ASSIGNMENT 1

Full Name	UBIT Name	UB Number
Kusha Kumar Dharavath	kushakum	50511275
Dharma. Acha	dharmaac	50532663

Part I: CNN Classification

Summary of the VGG Model:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 64, 64]           1,792
              ReLU-2           [-1, 64, 64, 64]               0
            Conv2d-3           [-1, 64, 64, 64]          36,928
              ReLU-4           [-1, 64, 64, 64]               0
         MaxPool2d-5           [-1, 64, 32, 32]               0
            Conv2d-6          [-1, 128, 32, 32]          73,856
              ReLU-7          [-1, 128, 32, 32]               0
            Conv2d-8          [-1, 128, 32, 32]         147,584
              ReLU-9          [-1, 128, 32, 32]               0
        MaxPool2d-10          [-1, 128, 16, 16]               0
           Conv2d-11          [-1, 256, 16, 16]         295,168
             ReLU-12          [-1, 256, 16, 16]               0
           Conv2d-13          [-1, 256, 16, 16]         590,080
             ReLU-14          [-1, 256, 16, 16]               0
        MaxPool2d-15            [-1, 256, 8, 8]               0
           Conv2d-16            [-1, 512, 8, 8]       1,180,160
             ReLU-17            [-1, 512, 8, 8]               0
           Conv2d-18            [-1, 512, 8, 8]       2,359,808
             ReLU-19            [-1, 512, 8, 8]               0
        MaxPool2d-20            [-1, 512, 4, 4]               0
           Conv2d-21            [-1, 512, 4, 4]       2,359,808
             ReLU-22            [-1, 512, 4, 4]               0
           Conv2d-23            [-1, 512, 4, 4]       2,359,808
             ReLU-24            [-1, 512, 4, 4]               0
        MaxPool2d-25            [-1, 512, 2, 2]               0
           Linear-26                 [-1, 4096]       8,392,704
             ReLU-27                 [-1, 4096]               0
          Dropout-28                 [-1, 4096]               0
```

```
        Linear-29                [-1, 4096]      16,781,312
          ReLU-30                [-1, 4096]               0
       Dropout-31                [-1, 4096]               0
        Linear-32                   [-1, 3]          12,291
================================================================
Total params: 34,591,299
Trainable params: 34,591,299
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 16.39
Params size (MB): 131.96
Estimated Total Size (MB): 148.39
----------------------------------------------------------------
```

Influence of techniques on base model:

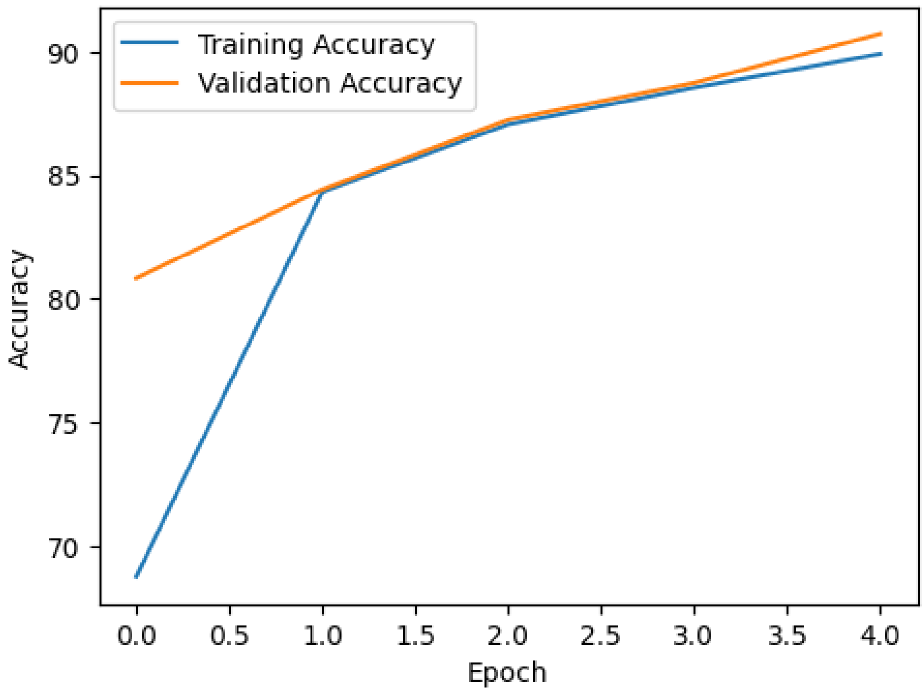
The test accuracy of the base model is 90 % and test loss is 0.250 when we apply the overfitting prevention techniques the test accuracy increased by 3 % i.e 93% and test loss is 0.185

Base VGG model:

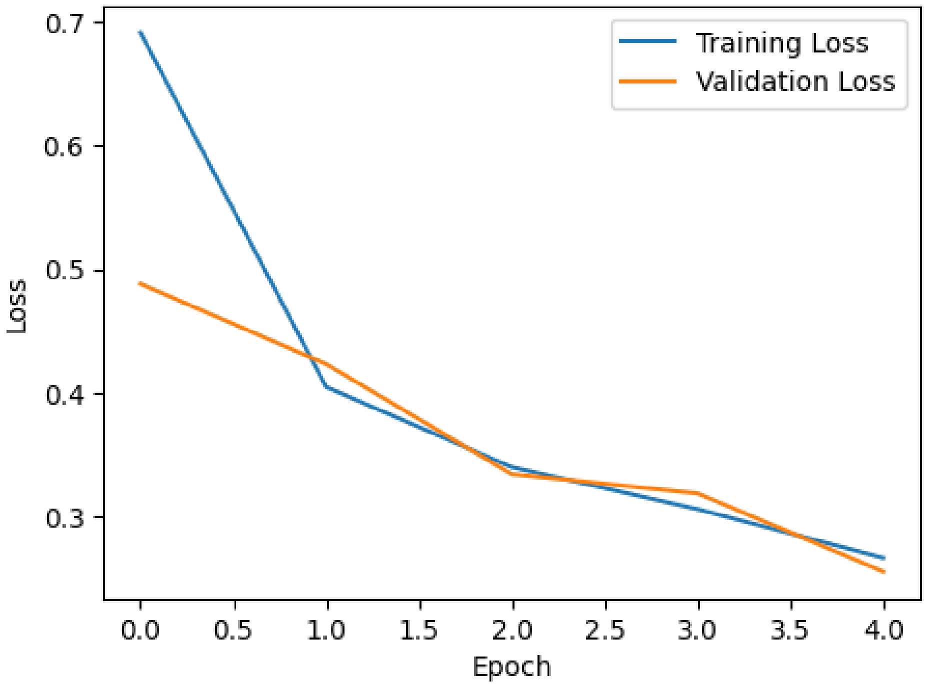
Train accuracy: 89.90%
Train loss: 0.26
Validation accuracy: 90.71%
Validation loss: 0.25
Test accuracy: 90%
Test Loss: 0.25

From the below graphs we can infer that as we increase the number of epochs it is clear that the accuracy of the training and validation increased. The training accuracy rocketed from approximately 68 % to 89 %. On the other hand training loss and validation loss decreased from 0.69 and 0.48 to 0.266 and 0.255 respectively.

