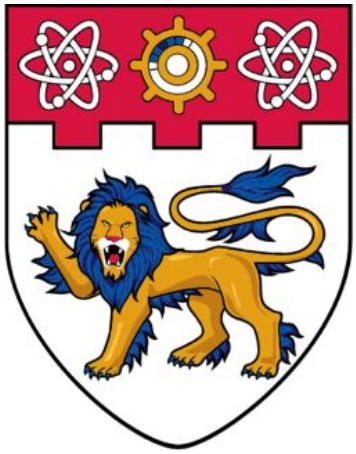


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Year 20/21 - Sem 1

CZ/CE 4042 Neural Networks and Deep Learning

Project

Car Make Recognition

Lee Kai Shern U1820793J
Yew Wei Chee U1820962G

1. Introduction

Fine-grained car classification and recognition can be used for various purposes especially in modern transportation systems such as regulation, description and indexing. The overall goal of this project is to classify an image into one of the 163 car makers and 1716 car models.

With reference to the currently available paper on this topic *[3]*, we decided to modify the previously published architecture to see whether improvement on model classification can be achieved. Besides, we will also extend the classification to more car makers and models.

The published paper only exploits part of the dataset to accomplish car model classification and verification. We will examine more advanced models to complete car model & maker classification and perform verification using more advanced loss functions.

2. Review on Related Work

Previous car model research papers emphasize only on the car model classification such as using wireframe models to fit a car *[1]* and constructing 3D space curves to match 2D image curves *[2]*. However, most of the works are only restricted to a small number of car models.

The latest paper related to car model classification uses convolutional neural networks (CNN) which has been gaining success in many computer vision topics to classify the car into 431 models and 75 makers *[3]*. The paper also tried to explore the unprecedented attribute prediction using the Overfeat *[4]* model. The researchers only used 30955 images for training and testing of classification tasks from CompCars *[3]*, which is only approximately a quarter of the images from the same source.

3. Method

3.1 GoogLeNet *[5]*

A type of CNN network with 22 layers introduced in 2014. It allows the network to choose between multiple convolutional filter sizes in each block and have max-pooling layers in between. However, the model is losing trend because of improvement achieved in other CNN-based models.

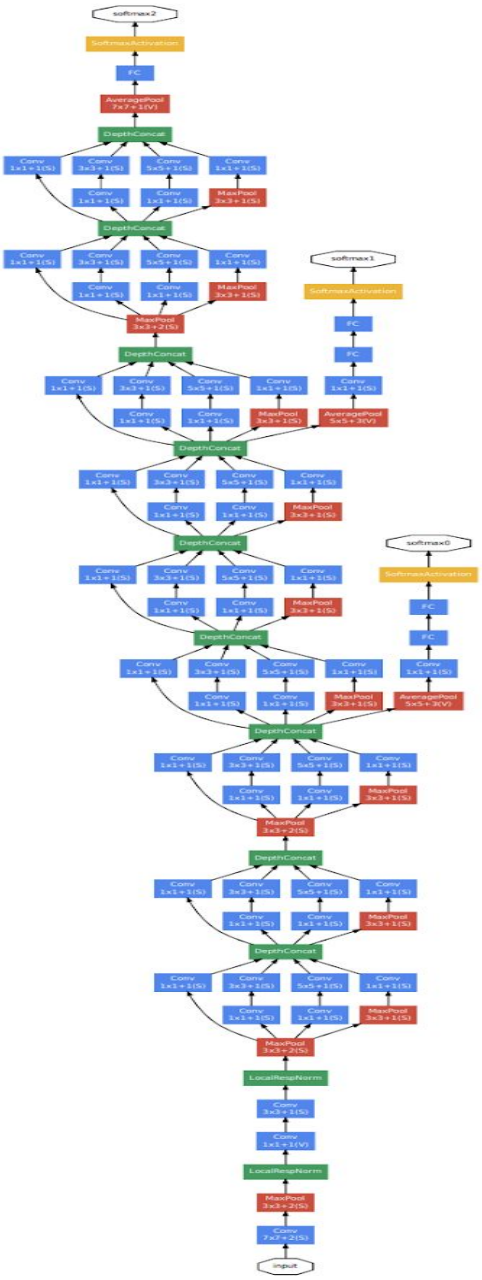


Image from https://paperswithcode.com/method/googlenet

3.2 Inception_V3 *[6]*

An Inception model introduced in 2015 and developed based on previous Inception models. The main improvements include Label Smoothing, factorized 7 x 7 convolutions and auxiliary classifier etc.

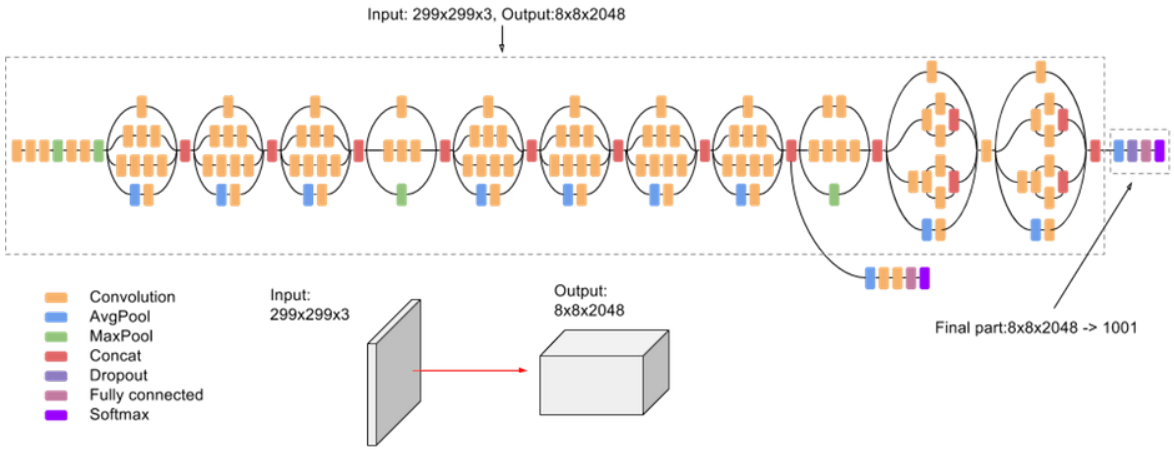


Image from https://paperswithcode.com/method/inception-v3

3.3 Triplet loss *[7]*

Loss function which compares baseline input to a positive and a negative input. The loss function will minimize the distance from baseline to positive input and maximized the distance from baseline to negative input. The classifier will be retrained every time when a new car model is added into the database. The loss function can be formulated using Euclidean distance function:

$$L(A, P, N) = max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

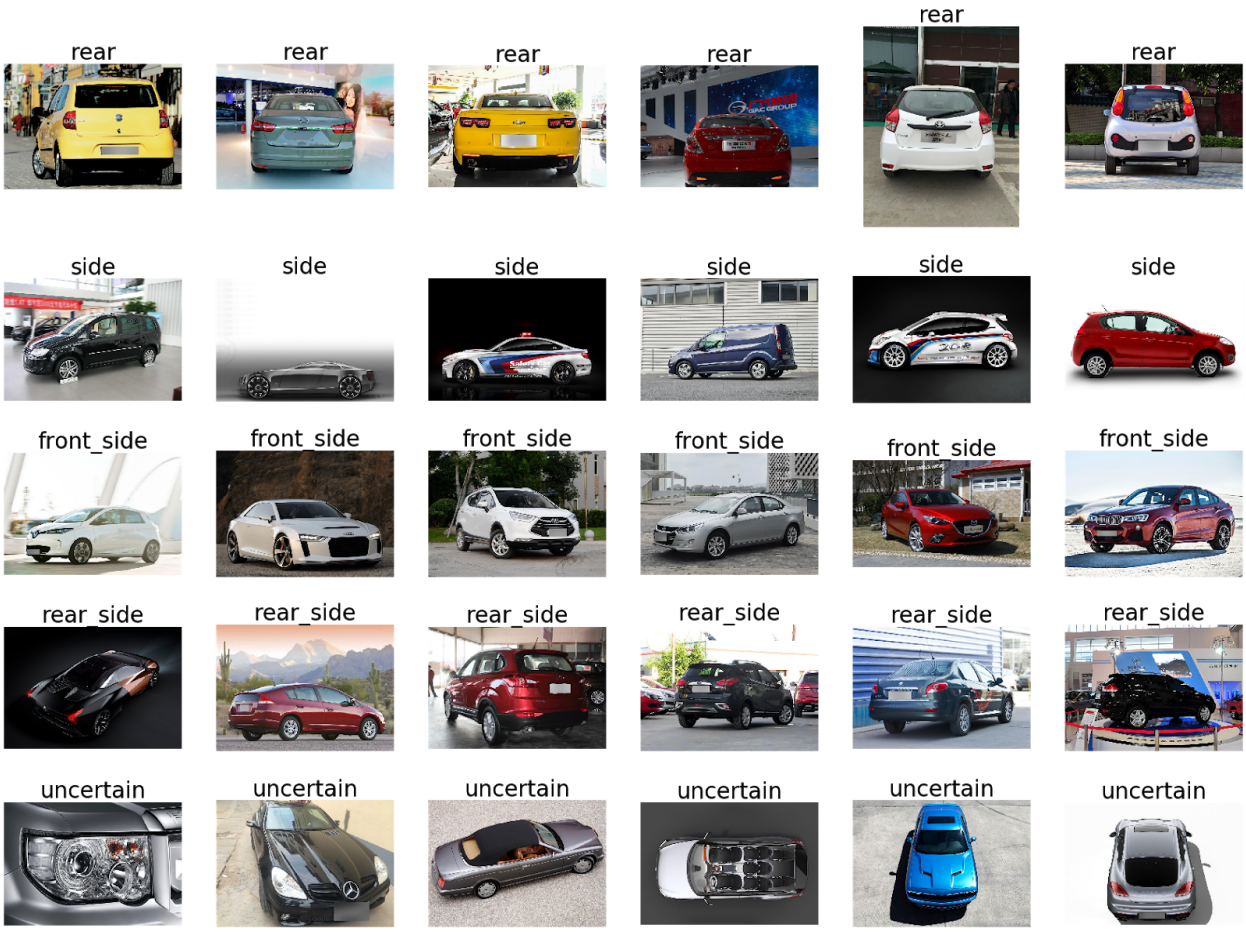
Where A is the baseline input, P is positive input and N is negative input. α is the margin between positive and negative pairs and f is the embedding.

4. Data

4.1. Introduction

The CompCars datasets have two scenarios which are images from web-nature and surveillance-nature images. For this project, we will only use the web-nature data that captures the entire car which is 136726 images in total. The images can be classified into 163 car makes with 1716 car models.

The dataset includes 4 unique features which are car hierarchy, car attributes, viewpoints and car parts. Car hierarchy contains information of car makers and car models that are the results of classification to be achieved. The other feature used will be the viewpoints. We used the viewpoints to split the data to ensure an even distribution of the different viewpoints of the same car model in test, train, validation data.

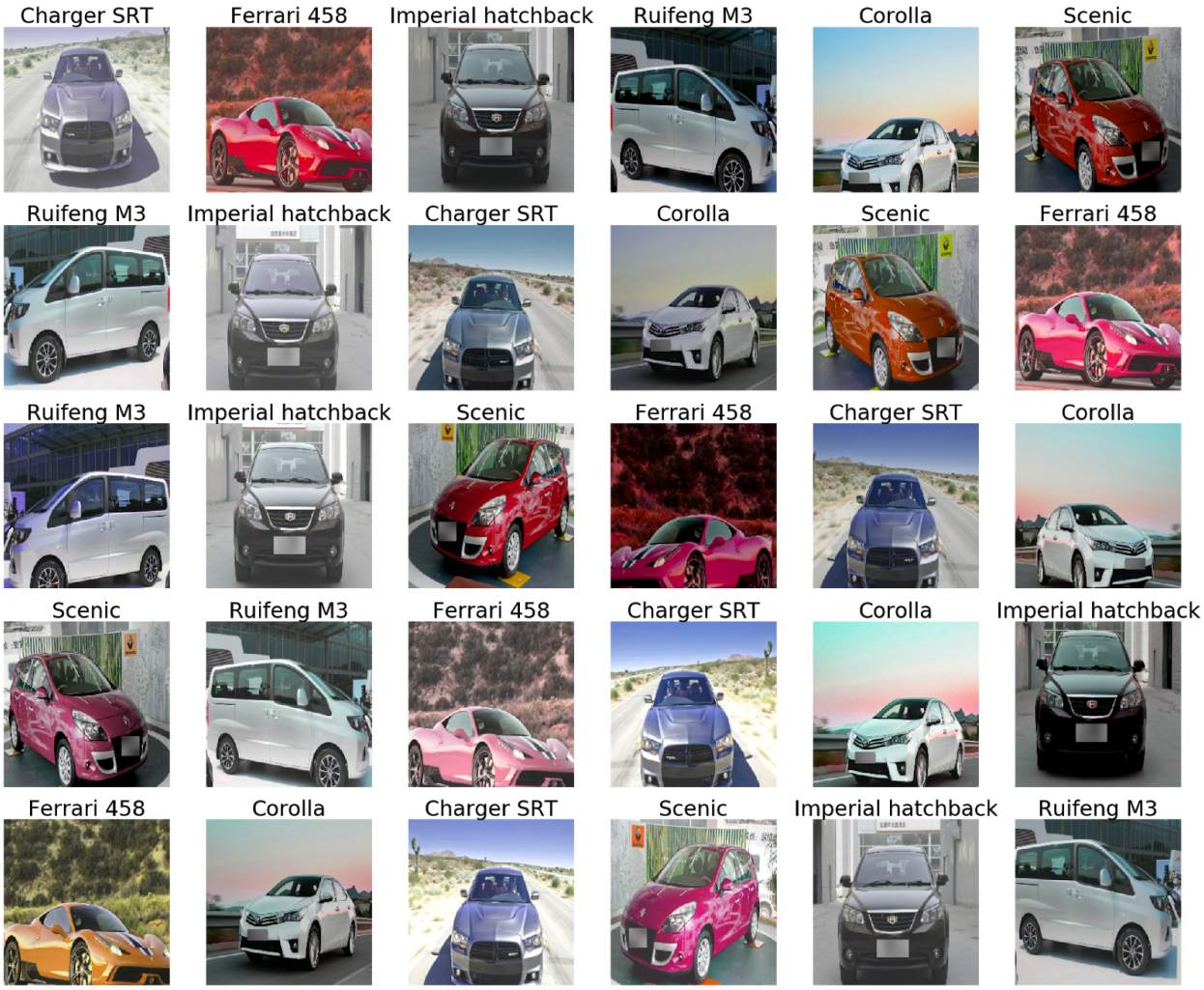


Each row of images display cars from different viewpoints

4.2. Preprocessing

The dataset is split into 3 subsets of Train:Test:Validation = 5:3:2. We used Tensorflow.dataset to load the images as it will carry out batch, shuffle, augmentation etc in an organized pipeline.

