Traffic Sign Recognition Using Deep Neural Networks

Darrel Belen*, Oleksandr Shashkov*

* Computer Science, *Illinois Institute of Technology*

***Abstract*-** It seems that with most modern technology, as it advances, daily life improves whether or not individuals realize it. One area where we can see the most improvement is in transportation. In particular, Advanced Driver Assistance Systems (ADAS) have vastly improved over the last few years. The goal of ADAS is to prevent deaths and injuries by trying to reduce the impact of car accidents which can be avoided. This research explores the Traffic Sign Recognition application of ADAS and how it detects and classifies various traffic signs.

***Index Terms*-** deep learning, ensemble classification, neural networks, traffic signs

I. INTRODUCTION

"Artificial Intelligence" has become a very popular buzzword in recent years. It has gradually become a part of daily life. From phones, to music, to home appliances, and more: AI is embedded itself in items individuals interact with every day. One area in particular that has seen vast improvements with regards to artificial intelligence is in transportation technology.

The Advanced Driver Assistance Systems, or ADAS, are standard systems in most modern cars. Since most vehicle accidents are caused by human error, the goal of ADAS is to prevent deaths and injuries by reducing the impact of car accidents which can be avoided [6]. There are several applications comprising each system of the ADAS. Some of these systems include pedestrian detection, automatic braking, and blind spot detection. The research conducted in this project focuses on the traffic sign recognition system of the ADAS.

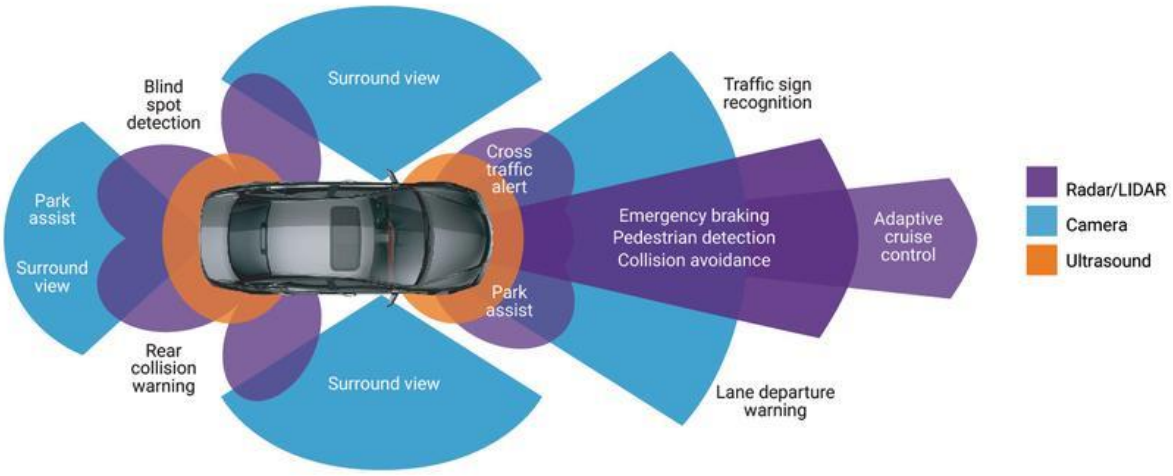


Figure 1: Advanced Driver Assistance Systems applications found on a modern vehicle

We will start by identifying the problem, as well as listing our contributions to the project and research as a whole. Next, we will identify the results of previous research on traffic sign recognition. This will not only give us an idea of what has been done, but also it will provide knowledge we can take and use when implementing our own research. Next, we will outline the data and list

important features. After this, we will describe implementation details of our network architectures and the ensemble of models. Following the discussion of implementation, we will analyze the results of our architecture and compare them with results of previous research. Finally, we will describe some possible limitations of our research and propose ideas for future research on traffic sign recognition.

1.1 Problem Statement

Most traffic sign recognition models that have been tested use European traffic sign datasets. We would have liked to expand upon research with US traffic sign datasets, but the data is currently too complex. We have instead created multiple different architectures using European traffic sign data. We will also combine these models into an ensemble for further analysis.

1.2 Contribution

The main contribution to this research is the ability to compare and contrast the performance of our models with other models that have been tested using traffic sign data. This serves to validate any previous work on traffic sign recognition and potentially bring traffic sign models up to date.

Group Member Contributions

Darrel Belen

- Literature Review
- Analysis and Implementation of Architectures
- Debugging

Oleksandr Shashkov

- Data Preparation
- Analysis and Implementation of Architectures
- Ensemble Implementation

II. LITERATURE REVIEW

An article that greatly motivated this research was a paper entitled, "Deep Learning Approach for U.S. Traffic Sign Recognition." In this paper, the authors explore the traffic sign recognition application of ADAS by trying to expand the various traffic signs it can detect. Currently, modern ADAS is limited to speed limit signs. By trying to cover all publicly available classes, they aim to broaden the capabilities of the traffic sign recognition system. Also, while most of the available public traffic sign data sets are European, this research focuses on US traffic signs. The US traffic sign data set (LISA) has not had nearly as extensive testing as the European data sets [2]. The authors of the paper obtain a notably high accuracy percentage on the LISA data set. They also observe that, had the model been trained a little longer, the performance could still be improved.

Unfortunately, while we would have liked to expand upon previous work on the LISA dataset, it became apparent that this dataset was very complex. The data set contains still frames of traffic signs extracted from a video. Bringing the US traffic sign data to usable form would require much reworking, so we decided to deviate from this initial plan. Instead of refactoring the LISA dataset, we decided to use a European dataset that was much simpler. In doing so, we could build a network more easily and compare our results to any previous work. As a result of switching datasets, we were able to create four different networks, capable of classification of German traffic signs, as well as combine the models into an ensemble.

III. DATA OVERVIEW

For the dataset, we selected a subset of small pictures from German Traffic Sign Detection Benchmark (see Appendix).

Figure 2: Traffic sign subset from Kaggle (https://benchmark.ini.rub.de/gtsrb_dataset.html)

The dataset is a collection of single-image pictures of German traffic signs representing 43 classes therefore defining the problem as multi-class classification. The collection consists of 861 images in total. The training, validation, and testing set structures described as follows. One directory per class and each directory contains sample images.

