ECE4179 Assignment#3

Evan Chun Yeung Tan

TOTAL POINTS

100 / 100

QUESTION 1

Question 1 40 pts

1.1 Question 1.1 20 / 20

✓ **- 0 pts** **Well done!**

1.2 Question 1.2, 1.3 and 1.4 20 / 20

✓ **- 0 pts** **Well done!**

QUESTION 2

Question 2 45 pts

2.1 Question 2.1 25 / 25

✓ **- 0 pts** **Good Job!**

2.2 Question 2.2 10 / 10

✓ **- 0 pts** **Good Job!**

2.3 Question 2.3 10 / 10

✓ **- 0 pts** **Good Job!**

QUESTION 3

3 Question 3 15 / 15

✓ **- 0 pts** **Great job , well done!**

ECE4179: Neural Networks and Deep Learning
Assignment 3

Name: Evan Tan
Student ID: 27401995
Student Email: etan0008@student.monash.edu

Model Selection

While low losses are ideal, training loss will always go to zero and training accuracy will be 100%. Validation and test accuracy are much more important as these metrics represent a model's ability to generalize on unseen data. Achieving a training accuracy of 100% is meaningless as it just means that the model has nothing more to learn from the given training data.

Despite evaluating the test accuracy at every epoch during training, relevant ShallowCNN and DeepCNN models were selected based on the **validation accuracy**. In reality, the only datasets available would be the train and validation datasets. The test dataset would not be available during development and choosing a model based on the test dataset is biased.

OneCycleLR Scheduler

For the learning rate scheduler used, OneCycleLR, the parameters Max. and Min. LR represents the upper and lower bound of learning rate values at different iterations during training. **Percentage Ramp Up** represents the number of iterations that learning rate increases to before decreasing, which has been left at default at 30%. OneCycleLR has the added benefits of sort of having warm-up iterations, during the initial stages where learning rate is increasing. This has the benefit of reducing any bias from data seen earlier by the model. Cosine annealing is used, which means that the maximum learning rate decays to minimum learning rate to form the shape of half a cosine curve seen below.

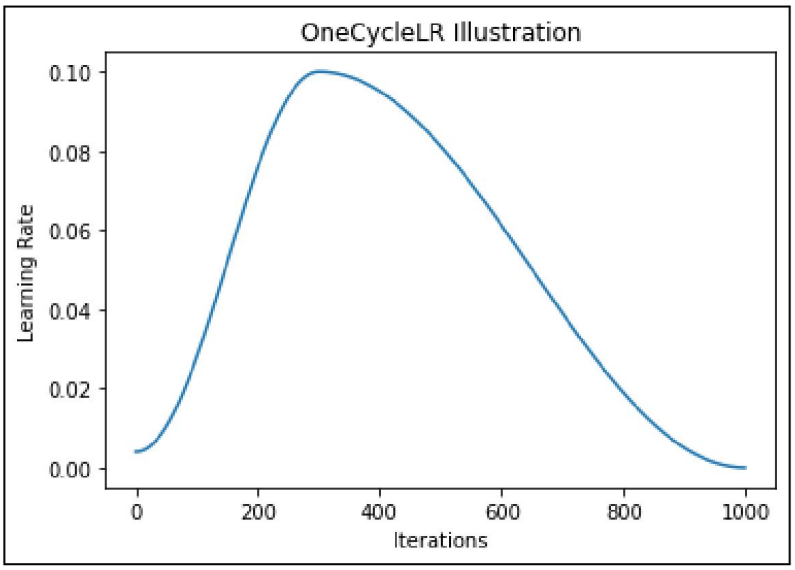

Fig. 1: OneCycleLR example, with 1000 iterations and Max. LR of 0.1

RGB Value Range

In my implementation, I have decided to use the RGB values in 0-1 instead of 0-255. I am declaring this because of the **bug in the STLData class** that prevents image data in the 0-255 range from hw3.npz from being converted into the 0-1 range that torchvision.transforms.ToTensor() would normally do. The images should be read as **np.uint8** instead of np.float32 to ensure proper transformation.

Compared to the 0-1 value range, using the 0-255 value range during training ShallowCNN for Q1.1 provides a small 3-4% performance increase, and DeepCNN for

Q2.1 provides a 10-12% performance increase. Performance of DeepCNN for Q2.2 remains the same.

