Monza: Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning

Derrick Liu
Stanford University
lediur@stanford.edu

Yushi Wang
Stanford University
yushiw@stanford.edu

Abstract

Car detection and identification is an important task in the area of traffic control and management. Typically, to tackle this task, large datasets and domain-specific features are used to best fit the data. In our project, we implement, train, and test several state-of-the-art classifiers trained on domain-general datasets for the task of identifying the make and models of cars from various angles and different settings, with the added constraint of limited data and time. We experiment with different levels of transfer learning for fitting these models over to our domain. We report and compare these results to that of baseline models, and discuss the advantages of this approach.

1. Introduction

Generally speaking, visual fine-grained classification can be very challenging due to more subtle differences between classes, compared to basic recognition or coarse classification, such as on ImageNet. Recognizing the makes and models for cars is one such task. For humans, this is usually a fairly straightforward task, especially for car aficionados. Cars can usually be identified by human eye due to certain key aspects, such has logos, hood ornaments, or lettering. However, due to the visual complexity of cars, this has traditionally been a hard task for computers.

The main challenge for fine-grained classification is unarguably the very fine differences between different classes. Typically, to learn these minute differences, a large dataset is needed. However, in a setting with limited time, computational power, or data, this is not feasible.

In our project, we design, implement, and test a lightweight end-to-end system that uses an out of the box deep learning framework to fine-tune pre-trained classifiers for a specific fine-grained classification test. Our approach is based on taking deep learning models trained on ImageNet, which typically have very general features, and changing as little as possible to fit our training data. We investigate the effects of varying the levels of tuning on the performances of these fine-tuned classifiers.

We conducted experiments on the Cars dataset [9], a fine-grained dataset containing 196 different classes of cars. This dataset is particularly challenging due to the freeform nature of the images, which contained cars in many different sizes, shapes, and poses. Despite our resource limitations and the difficulty of the task, we were able to obtain very high quality results from fine-tuning.

2. Previous Work

2.1. CNNS

In recent years, much work on image processing and classification has been done with convolutional neural networks (CNNs). The power of CNNs is their capacity for learning not only the weights of features, but the features themselves as well. Recently, these CNNs have achieved state of the art accuracy on generic image classification [11]. In this project, we make extensive usage of CNNs as our primary architecture of classifiers.

2.2. Transfer Learning

Transfer learning is a machine learning technique that focuses on repurposing learned classifiers for new tasks [10]. In transfer learning for CNNs, a base network is trained on a base dataset to create weights and features. This classifier is then transferred to a new dataset by retraining a subset of the base network's learned weights and features. The overall effect is a classifier that fits the new dataset with significantly less work than retraining a new network.

When the target dataset is significantly smaller than the base dataset, transfer learning can be a powerful tool to enable training a large target network while minimizing overfitting. In certain tasks, transfer learning has been shown to achieve near state of the art results [12].

2.3. Fine-Grained Classification

There have been many investigations on fine-grained classification in a variety of fields, such as birds [5], plants [6], and cars [1], most of which use CNNs. However,

previous works in the specific task of identifying car make and model have usually involved a single or small fixed number of viewpoints [1, 2]. Furthermore, for most tasks, the number of examples per classification is usually quite large to allow for good generalization accuracy.

3. Approach and Algorithms

To alleviate the data and time constraints imposed on us, we chose an approach focused around using transfer learning to quickly create and train neural networks. For comparison, we also implemented two simple baseline networks. We used Caffe, a deep learning framework [4], to construct, train, and test our networks. The following subsections describe the models used in this project.

3.1. Baselines

We implemented two simple baselines: an SVM and a 1-layer CNN. Our baseline SVM setup consists of a single fully connected layer with softmax loss. This baseline provides a reference for the performance of a simple non-conv-net setup.

Our baseline convolutional neural network consists of a Conv-ReLU-Pool set, followed by a fully-connected layer with softmax loss. This baseline provides a reference for the performance of a simple CNN approach.

3.2. CaffeNet

The CaffeNet CNN model [4] is a replication of the AlexNet model [11]. AlexNet was originally designed to classify over ImageNet, and contains 5 convolutional and 3 fully connected layers. In addition, it uses dropout to avoid overfitting. CaffeNet is more or less identical to AlexNet, with a few minor differences in its default hyperparameters.

