VehID – Milestone Evaluation

Members: Remington Greko, Spencer Hirsch, Thomas Johnson, and Alexis Nagle

Faculty Advisor: Dr. Silaghi

Client: Clayton Levins

Milestone 3

Milestone 3 Progress Matrix

Task	Completion %	Remington	Spencer	Thomas	Alexis	To-do
Sprint Planning	100%	25%	25%	25%	25%	NA
Data preprocessing	100%	20%	20%	30%	30%	NA
Split Dataset	100%	20%	30%	20%	30%	NA
Create Body Type Model	100%	25%	25%	25%	25%	NA
Hyper-parameter tuning	100%	30%	30%	20%	20%	NA
Milestone Evaluation	100%	25%	25%	25%	25%	NA

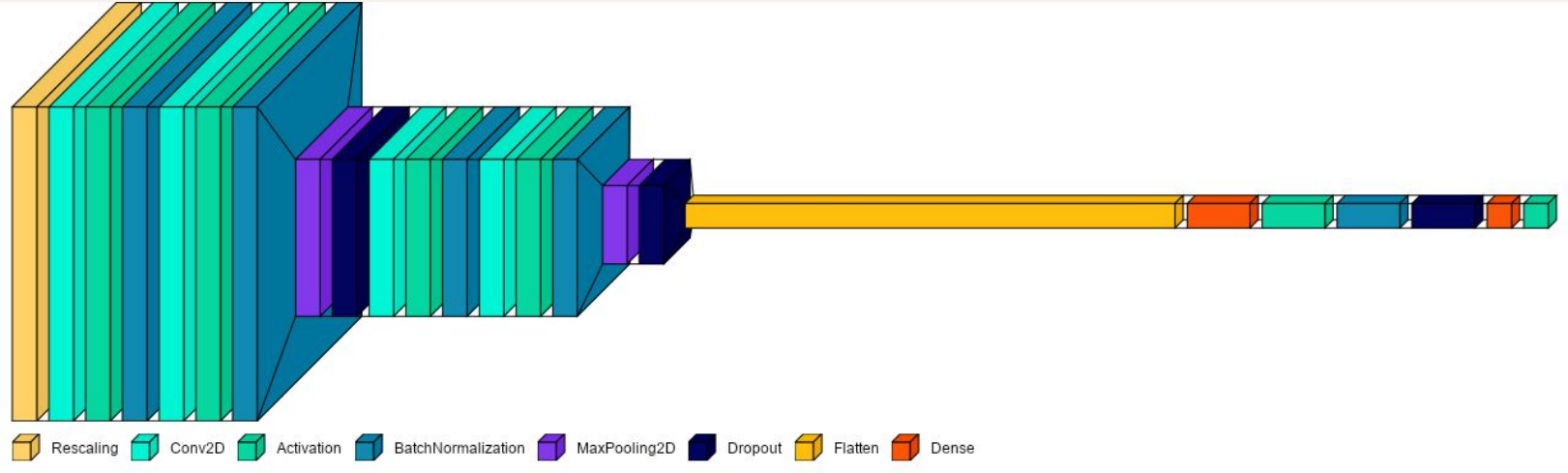
Tasks - Data Preparation

- Data Preprocessing
  - Make/Model
    - Grey-scaled images because the classes were biased by color
      - All Hummers were yellow
      - All Cadillacs were red
    - Scaled images to a uniform 64x64
  - Body Type
    - 1st dataset found had many issues
      - Confusing categories
      - Images of small sections of a car
        - Just the door, grill, etc
    - Scaled images to a uniform 64x64
- Split Dataset
  - Make/Model: Continued with pre-divided split ~50/25/25
  - Body Type: Continued with pre-divided split ~70/20/10

