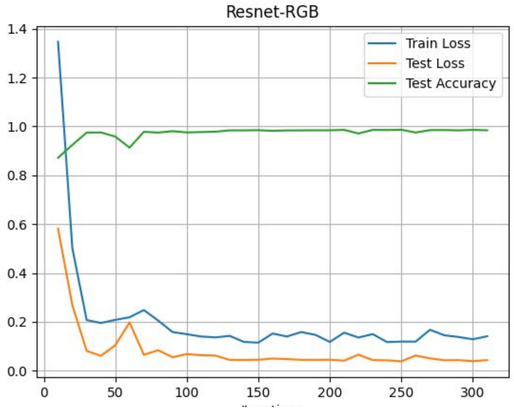


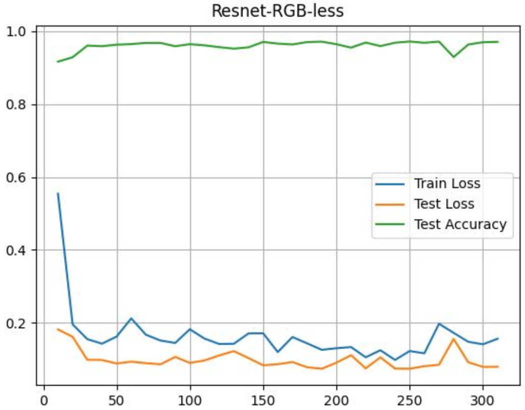


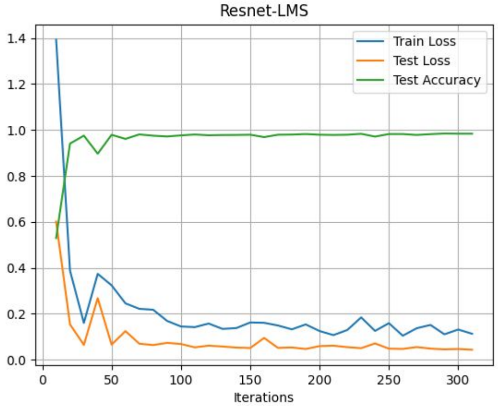


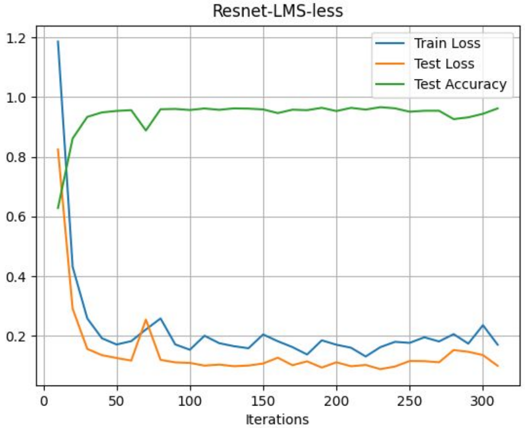


ResNet-50

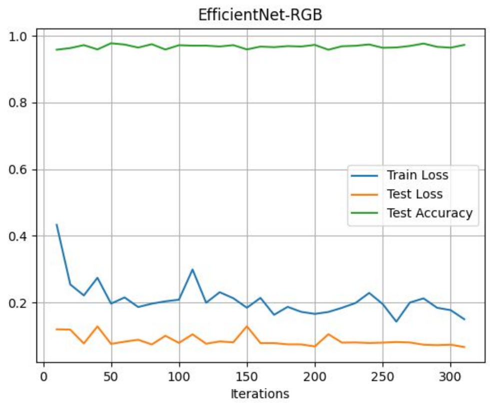

Resnet-RGB
Train Loss
Test Loss
Test Accuracy

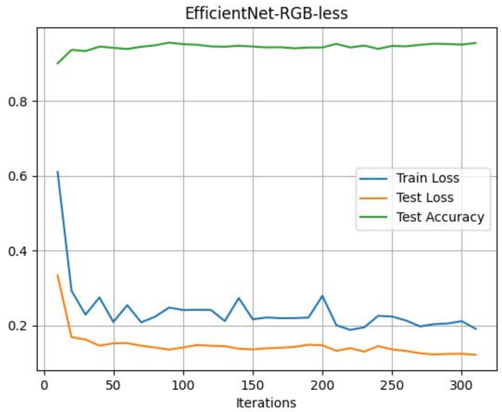

Resnet-RGB-less
Train Loss
Test Loss
Test Accuracy

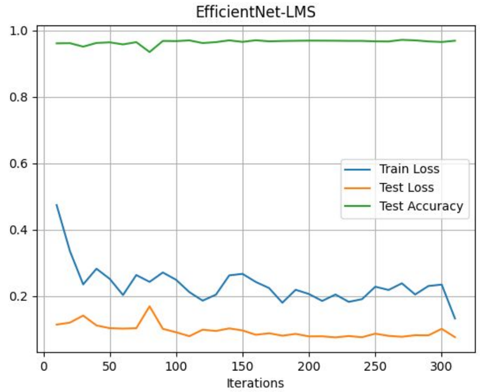

Resnet-LMS
Train Loss
Test Loss
Test Accuracy
Iterations

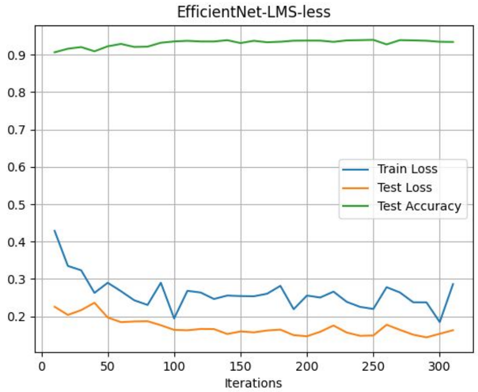

Resnet-LMS-less
Train Loss
Test Loss
Test Accuracy
Iterations

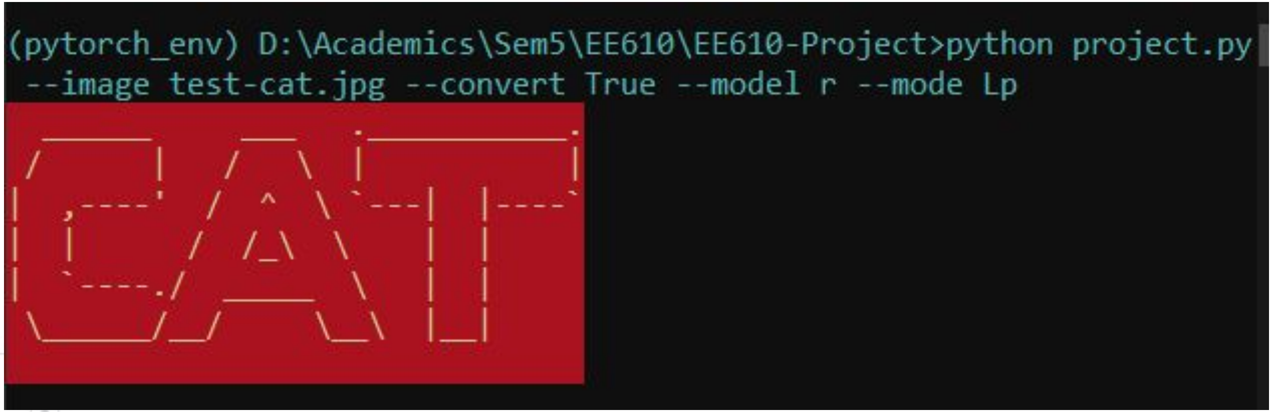
EfficientNet-B1










(pytorch_env) D:\Academics\Sem5\EE610\EE610-Project>python project.py
--image test-cat.jpg --convert True --model r --mode Lp

Contributions in Code

- Aaryan Gupta - DenseNet-121
- Eeshaan Jain - ResNet-50, Pipeline Model, Command Line Interface for Image Classification
- Vipin Singh - EfficientNet-B1, Data Transformation

Challenges faced

- Working with pytorch was a challenging task as it involved figuring out problems & debugging hundreds of lines of code.
- Literature Review proved to be challenging as we had to go through a huge amount of literature in an optimized way so as to know about the latest techniques employed for image classification & how they could be improved upon in relation to our project.
- Testing of CNNs on various color spaces hasn’t been explored much & hence was challenging to find relevant literature.
- Dealing with a huge amount of parameters caused laptop overloading that affected productivity.