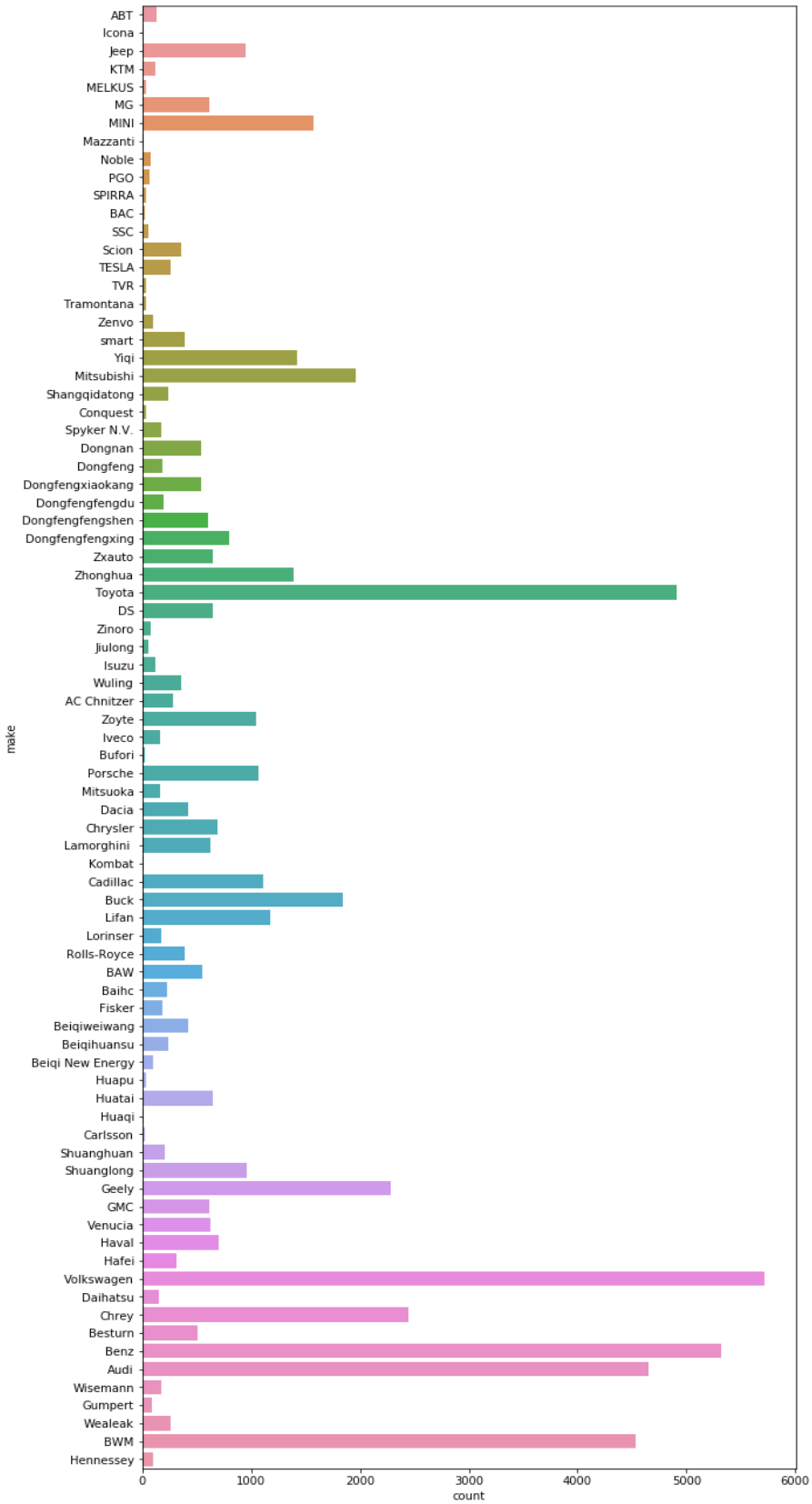
The images are then scaled down by a factor of 255 to transform every pixel value [0, 255] to [0, 1] to ensure images contribute more evenly to the total loss. The images will be resized to the required dimension: 224x224x3 for GoogLeNet and 299x299x3 for Inception_V3. Images will be augmented eg: flipped laterally, cropped, adjusted contrast etc. increase the training data. The datasets will be shuffled and batched before feeding into the networks.

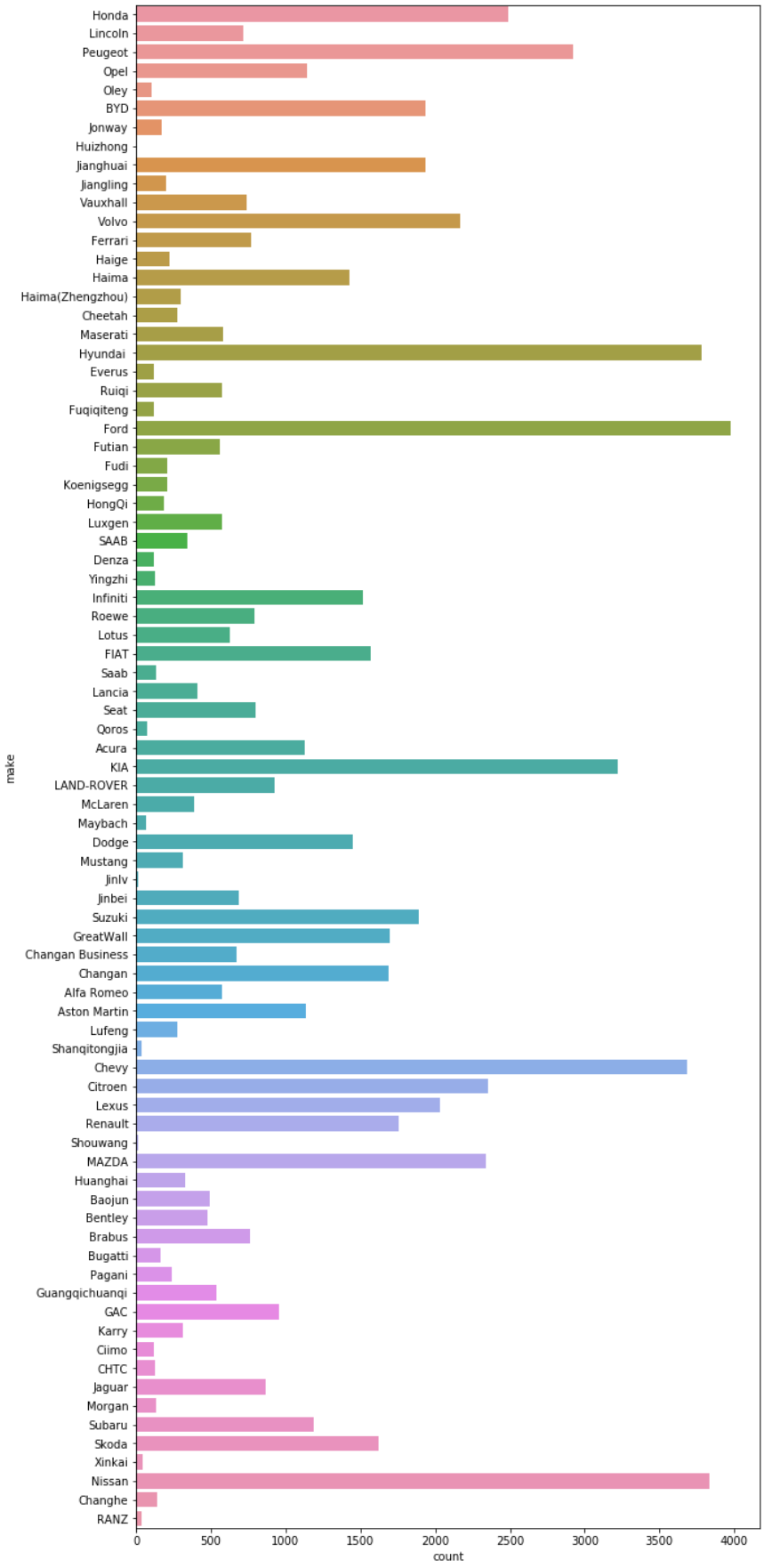


Images after augmentation

4.3. Exploration

The dataset is imbalanced with some car models having extremely high frequency. This might cause bias to the high frequency car models as it has higher appearance.





Countplot showing the imbalanced dataset

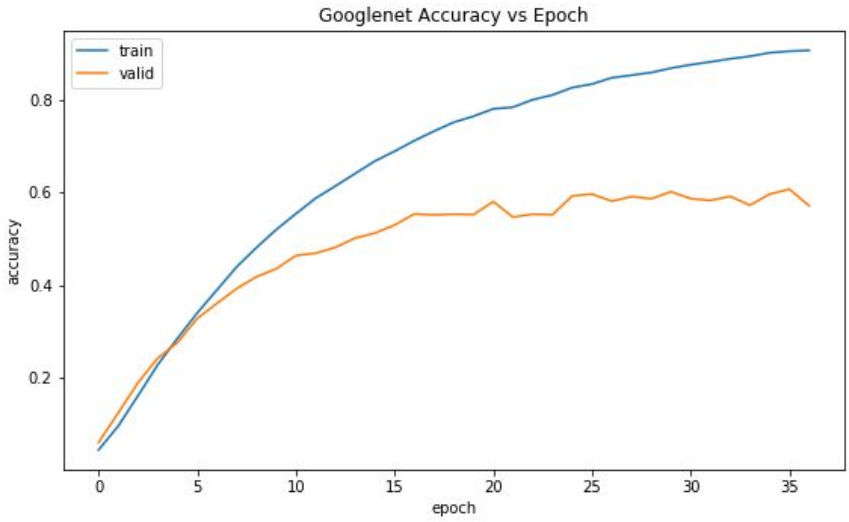
To tackle this issue, we use resampling technique by creating different datasets for each class and sample an equal amount of data from each dataset to form the new training datasets.

5. Experiment and Results

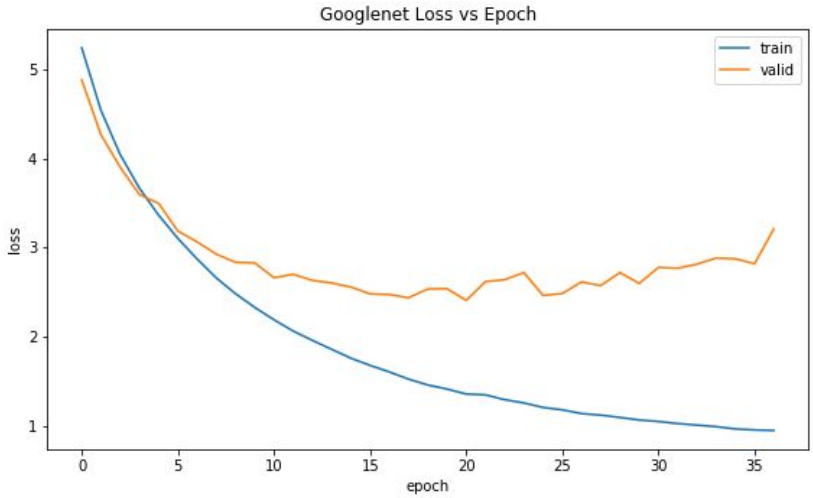
5.1. Improvement to published model on car makers classification

a. GoogLeNet to classify more classes

The paper *[3]* only exploits part of the datasets to classify cars into 75 makers. We take the whole datasets to perform the classification into 163 car makers. We use GoogLeNet but added a dense layer of 163 output to check the accuracy.



Graph showing accuracy against epoch of GoogLeNet



Graph showing loss against epoch of GoogLeNet

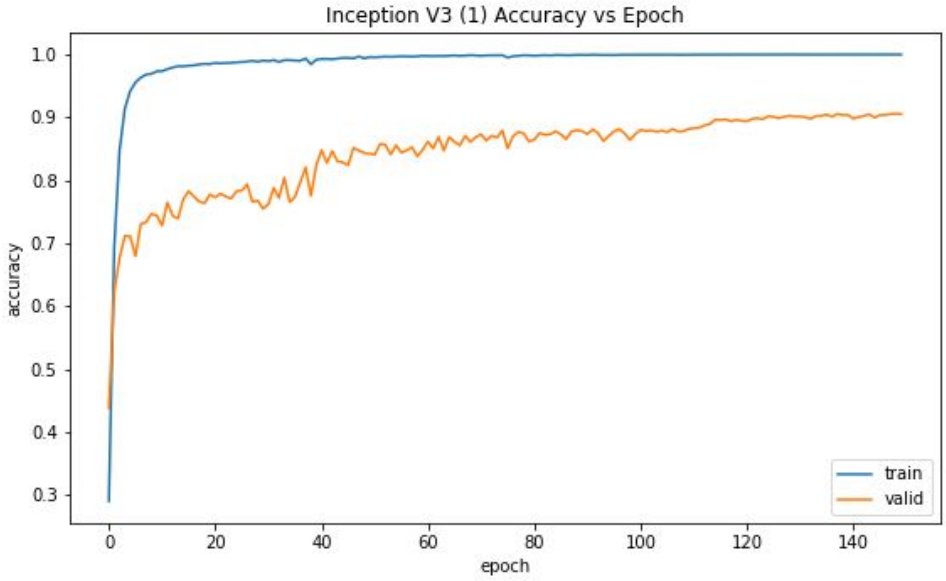
The final accuracy obtained is 0.628 which is lower than the one published(0.829). This might be caused by a greater number of classification classes.

b. Inception_V3

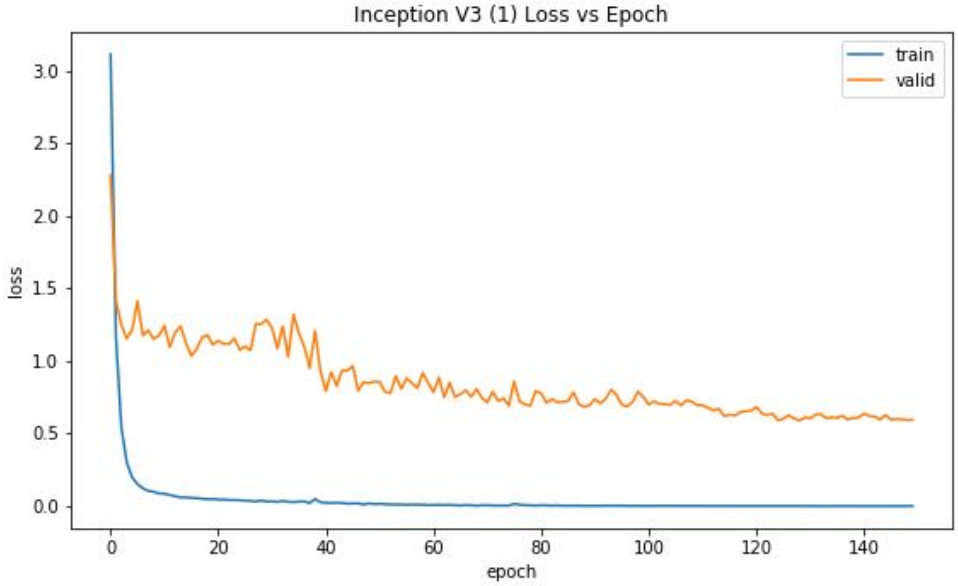
Inception_V3 has better performance in the computer vision field compared to GoogLeNet, we will use the basic Inception_V3 and other Inception_V3 with added layers to train the model. We will name the models as below for convenience:

Model	**Layers**
Inception V3(1)	Layer 1: Inception V3 Layer 2: Output Softmax Layer
Inception V3(2)	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: Output Softmax Layer
Inception V3(3)	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: 1000 Neurons Hidden Layer Layer 4: Output Softmax Layer
Inception V3(4)	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: 1000 Neurons Hidden Layer Layer 4: Dropout (0.2 drop rate) Layer 5: 1000 Neurons Hidden Layer Layer 6: Output Softmax Layer

i. Inception V3(1)

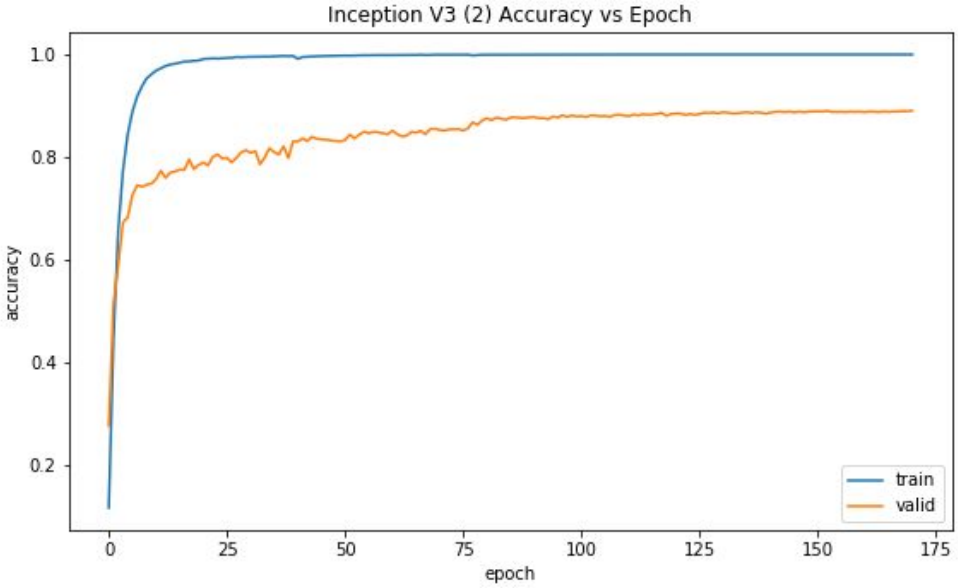


Graph showing accuracy against epoch of Inception V3(1)

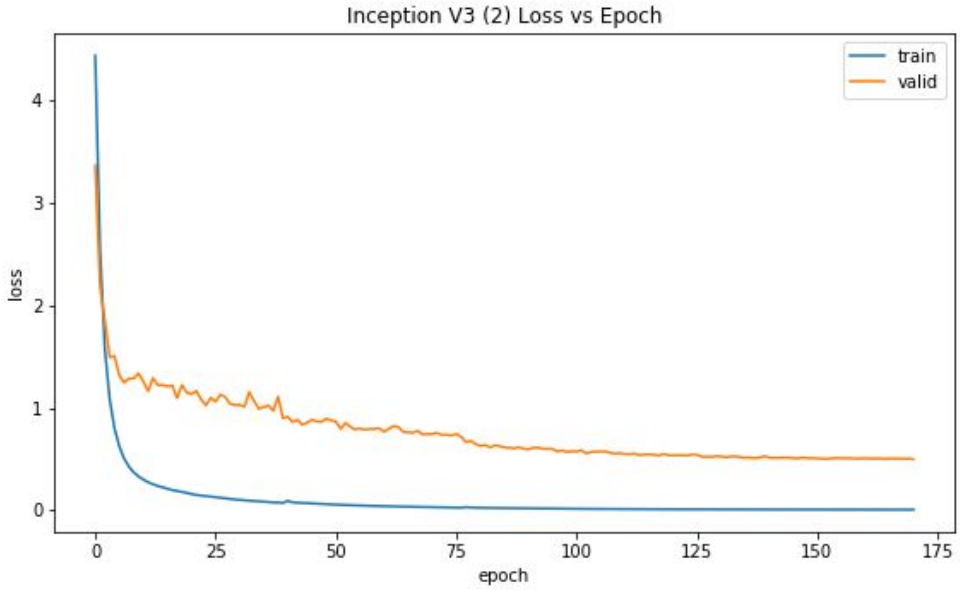


Graph showing loss against epoch of Inception V3(1)

ii. Inception V3(2)

