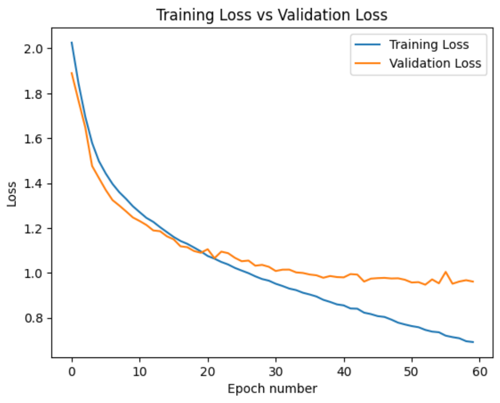
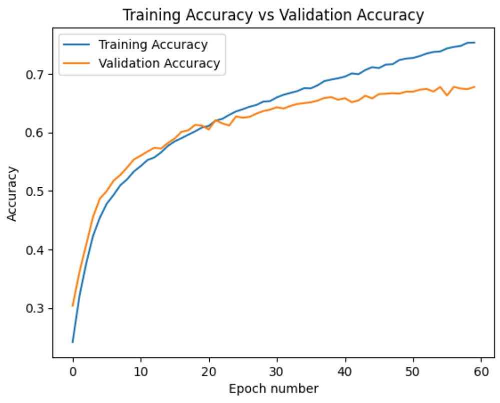
Parameter Name	Value	Parameter Name	Value
Train Data Ratio	0.8	Patch Size	4 x 4
Heads	4	Overlap Patches	False
Class Token from different Layers	False	Embedding Dimension	256
Number of Encoders	12	MLP Size	3072
Dropout (Attention, MLP, Projection, Embedding)	0.1	Batch Size	128
Epochs	60 and 100 (for few)	Learning Rate	1e-5

Table 1: Default configuration data for training the vision-transformer

1 Experiment-1



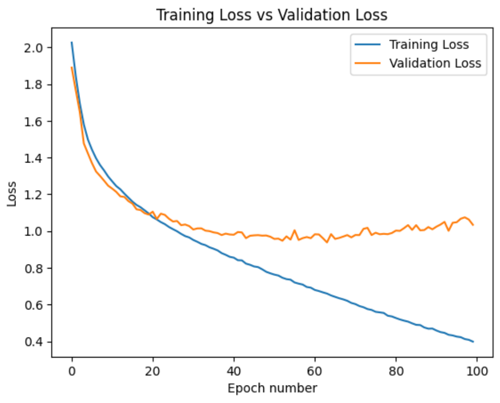
(a) Loss curve for 60 epochs



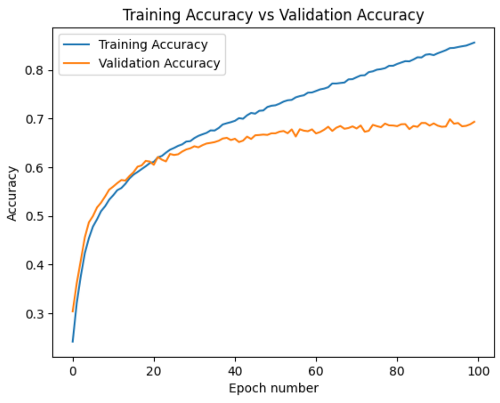
(b) Accuracy curve for 60 epochs

Epochs	Train Accuracy	Test Accuracy
60	80.10%	68.06%
100	90.41%	69.77%

(e) Accuracy data for different epochs



(c) Loss curve for 100 epochs



(d) Accuracy curve for 100 epochs

From the above results, we can observe that we achieve an accuracy $> 68\%$ when trained vision-transformer model with cifar-10 data with default configuration (1). By increasing the number of epochs for training we see that the model starts to overfit due to the complexity involved. Hyperparameter tuning of encoder depth, heads, dropouts etc will help further improve the accuracy which is not investigated here.

2 Experiment-2

Default configuration (1) is used, except the **Train Data Ratio** parameter is changed before passing it. From the results (2) we can observe that, the more the data, the better the accuracy of the model. Similar to previous experiment we train for 2 different epochs values (60 and 100) to check if the model starts over-fitting after certain iterations (Relevant plots in the notebook). The gap between the train and validation loss kept increasing after certain iteration. This can be fixed by reducing the complexity of model with help of hyperparameter tuning.

