Small but Smart:
Optimizing AI for Efficient Driver Distraction Detection using Minimal Model Parameters

Legolas Zhang
lele733@hotmail.com

Abstract

Distracted driving is dangerous and difficult to monitor, therefore timely detection is critical to mitigating injury and fatalities. However, most research in this area relies on large, inefficient computer vision models. This paper introduces a novel, custom-built AI model specifically designed for real-time distracted driving detection, showcasing how targeted model design can dramatically reduce runtime and memory requirements without compromising accuracy. Starting from a basic model, design iterations made the model as compact as possible while maintaining high accuracy. The training dataset, composed of over 13,000 images, was also augmented to simulate real-world challenges such as rotations, translations and noise. The augmentations also make the dataset more complex, while preserving high efficiency. This model is a demonstration of how compact and efficient models can achieve the same accuracy as larger pre-trained models, like AlexNet, in specialized tasks, despite having 0.6% of the parameters and file size. This paper's model, while around the same size as ultra-light weight models such as SqueezeNet, drastically outperforms in FLOP efficiency while maintaining accuracies of 99.5%. This research demonstrates the viability of using customized task-specific AIs for low-compute, high-accuracy applications, such as in-car driver distraction detection.

2. Introduction

More than 20% of fatal collisions in Canada are caused by distracted driving, according to the Canadian Association of Chiefs of Police [1]. The CAA also reported 80% of surveyed drivers admit to being distracted while driving [2]. The growing numbers of distracted drivers are becoming a heavy workload for human police to handle or detect on their own. That is why AI should step in.

This project aims to design a model to detect when a driver is distracted by analyzing an image taken within the vehicle. A distracted driver could be doing one of many things - such as using the radio, drinking coffee, or talking to someone in the back seat. This makes the task of identifying whether a driver is distracted or not difficult. Therefore, the right dataset of distracted driver images was necessary. These images would be used to identify common trends between different distracted drivers and flag them.

This project is also timely in that self-driving technology, especially the popular level 2 systems, may allow drivers to become more complacent and therefore easily distracted while they are on the road [3] [4]. However, this extra technology also allows the same computer systems in the vehicle to monitor the driver for distracted behavior. For example, webcams on the dashboard of the car may be installed to watch the driver.

In order to process image data in a machine learning environment, this project explores Convolutional Neural Networks (CNNs) for their ability to study what is going on within a 2D image. The model was built in Python, which has a rich set of libraries for machine learning and

is efficient in data processing, making it an ideal language for this type of project. In the end, I came up with a lightweight yet highly accurate network architecture, useful for detecting distracted driving in the computers of a car rather than a dedicated Google server.

3. Background

As recently as last year, a study researched the impact of self-driving on distraction [3] [4] [5]. This study showed a worrying trend; that semi-autonomous driving could increase distracted driving and potentially more accidents. For that reason, reducing accidents in self-driving cars is clearly an important goal that will become more relevant over time.

Other studies have already investigated the use of pre-made CNNs, such as VGG-16 and AlexNet, to detect distracted driving. [6] However, the accuracy of those models comes at the cost of a very large number of parameters and very slow runtime, making them harder to run on a device without a GPU dedicated to the model.

More broadly, there is plenty existing research on building optimally small classification models for images. SqueezeNet [9] is a model built to match AlexNet accuracies on the ImageNet dataset while using 50x less parameters. Significantly, SqueezeNet is able to fit onto specialized integrated circuits, lowering required costs and power consumption. The paper also discusses a new approach to making deep CNNs: bundling multiple convolutional layers into one module and chaining those modules together. SqueezeNet is very good at its intended purpose of high ImageNet accuracy with low parameter count. However, the number of FLOPs (Floating Point OPerations, a measure of computation) required to run SqueezeNet is almost double that of AlexNet, due to the focus on convolutional layers that use a large amount of FLOPs relative to parameters. SqueezeNet is also designed for the ImageNet dataset, which has over a million images spanning a thousand categories, a monumentally difficult task compared to transfer learning tasks which often have a lower number of categories. Thereforewhile SqueezeNet may be small for the ImageNet dataset, it is still overkill for less complex, more specialized tasks.

