Infrared and RGB Image Processing w/ Deep Neural Networks

J.B. Speed School of Engineering: CSE547-51

Phoenix Daak

April 24, 2025

Presentation Link: 2025-04-24 17-02-36.mov

I. INTRODUCTION

This project covers image processing with Infrared and RGB image data for autonomous driving. Convolutional neural networks (CNN) were constructed from scratch, constructed with pretrained VGG16 models, and lastly constructed with Auto Encoding models paired with classification layers. Networks were built for each type of datasets, fine tuning the models based on the type of data being fed into the networks. Complexity of the models were also experimented, varying the height and width of the entire network. Regularization was used to improve generalization between training and validation. The models were evaluated where the predictions were compared from one model to the next.

II. SCRATCH CNN

A. Outline

For the first experiment of this project, a convolutional neural network was built from scratch for both IR data and RGB data. The number of layers were tested to increase the overall complexity of the network. The number and size of filters were experimented as well. To address overfitting, L1, L2, and dropout layers were tested with the network and eventually used in the more complex models. Lastly, augmentation was used to generate more image data from the existing dataset. Combining these features produced generalized models with high accuracy scores. At the end of this experiment, a confusion matrix was generated using test data that was split on the validation dataset. This will be shown for each data type.

B. Infrared Data

A base model was constructed to define a baseline for the following experiments. This base model started simple with the architecture. It was made up of an input layer with size of 64x64x3, followed with 3 CNN layers and 2 dense layers for classification. Each layer besides the output layer utilized rectified linear unit as the activation function. The CNN layers varied in the number of 3x3 filters, with increasing number of filters. The output layer utilized the 'softmax' activation for categorical classification. The architecture of the base model is shown below:

Layer (type)	Output Shape	Param #
conv2d_35 (Conv2D)	(None, 62, 62, 32)	896
conv2d_36 (Conv2D)	(None, 60, 60, 64)	18,496
conv2d_37 (Conv2D)	(None, 58, 58, 128)	73,856
flatten_8 (Flatten)	(None, 430592)	0
dense_21 (Dense)	(None, 64)	27,557,952
dense_22 (Dense)	(None, 8)	520

Figure 1. Base model architecture

The total trainable parameters were 27,651,720. After compiling the model, it was then trained on the provided training dataset. This model utilized early stopping and completed the training at 19 epoch. History of the training process was then plotted to analyze the generalization.

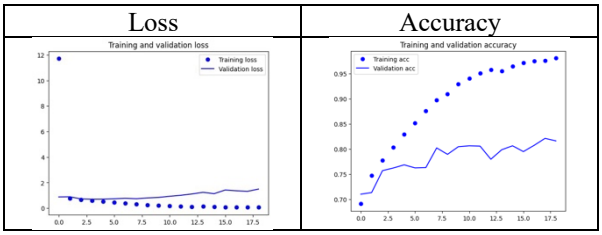


Figure 2. Training and validation loss/acc

The base line model started overfitting instantly during training. The validation accuracy reached ~82%. To increase the overall accuracy, the complexity of the base model was increased.

To increase the complexity, additional CNN and dense layers were added to the architecture. These layers held more filters and neurons compared to the preexisting layers. A max-pooling layer was added before the new CNN layer because of the large number of features extracted in the previous layers.

Despite the added layers, this decreased the trainable parameters because of the max-pooling layer. This layer gathered the most important features and passed them on to the added CNN layer. The model was then compiled and trained again.

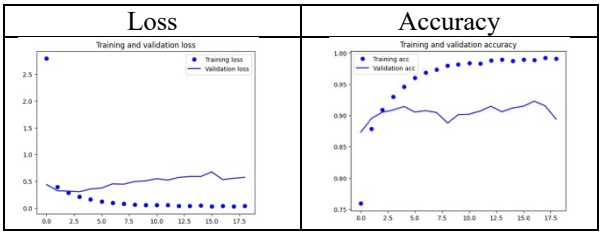


Figure 3. Results from increased complexity

While the model started to overfit almost instantly, the accuracy increased by ~10%, with a new average around 90-92%. To address the overfitting, regularization was implemented into the same model. A dropout layer was added after flattening the weights and L2 regularization was added to the dense layers. Once regularization was added, the model was compiled and trained once more.