Single-Image Format:

- The images contain one traffic sign each
- Images contain a border of 10 % around the actual traffic sign (at least 5 pixels) to allow for edge-based approaches
- Images are stored in PPM format (Portable Pixmap, P6)
- Image sizes vary between 15x15 to 250x250 pixels
- Images are not necessarily squared
- The actual traffic sign is not necessarily centered within the image. This is true for images that were close to the image border in the full camera image

Original limited dataset was split in 3 parts - 70% for training, 15% for validation and 15% for testing. Due to the limited size of the training data, we used samples generator with data augmentation producing additional training samples generated out of original images (see picture below)

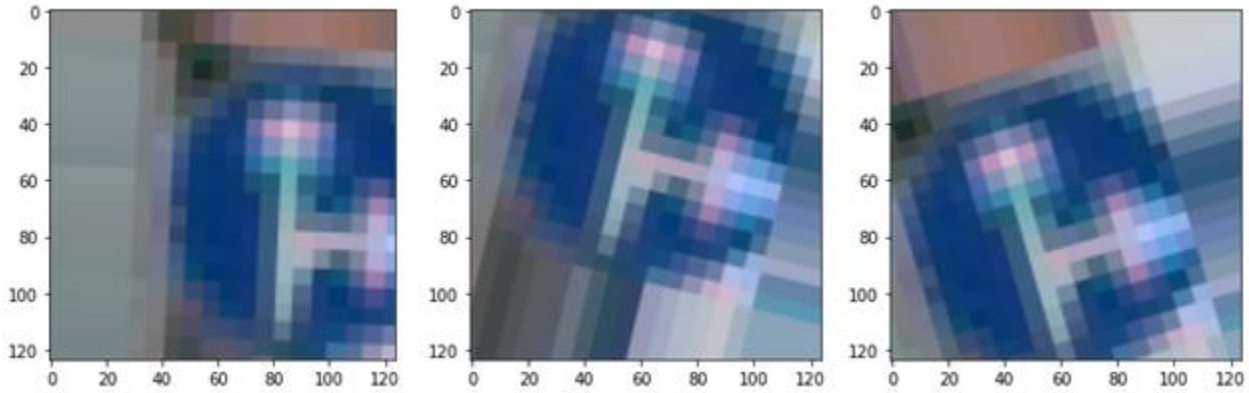


Figure 3: Generator with Data Augmentation

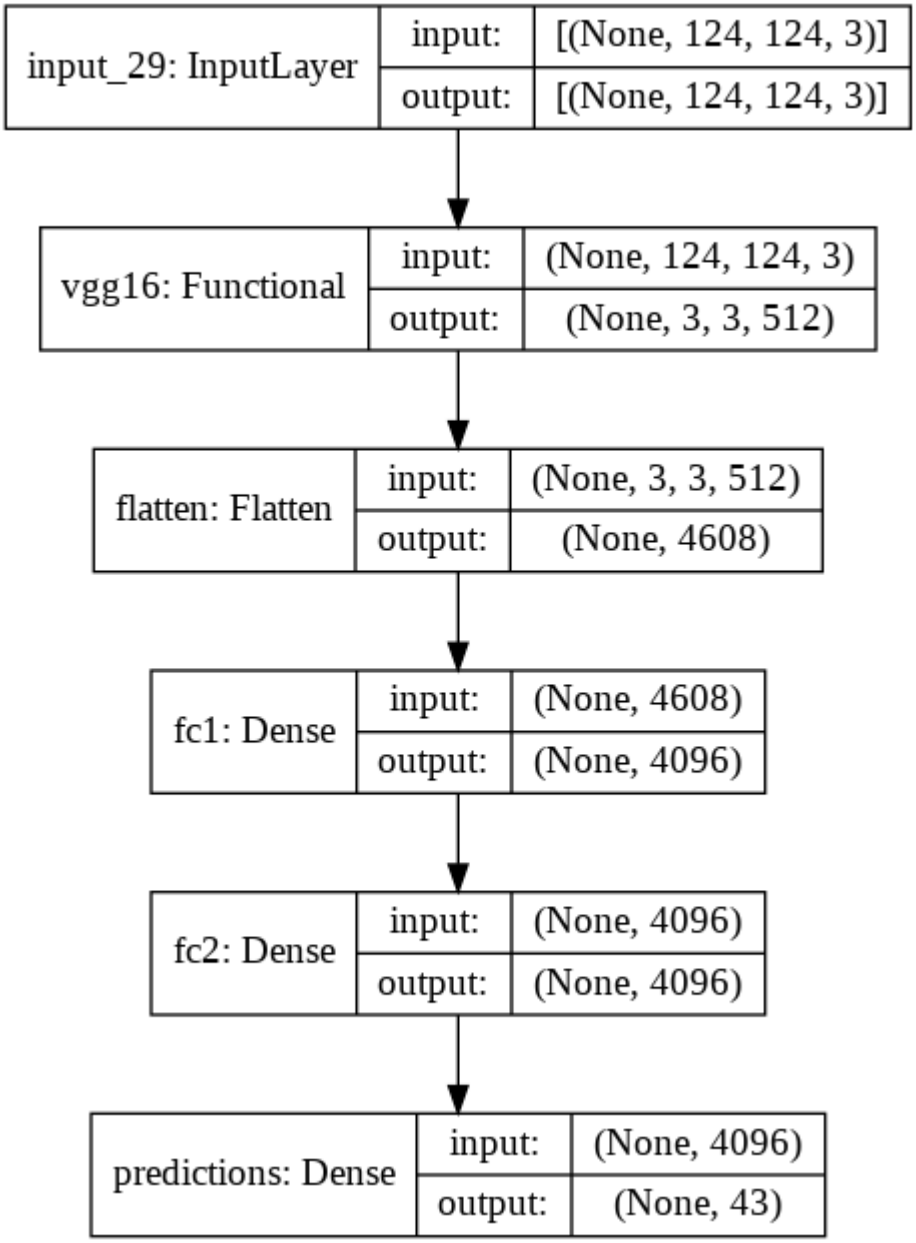
IV. IMPLEMENTATION DETAILS

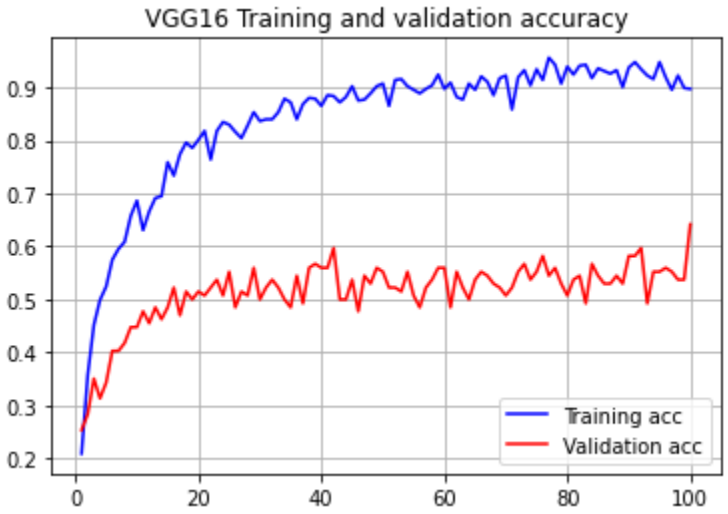
Four convolutional network architectures available in Keras were evaluated during this study: 1) VGG16 [4]; 2) MobileNetV2 [5]; 3) Xception [3]; and 4) InceptionResNetV2 [1].