4. Dataset

The data chosen contained 10 different categories of driver. Here is a sample image from the dataset. (Figure 1) One of the categories was “safe driving” and the other nine were different forms of distracted driving. This dataset was generated and provided by the insurance company State Farm, as part of their research into driver safety [7]. The dataset contained over 13,000 images of resolution 360 x 240 pixels. Importantly, the images also contained a diverse set of drivers regarding gender, ethnicity and age, meaning we can be confident that the model is, or at least very close to, unbiased and will work for the whole population.

Figure 1. A sample image from the dataset

5. Methodology/Models

The State Farm dataset represents nearly perfect images, but a model trained purely on that might struggle with more difficult conditions, such as if the camera is lower resolution or if the person is simply not in the same place as the sample images. To build resilience in the model, the dataset is augmented using rotations, panning, and zoom to simulate different driver positions, and adding noise to the images to simulate real-life distortion and to prevent overreliance on specific pixels.

The initial CNN model was made using Tensorflow Keras, which is a library in Python published by TensorFlow [8]. This model helped to build out an initial attempt, which was then refined through experimenting with and reverse engineering the model.

The set of experiments occurred in two phases. The first was to get a general understanding of how different machine learning approaches responded to the image data. This would help in

validating the initial assumption that CNN's were the optimal machine learning technique for this project. The second phase was to optimize and configure the model to better fit in the real world for detecting distraction, both in regard to accuracy and stability. Different configurations were iterated on to arrive at the final model.

As part of phase one of model investigation, a linear regression model was trained on a reduced dataset of 64x64 pixel images and 4 categories. A linear regression model isn't supposed to be used on images or classification problems in the first place. The linear model did better than expected, with an accuracy of 82.3%. This was surprising, as essentially all the model was doing was drawing lines of best fit corresponding to the colors in the image. Based on these lines alone, the model was able to sort the image into one of ten categories correctly over 80% of the time. However, despite being surprisingly high, the accuracy was significantly lower than any other model, even with the reduced dataset. When looking at the model's coefficients, there appeared to be little pattern in those, suggesting that the model might have been focusing in on a few very specific pixels. There were transformations added to the reduced dataset used by the linear model, but they would have affected the linear model even more than any other type of model, as the color values of the specific pixels that the model directly relies on would fundamentally change as result, therefore preventing the model from being able to properly train. The linear model is still of note, however, and provides a useful baseline by which to compare future models.

Given the unusual success of the linear model, a special kind of neural network was investigated. A Perceptron essentially builds a linear model for each of the different output categories. The Perceptron model was applied to the full-scale dataset, using higher resolution images of 128x128 pixels. It had nearly perfect accuracy, though transformations would have still affected this model more than the CNN, given how the Perceptron focused on specific regions of the image which would have been shifted around and become inaccurate.

Since a Perceptron is very simple compared to other types of models, visualizing the model's relationship to the input for each of the unique 10 output categories is easy. If an area in the image for a specific category is bright, then brightness in that area will make the model more likely to output that category. Color in these images represent colors in the real input images. These are shown below (Figure 2):

