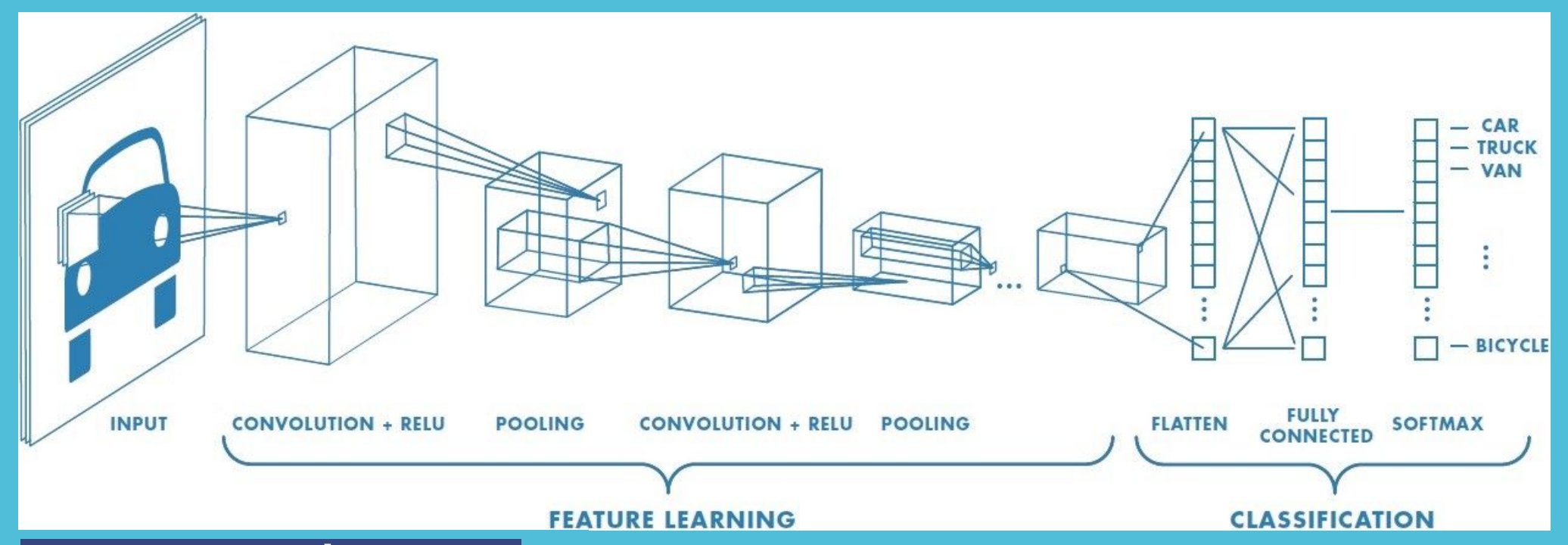
Convolutional Neural Network

Ian Ryan • Dr. Soltys • COMP 499 - Capstone

Channel Islands
CALIFORNIA STATE UNIVERSITY

Introduction

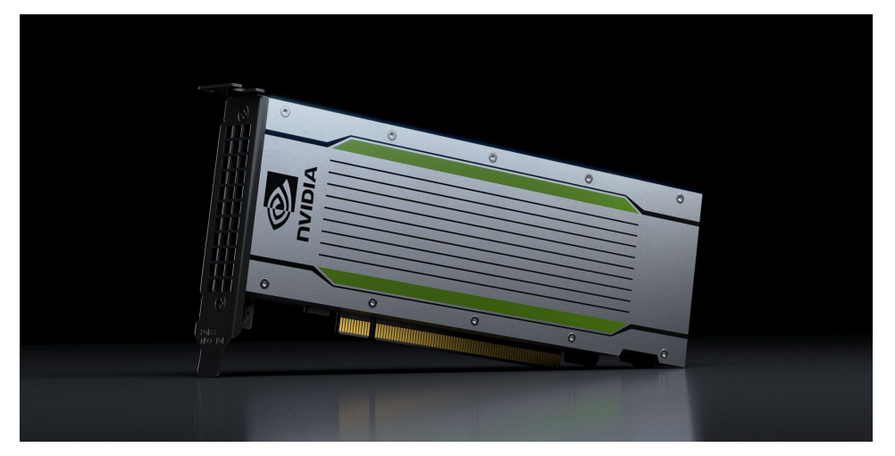
This semester I have chosen to pursue working on a Convolutional Neural Network (CNN) project. In recent years as Artificial Intelligence and Machine Learning have become synonymous with our everyday lives. Despite the current hotspot, we are seeing more and more videos in media of machines utilizing computer vision. The prerogative has been create a CNN and get the model to perform at an acceptable level. Then the goal was to take the project a step further and to learn PyTorch and YOLOv8 along with going from a CNN to a ResNet-18. After successfully processing the data I was able to make adjustments and increase the accuracy tenfold. I started with INRIA Human Dataset which contained 902 human images, To have a more balanced model I combined it with a dataset called Animals which I refined to 902 images of non-humans to a total of 1804 images and 18 classes. To take a stab at Object-Detection as opposed to the classification CNN model I crafted. I decided to try and connect my model to a