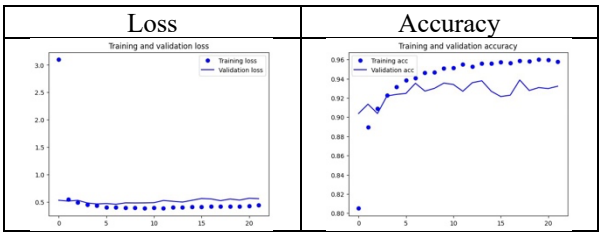


Figure 4. Results from regularization

Adding regularization improved the overall generalization between the training and validation curves. The training also ran for more epoch. This is because of the shrinking of weight dependencies from L2 regularization. The accuracy slightly increased from the previous model with an average around 92-93%. This was due to more epochs during training.

To achieve better generalization and to increase the overall accuracy of the model, the complexity was increased again, and more regularization was applied. This would serve as the most optimal architecture of this experiment. A series of CNN layers were added after the preexisting ones. An additional dense layer was also added. L1 and L2 were both used on the dense layers. The architecture of this more complex model is shown in the following figure:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
conv2d_1 (Conv2D)	(None, 60, 60, 64)	18,496
conv2d_2 (Conv2D)	(None, 58, 58, 128)	73,856
max_pooling2d (MaxPooling2D)	(None, 29, 29, 128)	0
conv2d_3 (Conv2D)	(None, 27, 27, 32)	36,896
conv2d_4 (Conv2D)	(None, 25, 25, 64)	18,496
conv2d_5 (Conv2D)	(None, 23, 23, 128)	73,856
max_pooling2d_1 (MaxPooling2D)	(None, 11, 11, 128)	0
flatten (Flatten)	(None, 15488)	0
dropout (Dropout)	(None, 15488)	0
dense (Dense)	(None, 128)	1,982,592
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 8)	520

Figure 5. Final structure w/ complexity

The added CNN layers allowed for more features to be extracted from the images and learned by the model. Once compiled, the model was trained on the image dataset.

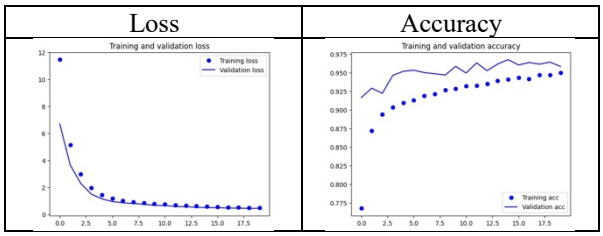


Figure 6. Results of complex structure

This model yielded the best generalization in this experiment. Validation accuracy also increased with an average around 95-96% accuracy.

The last experiment to improve the generalization was data augmentation. Generators were constructed that augmented the existing dataset to provide more data to train on. This increased the total running time of the training process with minimal improvement to generalization.

C. RGB Data

These series of experiments dealt with the exploration of RGB data and different model architectures. Like before, a base model was

constructed as a foundational network to build off. This network included 3 CNN layers, a max-pooling layers, and 2 dense layers for categorical classification. The filter size of the CNN layers was 3x3 and the activation used was rectified linear unit. Once this model was compiled, it was trained.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 62, 62, 32)	896
conv2d_4 (Conv2D)	(None, 60, 60, 64)	18,496
conv2d_5 (Conv2D)	(None, 58, 58, 128)	73,856
max_pooling2d (MaxPooling2D)	(None, 29, 29, 128)	0
flatten_1 (Flatten)	(None, 107648)	0
dense_2 (Dense)	(None, 64)	6,889,536
dense_3 (Dense)	(None, 8)	520

Figure 7. Base architecture for RGB model

The history of the training process was then plotted to analyze the generalization of the training and validation curves.

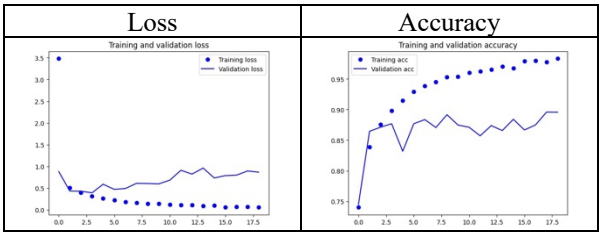


Figure 8. Plots of base model training

The base model performance was mediocre and started to overfit around 2 epoch. The validation accuracy remained around 87% for the duration of the training.