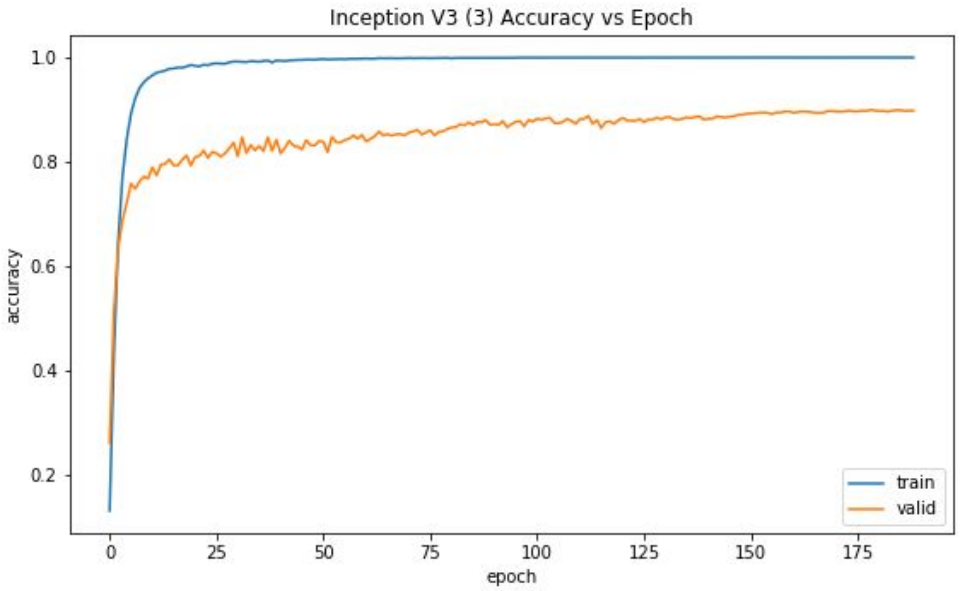


Graph showing accuracy against epoch of Inception V3(2)

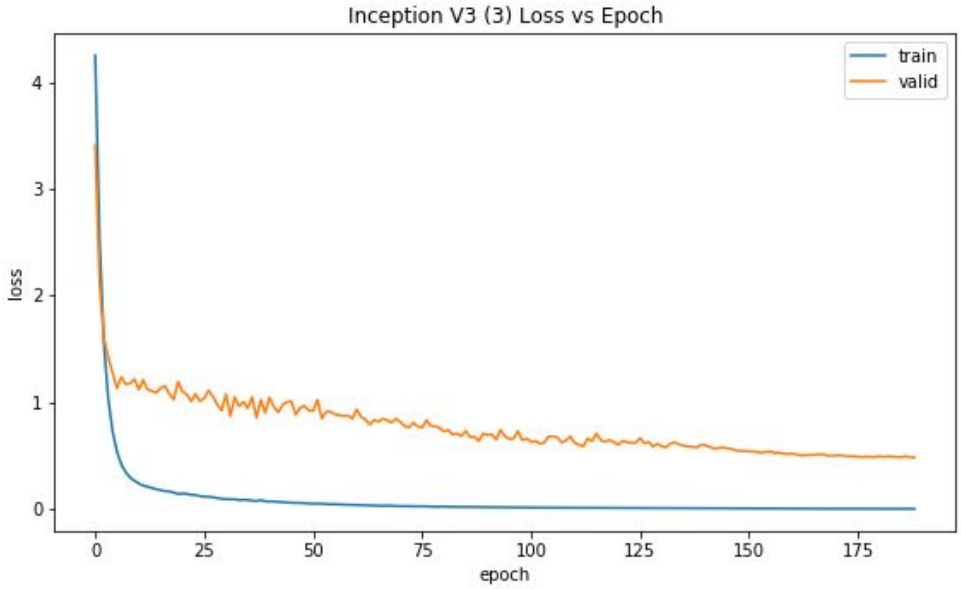


Graph showing loss against epoch of Inception V3(2)

iii. Inception V3(3)

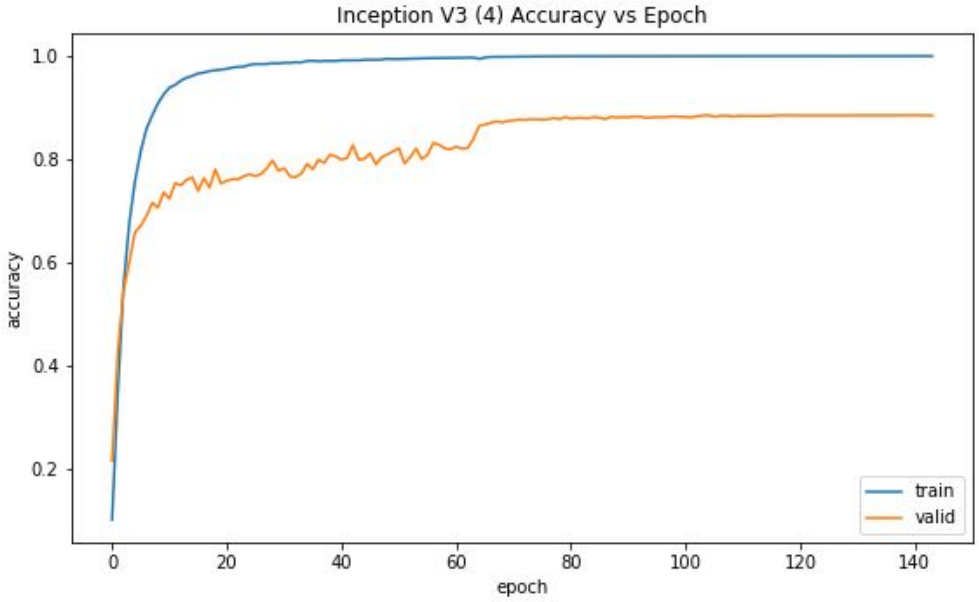


Graph showing accuracy against epoch of Inception V3(3)



Graph showing loss against epoch of Inception V3(3)

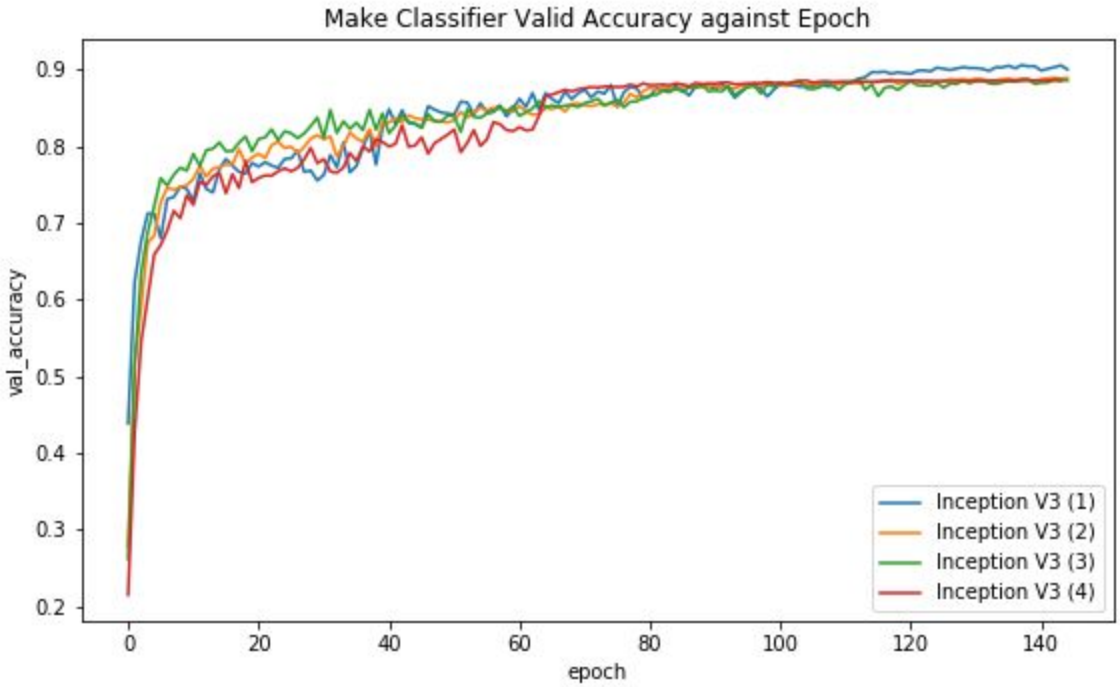
iv. Inception V3(4)



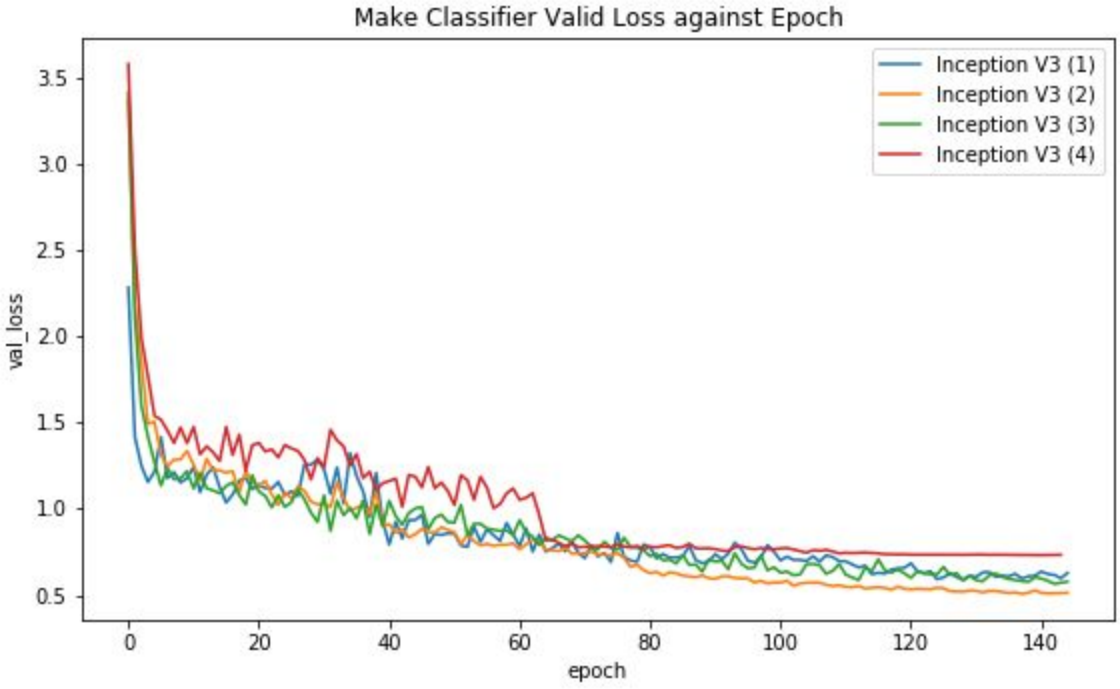
Graph showing accuracy against epoch of Inception V3(4)

Graph showing loss against epoch of Inception V3(4)

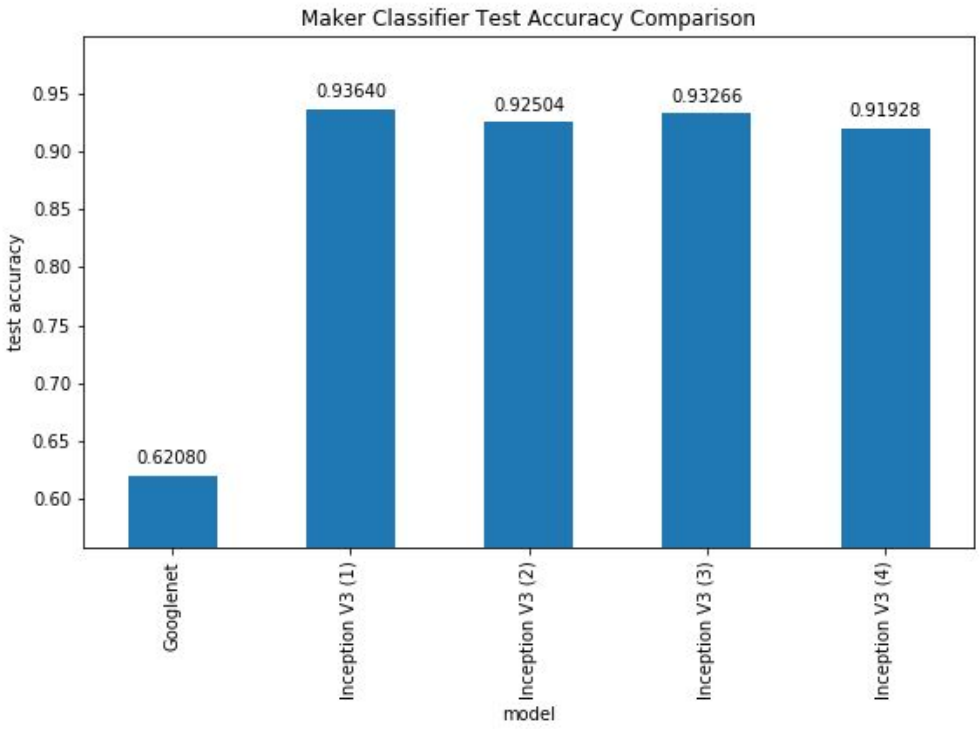
c. Comparison of different models



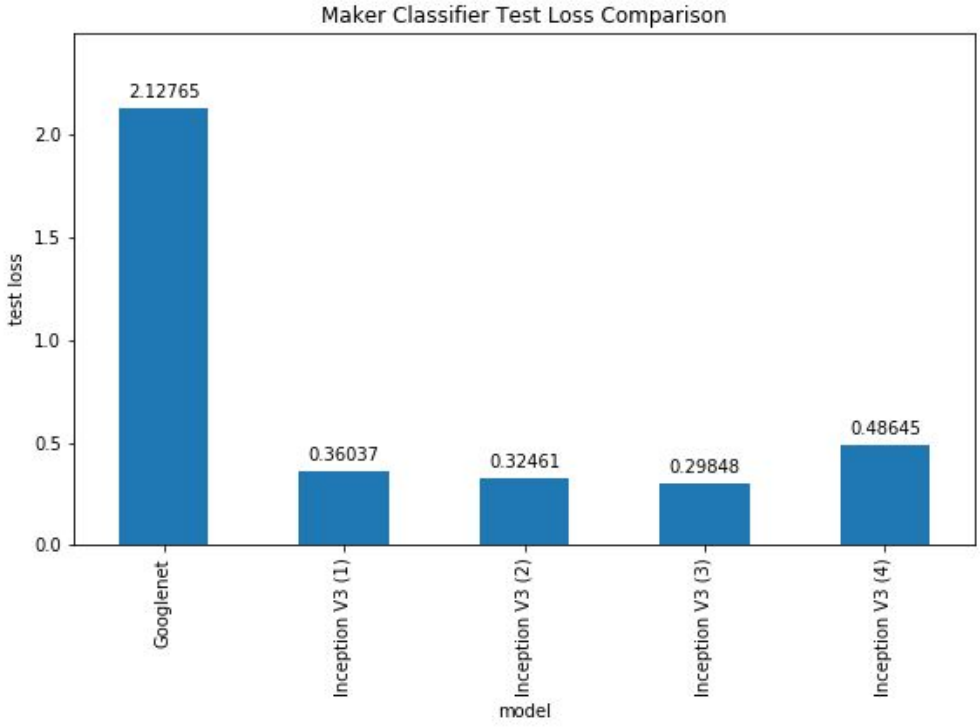
Graph of valid accuracies of different models



Graph of valid loss of different models



Barchart of test accuracies of different models



Barchart of test loss of different models

We can observe that Inception V3 performs significantly better than GoogLeNet in terms of test accuracies and loss. Inception V3 is deeper (42 layers) and it has more parameters than GoogLeNet, more features can be learnt from the network, resulting in higher accuracy.