live camera feed and accurately have boxes bound around humans by creating a YOLOv8 algorithm from the datasets that I concocted. using my own CNN Model instead of pre-trained. It is beneficial for every industry, farmers are able to make more informed decisions about their crops and automate mundane tasks. First responders are able to use drones for example to find missing or injured citizens or enemies in a warzone like we have seen in Ukraine and Palestine/Israel. It is becoming more accessible to everyday people and in turn allows for different perspectives and innovations.



"Image courtesy of [MathWorks]"

Specifications

- **AWS SageMaker Jupyter Notebook**
- **Notebook Instance Type:** ml.g4dn.2xlarge
- **Processor:** NVIDIA T4 Tensor Core GPU with 16 GB GDDR6 VRAM
- **Memory:** 32 GB System RAM
- **Storage:** 5 GB EBS Volume
- **OS:** Amazon Linux 2 with Jupyter Lab 3 (notebook-al2-v2)
- **Lifecycle Configuration:** None
- **Elastic Inference:** Not Enabled
- **Minimum IMDS Version:** 2

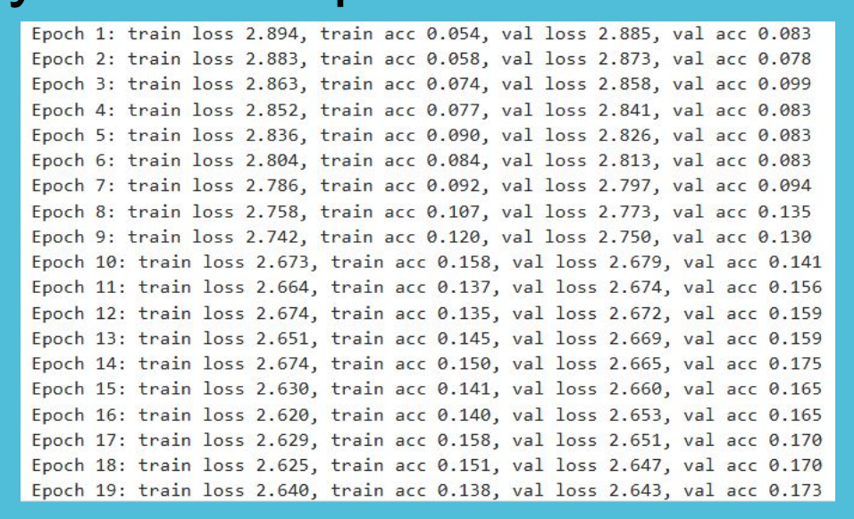


Instance Type	vCPUs	Memory (GiB)	GPU	GPU Memory (GiB)	Storage (GB)	Network Bandwidth (Gbps)	Price (On-Demand)
g4dn.2xlarge	8	32	1	16	225 NVMe	Up to 25	$0.752/hour