To combat the overfitting, regularization was applied to the base architecture. L1 and L2 regularization were tested separately and resulted in more generalized fitting through the training process for both types. The model was trained, and the history was plotted.

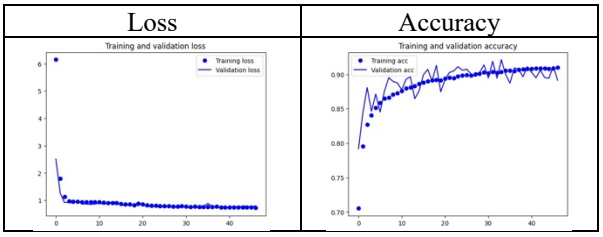


Figure 9. Results of L1 regularization

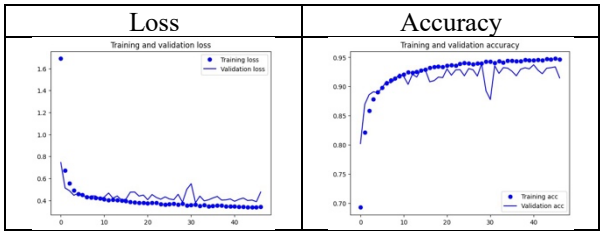


Figure 10. Results of L2 regularization

Regularization greatly improved the generalization of the model. This allowed for the training to run more epoch due to the constant improvement of the validation loss. The more epochs increased the accuracy of the model during both experiments with accuracies reaching 90-92% for both L1 and L2 regularization.

The next experiment focused on increasing the overall accuracy while also maintaining the generalization previously generated. Like in the IR experiment, an extra series of CNN layers were added along with an additional dense layer. The architecture is shown below.

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 62, 62, 32)	896
conv2d_7 (Conv2D)	(None, 60, 60, 64)	18,496
conv2d_8 (Conv2D)	(None, 58, 58, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 29, 29, 128)	0
conv2d_9 (Conv2D)	(None, 27, 27, 32)	36,896
conv2d_10 (Conv2D)	(None, 25, 25, 64)	18,496
conv2d_11 (Conv2D)	(None, 23, 23, 128)	73,856
max_pooling2d_3 (MaxPooling2D)	(None, 11, 11, 128)	0
flatten_1 (Flatten)	(None, 15488)	0
dropout_1 (Dropout)	(None, 15488)	0
dense_4 (Dense)	(None, 256)	3,965,184
dense_5 (Dense)	(None, 128)	32,896
dense_6 (Dense)	(None, 64)	8,256
dense_7 (Dense)	(None, 8)	520

Figure 11. Architecture of final RGB model

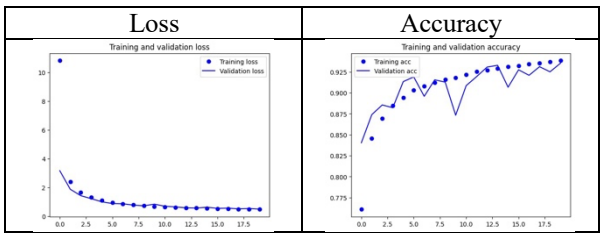


Figure 12. Training and validation loss/acc

Training this model at 20 epoch provided the best generalization and accuracy combination. The model stayed generalized while maintaining higher accuracy peaking around 92.5%.

Like in the IR experiment, the final test was to use data augmentation for more training data. This however, provided little to no improvement to the overall generalization and accuracy of the model. The training time also because extensive when using data augmentation.

D. Results

Increasing overall complexity of each model while also incorporating regularization provide better integrity of the model's ability to predict classes. Below are the confusion matrices of each of the best models from the previously described experiments.

