Multi-Label Classification

Project Proposal – Group2

Author(s): Hemangi Kinger, Varun .R. Shah, Ishan Kuchroo

12/13/2022

Abstract

We can define Deep Learning as a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. It is a technique that teaches computers to do what comes naturally to humans: learn by example. The final project details the implementation of tools, libraries and techniques acquired during the course.

In this project, the goal is to implement multi-label image classification using Convoluted Neural Networks. The project includes multiple methods and procedures used including our CNN trained model, along with fine-tuned pretrained models EfficientNet, Resnet-101 and RegNet_x_800.

For Metrics, we use the Accuracy Scores and Hamming Loss to evaluate our best model.

Introduction

Multi-label image classification is the task of predicting a set of labels corresponding to objects, attributes or other entities present in an image. Unlike normal classification tasks where class labels are mutually exclusive, multi-label classification requires specialized machine learning algorithms that support predicting multiple mutually non-exclusive classes or "labels."

The goal of this project is to implement multi-label classification using Convoluted Neural Networks along with other pretrained models based on CNN architecture such as EfficientNet, Resnet-101 and RegNetX.

The data consists of approximately 10,000 images, with each image having between 0 to 16 different labels. Its taken from the IDD (Indian Driving Dataset) dataset, which is a dataset for road scene understanding in unstructured environments used for semantic segmentation and object detection for autonomous driving. The use case for this project is to accurately label the objects on an image which could be further used for object identification or semantic segmentation useful for Autonomous Driving

We use different fine-tuning methods such as data augmentation, changes in model definition, changes in optimization algorithm, changes in batch size and other parameters.

For metrics, we use Accuracy Score and Hamming Loss.

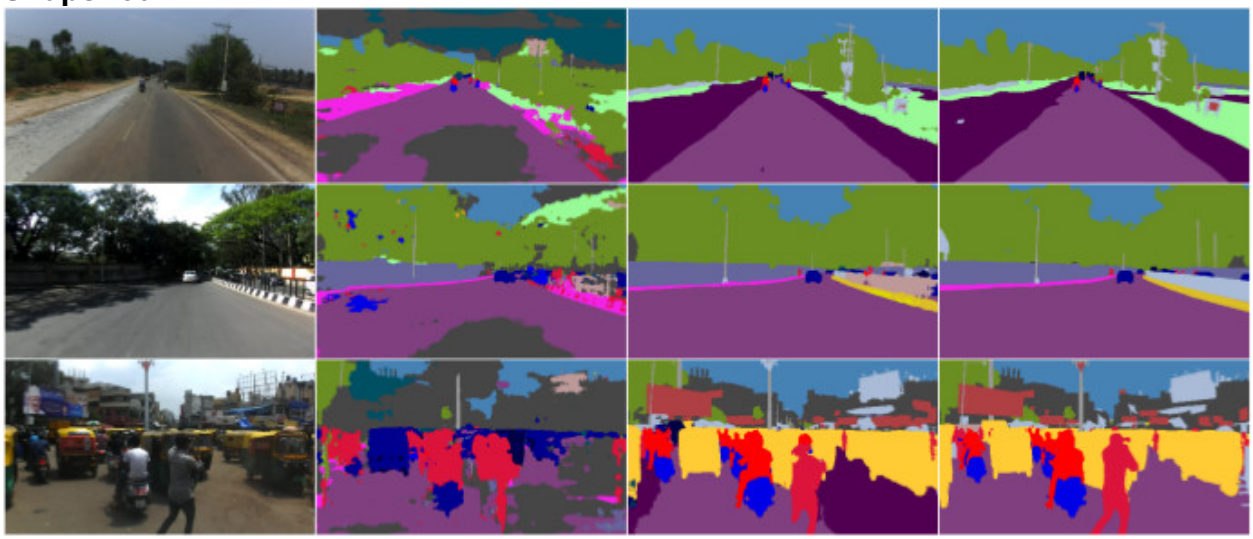
Data Overview

Overview:

IDD consists of 10,004 images, finely annotated with 34 classes collected from 182 drive sequences on Indian roads. The label set is expanded in comparison to popular benchmarks such as Cityscapes, to account for new classes. It also reflects label distributions of road scenes significantly different
from existing datasets, with most classes displaying greater within-class diversity. Consistent with real driving behaviors, it also identifies new classes such as drivable areas besides the road. The dataset is inspired from the one used in the 2018 paper *IDD: A Dataset for Exploring Problems of Autonomous Navigation in Unconstrained Environments*

Source:
http://idd.insaan.iiit.ac.in/dataset/details/

Snapshot:



Computer Vision – Methods and Procedures

For this project, we model unstructured environments using Convolution Neural Networks, and show that they achieve state-of-the-art performance.

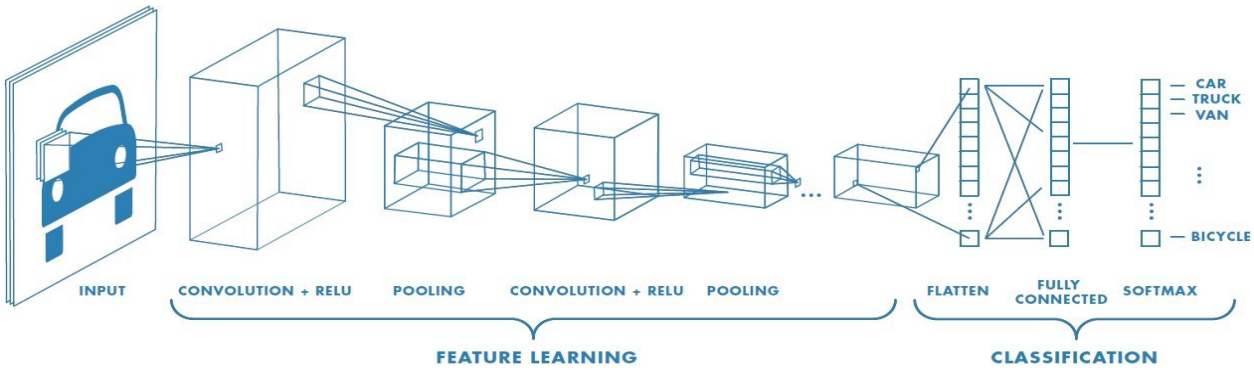
Convolution Neural Networks:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. A CNN can capture the Spatial and Temporal intricacies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights.