Comparing Inception V3(1) with Inception V3(2), the dropout does not help in test accuracy but it results in lower test loss. Introducing hidden layers of 1000 neurons in Inception V3(3) improves the accuracy and lowers the loss compared to Inception V3(2).

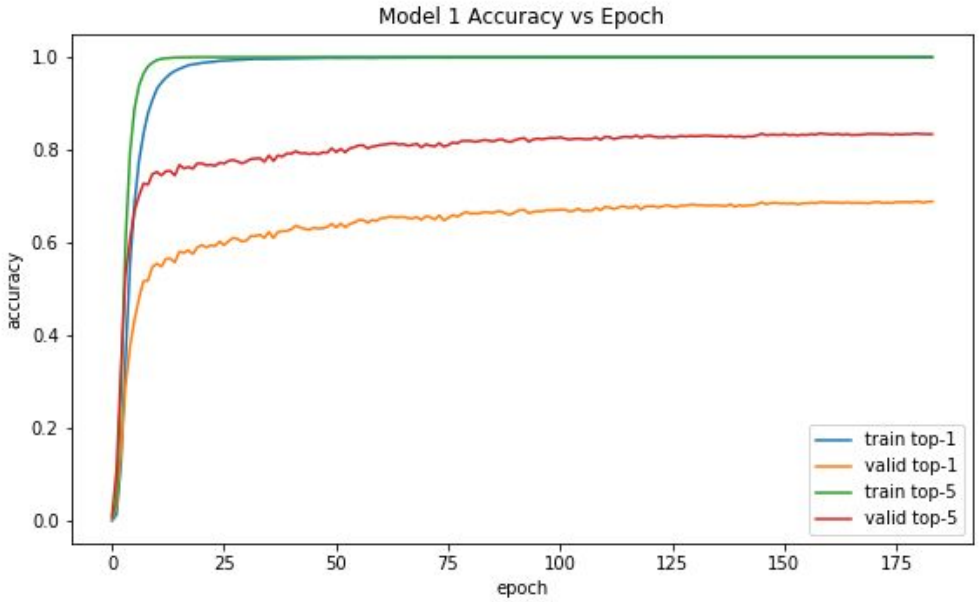
When 2 hidden layers of 1000 neurons are added in Inception V3(4), the accuracies and losses become worse which might be caused by overfitting.

5.2. Improvement to published model on car model classification

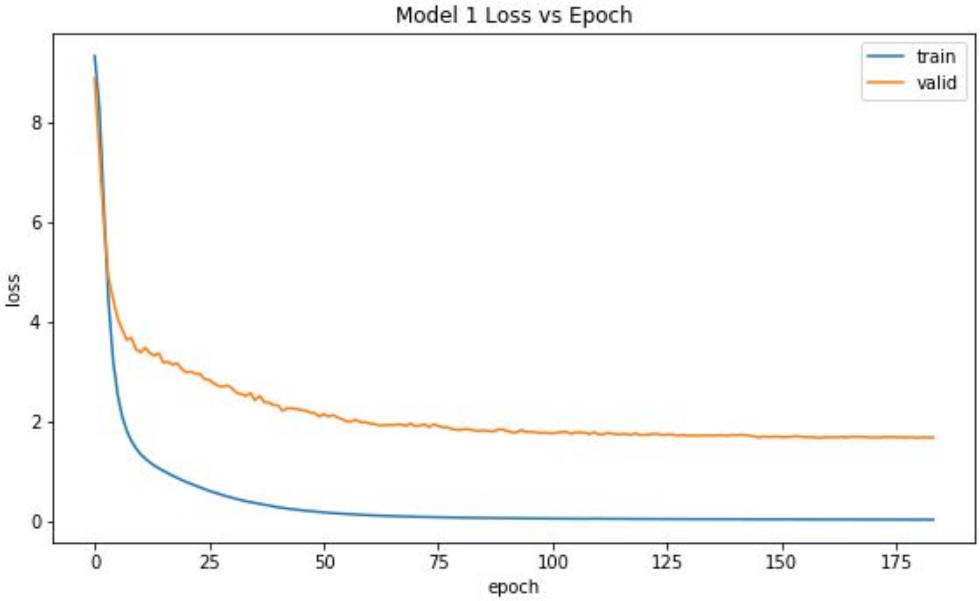
The original model in the paper classifies only 431 car models out of 1716 car models. Here, we will use Inception_V3 as the base model and fine tune the models' hyperparameters to find the best model to classify all 1716 car models. We will name the models as below for convenience:

Model	Layers	Optimizer
Model 1	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: Output Softmax Layer	Adam
Model 2	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: 1000 Neurons Hidden Layer (L2 Regularization 0.001) Layer 4: Output Softmax Layer	Adam
Model 3	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: 2000 Neurons Hidden Layer (L2 Regularization 0.001) Layer 4: Output Softmax Layer	Adam
Model 4	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: Output Softmax Layer	RMSprop
Model 5	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: 1000 Neurons Hidden Layer (L2 Regularization 0.001) Layer 4: Output Softmax Layer	RMSprop
Model 6	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: 2000 Neurons Hidden Layer (L2 Regularization 0.001) Layer 4: Output Softmax Layer	RMSprop
Model 7	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: 2000 Neurons Hidden Layer (No Regularization) Layer 4: Output Softmax Layer	Adam
Model 8	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: 2000 Neurons Hidden Layer (L2 Regularization 0.001) Layer 4: Dropout (0.2 drop rate) Layer 5: 2000 Neurons Hidden Layer (L2 Regularization 0.001) Layer 6: Output Softmax Layer	Adam
Model 9	Layer 1: Inception V3 Layer 2: Dropout (0.2 drop rate) Layer 3: Output Softmax Layer	SGD

i. Model 1

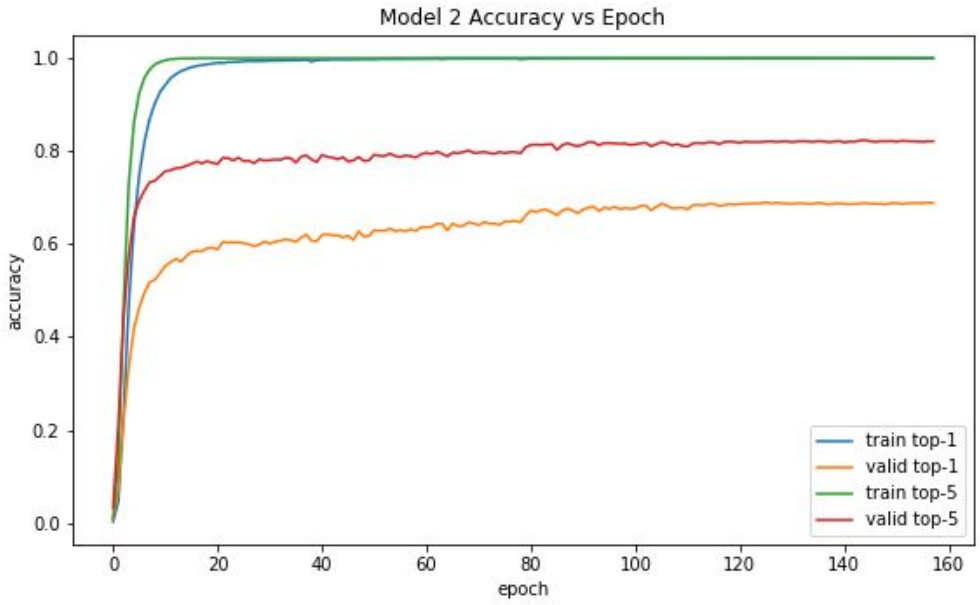


Graph showing top-1 and top-5 train and valid accuracies against epoch of Model 1

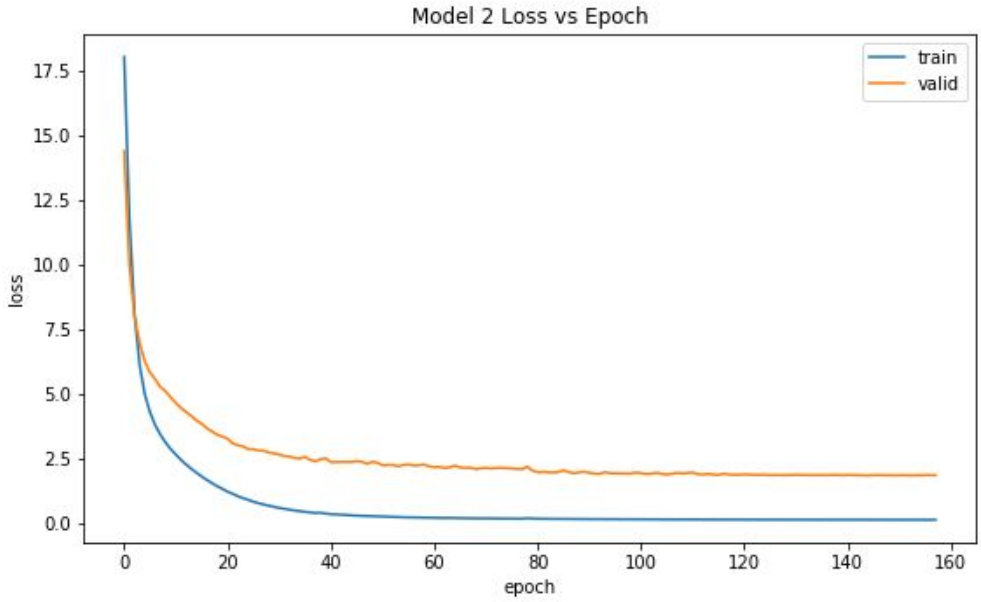


Graph showing train and valid loss against epoch of Model 1

ii. Model 2

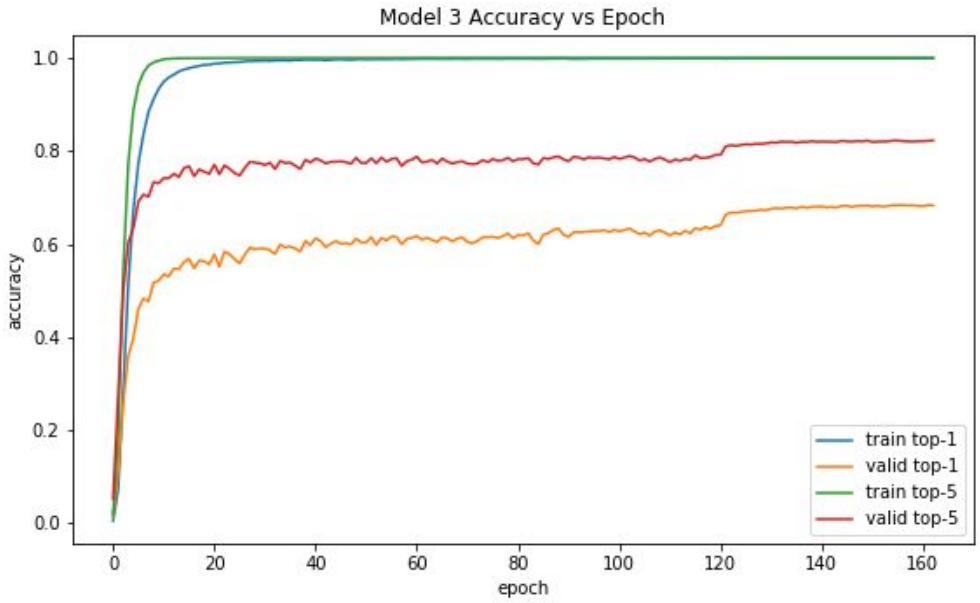


Graph showing top-1 and top-5 train and valid accuracies against epoch of Model 2

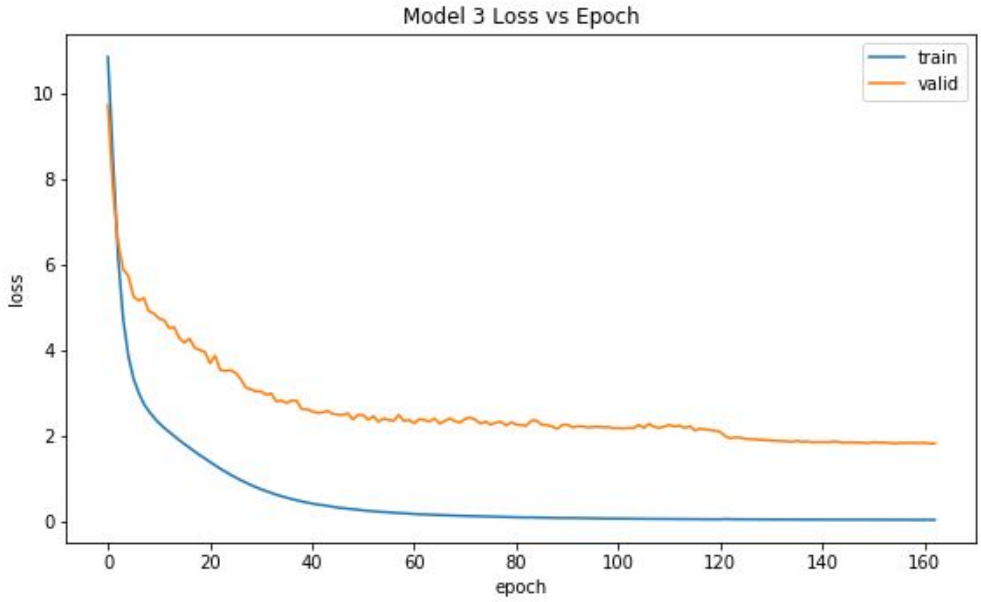


Graph showing train and valid losses against epoch of Model 2

iii. Model 3

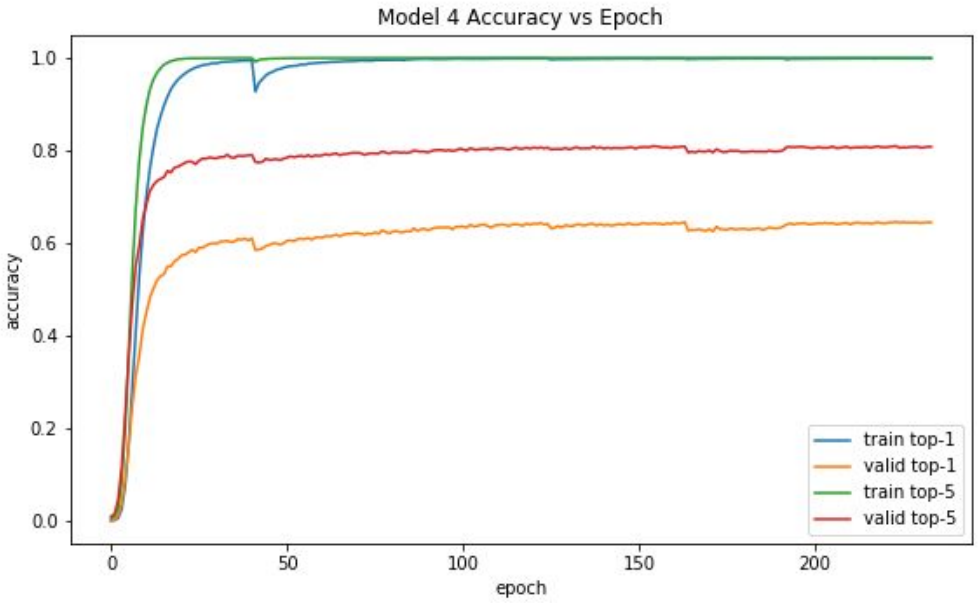


Graph showing top-1 and top-5 train and valid accuracies against epoch of Model 3