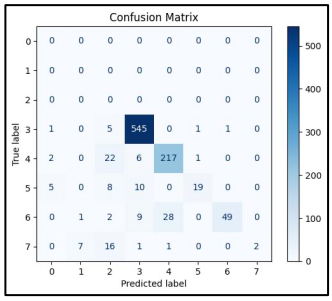


Figure 13. Confusion matrix of IR CNN

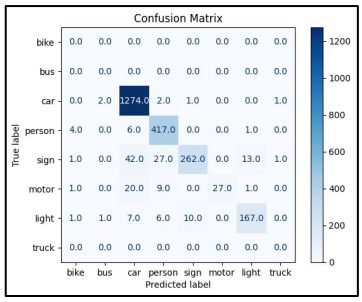


Figure 14. Confusion matrix of RGB CNN

Each model was predicted patches in an accurate fashion. As displayed in the matrices, the diagonals are the most prominent. The testing data was split from the validation set which resulted in small amounts of testing data. Allocating more testing data would improve the overall structure of the confusion matrix for each model.

III. VGG16

A. Outline

For this experiment, Kera's pretrained CNN model, VGG16 was implemented into the network architecture. Different strategies like freezing layers/blocks, regularization, and data augmentation were applied to increase accuracy and improve generalization.

B. Infrared Data

The first set of experiments started with the infrared dataset. Like before, this dataset included grayscale images which were loaded into the program to train the models. To begin, a base model was constructed to be built off of later down the line. This base model contained a convolutional base which included the pretrained model along with 2 dense layers for classification.

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 2, 2, 512)	14,714,688
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 64)	131,136
dense_1 (Dense)	(None, 8)	520

Figure 15. Architecture of VGG16 IR model

The convolutional base itself held many trainable parameters. This is because of the structure of the pretrained model. It holds several different blocks that contain multiple CNN layers. These blocks are then connected utilizing max-pooling.

After the model was compiled, it was then trained. The history of the training was then plotted to once again analyze the generalization of the model.

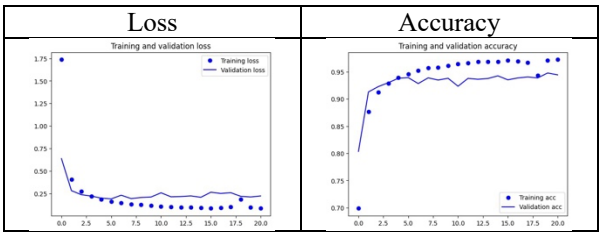


Figure 16. Training and validation loss/acc

Immediately, this base model performed the same as the most complex model from the CNN from scratch experiment. The generalization needed more improvement, and the accuracy still had room to increase.

The goal of the next experiment was to freeze blocks of the pertained model. This would allow for the model to preserve the pretrained weights for the later stages of training. For this experiment, the first 3 blocks of the model were frozen, and the model was trained. Below are the results of the model's training.

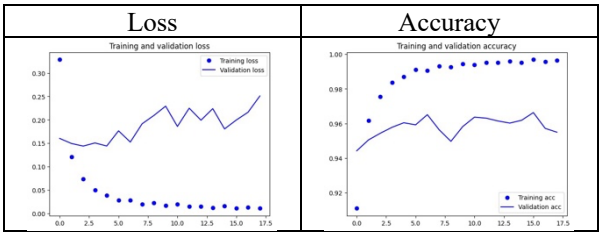
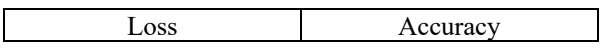


Figure 17. Training and validation loss/acc

This introduced greater accuracies but started overfitting instantly. This was due to the large number of layers that were frozen initially. The weights were not able to be updated during backpropagation which resulted in poor validation performance. Addressing the overfitting, regularization was applied to the model.

For the next experiments, regularization was applied to the model. L1 and L2 were tested separately while also including dropout. The results are shown below.



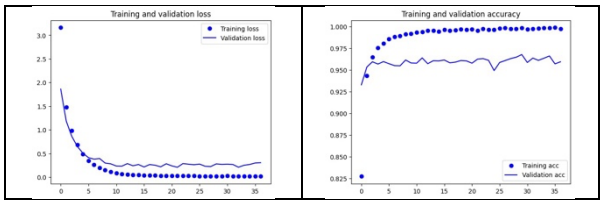


Figure 18. L1 regularization applied

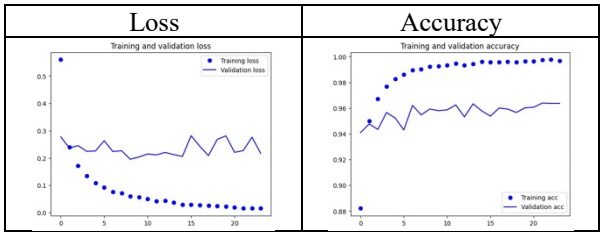


Figure 19. L2 regularization applied

L1 regularization outperformed L2 when it came to generalization. This is because in L1, weight dependencies are fully disregarded if they become too large. L2 however, only shrinks the weights. The higher accuracy was maintained averaging around 95-96%.

