Building a Fine-Grained Vehicle Classifier from a Small Image Dataset Using Convolutional Neural Networks and Transfer Learning

Anders D. Sleire

University of Bergen/ Tryg Forsikring

31.01.2018

Final assignment in PhD course *Deep Learning*,

Technical University of Denmark

Abstract

Convolutional Neural Networks (CNNs) have proven to be very successful in a wide range of image recognition problems, but they require large amounts of data and computational power. In many practical applications, the volume of data to learn from is limited, making training a deep CNN from scratch unfeasible. This may be particularly challenging in problems where there are relatively small differences between the objects we try to identify, so called fine-grained classification problems. One approach to solving such tasks is to modify deep pre-trained networks via a process called transfer learning. In this paper, we pick some of the models that have shown good performance in the ImageNet Challenge, and make use of a transfer learning technique called fine tuning. The final models are used for classification of car *Make, Model* and *Year* for 15 of the classes in the Stanford Cars Dataset.

Keywords: Convolutional neural networks, transfer learning, fine grained classification

Project source code:

https://github.com/sleire/02456-deep-learning_spes

Contents

Abstract 2

Introduction 4

The Cars Dataset 5

Theoretical background and previous studies 6

Convolutional Neural Networks 6

Transfer learning 7

Some previous work 8

Implementation 8

Data preprocessing and augmentation 8

Baseline model 10

Fine-tuned VGG models 10

Results 12

Summary and conclusions 15

References 16

Introduction

In deep learning, the Convolutional Neural Networks (CNNs) have proven to be highly successful in a wide range of image recognition problems. Artificial Neural networks are not new, but recent advances in computational power and access to large datasets have made us able to train multi-layered (deep) neural networks to lift performance. One of the important contributions is ImageNet (Deng, et al., 2009), a large-scale hierarchical image database developed by researchers at the computer science department at Princeton University. As of 30 January, 2018 the ImageNet database contains more than 14M images, where over 1M are annotated with labels and bounding boxes (ImageNet, 2018). Datasets such as this have helped lead us to where we are today, where traditional hand-engineered image recognition methods have been surpassed by *machine learning* methods that can learn from large volumes of images, and then generalize to solve problems with new data.

One particularly challenging task in image based classification is situations where the different classes of objects do not differ greatly from each other. There might be subtle differences between two groups, and they might be difficult to tell apart for a human non-expert. A machine learning algorithm would typically need a large volume of training data to be able to classify objects in such a dataset correctly. In many cases, a big dataset is not available, or expensive to collect. This is where transfer learning techniques can help. Distinguishing between different car brands and models is an example of fine grained classification. Even though the large CNNs that have been trained on ImageNet only distinguish between broader categories such as “truck” and “station wagon”, they can still be used to classify cars into a more detailed set of groups. This is the topic for this paper.

In the following, we will give an introduction to the dataset and describe the 15 cars selected for the classification task in further detail. Furthermore, we provide a short description of how CNNs can be modified with transfer learning techniques when dealing with small volumes of training images. We also describe how data preprocessing and data augmentation techniques can increase the quality and volume of our training data, and thereby lift performance. Finally, we implement a baseline model to be trained from scratch along with two deep CNNs trained on ImageNet to be fine-tuned for cars dataset. We evaluate the classification performance of the models and provide some suggestions for future work.

The Cars Dataset

The *Cars* Dataset (Krause, Stark, Deng, & Fei-Fei, 2013) is a well-known image dataset for fine grained classification. It consists of 16,185 images of 196 car classes. It is a balanced dataset, with 8,144 training images and 8,041 testing images. Classes are labelled *Make, Model, Year*, e.g. Audi R8 Coupe 2012. The quality of the pictures varies considerably. Some are taken by professional photographers, while others are of poor quality. There is also large variation in the positioning and orientation, and many pictures include a lot of the surrounding environment.