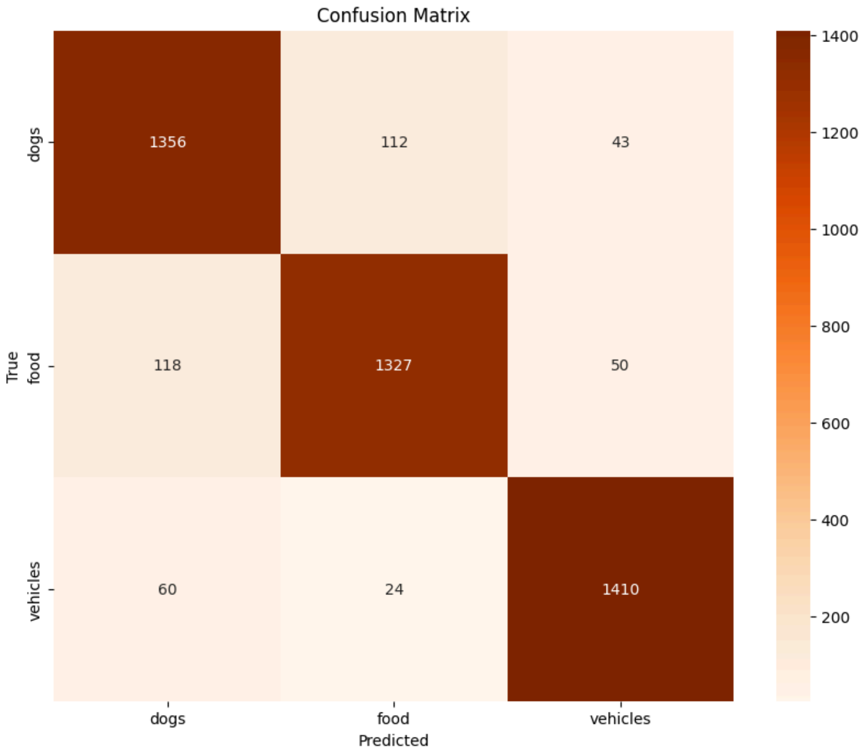
Training and Validation Accuracy:



Training and Validation Loss:



Confusion matrix:



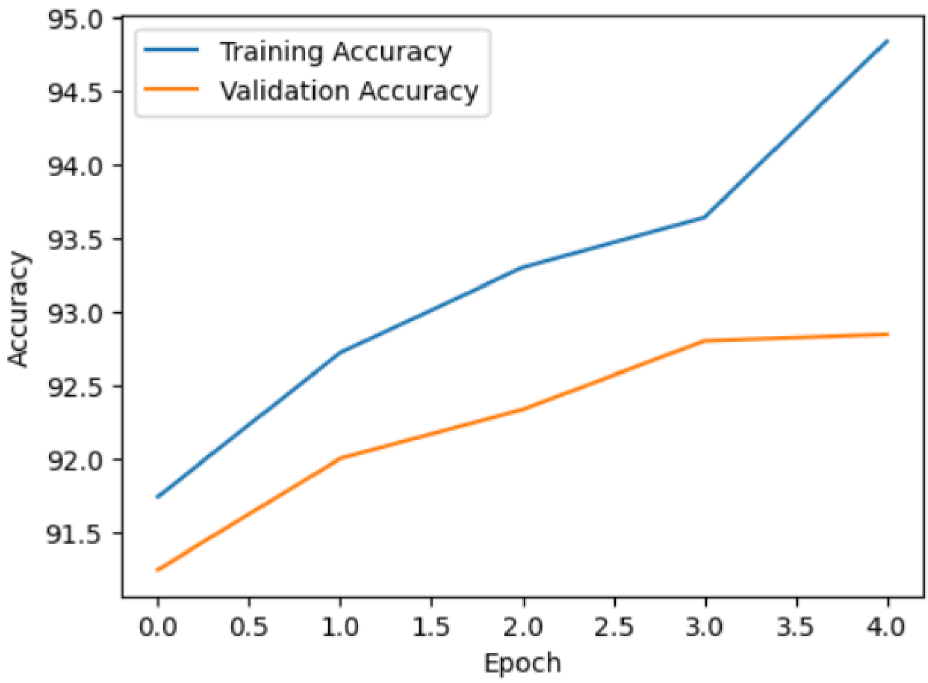
Evaluation metrics:

Precision: 0.91
Recall: 0.91
F1 Score: 0.91

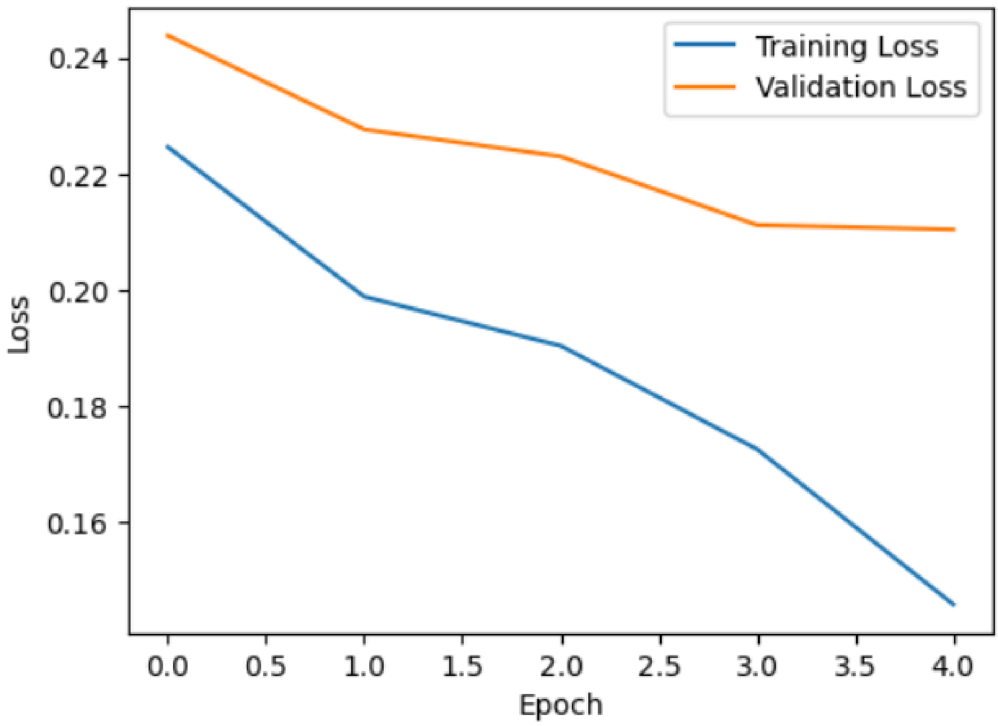
Base + L2 Regularization:
Train accuracy: 94.84%
Train loss: 0.14
Validation accuracy: 92.84%
Validation loss: 0.21
Test accuracy: 92%
Test Loss: 0.21

From the below graphs we can say that as we go on increasing the number of epochs the training accuracy and validation accuracy increased to 94 % and 92 % respectively whereas training loss and validation loss decreased at every epoch.

