

# Car Classification using Neural Networks

Feifan Xu, A53253946, [fex014@ucsd.edu](mailto:fex014@ucsd.edu)

Ke Han, A53244302, [keh101@ucsd.edu](mailto:keh101@ucsd.edu)

Yifan Ruan, A99007477, [yir021@ucsd.edu](mailto:yir021@ucsd.edu)

Yincheng Mao, A53286835, [y7mao@ucsd.edu](mailto:y7mao@ucsd.edu)

**Abstract**—In this project, we used neural networks which helped us to classify 196 different brands of cars. With a relative small dataset, we implemented transfer learning in training process from models such as ResNet, VGG and MobileNet to make classification.

## I. INTRODUCTION

Car classification and detection are crucial to multiple fields including traffic management by government and vehicle business by companies. To enhance the accuracy and efficiency of classification, neural network is a promising method. In our project, we will implement several models to classify a dataset of car images and compare their performance. The input to our algorithm is an image of a car. We then use neural network to output a predicted brand of the car in the image.

## II. RELATED WORK

CNNs are widely used in CV field, especially in image classification area. It's task oriented and can learn the features needed itself. We aim to reproduce some of the work by using some common CNN models.

One of the previous work is done on caffe[1], their idea is based on transfer learning, use CaffeNet, GoogleNet and VGGNet with fine-tuning method.

Other improvements for the classification task focus on different parts: one is augment the dataset, using image research engine to find similar pictures[2]. Other improvements seeks to find new method about certain layers[3] or regularization method[2].

## III. DATASET AND FEATURES

The dataset we used is the Stanford Cars Dataset [4]. The Cars dataset contains 16,185 images of 196 classes of cars. 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. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

In this training set, we have images of different car classes named as numbers together with a csv file storing the annotated information corresponding to each image. We will separate the whole into training set and validation set while needed for neural network models by applying a random split under an 80-20 scale. Images in this dataset have different shapes and sizes. Therefore, they are in need of further processing to be fit in our models. In addition, as the images are taken with different backgrounds, a devkit, including class labels for training images and bounding boxes for all images, is also provided so that the cars needed to be classified can be figured out for the models.

First, we should check the original images in the cars dataset. Some examples of different classes are shown in Figure 1.



Fig. 1 Images from Training Set. Left to Right, Top to Bottom: 2012 Audi R8 Coupe, 2008 Acura TL Type-s, 2008 Dodge Magnum Wagon, 2007 Hyundai Elantra Sedan

Each class in the training set contains roughly 40 images. They are cars of the same class label with different colors, directions and backgrounds. Examples shown in Figure 2 are for the label: AM General Hummer SUV 2000.



Fig. 2 Images from the Class: AM General Hummer SUV 2000

From those examples in the dataset, it is clear that they are in different resolutions, calling for a need for the preprocessing. Therefore, in this project, we crop images to different sizes to fit our models. The examples shown in Fig. 3 are the images in Figure 1 cropped into  $224 \times 224$  pixels and the RGB scale normalized to  $[-1, 1]$ .

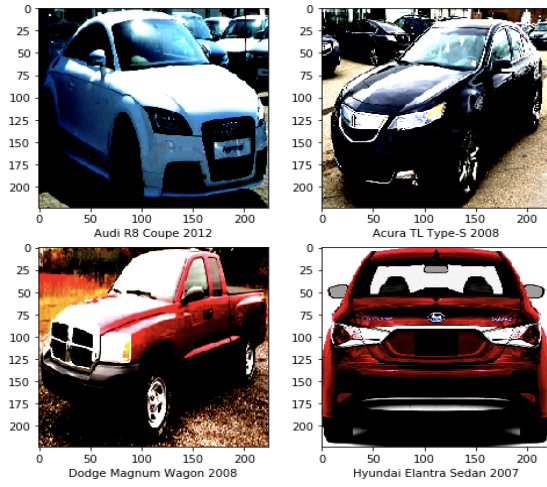


Fig. 3 Preprocessed Images of Examples Shown in Fig. 1

#### IV. METHODS

While we have finished some preprocessing, we are now ready to fit them into the models. We will develop a shallow CNN as a baseline for non transfer learning. And we will config several deep learning models for comparisons.

As the dataset contains about 16000 images, it is not large enough to fully train the deep learning models. In addition, due to the complexity of deep learning neural networks, we would perform the transfer learning to train those models. Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. It is an optimization that allows rapid progress or improved performance when modeling the second task. Therefore, it is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained. In this project, we implement the following 2 deep learning networks: ResNet and VGG.

