Open in app Get started

Published in Towards Data Science

Bhanu Yerra Follow
Dec 20, 2019 · 6 min read · Listen

Save

Car Image Classification Using Features Extracted from Pre-trained Neural Networks

Is that a Corvette?

Source

various stages of purchasing: searching, pre-qualifying, applying and finally buying. Popular websites for car buyers include AutoTrader.com, Kelly Blue Book, Cars.com, Carvana.com.

Cox Automotive's report indicates that most market leaders and Silicon Valley startups speculate that car sales will shift **completely** to online retailing. This might be an extreme speculation, but these market leaders are interested in providing better user experience while buying a car online, and better recommender systems when a user searches for cars. Peer-to-peer sales platforms like Craigslist, Shift, eBay Motors, etc are also interested in better fraud detection and monitoring of user postings.

A car image classification system can address these business issues:

1. Ground-truthing of posted used car images on peer-to-peer sales platforms — are these images actually the cars they specify? Do multiple exterior pics represent the same car?

2. Organizing web-page displays based on the user uploaded images

3. Recommending alternative cars available in the inventory that have similar looks and price

In addition, a simple classification system of cars can help in identifying the fine-grained features important for 3D object detection for self-driving cars.

Approach

The approach used for this analysis has three stages: feature extraction stage, model building stage, and a model evaluation/error analysis stage.

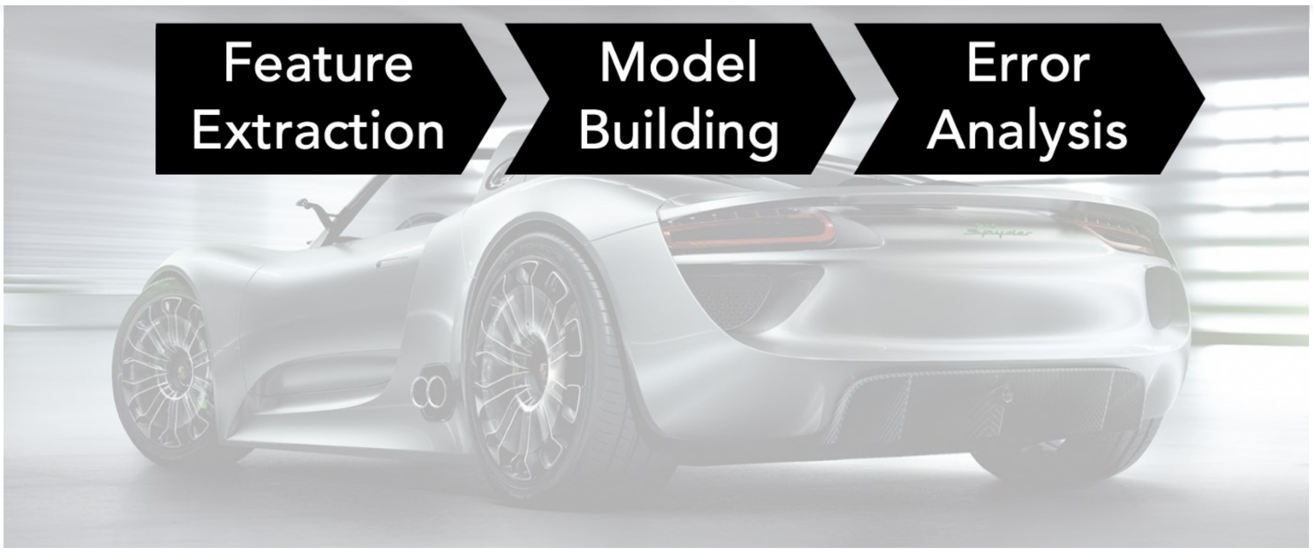


Figure 1: Approach Used for the Analysis

Stanford's car image dataset was used for this analysis. This dataset consists of 16,185 total images (train set + test test) labeled with 196 classes based on the car's Make/Model/Year. These images come in various sizes and resolutions. For this analysis, the 196 image labels in the dataset were consolidated to five vehicle types as shown in the figure below. Although these represent somewhat 'crude' consolidation of vehicle types, they proved to be more manageable, and adequate for the image classification task.