Figure 1. Images from the Cars Dataset, http://ai.stanford.edu/~jkrause/cars/car_dataset.html

Index	Label	Train images	Test images
0	AM General Hummer SUV 2000	45	44
1	Aston Martin V8 Vantage Convertible 2012	45	45
2	Audi R8 Coupe 2012	43	43
3	Dodge Sprinter Cargo Van 2009	40	39
4	FIAT 500 Abarth 2012	28	27
5	Ford Focus Sedan 2007	45	45
6	GMC Terrain SUV 2012	42	41
7	Jeep Patriot SUV 2012	44	44
8	Lamborghini Aventador Coupe 2012	44	43
9	MINI Cooper Roadster Convertible 2012	37	36
10	Mazda Tribute SUV 2011	36	36
11	Mercedes-Benz 300-Class Convertible 1993	48	48
12	Mitsubishi Lancer Sedan 2012	48	47
13	Nissan Leaf Hatchback 2012	42	42
14	Rolls-Royce Ghost Sedan 2012	39	38

Table 1. The 15 car classes, with number of train/test images

Theoretical background and previous studies

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a group of neural networks with an architecture suitable to make use of a 2D structure of image data.

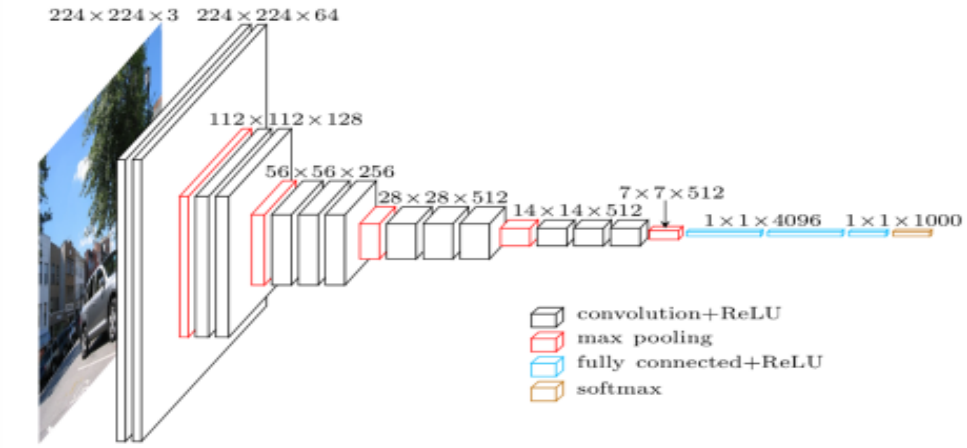


Figure 2. Deep CNN architecture for ImageNet, (Frossard, 2016)

Some of the main building blocks of the CNN are the convolutional and subsampling layers, the fully connected layers and an activation layer, e.g. softmax. In the above figure, we see that the CNN has been designed for the 1000 classes in ImageNet. For our problem, the final output will be predictions for the 15 classes of cars. When training a CNN, we learn the convolutional kernels (filters) and corresponding shared weights. The kernels act as feature detectors, at different levels of abstraction. The lower levels such as lines and edges can be relevant in a large range of applications. Higher level features can be too domain/ data specific to be useful elsewhere.

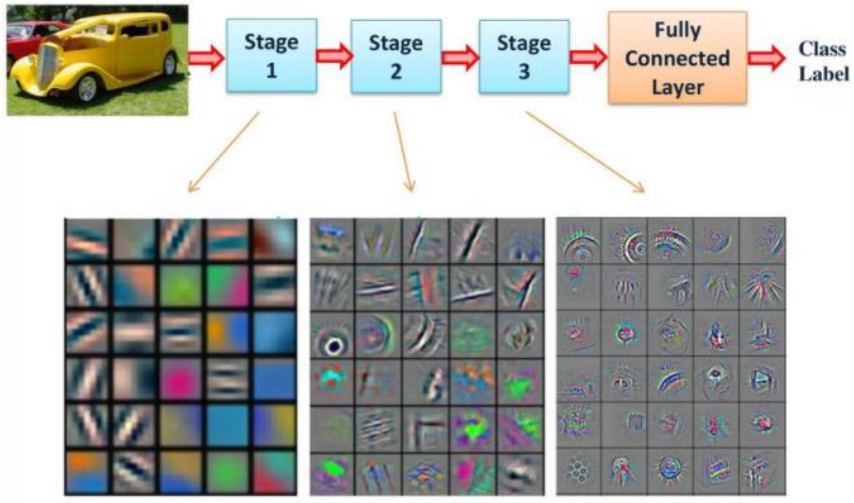


Figure 3. Convolutional kernels learned by a CNN from ImageNet (Zeiler & Fergus, 2014)