3.3. GoogLeNet

GoogLeNet was designed to be a direct improvement over AlexNet for the task of classifying ImageNet [7]. It has 22 layers, compared to AlexNet and CaffeNet's 8 layers, though the number of parameters in the model is purportedly 12 times smaller, due to the smaller number of weights per fully connected layer.

GoogLeNet's model generates 3 outputs for each input, at various depths. However, for the sake of brevity, we only use results from the last output, as it became apparent early on that the performance of the first two outputs tended to be strictly worse.

3.4. VGGNet

VGGNet was an attempt to improve upon the original AlexNet design by adding many layers, similar to GoogLeNet, albeit not as compact in terms of number of parameters. The architecture consists of multiple stacks of convolutional layers, interspersed with several max pools. Like the previous two networks, this was designed to classify over ImageNet [8].

4. Data

4.1. Source

We exclusively use the Cars dataset provided by the paper 3D Object Representations for Fine Grained Categorization by Jonathan Krause, et al. This dataset contains 16,185 image-classification pairs of 196 different classes, split into 8,144 training and 8,041 test images. Each of the 196 classes is very fine-grained on the order of year, make and model of a vehicle.

Although the classes are fine-grained, each class is visually distinct from one another; for example, the dataset contains a 2012 Volkswagen Golf and a 1991 Volkswagen Golf, which are visually very distinct, relatively speaking, but it does not contain a 2011 Volkswagen Golf, which is virtually identical to the 2012 model.

Figure 1: A sample of images from the Cars dataset, demonstrating the range of cars, image type, and image quality.

Each image consists of a car in the foreground against various backgrounds and viewed from various angles. The quality of each image, as described by characteristics like the focal length, lighting, and positioning of the car and camera, varies significantly from image to image – some images are professionally-taken press shots; others are relatively low-quality images collected from classifieds ads and other places on the internet.

4.2. Preprocessing

We created our preliminary training and validation sets by taking a stratified 1-fold of the provided training set, which split the provided training set 80-20 into two sets with the same class distribution as the provided training set. To exclude extraneous noise in the training data, we then cropped the images using bounding boxes provided with the dataset that describe the location of the cars actually present in each image. To preserve some context surrounding the cars, we expanded each bounding box by 16 pixels on each side before cropping.

As the training set contains a variety of image dimensions and aspect ratios, we resized each cropped

image to a square aspect ratio and a resolution of 227x227 as required by the models. After discussions with Krause, we decided to squash images without preserving their original aspect ratios instead of scaling and cropping the image.

5. Experiments

For our baseline SVM and single-layer conv-net models, we performed a single experiment that applied each model to raw pixel data for 50,000 iterations with a learning rate of 0.001 and a decay rate of 0.9. This produced an acceptable baseline for which to compare the more complex conv-net models to.

For CaffeNet and GoogLeNet, we performed four experiments each. Since GoogLeNet has multiple Softmax loss outputs placed at different depths of its network, each of the fine-tuning experiments below affected the learning rates for the layers before each of the loss outputs.

Fine-tuned last layer After pre-initializing each of the networks with ImageNet-trained weights, we adjusted the learning rates of the models so the last fully-connected layer learned at normal rates while the other layers learned at a diminished 0.1 rate. In our results, this is represented by the "fine-tuned" figures.

Fine-tuned last three layers After pre-initializing each of the networks with ImageNet-trained weights, we took the last-layer fine-tuning from the previous experiment and expanded it to apply to the last three layers of these networks. For CaffeNet, this allowed tuning of the last three fully-connected layers before the loss output. For GoogLeNet, this allowed tuning of the two fully-connected layers and convolutional layer immediately before each of the loss outputs. In our results, this is represented by the "partial-train" figures.

Fully-train all layers After pre-initializing each of the networks with ImageNet-trained weights, we allowed the entire network to be trained at a normal rate. In our results, this is represented by the "full-train" figures.

Fully-train all layers from scratch For this experiment, rather than pre-initializing the network with ImageNet-trained weights, we initialized the network with small random weights. Then, we allowed the entire network to be trained at a normal rate with very negligible priors.

