Executive Summary

Road Sign Image Detection Machine Learning Analysis Report

## Presented By:

Becca Roeth

Ethan Houser

Kyle Bell

Kyle Parks

Makayla Whipple

Chantal Borchardt

# I. Introduction Summary

We were tasked with building a machine learning model that is able to detect road sign images for GehirnWagen, a German tech startup, to help with their self-driving car initiative. To do this, we need to use a Convolutional Neural Network (CNN) model.

We started by importing and running different pre-build models, while slightly adjusting the parameters, amount of layers and a little bit of data augmentation. After running the different models, we discovered that ResNet50 was the highest performing, using the f1-score measurement of performance. The models that we tried initially tried were:

* ResNet50 - accuracy: 96%
* VGG-16 (VGSystem) - accuracy: 89%
* Inception - accuracy: 80%

## Roadblock: Computational Power

The only issue that we faced was the time required to run the various models. To be sure that we ran a sufficient number of models, the six team members all ran different variations of the model to determine the best hyperparameters.

# II. Addressing Questions

When it comes to building a neural network that can correctly classify German road signs we were able to attempt many different convolutional network approaches including, VGG-16, ResNet, and Inception and found that ResNet was able to have the best performing base model overall. After adding data augmentation and adding layers to the models both Inception and ResNet were able to have models that performed better overall. When it came to our main model we chose to apply ResNet due to performing best overall.

Preprocessing the data is essential to improve the quality and performance of the model. When it came to improving performance of the model we chose to apply data augmentation for many different models. Some of the data augmentation that we applied included width shift, height shift, zoom, brightness, and noise.

Data Augmentation:

* **Zoom**: this allows the model to learn and recognize features that are different scales and sizes.
* **Brightness:** This helps the model train on images that may be too dark or too light.
* **Noise**: This helps the model train on possible imperfections within the images. For example when it comes to signs there may be scenarios where the sign may have graffiti on them or have items obscuring the view.
* **Width and Height Shift:** This helps the model recognize positions of objects within the image.

The ones that were shown to be most successful were noise and rotation.

1. This seems like one of those cases where straight accuracy might not be the best metric for model evaluation, but what do you think?

F1 score is the metric that best represents the success of our model. It compares correct predictions in proportion to incorrect predictions. We chose F1 because false positives (for example: Stopping at a stop sign that does not exist) are equally as dangerous as false negatives (such as ignoring a real stop sign).

1. What performance would you require before you’d ride in the car?

* In order to be comfortable while riding the car using our model, we would need an f-1 score of at least .98. Our model is able to meet those expectations.
* Only some of the signs have an F1 score of 1.00, but all signs had an F1 score of 0.9 or higher, with three exceptions as noted in “Issues with the Data”.
* All stop signs had a recall score of 1.00.
* The model has an overall accuracy of 99%.

# III. Data Exploration



Figure 3.1:

This figure is able to show that our main model accurately predicts each class for most of the signs. The true label is able to represent the actual class of the sign, while the Predicted class represents what our model was able to predict.



figure 3.2

This figure shows that the model is able to accurately predict and recognize stop signs in various scenarios.

## Issues with the Data

Issues that cannot be changed because this is how they exist in the real world and should be kept in the data

* Some pictures are very blurry or dark
* Some signs have graffiti on them

Three signs were missing from the Holdout dataset: “left curve”, “roundabout”, and “speed limit 20”. While our model is trained to identify these signs, without further field testing we cannot make claims about whether these specific signs will be identified accurately.