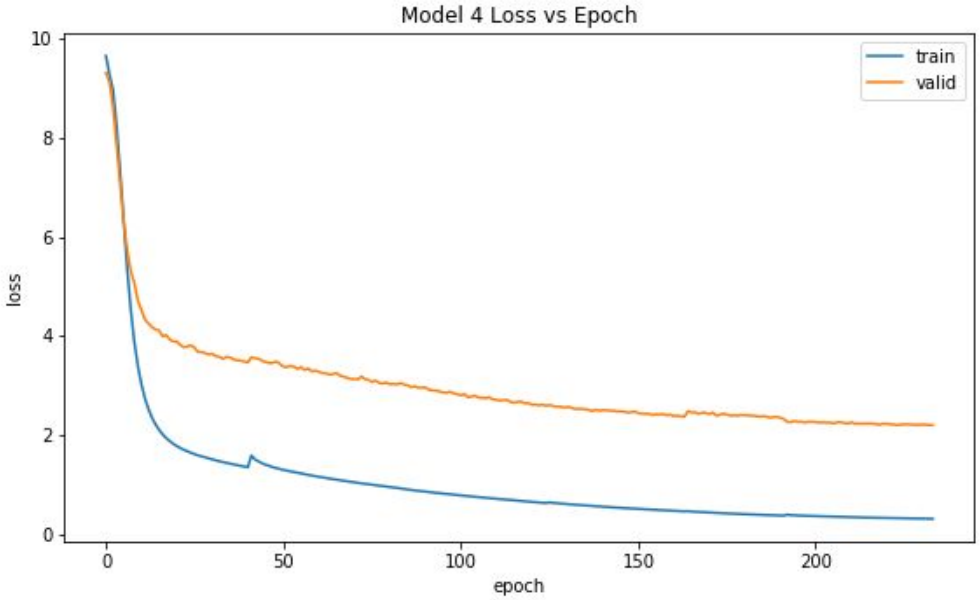


Graph showing train and valid losses against epoch of Model 3

iv. Model 4

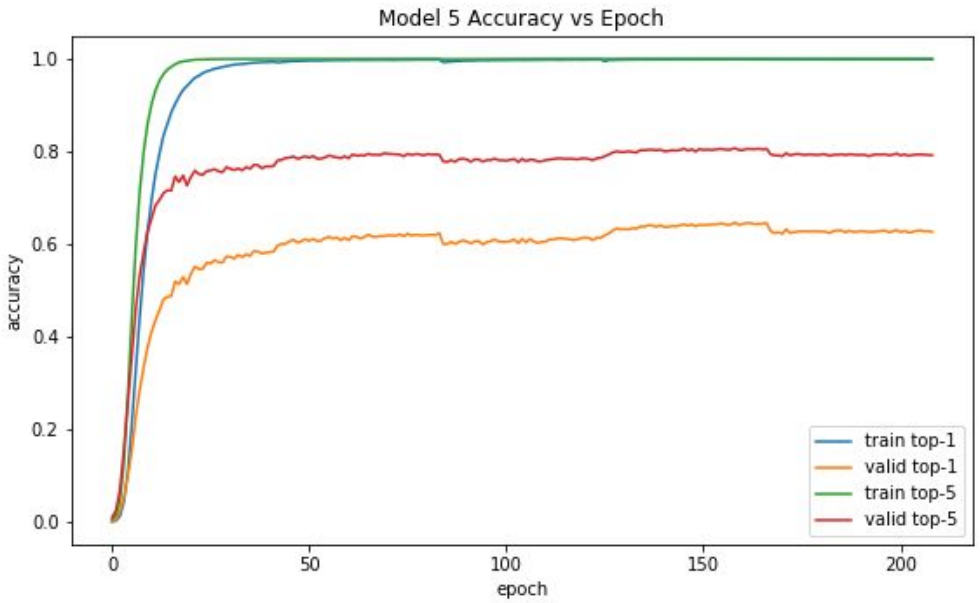


Graph showing top-1 and top-5 train and valid accuracies against epoch of Model 4

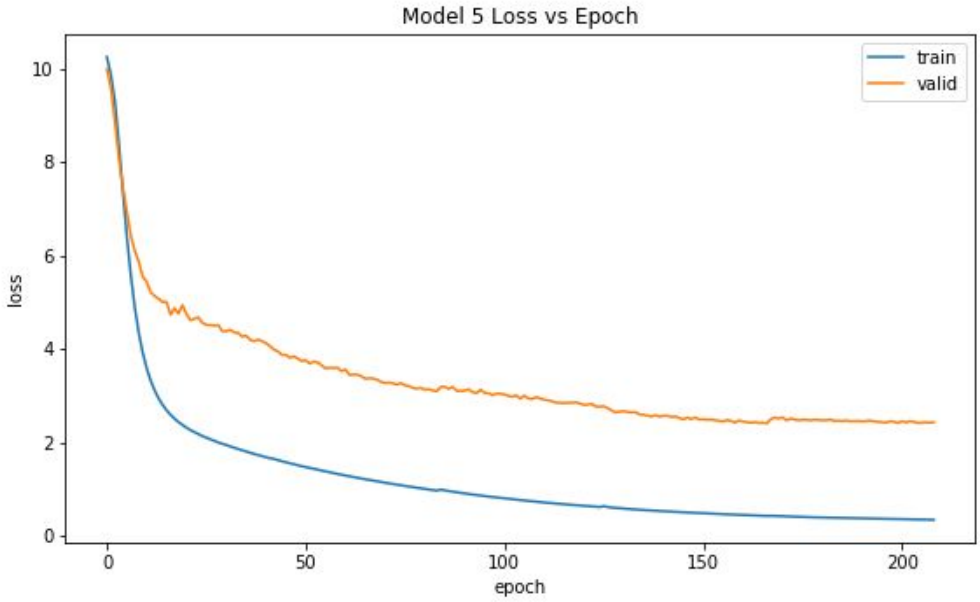


Graph showing train and valid losses against epoch of Model 4

v. Model 5

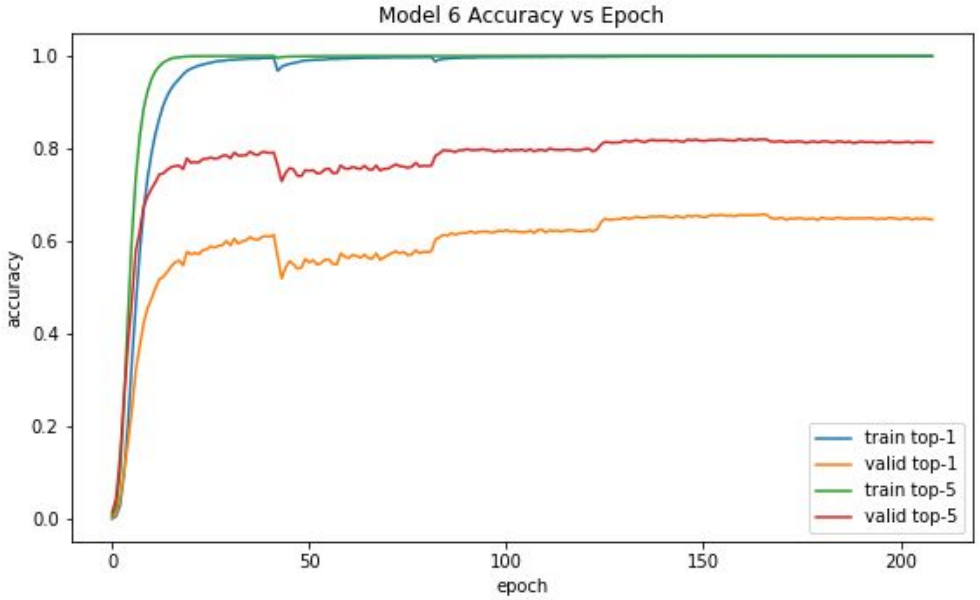


Graph showing top-1 and top-5 train and valid accuracies against epoch of Model 5