For an additional point of comparison, we also performed last-layer fine-tuning and full-training from scratch using a 16-layer VGGNet model.

6. Results

	Top 1 accur.	Top 5 accur.	Final loss
Baseline SVM	0.031		55.48
Baseline ConvNet	0.062		42.73
CaffeNet fine-tuned	0.447		2.62
CaffeNet partial-train	0.418		2.61
CaffeNet full-train	0.417		3.21
CaffeNet scratch	0.005		5.31
GoogLeNet fine-tuned	0.774	0.943	1.25
GoogLeNet partial-train	0.775	0.943	1.28
GoogLeNet full-train	**0.800**	**0.951**	1.09
GoogLeNet scratch	0.347	0.636	6.42
VGGNet fine-tuned	0.789	0.942	**1.01**
VGGNet scratch	0.008	0.031	5.51

Table 1: Top 1 accuracy, Top 5 accuracy, and loss results of all experiments. The best performers in each category are bolded.

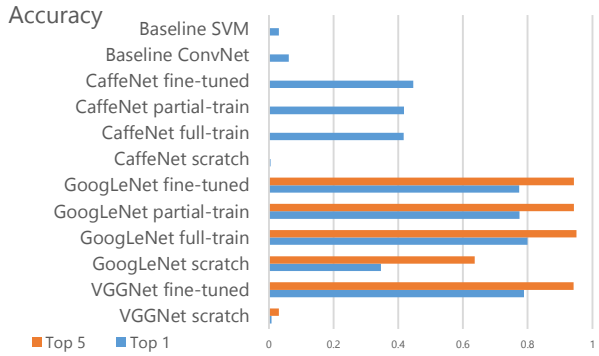

Figure 2: Accuracy comparison of all experiments

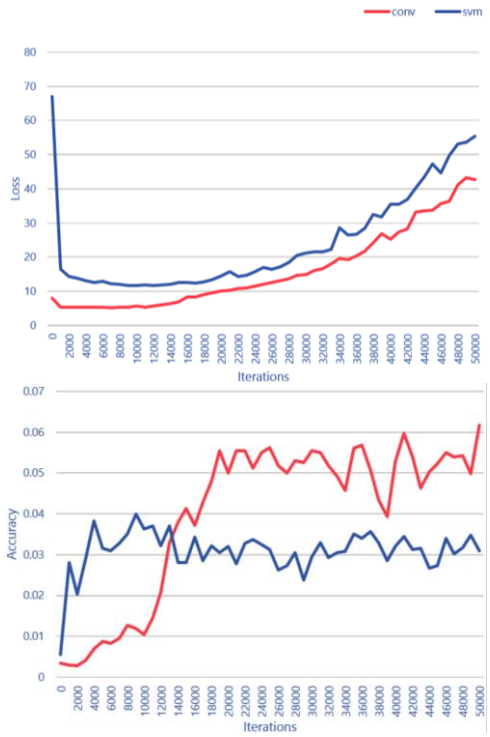

Figure 3: Baseline loss and accuracy

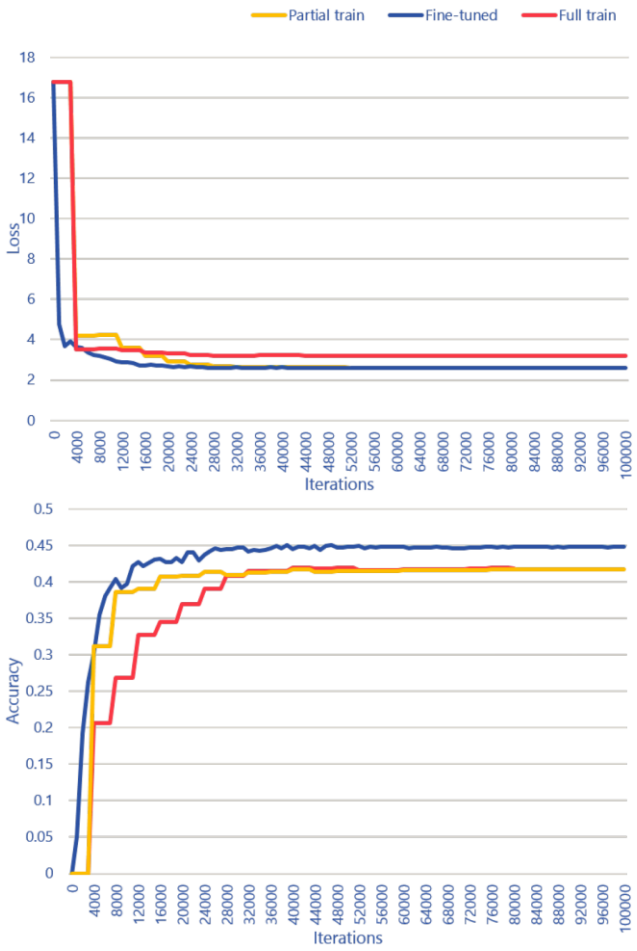

Figure 4: CaffeNet loss and accuracy

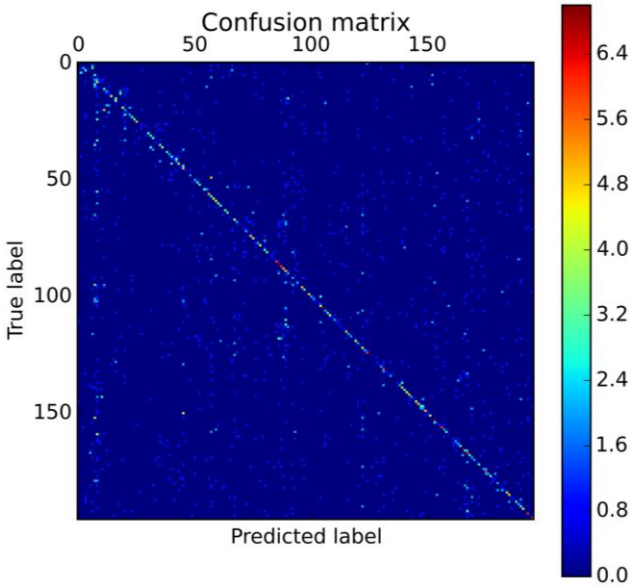

Figure 5: CaffeNet confusion matrix

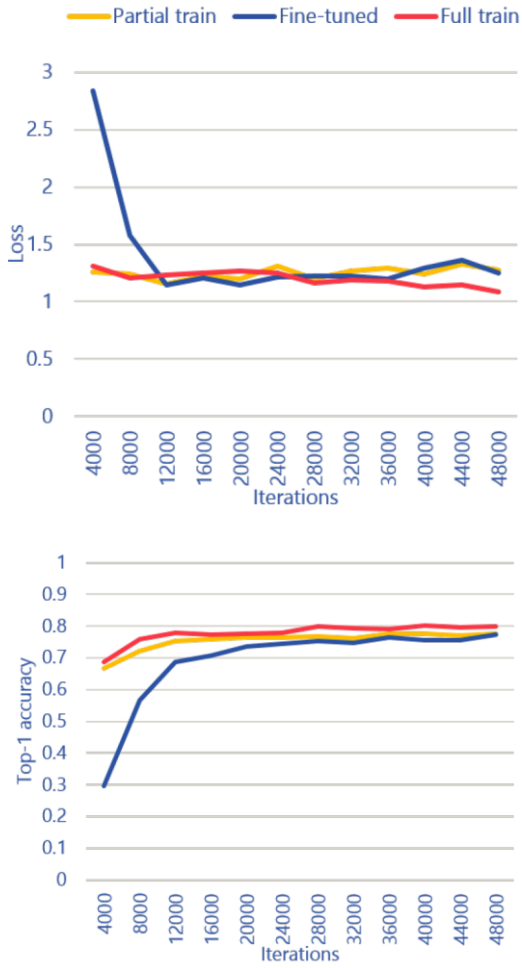

Figure 6: GoogLeNet loss and accuracy

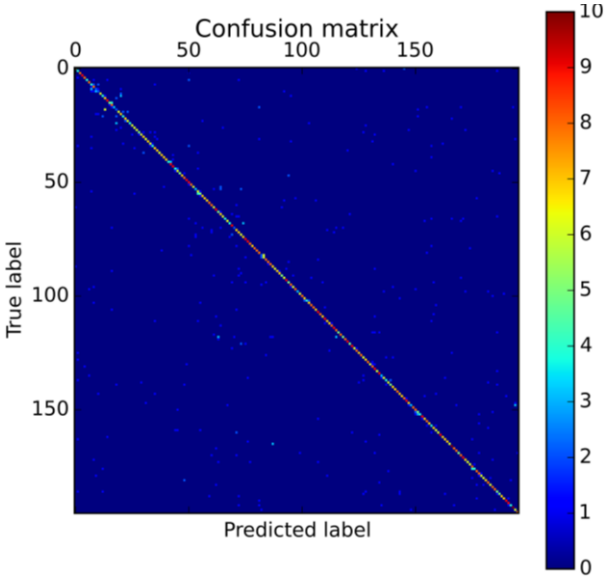

Figure 7: GoogLeNet confusion matrix

7. Discussion

As expected, due to the large number of classes and small amount of data, transfer learning was almost necessary to achieve decent performance, outperforming nets trained from small, randomly initialized weights.

Baselines Neither baseline model did particularly well. This was more or less expected; the simplistic models meant that the more sophisticated features present in the deeper models never had a chance to be learned.