##### A. Shallow CNN (Baseline)

The baseline model is a simple convolutional neural network, the structure is as below:

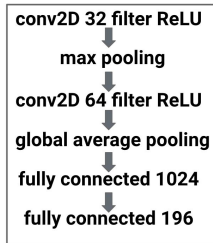


Fig. 4 Structure of Baseline

##### B. MobileNet

MobileNet[5] is a class of efficient model for mobile and embedded version of applications. The main idea of MobileNet is to divide traditional convolution layers into depthwise convolution layer and pointwise convolution layer. Consequently, the computation is reduced dramatically thanks to smaller number of parameters compared to VGG.

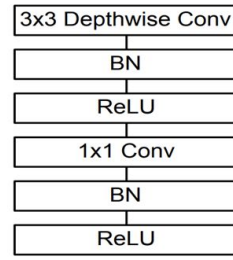


Fig. 5 Architecture of MobileNet

##### C. VGG16

The VGG network [6] architecture was introduced by Simonyan and Zisserman [6] in 2014. This network is characterized by its simplicity, using only  $3 \times 3$  convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier. The “16” stands for the number of weight layers in the network. Aside from the straight-forward architecture, VGGNet consists of 138 million parameters, which is challenging to train. In order to make training easier, VGG needs to be trained from smaller networks and use them as initialization (pre-training). While the whole VGGNet is slow to train, we will use the pre-trained models. The figure below shows the architecture of VGG.

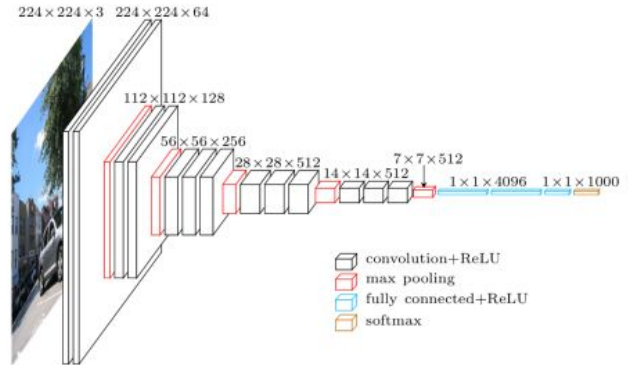


Fig. 6 A Visualization of the Architecture of VGG Network

##### D. ResNet

Kaiming He et al introduced Residual Neural Network [7], the so-called ResNet, in 2015. It is well known as the solution to solve the network degrading problem.

First, we need to take a look at the degradation problem. When deeper networks starts converging, with the network depth increasing, accuracy gets saturated and then degrades rapidly. As a result, naively adding layers would lead to higher training errors. In some worst case scenarios, while accuracy of shallow networks is saturated, the deeper networks is degrading the model.

To solve this, the authors introduced the Residual function using  $F(x)=H(x)-x$  instead of a direct mapping of  $H(x)$ . In the network, this is transferred to a Residual block:

$$y = F(x, \{W_i\}) + x, \quad (1)$$

while the function  $F(x, \{W_i\})$  is the residual mapping function we need to train.

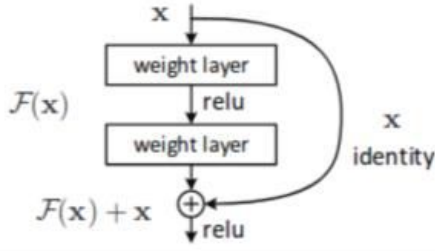


Fig. 7 A Building Block of Residual Learning

The design of ResNet networks uses  $3 \times 3$  filters mostly, downsampling with CNN layers with stride 2 and consists of global average pooling layer and a 1000-way fully-connected layer with Softmax in the end.