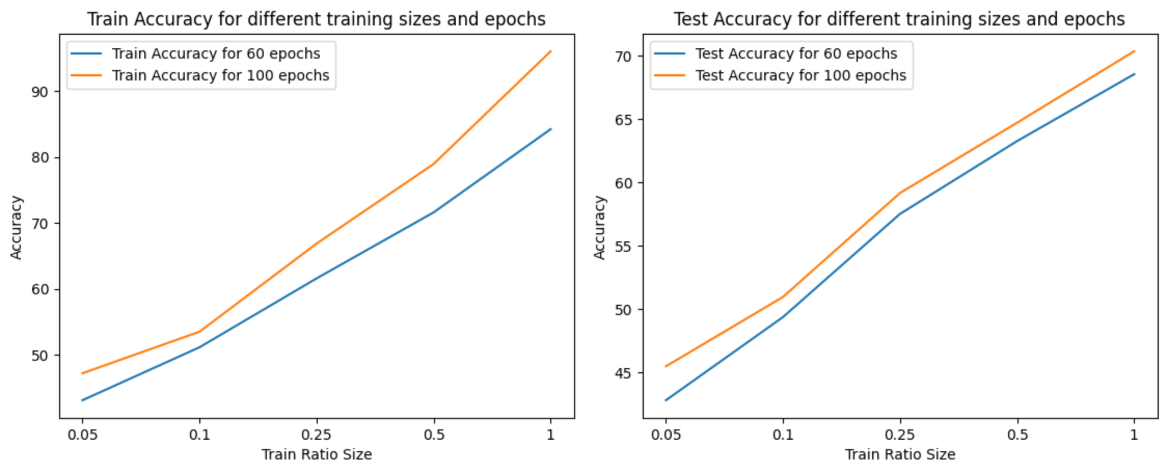
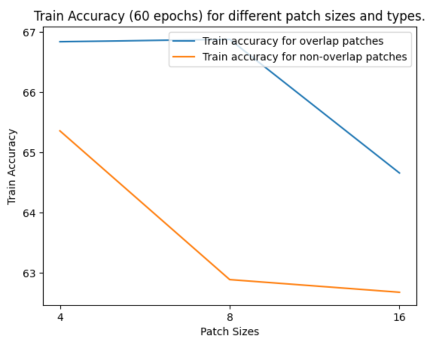
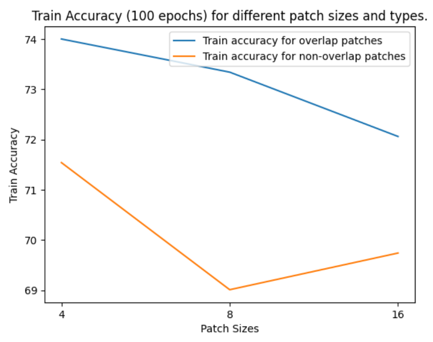


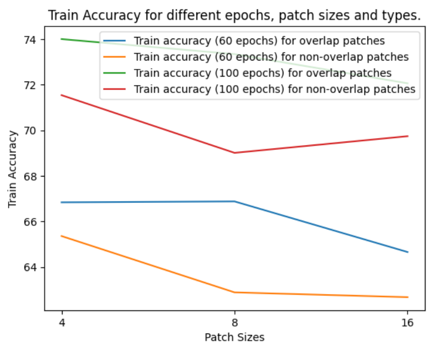
Figure 2: Effect of train percent on model



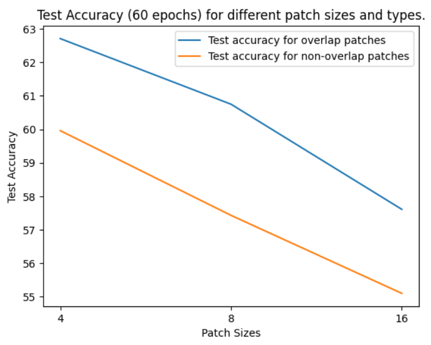
(a) Train accuracy (60 epochs)



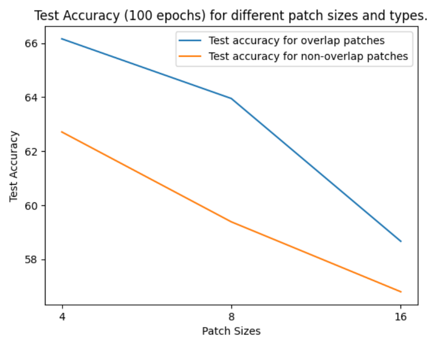
(b) Train accuracy (100 epochs)



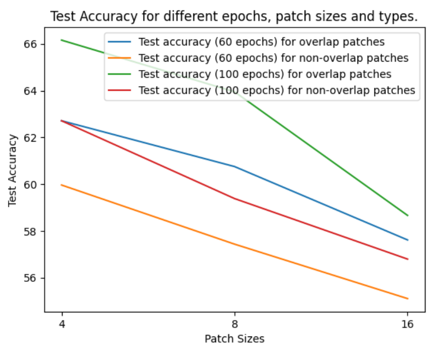
(c) Train accuracy for different epochs



(d) Test accuracy (60 epochs)



(e) Test accuracy (100 epochs)



(f) Test accuracy for different epochs

Figure 3: Effect of overlap vs non-overlap patches with different patch sizes.