1.1 ShallowCNN Training

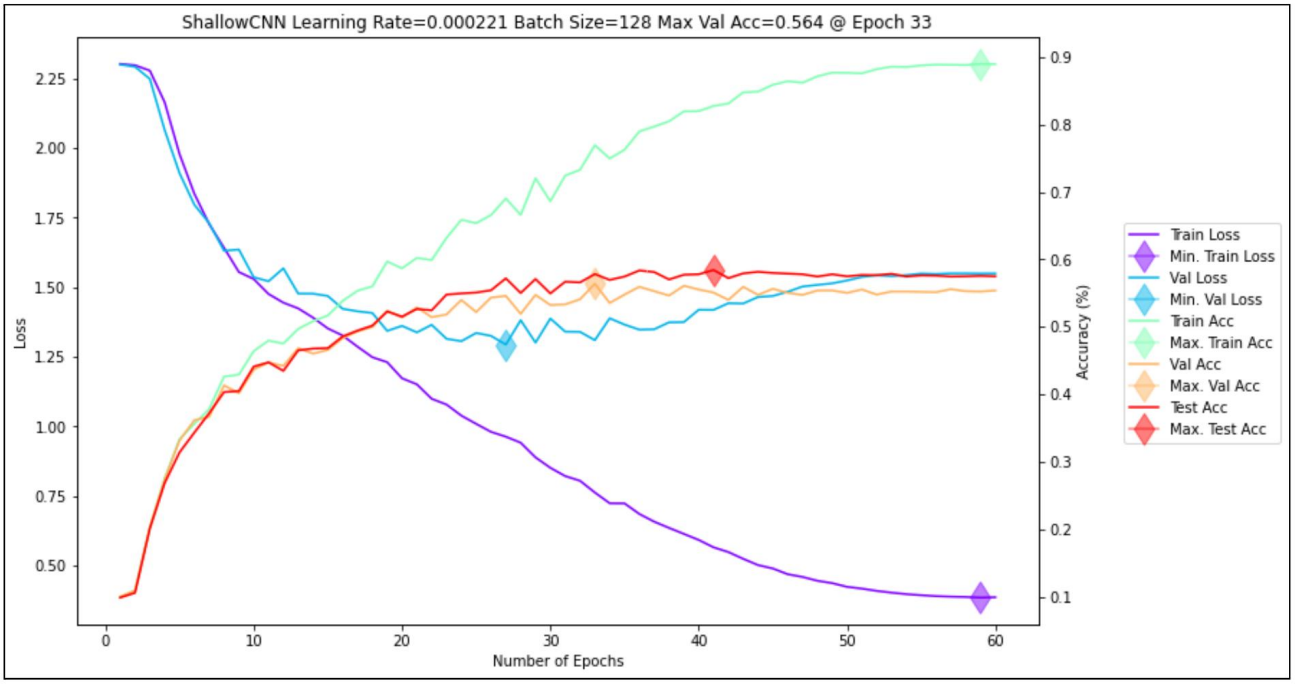

Fig. 2: ShallowCNN model training progress

Training Parameters	
Optimizer	Adam
Learning Rate (LR)	2.21e-4
Number of Epochs	60
Weight Decay	0
Scheduler	OneCycleLR
Scheduler Parameters	Max. LR = 2.21e-4 Min. LR = 2.21e-4 / 25 / 1e2 = 8.84e-8 Annealing = cosine Percentage Ramp Up = 30%
Batch Size	128
Transforms	None

Table 1: ShallowCNN model training parameters

The ShallowCNN model at Epoch 33 was selected, with the best validation accuracy of **56.40%**, and this model achieves a test accuracy of **57.83%.**

1.2 Top 5 Correct Predictions

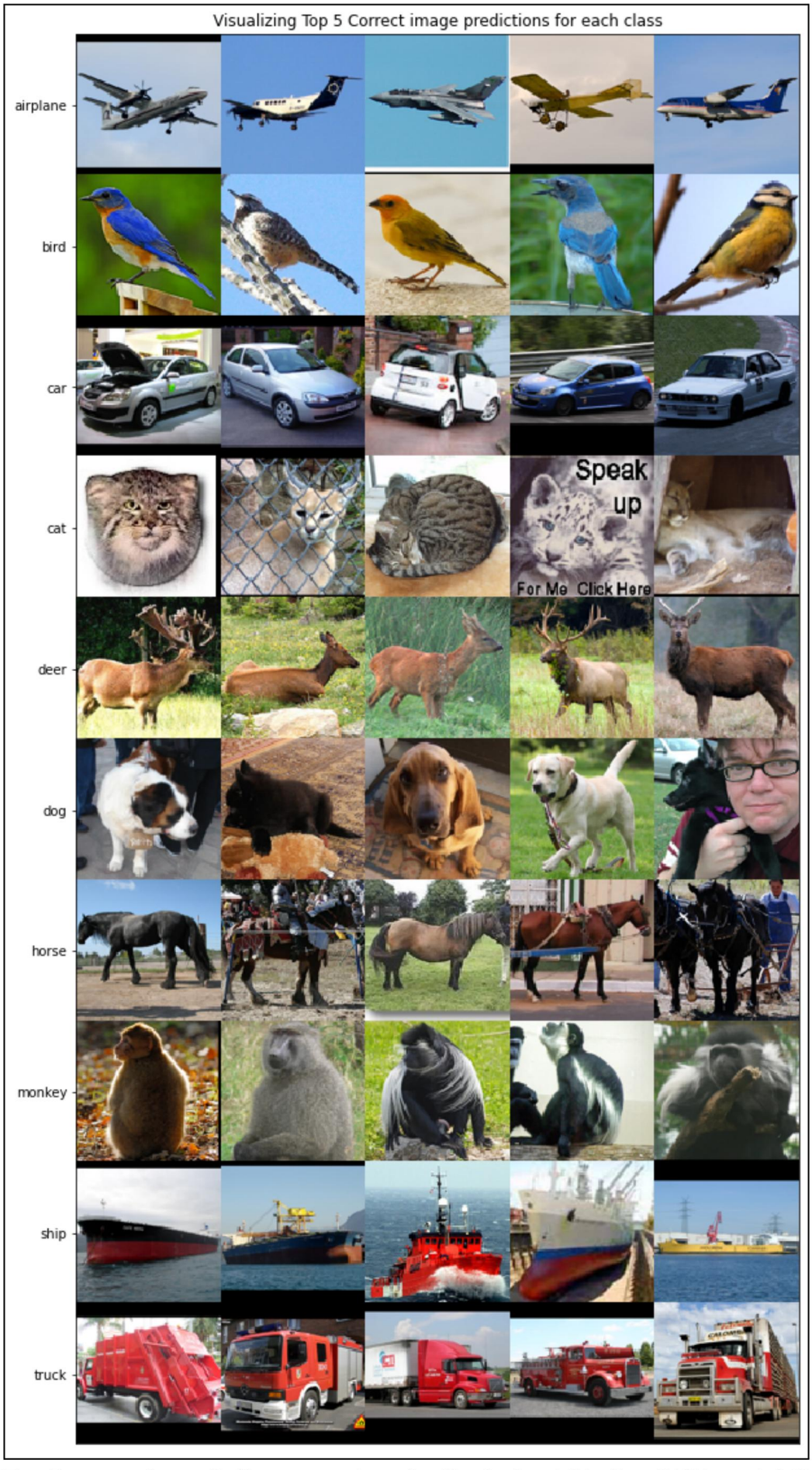

Fig. 3: Top 5 ***correct predictions*** *per class for the trained ShallowCNN model @ Epoch 33*

1.3 Top 5 Wrong Predictions

Fig. 4: Top 5 ***wrong predictions*** *per class for the trained ShallowCNN model @ Epoch 33*