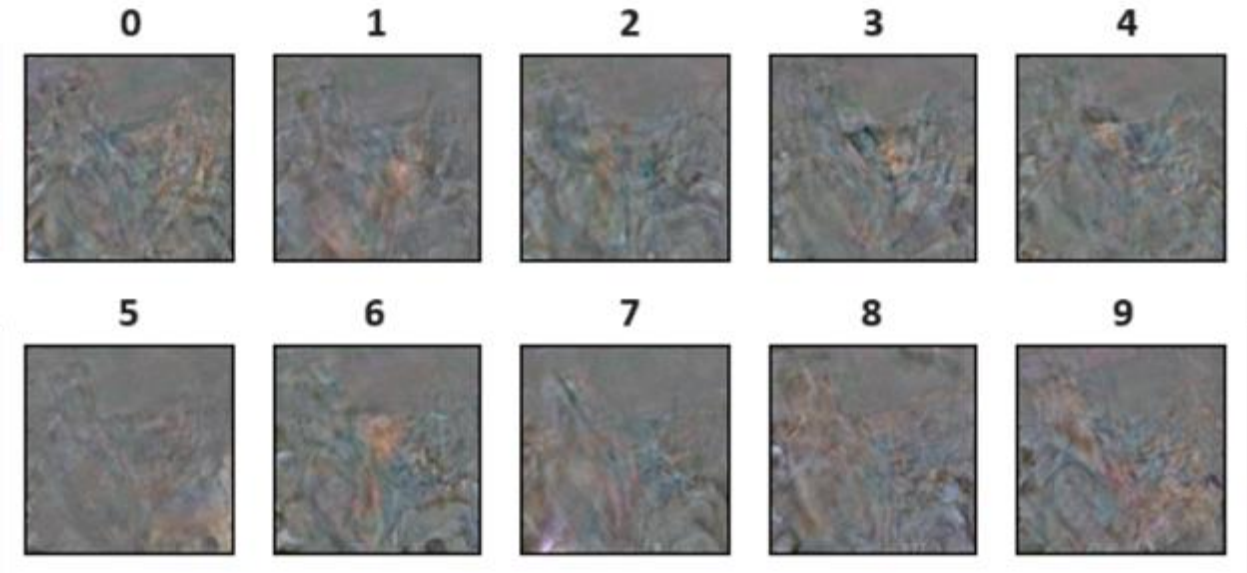

Figure 2. A visualization of the Perceptron model, showing key areas for each category

While quite messy, these outputs do seem to show areas of interest, such as the coffee cup in image 6 being used to identify that form of distraction, or the extra relevance of the bottom left region in image 7 that corresponds to reaching to the back seat.

While the accuracy was extremely high, at 99.3%, this type of model would have low accuracy when handling noise or transformations, due to the 1:1 relationship between the input and the output. For instance, if the coffee cup moved to a different part of the image, the model would ignore it, because the cup was not exactly where it was when the model was trained. The conclusion was reached that a different model type was needed, which led to the CNN architecture as the logical next step.

Due to being in the sweet spot between complexity and time efficiency as well as having the ability to deal with nosier and shifted images, a CNN is one of the most relevant types of model for real-world use. Thus, a new model was built for experimentation.

During this second phase, visualization techniques were used that allow for ‘seeing’ inside the model and get a sense for how it was learning, much like with the Perceptron. This allowed for more accurate and more effective modifications to the model. For instance, if an image representation of the model’s network showed that many of the neurons weren’t being used, a good modification would be to reduce the number of neurons so that the model would train and run more efficiently. If the evidence suggested the model could benefit from more neurons, layers were easily expanded to see how the model reacted.

Over the course of the experimentation, many different kernel and neuron configurations were tried to see which combination worked best. Increasing the number of kernels was able to offload some of the pattern recognition from the neural network, which made the model much faster. Large kernels were both able to pick up complex patterns and also deal with noise added to the dataset to simulate real world conditions.

To provide points of comparison, existing models were set up and trained on the same dataset. SqueezeNet [9] and AlexNet [10] were trained from scratch, while MobileNet V2 [11], VGG 16 [12], and Inception V3 [13] were used with transfer learning. AlexNet was able to use augmented data to train. However, the other models would fail to generalize to the validation set and necessitated using unaugmented data to train on. Then, the number of FLOPs, number of parameters, unaugmented accuracy, and augmented accuracy were compared between all of the models.

6. Results and Discussion

After training for 120 epochs on augmented data and 20 epochs on unaugmented data, the model has 99.5% accuracy on the unaugmented validation set. Here is the confusion matrix. As expected from a very accurate model, all but a few predictions match the true label.

