# Car Category Prediction

Zhiwen Xu zxu365@wisc.edu
Runlin Tang rtang45@wisc.edu
Bochao Li bli339@wisc.edu

## 1. Abstract

Since the start of industrial evolution, people have become increasingly relying on cars as a mode of transportation and a symbol of wealth. We conduct this project to better understand how to use deep learning techniques to predict categories of cars accurately and help people identify the categories of cars conveniently and with high accuracy. More importantly, we are enabling police to use this algorithm to be implemented in the traffic monitoring cameras to better monitor road conditions and track down criminals. Ethically, however, this project may raise serious questions about violating people's privacy when the algorithm is implemented by the police. To handle both the technical and ethical part of this project, the authority needs to set a clear set of principles, inform and educate the public before implementing the algorithm. To build this model, we will first clean and divide the dataset into training, testing, and validation datasets. Then we will use background removal techniques to focus only on the cars but not the noisy backgrounds. Finally, we will use multilayer perceptron and convolutional neural network algorithms and we will then compare the accuracies between these algorithms to determine the best algorithm to be implemented in our model.

## 2. Introduction

People in the US rely on cars to travel almost everyday. It is important to ensure that when they are buying a car, they are buying the right category of car they want. For example, if one wants to buy a SUV, our algorithm can help him/her recognize a SUV accurately. We obtained a dataset called Cars from Krause, Stark, Deng, and Li's research [2]. The dataset includes 16,185 images of 196 classes of cars. Each class contains the information about the make, model, year, and the category of the car. For example, the 2012 BMW M3 coupe is one of the classes of cars in the dataset. In our project, we concentrate on recognizing the category of the car and divide the data into 9 class (categories of cars). We split the data into 12,948 training images, 2,428 testing images, and 809 validation images to ensure that we have roughly a 80-15-5 split of the dataset into training, testing, and validation dataset. Since our dataset is not big enough, and we are not trying to predict the year, make, and model of the car because we think that it may be hard and complicated to predict, we are confident that our algorithm will achieve accurate enough results.

## 3. Related Work

This project uses the same dataset as ours. They mainly fine tunes a ResNet-152 model to fit the data. They have a total 196 different classes, and the data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split.They split the training images by 80:20 rule (6,515 for training, 1,629 for validation).This project achieved a high validation accuracy of 88.70% and test accuracy of 88.88%. Comparing their work with ours, we find their epoch is 100000, which is much larger than the longest one we used in our model. And they also use other dependencies like Tensorflow, Keras and OpenCV [3]. In our project, we used OpenCV to complete data preprocessing steps like background removal and pytorch to train and load models. When researching better background removal techniques, we found a paid website that provides clearer background removal [1]. Unfortunately, these sites do not show the algorithms or packages they use, and we cannot handle such a huge dataset on their website. Our background removal method is based on the grabcut algorithm from the openCV package, which can provide acceptable but less precise results compared to the results from those websites.

## 4. Proposed Method

To recognize different car images,we chose the multilayer perceptron(MLP) and the convolutional neural network(CNN) to fit the data. Since the accuracy of multinomial logistic regression is too low, we used MPL as baseline and CNN models will be compared with it to check improvements.

A MLP model has the following architecture:

In figure 1, $x_i$ are the sample features, which are presented by a long vector. In our data, images were first transformed into a 2D array, 224*224 for example, and then transformed to a vector of length 50176. There are four

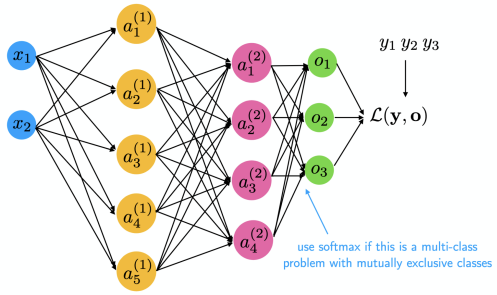


Figure 1. MLP Architecture

layers in total, with $x_i$ as layer 0 or input layer, $a_j^{(1)}$ as layer 1 or first hidden layer, $a_k^{(2)}$ as layer 2 or second hidden layer, and lastly $o_n$ as layer 3 or output layer. The first hidden layer is calculated from the input layer by multiplying weights to each sample feature vector. The second hidden layer is calculated similarly from the first hidden layer. To calculate outputs, softmax activation is applied since we have multiple classes:

$a(x_i) = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)}$

As we train our model in the multilayer perceptron and convolutional neural network, multi-category cross entropy loss function will be used to calculate loss:

$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{k} -y_j^{(1)} \log\left(a_k^{[n]}\right)$

- n: number of samples - k: number of classes - $y$ : one-hot encoding label - a: softmax output

We will calculate the accuracy from the confusion matrix as following:

$\frac{\sum \text{ True positive} + \sum \text{ True negative}}{\sum \text{ Total population}}$

Another model that we built is Convolutional Neural Network (CNN), which is commonly used for computer vision and image classification. A typical CNN has the following architecture:

In figure 2, input data is a 28*28 pixels, black and white image. Since it is black and white, it only has one channel, and therefore its dimension is 28*28*1. The input image is processed by convolutions and feature maps are generated therefore. During this process, a kernel, filer, or feature detector, is slided over the input matrix and sums up the products of weights and inputs within the window. In figure 2, the kernel size is five. After the convolution, pooling layers are applied to feature maps, and max-pooling means passing the maximum cell within the window framework to the next map. If an average-pooling layer is used, the mean of all cells within the window framework is calculated and passed to later layers. This step further filters the data after the convolution. After several convolutions and pooling layers, the architecture is similar to a multilayer perceptron model, which has been discussed above.

There are many different implementations of CNN, such as resnet50, resnet34, vgg16, inception, etc. They are different in their combination of convolution layers and number of layers For this project, we mainly used resnet50 and resnet34 to build models and compare results. Figure 3 shows the architecture of resnet34 and its comparison with a 34-layer neural network:

As figure 3 shows, resnet34 skips several convolutional layers, improving the learning of neural nets. Resnet50 is similar to resnet34 but skips convolutional layers based on 50-layer NN.

In order to improve the accuracy and mitigate the effect of small sample size, transfer learning was considered. Pretrained models of resnet34 and resnet50 are loaded and were further fine-tuned to adjust for our dataset.

## 5. Experiments

### 5.1. Dataset

Our dataset contains 16185 different car images in total, and these images are from a total 196 different classes such as 'AM General Hummer SUV 2000', 'Audi 100 Wagon 1994', and 'Ford GT Coupe 2006'. Our initial design was to train the model to recognize these 196 classes. Because of the small sample size, we splitted these images by 80-15-5 ratio to train,test and validation, respectively. We also splitted the dataset category by category to ensure that each class is splitted by the ratio of 80-15-5. And to further improve our train accuracy, we cut off part of the background. In the data preprocessing steps of the MLP model, we turned the data into black-and-white before training them.The example of this data preparation is shown in Figure 4.

Fitting the black-and-white and cropped images in the MLP model turned out not to be ideal. We met problems such as overfitting and low validation accuracy, so we decided to apply background removal to our dataset. We tried different techniques of background removal, and some of them turned out not to be satisfying. For instance, in Figure 5 and Figure 6, the left image is the original image that we gathered, the middle image is our first attempt, and the right image is the final image to fit the models. The method we used to generate the final image is based on the grabcut algorithm from the openCV package.

Before loading the cropped and background-removed images to our models, we resize each image to a uniform size. We tried different dimensions, such as 32*32, 100*100, and 224*224 to see if there will be improvements in accuracy with different resolutions. We also loaded colored images in the CNN models and compared the results with the black-and-white ones in the MLP model. When generating batches of images, we used different batch normalizations, such as normalizing by mean of 0.5 and standard deviation of 0.5, normalizing by the means and standard deviations from our dataset, and normalizing by the