1.4 Confusion Matrices (CMs)

1.4.1 CM on Train Dataset

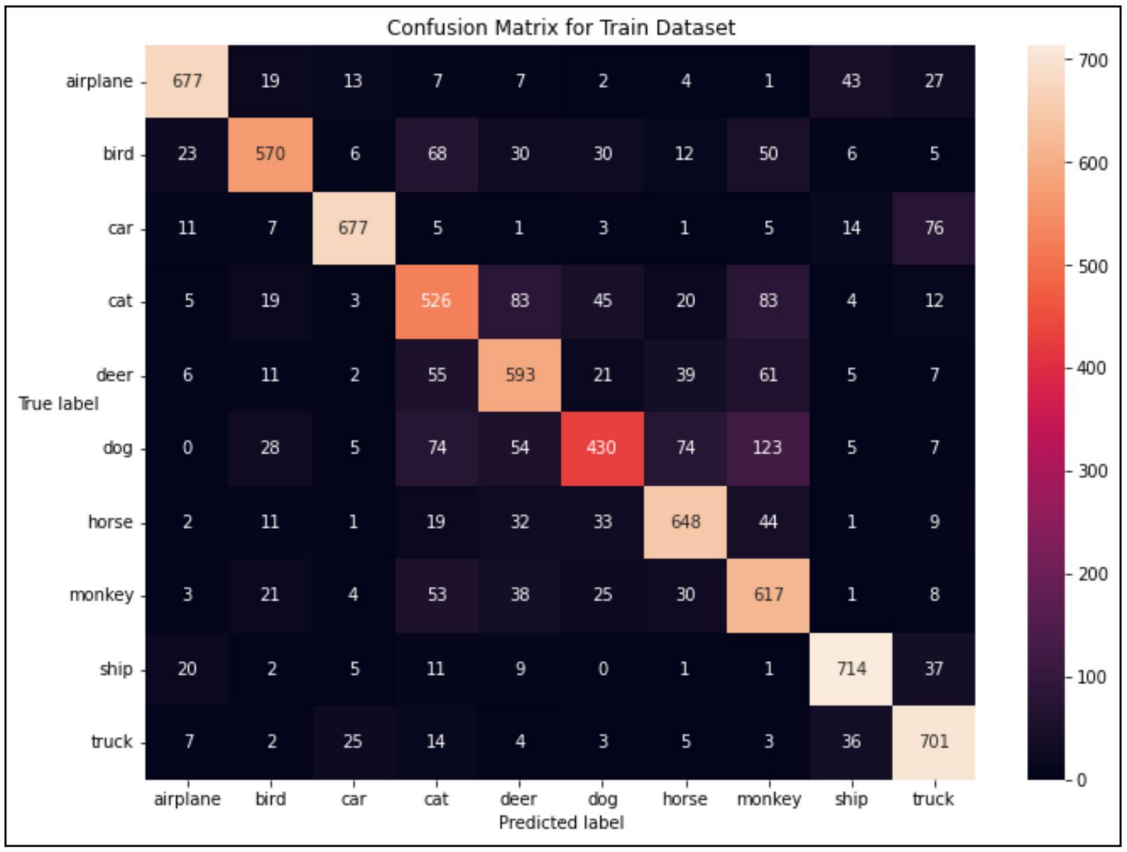

Fig. 5: Confusion matrix on the train dataset for trained ShallowCNN model @ Epoch 33

1.4.2 CM on Validation Dataset

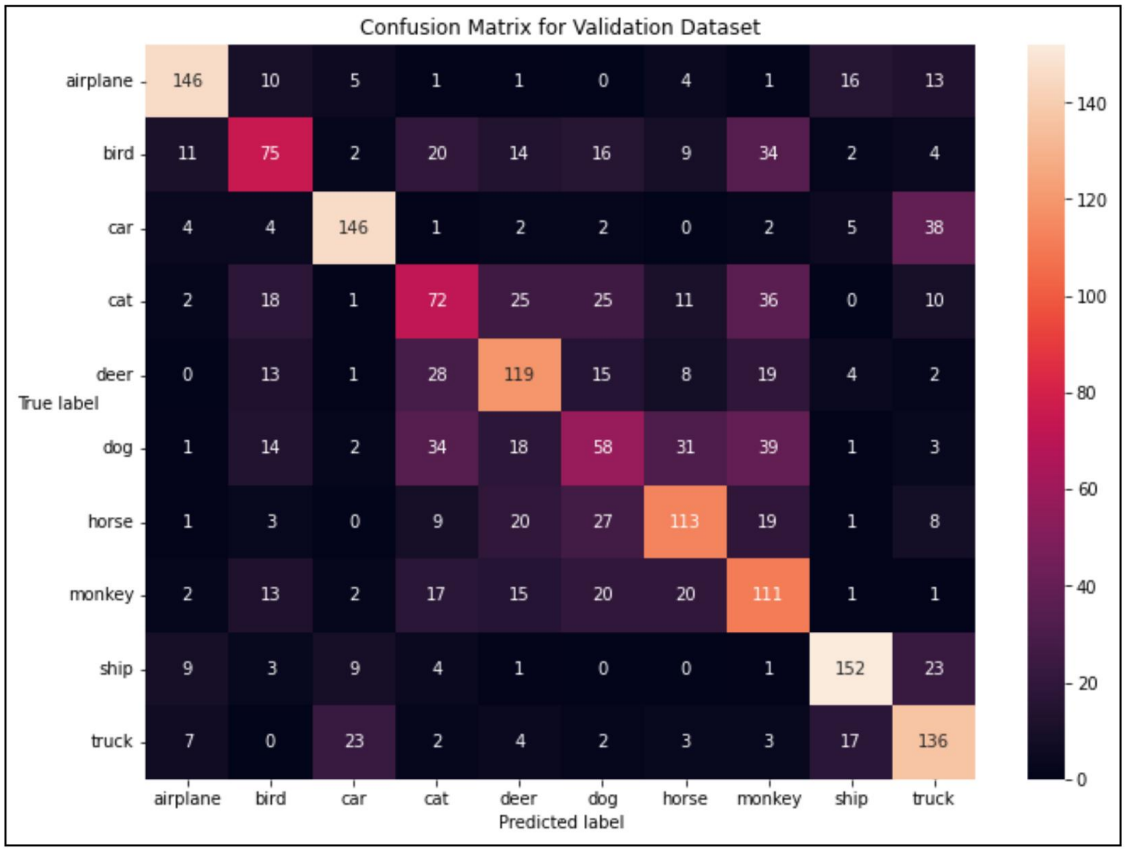

Fig. 6: Confusion matrix on the validation dataset for trained ShallowCNN model @ Epoch 33