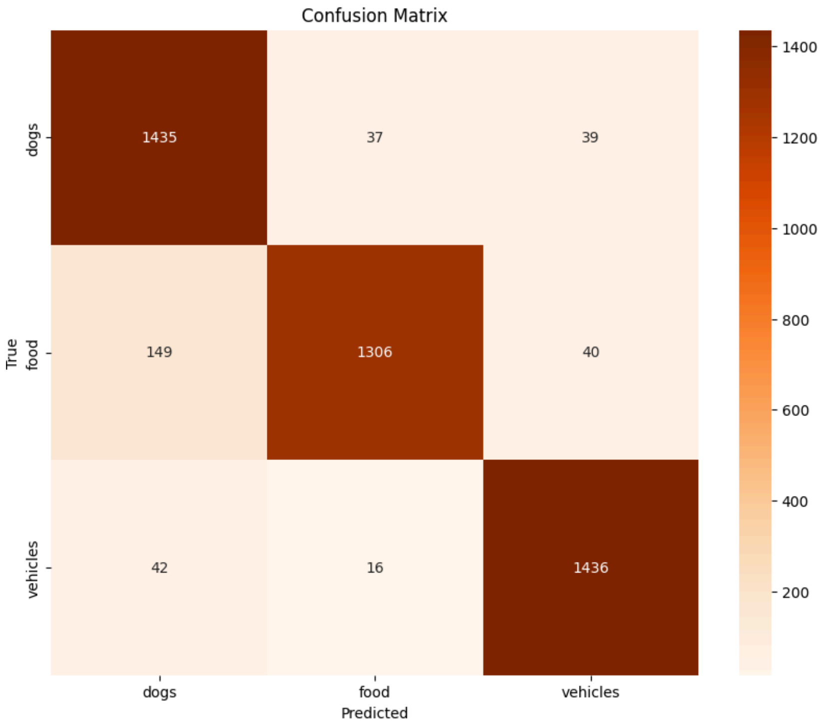
Training and Validation Accuracy:



Training and Validation Loss:



Confusion Matrix:



Evaluation Metrics:

Precision: 0.93
Recall: 0.93
F1 Score: 0.93

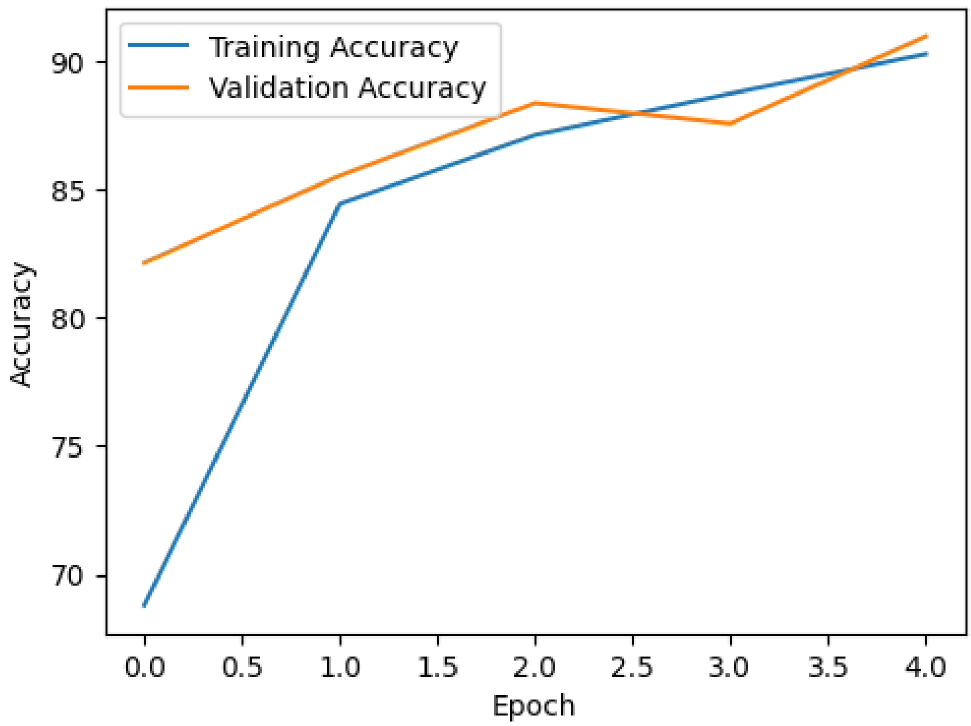
Base + L2 Regularization + DropOut

Train accuracy: 90.26%
Train loss: 0. 26
Validation accuracy: 90.96%
Validation loss: 0.25
Test accuracy: 90%
Test Loss: 0.24

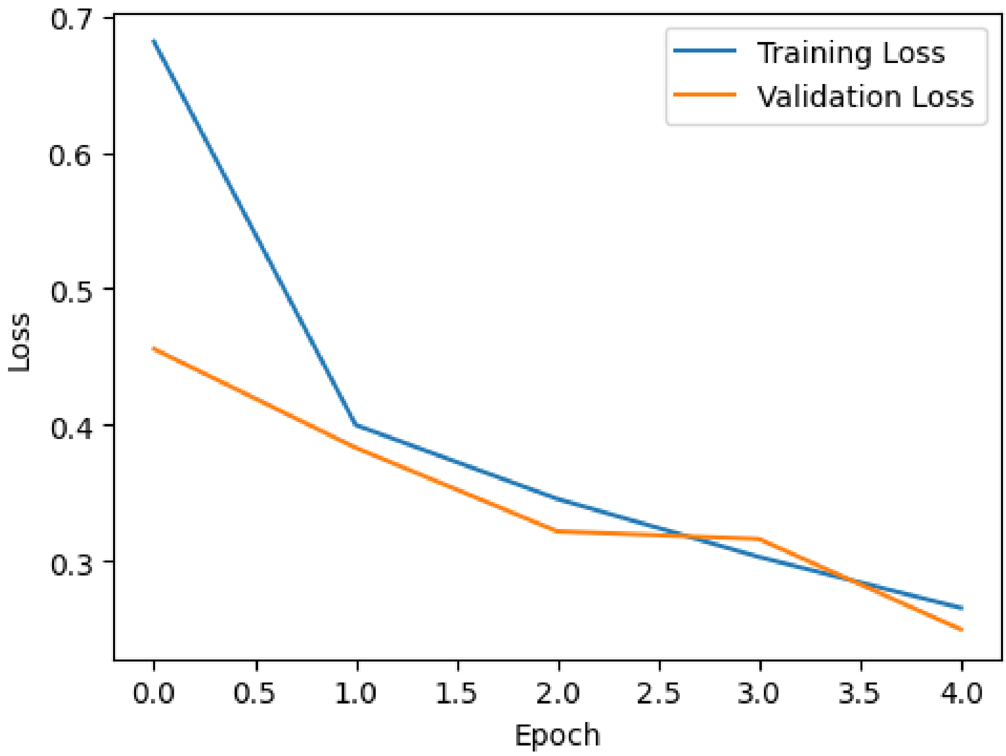
At the 1st Epoch the training accuracy is very low i.e 68% and validation accuracy is 82% after running 5 epochs training accuracy and validation accuracy settled at 90% and 90.96% respectively. See the below graphs for better understanding

When we look into the validation and Training loss the losses decreased.