For the final experiment, the overall complexity was increased again. This pertained to the dense layers as both the height and width of the densely connected layers were increased. The number of frozen blocks decreased to only the first block of the CNN base. L1 and L2 regularization were also used for the dense layers. Once compiled, the model was trained.

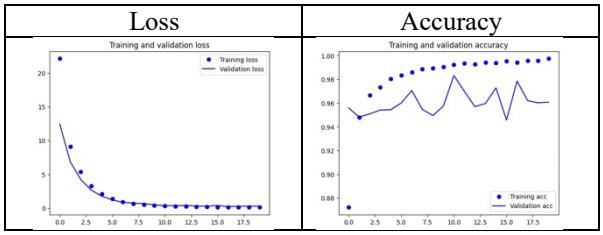


Figure 20. Results from complexity w/ VGG16

The results were the best achieved for the IR dataset. The model was generalized throughout the entire training process while also reaching accuracies of 96-98%. This model was used for further testing to experiment on what kind of images worked best.

The last step was data augmentation. Like before, this provided improved the generalization slightly. Providing more data on the smaller dataset allowed for the model to see more of those objects which tightened the spread on the confusion matrix.

C. RGB Data

The following set of experiments followed the same structure as the previous. RGB data was used instead of the IR. First a base model was constructed to build off. This model held similar characteristics as the previous model with the pretrained convolutional base and 2 dense layers for classification. Despite have more dimensions from the IR data, the base model still performed well when it came to generalization and accuracy.

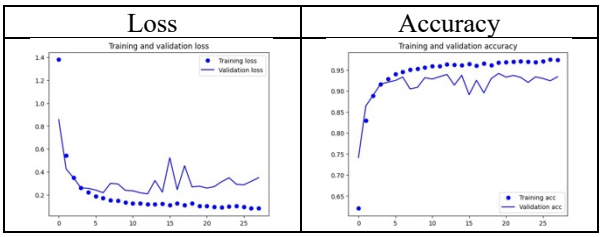


Figure 21. RGB base VGG16 model

The model started generalized but started overfitting around 6 epoch. The accuracy was still high for the base model averaging around 92%.

The next experiment was to freeze blocks of the convolutional base like before. The first 3 blocks were frozen, and the model was trained again. The results were like the IR model when the layers were frozen. The model showed great overfitting earlier on in the training. Regularization was then used to reduce the overfitting.

L1 and L2 regularization was used again and tested separately to show the difference in impact on the model. The results are shown in the following plots.

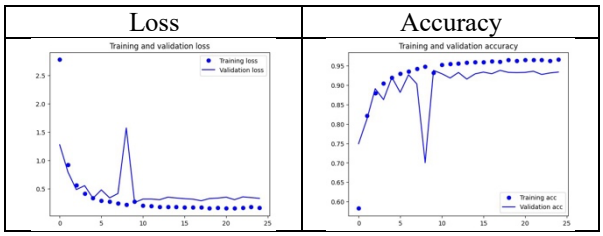


Figure 22. L1 regularization

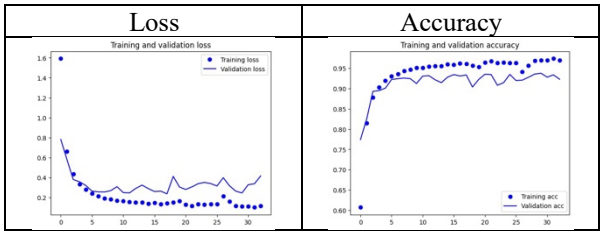


Figure 23. L2 Regularization

A dropout layer was also incorporated between the CNN and dense layers. As the results show, L1 and L2 improved the generalization similarly. The accuracy remained the same even though the training involved more epochs.