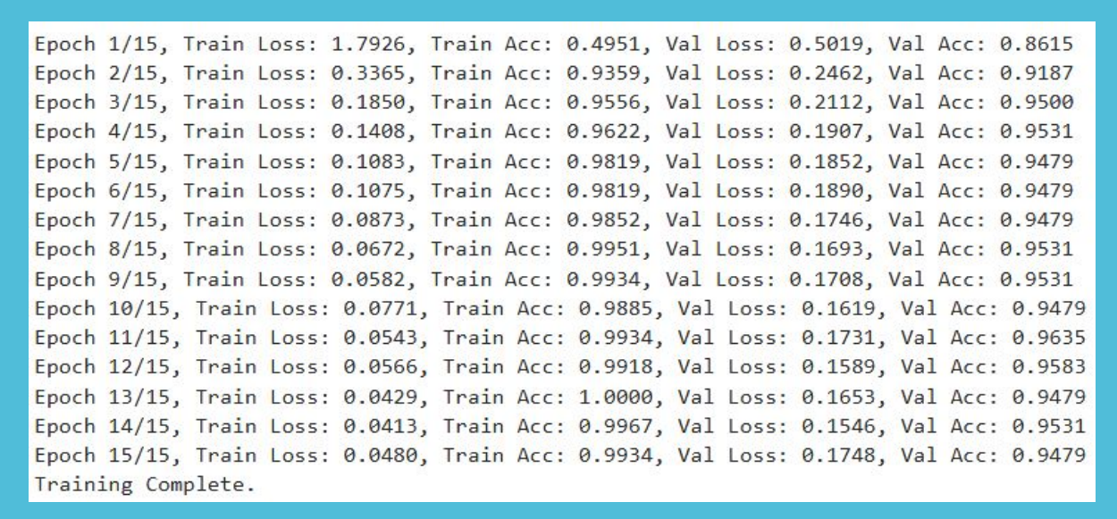
Methodology

When I first started Capstone my goal for this project was to cover the entire process of turning raw data into datasets and utilizing them on neural networks. The first step was to find a dataset that I could train a model on to an acceptable degree. After finding the INRIA Human Dataset I realized that despite having 902 images, they were all of humans and since we needed images of non-humans and landscapes. That is when I stumbled upon the Animal Dataset that I matched the number of non-human pictures. This gave me over 1800 images after countless hours of adjusting preprocessing values. There were some redundant actions but the amount of time it took me a while to resize the animal dataset and normalizing the images it took longer than expected and some adjustments to allow the data to be combined into a single dataset for preprocessing and move on to the processing phase. Everything was evaluated before making CNN models. To learn from my mistakes.