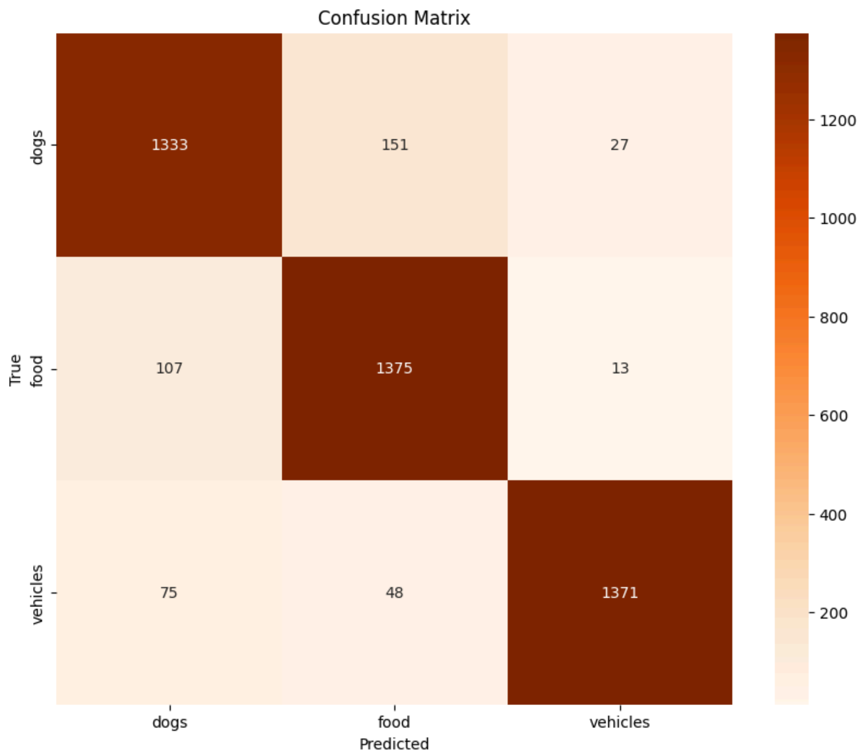
Training and Validation Accuracy:



Training and Validation Loss:



Confusion matrix:



Evaluation Metrics:

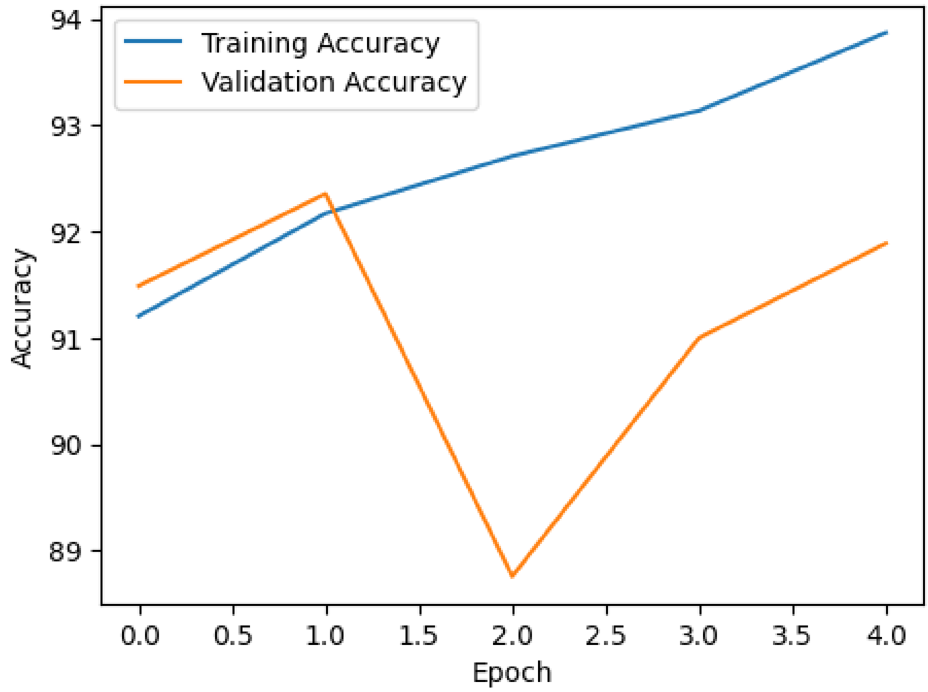
Precision: 0.91
Recall: 0.91
F1 Score: 0.91

Base + L2 Regularization + DropOut + Early Stopping:

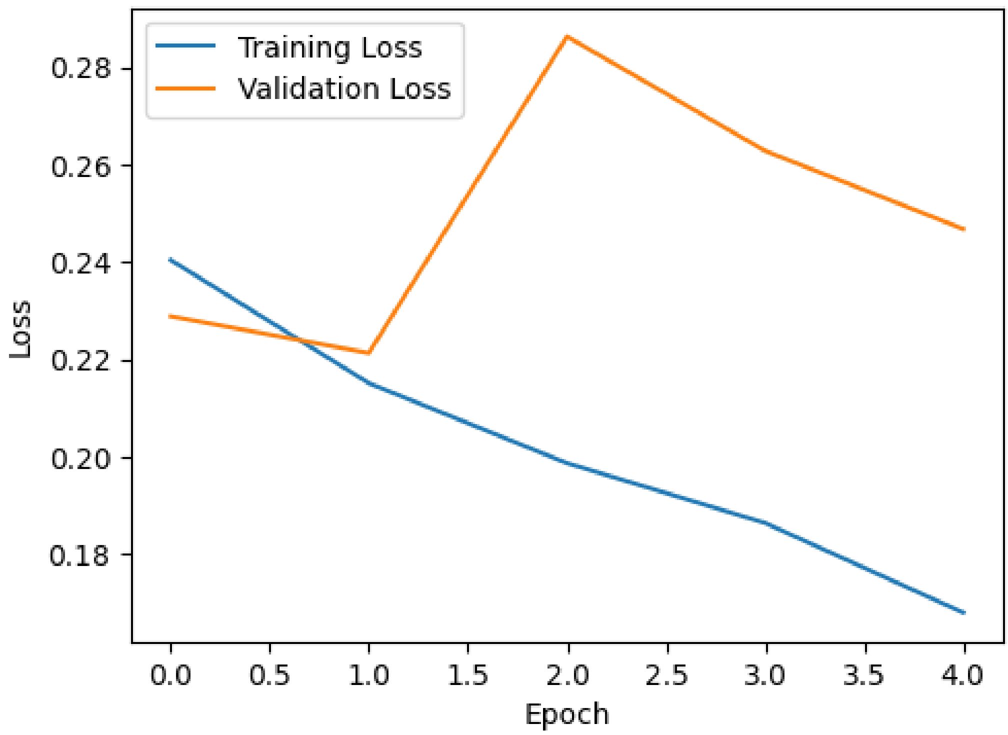
Train accuracy: 93.87%
Train loss: 0. 16
Validation accuracy: 91.89%
Validation loss: 0.24
Test accuracy: 91%
Test Loss: 0.24

The graph depicts the training accuracy increased to 93.87% and validation accuracy is 91.89% and respective losses also decreased as number of epochs increased.

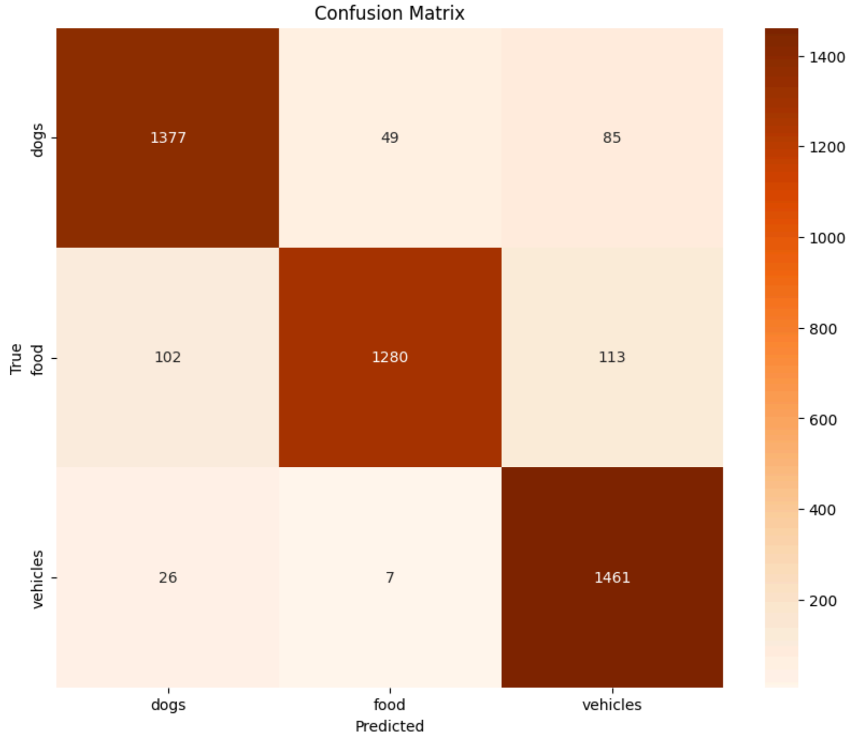
Training and Validation Accuracy:



Training and Validation Loss:



Confusion matrix:



Evaluation Metrics:

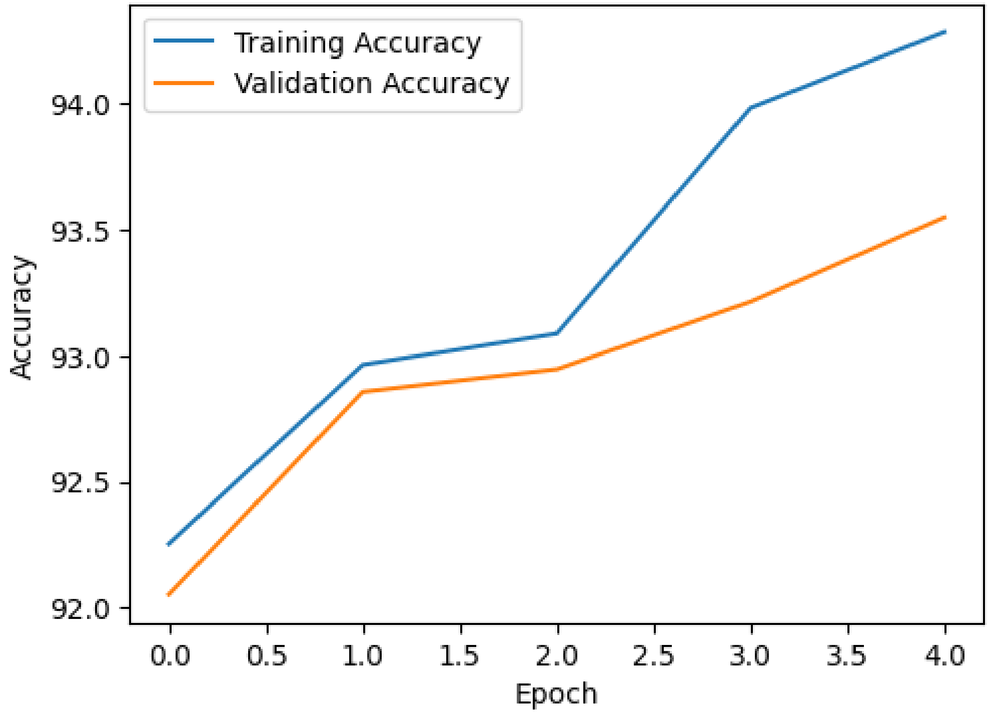
Precision: 0.92
Recall: 0.92
F1 Score: 0.91

Base + L2 Regularization + DropOut +Early Stopping + Image Augmentation:

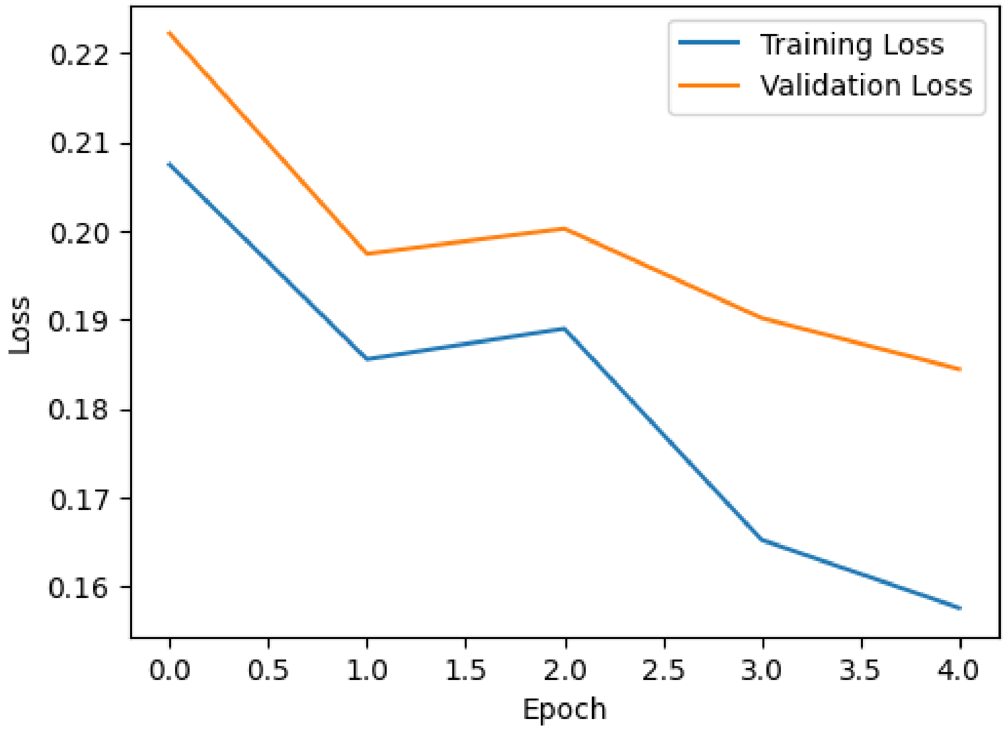
Train accuracy: 94.28%
Train loss: 0. 15
Validation accuracy: 93.55%
Validation loss: 0.18
Test accuracy: 93%
Test Loss: 0.18

From the below graphs we can infer that as we increase the number of epochs it is clear that the accuracy of the training and validation increased. On the other hand training loss and validation loss decreases.