The last modification to the model was to introduce more complexity as before. For this an additional dense layer with more neurons were added to the network. The number of frozen blocks decreased again from 3 blocks to only the first block. This model was compiled and trained again.

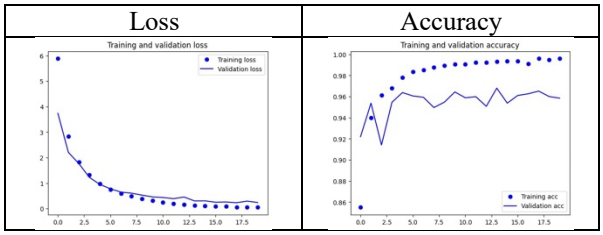


Figure 24. Results of complex VGG16 model

This model provided the best generalization along with the highest accuracy when compared to all previous experiments. The validation accuracy achieved averages around 96%. This model was also used to perform

more test to demonstrate what data worked best with the predictions.

D. Results

Kera's pretrained VGG16 model improved the overall generalization and accuracy for both IR and RGB models. The accuracy in both models were the highest when compared to the CNNs from scratch. This was because of the more complex architecture of the pretrained model. Freezing the pretrained weights allowed for the low-level features to persist during the later training stages. Below are the confusion matrices for the IR and RGB models.

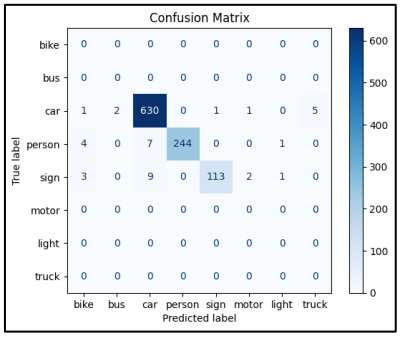


Figure 25. Confusion matrix:IR VGG16

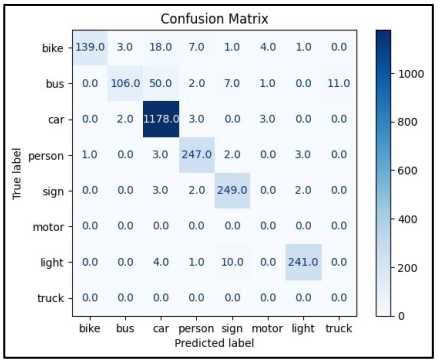


Figure 26. Confusion matrix:RGB VGG16

These matrices were tested using data split from the validation dataset provided. Because the IR dataset was much smaller, there were samples that could not be tested. However, the RGB dataset was much larger which is why the confusion matrix is much more prominent.

IV. AUTO ENCODING

A. Outline

The next set of experiments involved the utilization of auto encoding with a dense network classifier. An auto encoding model first had to be initialized and trained. The encoder was then extracted from the model and paired with densely connected layers. This was completed for both IR and RGB datasets.

B. Experiments w/ IR and RGB Data

The first experiment was conducted with the infrared data. An auto encoding model was initialized with both an encoder and decoder. The architecture is shown below:

Layer (type)	Output Shape	Param #
input_layer_7 (InputLayer)	(None, 64, 64, 1)	0
conv2d_17 (Conv2D)	(None, 64, 64, 16)	160
conv2d_18 (Conv2D)	(None, 64, 64, 8)	1,160
max_pooling2d_3 (MaxPooling2D)	(None, 32, 32, 8)	0
conv2d_19 (Conv2D)	(None, 32, 32, 8)	584
max_pooling2d_4 (MaxPooling2D)	(None, 16, 16, 8)	0
conv2d_21 (Conv2D)	(None, 16, 16, 8)	584
conv2d_22 (Conv2D)	(None, 16, 16, 16)	1,168
up_sampling2d_4 (UpSampling2D)	(None, 32, 32, 16)	0
conv2d_23 (Conv2D)	(None, 32, 32, 1)	145

Figure 27. Auto encoder architecture

This architecture held symmetric characteristics for the encoding and decoding aspects. Once constructed, this was then trained on the training dataset, and images were reconstructed using the auto encoder.