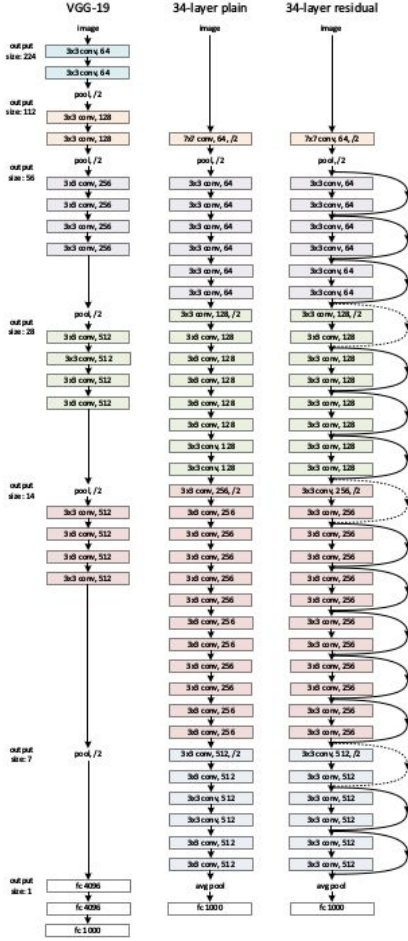


Fig. 8 Plain VGG and VGG with Residual Blocks

The figure above shows a plain 34-layer network and a 34-layer residual network [8]. From testing, the plain 34 layer network has higher validation error than the 18 layers plain network. This is where we realize the degradation problem. And the same 34 layer network when converted into the residual network has much lesser training error than the 18 layer residual network. Thus, ResNet models can have an incredibly high depth up to 152. In this project, we will implement several ResNet models with different depths.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	$112 \times 112$	7x7, 64, stride 2				
conv2.x	$56 \times 56$	3x3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3.x	$28 \times 28$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4.x	$14 \times 14$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5.x	$7 \times 7$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	$1 \times 1$	average pool, 1000-d fc, softmax				
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

Fig. 9 ResNet Architectures with Different Depths of Layers

## V. EXPERIMENTS

Some of the models have an incredibly high number of parameters, for example, VGG16. They are too complicated for our dataset and might cause the overfitting problem. To achieve better performance, we would need some data processing.

### A. Transfer Learning

Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. It is an optimization that allows rapid progress or improved performance when modeling the second task. Therefore, it is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained. Due to the difference between problems, we would need to freeze all but the last layers to make the layer weights invariant. In this project, we download the pre-trained models of VGG and ResNet and apply them to our image sets for time saving and better performance.

### B. Stochastic Gradient Descent (SGD)

Stochastic gradient descent (often abbreviated SGD) [9] is an iterative method for optimizing an objective function with suitable smoothness properties. It is called stochastic because the method uses randomly selected (or shuffled) samples to evaluate the gradients.

Instead of computing the gradient directly, SGD estimates the gradient adding a randomly-chosen example in each iteration with a stochastic process  $w_t, t=1,2,\dots$ :

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t), \quad (2)$$

As SGD is generally noisier than typical Gradient Descent, it usually took a higher number of iterations to reach the minima, because of its randomness in its descent. Even though it requires a higher number of iterations to reach the minima than typical Gradient Descent, it is still computationally much less expensive than typical Gradient Descent.

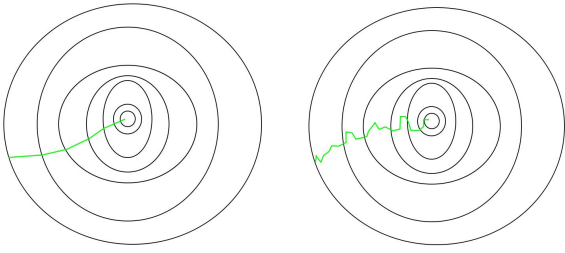


Fig. 10 Path Taken by Batch Gradient Descent (Left) and Stochastic Gradient Descent (Right)

### C. Adam Optimization

Adam [11] is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based on training data. The name Adam is derived from adaptive moment estimation [6]. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. In this project, learning rate would influence the performance of our models, as when the learning rate is too low, models take a long time to converge; when the learning rate is too high, there would be a lot of fluctuations in losses or might even diverge. As a result, we need an optimizer to help us maintain the learning rates during training and we choose Adam optimization.

### D. Cross-Entropy Loss

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. In multiclass classification ( $M > 2$ ), we calculate a separate loss for each class label per observation and sum the result:

$$Loss = \sum_{c=1}^M y_{o,c} \log(p_{o,c}), \quad (3)$$

where  $M$  is the number of classes,  $y$  is the binary indicator (0 or 1) if class label  $c$  is the correct classification for observation  $o$ , and  $p$  is the predicted probability observation  $o$  is of class  $c$ .