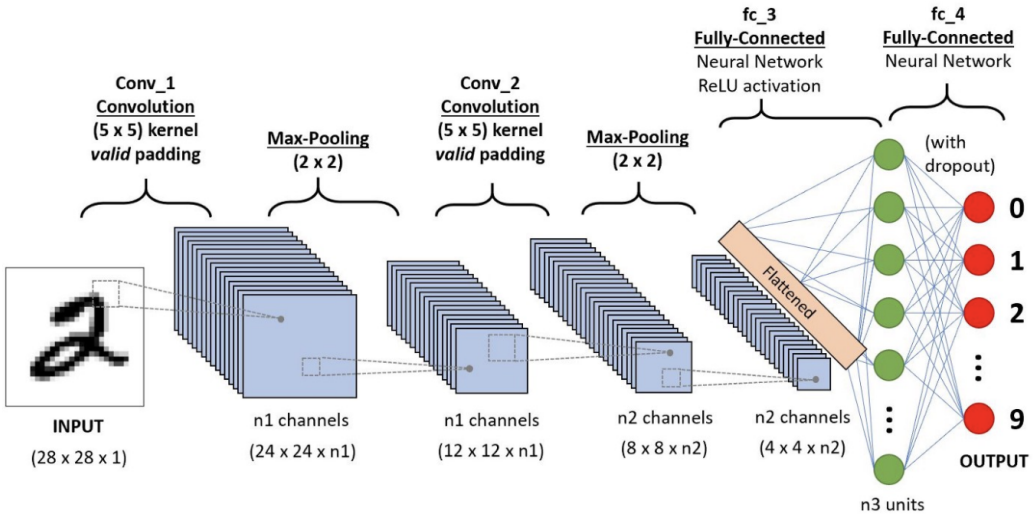


Figure 2. CNN Architecture

mean and standard deviation parameters of pretrained models.

### 5.2. Software

We used jupyter lab to write and run our code in python. We also used Google Colaboratory to run it with GPU

### 5.3. Hardware

We run MLP model and CNN model in different laptops of our teammates. For MLP model we use 2.3GHz Dual-core Intel Core i5 CPU to run our model, and the code runs with approximately 1.25 minute/epoch. For the CNN model we use google colab to run it with Tesla K80 GPU, and the code runs with approximately 0.5minute/epoch to 2minute/epoch, depending on the architecture of CNN.

### 5.4. MLP Model

We tried different implementations of MLP models and compared them. We tuned the parameters such as learning rate, batch size, epoch, number of hidden layers, and size of hidden layer. We tried different combinations of learning rates of 0.1 and 0.07, batch sizes of 8, 16, and 32, epochs of 10, 20, 25, and 50, one hidden layer of 1000 neurons or two hidden layers of 1500 and 800 neurons.

### 5.5. CNN Model

We implemented a resnet34 model and also loaded a pretrained resnet34 model. We used batch sizes of 64, 128, and 256, learning rates of 0.01 and 0.001, and epochs from 10 to 50 to train different models. For fine-tuning based on pretrained models, we tried to unfreeze the last layer, the last three layers, and all layers. A resnet50 model was loaded as well to compare with the resnet34 model.

## 6. Results and Discussion

### 6.1. Results

The results for training our dataset on 196 labels are not satisfying as the highest validation and test accuracy among all models is less than 10 percent. Failing to achieve our original goal, we merged certain classes to check if our model can identify cars more generally. Instead of predicting specific make, model and year, we tried to predict the basic category and generated new class labels: SUV, Coupe, Sedan, Convertible, Wagon, truck, van, MPV, and mini car.

Among all the experiments on new labels, MLP model that has epoches of 25, learning rate of 0.075, one hidden layer.

Among all the MLP models, MLP model with learning rate of 0.075, batch size of 8, and one hidden layer of 1000 neurons has the best performance on accuracy. It has a training accuracy of 40.22%, validation accuracy of 27.22%, and test accuracy of 28.73%. This MLP model is set as the baseline and CNN models will be compared with it. When resnet34 is run for 50 epochs and a batch of 128 data is loaded, the model reaches its highest accuracy. The training accuracy is 97.38%, validation accuracy is 30.43%, and test accuracy is 32.64%. When the pretrained resnet34 is run for 10 epochs and has all layers unfreezed, it improves a lot in validation and test accuracies, as validation accuracy rises to 67.84% and test accuracy rises to 64.91%. For the best resnet50 model, we chose the batch size to be 128 and trained the model for 20 epochs. It achieves a training ac-

| Model | Training Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| MLP | 40.22% | 27.22% | 28.73% |
| Resret34 | 97.48% | 30.43% | 32.64% |
| Pretrained Resnet34 | 96.44% | 67.84% | 64.91% |
| Resnet50 | 98.84% | 54.00% | 55.26% |