# 

# IV. Machine Learning Model

## Initial Models

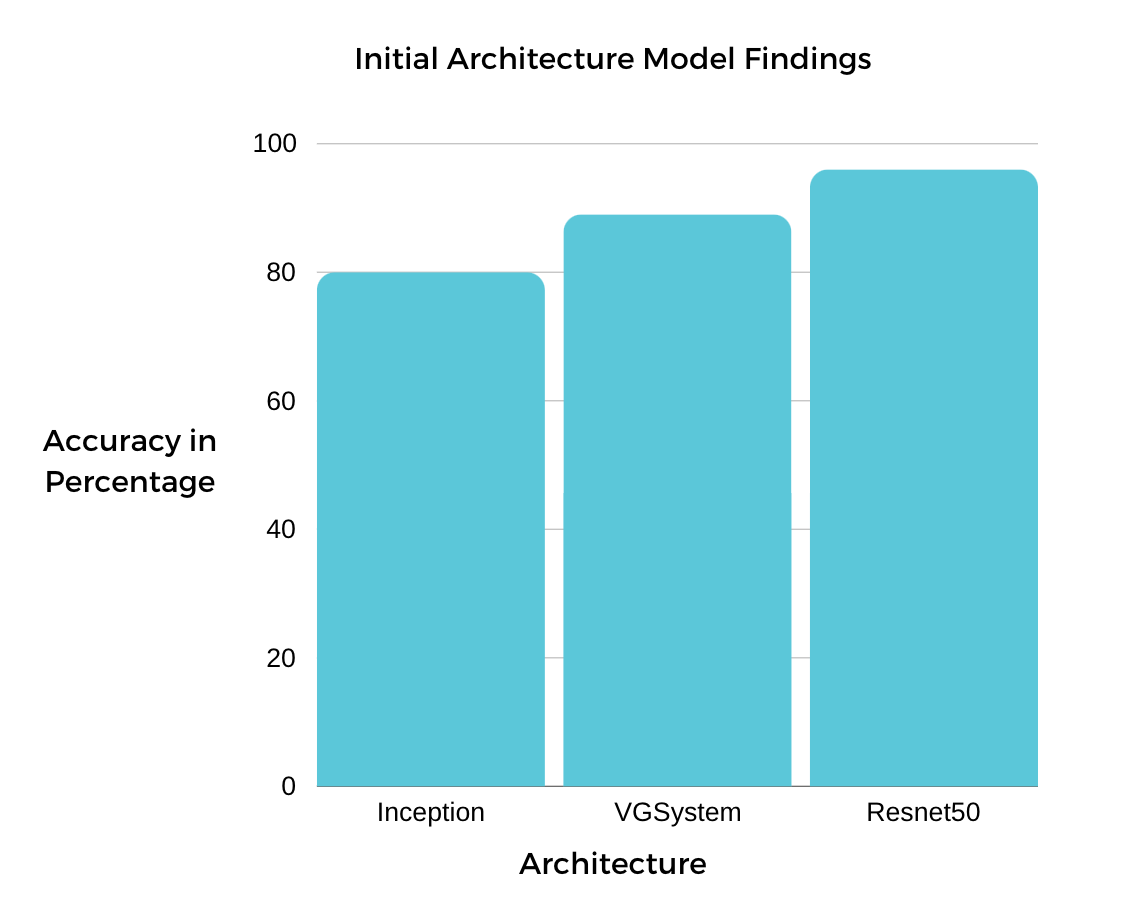


Fig. 4.1 Shows the percentage from the validation accuracy score

of the architectures tried turning the exploratory stage.

While we will not use accuracy as the main scoring method, the accuracy score was what was recorded in the exploratory stage and shows what is and isn’t an effective architecture for our model.

* Note: Resnet50 Model already used some of the Model Methods and may be biased

### **Regularization**

* Regularization (20 degrees)
  1. **Preventing Overfitting**: By penalizing large weights, L2 regularization encourages the model to favor simpler solutions and prevents it from memorizing noise or irrelevant details in the training data.
  2. **Improving Generalization**: Overfitted models often perform well on training data but fail to generalize to new, unseen data. L2 regularization helps mitigate this issue by promoting models that generalize better to diverse datasets.
  3. **Enhancing Model Stability**: Regularizing the weights stabilizes the training process, reducing the risk of large weight updates that can lead to model instability or divergence during optimization.
* Dropout Regularization
  1. During training, randomly-selected neurons of the model are “dropped out”. This helps the model avoid getting “stuck” and over relying on those specific parts. The model becomes more resilient to unexpected situations.
* Gaussian Noise
  1. We used Gaussian Noise to introduce randomness to the features learned by the model. This allows the model to avoid an over reliance on specific features. This allows it to learn more general patterns and make more accurate predictions.