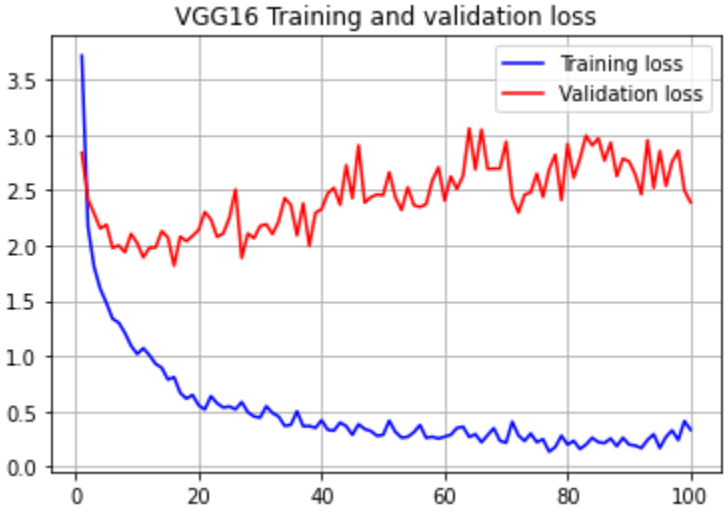
VGG16

VGG16 is the Deep Convolutional network architecture which was made publicly known during the ImageNet Challenge in 2014. The network capitalizes on increased depth (16 weight layers) and utilization of very small (3x3) convolution filters. The model is known for its high accuracy achieved on Imagenet classification dataset. However, this architecture did not learn well on our limited training dataset and we had to opt to freeze the convolutional base and use pre-trained Imagenet weights. Below are the results

demonstrating model use in the setup for our classification task and graphs showing training and validation metrics along with the final accuracy.