The design of a basic Convolutional Neural Network (CNN) starts with an Image Input Layer (dimensions: height, width, channels), subsequently convolved with a combination of some convolutional filters, max or average pooling layers followed by a fully connected layer and in the end, a Softmax or a Sigmoid output function for class label prediction.

Convolution is a specialized type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map.

Two key hyperparameters that define the convolution operation are size and number of kernels.

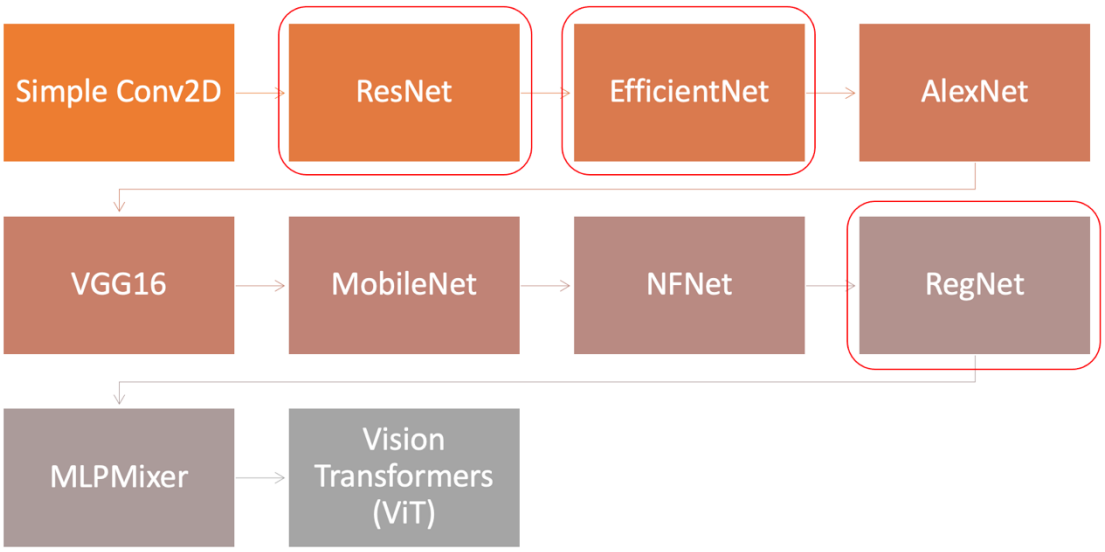


Reference: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Framework:

PyTorch
PyTorch is a machine learning framework based on Torch library, used for applications such as computer vision and natural language processing. We'll be using various methods from Pytorch like "trainsform.resize", "cast" etc.
Reference: https://pytorch.org/

Model Journey:



The Top 3 Models

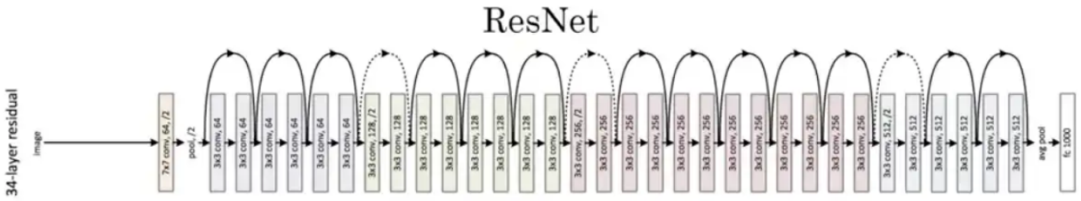
ResNet101:

ResNet or Residual Network is an innovative neural network that was first introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their 2015 computer vision research paper titled 'Deep Residual Learning for Image Recognition'. ResNet-101 is a convolutional neural network that is 101 layers deep.

To understand ResNet, we have to look at LeNet, AlexNet and VGGNet-16/19 which was developed by increasing the number of layers on the base CNN model. VGGNet was trained on ImageNet dataset deepest these three architectures with 138 million training parameters.

As we increase the depth of these networks further to make the model more robust and enhance its performance, according to theory, the error rate while training and testing, should keep decreasing as we go deeper in a network, however, instead of steadily decreasing after attaining a minimum value, the error rate starts increasing again. This happens due to the **exploding and vanishing gradient descent** problem which also causes overfitting of the model, hence increasing the error.

Residual Networks solves this problem as they use a skip connection or a "shortcut" between every two layers along with using direct connections among all the layers. One block of such a connection is called the **"residual block"**, these are stacked on top of one another in a ResNet to maintain efficient learning of parameters from the identity function, even in much deeper layers.