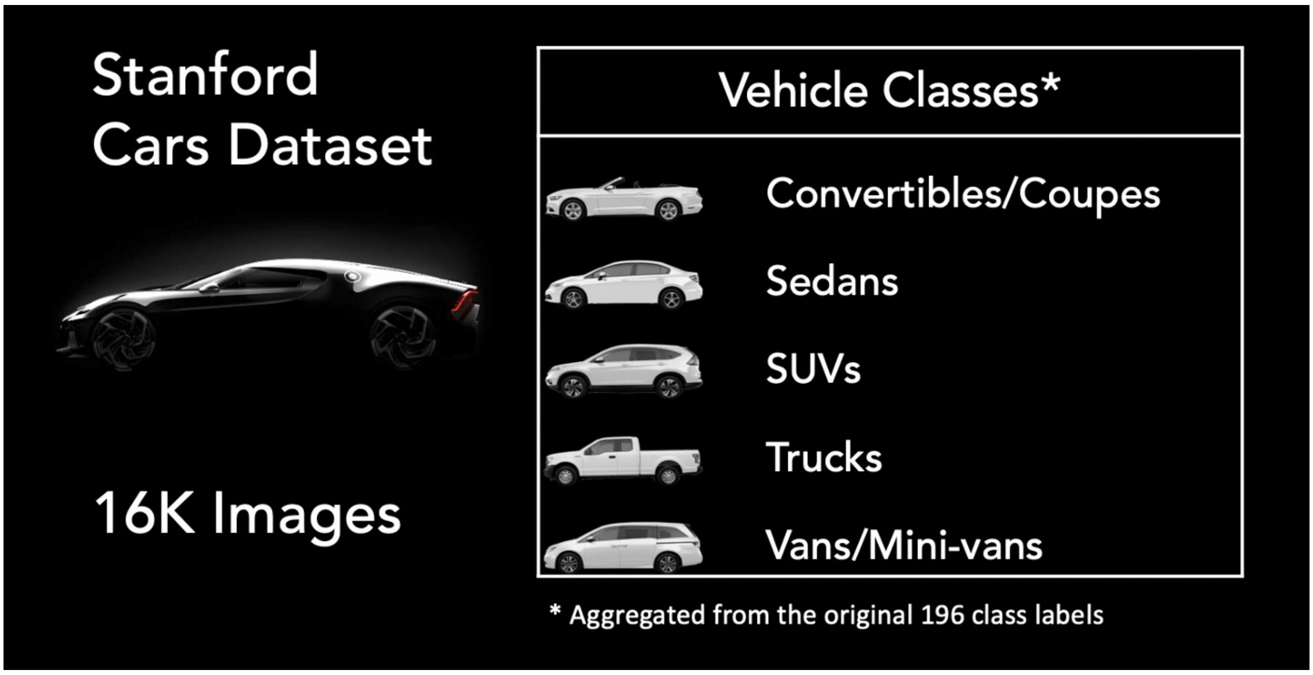
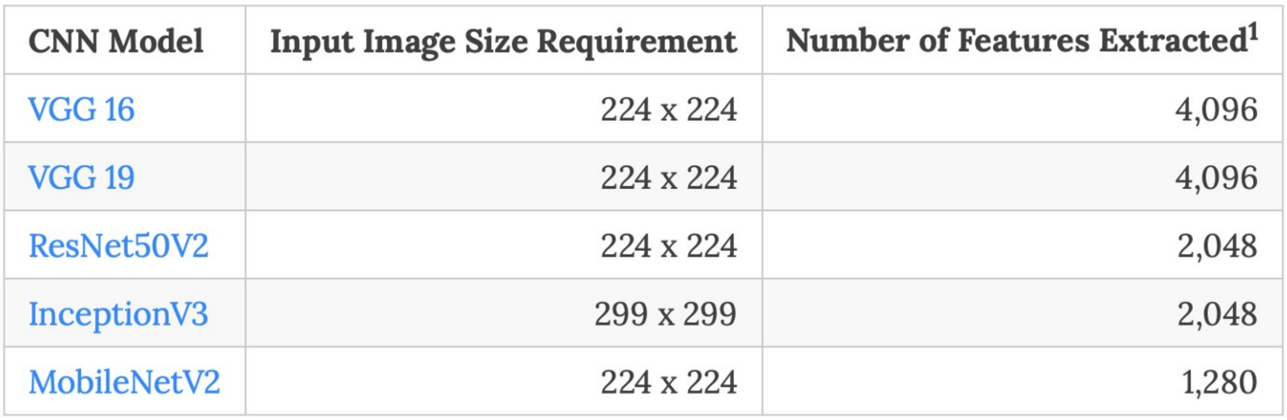