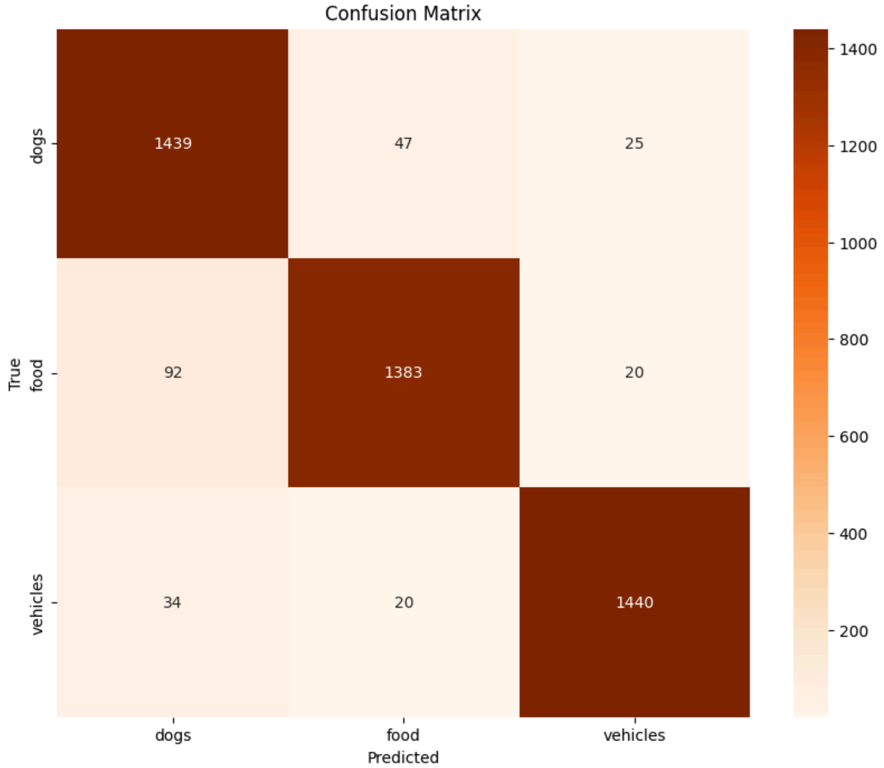
Training and Validation Accuracy:



Training and Validation Loss:



Confusion Matrix:



Evaluation Metrics:

Precision: 0.95
Recall: 0.95
F1 Score: 0.95

Part II: Implementing ResNet Architecture

Resnet

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	9,408
BatchNorm2d-2	[-1, 64, 32, 32]	128
ReLU-3	[-1, 64, 32, 32]	0
MaxPool2d-4	[-1, 64, 16, 16]	0
Conv2d-5	[-1, 64, 16, 16]	36,864
BatchNorm2d-6	[-1, 64, 16, 16]	128
ReLU-7	[-1, 64, 16, 16]	0
Conv2d-8	[-1, 64, 16, 16]	36,864
BatchNorm2d-9	[-1, 64, 16, 16]	128
ReLU-10	[-1, 64, 16, 16]	0

BasicBlock-11	[-1, 64, 16, 16]	0
Conv2d-12	[-1, 64, 16, 16]	36,864
BatchNorm2d-13	[-1, 64, 16, 16]	128
ReLU-14	[-1, 64, 16, 16]	0
Conv2d-15	[-1, 64, 16, 16]	36,864
BatchNorm2d-16	[-1, 64, 16, 16]	128
ReLU-17	[-1, 64, 16, 16]	0
BasicBlock-18	[-1, 64, 16, 16]	0
Conv2d-19	[-1, 128, 8, 8]	73,728
BatchNorm2d-20	[-1, 128, 8, 8]	256
ReLU-21	[-1, 128, 8, 8]	0
Conv2d-22	[-1, 128, 8, 8]	147,456
BatchNorm2d-23	[-1, 128, 8, 8]	256
Conv2d-24	[-1, 128, 8, 8]	8,192
BatchNorm2d-25	[-1, 128, 8, 8]	256
ReLU-26	[-1, 128, 8, 8]	0
BasicBlock-27	[-1, 128, 8, 8]	0
Conv2d-28	[-1, 128, 8, 8]	147,456
BatchNorm2d-29	[-1, 128, 8, 8]	256
ReLU-30	[-1, 128, 8, 8]	0
Conv2d-31	[-1, 128, 8, 8]	147,456
BatchNorm2d-32	[-1, 128, 8, 8]	256