Reference: https://viso.ai/deep-learning/resnet-residual-neural-network/

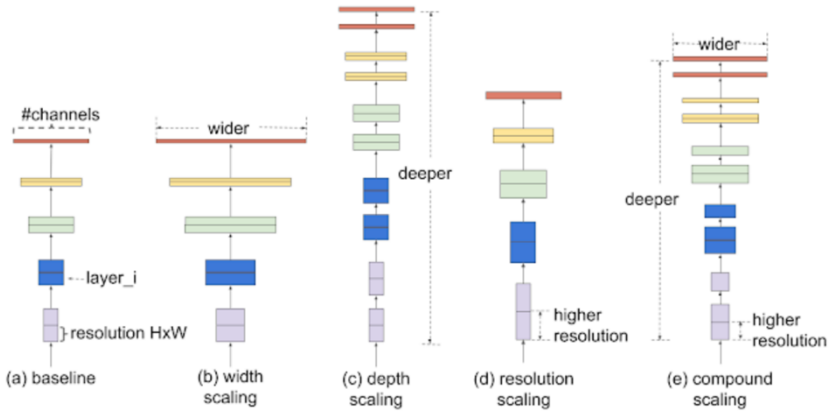
EfficientNet:

The conventional practice for model scaling is to arbitrarily increase the CNN depth or width, or to use larger input image resolution for training and evaluation. While these methods do improve accuracy, they usually require tedious manual tuning, and still often yield suboptimal performance. For example, ResNet can be scaled up from ResNet-18 to ResNet-200 by increasing the number of layers.

EfficientNet is an architecture that includes model scaling method that uses a simple yet highly effective *compound coefficient* to scale up CNNs in a more structured manner. Unlike conventional approaches that arbitrarily scale network dimensions, such as width, depth and resolution, EfficientNet method uniformly scales each dimension with a fixed set of scaling coefficients.

While scaling individual dimensions improves model performance, the architects of EfficientNet observed that balancing all dimensions of the network—width, depth, and image resolution—against the available resources would best improve overall performance.

The resulting architecture uses mobile inverted bottleneck convolution (MBConv), with a slightly increased FLOP budget. It is then scale up the baseline network to obtain a family of models, called *EfficientNets*.



Comparison of different scaling methods. Unlike conventional scaling methods (b)-(d) that arbitrary scale a single dimension of the network, our compound scaling method uniformly scales up all dimensions in a principled way.

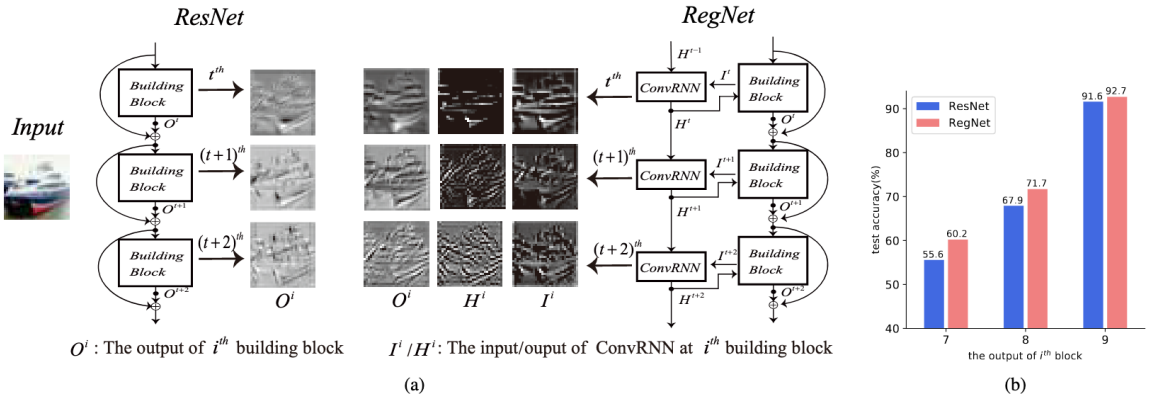
Reference: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

RegNetX

Traditionally, convolutional neural network architectures have been designed and optimized for one specific purpose. For example, the ResNet model family was optimized for the highest accuracy on ImageNet at the time of its initial release. MobileNets, as the name suggests, are optimized to run on mobile devices. Lastly, EfficientNet was designed to be highly efficient for visual recognition tasks.

The ResNet and its variants have achieved remarkable successes in various computer vision tasks. Despite its success in making gradient flow through building blocks, the simple shortcut connection mechanism limits the ability of re-exploring new potentially complementary features due to the additive function. To address this issue, the designers of RegNet propose to introduce a regulator module as a memory mechanism to extract complementary features, which are further fed to the ResNet. In particular, the regulator module is composed of convolutional RNNs (e.g., Convolutional LSTMs or Convolutional GRUs), which are shown to be good at extracting spatio-temporal information. Its named as new regulated networks as RegNet. The regulator module can be easily implemented and appended to any ResNet architectures.

