# Car Image Classification

Arjun Keerthi
Vanderbilt University
Nashville, TN, USA
arjun.b.keerthi@vanderbilt.edu

## Abstract

This report attempts to use transfer learning to classify images of different vehicles using Stanford's Cars dataset. Multiple pretrained networks were tested with Keras and Tensorflow, retaining the feature extraction layers and replacing the top layers of these networks with dense layers that are trained on this specific dataset and compared to relatively simple convolution neural network built from scratch (control). It was found that the pretrained networks showed far less signs of overfitting compared to the control, but all learned at very low rates. The networks based on VGG19 and MobileNetV2 performed the best, displaying the least signs of overfitting.

## Introduction

Deep learning has showed impressive results in learning properties about various datasets and making accurate predictions about new data from domains similar to the training data. However, training such networks to be accurate requires large quantities of data, powerful computing resources, and extended periods of time. Sourcing these for every different domain of data can quickly become prohibitive.

To mediate these restrictions, transfer learning is used to take advantage of previously-trained, well-performing networks. Transfer learning involves taking information learned by a network from a certain set of data and applying to a different domain of data [1]. This is accomplished by "freezing" certain layers in the pretrained network so they will not be updated in the training process with the new dataset. Convolution neural networks begin with convolution blocks that can extract features from different images. Such features exist across different image subjects and thus are useful for images of different objects. As a result, freezing these feature extraction layers is useful for transfer learning. The last few dense fully-connected layers help classify the results specific to the dataset being used for training. Replacing these last layers with new untrained layers and then training the network on the target dataset allows the network to take advantage of the feature extraction learned previously by the pretrained base network and apply this to the new data via the end classification dense layers.

## Purpose

Transfer learning is utilized in this report in order to classify images of vehicles from Stanford's Cars dataset with Keras and Tensorflow [2]. The vehicles imaged in this dataset appear in different orientations and colors, but the overall structure of the vehicles tend to be similar, making classification a potentially difficult task. As a result, potential networks built on top of existing pretrained networks are explored to see how capable they are at learning these images.

## Methods

Tensorflow provides Stanford's Cars dataset inside their tensorflow_datasets package (the dataset is called cars196), so this was used for training and testing for convenience [3]. Keras provides multiple pretrained networks available for importing in Python, which was also utilized for sourcing the base networks. The networks were trained on Google Colab, utilizing its free GPU services.

First, as a control, a basic convolutional network was implemented from scratch and trained on the dataset. This network's architecture consists of a convolution layer with 32 filters and a 3x3 kernel, a convolution layer with 64 filters and a 3x3 kernel, a max pooling layer, and then the outputs are flattened and inputted to two dense layers, the first with 256 units and the second with 196 units. Dropout is used between the pooling layer and flattening (rate of 0.25) and between the dense layers (rate of 0.5).

Next, five different networks were constructed, utilizing the following pretrained networks: Xception [4], VGG19 [5], ResNet50V2 [6], InceptionV3 [7], and MobileNetV2 [8]. These networks were trained on the ImageNet dataset, that provided 1000 classes of images. The top dense fully-connected layers are replaced with a global max pooling layer, a dense layer with 256 units, and a second dense layer (output) with 196 units. Dropout with rate 0.5 is used between the two dense layers.

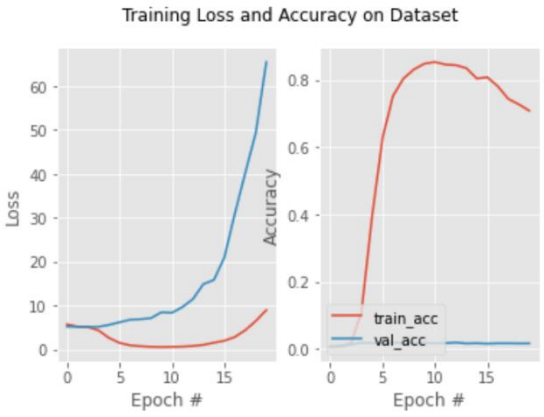
For every network tested, Adam was the optimizer, the learning rate was fixed at 0.001, and categorical cross-entropy was the loss function. These networks were trained on the dataset with 8144 training images using a batch size of 32 for 20 epochs (to fit time and computing restraints). At every epoch, in addition to the training loss and accuracy, the validation loss and accuracy were calculated using the entire test set (8041 samples). After training completed, the losses and accuracies for both the training and validation sets were plotted.