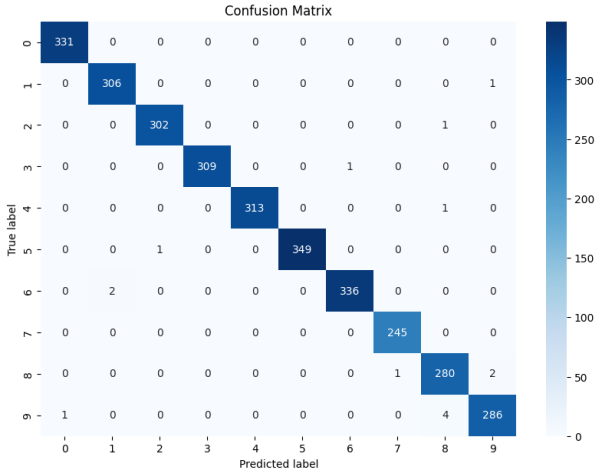

Figure 3. A confusion matrix showing the accuracy of the model on unaugmented data

While the model trained on the augmented training set, the images are augmented differently every time the training set is accessed. As a result, the accuracy on the augmented training set is lower than the validation set.

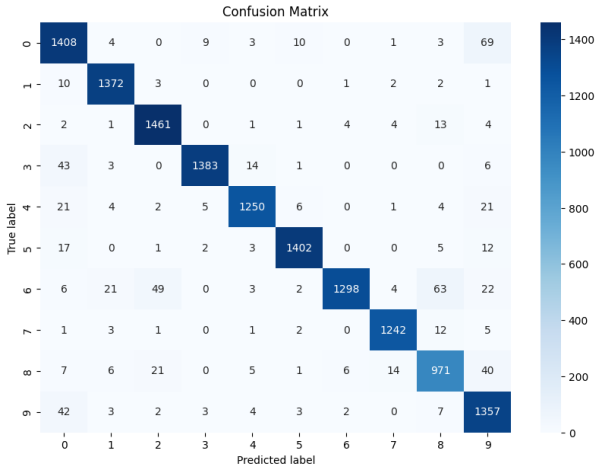

Figure 4. A confusion matrix showing the accuracy of the model on augmented data

At the end of training, the model had 147,926 parameters and ended up at 611 KB in size. This is significantly less than previous iterations of the model, which had 372,114 parameters, and was 2.8 MB in size.

As shown in the graph below, the Custom Conv Model (the same model mentioned before) is significantly smaller and faster than any of the other pre-existing models. While SqueezeNet may be able to compete in terms of number of parameters, no model is able to achieve a comparably low number of FLOPs. Interestingly, model FLOPs aren't necessarily correlated with model size. For example, convolutional layers apply a relatively small number of parameters many times, driving up FLOPs while not contributing much to the model size.