1.4.3 CM on Test Dataset

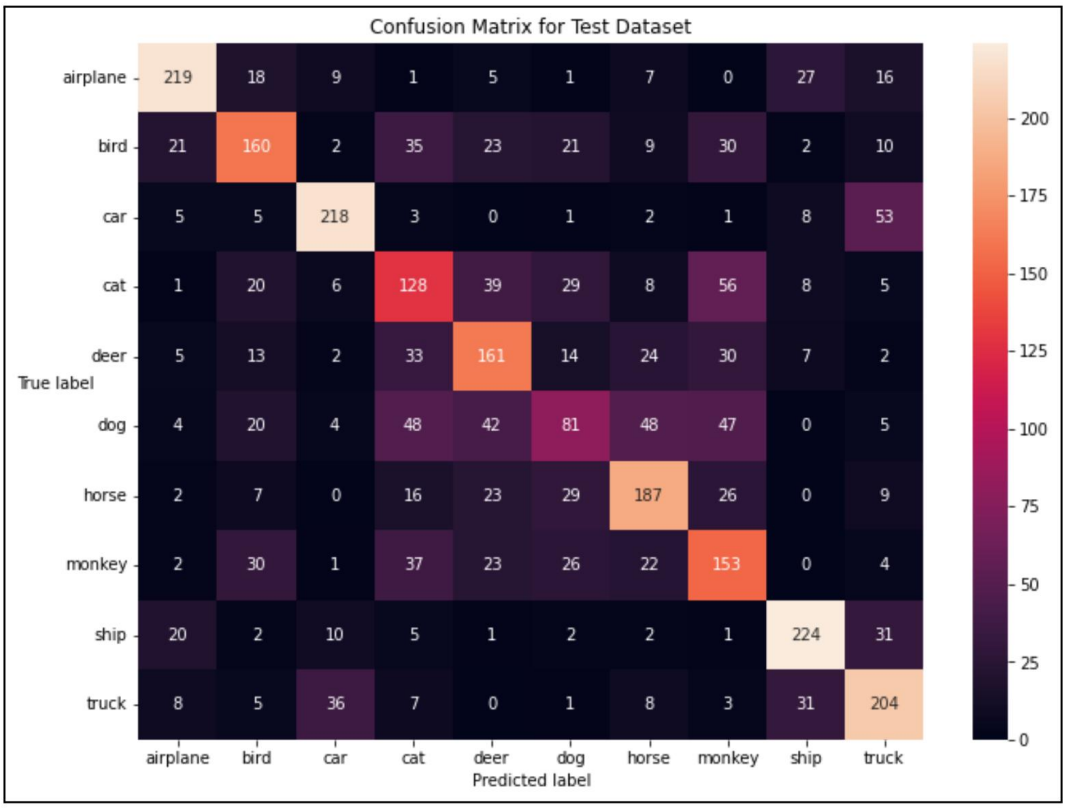

Fig. 7: Confusion matrix on the test dataset for trained ShallowCNN model @ Epoch 33

	Correct Class Predictions									
	airplane	bird	car	cat	deer	dog	horse	monkey	ship	truck
Train	677	570	677	**526**	593	**430**	648	617	714	701
%	11.00	9.26	11.00	**8.55**	9.64	**6.99**	10.53	10.03	11.60	11.39
Val	146	75	146	**72**	119	**58**	113	111	152	136
%	12.94	6.65	12.94	**6.38**	10.55	**5.14**	10.02	9.84	13.48	12.06
Test	219	160	218	**128**	161	**81**	187	153	224	204
%	12.62	9.22	12.56	**7.38**	9.28	**4.67**	10.78	8.82	12.91	11.76

*Table 2: Confusion matrix summary on different datasets, with **percentage of correct predictions** per class*

The confusion matrices across different datasets show the performance of the trained model on images of different classes. It demonstrates the strengths and weaknesses of the trained model, for example on the training dataset the lowest number of correct class predictions are for **dogs and cats**, only achieving 6.99% and 8.55% of *all correct predictions* (for all classes) respectively.

This means that the model has not learned the features of these 2 classes in a robust way. This is subsequently reflected in the validation and test dataset confusion matrices, where the number of cats and dogs correctly classified is lowest compared to other classes. This suggests that the model seems to work fine for the other 8 classes, but could be improved

by using a separate classifier for cats and dogs. Across all datasets, the model has difficulty differentiating between cars and trucks.