## Results

Below are four samples taken from the training set.

First, the control network was tested. A summary of this network's architecture as well as the plot of its losses/accuracies are given below.

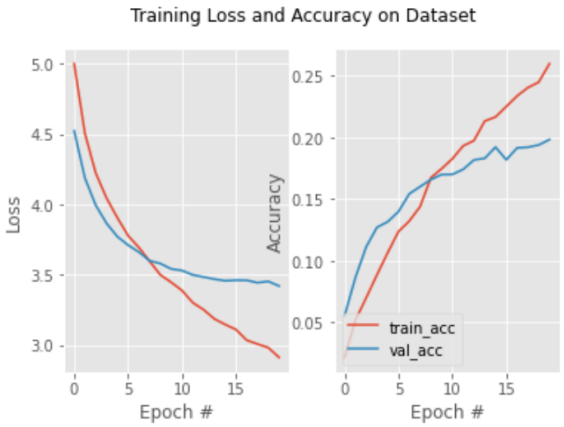
```
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 158, 158, 32)      896
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 156, 156, 64)      18496
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 78, 78, 64)        0
_________________________________________________________________
dropout_23 (Dropout)         (None, 78, 78, 64)        0
_________________________________________________________________
flatten_10 (Flatten)         (None, 389376)            0
_________________________________________________________________
dense_43 (Dense)             (None, 256)               99680512
_________________________________________________________________
dropout_24 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_44 (Dense)             (None, 196)               50372
=================================================================
Total params: 99,750,276
Trainable params: 99,750,276
Non-trainable params: 0
_________________________________________________________________
None
```



This network achieved a relatively high training accuracy (85%) but a very low test accuracy (1.6%), with the validation loss increasing rapidly. This suggests that this network was heavily overfitting and is not capable of discerning differences between the vehicle images. Additionally, the network's training accuracy peaked in the middle of training, suggesting that there is likely fundamental issues with its architecture (perhaps insufficient capacity).

Next, the results for the Xception-based network are given below.

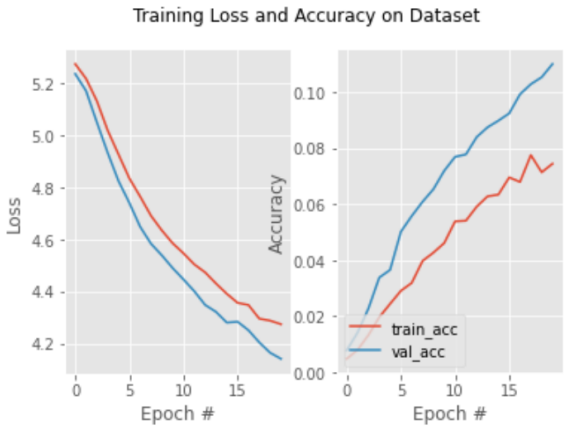
```
Model: "sequential_28"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Model)             (None, 5, 5, 2048)        20861480
_________________________________________________________________
global_average_pooling2d_21  (None, 2048)              0
_________________________________________________________________
dense_47 (Dense)             (None, 256)               524544
_________________________________________________________________
dropout_26 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_48 (Dense)             (None, 196)               50372
=================================================================
Total params: 21,436,396
Trainable params: 574,916
Non-trainable params: 20,861,480
_________________________________________________________________
None
```




This network performed better, with the training accuracy decreasing with the validation accuracy. It does appear that the validation accuracy is starting to level out, which is concerning since the network hadn't reached a very high level of accuracy yet after only 20 epochs (training was limited to this duration to save time on extended training periods). As a result, this network does seem to avoid overfitting better than the control, and looks to be heading towards high accuracies if allowed to train longer, but the leveling of the validation accuracy is problematic.

Next, the results for the VGG19-based network are given below.

