Automobile Image Classification

By Bradley Reardon, Salim Haruna, Divya Parmar

Table of Contents

Introduction

- Automobile & adjacent industries operating largely online
  - Auto Insurance
  - Dealerships
  - Vehicle Resale
- Many actions require vehicle image uploads
- Confirming images match descriptions is tedious for manual labor
- Problem Statement: Automating the process of automobile image classification using computer vision to save companies time and money

Dataset Overview

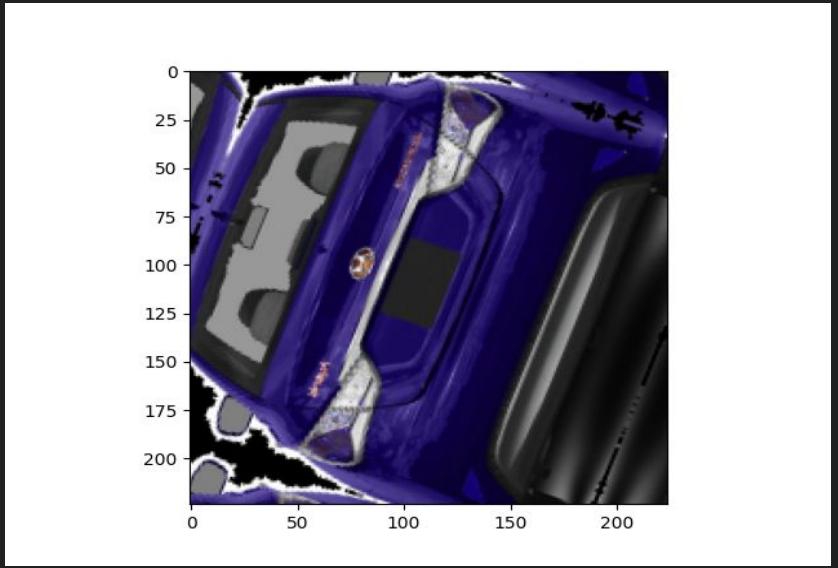
- Stanford Cars Dataset
- 16,185 metadata and images
- Split into 8,144 training images and 8,041 testing images
- 196 classes of cars - i.e. *2012 BMW M3 coupe*
- Target balance: 30-50 images per class with class 67 having 68 images.

Preprocessing

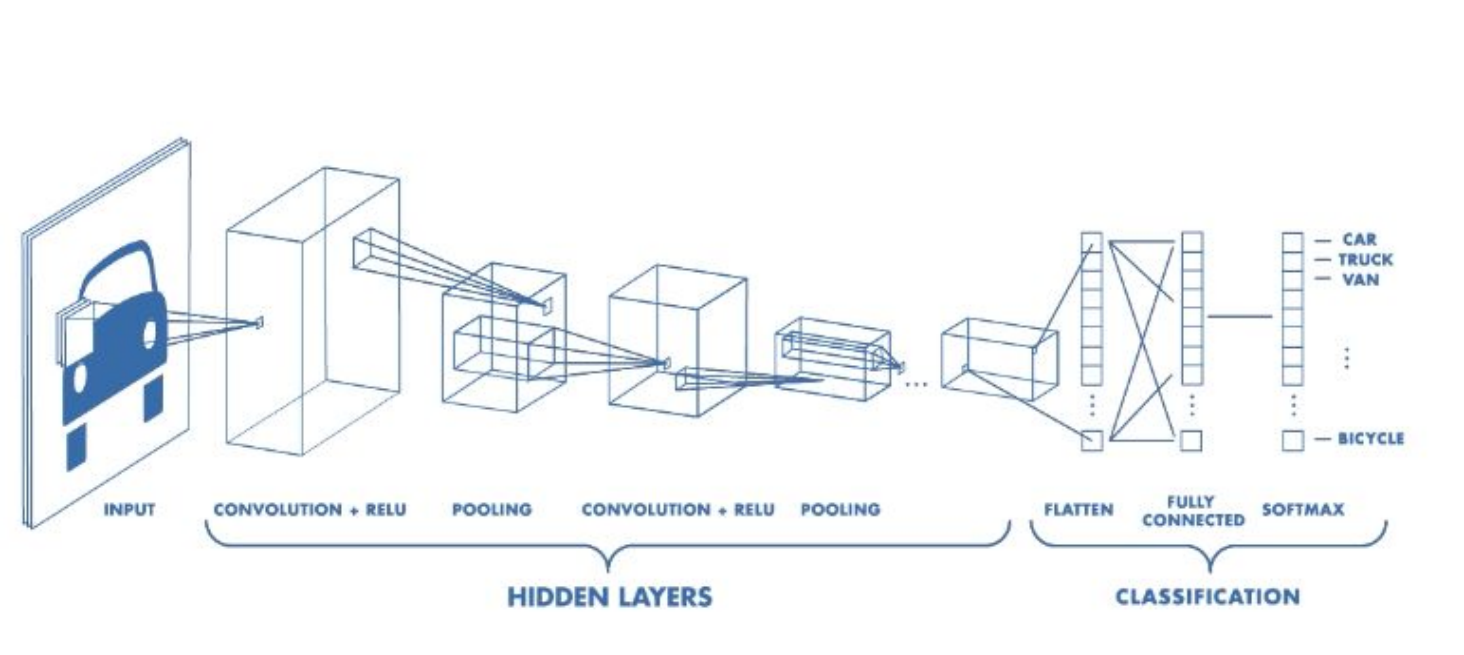
- Load Train Meta data with pandas into a dataframe.
- Load image path from meta data with cv2 and generate image convolution.
- Crop Images using the x1,x2,y1,y2 values in the meta data.
- Resize image into 224 by 224.
- Augment Images by random rotate and random flip.
- Remove Background by filtering salt pepper noise by the median filter and identifying significant contours.
- Image Normalization by making mean = 0 and standard deviation 1.
- Saved image matrices into a numpy file (.npy)
- Train and validation split using the pareto rule of 80 : 20
- Training was done on 42,834 image samples.
- Perform the same process on the test dataset.