## VI. RESULTS

### A. Shallow CNN (Baseline)

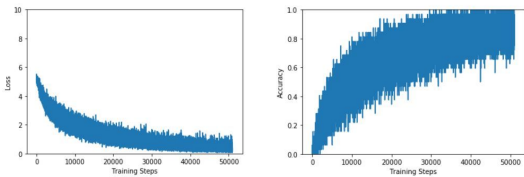


Fig. 11 Loss (Left) and Accuracy (Right) of Baseline

### B. MobileNet

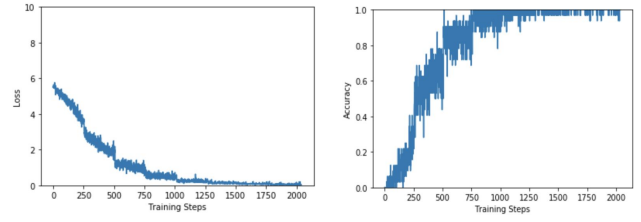


Fig. 12 Test Loss (Left) and Test Accuracy (Right) of MobileNet

### C. VGG16

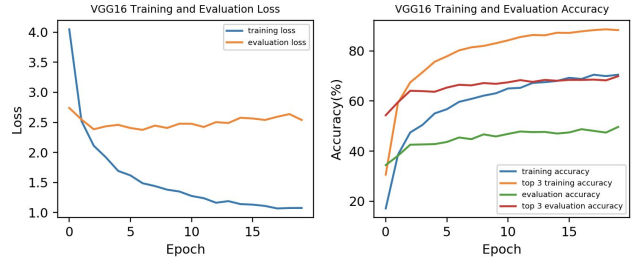


Fig. 13 Loss (Left) and Accuracy (Right) of VGG16

### D. ResNet

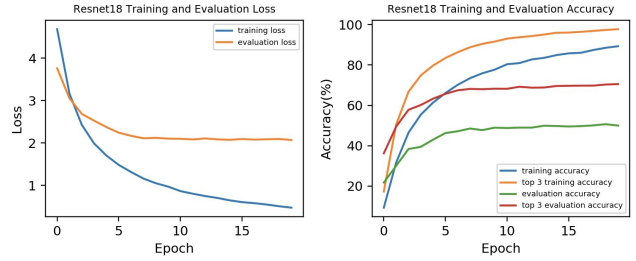


Fig. 14 Loss (Left) and Accuracy (Right) of ResNet18

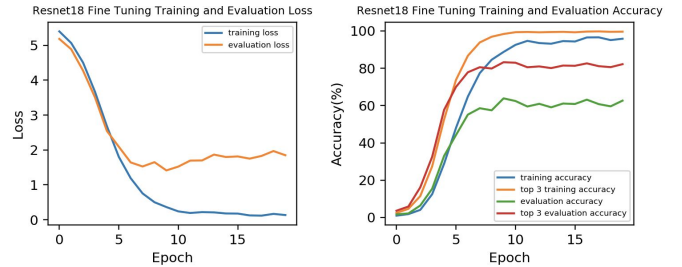


Fig. 15 Loss (Left) and Accuracy (Right) of ResNet18 (Fine-Tune)

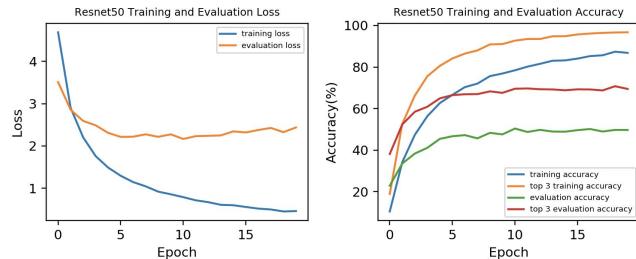


Fig. 16 Loss (Left) and Accuracy (Right) of ResNet50

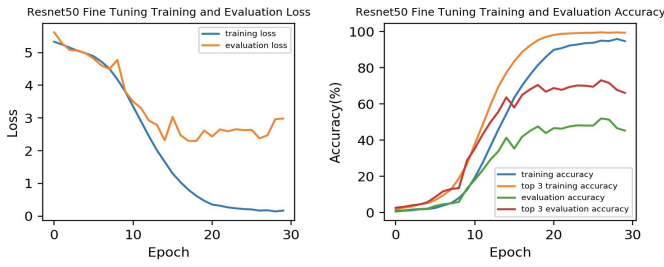


Fig. 17 Loss (Left) and Accuracy (Right) of ResNet50 (Fine-Tune)

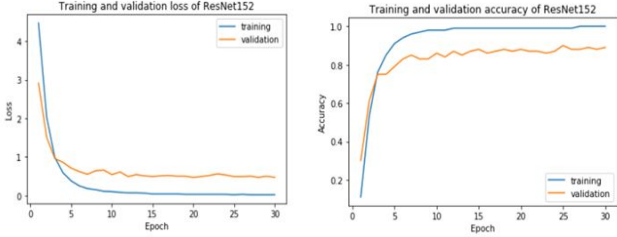


Fig. 18 Loss (Left) and Accuracy (Right) of ResNet152

### E. Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known, with the horizontal axis to be the actual class and the vertical axis to be the predicted class.