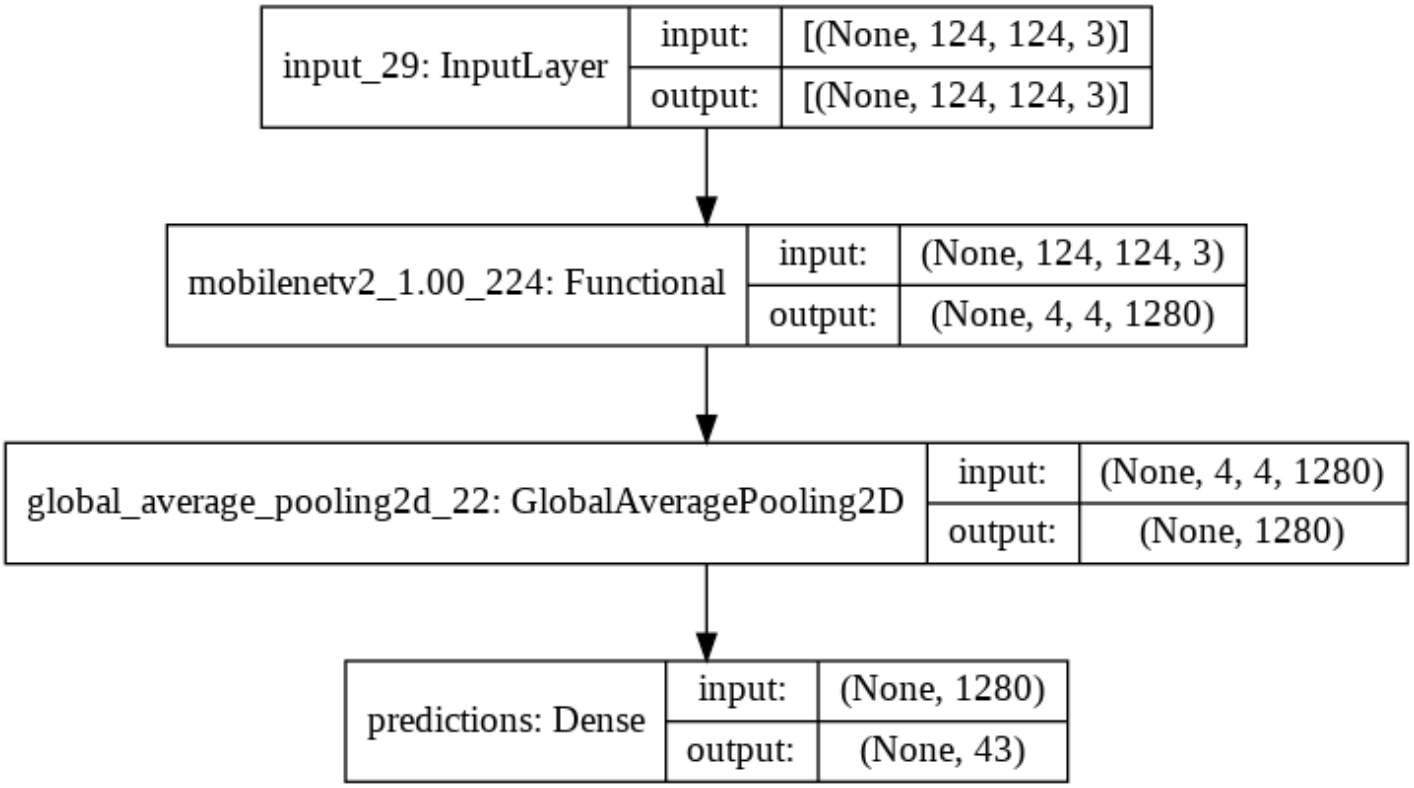


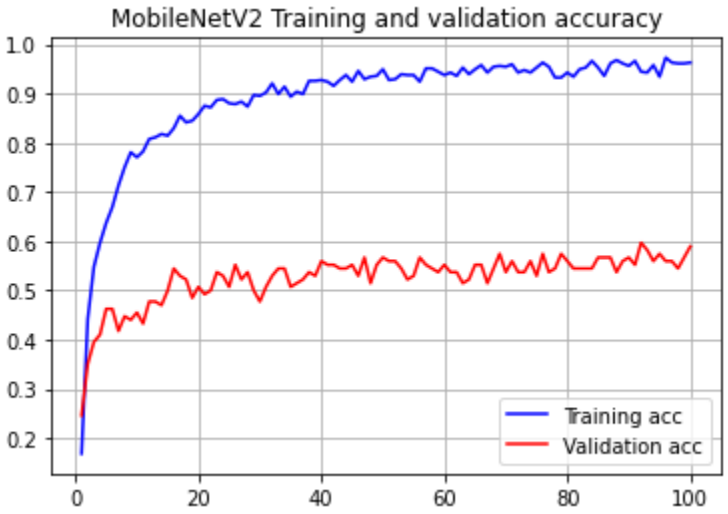
Test loss: 2.94 , Test accuracy 57.46 %

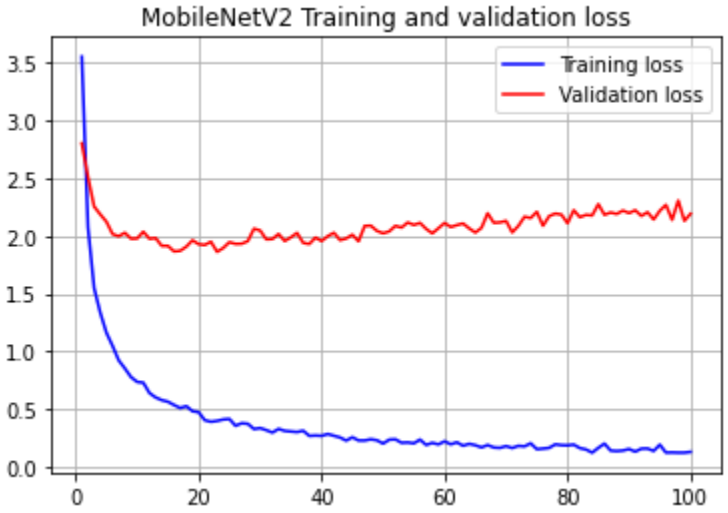
MobileNetV2

MobileNetV2 is the version of MobileNets architecture designed for mobile and embedded vision applications. MobileNets are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks. The architectures introduced two simple global hyper-parameters that efficiently trade off latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem.

While MobileNet was much lighter and it trained faster than any other models we explored, it did not train well on the limited dataset we had available. Similar to VGG16, this model did not demonstrate acceptable performance after training on our dataset. Therefore we opted to freeze convolutional base and use pre-trained “Imagenet” weights. Below are the structure of the model and the training and validation data followed by final accuracy.


input_29: InputLayer
input:
[(None, 124, 124, 3)]
output:
[(None, 124, 124, 3)]
mobilenetv2_1.00_224: Functional
input:
(None, 124, 124, 3)
output:
(None, 4, 4, 1280)
global_average_pooling2d_22: GlobalAveragePooling2D
input:
(None, 4, 4, 1280)
output:
(None, 1280)
predictions: Dense
input:
(None, 1280)
output:
(None, 43)