Transfer learning

In transfer learning, we take pre-trained models (model architecture and weights learned from a specific dataset), and modify them by estimating the weights in higher level layers again, with our data. The strategy is to keep the layers in the original model that is expected to be useful in the new problem by freezing them, and retraining the rest. As we saw above, the first and

intermediate layers in the CNN consists of detectors of lower level features like curves and edges. A large model that has been trained on ImageNet is likely better at detecting low level features than an alternative model trained from scratch on thin data. Transfer learning is useful when data is limited, but is also appealing due to the relatively low effort required to build high-performing models. One possible pitfall is the case where the original model is trained on a completely different domain, such as for example medical imaging data. In our case with ImageNet and the Cars Dataset, this is not expected to be a problem.

Some previous work

Fine-grained classification problems have been studied in many domains, for example in biology, for classification of species of fish, plants and birds, see for example (Berg, et al., 2014). There have also been studies on car image data. See in particular (Liu & Wang, 2017) where the authors apply transfer learning techniques for classification of all the 196 classes in the Cars Dataset with good results.

Implementation

Data preprocessing and augmentation

Due to the differences in image quality and variation in how much of the surrounding environment that is visible around the vehicle, we do some preprocessing of the images before they are fed to the training algorithms. We remove the parts of the original image outside a bounding box defined by a set of coordinates (x1, y1) and (x2, y2). In real-life applications, we

would have to apply an object detection algorithm such as YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016), but in this problem, the coordinates are given to us.

Figure 4. Example image, original and cropped version.

After cropping the images based on the bounding boxes, we apply data augmentation techniques to increase the size of the training data. Using the *ImageDataGenerator* function of the *Keras* framework, we generate new batches of image data from the original with operations such as

- Horizontal flipping
- Rotation change
- Height shift
- Width shift
- Zoom range

These are utilized in training, along with rescaling to normalize the input data. We increase the number of training examples considerably, but the test is of the original size of 618 images. This is a convenient way to generate more data, but it should be used with caution to generate training data that is relevant for the test set/ real-life classification task.

Baseline model

For comparison, we fit a CNN without utilizing any pre-trained model. It consists of two blocks of Conv-ReLu-MaxPool-Dropout layers, followed by a fully connected layer with softmax loss function. Since this model will only be trained on the small image data set from the 15 car classes, we do not expect it to perform as well as the large, fine-tuned models.

Fine-tuned VGG models

The VGG network architecture was developed by the Visual Geometry Group at the University of Oxford, and introduced in (Simonyan & Zisserman, 2014). They stack 3x3 convolution layers in blocks of increasing depth, followed by max pooling and three fully connected layers and softmax activation for classification. The VGG architecture is also illustrated in *Figure 2* above.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2. Alternative VGG configurations, grouped in columns (Simonyan & Zisserman, 2014)

Here, we will be working with the *VGG16* and *VGG19* models described in column D and E in the table from Simonyan and Zisserman. To perform fine-tuning, we will follow a procedure inspired by (Chollet, 2016) where we:

1. **Use the VGG models as feature extractors,** by loading all layers below the fully-connected layers and running this model on our data. The output from this process are the last activation maps before the fully-connected layers, the "bottleneck features". These are stored as numpy arrays to be used as input data in step 2).
2. **Train a small fully-connected model on top of the extracted features.** This model has a fully connected layer corresponding to the number of classes in our problem (15 cars). The estimated weights are stored as a .H5 file, to be used in step 3).
3. **Add the fully-connected model to the original VGG and retrain the final layers.** By loading the estimated weights of the top model from step 2) and freezing the lower levels of the VGG architecture, we can keep (transfer) the lower level feature detectors learned from ImageNet, while estimating the rest. We can choose to retrain larger parts of this final VGG model, but here we will *fine tune* it by retraining only the last layers. The final product is stored as a .H5 file for future use.

Results

Model	Train accuracy	Test accuracy	Train loss	Test loss
Baseline	0.0719	0.0728	14.9594	14.9444
VGG16_tuned	1.0	0.9612	0.0031	0.1846
VGG19_tuned	1.0	0.9612	0.0028	0.1871

Table 3. Accuracy and loss for the training and test sets.