3 Experiment-3

Patch Size, Overlap Patches, Number of Encoders parameters from default configuration (1) are changed before passing it for training the model. Number of Encoders is changed here because more patches are created with smaller patch size leading to cuda memory limits. From the results (3), we can observe that the overlapping patches was always able to get better results compared to it's counterpart and sometimes (3f) even when trained on more epochs because it's able to see better representation compared to non-overlap where it might miss details part of continuity in the image. Also smaller the patch-size better the results. With smaller patch-size it's able to capture more number of representations (patch count) of the image thus allowing to capture fine-grained details within the image. Sometimes there is an increase in train-accuracy when patch-size increases, this might be because of other configuration parameter of the model.

4 Experiment-4

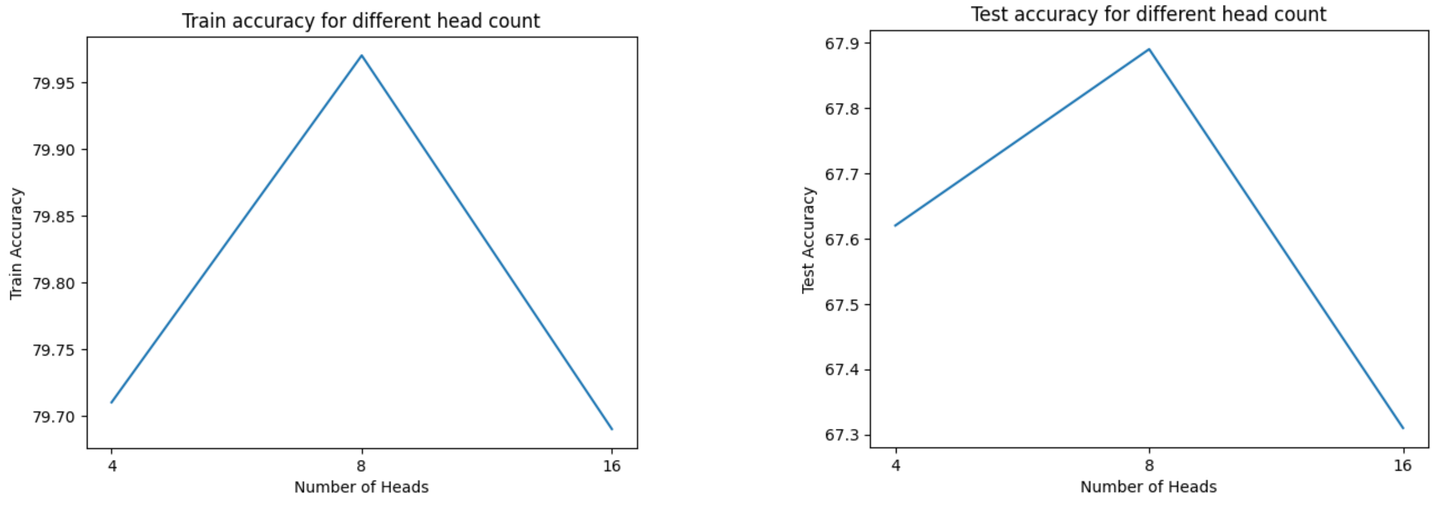


Figure 4: Effect of number of heads on model training.

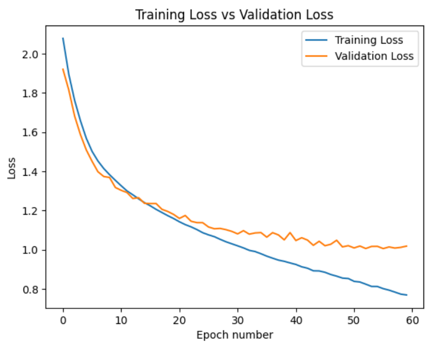
Heads parameter from default configuration (1) are changed before passing it for training the model. From the results (4), we see that the accuracy's are pretty much close to each other even when varying the number of heads which tries to look into different features. This can be due to the high-complexity of the model which has 12 encoder layers so pretty much all the heads are able to see the proper representation with slight change. Can take benefit of increasing the number of heads till an extent after which we will be observing the opposite effect. An increased accuracy is observed when training with 100 epochs for number of heads = 4 (Can be seen in the notebook, for other heads failed due to network failure) but the model is over-fitting already so need hyperparameter tuning for better generalization.