2.1 DeepCNN Without Data Augmentation

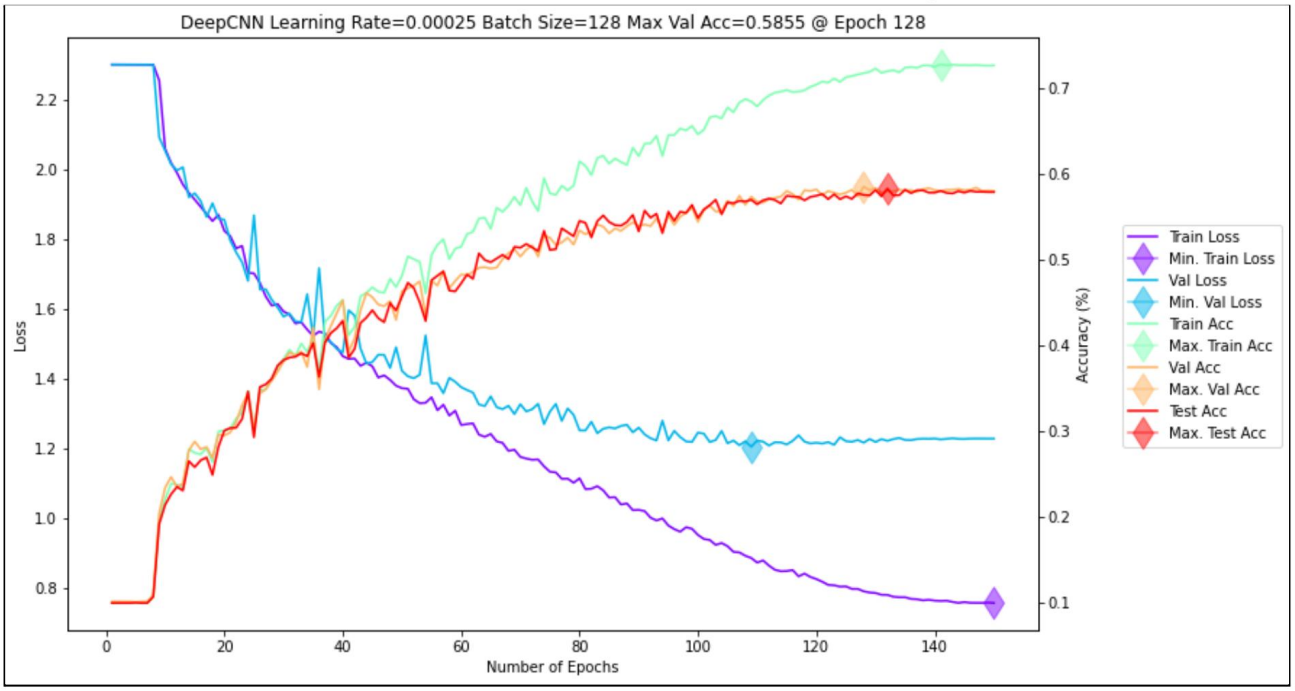

Fig. 8: DeepCNN model (without data augmentation) training progress

Training Parameters	
Optimizer	Adam
Learning Rate (LR)	2.5e-4
Number of Epochs	150
Weight Decay	0
Scheduler	OneCycleLR
Scheduler Parameters	Max. LR = 2.5e-4 Min. LR = 2.5e-4 / 25 / **2** = 5e-6 Annealing = cosine Percentage Ramp Up = 30%
Batch Size	**128**
Transforms	None

Table 3: DeepCNN (without data augmentation) training parameters

The trained DeepCNN model at Epoch 128 was selected based on the highest validation accuracy of **58.55%** and this model achieves **57.60%** test accuracy.

2.2 DeepCNN with Data Augmentation

The following data augmentations were used for training the following DeepCNN models. In the following table, the **symbol p** represents the *probability* of the transform being applied.

Data Augmentation	
Mean **(Train Dataset)**	[0.44723063707351685, 0.4396424889564514, 0.40495729446411133]
Standard Deviation **(Train Dataset)**	[0.22490786015987396, 0.22173990309238434, 0.22371038794517517]
Test Transform	Normalize with **Train Dataset** Mean and Standard Deviation
Validation Transform	Normalize with **Train Dataset** Mean and Standard Deviation
Train Transform	Normalize with Train Dataset Mean and Standard Deviation, Rotation, between -45 to +45 degrees, Horizontal Flip, p = 0.5, Vertical Flip, p = 0.05, Grayscale, p = 0.2, Erasing, p = 0.5,

Table 4: Data Augmentation parameters for DeepCNN training

The train, test and validation datasets were normalized using means normalization from mean and standard deviation calculated from the training dataset. This shifts the range of values from [0,1] (after the ToTensor() transform) to between [-1.9885, 2.6599] which is strange as [-1,1] is expected but all 3 channels across the dataset now have a mean of 0 and variance of 1. By performing normalization of the dataset, the data is now normally distributed and it is easier for the model to learn. These values for normalization were also used for the test and validation datasets.