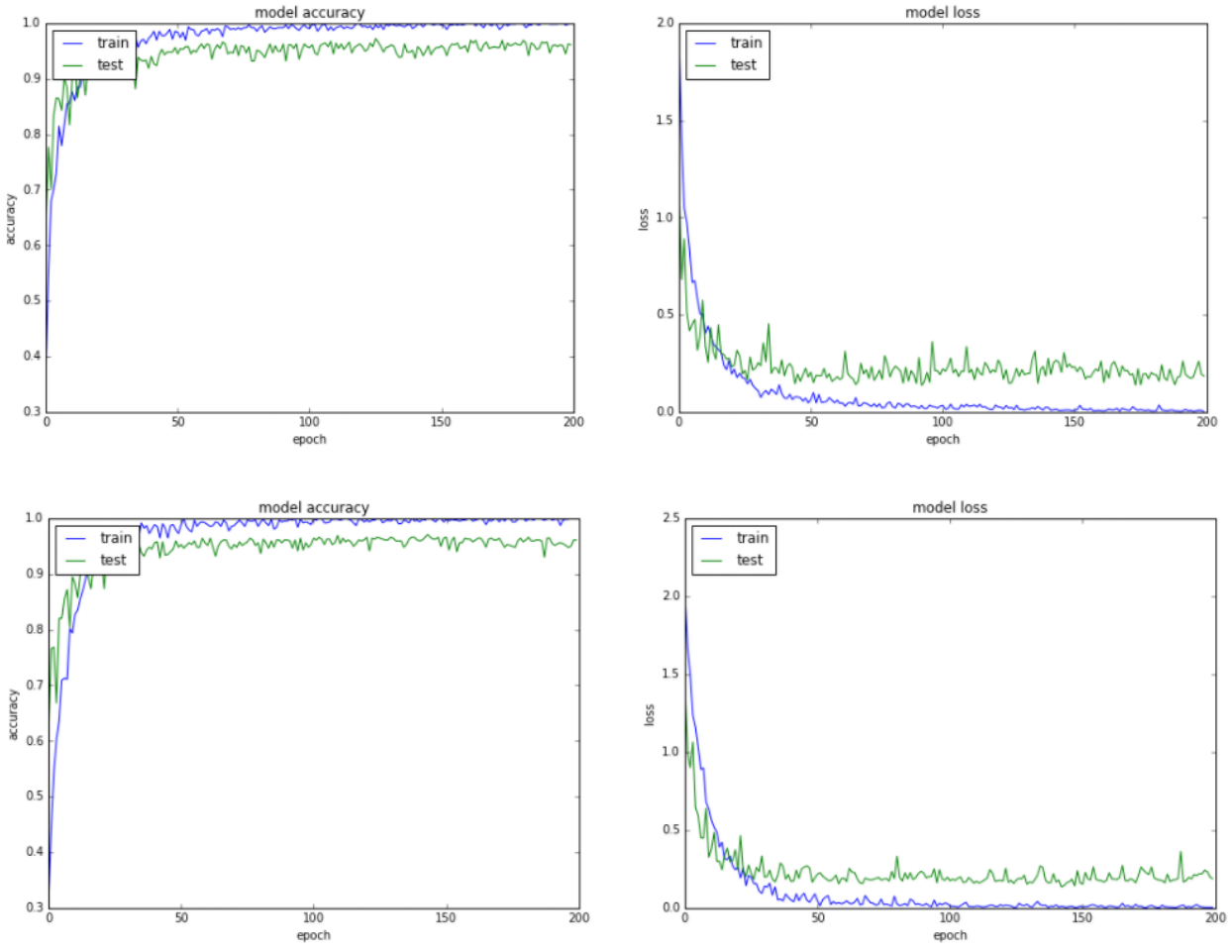
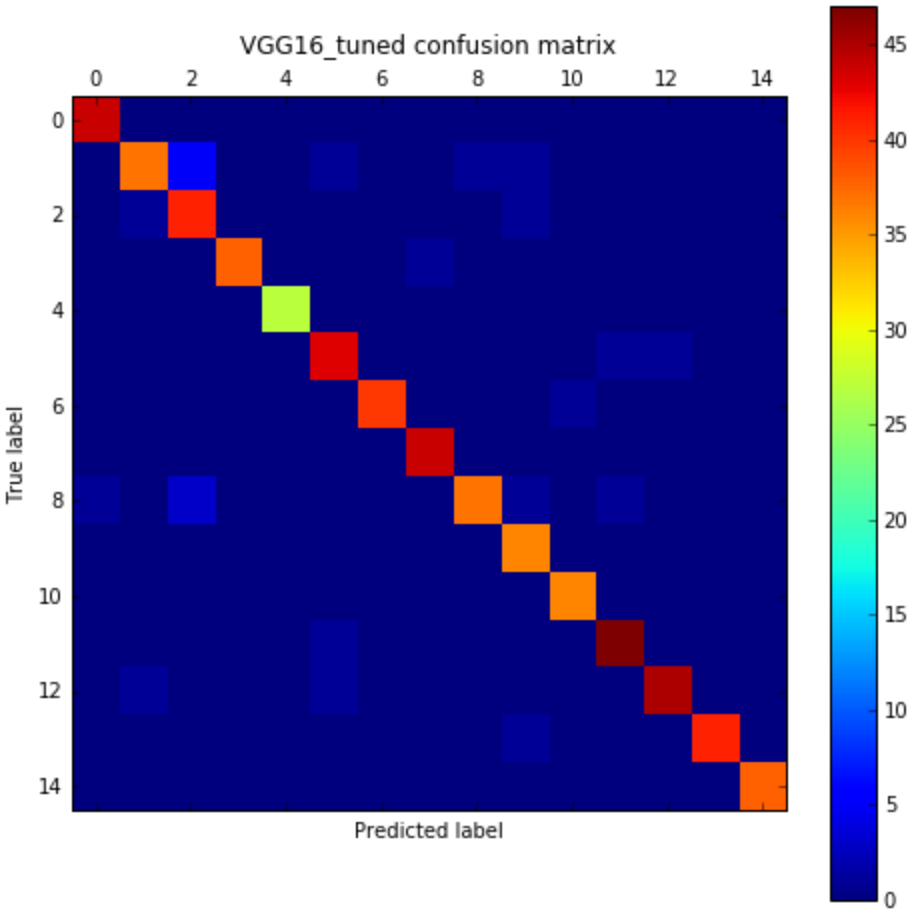