Tasks - Color Recognition CNN Creation

- Make/Model
  - Tested various architectures with no improvement
    - Single Layer, MiniVGG, AlexNet, ResNet, Inception Net
  - Client advised we switch gears
- Body Type
  - Tested multiple architectures with various optimizers
    - Single Layer Benchmark - 50% Accuracy
    - Mini VGGNet
      - With Adam Optimizer - 75% Accuracy
      - With SGD Optimizer - 72% Accuracy
    - AlexNet
      - With Adam Optimizer - 53% Accuracy
      - Chose not to test SGD with poor results of Adam

Model Visualization



- 4 Convolutional Layers - Filtering and feature extraction
- 6 Activation Layers - Feature processing
- 2 Dense Layers - Neuron connectivity and pattern recognition
- 2 Max Pooling Layers - Feature Selection and downsampling

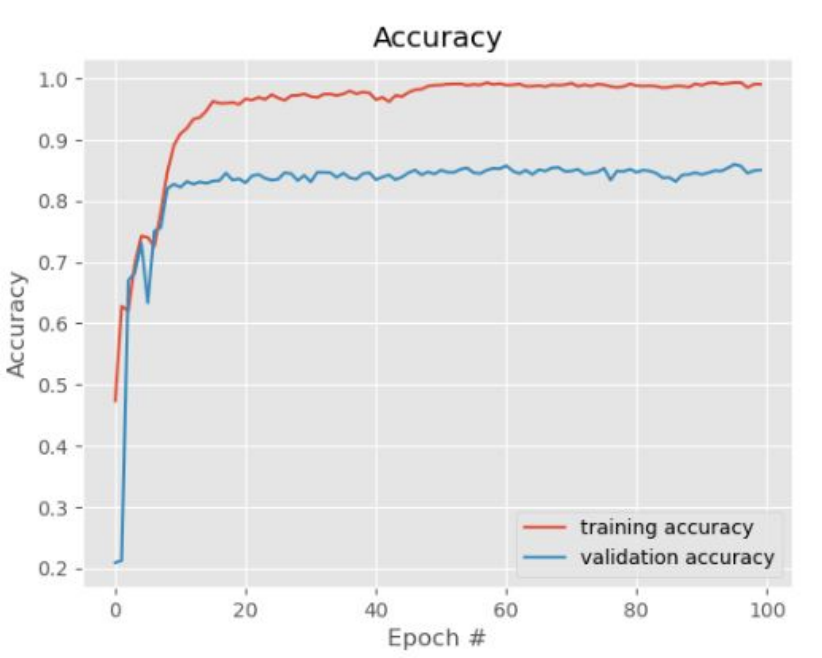
Hyper-parameter Tuning

- Hyper-parameter tuning on Mini VGGNet w/ SGD
  - Learning Rate (LR)
  - L1 & L2 Regularization
    - Used to prevent overfitting model
      - L1 (Lasso): Useful for feature selection
      - L2 (Ridge): Distributing influence of features in the model
  - Random Weight Initialization
  - Early Stopping and Reduce LR on Plateau
  - Restore Best Weights