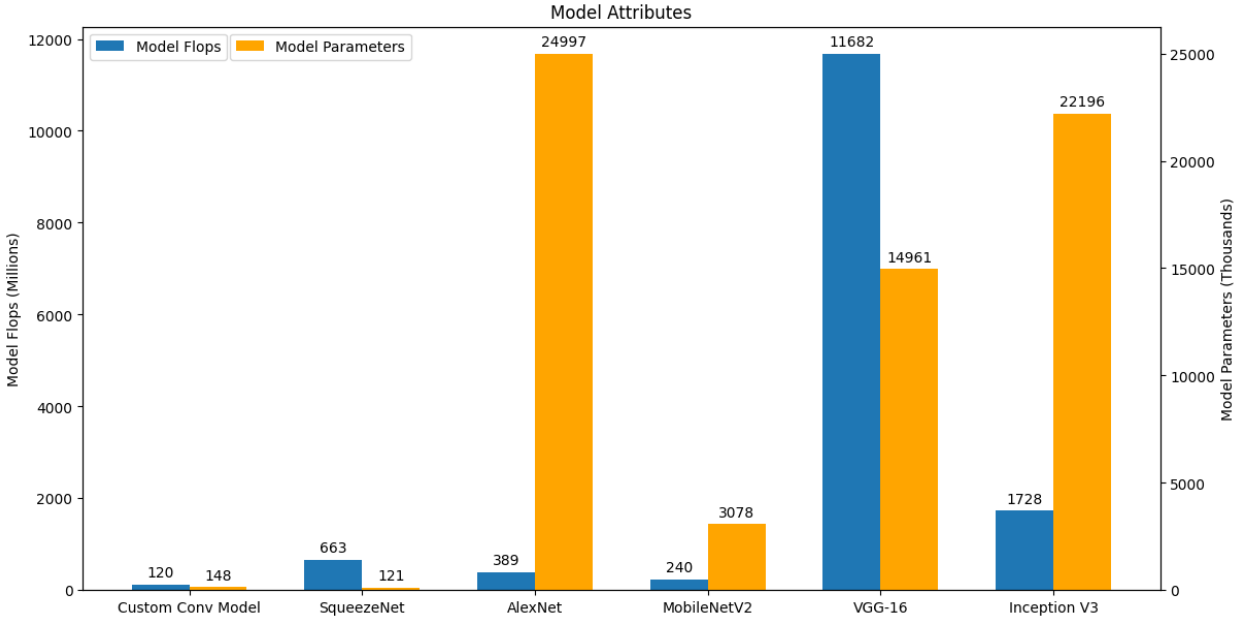

Figure 5. A comparison of performance characteristics between six different models

The relative accuracies of the model are shown below. The most relevant point of the graph is that all the models are very accurate on the testing dataset, so the usefulness of each model is comparable to all other models. The augmented accuracy would represent a worse input, such as the camera having noise or not positioned correctly. Note that only the Custom Conv Model and AlexNet were trained using augmented data, while the other models were unable to converge on augmented data. One interesting fact is that if the Custom Conv Model is initially trained on unaugmented data, it will start overfitting quickly. Training on augmented data before training on unaugmented data is required to find a global optimum rather than getting stuck in a rut.

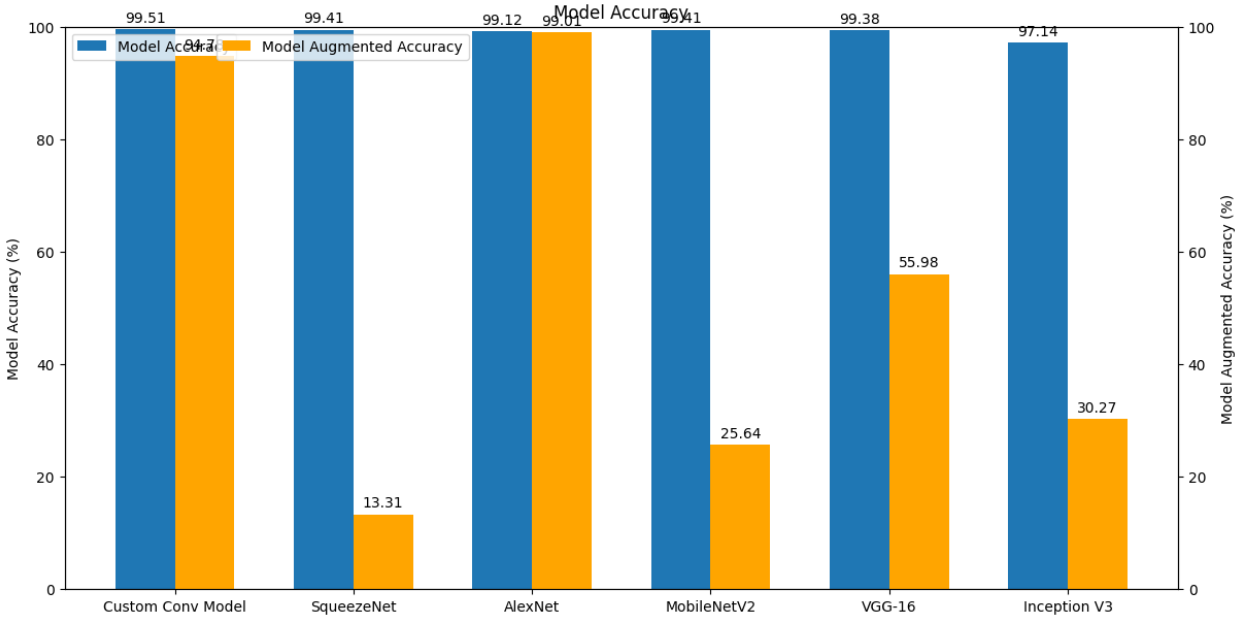

Figure 6. A comparison of accuracies between six different models

The quality of the results is apparent and come as a process of experimentation and observation of the adjustment of hyperparameters. This is non-trivial, and time expensive in the development cycle, vs. simply using AlexNet and having a good, but grossly over-large model. There are strategies that would streamline this process, and other techniques and methodology that could further allow for the discovery of optimal model structure.