Here is a screenshot of my first attempt at training a model (a lousy 17%) followed by a screenshot of my best trained model a ResNet-18 (96%) which is a CNN but has residual layers meaning that every time there is a new layer, the new layer can skip a neural connection need be. This leaves ResNets with a higher accuracy than dense layers in deep neural networks



```
Epoch 1: train loss 2.894, train acc 0.054, val loss 2.885, val acc 0.083
Epoch 2: train loss 2.883, train acc 0.058, val loss 2.873, val acc 0.078
Epoch 3: train loss 2.863, train acc 0.074, val loss 2.858, val acc 0.099
Epoch 4: train loss 2.852, train acc 0.077, val loss 2.841, val acc 0.083
Epoch 5: train loss 2.836, train acc 0.090, val loss 2.826, val acc 0.083
Epoch 6: train loss 2.804, train acc 0.084, val loss 2.813, val acc 0.083
Epoch 7: train loss 2.786, train acc 0.092, val loss 2.797, val acc 0.094
Epoch 8: train loss 2.758, train acc 0.107, val loss 2.773, val acc 0.135
Epoch 9: train loss 2.742, train acc 0.120, val loss 2.750, val acc 0.130
Epoch 10: train loss 2.673, train acc 0.158, val loss 2.679, val acc 0.141
Epoch 11: train loss 2.664, train acc 0.137, val loss 2.674, val acc 0.156
Epoch 12: train loss 2.674, train acc 0.135, val loss 2.672, val acc 0.159
Epoch 13: train loss 2.651, train acc 0.145, val loss 2.669, val acc 0.159
Epoch 14: train loss 2.674, train acc 0.150, val loss 2.665, val acc 0.175
Epoch 15: train loss 2.630, train acc 0.141, val loss 2.660, val acc 0.165
Epoch 16: train loss 2.620, train acc 0.140, val loss 2.653, val acc 0.165
Epoch 17: train loss 2.629, train acc 0.158, val loss 2.651, val acc 0.170
Epoch 18: train loss 2.625, train acc 0.151, val loss 2.647, val acc 0.170
Epoch 19: train loss 2.640, train acc 0.138, val loss 2.643, val acc 0.173
```