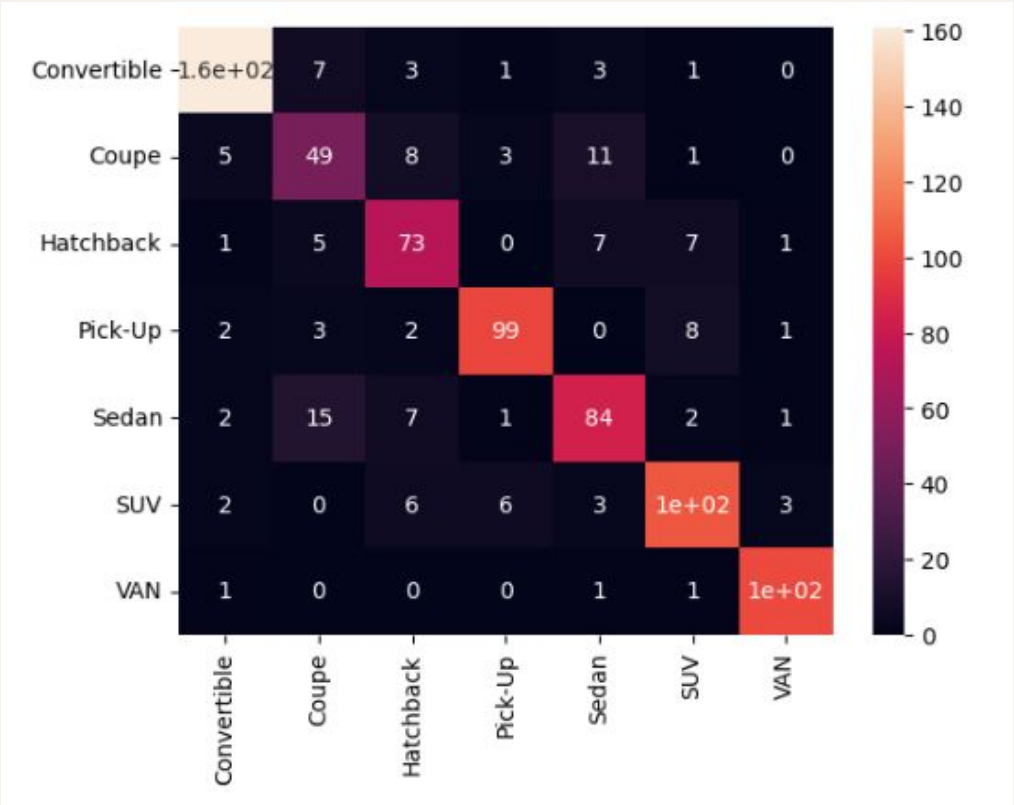
Accuracy Results

```
26/26 [==============================] - 2s 72ms/step
```

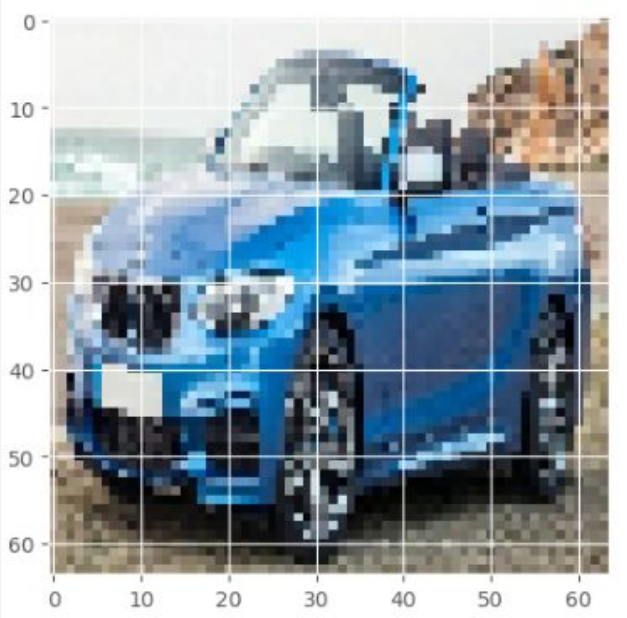
	precision	recall	f1-score	support
Convertible	0.93	0.91	0.92	176
Coupe	0.62	0.64	0.63	77
Hatchback	0.74	0.78	0.76	94
Pick-Up	0.90	0.86	0.88	115
Sedan	0.77	0.75	0.76	112
SUV	0.84	0.84	0.84	125
VAN	0.94	0.97	0.96	103
accuracy			0.84	802
macro avg	0.82	0.82	0.82	802
weighted avg	0.84	0.84	0.84	802