ReLU-33	[-1, 128, 8, 8]	0
BasicBlock-34	[-1, 128, 8, 8]	0
Conv2d-35	[-1, 256, 4, 4]	294,912
BatchNorm2d-36	[-1, 256, 4, 4]	512
ReLU-37	[-1, 256, 4, 4]	0
Conv2d-38	[-1, 256, 4, 4]	589,824
BatchNorm2d-39	[-1, 256, 4, 4]	512
Conv2d-40	[-1, 256, 4, 4]	32,768
BatchNorm2d-41	[-1, 256, 4, 4]	512
ReLU-42	[-1, 256, 4, 4]	0
BasicBlock-43	[-1, 256, 4, 4]	0
Conv2d-44	[-1, 256, 4, 4]	589,824
BatchNorm2d-45	[-1, 256, 4, 4]	512
ReLU-46	[-1, 256, 4, 4]	0
Conv2d-47	[-1, 256, 4, 4]	589,824
BatchNorm2d-48	[-1, 256, 4, 4]	512
ReLU-49	[-1, 256, 4, 4]	0
BasicBlock-50	[-1, 256, 4, 4]	0
Conv2d-51	[-1, 512, 2, 2]	1,179,648
BatchNorm2d-52	[-1, 512, 2, 2]	1,024
ReLU-53	[-1, 512, 2, 2]	0
Conv2d-54	[-1, 512, 2, 2]	2,359,296

```
      BatchNorm2d-55           [-1, 512, 2, 2]           1,024
           Conv2d-56           [-1, 512, 2, 2]         131,072
      BatchNorm2d-57           [-1, 512, 2, 2]           1,024
             ReLU-58           [-1, 512, 2, 2]               0
       BasicBlock-59           [-1, 512, 2, 2]               0
           Conv2d-60           [-1, 512, 2, 2]       2,359,296
      BatchNorm2d-61           [-1, 512, 2, 2]           1,024
             ReLU-62           [-1, 512, 2, 2]               0
           Conv2d-63           [-1, 512, 2, 2]       2,359,296
      BatchNorm2d-64           [-1, 512, 2, 2]           1,024