5 Experiment-5

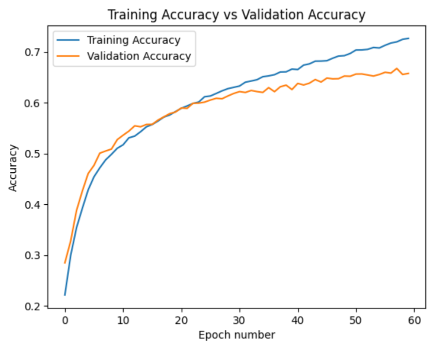
Class Token from different Layers parameter from default configuration (1) are changed before passing it for training the model. From the results (5), we can observe that using the mean of class representation data from different layers is reducing the overall accuracy of the model. This could be due to the number of encoder layers being used (12 in this case). Class token from different layers represent different information which might be conflicting with the underlying class label, thus giving an equal weight might not be the ideal case always.

6 Experiment-6

From (6), We can observe that the attention maps in correct classification, the model is looking at the truck wheels and few parts of the truck. While in incorrect classification which is of similar class (automobile in this case), the model is looking at the wheel in the image which is also present in the similar position for the actual class and also with few other places and incorrectly classified. Thus the model is able to learn meaningful representation from the train dataset.



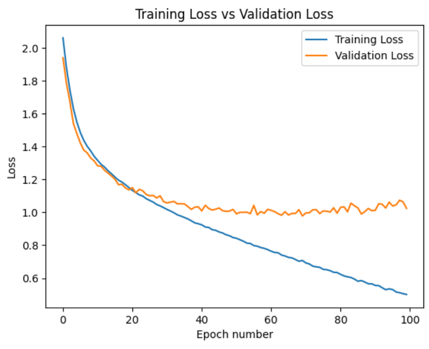
(a) Loss curve for 60 epochs



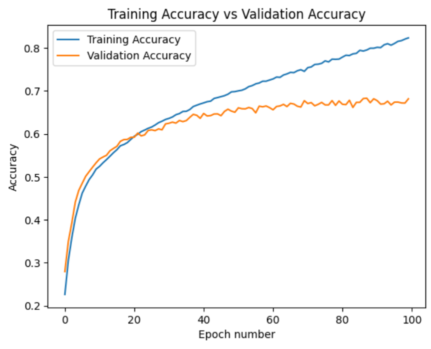
(b) Accuracy curve for 60 epochs

Epochs	Train Accuracy	Test Accuracy	Class Toker
60	80.10%	68.06%	No
60	76.7%	66.04%	Yes
100	90.41%	69.77%	No
100	87.03%	68.2%	Yes

(e) Accuracy data for different epochs



(c) Loss curve for 100 epochs



(d) Accuracy curve for 100 epochs

Figure 5: Effect on class token on accuracy

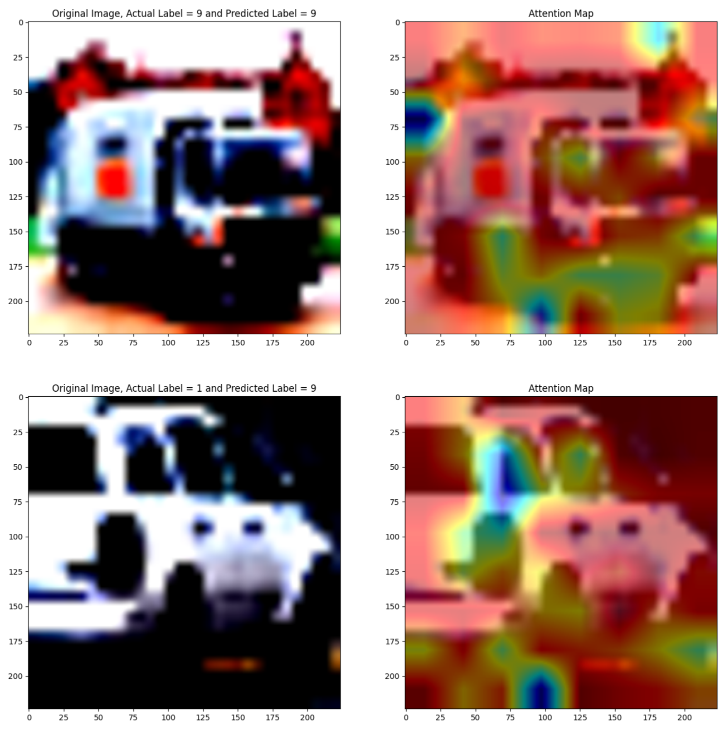

Original Image, Actual Label = 9 and Predicted Label = 9
Attention Map
Original Image, Actual Label = 1 and Predicted Label = 9
Attention Map

Figure 6: Attention Maps for Correct and Incorrect classification