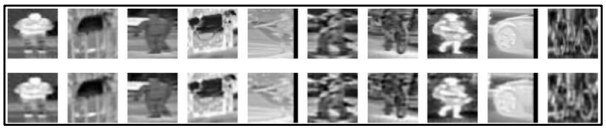


Figure 28. Image reconstruction w/ IR

Figure 29. Image reconstruction w/ RGB

After defining an encoder for the model, it was then extracted and paired with a densely

connected classifier. The weights of the encoder were frozen, and a training dataset was generated using the encoder. This dataset was then used to train the dense network for categorical classification. Once the models were trained with the encoder, the plots of the history were generated.

C. Results

Both models for IR and RGB performed poorly even when increasing complexity and adding regularization. Because of this, augmentation was not used for experiments. The plots of the training process are shown below.

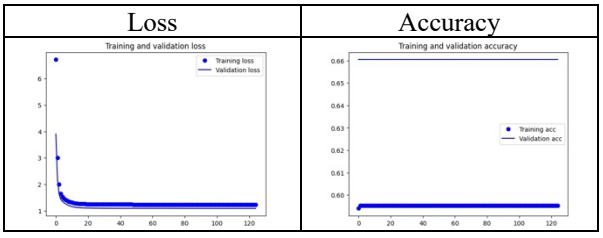


Figure 30. IR: Training and val. results

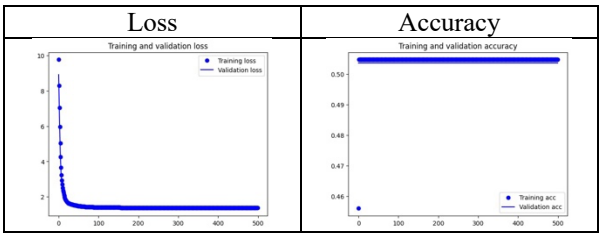


Figure 31. RGB: Training and val. Results

While the generalization of both models performed well, the accuracy was quite low. The IR model averaged 59% validation accuracy while the RGB model averaged 51%. Increasing complexity to both the encoder and densely connected classifier showed no improvement to the model's performance. Due to the poor performance, augmentation was not tested on this model.

V. BEST ARCHITECTURES

A. Outline

The best model for both the IR and RGB datasets were tracked to apply additional testing to. These are models that demonstrated the best generalization and while also maintaining the highest accuracies.

B. IR Data

The best model that obtained the most performance for the infrared dataset was the pretrained VGG16 complex model. This model held the best generalization with accuracies averaging around 96-98%. After additional testing, images that were misclassified were gathered and displayed.

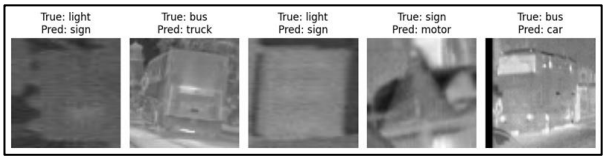


Figure 32. Misclassification of IR data

The data samples that were misclassified are related classes (i.e. busses, trucks, and cars have similar features). Other images that were misclassified have objects of similar shape between classes which causes ambiguity when predicting the class of an object. This is shown in the first image where the light was misclassified as a sign. Since there is no color values correlated to the light signal, they have similar shapes to the signs in the training dataset. The fourth image shows a sign, but the model predicts that it is a motorcycle. Behind the sign are rounded shapes that the model picked up to be motorcycle wheels. Overall, the misclassifications are between classes that are related or due to the image having low qualities of the objects.

C. RGB Data

The best model for RGB data also utilized the pretrained VGG16 model. This too held the best generalization with the highest accuracy, averaging 96%. This model performed more testing on data to predict classes on objects.

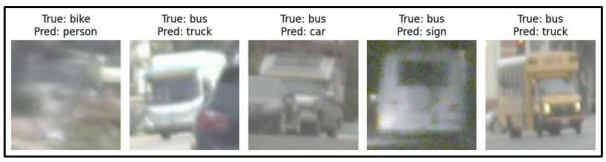


Figure 33. Misclassification of RGB data

The misclassifications follow the same pattern as the IR model predictions. The model struggles to differentiate between cars, trucks and buses. This is because of the similar features that they have (windshields, headlights, taillights, wheels, etc.). The first image shows a person on a bike which is why there is a misclassification in this image. The fourth shows a straight shot of a silver bus which was also confused with a sign. This could be from the shape and color correlated with both, the bus and the signs the model was trained on.