             ReLU-65           [-1, 512, 2, 2]               0
       BasicBlock-66           [-1, 512, 2, 2]               0
AdaptiveAvgPool2d-67           [-1, 512, 1, 1]               0
           Linear-68                   [-1, 3]           1,539
================================================================
Total params: 11,178,051
Trainable params: 11,178,051
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 5.13
Params size (MB): 42.64
Estimated Total Size (MB): 47.82
----------------------------------------------------------------
```

Train accuracy: 97.21%
Train loss: 0. 08
Validation accuracy: 88.58%
Validation loss: 0.42
Test accuracy: 88%
Test Loss: 0.38

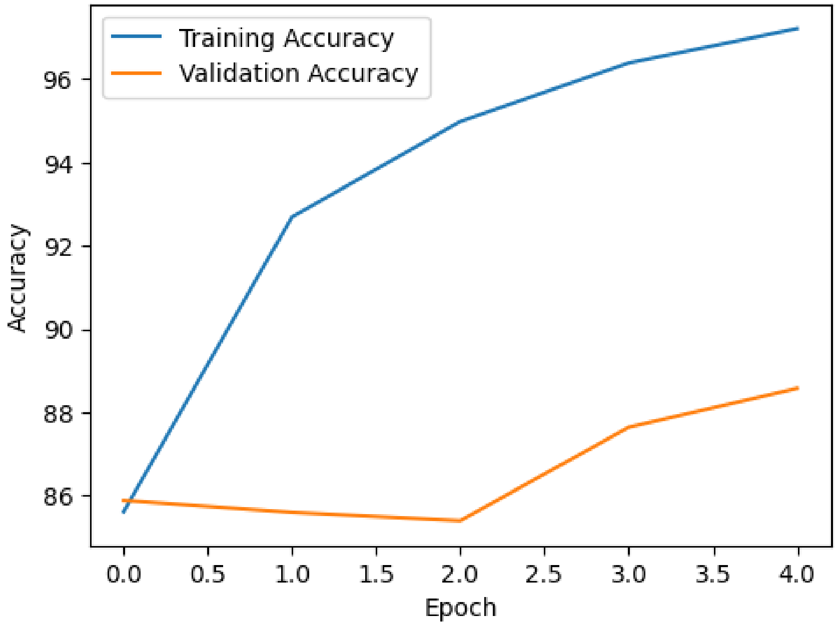
From the below graphs, we can see that there's a consistent improvement in training accuracy, from 85.61% in the first epoch to 97.21% by the fifth epoch. This indicates that the model is effectively learning from the training data over time.

Validation accuracy started at 85.89% in the first epoch and experienced minor fluctuations, eventually improved to 88.58% by the fifth epoch

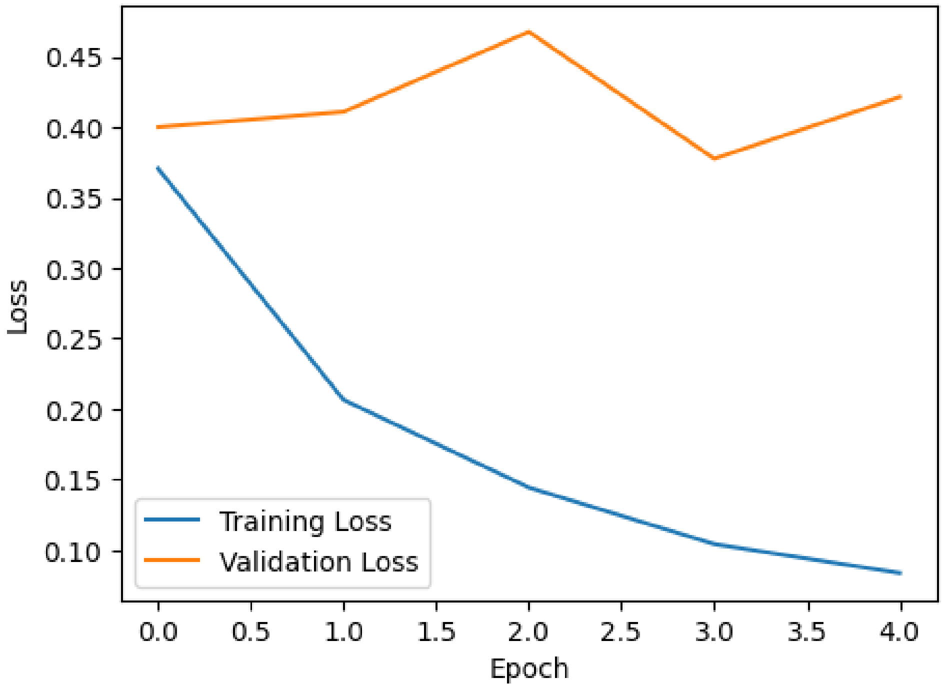
Training loss decreased significantly from 0.37 to 0.08, suggesting that model's increasing proficiency in fitting the training data.