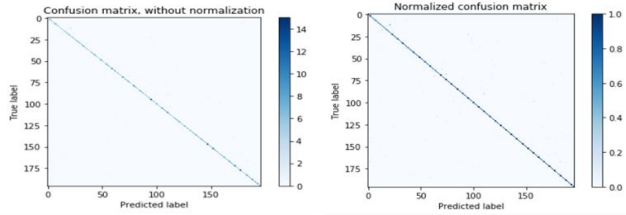


Fig. 19 Confusion Matrix without (Left) and with (Right) Normalization

Figure 19 shows the confusion matrix corresponding to the ResNet152 model. It is clear that the confusion matrix mostly concentrate on the diagonal, indicating that most of the predicted class labels correspond to the actual classes, which is an indication that the accuracy of the model is fairly satisfactory.

### F. Overall Results

TABLE I. OVERALL LOSS AND ACCURACY

Models	Loss (Train/Test)	Accuracy (Train/Test)	Top-3 Accuracy (Train/Test)
Baseline	1.0000/-	0.8600/-	-
MobileNet	0.9063/-	0.9986/0.7827	-
VGG16	1.0756/2.5395	0.7051/0.4966	0.8831/0.6990
ResNet18	0.4723/2.0685	0.8931/0.4996	0.9777/0.7056
ResNet18 (Fine-tune)	0.1361/1.8493	0.9580/0.6262	0.9958/0.8216
ResNet50	0.4600/2.4387	0.8684/0.4971	0.9672/0.6946

Models	Loss (Train/Test)	Accuracy (Train/Test)	Top-3 Accuracy (Train/Test)
ResNet50 (Fine-tune)	0.1723/2.9833	0.9473/0.4534	0.9939/0.6615
ResNet152	0.0192/0.5024	0.9963/0.8950	-

## VII. CONCLUSION & FUTURE WORK

From the results, we have the following conclusions:

1. baseline is easy to train but hard to converge, this may due to the structure of the network or the global pooling layer.
2. MobileNet and Resnet18 with fine-tuning can achieve a relatively good classification test accuracy. This may come from the fact that MobileNet and ResNet18 have less parameters which are easier to implement gradient descent given a relatively small number of samples of each class. This can be shown from training of ResNet50 with fine-tuning. The loss and accuracy has zig-zag shape due to limitation of dataset size.
3. For the training of ResNet152, we also implemented some data augmentation which enlarges the training set and aim to make the model more robust by randomly adding flips and rotations. As a result, ResNet152 achieves a fairly satisfactory validation and test accuracy in a reasonable amount of time.

Based on the results and conclusions, we plan to improve our methods in the following ways:

1. Since some of the details of cars are hard to be detected for networks, we plan to use different Loss Function such as Focal Loss
2. We will test the performance of image augmentation on other ResNets we have used.
3. The classifier attached after the feature extractors can also be modified, such as adding dropout, to be more generalized.

### Contributions:

1. Feifan Xu: Check related work, build baseline model, write report.
2. Ke Han: Experiment on MobileNet, ResNet18, ResNet50, VGG16, write report.
3. Yifan Ruan: Experiment on ResNet50, ResNet152, write report
4. Yincheng Mao: Write Report and poster.

## REFERENCES

- [1] Liu D, Wang Y. Monza: image classification of vehicle make and model using convolutional neural networks and transfer learning[J]. 2017.
- [2] Xie S, Yang T, Wang X, et al. Hyper-class augmented and regularized deep learning for fine-grained image classification[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 2645-2654.
- [3] Q. Hu, H. Wang, T. Li and C. Shen, "Deep CNNs With Spatially Weighted Pooling for Fine-Grained Car Recognition," in *IEEE*

- [4] Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei, "3D Object Representations for Fine-Grained Categorization," 4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13). Sydney, Australia. Dec. 8, 2013.
- [5] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
- [6] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
- [7] He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep residual learning for image recognition". In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- [8] Prakash Jay. "Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook".
- [9] Leon Bottou. "Large-Scale Machine Learning with Stochastic Gradient Descent".
- [10] Jason Brownlee. "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning"
- [11] Diederik P. Kingma, and Jimmy Lei Ba. "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION" arXiv preprint arXiv:1412.6980v9 (2017).