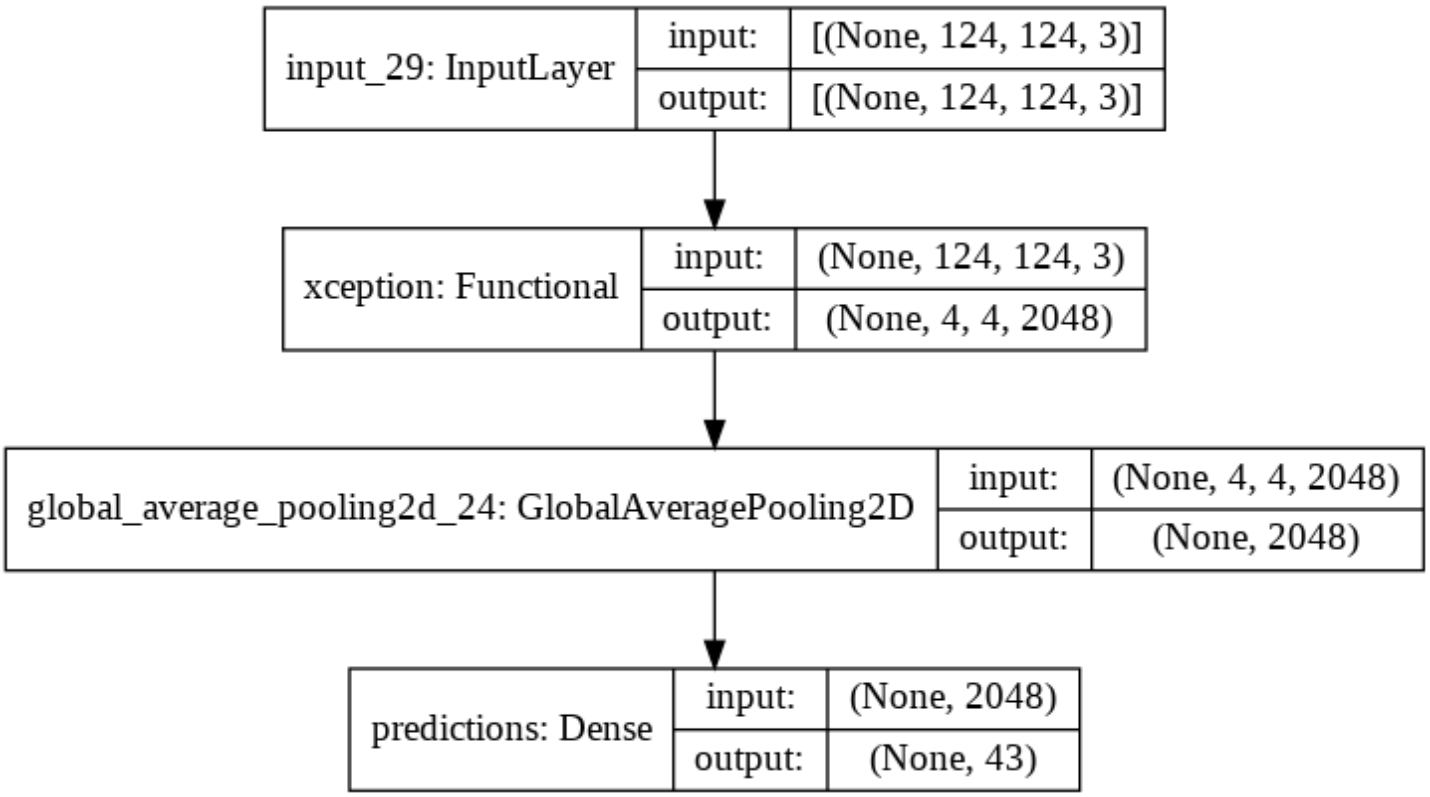


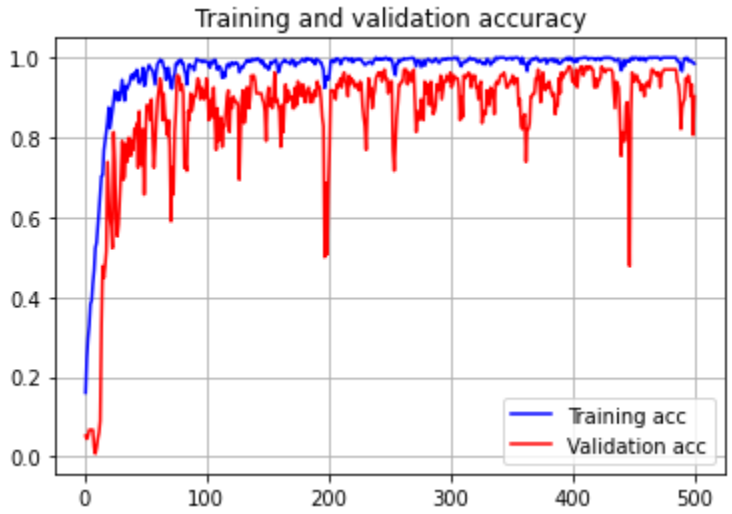


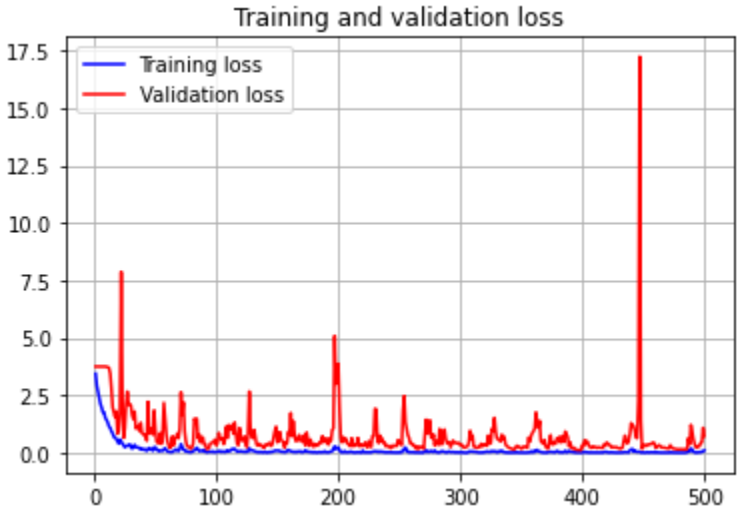

```
Test loss:      2.12 , Test accuracy     61.19 %
```

Xception

Xception is a variation of architectures utilizing Inception modules in convolutional neural networks. These modules are based on the idea of stacking a depthwise convolution followed by a pointwise convolution. The authors of the architecture suggest an interpretation as an Inception module with a maximally large number of towers. Therefore they proposed a novel deep convolutional neural network architecture where Inception modules have been replaced with depthwise separable convolutions. This architecture demonstrated performance gain over Inception architectures which was not due to increased capacity but due to more efficient use of the model. The model learned exceptional well from our dataset and therefore was trained as is without any hints of transfer learning. Below are the details demonstrating implementation and performance.





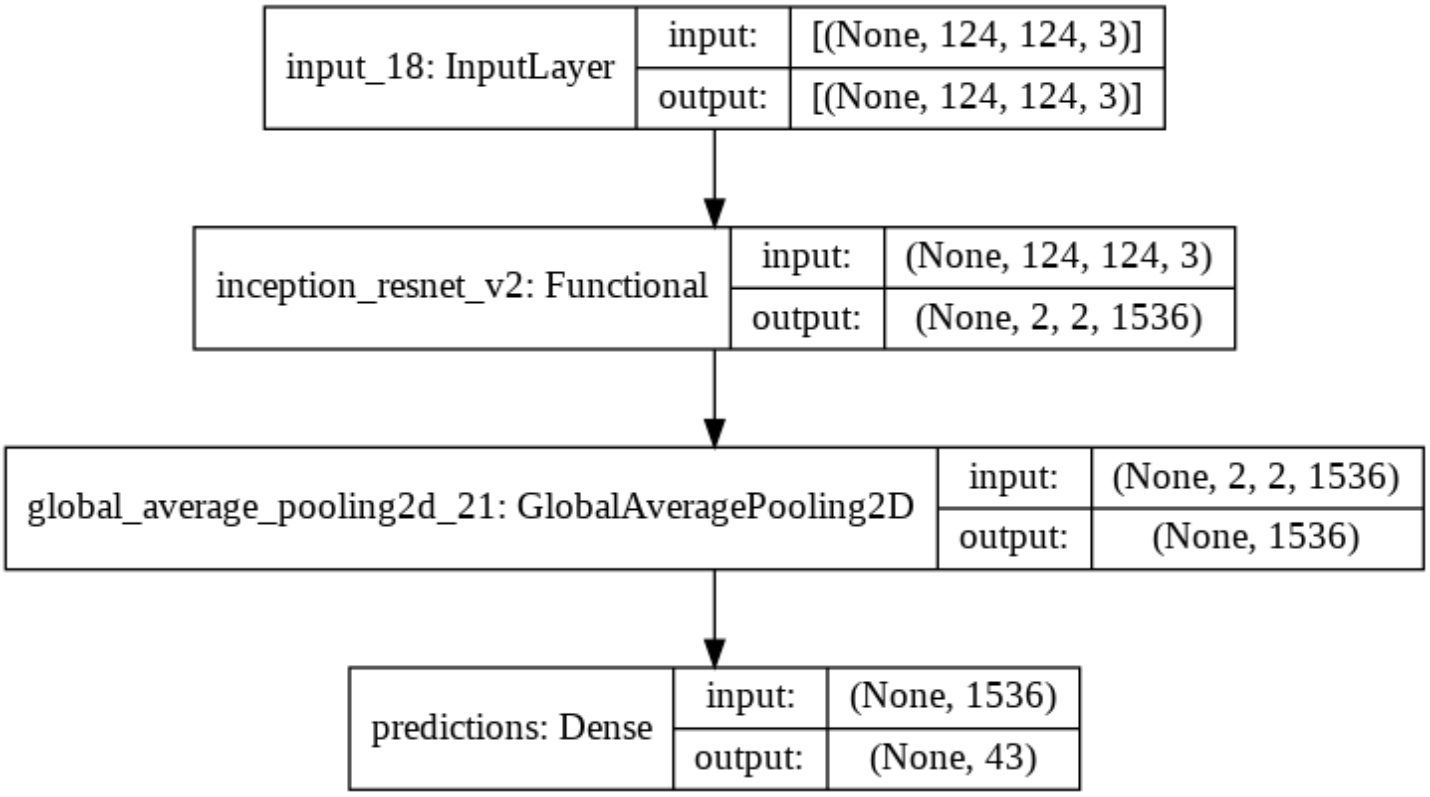


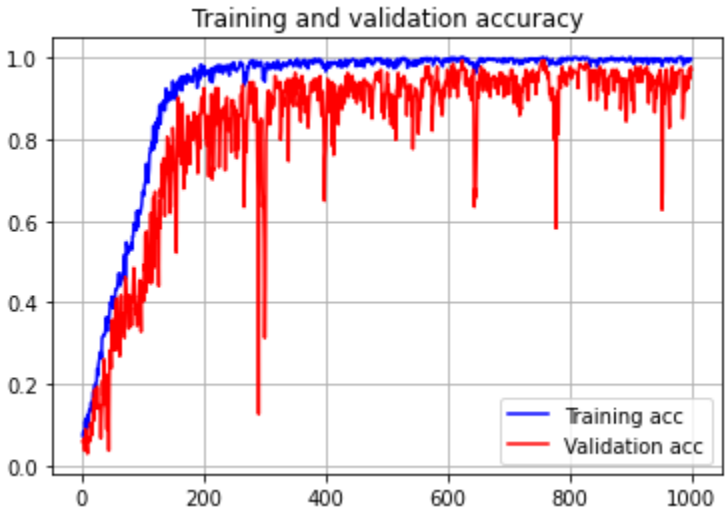

```
Test loss:     0.20 , Test accuracy    94.78 %
```