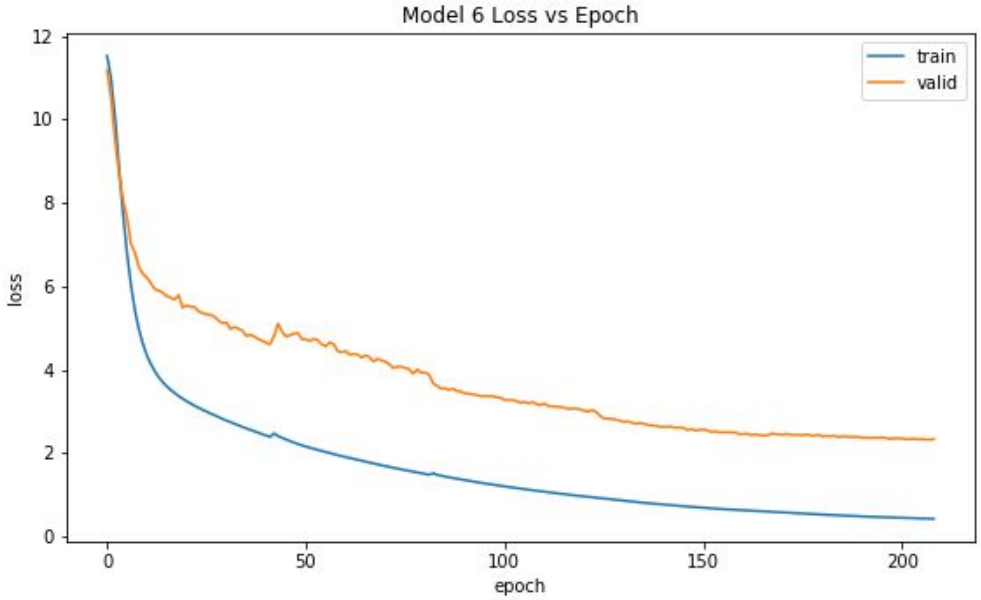


Graph showing train and valid losses against epoch of Model 5

vi. Model 6

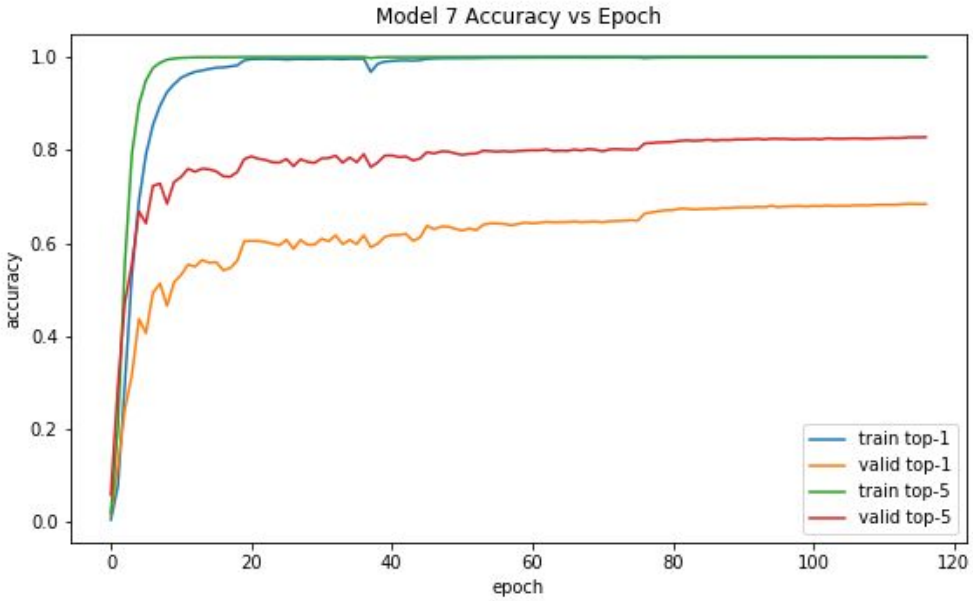


Graph showing top-1 and top-5 train and test accuracies against epoch of Model 6

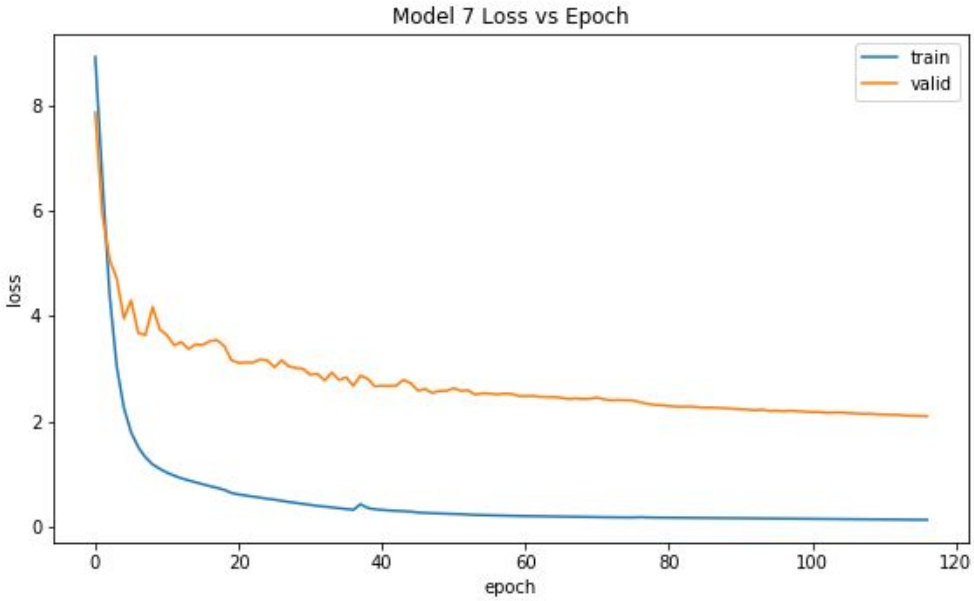


Graph showing train and valid losses against epoch of Model 6

vii. Model 7

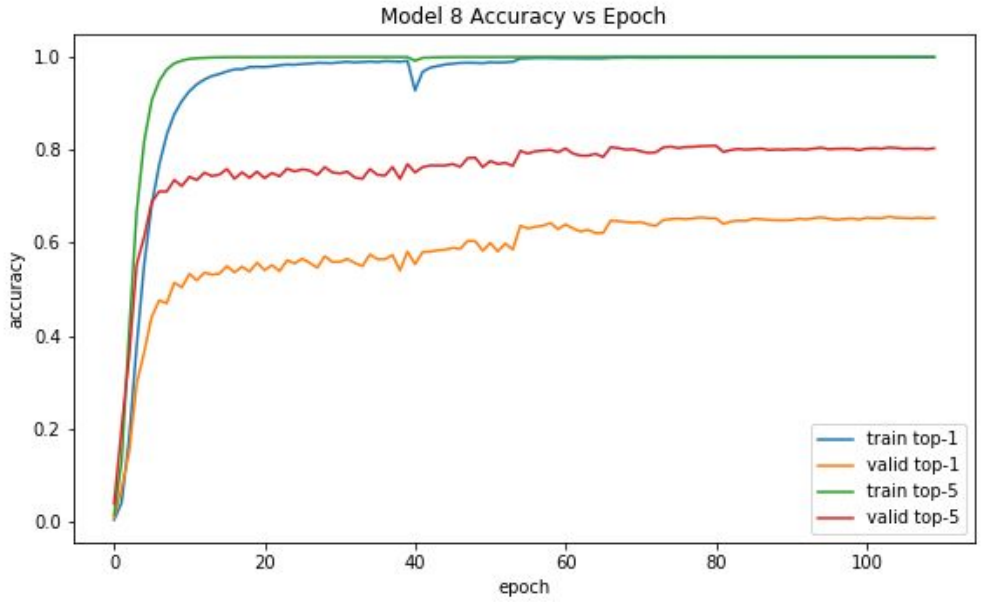


Graph showing top-1 and top-5 train and test accuracies against epoch of Model 7

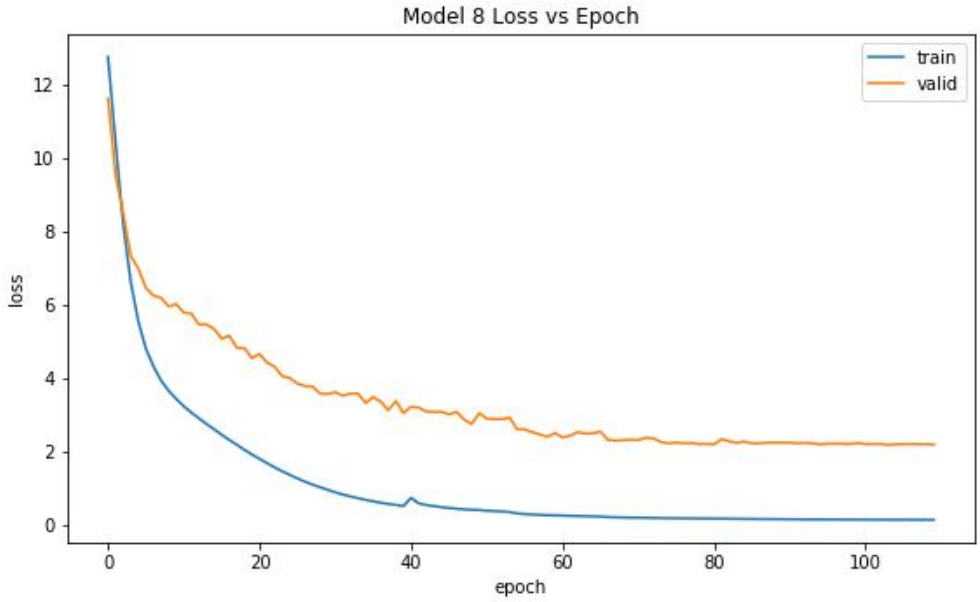


Graph showing train and valid losses against epoch of Model 7

viii. Model 8

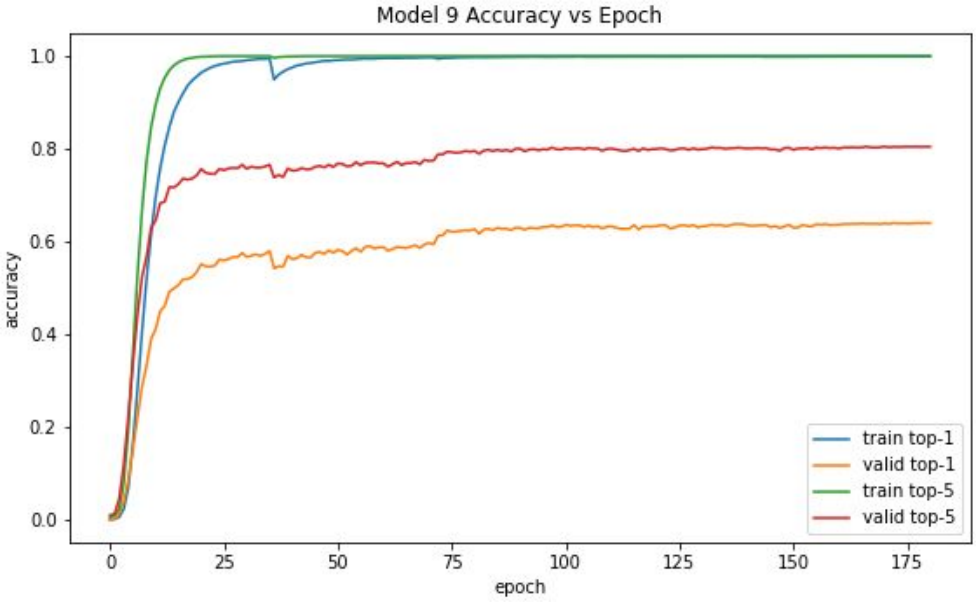


Graph showing top-1 and top-5 train and test accuracies against epoch of Model 8

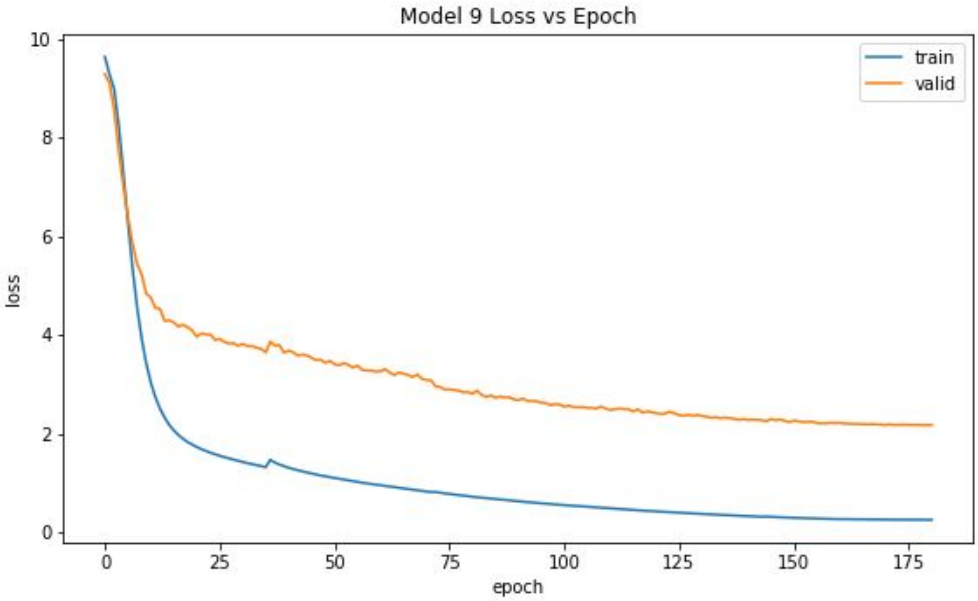


Graph showing train and valid losses against epoch of Model 8

ix. Model 9



Graph showing top-1 and top-5 train and test accuracies against epoch of Model 9

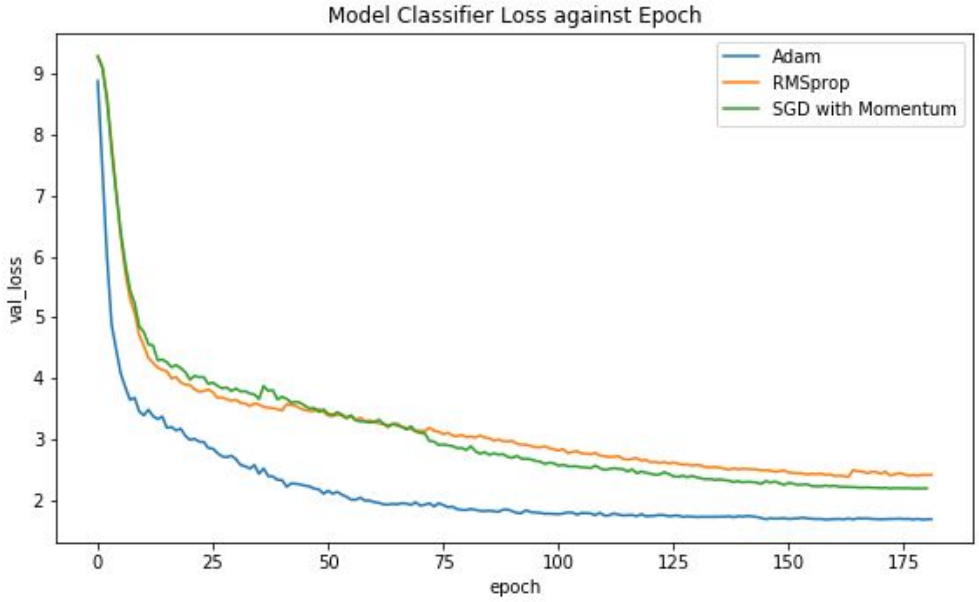


Graph showing train and valid losses against epoch of Model 9

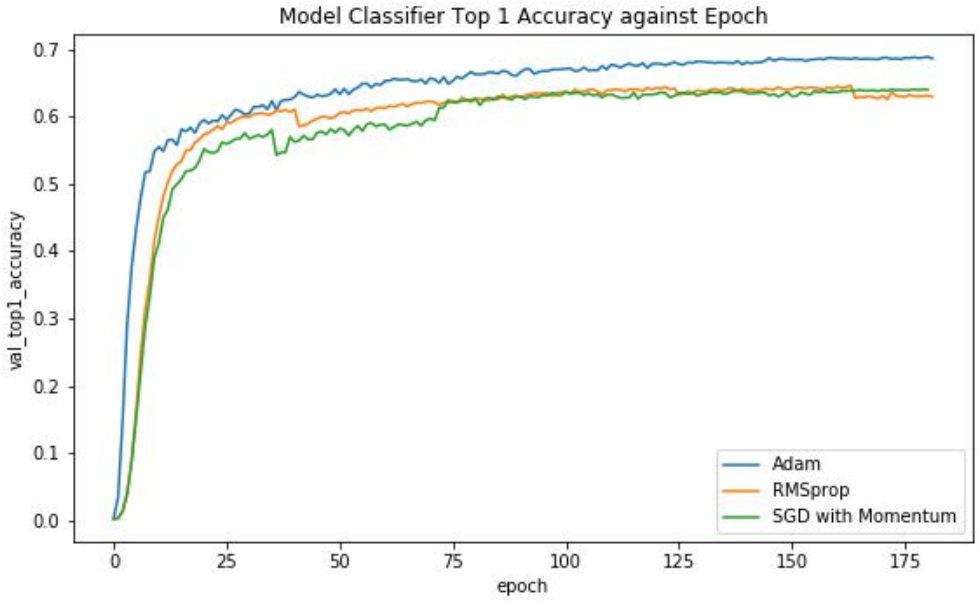
Comparison of different models

Optimizer

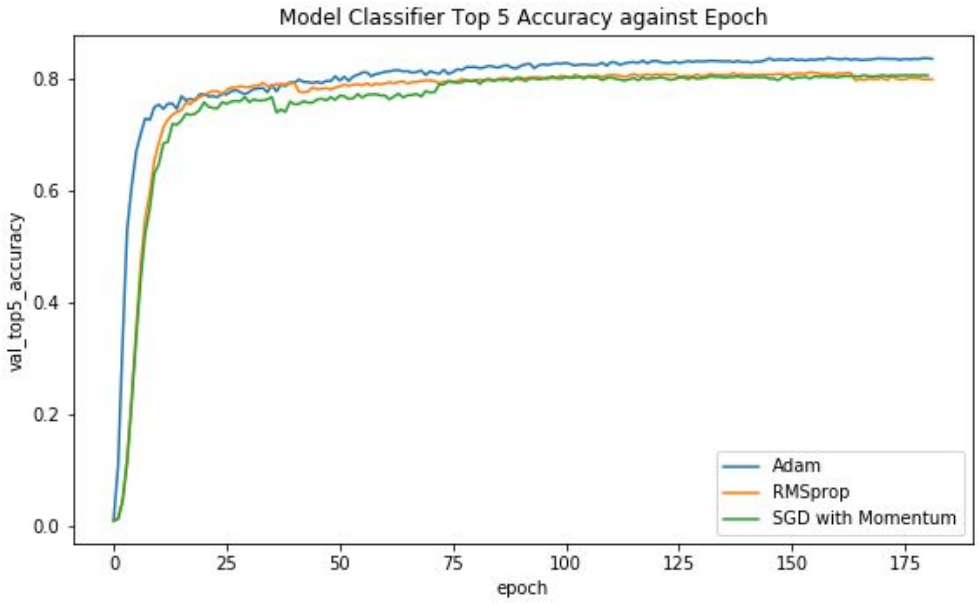
We have tried out different optimizers, which are Adam, RMSProp and SGD with Momentum for the same model architecture (Model 1, Model 4 and Model 7). Among these optimizers, we can observe that Model 1 using Adam optimizer is the best among all as it yields the lowest validation loss and highest validation top-1 and top-5 accuracy.



Graph showing model classifier valid loss against epoch for different optimizers



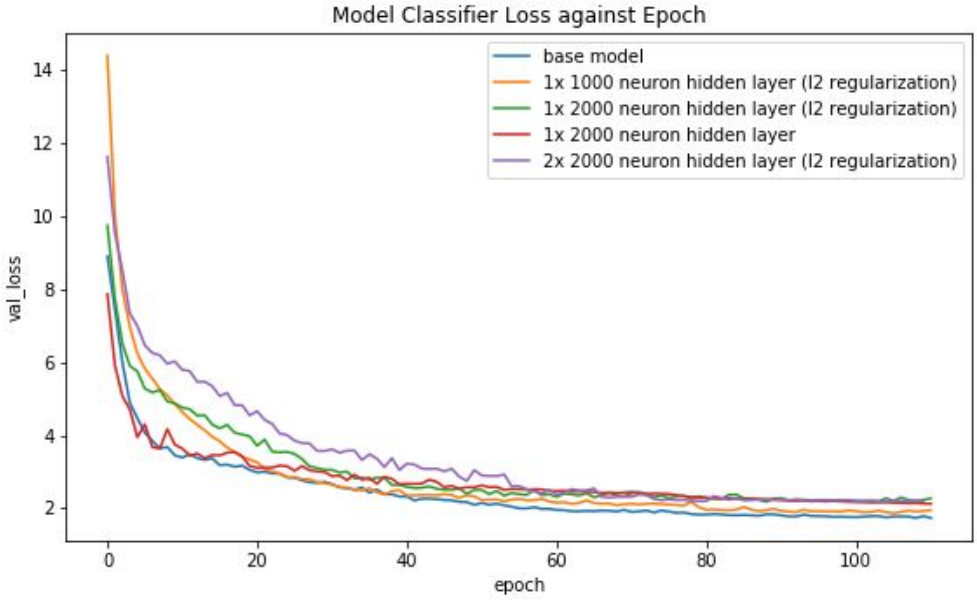
Graph showing model classifier valid top-1 accuracy against epoch for different optimizers



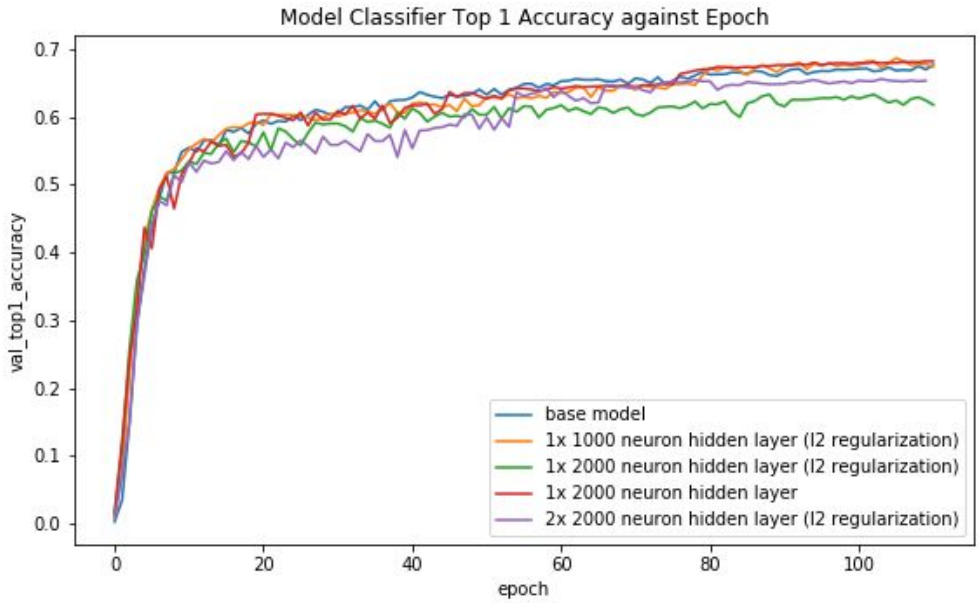
Graph showing model classifier valid top-5 accuracy against epoch for different optimizers

Model Architecture

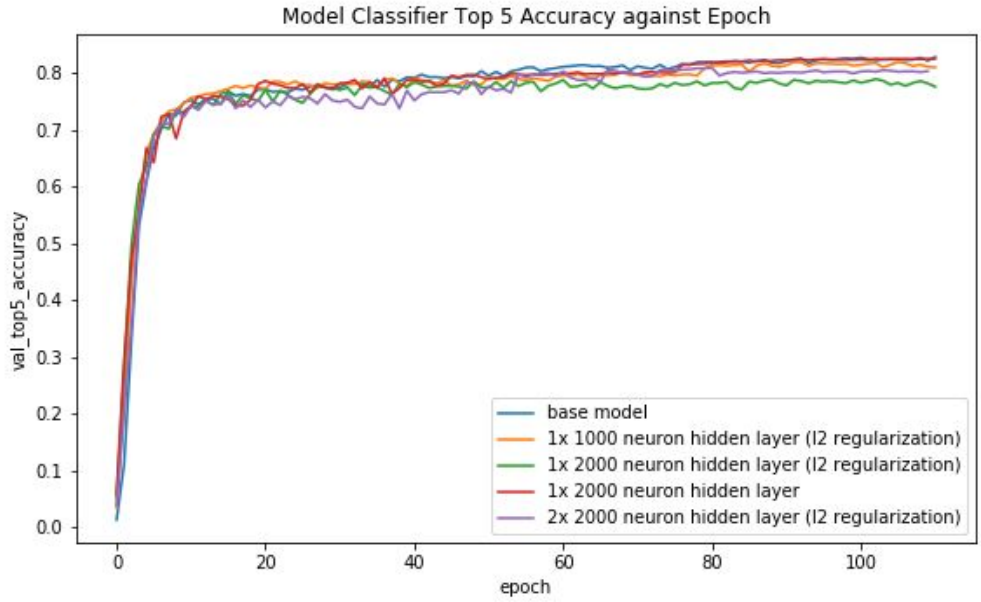
Different models (Model 1, Model 2, Model 3, Model 5 and Model 7) are trained using Adam optimizer and compared below. The base model (Model 1) manages to achieve the lowest validation loss, followed by Model 2 with an additional 1000 neuron hidden layer using L2 regularization with decay rate of 0.001. In terms of validation top-1 and top-5 accuracies, we can see that the Model 1, Model 4 and Model 7 achieves similar and the best results among all models.