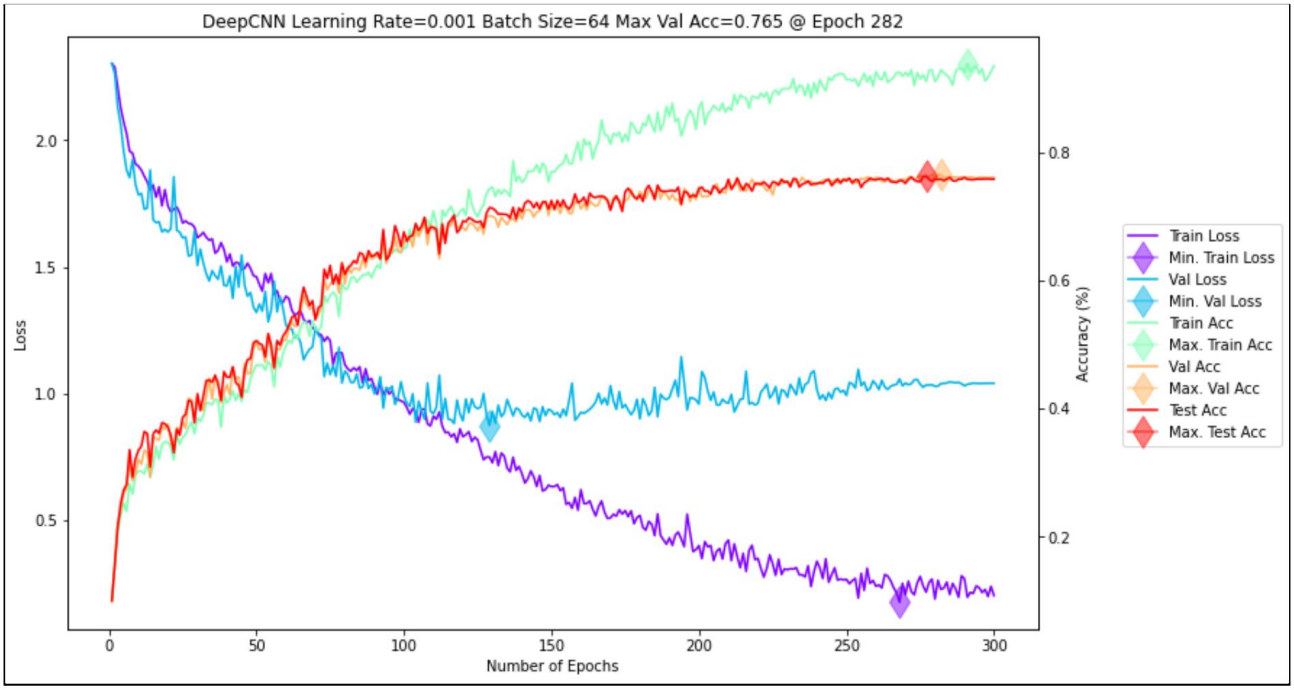

Fig. 9: DeepCNN model (with data augmentation) training progress

Training Parameters	
Optimizer	Adam
Learning Rate (LR)	1e-3
Number of Epochs	300
Weight Decay	0
Scheduler	OneCycleLR
Scheduler Parameters	Max. LR = 1e-3 Min. LR = 1e-3 / 25 / 1e3 = 4e-8 Total Steps = Epochs * Batch Steps Annealing = cosine Percentage Ramp Up = 30%
Batch Size	64
Transforms	**see Table 4**

Table 5: DeepCNN (with data augmentation) model training parameters

The DeepCNN model at Epoch 269 was selected with the highest validation accuracy of **76.50%** and this model achieves a test accuracy of **75.80%**.

Using data augmentation, the DeepCNN model improves performance by 17.95% on the validation dataset, and 18.20% on the test dataset, and greatly outperforms the model in Q2.1 that was trained without data augmentation.

2.3 Network Design

Changes	Rationale
LeakyReLU	All ReLU activations are changed to **LeakyReLU** activations in order to reduce any vanishing gradient problems, as the gradient of ReLU when input < 0 is 0 whereas LeakyReLU has a tiny gradient of 0.01 * input when input < 0.
Remove middle convolution layer in conv blocks	The middle convolution layer in each block is removed as it is acting as a fully connected layer with the same number of input and output channels and a 1x1 feature map.
Regularization - Dropout	Added after each LeakyReLU activation with a probability of 0.25 to be zeroed during training, and the remaining inputs are scaled accordingly.
Regularization - Batch Norm	Added to reduce covariate shift during training, by ensuring input to each hidden layer has a mean of 0 and variance of 1, reducing the problems of input values changing by ensuring the batch data is normally distributed.
ConcatPool2d	Concatenating the average pooling layer with a maxpool layer, as recommended by fastai which saw a small improvement compared to just using average pooling as this preserves more information. Since input data is always changing, using a single average pooling or max pooling may not work all the time. By using both, it is a lazy way to make the network more robust.

Table 6: Details of network changes and rationale

The new network is referred to as **CustomCNN**.

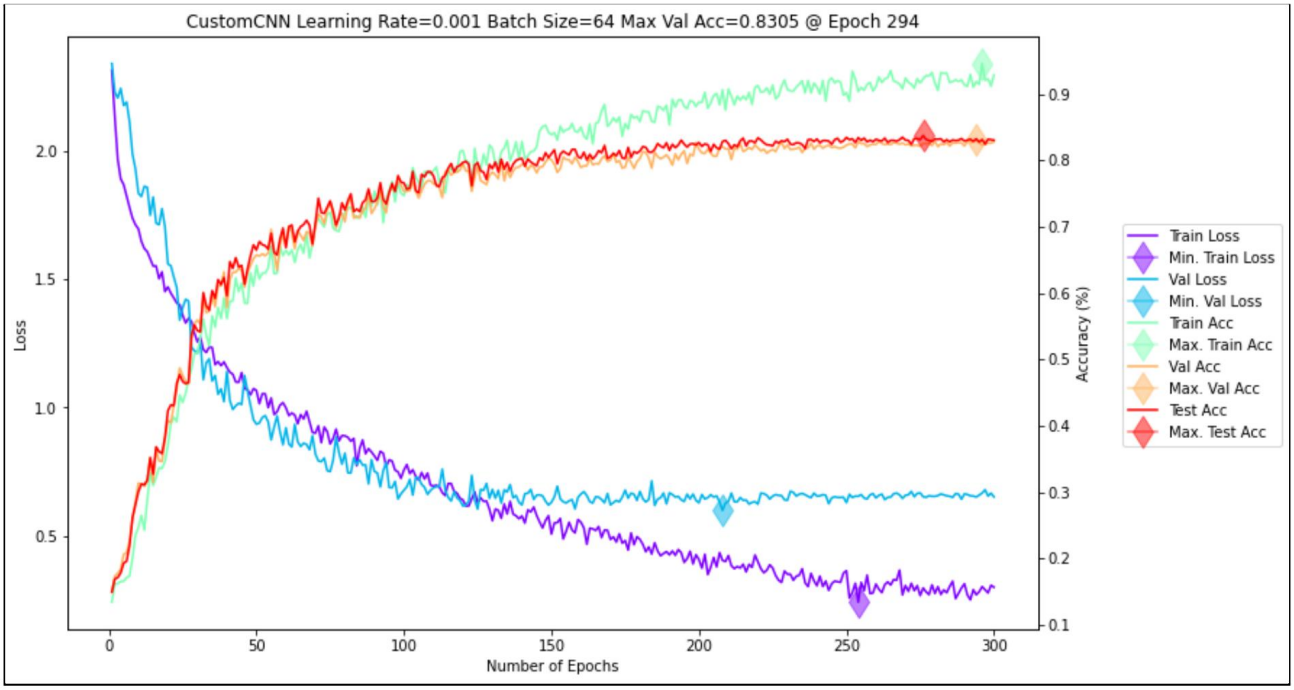

Fig. 10: CustomCNN model (with data augmentation) training progress

Network	Model Epoch	Number of Parameters	Validation Accuracy (%)	Test Accuracy (%)
DeepCNN	282	901,066	76.50	75.80
CustomCNN	294	1,476,458	83.05	83.26

Table 7: Comparison of network architectures

Keeping the training parameters exactly the same, there is some performance gain as seen in Table 7, where CustomCNN is better by 6.55% validation accuracy, and by 7.46% test accuracy. **However**, there is a large increase in the number of trainable parameters, where CustomCNN has an increase of 63.86% in the number of parameters. Both models converge in a similar manner but given that CustomCNN has many more parameters, some training parameters could be tuned in order to improve the model performance further.