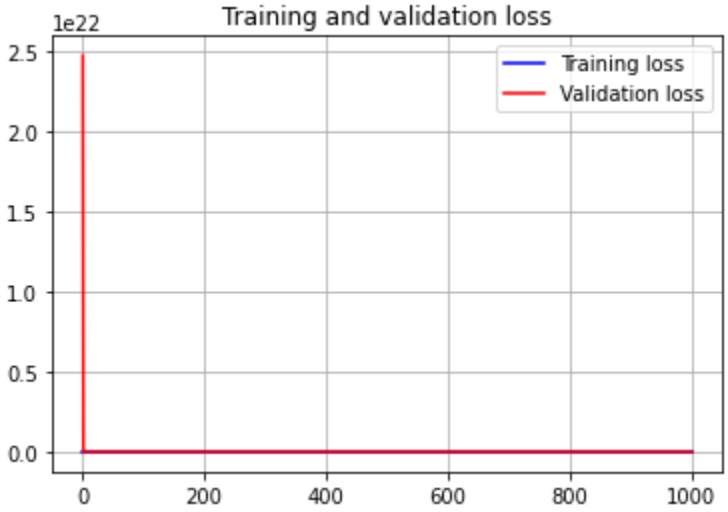
InceptionResNetV2

InceptionResNetV2 is the deep convolutional network combining two novel architectural solutions in one - Inception blocks and Residual blocks. This architecture allowed for better accuracy and faster training when comparing to the models with just either Inception blocks or Residual blocks.

This model demonstrated excellent learning capabilities and provided for exceptional performance. We did train the network fully without freezing any layers or using pre-trained weight. The implementation and the results are shown below.


input_18: InputLayer
input:
[(None, 124, 124, 3)]
output:
[(None, 124, 124, 3)]
inception_resnet_v2: Functional
input:
(None, 124, 124, 3)
output:
(None, 2, 2, 1536)
global_average_pooling2d_21: GlobalAveragePooling2D
input:
(None, 2, 2, 1536)
output:
(None, 1536)
predictions: Dense
input:
(None, 1536)
output:
(None, 43)





Test loss: 0.53 , Test accuracy 93.28 %

Ensembles

Each stand along architectures described above were instantiated, trained, and evaluated using Keras. An ensemble of all 4 architectures was then created to evaluate combined performance. We used two similar ensembles - One was averaging prediction outputs from the networks over each class and the other was taking maximum prediction outputs from each class. Below are the summaries of each ensemble models.

```
Model: "ensemble_average"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_29 (InputLayer)           [(None, 124, 124, 3) 0
__________________________________________________________________________________________________
VGG16 (Functional)              (None, 43)           50550635    input_29[0][0]
__________________________________________________________________________________________________
MobileNetV2 (Functional)        (None, 43)           2313067     input_29[0][0]
__________________________________________________________________________________________________
Xception (Functional)           (None, 43)           20949587    input_29[0][0]
__________________________________________________________________________________________________
InceptionResNetV2 (Functional)  (None, 43)           54402827    input_29[0][0]
__________________________________________________________________________________________________
average_20 (Average)            (None, 43)           0           VGG16[0][0]
                                                                 MobileNetV2[0][0]
                                                                 Xception[0][0]
                                                                 InceptionResNetV2[0][0]
```

```
==================================================================================================
Total params: 128,216,116
Trainable params: 111,128,372
Non-trainable params: 17,087,744
__________________________________________________________________________________________________
Model: "ensemble_maximum"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_29 (InputLayer)           [(None, 124, 124, 3) 0
__________________________________________________________________________________________________
VGG16 (Functional)              (None, 43)           50550635    input_29[0][0]
__________________________________________________________________________________________________
MobileNetV2 (Functional)        (None, 43)           2313067     input_29[0][0]
__________________________________________________________________________________________________
Xception (Functional)           (None, 43)           20949587    input_29[0][0]
__________________________________________________________________________________________________
InceptionResNetV2 (Functional)  (None, 43)           54402827    input_29[0][0]
__________________________________________________________________________________________________
maximum_13 (Maximum)            (None, 43)           0           VGG16[1][0]
                                                                 MobileNetV2[1][0]
                                                                 Xception[1][0]
                                                                 InceptionResNetV2[1][0]
==================================================================================================
Total params: 128,216,116
Trainable params: 111,128,372
Non-trainable params: 17,087,744
__________________________________________________________________________________________________
```