Preprocessing Results

Preprocessing Results



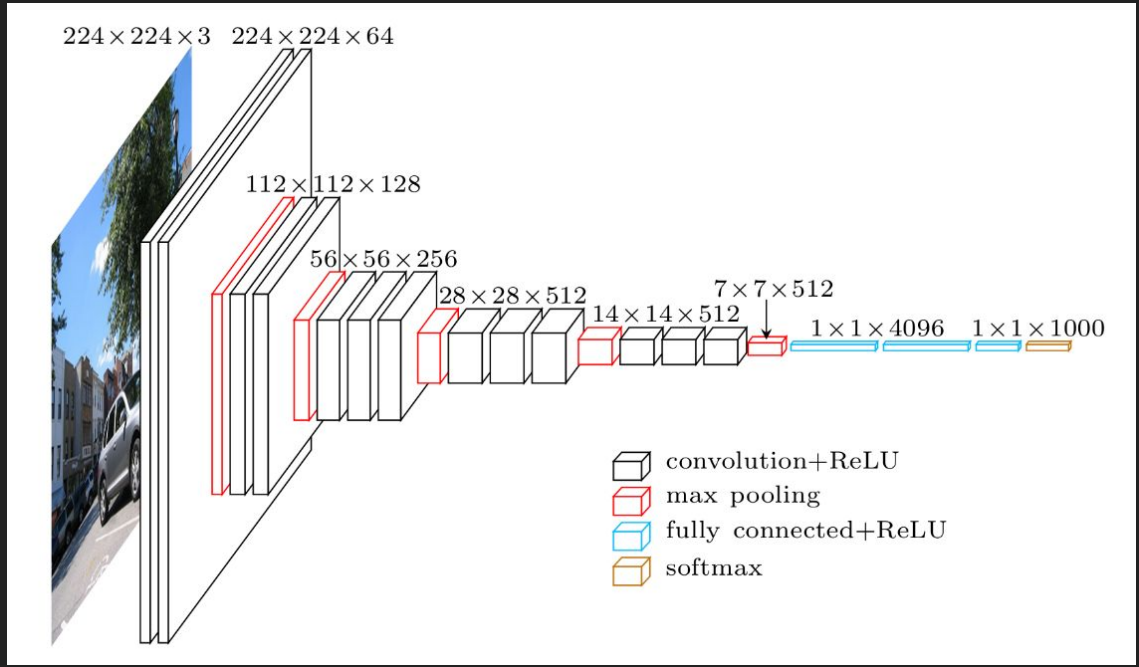
Convolutional Neural Network (CNN)



Convolutional Neural Network Architecture (Source)