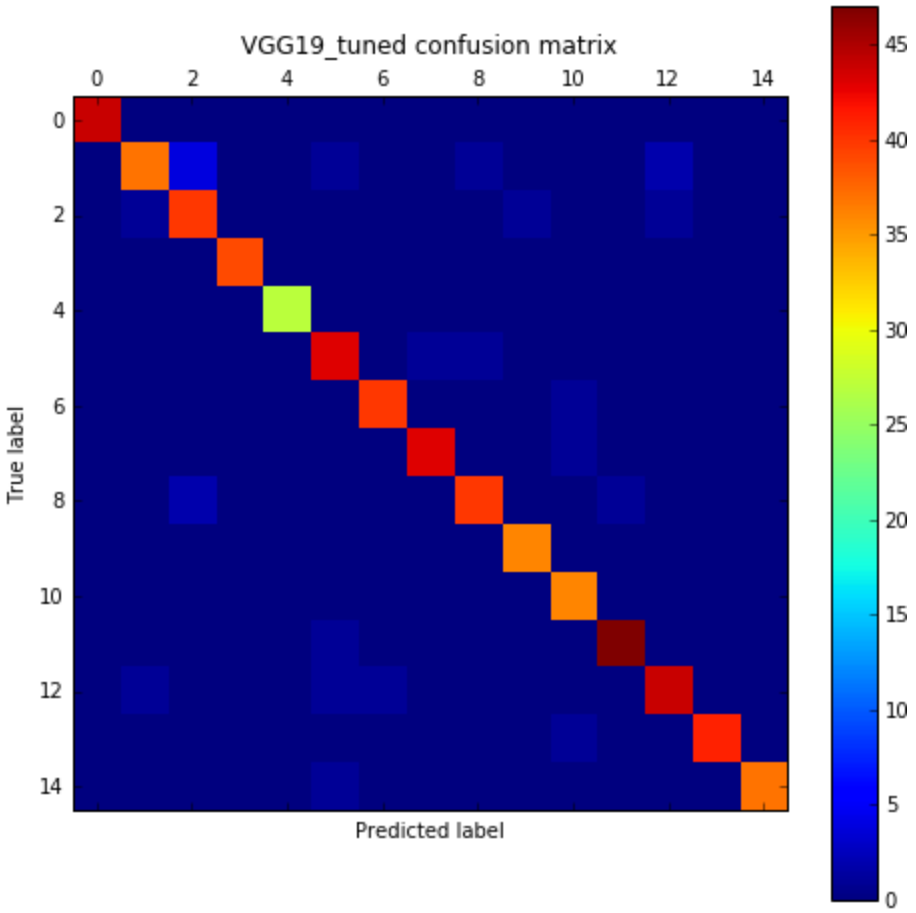