Table 1. Summary of Accuracies of Different Models

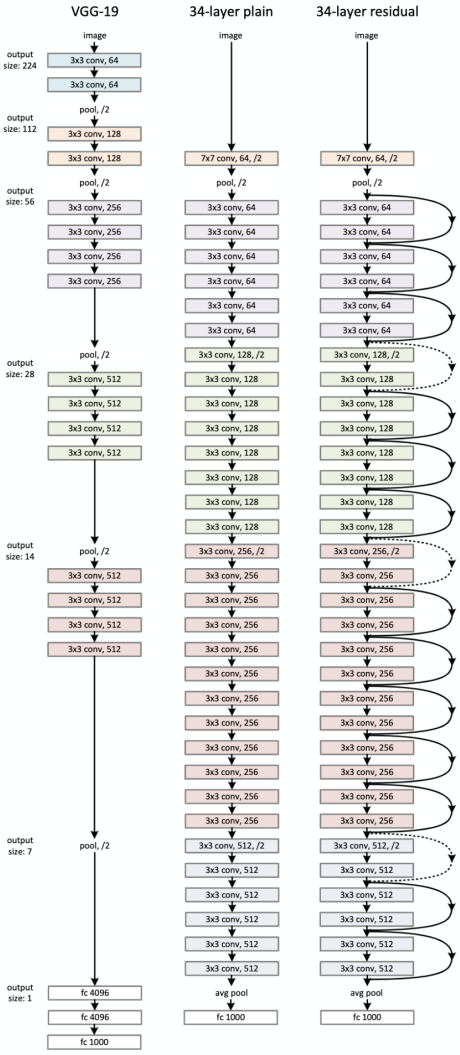


Figure 3. Architecture of Resnet34 and Comparison with 34-layer Neural Net(NN)

curacy of 98.84%, validation accuracy of 54.00%, and test accuracy of 55.26%. See table 1 for a summary of performances of different models.

### 6.2. Discussion

After applying background removal, the validation and the test accuracy improve by 6% in our initial design.

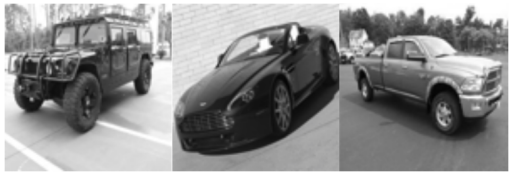

Figure 4. Black-and-white images transformed from original dataset

Figure 5. Image Preparation

Figure 6. Image Preparation

Therefore, it helps boost model performance and we continue using the background-removed images in later experiments with the MLP and the CNN models. However, the background removal method we used still has limitations and does not work well for all images. For images which have simple and clear backgrounds, this method worked well and could successfully separate foreground from background. The first image in Figure 7 gives an example of successful background removal. However, for images which have a complex background such as trees, people, and other cars, the result of background removal was not ideal. For example, the second and the third images in Figure 7 still have some parts from the background remaining. The remaining trees and other cars from the background might lead to false recognition of car type. As mentioned, we found a background removal website that generates better images. In Figure 8, the last image is the resulting image from this website. For future experiments, new techniques and implementations of background removal are needed.

As shown in table 1, MLP and CNN models generate dif-

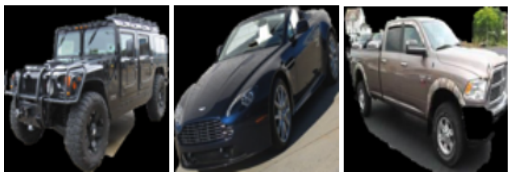

Figure 7. Example of Background Removal

Figure 8. Example of Successful Background Removal

ferent results on our dataset, and the comparison between them is worth further exploration. In the comparison of resnet34 with MLP, the training accuracy improves a lot, but there is no obvious improvement for validation accuracy from 27.22% to 30.43% and test accuracy from 28.73% to 32.64%. Looking at the training and validation accuracies at each epoch shown in Figure 9, we found out that after epoch 10, the validation accuracy stops increasing while training accuracy continues to climb up. One possible reason for the discrepancy between training and validation accuracies is overfitting, since our dataset is relatively small. To prevent overfitting, early stop is used and in later experiments with resnet34 and resnet50, the model is trained for 10 or 20 epochs.

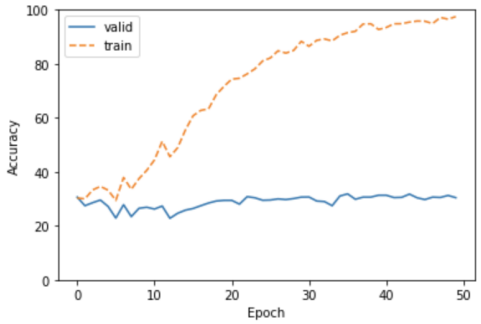


Figure 9. Accuracies of MLP

After applying transfer learning to resnet34, the model doubles its accuracy in validation and test dataset, indicating that it is better at learning features from different classes and generalizing the learning to unknown datasets. The deep architecture of resnet34 allows the model to extract and code features of different classes more accurately. To see if deeper architecture boosts accuracy, a pre-trained resnet50 model is loaded. Validation and test accuracies drop by 7% and 9%, compared with pretrained resnet34. The reason for the decrease is complex and hard to figure out, such as an inappropriate combination of hyper parameters like learning rate and batch size, and a small number of training epochs. It does not necessarily mean that deeper architectures do not have high prediction ability.