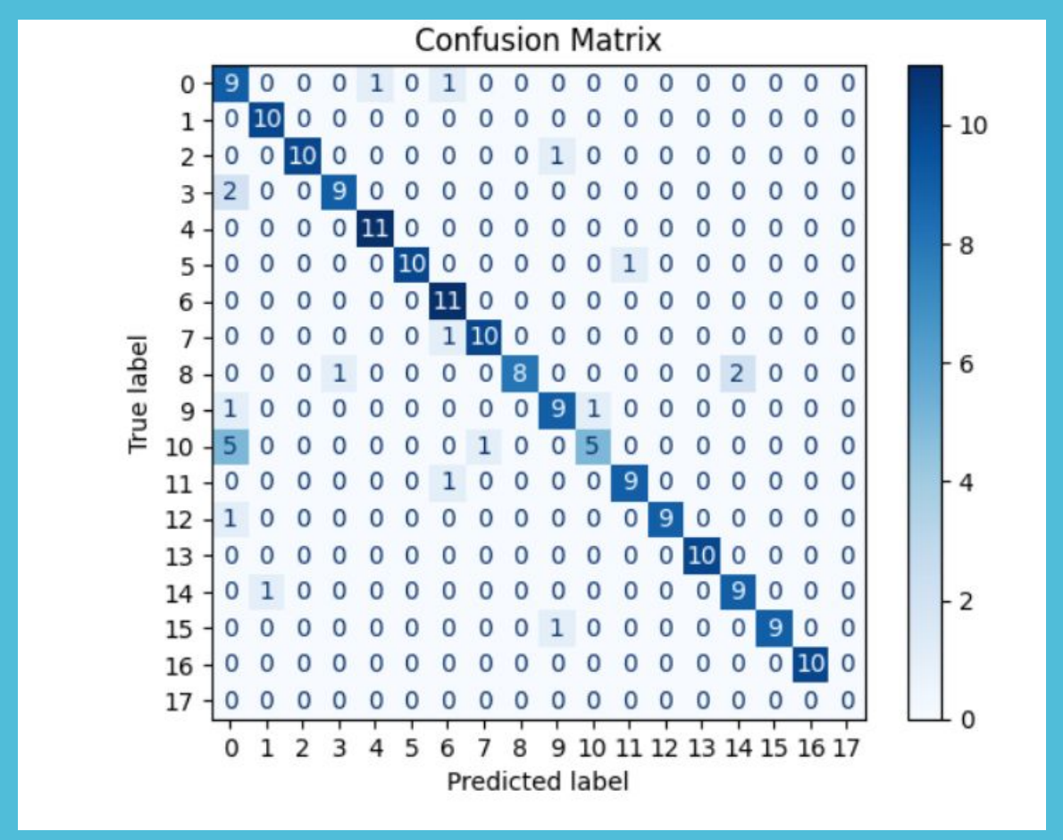


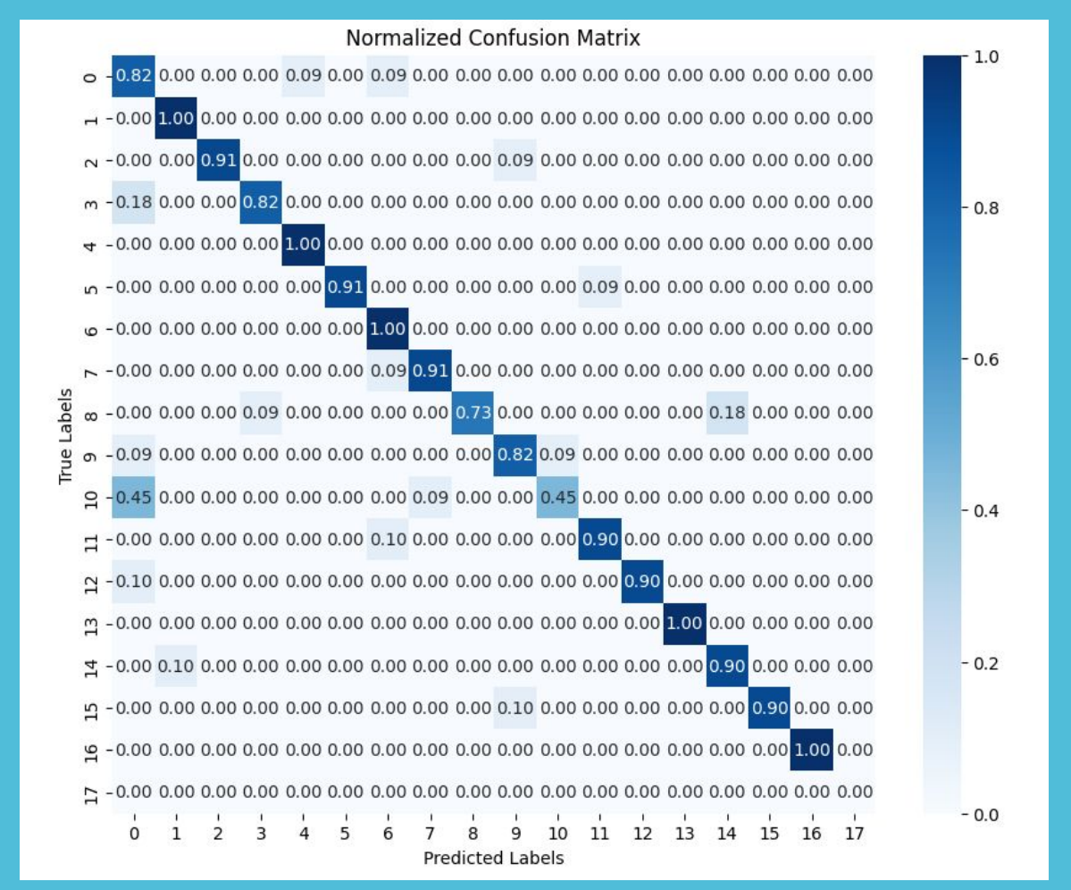
```
Epoch 1/15, Train Loss: 1.7926, Train Acc: 0.4951, Val Loss: 0.5019, Val Acc: 0.8615
Epoch 2/15, Train Loss: 0.3365, Train Acc: 0.9359, Val Loss: 0.2462, Val Acc: 0.9187
Epoch 3/15, Train Loss: 0.1850, Train Acc: 0.9556, Val Loss: 0.2112, Val Acc: 0.9500
Epoch 4/15, Train Loss: 0.1408, Train Acc: 0.9622, Val Loss: 0.1907, Val Acc: 0.9531
Epoch 5/15, Train Loss: 0.1083, Train Acc: 0.9819, Val Loss: 0.1852, Val Acc: 0.9479
Epoch 6/15, Train Loss: 0.1075, Train Acc: 0.9819, Val Loss: 0.1890, Val Acc: 0.9479
Epoch 7/15, Train Loss: 0.0873, Train Acc: 0.9852, Val Loss: 0.1746, Val Acc: 0.9479
Epoch 8/15, Train Loss: 0.0672, Train Acc: 0.9951, Val Loss: 0.1693, Val Acc: 0.9531
Epoch 9/15, Train Loss: 0.0582, Train Acc: 0.9934, Val Loss: 0.1708, Val Acc: 0.9531
Epoch 10/15, Train Loss: 0.0771, Train Acc: 0.9885, Val Loss: 0.1619, Val Acc: 0.9479
Epoch 11/15, Train Loss: 0.0543, Train Acc: 0.9934, Val Loss: 0.1731, Val Acc: 0.9635
Epoch 12/15, Train Loss: 0.0566, Train Acc: 0.9918, Val Loss: 0.1589, Val Acc: 0.9583
Epoch 13/15, Train Loss: 0.0429, Train Acc: 1.0000, Val Loss: 0.1653, Val Acc: 0.9479
Epoch 14/15, Train Loss: 0.0413, Train Acc: 0.9967, Val Loss: 0.1546, Val Acc: 0.9531
Epoch 15/15, Train Loss: 0.0480, Train Acc: 0.9934, Val Loss: 0.1748, Val Acc: 0.9479
Training Complete.
```