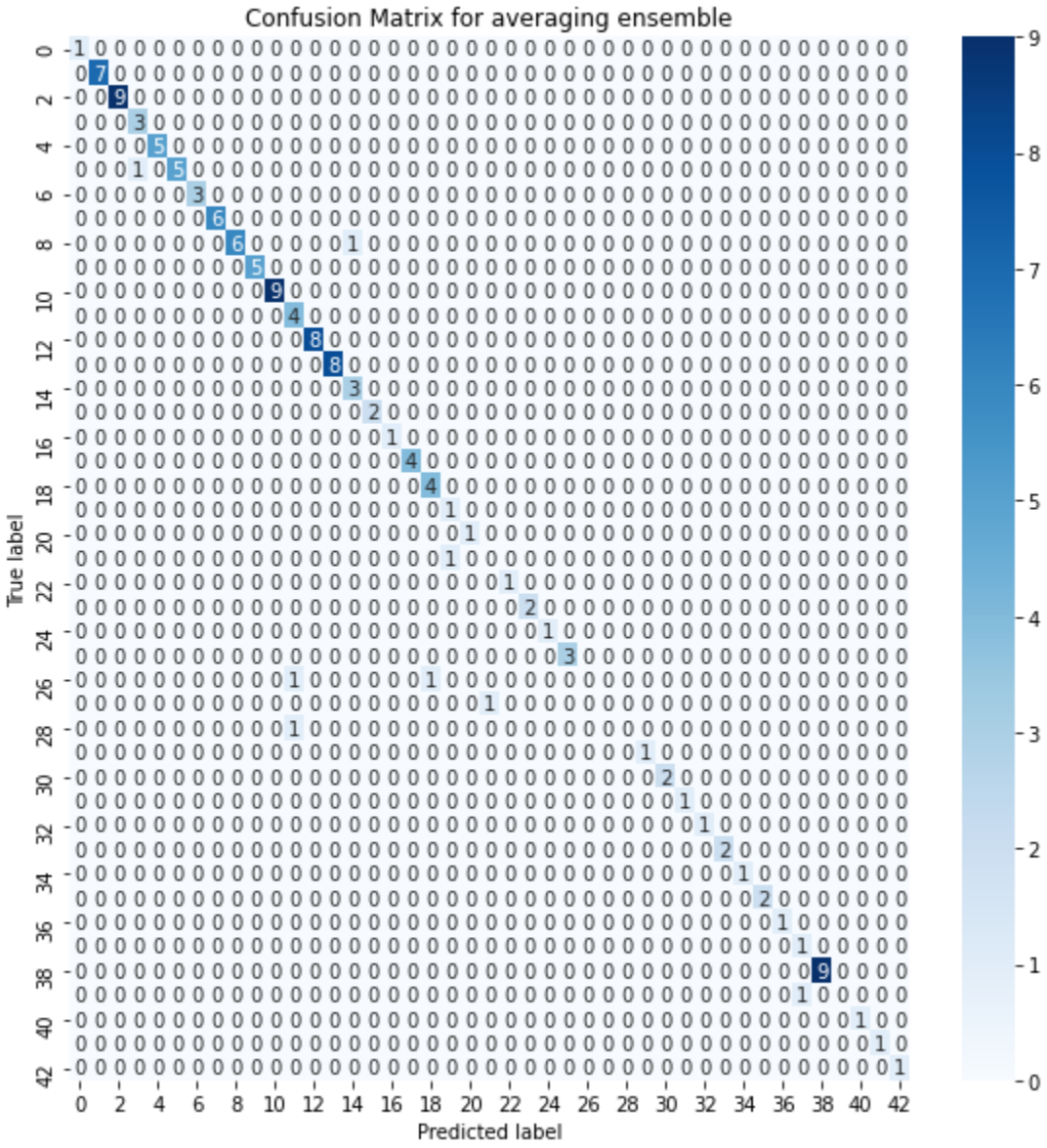
V. ANALYSIS

ACCURACY FOR EACH INDIVIDUAL MODEL:

```
Model: VGG16                , Test loss:     2.94 , Test accuracy    57.46 %
Model: MobileNetV2          , Test loss:     2.12 , Test accuracy    61.19 %
Model: Xception             , Test loss:     0.20 , Test accuracy    94.78 %
Model: InceptionResNetV2    , Test loss:     0.53 , Test accuracy    93.28 %
```

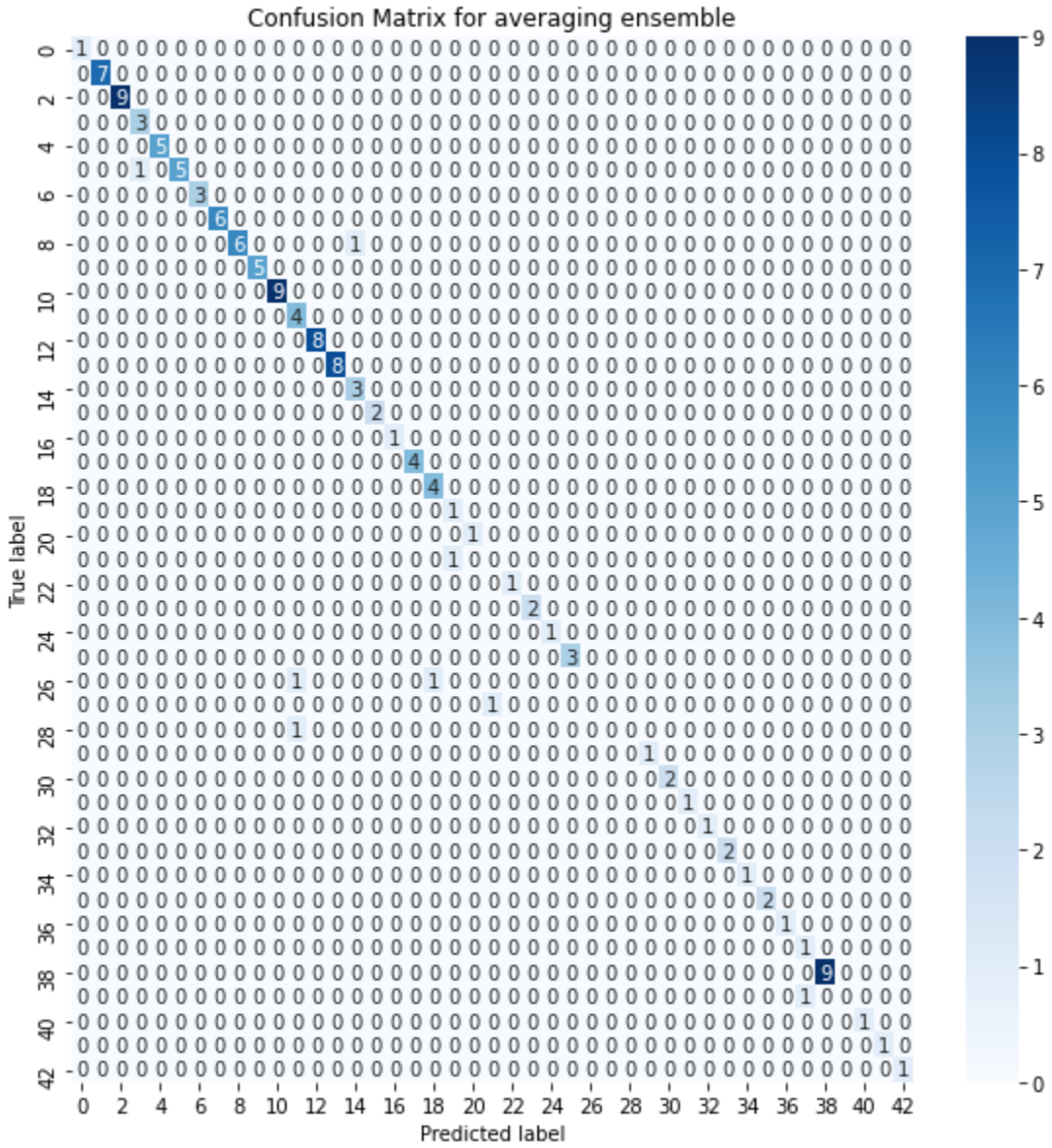
ACCURACY OF ENSEMBLES:

```
Averaging Ensemble: Test loss:     0.46 , Test accuracy    94.03 %
Maximizing Ensemble: Test loss:     0.64 , Test accuracy    92.54 %
```


Confusion Matrix for averaging ensemble
True label
Predicted label
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42
9 8 7 6 5 4 3 2 1 0