Figure 2: Vehicle Classes Used in the Analysis

these CNNs provide these weights freely, and modeling platform Keras provided a one stop access to these network architectures and weights. Overall, five CNNs, as shown in the figure below, were used in extracting the features and were separately used for classification task to compare which deep learning architecture performed the best. The features extracted were from the first fully connected hidden layer above the final softmax layer.



CNN Model	Input Image Size Requirement	Number of Features Extracted[1]
VGG 16	224 x 224	4,096
VGG 19	224 x 224	4,096
ResNet50V2	224 x 224	2,048
InceptionV3	299 x 299	2,048
MobileNetV2	224 x 224	1,280

Figure 3: Vehicle Classes Used in the Analysis ([1] Features were extracted from the fully connected hidden layer below the final softmax layer)

Model Building

The first step performed in the model building stage was reducing the dimensionality of the extracted features. This step helped in bringing the dimensions of the features to a manageable number and increases the speed of iterative model building step. Dimensionality reduction was performed using Principal Component Analysis (PCA), and a systematic search resulted in using a reduced dimension of 500 features that captured about 85–90% of the variance in the features across five CNNs.

The original dataset has roughly 50–50% train-test split. For this analysis, these tests were combined together and a train-test split of 80–20% was used. Train-test sets were established such that the class distribution is maintained.

A number of classification methods were employed in this stage of the process. These include: Logistic Regression, Random Forests, SVMs with polynomial kernels, XGBoost, and a shallow neural network (Multi-layer Perceptron classifier from sklearn).

The dataset after consolidating to five classes has a class imbalance with Vans oversampling, and Synthetic Minority Oversampling TEchnique (SMOTE).

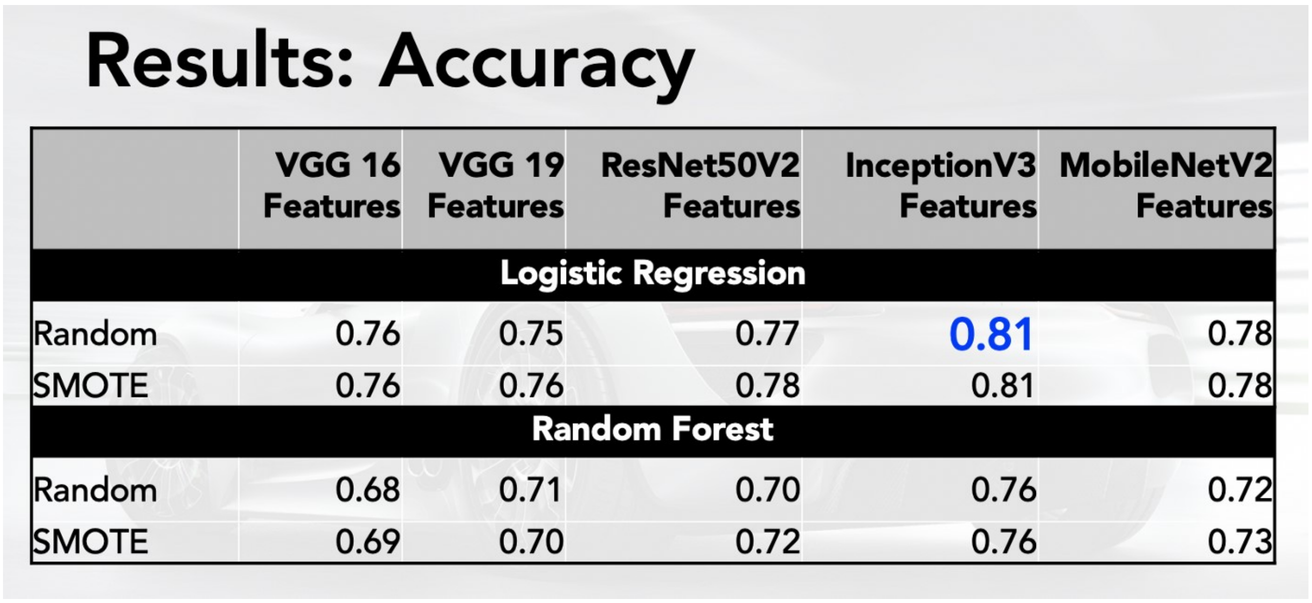
In addition, a stacked classifier was tested to see if combining the results from the best Logistic Regression and Random Forest models will yield better results than either of them could individually.