Changing neural network models improved my understanding of more specialized or specific models and usecases for them. Overfitting was another area that was consuming and was either because the model in question is too complex. We encountered appropriate fitting as well as underfitting while debugging the project. Early stopping a model in training, the weight decay, and the dropout amount are all useful to prevent overfitting. Loss functions are valuable in the optimization process.Currently the project is still underway and I am working to implement the YOLOv8 algorithm for live video coverage.

Results

Post Model Convolutional Neural Network Confusion Matrix (Left) followed by Post Model ResNet-18 Confusion Matrix (Right).





For my CNN the best accuracy I obtained was 84%. For my ResNet-18 that I will be using with the YOLO algorithm hopefully by new years was able to achieve 99% multiple times so I need to adjust the classes.

I made progress getting data and annotations to work with COCO and JSON and pathing in the notebook environment.

Literature Cited

Brownlee, J. "How to Develop Convolutional Neural Network Models for Time Series Forecasting." *Machine Learning Mastery.* Available at: https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/.

Eliot, D. *Deep Learning with PyTorch Step-by-Step: A Beginner's Guide, Volume I: Fundamentals.* Self-published, 2020.

Dougherty, G. *Pattern Recognition and Classification: An Introduction.* Springer, 2012.

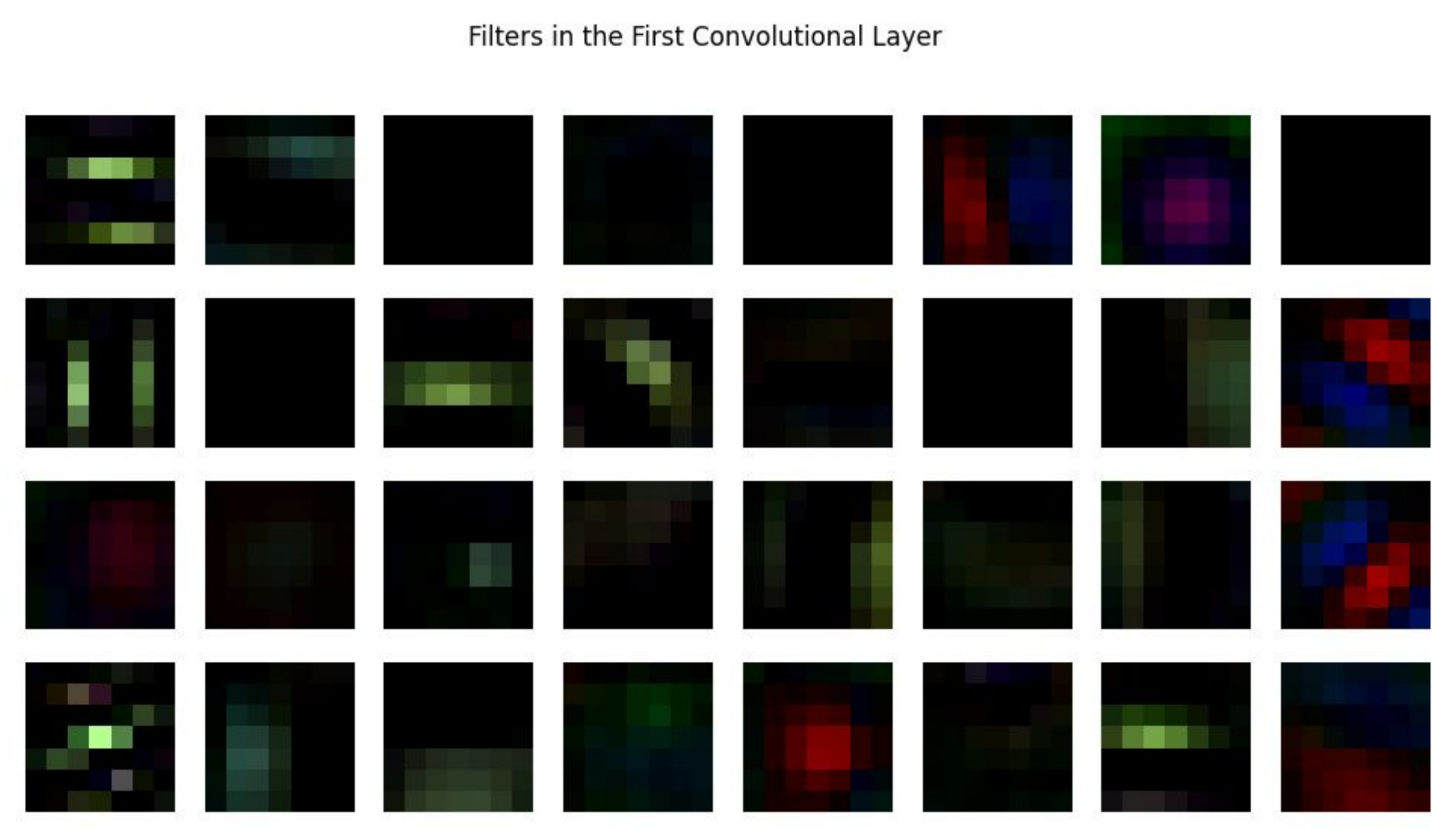
Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.* arXiv preprint arXiv:1901.08688.

Conclusions

In farewell, this project provided a valuable opportunity to explore the practical application of Convolutional Neural Networks and advanced techniques like ResNet-18 and YOLOv8. Starting with a balanced dataset of human and non-human images, I was able to refine my CNN model and significantly improve its accuracy. Transitioning from image classification to object detection with a live camera feed highlighted the versatility and power of computer vision technologies.The broader implications of these advancements, from aiding farmers and first responders to fostering innovation in everyday life, emphasize the importance of democratizing AI tools. Overall, this project not only strengthened my technical skills but also underscored the potential of AI to address real-world challenges.

Acknowledgements

I would like to thank Professor Soltys and Professor Barber for introducing me to and peaking my interest in Machine Learning. Having two mentors both of whom work with Pattern Recognition and ML problems in their fields as a computer scientist and a physicist. They are both so knowledgeable and look at problems in different ways.



Gradient Visualization - Visualize the gradients to see which parts of the input are most important for the model's predictions.