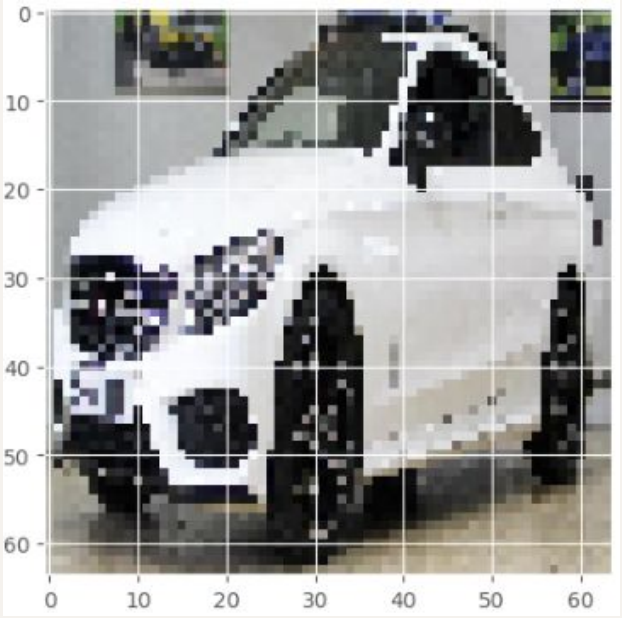
Model Confusion Matrix



CNN Example Predictions



The model is 99.42% confident that the image is a Convertible vehicle
The correct color of this car is Convertible



The model is 87.88% confident that the image is a Sedan vehicle
The correct color of this car is Coupe

Video Demo

MiniVGG w/ Random Weight initialization L1&L2 Regularizer RLR on Plateau, Restore Best Weights

```
[6]: class MiniVGGNet_IncrDropout:
         def build(height, width, depth, classes, l1, l2):
             model = Sequential(name = 'MiniVGGNet')
             model.add(Rescaling(1./255))

             model.add(Conv2D(32, (3, 3), padding = 'same', kernel_initializer=RandomNormal(stddev=0.01),
                              kernel_regularizer=L1L2(l1=l1, l2=l2), input_shape = (height, width, depth)))
             model.add(Activation('relu'))
             model.add(BatchNormalization())

             model.add(Conv2D(32, (3, 3), padding = 'same', kernel_initializer=RandomNormal(stddev=0.01), kernel_regularizer=L1L2(l1=l1, l2=l2)))
             model.add(Activation('relu'))
             model.add(BatchNormalization())

             model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))
             model.add(Dropout(0.5))

             model.add(Conv2D(64, (3, 3), padding = 'same', kernel_initializer=RandomNormal(stddev=0.01), kernel_regularizer=L1L2(l1=l1, l2=l2)))
             model.add(Activation('relu'))
             model.add(BatchNormalization())

             model.add(Conv2D(64, (3, 3), padding = 'same', kernel_initializer=RandomNormal(stddev=0.01), kernel_regularizer=L1L2(l1=l1, l2=l2)))
             model.add(Activation('relu'))
             model.add(BatchNormalization())

             model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))
             model.add(Dropout(0.5))

             model.add(Flatten())
```

Client Feedback - Clayton Levins

- Happy with our division of the project and progress made so far
- Expressed satisfaction with the split of the dataset
- Covered hardships with the make/model CNN
    - Discussed the bias of color and he agreed with our decision to greyscale
    - Advised us into shifting gears to body type recognition
        - This would allow us to obtain the results need to produce a real-world application
        - Weren’t able to present the progress made on this to the client

Advisor Feedback – Dr. Silaghi

- Didn’t have any comments about the organization of our project
  - Sprint Planning
  - Data Preprocessing
  - Splitting Dataset
- Didn’t express any concerns with our shift in the objective of the Milestone
  - Shared our hardships with the Make/Model recognition
  - Mentioned our Client’s support in the shift in Milestone Objective

Milestone 4

Milestone 4 Tasks

- Review and Revise Dataset
- Split Dataset
- Data Preprocessing
- Create Convolutional Neural Network for Make Detection/Recognition
- Hyper-parameter tuning
- Sprint Planning
- Milestone 4 Evaluation

Task Matrix - Milestone 4

Task	Remington	Spencer	Thomas	Alexis
Split Dataset	20%	30%	20%	30%
Create make detection model	25%	25%	25%	25%
Hyper-parameter tuning	30%	30%	20%	20%
Data preprocessing	20%	20%	30%	30%
Sprint Planning	25%	25%	25%	25%
Milestone Evaluation	25%	25%	25%	25%

Questions?