Classification Report for averaging ensemble

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
1	1.00	1.00	1.00	7
2	1.00	1.00	1.00	9
3	0.75	1.00	0.86	3
4	1.00	1.00	1.00	5
5	1.00	0.83	0.91	6
6	1.00	1.00	1.00	3
7	1.00	1.00	1.00	6
8	1.00	0.86	0.92	7
9	1.00	1.00	1.00	5
10	1.00	1.00	1.00	9
11	0.67	1.00	0.80	4
12	1.00	1.00	1.00	8
13	1.00	1.00	1.00	8
14	0.75	1.00	0.86	3
15	1.00	1.00	1.00	2
16	1.00	1.00	1.00	1
17	1.00	1.00	1.00	4
18	0.80	1.00	0.89	4
19	0.50	1.00	0.67	1
20	1.00	1.00	1.00	1
21	0.00	0.00	0.00	1
22	1.00	1.00	1.00	1
23	1.00	1.00	1.00	2
24	1.00	1.00	1.00	1
25	1.00	1.00	1.00	3
26	0.00	0.00	0.00	2
27	0.00	0.00	0.00	1
28	0.00	0.00	0.00	1
29	1.00	1.00	1.00	1
30	1.00	1.00	1.00	2
31	1.00	1.00	1.00	1
32	1.00	1.00	1.00	1
33	1.00	1.00	1.00	2
34	1.00	1.00	1.00	1
35	1.00	1.00	1.00	2
36	1.00	1.00	1.00	1
37	0.50	1.00	0.67	1
38	1.00	1.00	1.00	9
39	0.00	0.00	0.00	1
40	1.00	1.00	1.00	1
41	1.00	1.00	1.00	1
42	1.00	1.00	1.00	1
accuracy			0.94	134
macro avg	0.84	0.88	0.85	134
weighted avg	0.92	0.94	0.93	134



Classification Report for maximizing ensemble

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
1	1.00	0.86	0.92	7
2	1.00	1.00	1.00	9
3	0.75	1.00	0.86	3
4	1.00	1.00	1.00	5
5	1.00	0.67	0.80	6
6	1.00	1.00	1.00	3
7	1.00	1.00	1.00	6
8	0.86	0.86	0.86	7
9	1.00	1.00	1.00	5
10	1.00	0.89	0.94	9
11	0.67	1.00	0.80	4
12	0.89	1.00	0.94	8
13	1.00	1.00	1.00	8
14	0.75	1.00	0.86	3
15	1.00	1.00	1.00	2
16	1.00	1.00	1.00	1

17	0.80	1.00	0.89	4
18	0.80	1.00	0.89	4
19	1.00	1.00	1.00	1
20	1.00	1.00	1.00	1
21	1.00	1.00	1.00	1
22	1.00	1.00	1.00	1
23	0.67	1.00	0.80	2
24	1.00	1.00	1.00	1
25	1.00	1.00	1.00	3
26	0.00	0.00	0.00	2
27	0.00	0.00	0.00	1
28	0.00	0.00	0.00	1
29	1.00	1.00	1.00	1
30	1.00	1.00	1.00	2
31	1.00	1.00	1.00	1
32	1.00	1.00	1.00	1
33	1.00	1.00	1.00	2
34	1.00	1.00	1.00	1
35	1.00	1.00	1.00	2
36	1.00	1.00	1.00	1
37	1.00	1.00	1.00	1
38	0.90	1.00	0.95	9
39	0.00	0.00	0.00	1
40	1.00	1.00	1.00	1
41	1.00	1.00	1.00	1
42	1.00	1.00	1.00	1
accuracy			0.93	134
macro avg	0.86	0.89	0.87	134
weighted avg	0.90	0.93	0.91	134

Observations

- VGG16 and MobileNet did not learn on limited the dataset we had available
- Xception and InceptionResNet demonstrated outstanding learning capabilities and achieved 90%+ accuracy on the same data
- Ensembles demonstrated accuracy comparable with Xception and IncpetionResNet as standalone networks. The reason for that is the very high accuracy of those two models. Ensembles might make better sense for combining less accurate members with different strengths to achieve higher quality of classification.

VI. LIMITATIONS

During the research process we encountered and had to accept several limitations. The most noticeable limitation is that we were unable to find a publicly available dataset containing US traffic sign pictures in small size suitable for quick training and processing by a neural network. Instead, small sign images from GTSDB were used with the assumption that the models would demonstrate similar performance on the US traffic signs. Another limitation we encountered is that the models were trained and evaluated on single sign compact images. It would be beneficial to adopt and evaluate model architectures for real life street images with potentially multiple traffic signs in sight. Finally, the limited size of the available dataset used restricted our capabilities to provide enough training samples for some models (specifically VGG16 and MobileNetV2). We might have had better performance of VGG16 and MobileNetV2 models and would not have had to opt to use a pre-trained convolutional base should we have had a large training set.

VII. CONCLUSION

During the research we evaluated performance of known deep networks available in the Keras package for traffic sign recognition and classification. We also combined individual network architectures into ensembles and compared their performance with individual models. It seems that highly accurate network members dominate in the ensemble output. Therefore, with traffic sign recognition, it might be warranted to use an individual model with high accuracy score or opt to use the ensemble of less accurate, but less computationally costly members.

APPENDIX

The German Traffic Sign Detection Benchmark. https://benchmark.ini.rub.de/gtsdb_news.html
GTSDB - German Traffic Sign Detection Benchmark. https://www.kaggle.com/safabouguezzi/german-traffic-sign-detection-benchmark-gtsdb

REFERENCES

[1] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *Cornell University*, 2016, Available: arXiv.org, https://arxiv.org/abs/1602.07261. [Accessed April 28, 2021].

[2] Emmanuel B. Nuakoh, Kaushik Roy, Xiaohong Yuan, and Albert Esterline, "Deep Learning Approach for U.S. Traffic Sign Recognition," in *ICDLT 2019: Proceedings of the 2019 3rd International Conference on Deep Learning Technologies, July 2019*, pp. 47-50

[3] François Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *Cornell University*, 2016, Available: arXiv.org, https://arxiv.org/abs/1610.02357. [Accessed April 25, 2021].

[4] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Cornell University*, 2015, Available: arXiv.org, https://arxiv.org/abs/1409.1556. [Accessed April 25, 2021].

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Cornell University*, 2018, Available: arXiv.org, https://arxiv.org/abs/1801.04381. [Accessed April 25, 2021].

[6] Synopsys, "What is ADAS?," *Synopsys*, 2020, Available: synopsys.com, https://www.synopsys.com/automotive/what-is-adas.html. [Accessed March 30, 2021].

AUTHORS

First Author – Darrel Belen, Illinois Institute of Technology, dbelen@hawk.iit.edu
Second Author – Oleksandr Shashkov, Illinois Institute of Technology, oshashko@hawk.iit.edu