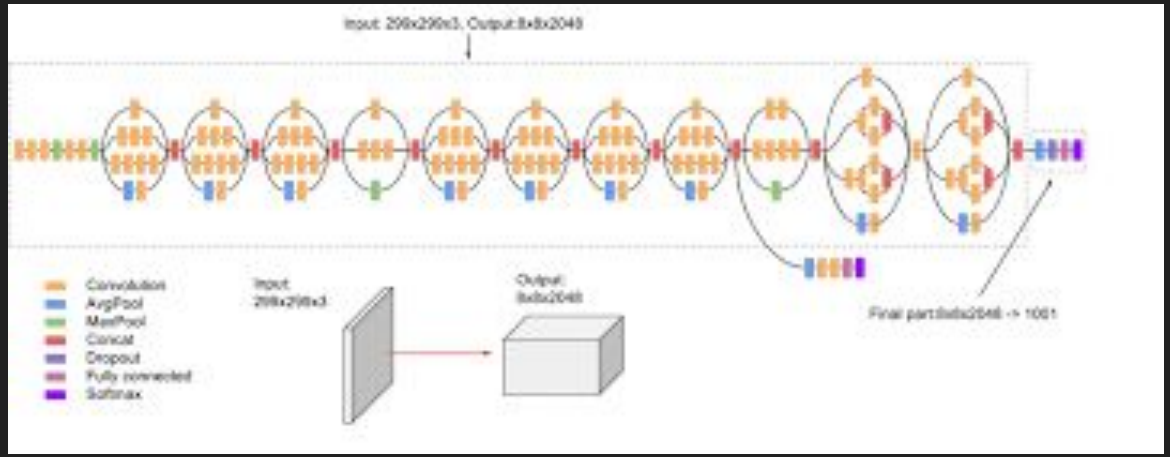
Network & Models - Pre-trained VGG19

- Built by a group named Visual Geometry Group at Oxford's
- 19 Layers
- Trained on 14M+ images in the ImageNet database
- Highly optimized weights



Network & Models - Inception V3

- 48 Layers
- Computationally efficient - smaller convolutions & a lot of pooling
- Trained on 1M+ images
- Can classify images up to 1000 classes
- Auxiliary classifier: an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss.



Network & Models - Custom CNN

- Layers: Conv3x3 (16), BatchNorm, MaxPool, Conv3x3 (43), BatchNorm, AvgPool, Fully Connected (400), Dropout, BatchNorm, Fully Connected (197)

- Tested as a baseline comparison to the VGG19 model.

- Shallow model provides very poor performance

Results Table

Model	Parameters	Hamming	Accuracy	Cohen Kappa	F1 score
VGG	Imagenet Batch:64 Epochs : 5 Adam	0.654	0.40	0.3425	0.41
Inception V3	Imagenet Batch:64 Epochs : 5 Adam	0.719	0.281	0.277	0.27
Custom	Random weights Batch:64 Epochs : 5 Adam	0.970	0.03	0.0252	0.02

Results Summary

The best metrics obtained using the VGG19 model are as follows:

1. Hamming loss : 0.654
2. Accuracy Score : 0.40
3. Cohen Kappa Score : 0.3425
4. F1-macro Average : 0.41
5. Last loss value : 0.941

To achieve the metrics above during training, we used the following hyper parameters

1. Activation : Softmax
2. Weights : Imagenet
3. Epochs : 5
4. Batch Size : 64
5. Optimizer : Adam

Conclusion

- The VGG19 model with pre-trained weights greatly outperformed our custom model with random weights. This demonstrates the value of architecture tested by research scientists and the value of time/computing power spent on pre-training.

- Using data augmentation and image pre-processing improved our results. This speaks to the power of (1) more data and (2) removing noise from the image so the model can learn from the correct information.

Areas for Improvement

- We realized late in the process that we could save intermediate files to avoid duplicating pre-processing steps. This cut down our model run time from ~16 hours to under 1 hour. This would have allowed us to test more models.

- With a more efficient model pipeline mentioned above, we could have tested more model architectures, optimizers, learning rates, etc. to optimize our final metrics.

Citations

- CNN Architecture
  - https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
- VGG19 Architecture
  - https://iq.opengenus.org/vgg19-architecture/
- Median Blur technique for pre-processing
  - https://theailearner.com/tag/cv2-medianblur/
- Random flip technique for pre-processing
  - https://www.tensorflow.org/api_docs/python/tf/keras/layers/RandomFlip
- https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/