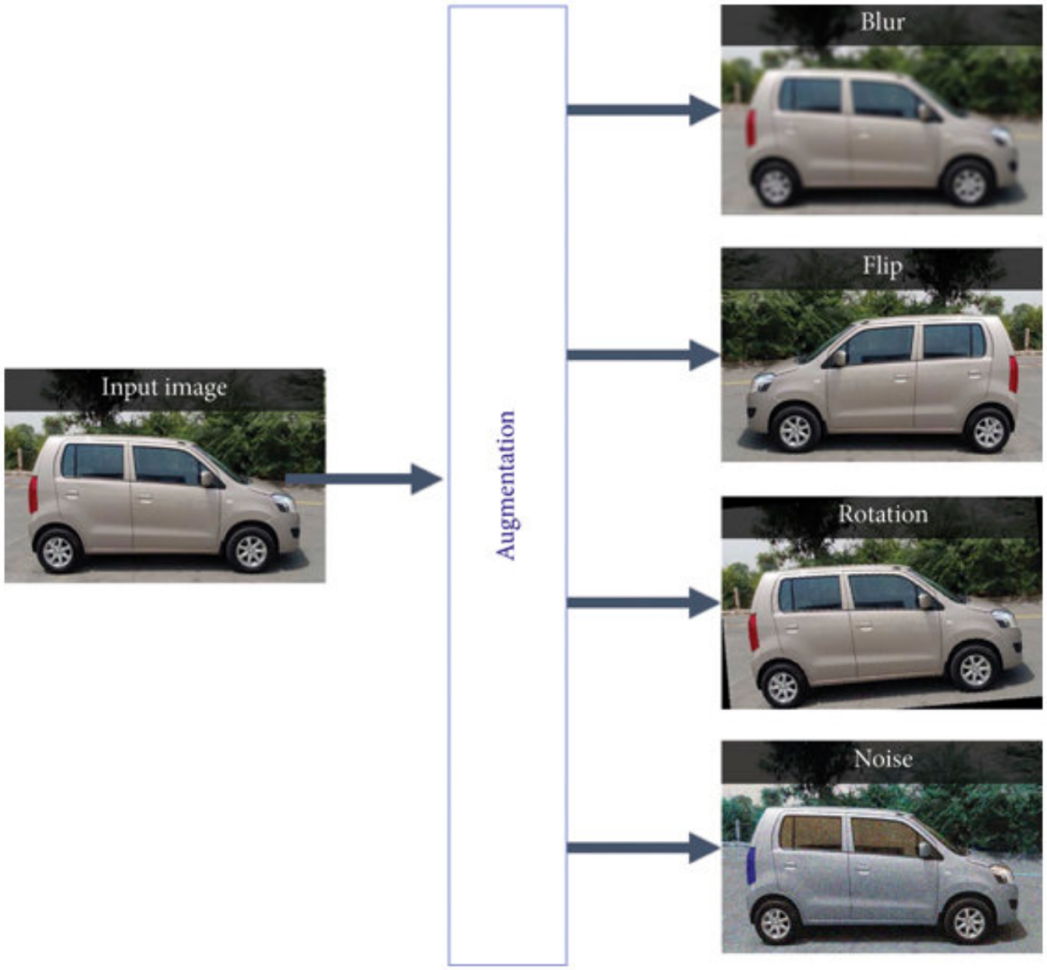
A potential solution to address the above problems is to capture the spatio-temporal dependency between building blocks while constraining the speed of parameter increasing. To this end, the designers introduce a new regulator mechanism in parallel to the shortcuts in ResNets for controlling the necessary memory information passing to the next building block. In detail, we adopt the Convolutional RNNs (“ConvRNNs") [12] as the regulator to encode the spatio-temporal memory. We name the new architecture as RNN-Regulated Residual Networks, or “RegNet" for short.



Fine Tuning

For fine tuning, we started to tune the CNN model definition with various permutations such as Image Augmentation, changing the learning rate, epochs, batch size, adding additional convoluted layers and dense layers. We also fine-tuned different pretrained models and the top three results were from ResNet101, EfficientNet and RegNet.

We performed various Data Augmentation technique to improve the model. As more parameters are added to a CNN, it requires more examples to show to the machine learning model. Deeper networks can have higher performance.



Results from Fine-Tuning:

EfficientNet and CNN:

Epoch	Batch Size	Augmentation	Optimazation	Learning Rate	CNN Layers	Pretrained Model	Accuracy
10	32	RandomRotation(degrees=45	Adam	0.0001		efficient_netb3	0.2
		Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])]					
10	32	RandomRotation(degrees=45	Adam	**0.001**		efficient_netb3	0.39
		Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])]					
10	32	RandomRotation(degrees=45	Adam	0.0001	(with 2 dense layers)	FALSE	0.05
		Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])]					
		RandomFlip					
10	32	RandomRotation(degrees=45	Adam	**0.001**	(with 3 dense layers)	FALSE	0.05
		Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])]					
		RandomAffine(30, 70)					

ResNet and RegNet:

	1	2	3	4	5
Optimizer	Adam	Adam	Adam	Adam	Adam
Model	**Resnet18**	**Resnet50**	**Resnet50 (with different weights)**	**Regnet_x_800mf**	**Regnet_y_800mf**
Epoch	25	25	25	10	10
Learning Rat	0.0001	0.0001	0.0001	0.00001	0.00001
Batch Size	16	16	16	32	32
Image Size	224	224	224	400	400
Test Acc	0.43096	0.41736	0.45502	0.36402	0.3368
Test hlm	0.07263	0.07204	0.06917	0.08028	0.08362
Test sum	1.29249	1.27739	1.32941	0.4443	0.42044

Overall Performance:

EPOCH = 40, Learning Rate = 0.0001, Optimizer = Adam, Batch Size = 32, Image Size = 224					
Change No.	Change Details	Test_Accuracy	hlm	Sum_Metric	Keep Changes
1.1	ResNet101	0.48326	0.06629	0.54956	Yes
1.2	AlexNet	0.06067	0.14697	0.20764	No
1.3	VGG16	0.07531	0.14769	0.223	No
1.4	EfficientNet_B3	0.4662	0.06681	0.54171	Yes
1.5	MobileNet_V3	0.38	0.07538	0.45822	No
1.6	MobileNet_V2	0.42	0.085	0.505	No
1.7	ResNet152	0.44	0.07048	0.51085	No
1.8	NFNet	0.33	0.09336	0.42286	No
1.9	RegNet_x_800mf	0.45534	0.06623	0.53694	No
2	RegNet_y_800mf (epochs = 40)	0.43	0.06956	0.50366	No
2.1	RegNet_x_400mf (epochs = 40)	0.4341	0.06832	0.50242	No
2.2	MLMIxer	0.25	0.11199	0.36617	No
2.3	ViT	0.06	0.1607	0.21613	No

Metrics

Accuracy Score:

Accuracy is the proportion of examples that were correctly classified. More precisely, it is sum of the number of true positives and true negatives, divided by the number of examples in the dataset. In multi-label classification, a misclassification is no longer a hard wrong or right. A prediction containing a subset of the actual classes should be considered better than a prediction that contains none of them.