Model Results and Error Analysis

For comparing the model performance, multiple metrics were considered. But after few iterations, accuracy was noted to be a sufficient metric to assess the model performance. Selecting any other metric (say F1 Score) would have resulted in arriving at similar conclusions. A detailed error analysis was performed by examining the misclassified images and exploring any similarities in these mis-classifications.

Results

The results of the best performing models are presented in the following figure. As seen below, the model using features extracted from InceptionV3 performed best in general, with Logistic Regression resulting in slightly better accuracy than Random Forest. Model using InceptionV3 features, Logistic Regression with L2 regularization, and random oversampling technique was selected as the *Preferred Model* for this analysis.

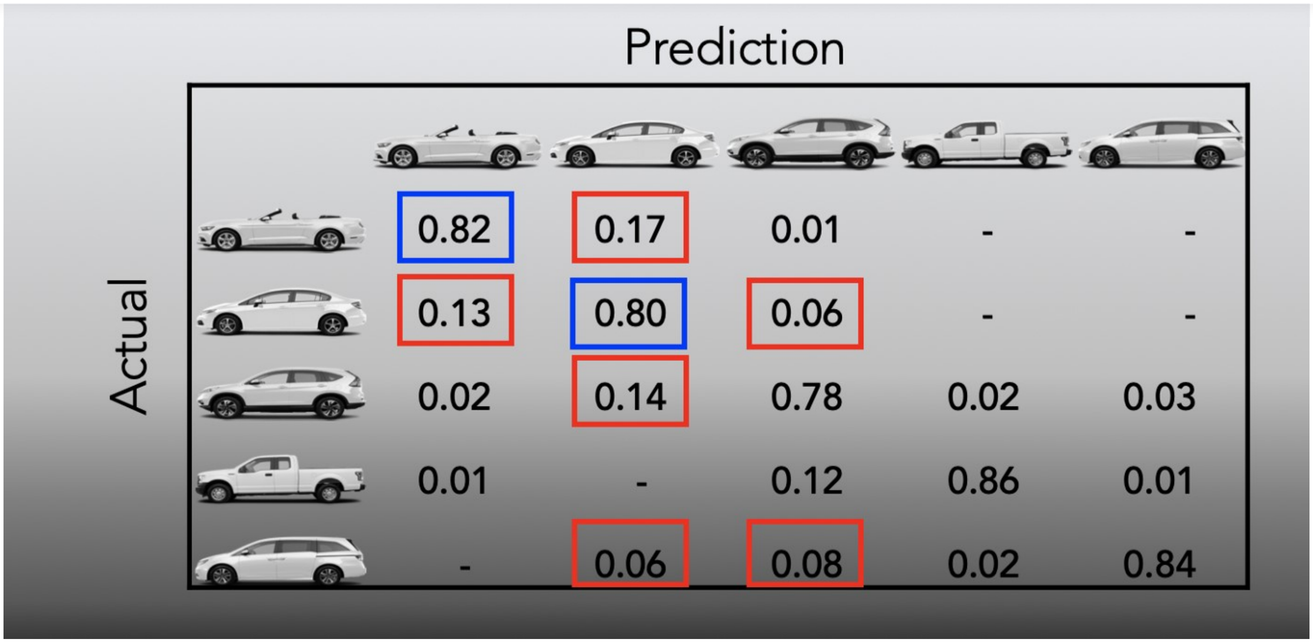


Results: Accuracy

	VGG 16 Features	VGG 19 Features	ResNet50V2 Features	InceptionV3 Features	MobileNetV2 Features
Logistic Regression					
Random	0.76	0.75	0.77	0.81	0.78
SMOTE	0.76	0.76	0.78	0.81	0.78
Random Forest					
Random	0.68	0.71	0.70	0.76	0.72
SMOTE	0.69	0.70	0.72	0.76	0.73