3 Occlusion Sensitivity Study

By varying the size of the patch, the defining features for the input image can be identified. This is because those patches are required for the model to output the correct class. By using a patch of size (1,1) and stride of 1, the output pretty much just traces the outline of the input image seen in the following figures. However, if the size of the patch is increased to (8,8) and stride of 8, features that allow the model to classify the input images are removed, and the model has a reduced ability to correctly classify these images.

(see next page)

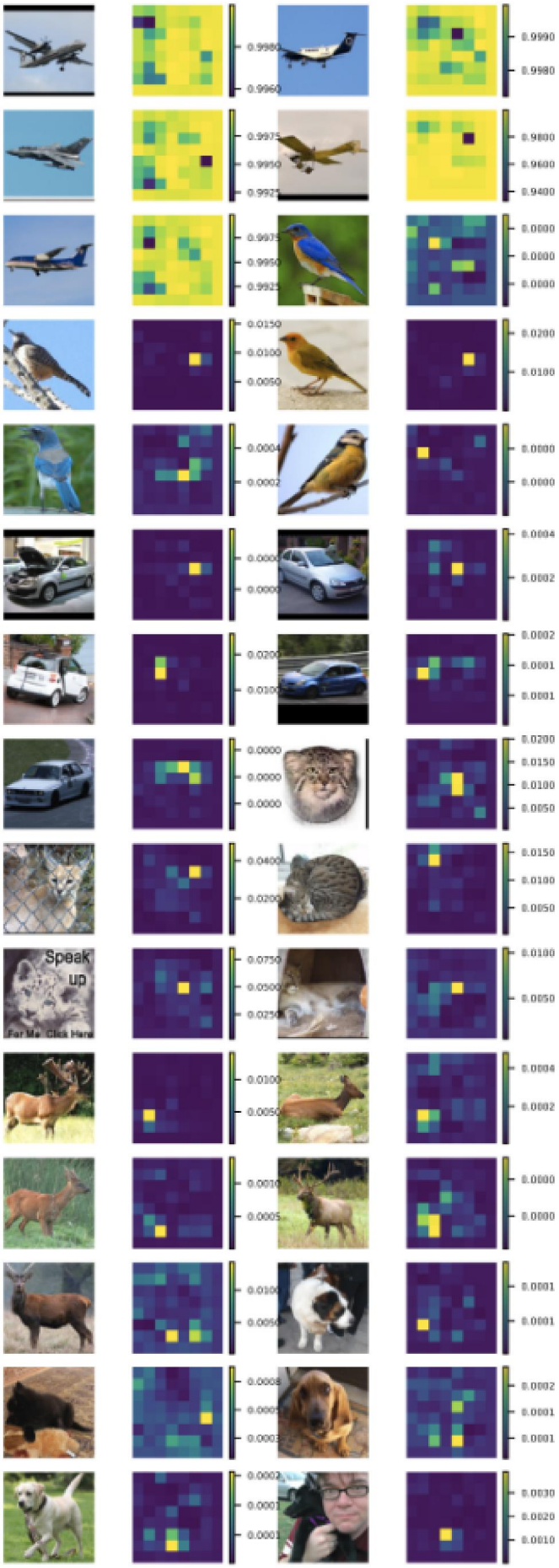

Fig 11: Occlusion for images in Q1.2, with patch size (8,8) stride 8

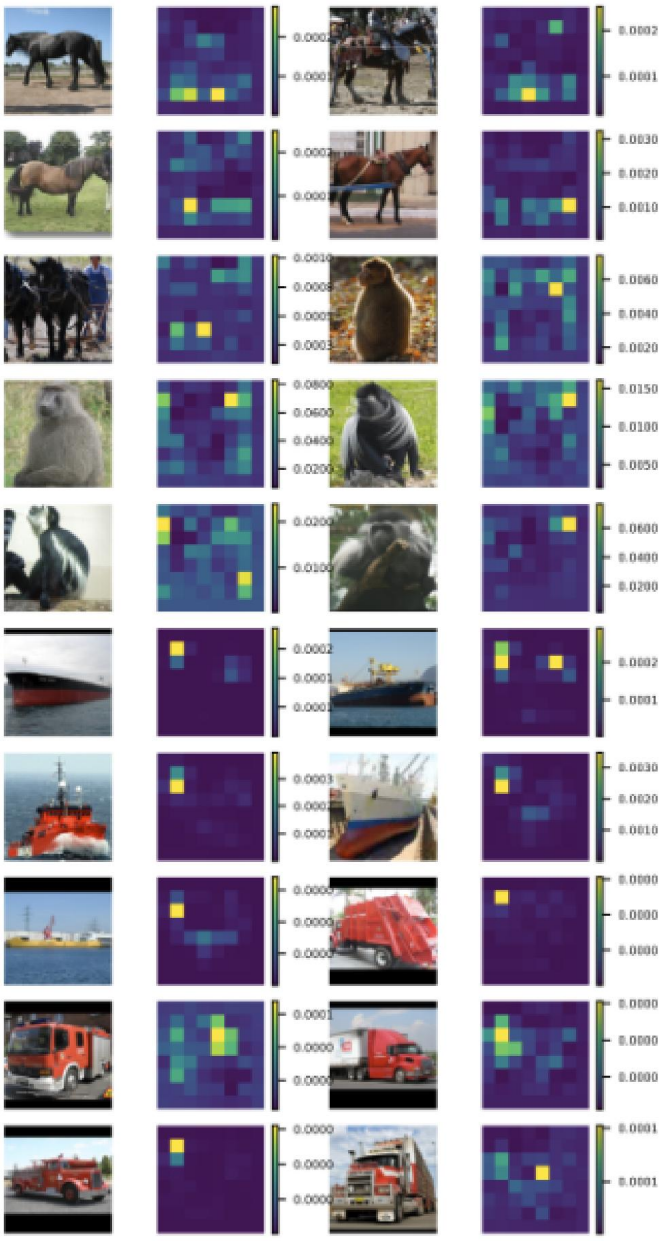

Fig. 12: Occlusion for images in Q1.2, with patch size (8,8) stride 8

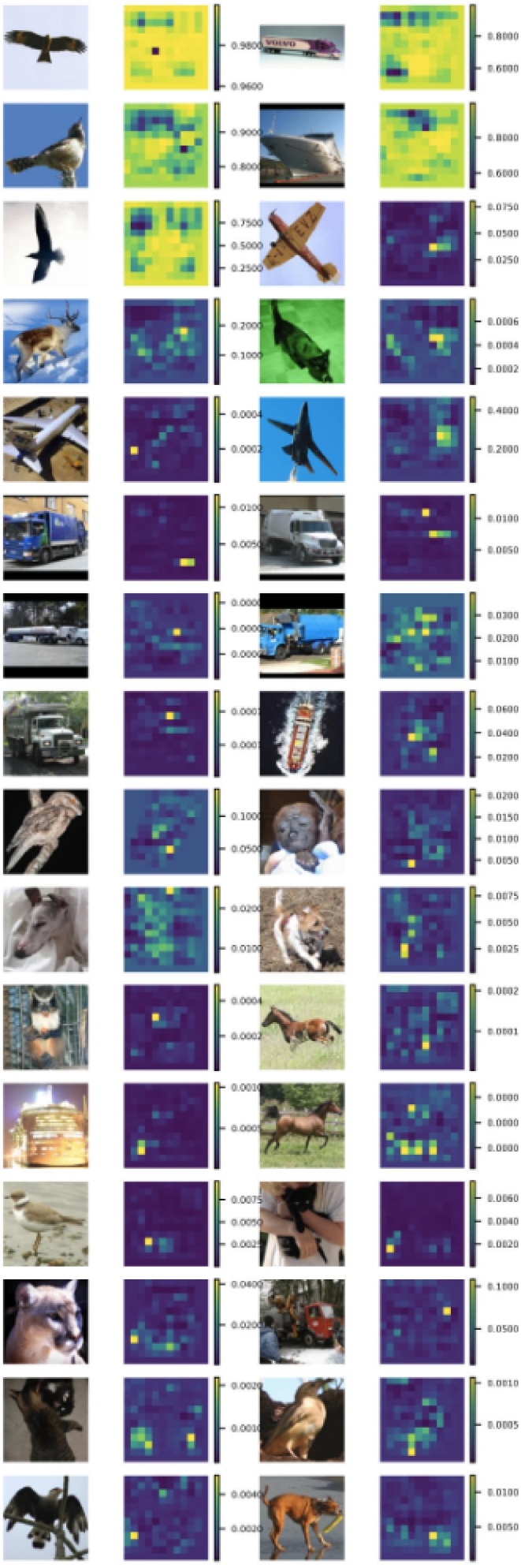

Fig. 13: Occlusion for images in Q1.3, with patch size (8,8) stride 8

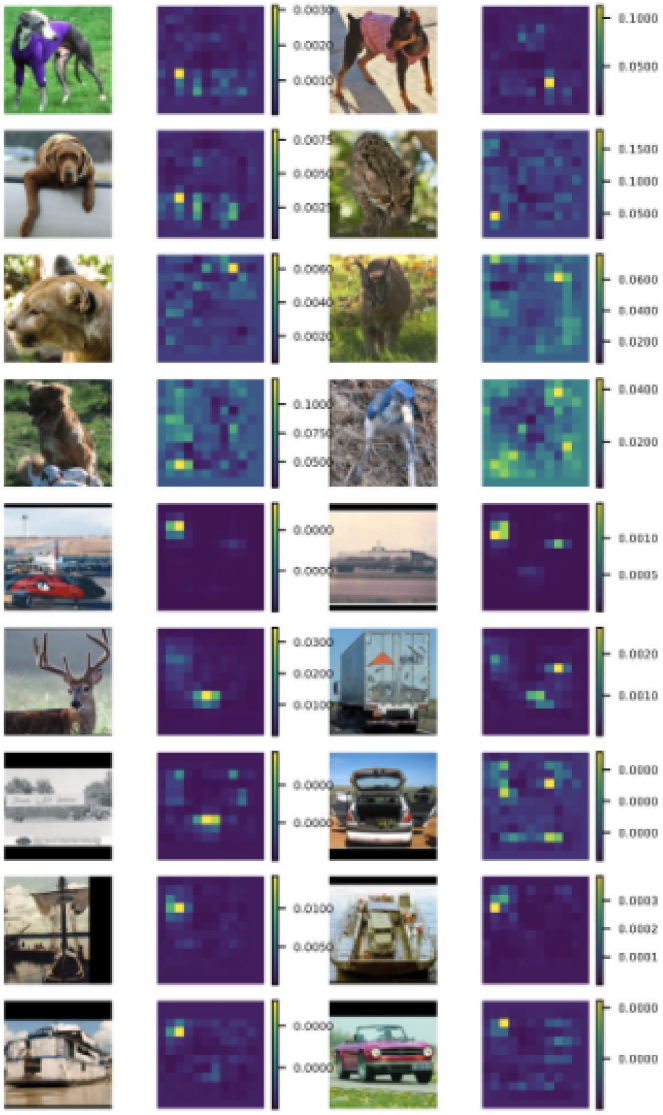

Fig. 14: Occlusion for images in Q1.3, with patch size (8,8) stride 8