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN}$$

Hamming Loss:

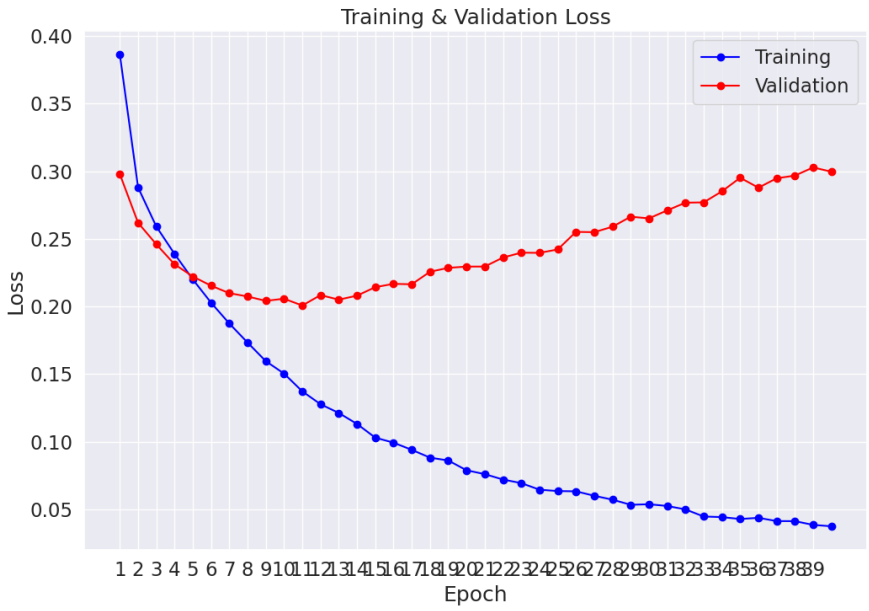
Hamming-Loss is the fraction of labels that are incorrectly predicted, i.e., the fraction of the wrong labels to the total number of labels. Lower the Hamming Loss, better the performance of the model.

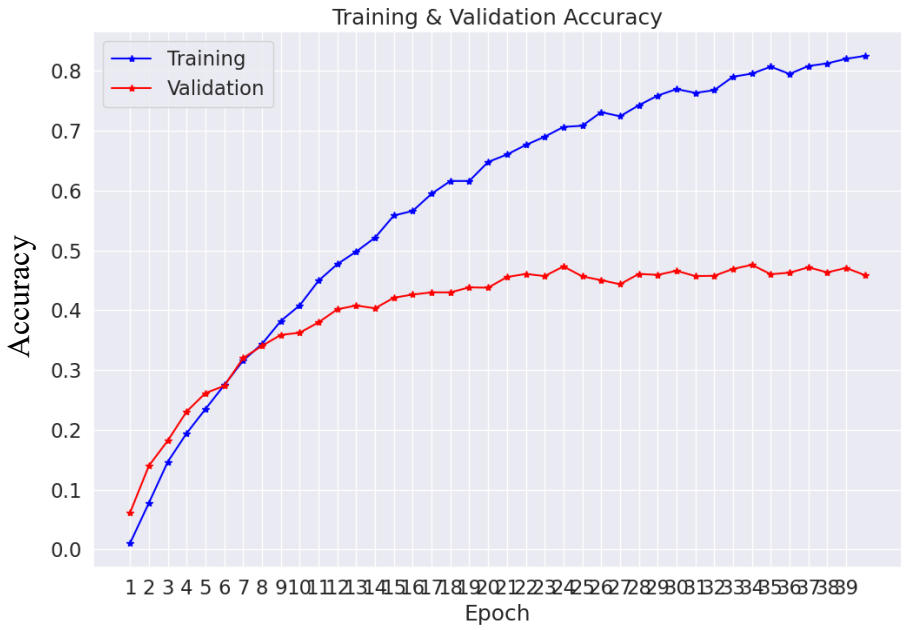
$$\frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} \text{xor}(y_{i,j}, z_{i,j}), \text{ where } y_{i,j} \text{ is the target and } z_{i,j} \text{ is the prediction.}$$

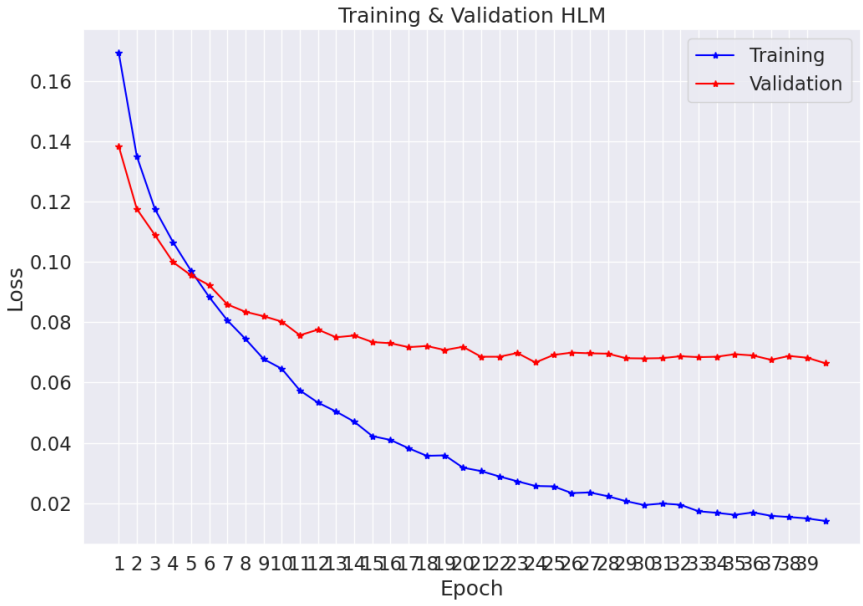
Metric Results

EfficientNet

Test Accuracy: 46% and Test Hamming: 0.06681
Average Individual Class – F1: 0.68




Training & Validation Accuracy
Training
Validation
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
Epoch