Validation loss, conversely, increases substantially, especially notable in the third epoch where it peaks at 0.46, and slightly reduces at fifth epoch

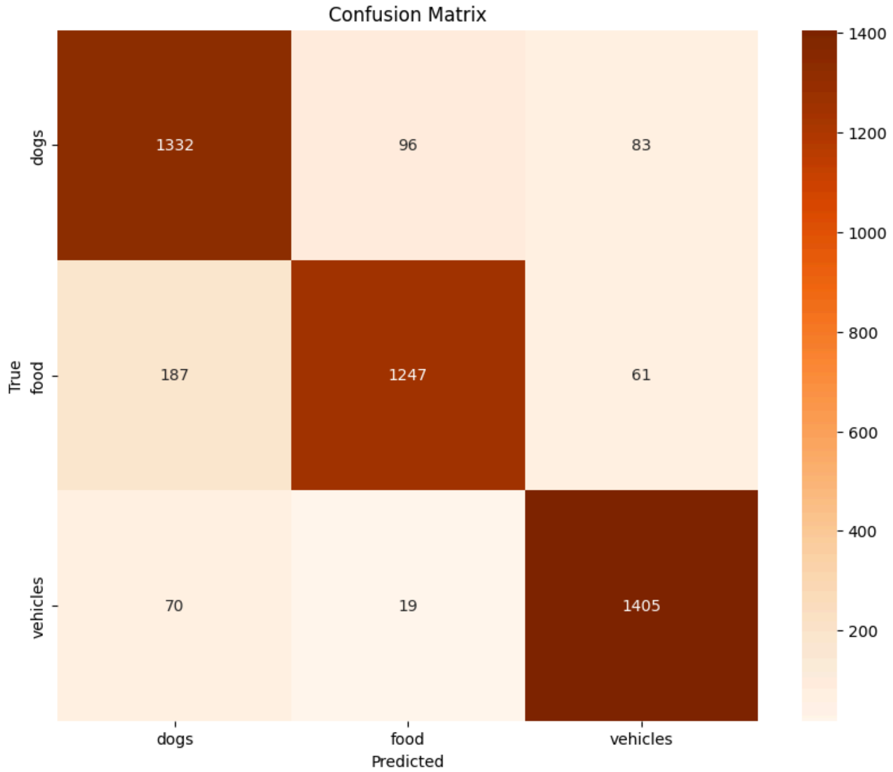
Training and Validation Accuracy:



Training and Validation Loss:



Confusion Matrix:



Evaluation Metrics:

Precision: 0.89
Recall: 0.89
F1 Score: 0.89

Part V.I CNN

1. What is the output size after the first layer?
Given:
The input images are 32x28 RGB - so, W=32, H=28

P=0
F=5
D=10
S=1

Output Width = (W-F+2P)/S +1 = (32-5+0)/1 +1 = 28
Output Height= (H-F+2P)/S +1 = (28-5+0)/1 +1 = 24

Output Size = WxHxD = 28x24x10

2. How many parameters are there in the first layer?
Given:

Filter size = 5x5
No.of filters in the layer = 10
No.of input channels = 3(RGB)
No.of parameters = (5x5x3+1)x10 = 760

3. What would be the output size if a padding of 1 was used instead of zero Padding?
If P=1,
Output Width = (W-F+2P)/S +1 = (32-5+2x1)/1 +1 = 30
Output Height= (H-F+2P)/S +1 = (28-5+2x1)/1 +1 = 26

Output Size = WxHxD = 30x26x10

4. What would be the number of parameters if the input images were grayscale instead of RGB?
If input images were grayscale, then no.of input channels = 1
Filter size = 5x5
No.of filters in the layer = 10
No.of input channels = 1
No.of parameters = (5x5x1+1)x10 = 260

5. Considering the above specific task and the requirement of probabilistic outputs, which activation function would be most suitable for the output layer in this scenario?
Softmax Function

6. Prove that the activation function used here is invariant to constant shifts in the input values, meaning that adding a constant value to all the input values will not change the resulting probabilities.

6. The activation function used is softmax.

Let's prove that softmax activation function is invariant to constant shifts in the input values

Softmax function for neural network A, with components $a_1, a_2, \dots a_n$ for a classification problem with n classes, for i^{th} component is

$$\text{softmax}(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{n} e^{a_j}}$$

Let's say a constant c is added to each input value, then new input vector is $A' = A + c$ (where $A' = a_1+c, a_2+c, \dots + a_n+c$)

Softmax function applied to an element $a_i + c$ of A' is

$$\text{softmax}(a_i+c) = \frac{e^{a_i+c}}{\sum_{j=1}^{n} e^{a_j+c}}$$

$$\Rightarrow \frac{e^{c} \cdot e^{a_i}}{\sum_{j=1}^{n} e^{c} \cdot e^{a_j}}$$

$$\Rightarrow \frac{\cancel{e^{c}} \cdot e^{a_i}}{\cancel{e^{c}} \cdot \sum_{j=1}^{n} e^{a_j}}$$

$$= \frac{e^{a_i}}{\sum_{j=1}^{n} e^{a_j}}$$

which is equal to softmax function applied to a_i

$\therefore$ Softmax activation function is invariant to constant shifts in the input values.

References:

1. https://arxiv.org/abs/1409.1556
2. https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html
3. https://en.wikipedia.org/wiki/Early_stopping
4. https://pytorch.org/vision/stable/transforms.html
5. CSE 574 Machine Learning Assignment 2 submission by Dharma. Acha
6. https://pandas.pydata.org/