## Model Construction

In both models:

* Basic data augmentation was used through tensorflow’s ImageDataGenerator function.
* Using ImageDataGenerator, our available data was randomly split into a training set and validation set.
* The Model analyzes an image, identifying groups of pixels that represent specific types of lines or shapes.
* The output determines which of 43 signs an image is likely to be.
* The model is saved in the CoreML “.mlprogram” format compatible with the Apple ecosystem

## Pretrained Model

* The base model is ResNet50 from tensorflow, using weights trained on the “ImageNet” image database.
  + Because the model was already trained to recognize elements of images, we only needed to fine-tune its training to recognize German street signs.
* The base model was not frozen- that is, it was allowed to change its weights to better classify German road signs.
* A layer of Gaussian Noise just before the final decision-making layer “regularizes” the features the model detects. This forces the model to learn more general patterns instead of over relying on specific details.

## Custom Model

* Sequential Model
  + Our model was constructed by hand-picking which machine learning elements to include, and in which order.
  + Because this model’s structure was built from scratch, it required more time and resources spent on training, when compared to the ResNet50 model
* The pooling layers used were the Keras layer function MaxPooling 2D using a 3X3 filter grid size.
* Gaussian Noise layer as a means of data normalization for the model to prevent overfitting.
* This model had fewer and less densely-interconnected layers that the ResNet50 model, giving less potential to learn patterns in the images.