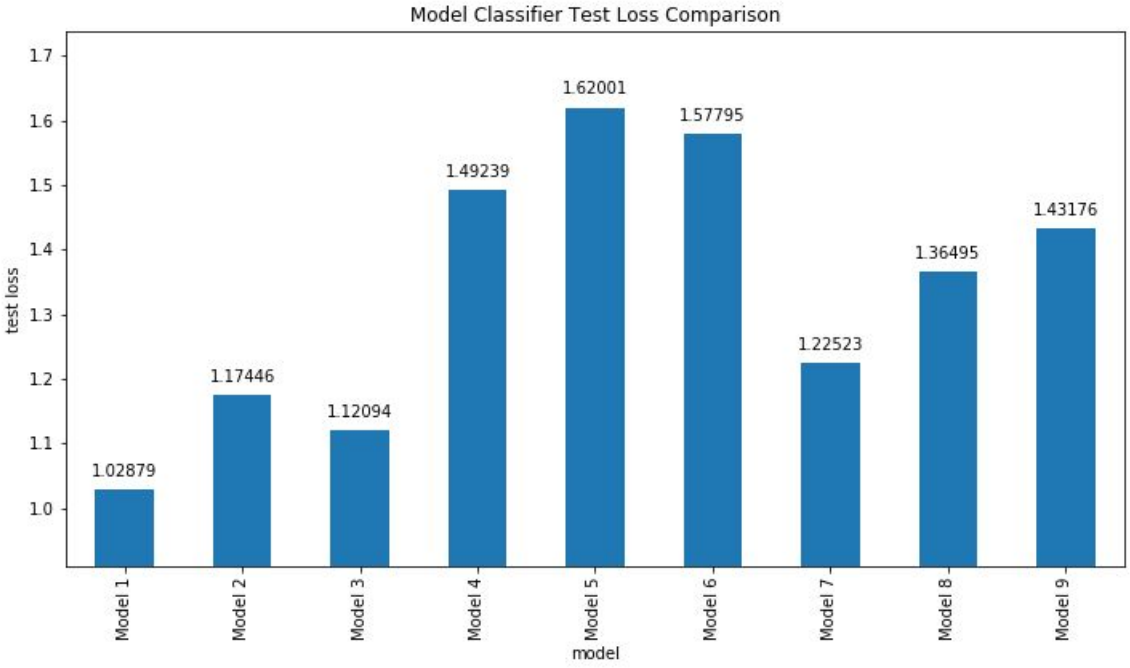
Graph showing model classifier valid loss against epoch for different model architectures



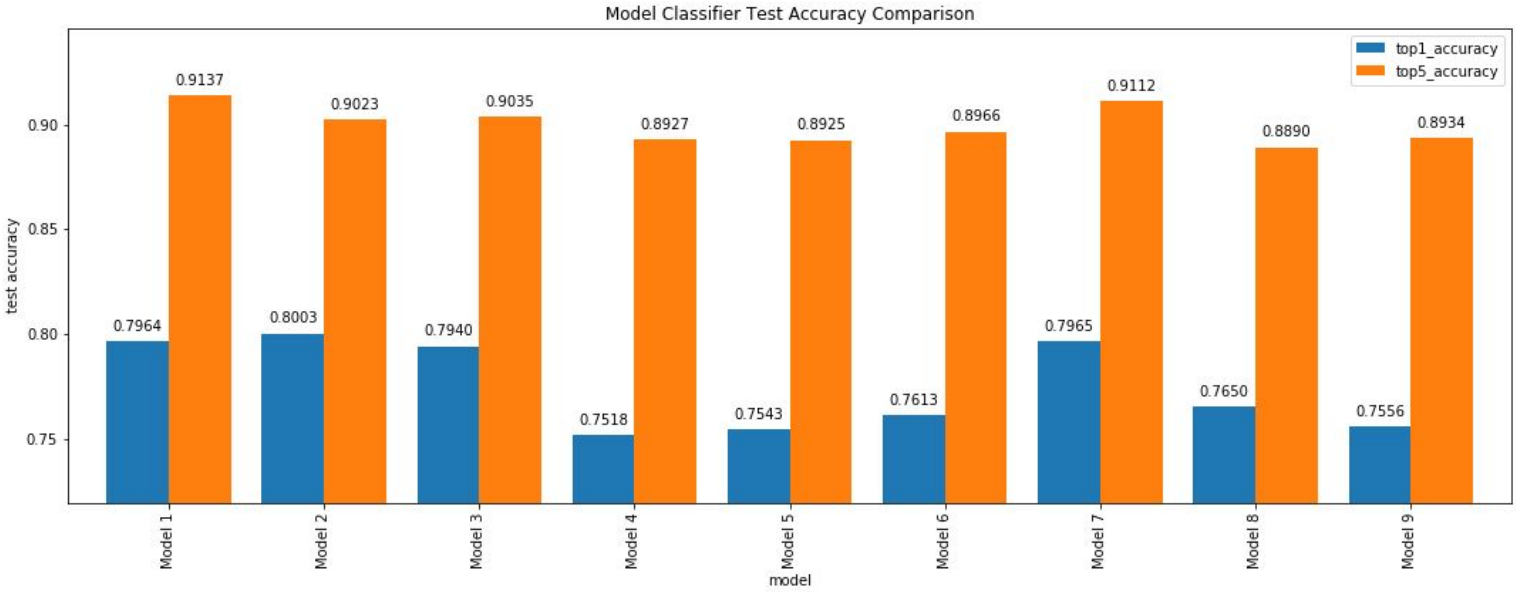
Graph showing model classifier valid top-1 accuracies against epoch for different model architectures



Graph showing model classifier valid top-5 accuracies against epoch for different model architectures



Barchart of test loss of different car model classifier



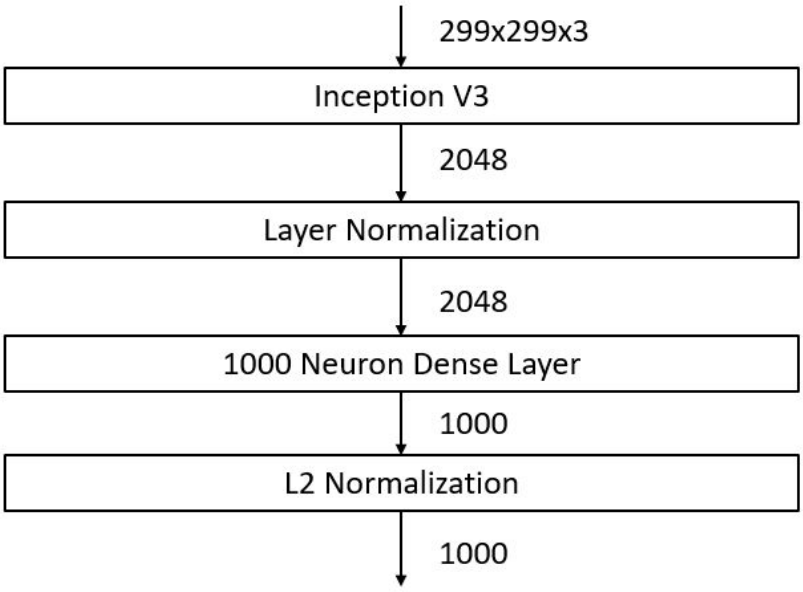
Barchart of test accuracies of different car model classifier

The models are then tested with the test dataset and the results are as shown above. We can observe that the base model (Model 1) manages to achieve a significantly low test loss compared to other models. This means that the original model is able to generalize well enough. Adding additional layers, even with the help of dropout and L2 regularization, leads to overfitting which does not help improving the model.

Therefore, we can conclude that the best model for car model classification is Model 1.

5.3 Triplet Loss and Car Verification Model

In this section, we use triplet loss to train a model for feature extraction. The model takes in the image as input and output an embedding with a size of 1000 for that image. The embedding represents the features extracted from the image. The embeddings will be L2 normalized so that all embeddings are projected onto the surface of a hypersphere at R^{1000} with a radius of 1 unit. This is to minimize the distance between embeddings and speed up the training. The architecture of the model is shown below:



Model layers for feature extraction

The goal of triplet loss is to train the embedding such that cars with the same car maker will be projected as close as possible in the embedding space, whereas cars with different car maker will be projected as far as possible in the embedding space. The value of margin (α) use for training is 0.5.

Let the distance between baseline and positive to be $d(a,p)$ and the distance between baseline and negative to be $d(a,n)$. There are three kinds of triplets, which are easy triplets, hard triplets and semihard triplets. When calculating triplet loss, we replace all negative losses (easy triplets) with 0 so as to let the training focus more on the semihard triplets and hard triplets. *[8][9]*

- **Easy triplets**: $d(a,p) + \alpha < d(a,n)$
- **Hard triplets**: $d(a,n) < d(a,p)$
- **Semihard triplets**: $d(a,p) < d(a,n) < d(a,p) + \alpha$

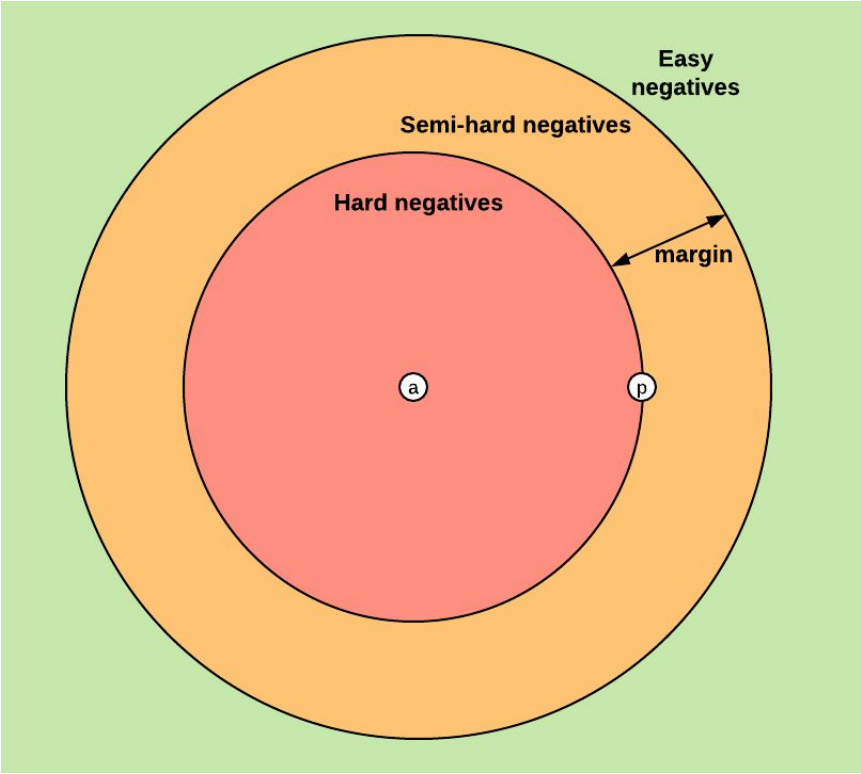


Image from https://omoindrot.github.io/triplet-loss

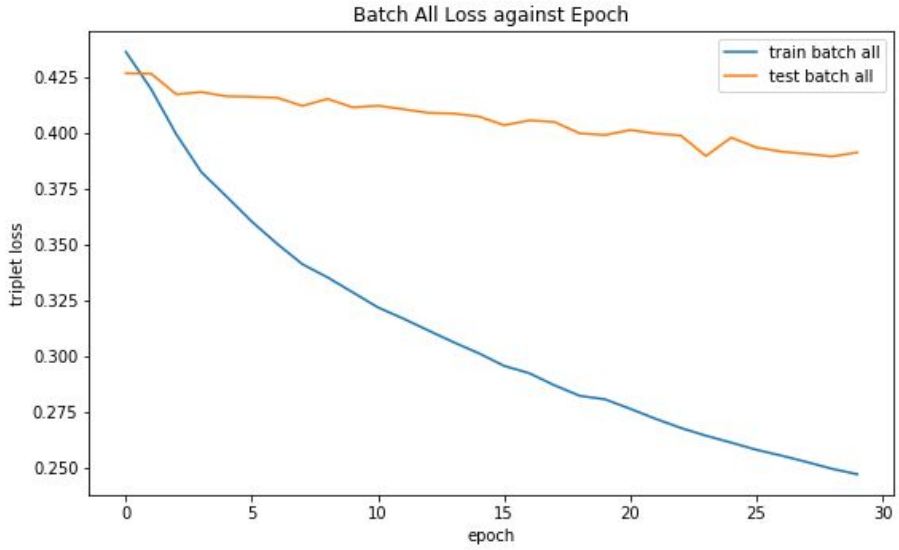
Here, we use online triplet mining whereby we compute useful triplets for each batch of inputs instead of producing the triplets before the training. In order to increase the possibility for each make class to have a valid triplet, we use a resampled dataset to yield batches of images with equal amount of images for each class. There are three kinds of online triplet mining strategies, which are batch all, batch hard and batch semihard.

- **Batch all**: In the batch of images, find all possible triplets and take the mean of their triplet losses.
- **Batch hard**: In the batch of images, make each image the baseline and find the triplet that yields the maximum triplet loss (hardest triplet) corresponding to the baseline, then take the mean of the maximum triplet loss across all baseline.
- **Batch semihard**: In the batch of images, make each image the baseline and find the semihard triplet that yields the maximum triplet loss corresponding to the baseline. If there is no semihard triplet for any baseline, then the hardest triplet loss is use.

When training, we use one type of the strategy as the loss function and the other as the metric to compare which online triplet mining strategy is the best among all.

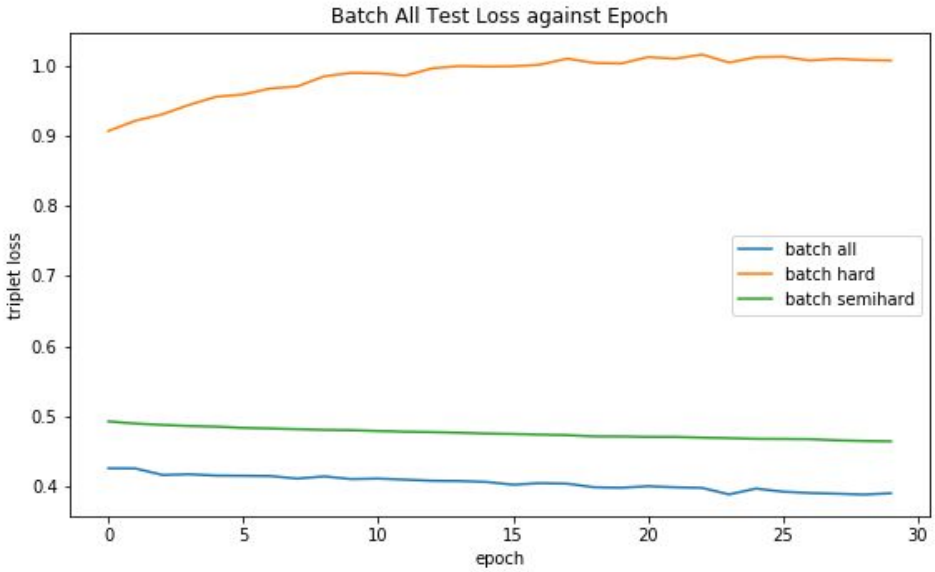
Batch All Strategy

Firstly, we use the batch all strategy as the loss function and use the batch hard and batch semihard strategy as the metrics. We can observe that the batch all triplet loss for training decreases drastically but the batch all triplet loss for testing decreases slightly only.



Graph showing training and testing batch all triplet loss against epoch

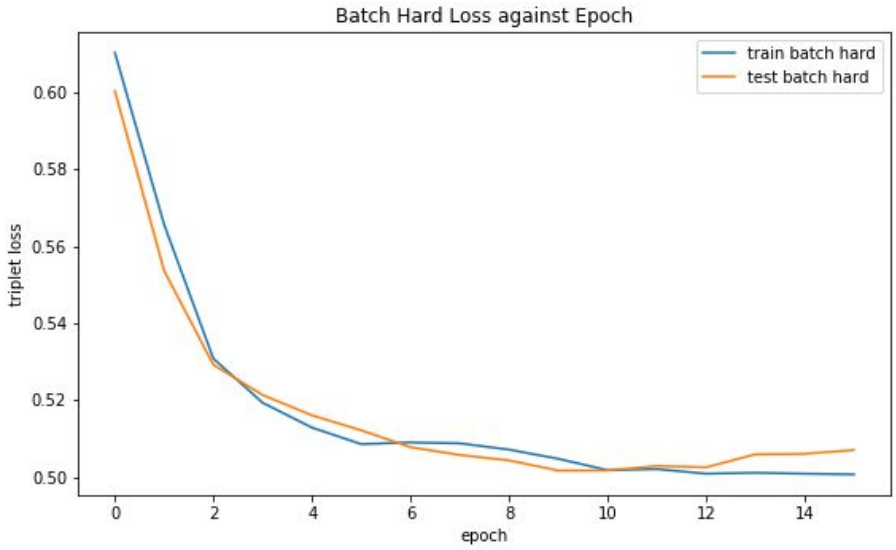
We can also observe that the testing metrics, batch semihard triplet loss decreases slightly whereas the batch hard triplet loss is increasing. The model shows an overfitting trend as the training loss decreases at a higher rate compared to the testing loss. Besides, the model does not train itself to minimize the triplet loss for outliers (hard triplets) but keep training itself to minimize the loss for easier triplets, this leads to overfitting on easier triplets and increasing loss for harder triplets. Therefore, the testing loss decreases at a slower rate.