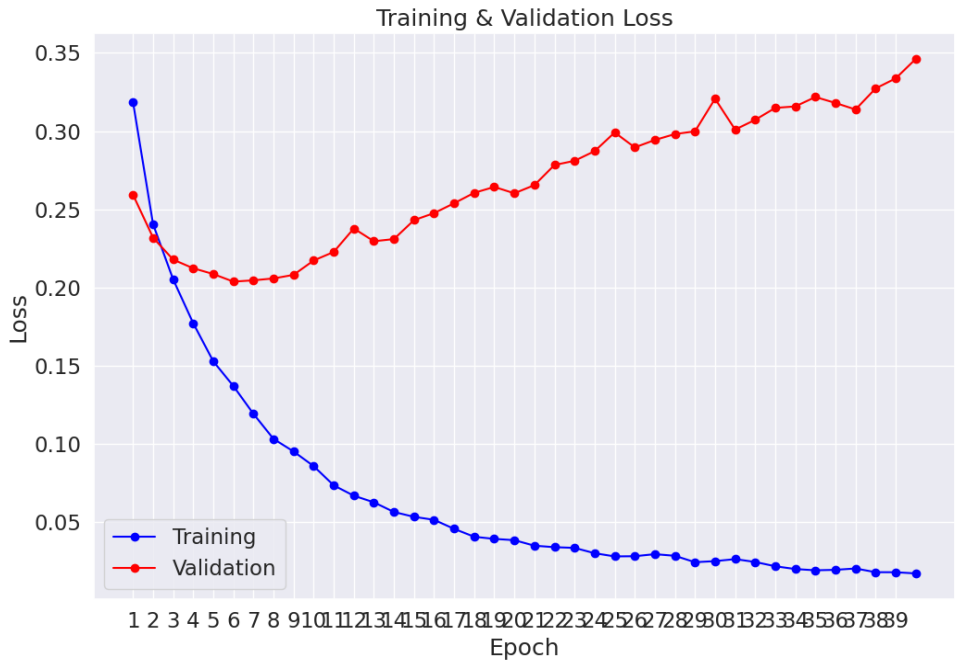

Training & Validation HLM
Training
Validation
Loss
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
Epoch

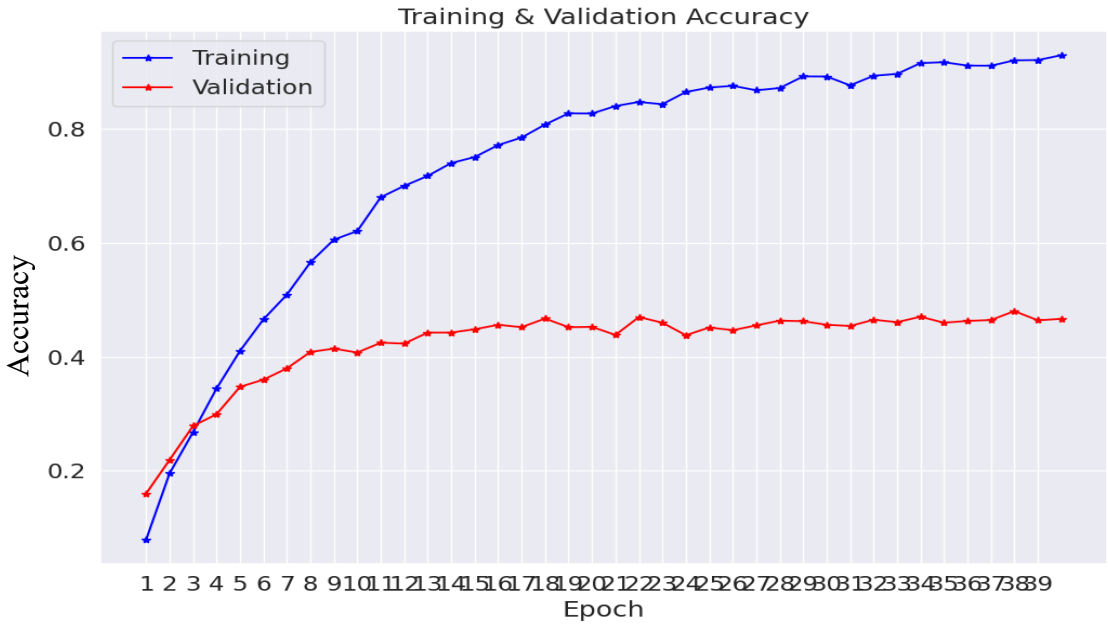
Classification Report for IDD Data :

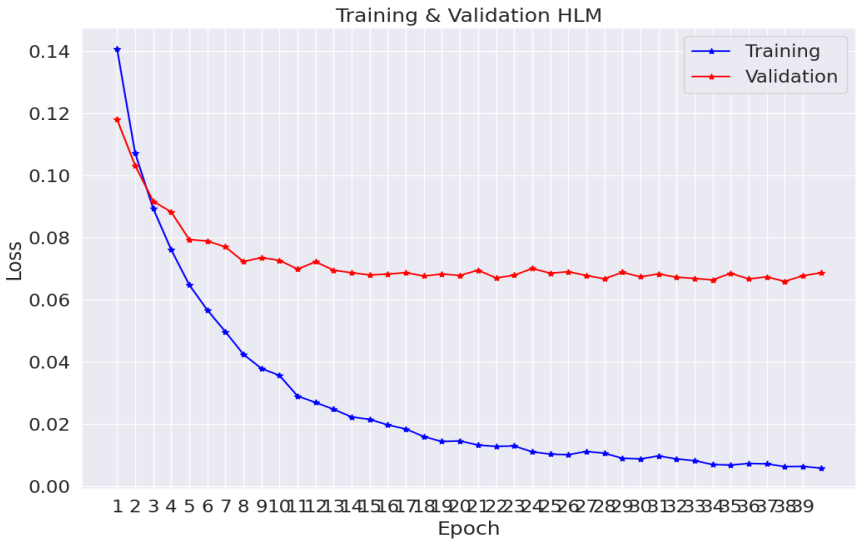
	precision	recall	f1-score	support
unknown	0.00	0.00	0.00	1
animal	0.69	0.53	0.60	64
autorickshaw	0.70	0.66	0.68	141
bicycle	0.81	0.59	0.69	59
bus	0.84	0.80	0.82	140
car	0.87	0.84	0.85	415
caravan	0.50	1.00	0.67	1
motorcycle	0.88	0.88	0.88	560
person	0.79	0.73	0.76	345
rider	0.88	0.88	0.88	513
traffic light	0.50	0.33	0.40	3
traffic sign	0.79	0.68	0.73	155
trailer	0.00	0.00	0.00	0
train	0.00	0.00	0.00	0
truck	0.88	0.81	0.85	302
vehicle fallback	0.57	0.53	0.55	142
micro avg	0.83	0.79	0.81	2841
macro avg	0.61	0.58	0.58	2841
weighted avg	0.83	0.79	0.81	2841
samples avg	0.83	0.79	0.79	2841