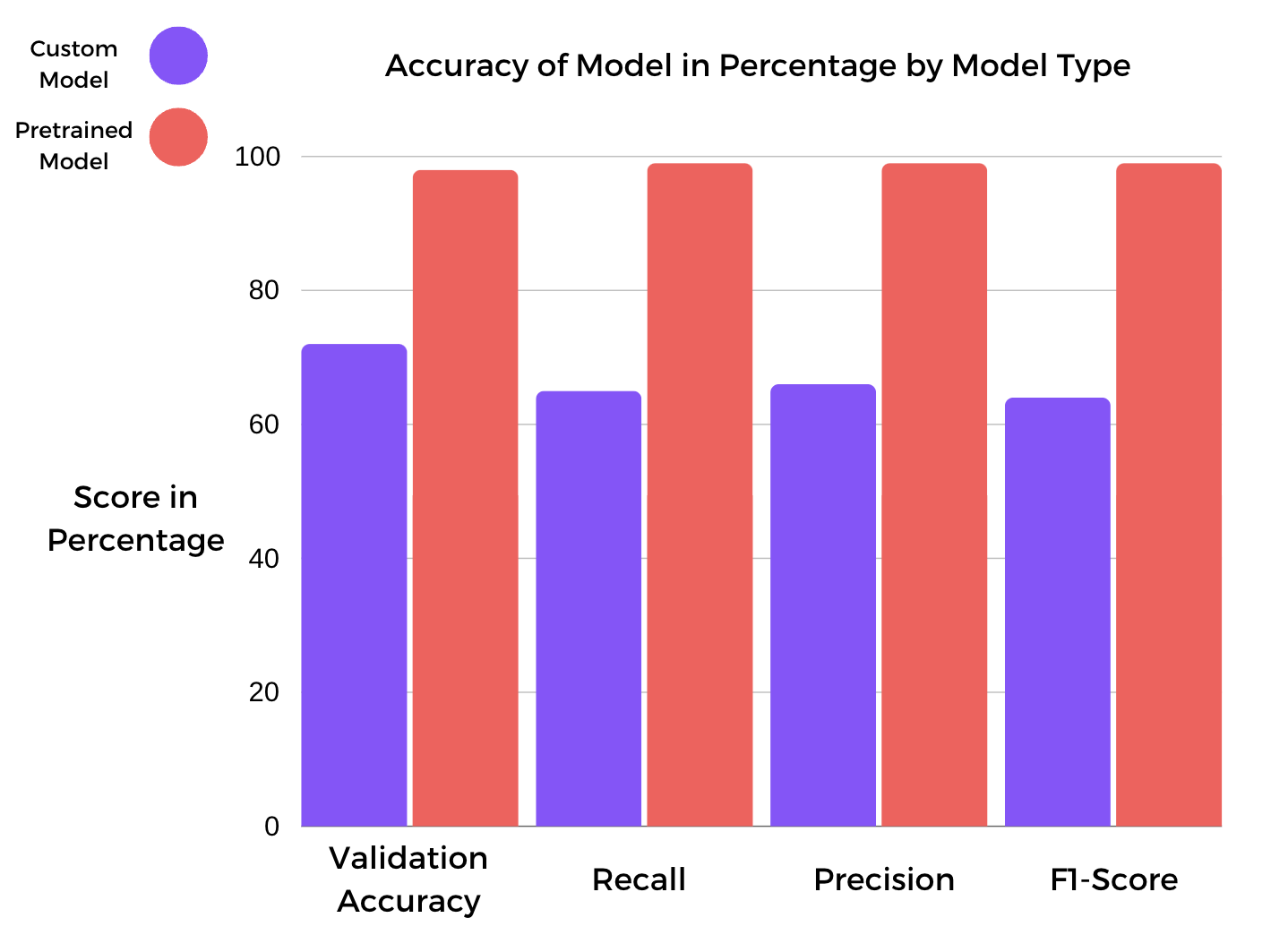


Fig. 4.2 Scores based off of the Custom and Pretrained Model in percentage form. The recall, precision, and F1-Scores are based off of the weighted average but showed similar trends for macro average as well.

# V. Results, Action Items, and Limitations

## Results

| **SCORES** | **RESNET50 MODEL** | **CUSTOM MODEL** |
| --- | --- | --- |
| **F1 Score** | 99% | 64% |
| **Validation Accuracy** | 98% | 72% |
| **Recall Score** | 99% | 65% |

**F1 Score -** This is a metric that combines the precision and recall scores. It is able to assess the performance of a model by taking into account class-wide performance rather than an overall accuracy.

**Validation Accuracy -** This refers to how well our convolutional network classifier performed on a validation set.

**Recall Score -** This metric measures the model's ability to correctly identify positive predictions.

## Action Items & Conclusions

* Our team recommends using the pretrained model. Creating a custom model from scratch requires time spent on experimentation in an already well-researched field.
* Future Refining of the Neural Network should include video data. Static pictures are a good start, but the intended product will be in motion.
* Have an in-house team primarily responsible for maintaining the traffic recognition system.
* Extensive field testing is highly recommended. Make sure to gather quality data from said testing for further improvement of the self driving system.

## Limitations

* The models presented are simply a good start. Neural Networks need to be constantly improved and updated.
* Your traffic recognition system is the backbone of your product. Safety is the first priority when it comes to any type of car.
* Training a model from scratch for commercial purposes requires millions and millions of data points.

# VI. Python Notebooks

Transfer Learning Master:

<https://colab.research.google.com/drive/1QQpOFjaLhELd0Nvs0ySwbq4uTwzurvsy#scrollTo=mA0HPVmIBT4C>

Custom Model Master:

<https://colab.research.google.com/drive/1I9ctH3mXPvwwh3vyBnbadlcm-mTVgZr9?authuser=2#scrollTo=nYlsUffNcQY->