Graph showing the testing batch all, batch hard and batch semihard triplet loss against epoch

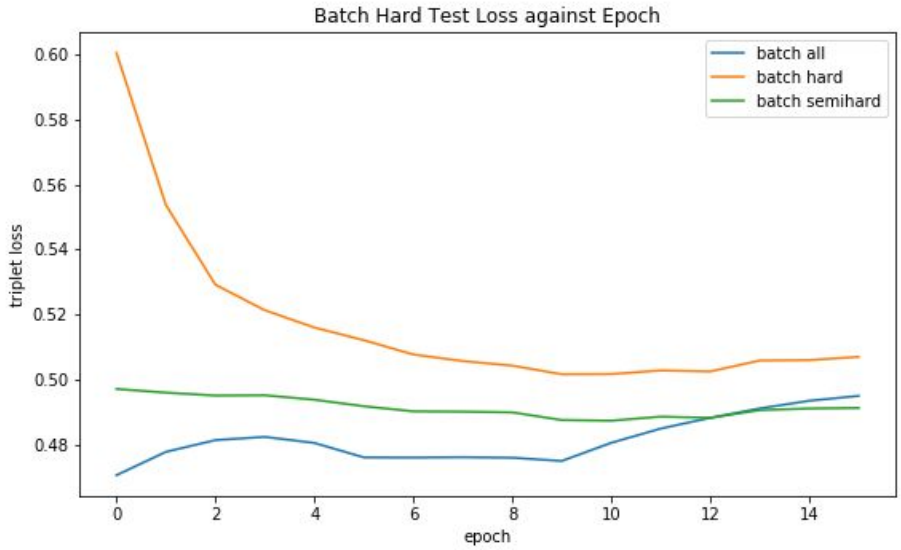
Batch Hard Strategy

Next, we use the batch hard strategy as the loss function and use the batch all and batch semihard strategy as the metrics. We can observe that the batch hard triplet loss for training and testing decreases at similar rate, but stopped decreasing as they reaches the margin.



Graph showing training and testing batch hard triplet loss against epoch

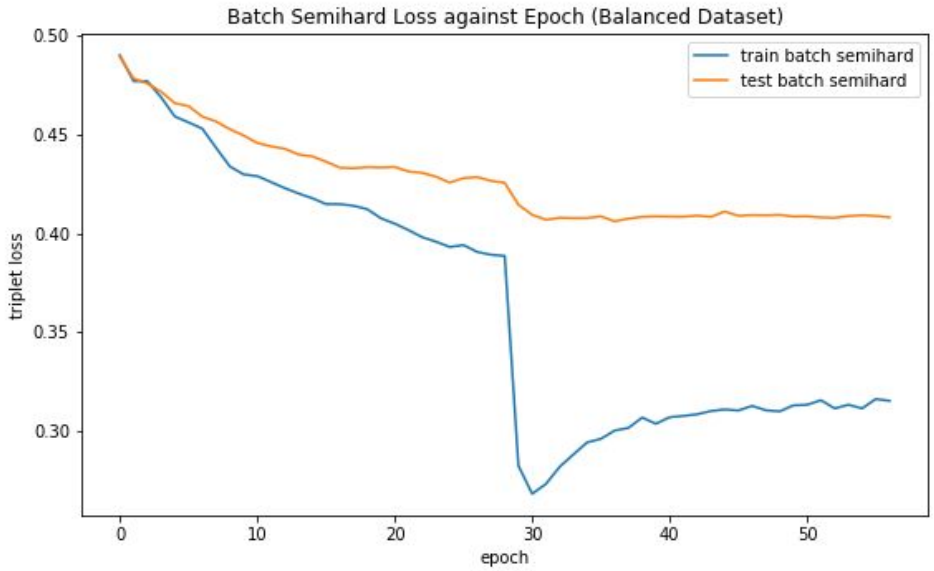
We can also observe that the testing metrics, batch all and batch semihard triplet loss tends to converge themselves to the margin. This indicates that the model tends to map all embeddings towards the same point. As a result, for every image, the embeddings would be similar and thus failed to extract features for different images.



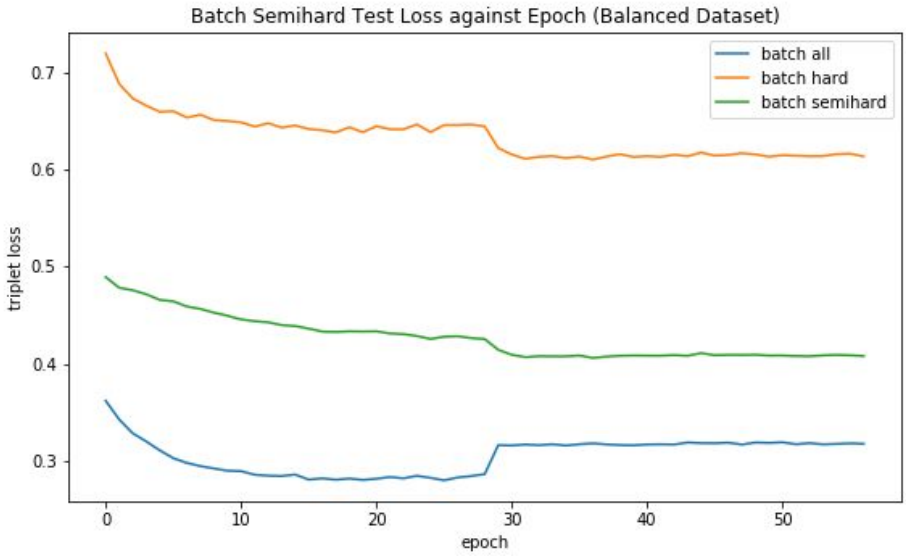
Graph showing the testing batch all, batch hard and batch semihard triplet loss against epoch

Batch Semihard Strategy

Lastly, we use the batch semihard strategy as the loss function and use the batch all and batch hard strategy as the metrics. We can observe that the batch semihard triplet loss for training experienced a drastic drop around epoch 28. This may be due to a sudden gradient exploding, which leads to many semihard triplets managing to cross the easy-semihard boundary, thus, a drastic drop in triplet loss for semihard triplets. However, we can see that the training loss eventually starts to increase, this may be due to hard triplets gradually crossing the semihard-hard boundary, causing the mean loss to increase as more semihard triplets with higher loss start to occur. In overall, we can see a decreasing trend for testing semihard triplet loss and eventually converge.



Graph showing training and testing batch semihard triplet loss against epoch

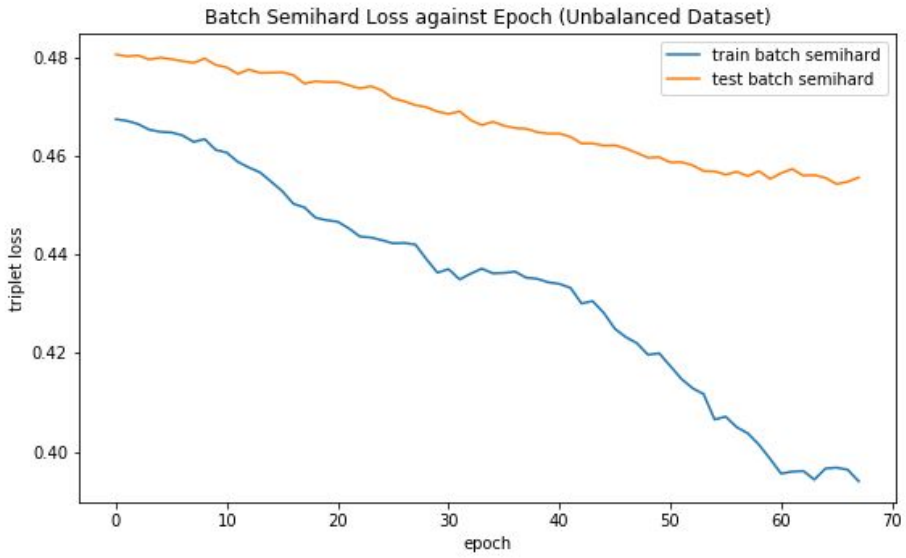


Graph showing the testing batch all, batch hard and batch semihard triplet loss against epoch

Across all strategies, we can conclude that the batch semihard strategy is the best of all. The summary of the final testing loss and metrics for each strategy is shown in the table below:

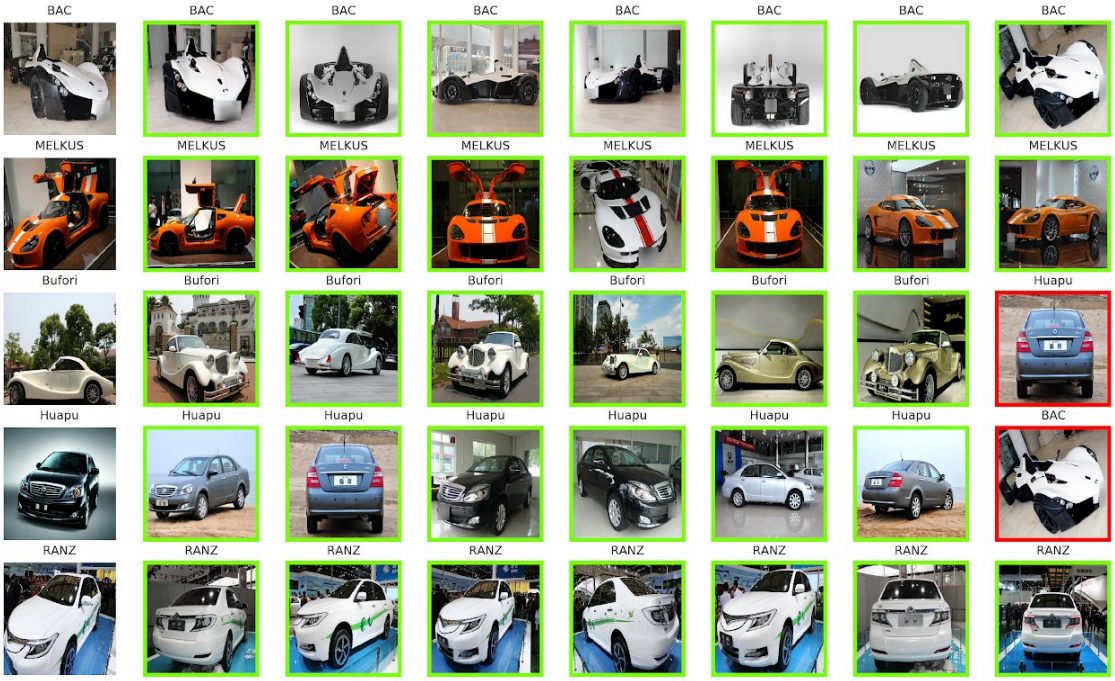
Strategy	Triplet Loss (All)	Triplet Loss (Hard)	Triplet Loss (Semihard)
Batch All	0.391235	1.006762	0.465088
Batch Hard	0.480766	0.501837	0.487558
Batch Semihard	0.318177	0.610343	0.406142

The problem with resampled dataset to yield balanced batches is that some images from classes with large number of images may not be used. Therefore, the model is then further trained using the original dataset.

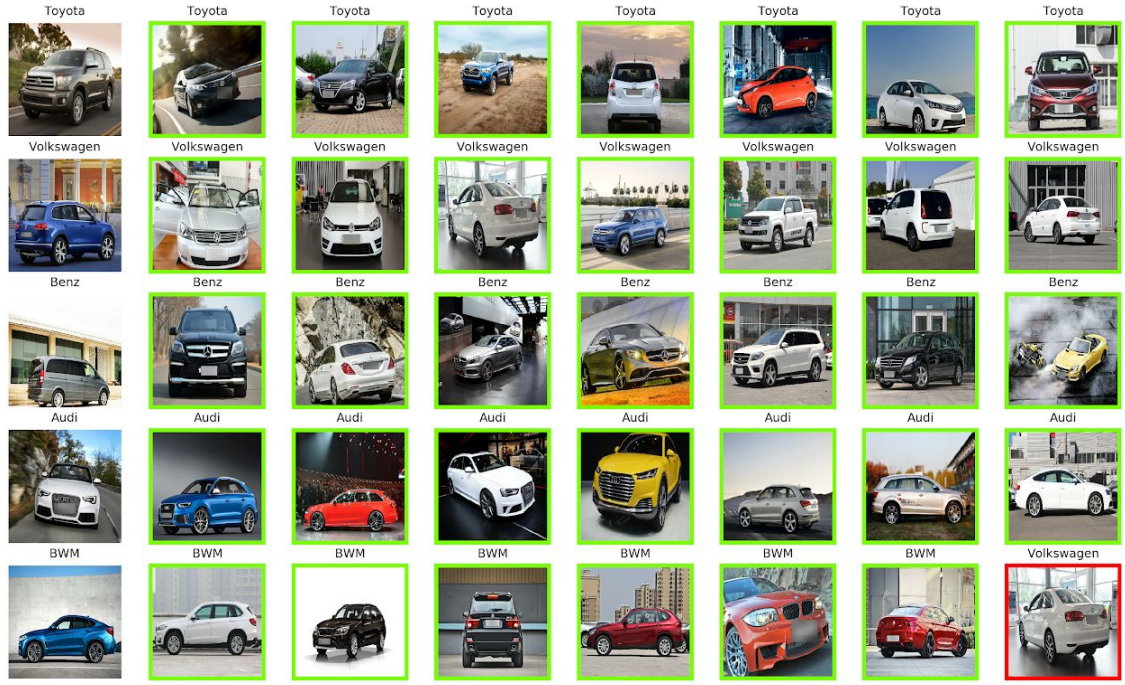


Graph showing batch semihard loss against epoch

Finally, the resulting model is used for car verification purpose. During testing, we choose 8 images from 5 classes, then from each class, we choose 1 random image as the query image and find out the top 7 images with the closest embedding to the query image. The result is shown below.



Each row indicates query image and top-7 retrievals for this query image. These 5 classes are the classes with the least number images.



Each row indicates query image and top-7 retrievals for this query image. These 5 classes are the classes with the most number images.

Each row indicates query image and top-7 retrievals for this query image. These 5 classes are the classes with the least number images.

We can see that in overall, the model manages to extract good embeddings as the top retrievals includes images from different view points.

6. Conclusion

Inception_V3 has significantly better performance than GoogLeNet mainly due to the increase in network size and number of parameters. We also extend the Inception_V3 architecture for car makers classification and find out that 1 more layer of 1000 neurons will help to increase the test accuracy, but not 2 layers. This might be caused by overfitting of the model.

For the car model classification, the base model performs the best, further adding of any layers or dropout will cause deterioration in the performance. The base model is able to generalize well enough for car model classification task.

For the car maker verification model, we can see that batch semihard strategy is the best strategy for online triplet mining. Model trained using triplet loss managed to extract good embeddings as they can differentiate cars from different makers even from different view points.

7. Discussion

We realised that all the models in the published papers which use CompCars datasets are limited to the web-nature datasets which might possess some drawbacks when the model is to be used on the modern transportation system mentioned in Introduction. Surveillance-nature dataset is more suitable to be used if the models developed are to be implemented in sophisticated transportation and road systems.

In contrast, the opposite is also true, the models trained using surveillance-nature dataset will not perform well for tasks like image ranking and users' interest predictions.

More advanced and state-of-art transformer architecture neural networks like ViT-H/14 can be used to achieve higher accuracy. The use of transformer architecture neural networks has gained a lot of success in NLP and researchers are working on using similar neural networks in image classification *[10].*

Besides, more advanced loss function like quadruplet loss inspired from triplet loss can also be further explored *[11]*.

References

[1] Z. Zhang, T. Tan, K. Huang, and Y. Wang. Three- dimensional deformable-model-based localization and recognition of road vehicles. IEEE Transactions on Image Processing, 21(1):1–13, 2012.

[2] E. Hsiao, S. N. Sinha, K. Ramnath, S. Baker, L. Zitnick, and R. Szeliski. Car make and model recognition using 3d curve alignment. In IEEE Winter Conference on Applications of Computer Vision, 2014.

[3] L. Yang, P. Luo, C. Loy, X. Tang. A Large-Scale Car Dataset for Fine-Grained Categorization and Verification, *In Computer Vision and Pattern Recognition (CVPR), 2015*.

[4] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229, 2013.

[5] C.Szegedy,W.Liu,Y.Jia,P.Sermanet,S.Reed,D.Anguelov,D.Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision, 2015.

[7] Schultz, M.; Joachims, T. (2004). "Learning a distance metric from relative comparisons" (PDF). Advances in Neural Information Processing Systems. 16: 41–48.

[8] Moindrot, O. (2018, March 19). Triplet Loss and Online Triplet Mining in TensorFlow. Retrieved November 22, 2020, from https://omoindrot.github.io/triplet-loss

[9] R. Kumar, E. Weill, F. Aghdasi, and P. Sriram. Vehicle re-identification: an efficient baseline using triplet embedding. *arXiv preprint arXiv:1901.01015*, 2019.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G.g Heigold, S. Gelly, J. Uszkoreit and N. Houlsby. An image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv preprint arXiv:2010.11929, 2020.

[11] H. Proença, E.Yaghoubi and P. Alirezazadeh. A Quadruplet Loss for Enforcing Semantically Coherent Embeddings in Multi-Output Classification Problems. arXiv:2002.11644