ResNet101

Test Accuracy: 48% and Test Hamming: 0.06629
Average Individual Class – F1: 0.72




Training & Validation Accuracy
Training
Validation
Accuracy
0.2
0.4
0.6
0.8
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
Epoch


Training & Validation HLM
Training
Validation
Loss
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
Epoch

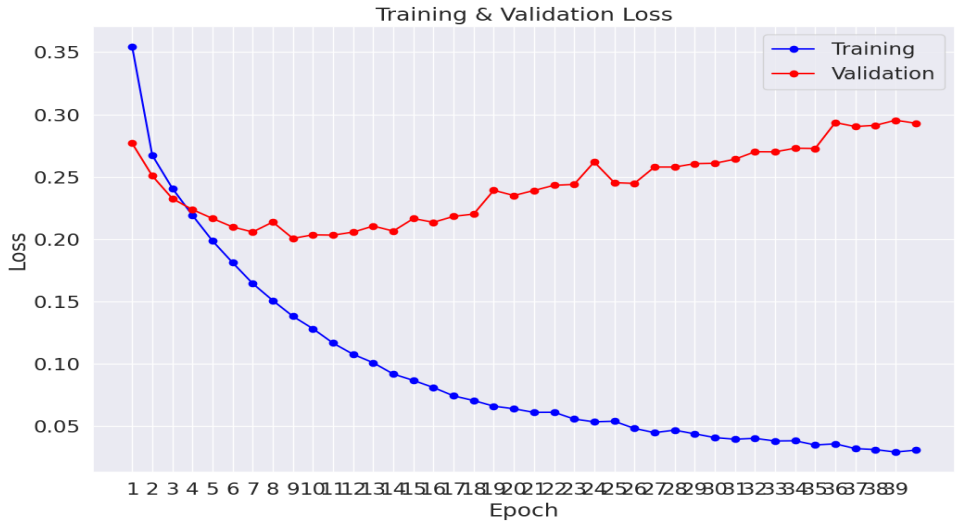
Classification Report for IDD Data :

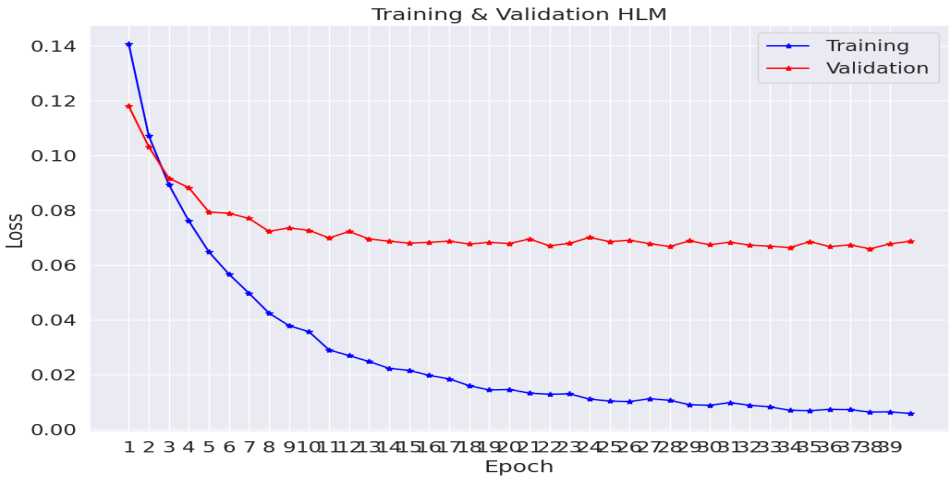
	precision	recall	f1-score	support
unknown	0.00	0.00	0.00	1
animal	0.73	0.62	0.67	64
autorickshaw	0.78	0.59	0.67	141
bicycle	0.71	0.54	0.62	59
bus	0.87	0.79	0.83	140
car	0.88	0.83	0.85	415
caravan	1.00	1.00	1.00	1
motorcycle	0.90	0.86	0.88	560
person	0.77	0.78	0.78	345
rider	0.89	0.88	0.88	513
traffic light	0.00	0.00	0.00	3
traffic sign	0.78	0.70	0.73	155
trailer	0.00	0.00	0.00	0
train	0.00	0.00	0.00	0
truck	0.92	0.75	0.83	302
vehicle fallback	0.68	0.47	0.56	142
micro avg	0.85	0.78	0.81	2841
macro avg	0.62	0.55	0.58	2841
weighted avg	0.85	0.78	0.81	2841
samples avg	0.84	0.79	0.80	2841