First, visualization tools are critical in examining how the model is performing. The presence of ‘dead’ neurons, or malfunctioning kernels for these structures was important in determining whether the model had too many, or too few.

7. Conclusion

My model is able to compete with other more complex models in terms of accuracy, while requiring less resources to run and less space to store. It can also generalize to more challenging conditions such as rotations and noise.

This study demonstrates the efficiency of smaller models in simpler tasks, rather than using a massive pretrained model. When using AI, it's common to see existing models repurposed to new tasks. But blindly applying large models to simple tasks is wasteful and inefficient. Using a compact model reduces the time needed to train and to apply, allowing systems without dedicated GPUs to run the model alongside other tasks.

One limitation of the study is the absence of guarantee that the presented model is the smallest possible model. Theoretically, there could be a smaller model than the one in the paper that has the same capabilities, which hasn't been rigorously looked for. Another limitation is the application of CNNs focused on distracted driving. Other potential applications, such as facial analysis, have not been examined.

Future work could be done to integrate a small model into a self-driving car and test how well it works in the real world. Additionally, the effectiveness of alerting the driver when they are distracted could also be measured. Future research may also aim to implement this prototype in other applications, for example facial analysis, to further demonstrate the broad effectiveness of this study's approach.

The approach of a small and dedicated model for a relatively simple task can be easily applied to other areas. If there are only a limited number of categories, a model that collects a large amount of information is unnecessary and a smaller model should be considered. When building the model, spending more time during training can lead to massive speedups in real-world use.

8. Acknowledgements

I would like to thank my mentor, Simeon Sayer, for giving me guidance to start building the CNN model.

I would also like to thank my family, for giving me advice and support throughout the project.

9. References

[1] Canadian Association of Chiefs of Police. “CANADA ROAD SAFETY WEEK – NATIONAL FACTS AND STATS May 16-24, 2023.” *CACP*, 24 May 2023, www.cacp.ca/_Library/Documents/20230510152555172895795 5_crsw2023may1624nationalfactsstats.pdf.

[2] “Distracted Driving Statistics.” *CAA National*, 16 Nov. 2022, www.caa.ca/driving-safely/distracted-driving/statistics/.

[3] Biondi, Francesco N., and Noor Jajo. *Human Factors Assessment of On-Road L2 Driving: Recommendations for the Implementation of Partially-Automated Vehicles.*, autoevolution, Oct. 2023, www.autoevolution.com/pdf/news_attachements/attention-span-goldfish-new-research-reveals-driving-an-l2-vehicle-can-be-twice-as-deadly-223496.pdf.

[4] Waddell, Dave. “Preliminary Findings of University of Windsor Study Shows Driver ...” *Windsor Star*, 12 Jan. 2023, windsorstar.com/news/local-news/preliminary-findings-of-university-of-windsor-study-shows-driver-assistance-systems-reduces-attentiveness.

[5] Dunn, N., Dingus, T.A. & Soccolich, S. (2019). *Understanding the Impact of Technology: Do Advanced Driver Assistance and Semi-Automated Vehicle Systems Lead to Improper Driving Behavior?* (Technical Report). Washington, D.C.: AAA Foundation for Traffic Safety.

[6] Baheti, Bhakti, Suhas Gajre, and Sanjay Talbar. "Detection of distracted driver using convolutional neural network." *Proceedings of the IEEE conference on computer vision and pattern recognition workshops.* 2018.

[7] SEOK JEONG-RO. “State Farm Distracted Driver Detection.” *Kaggle*, State Farm, 2021, www.kaggle.com/datasets/rightway11/state-farm-distracted-driver-detection.

[8] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden,

Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.
TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[9] Iandola, Forrest & Han, Song & Moskewicz, Matthew & Ashraf, Khalid & Dally, William & Keutzer, Kurt. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size.

[10] Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger (ed.), Advances in Neural Information Processing Systems 25 (pp. 1097--1105) . Curran Associates, Inc.

[11] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2019). MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1801.04381

[12] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1409.1556

[13] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1512.00567