Figure 4: Comparison of Model Accuracy for Select Runs

Confusion matrix for the preferred model is presented in the figure below:


Prediction
Actual

	0.82	0.17	0.01	-	-
	0.13	0.80	0.06	-	-
	0.02	0.14	0.78	0.02	0.03
	0.01	-	0.12	0.86	0.01
	-	0.06	0.08	0.02	0.84

Figure 5: Confusion Matrix for the Preferred Model(InceptionV3 Features + Logistic Regression)

In addition to tracking the model accuracy, a detailed error analysis was performed to assess where the model is performing poorly. This error analysis was critical in performing a quality check (QC check) on the class labeling, identifying model "blind spots" and future improvements.

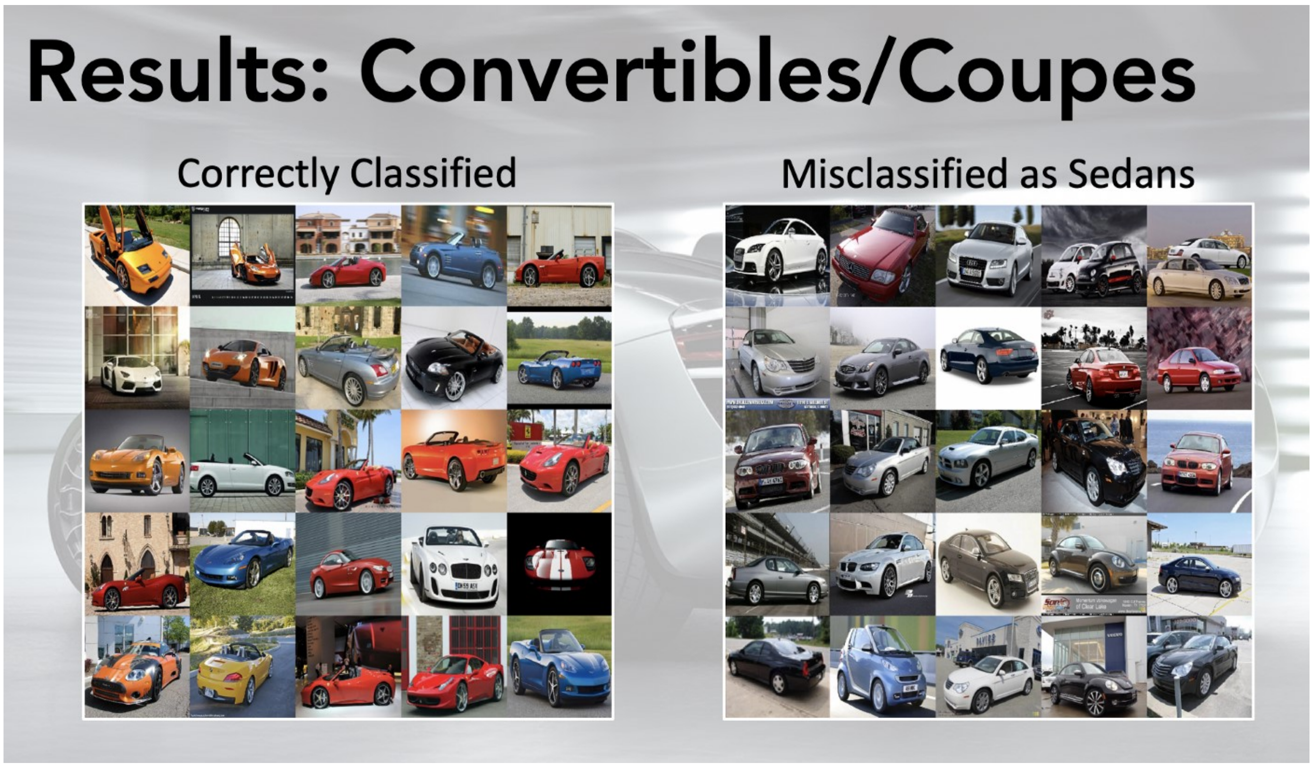


Figure 6: Error Analysis of Convertibles/Coupes for for Preferred Model

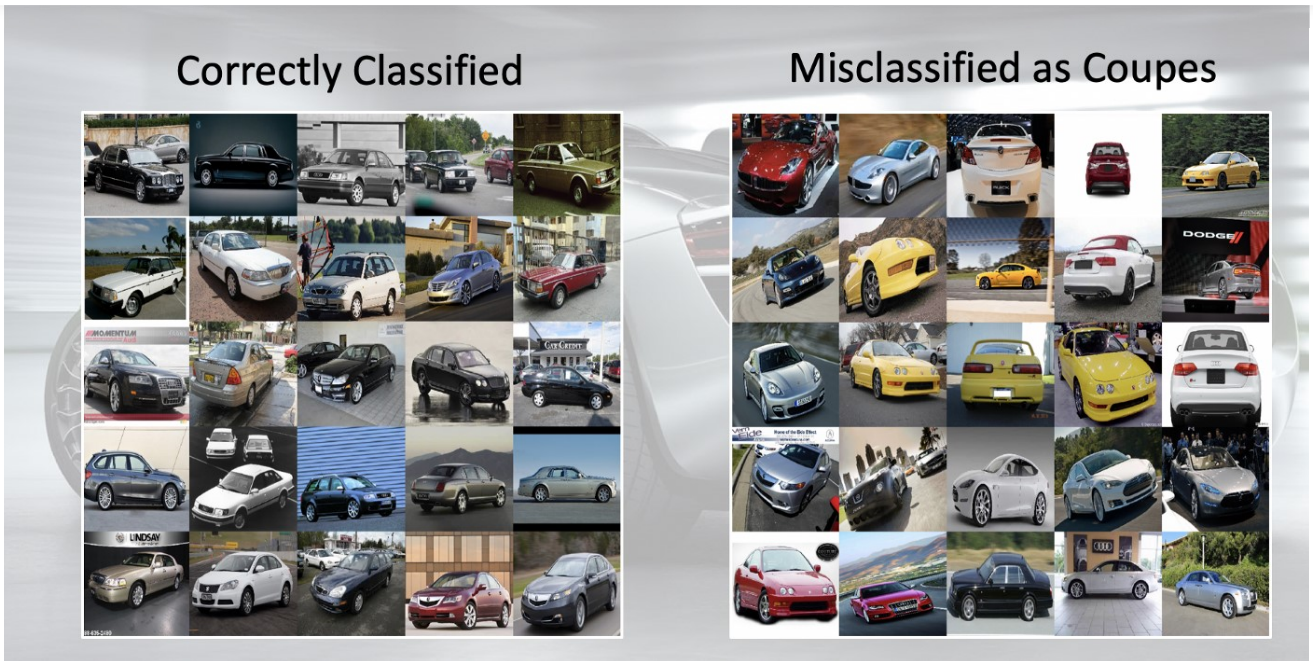


Figure 7: Error Analysis of Sedans for the Preferred Model

As seen in figures 6 & 7 above, the model seem to pick up colors as classification feature. Convertibles/Coupes correctly identified are rich in color, while those misclassified as Sedans are less colorful. Likewise, as seen in figure 7, Sedans that are misclassified as Convertibles/Coupes are more colorful than the Sedans that are correctly classified. This is an encouraging result in most cases as this will pick up colors specific to certain car Make/Model but the weight placed on color is higher than the weight placed on features that represent shape and size.

Conclusions

The primary conclusions from the above image classification analysis are:

1. Prototyping a classification model using pretrained CNN features is quite effective and easier than fully building a deep neural network from scratch.

2. Error analysis is quite useful, and provides insights on how models can be employed.

References

1. Stanford Cars Dataset

2. ImageNet Dataset

GitHub Repo:

mlBhanuYerra/Metis_prj3

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or...

github.com

Feature Extraction: extract_feature

Model Building: 3_FinalModelsRuns.ipynb

Error Analysis: 3_Results_Presentation.ipynb