In addition to comparison between different models, the confusion matrix is also checked to understand the model prediction ability. Figure 10 shows the confusion matrix output for resnet34 and figure 11 shows the one for pretrained resnet34.

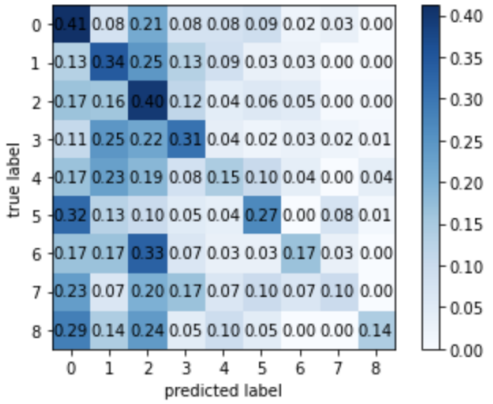


Figure 10. Resnet34

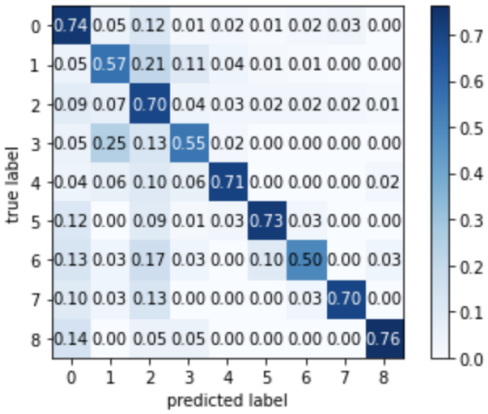


Figure 11. Pretrained Resnet34

The classes are labeled from 0 to 8, corresponding to SUV, Coupe, Sedan, Convertible, Wagon, truck, van, MPV, and mini car. As figure 10 shows, resnet34 performs poorly in identifying Wagon, van, MPV and mini car. After transfer learning is applied to the model, it improves its accuracy in predicting these labels. All the labels are predicted by more than half of chance in the final model. However, Coupe, Sedan, and van are the top three classes that the model gets confused. A coupe is likely to be mislabeled

as a sedan or a convertible, and a convertible is likely to be mislabeled as a coupe. Since the images for these categories do not differ a lot in shape but in details such as number of doors, it may be hard for neural net models to distinguish.

The major shortcoming of our models is that they do not have high validation and test accuracies. Since our dataset is not large enough and has an unbalanced number of images in different categories, it is difficult to train models. Additionally, the differences between each category are not so obvious as the difference between a cat and a car. Furthermore, within each category, there are different models of cars, and thus it becomes hard to only learn the features distinguishing classes. To collect more data, to improve background removal algorithms, and to try more pretrained models can be solutions for this limitation.

There is also an overfitting problem in our models as training accuracies are high but validation and test accuracies are low due to the small sample size. If possible, more new data, especially images of mini cars, can be collected to enrich the dataset. Additionally, data augmentation techniques such as rotation can be used to tackle this problem.

Furthermore, due to the limitation of computer CPU and limited usage of google colab, we only train the model for at most 50 epoches. If there is more time and better hardware available, we can train the model for longer epoches, such as 100, 1000, 10000, and even 100000. However, the overfitting problem could be another concern. We can also build deeper and larger neural networks to fit the data.

## 7. Conclusion

Based on the results, the Pretrained Resnet34 model has the best combined training, testing, and validation accuracy. People can now use this model to get fairly accurate predictions of the categories of cars when they try to buy a car. Furthermore, police can implement the algorithm of this model in traffic monitoring cameras and use this model to predict and recognize the categories of cars on the road to better determine traffic conditions and track down criminals by recognizing their cars. Finally, to make our predictions better and better serve other people, we can use a bigger dataset in the future so that we can achieve better accuracy for our model.

## 8. Acknowledgements

We got the dataset called Cars from Krause, Stark, Deng, and Li's research [2].

## 9. Contributions

We worked on the proposal together, and also researched and wrote codes together. We collaborated on writing Python codes in Jupyter Notebook and also met regularly to discuss the progress of our project and how we can improve our model and make our predictions more accurate. We finally worked on compiling the report together.

## References

[1] Kaleido. Remove background from image.

[2] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

[3] Y. Liu. Car-recognition, May 2018.