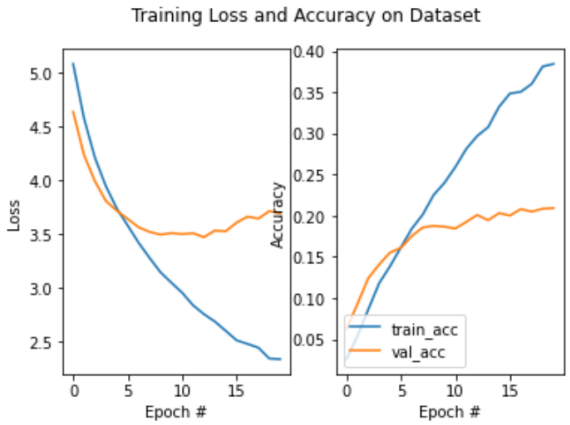
```
Model: "sequential_29"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Model)                (None, 5, 5, 512)         20024384
_________________________________________________________________
global_average_pooling2d_22  (None, 512)               0
_________________________________________________________________
dense_49 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_27 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_50 (Dense)             (None, 196)               50372
=================================================================
Total params: 20,206,084
Trainable params: 181,700
Non-trainable params: 20,024,384
_________________________________________________________________
None
```




Overfitting is much less evident here; the training and validation curves follow the same directions through the training process, with little separation (small generalization gap). However, the accuracies achieved so far in the training process are relatively low (~10%), making it difficult to judge whether the trends seen in the graph will continue.

Next, the results for the ResNet50V2-based network are displayed below.

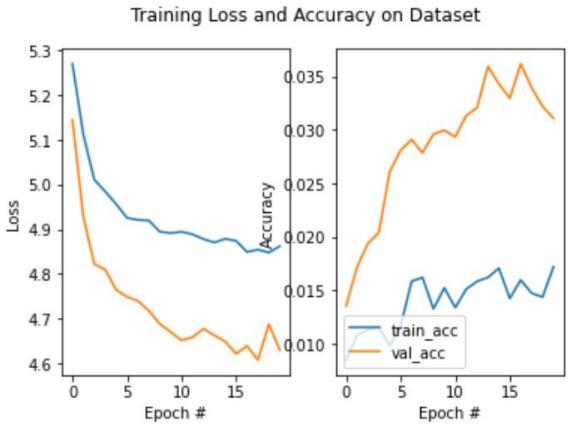
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50v2 (Model)           (None, 5, 5, 2048)        23564800
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               524544
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 196)               50372
=================================================================
Total params: 24,139,716
Trainable params: 574,916
Non-trainable params: 23,564,800
_________________________________________________________________
None
```




This network behaves similarly to the Xception-based network (note: the colors for the curves switched). The validation curve follows the training error initially before plateauing. This network is overfitting the data very quickly, after only 5 epochs.

Next, the results for the InceptionV3-based network are below.

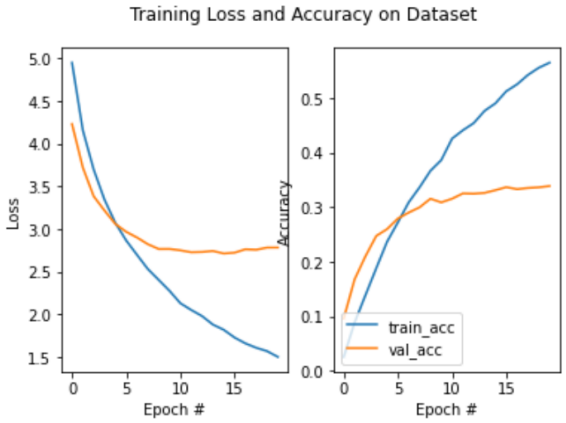
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 3, 3, 2048)        21802784
_________________________________________________________________
global_average_pooling2d_2 ( (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               524544
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 196)               50372
=================================================================
Total params: 22,377,700
Trainable params: 574,916
Non-trainable params: 21,802,784
_________________________________________________________________
None
```




Though the curves look similar to the VGG19-based network, upon close inspection it is clear that this network failed to even achieve slight accuracy with the dataset. It manages to only reach a maximum of 3.5% accuracy, indicating underlying problems with the network's architecture for classifying the vehicles.