Figure 5. Train/ test loss and accuracy for VGG16 (top) and VGG19 (bottom).




```
[[44  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 37  5  0  0  1  0  0  1  1  0  0  0  0  0]
 [ 0  1 41  0  0  0  0  0  0  1  0  0  0  0  0]
 [ 0  0  0 38  0  0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0 27  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 43  0  0  0  0  0  1  1  0  0]
 [ 0  0  0  0  0  0 40  0  0  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  0 44  0  0  0  0  0  0  0]
 [ 1  0  3  0  0  0  0  0 37  1  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 36  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 36  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0 47  0  0  0]
 [ 0  1  0  0  0  1  0  0  0  0  0  0 45  0  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0  0 41  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 38]]
```

Figure 6. VGG16_tuned confusion matrix, index corresponding to *Table 1*.




```
[[44  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 37  4  0  0  1  0  0  1  0  0  0  2  0  0]
 [ 0  1 40  0  0  0  0  0  0  1  0  0  1  0  0]
 [ 0  0  0 39  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 27  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 43  0  1  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 40  0  0  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  0 43  0  0  1  0  0  0  0]
 [ 0  0  2  0  0  0  0  0 40  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 36  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 36  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0 47  0  0  0]
 [ 0  1  0  0  0  1  1  0  0  0  0  0 44  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0 41  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  0  0 37]]
```

Figure 7. VGG19_tuned confusion matrix, index corresponding to *Table 1.*

Summary and conclusions

Fine grained classification problems can be challenging, but in this particular case, we have been able to build models with good performance by fine tuning deep pre-trained models that are available online for anyone to download. As expected, the baseline model built from scratch did not perform very well, which is probably due to lack of data to learn lower level features from. With small data and limited computational resources, it can be sensible to transfer components from other models that have been trained on larger data that have similarities to ours.

A natural next step after this work would be to develop models to classify all 196 cars in the dataset. We have not done that here, due to time constraints. It would also be sensible to investigate further the cases where the models perform misclassifications, to see if these can be corrected. One approach could be to look closer at Bayesian deep learning methods, that can quantify the uncertainty in classification. This could help making deep learning less of a "black-box". Having information about the degree of certainty in classification can also be helpful in many practical situations, where other models/ ensemble techniques or human experts can be involved.

References

Berg, T., Liu, J., Lee, S., Alexander, M., Jacobs, D., & Belhumeur, P. (2014). Birdsnap: Large-scale fine-grained visual categorization of birds. *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference*, 2019-2026.

Chollet, F. (2016, 6 5). *Building powerful image classification models using very little data*. Retrieved from Keras Blog: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition*, 248-255.

ImageNet. (2018, 01 30). *Stats*. Retrieved from ImageNet: http://image-net.org/about-stats

Krause, J. (2018, 01 31). *Cars Dataset*. Retrieved from Cars Dataset: http://ai.stanford.edu/~jkrause/cars/car_dataset.html

Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference*, 554-561.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 1097-1105.

Last Name, F. M. (Year). *Book Title.* City Name: Publisher Name.

Liu, D., & Wang, Y. (2017). Monza: image classification of vehicle make and model using convolutional neural networks and transfer learning.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779-788.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv: 1409.1556.*

Zeiler, M., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *European conference on computer vision*, 818-833.