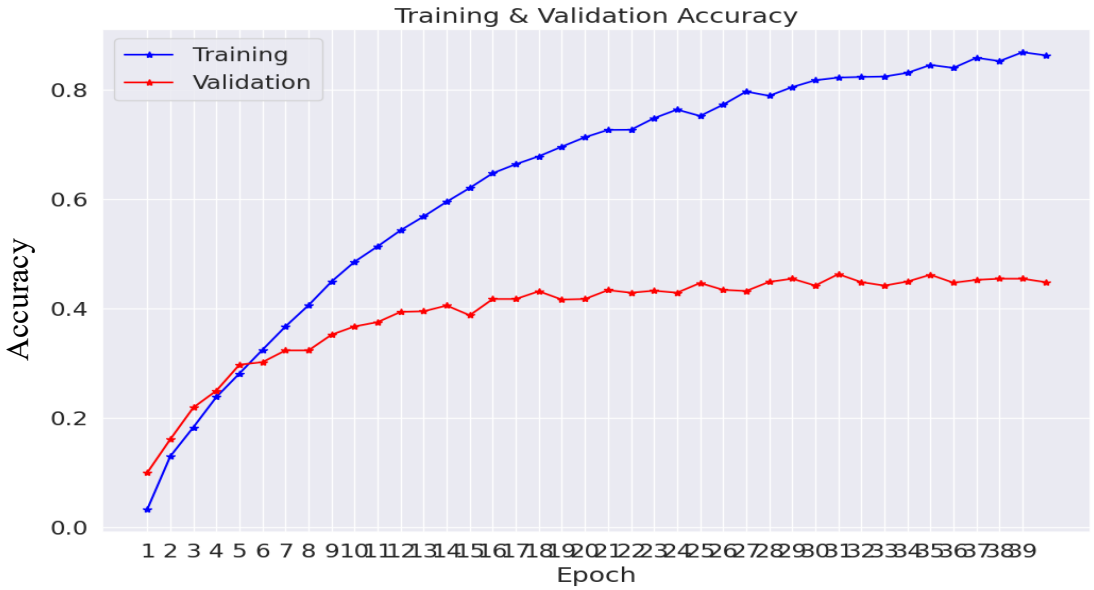
!!PREDICTION COMPLETE!!

RegNet_x_800

Test Accuracy: 45% and Test Hamming: 0.06623
Average Individual Class – F1: 0.67




Training & Validation HLM
Training
Validation
Loss
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
Epoch


Training & Validation Accuracy
Training
Validation
Accuracy
0.8
0.6
0.4
0.2
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
Epoch

Classification Report for IDD Data :

	precision	recall	f1-score	support
unknown	0.00	0.00	0.00	1
animal	0.64	0.50	0.56	64
autorickshaw	0.70	0.67	0.68	141
bicycle	0.84	0.63	0.72	59
bus	0.87	0.79	0.83	140
car	0.88	0.75	0.81	415
caravan	0.00	0.00	0.00	1
motorcycle	0.89	0.85	0.87	560
person	0.78	0.72	0.75	345
rider	0.87	0.86	0.86	513
traffic light	0.50	0.33	0.40	3
traffic sign	0.76	0.72	0.74	155
trailer	0.00	0.00	0.00	0
train	0.00	0.00	0.00	0
truck	0.88	0.80	0.84	302
vehicle fallback	0.68	0.54	0.60	142
micro avg	0.84	0.77	0.80	2841
macro avg	0.58	0.51	0.54	2841
weighted avg	0.84	0.77	0.80	2841
samples avg	0.83	0.78	0.78	2841

!!PREDICTION COMPLETE!!

Results:

Predicted Labels: Rickshaw, Bus, Car, Rider, Truck, Bicycle, Traffic Sign

Predicted Label: Bicycle, Car, Motorcycle, Rider, truck

Conclusion

The goal of this project was to accurately classify images with multiple labels. The dataset was approximately 10,000 images from Indian Driving Dataset which included images of Indian roads with cars, trucks, trailer with a total of 16 different labels.

We used PyTorch framework to implement the multi-label classification problem and used Convoluted Neural Networks defining our own model as well as fine tuning pretrained models such as ResNet101, EfficientNet and RegNet_x_800.

There were multiple steps taken to fine-tune our models such as data augmentation, change in learning rates, changes in batch size and optimization algorithm. After testing various parameters, our best model was the ResNet101.

For metrics, we used Accuracy Score and Hamming Loss, for ResNet101 mode, we got an Accuracy Score of Test Accuracy of 48% and Test Hamming at 0.06629 as the best result.

The overall project gave us an opportunity to work on a real-world problem which could be useful in the field of Autonomous Driving. The project gave us the opportunity to learn more about the Convoluted Neural Network and the different pretrained models that are built upon it.

Appendix

GitHub Link - https://github.com/IshanKuchroo/IDD-Indian-Driving-Dataset

Data Source

https://paperswithcode.com/dataset/idd

References

https://arxiv.org/pdf/1811.10200v1.pdf

https://machinelearningmastery.com/how-to-implement-major-architecture-innovations-for-convolutional-neural-networks/

https://github.com/rishikksh20/MLP-Mixer-pytorch/blob/master/mlp-mixer.py

https://towardsdatascience.com/self-driving-car-on-indian-roads-4e305cb04198

https://medium.com/mlearning-ai/vision-transformers-from-scratch-pytorch-a-step-by-step-guide-96c3313c2e0c

https://github.com/labmlai/annotated_deep_learning_paper_implementations

https://rwightman.github.io/pytorch-image-models/models/vision-transformer/

https://github.com/BrianPulfer/PapersReimplementations/blob/main/vit/vit_torch.py

https://medium.com/the-owl/imbalanced-multilabel-image-classification-using-keras-fbd8c60d7a4b

https://arxiv.org/pdf/2101.00590.pdf

https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html