Finally, the results for the MobileNetV2-based network are given below.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_160 (Model) (None, 5, 5, 1280)        2257984
_________________________________________________________________
global_average_pooling2d_3 ( (None, 1280)              0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               327936
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 196)               50372
=================================================================
Total params: 2,636,292
Trainable params: 378,308
Non-trainable params: 2,257,984
_________________________________________________________________
None
```




Similar to the Xception- and ResNet50V2-based networks, this network also suffers from overfitting, with the validation accuracies plateauing early in the training process. However, this network achieved higher accuracies that the other versions, reaching 30% validation accuracy so far in the training process.

## Conclusion

From the performances of the networks, it is first apparent that utilizing transfer learning to take advantage of a pretrained network is more performant than building a convolutional neural network from

scratch, as all the networks (except the network based on InceptionV3) performed better with higher validation accuracies and less overfitting.

However, in the 20 epochs that the training was allowed to continue for, MobileNetV2 achieved the highest validation accuracy at 30%, suggesting that it can learn at a faster rate than the other networks. But this network, along with the Xception and ResNet50V2-based networks, saw its validation curves plateau relatively early on in the training process, which will likely become problematic if training were to continue. The VGG19-based network did not show signs of plateauing and the least overfitting, but reached a smaller accuracy (~10%). If we look at the curves from the three aforementioned networks, they look similar to the VGG19-based network's curve up through accuracies of 10%, making it difficult to know whether this network will also plateau.

The networks likely failed to continue increasing their validation accuracies due to the specific nature of the dataset. The vehicles imaged tend to similar in form, which likely causes difficulties for these pretrained networks to identify differences between classes. The pretrained networks were trained on the ImageNet dataset, which includes numerous images of different object over 1000 classes. As a result, the weights included in these pretrained networks that were frozen in the transfer learning process used in this report are not adept at discerning the minute differences between vehicles. The features needed to be detected for this dataset are much more specific that those currently being detected by the pretrained networks.

To fix this in the future, the top convolutional blocks in the pretrained networks can be "un-frozen" so that their parameters may be updated during the training process. It may be worthwhile to un-freeze the entire network so that all parameters can be updated and tuned towards this car dataset. Additionally, data augmentation will likely have helped the networks generalize better.

Nonetheless, this report has shown that transfer learning can be useful and more capable than building a network from scratch, especially for situations where computing resources and time are not unlimited. But for this specific dataset, it turned out that the feature learned from the ImageNet dataset were not highly applicable for vehicle classification.

## Code

The code for this project can be found on Github: https://github.com/arjunkeerthi/car-classifier

## References


[1] Sarkar, "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning," *Medium*, 17-Nov-2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a. [Accessed: 30-Apr-2020].

[2] "Cars Dataset," *Stanford Artificial Intelligence Laboratory*. [Online]. Available: https://ai.stanford.edu/~jkrause/cars/car_dataset.html. [Accessed: 30-Apr-2020].

[3] "cars196 : TensorFlow Datasets," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/datasets/catalog/cars196. [Accessed: 30-Apr-2020].

[4] Chollet and François, "Xception: Deep Learning with Depthwise Separable

Convolutions," *arXiv.org*, 04-Apr-2017. [Online]. Available: https://arxiv.org/abs/1610.02357. [Accessed: 30-Apr-2020].

[5] Simonyan, Karen, Zisserman, and Andrew, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv.org*, 10-Apr-2015. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed: 30-Apr-2020].

[6] Zhang, Ren, Sun, and Jian, "Identity Mappings in Deep Residual Networks," *arXiv.org*, 25-Jul-2016. [Online]. Available: https://arxiv.org/abs/1603.05027. [Accessed: 30-Apr-2020].

[7] Vanhoucke, Vincent, Sergey, Jonathon, and Zbigniew, "Rethinking the Inception Architecture for Computer Vision," *arXiv.org*, 11-Dec-2015. [Online]. Available: https://arxiv.org/abs/1512.00567. [Accessed: 30-Apr-2020].

[8] Sandler, Mark, Howard, Andrew, Zhu, Zhmoginov, Andrey, Chen, and Liang-Chieh, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv.org*, 21-Mar-2019. [Online]. Available: https://arxiv.org/abs/1801.04381. [Accessed: 30-Apr-2020].