Furthermore, without any sort of transfer learning, both massively overfit the data, as shown by the low plateauing accuracy, as well as the steadily increasing validation losses, when training loss always converged to 0 (omitted from graphs for brevity).

Transfer Learning When trained from scratch, most models tended to overfit, as shown by the plateauing validation accuracy, On the other hand, transfer learning in each setup was able to converge to a reasonable performance most of the time. Interestingly, both models tended to generalize better when fewer layers were fine-tuned. This is probably due to the quality of the ImageNet features in the earlier layers, whereas tuning these on the cars dataset tended to cause overfitting.

7.1. Error Analysis

As shown by the confusion matrices in the previous section, the errors produced by CaffeNet trended towards misclassifying many classes as a single class (visualized by the vertical "streaks"), whereas GoogLeNet's errors appeared to be more random.

In both of these models, errors appear to be more frequent closer to the diagonal than otherwise. Though the classes were reasonably diverse, there were still a few classes that were virtually identical to each other (such as different years of the same model of car), and would've proven difficult even for a human to distinguish. Since the classes are sorted by make and then model of vehicle, this shows that CaffeNet and GoogLeNet's misclassifications are more frequently apparent when distinguishing cars from the same make. Examples include different models of Audi and BMW sedans, which tend to look very similar to each other. When presented with cars that have very distinctive appearances, however, the models tended to do very well. For example, when presented with a picture of a Bugatti Veyron (a well-known "hypercar" with unique looks), all of the CNN models predicted the class correctly and were very confident of their predictions.

There are also a few classes of different makes that tended to be confused for each other. These tended to belong to small clusters of makes that are rather distinctive compared to most other cars, but relatively similar to each other, such as hatchbacks from Nissan being confused for those of Toyota.

Other than visual similarities, another source of errors was the data split distribution. Due to lack of time, we were unable to do much k-folding on the data. Furthermore, with an average of 30 images per class in the training split, there were a few data-starved classes at training time, resulting in somewhat skewed weights for that class. These can be observed as horizontal streaks within the confusion matrix, which represent classes that were rarely predicted correctly.

8. Future work

Though we were able to achieve significant progress in this particular task, there is still much to explore.

Additional models Within the scope of this project, we only tested 3 distinct modern CNN models (CaffeNet, GoogLeNet, and VGG). Doubtless there are more that we could have performed transfer learning on. In addition, there was a noticeable lack of diversity within the models; all three were designed for and initialized with ImageNet weights.

A possible future direction is using pre-trained nets from other tasks, including other fine-grained datasets. Another more ambitious direction would have been to design, implement, and test our own CNN model, though this would have been somewhat difficult to pull off (or at least optimize) with our given timeframe.

Additional transfer learning experiments We have explored several options in transfer learning, but there are many combination of features/final classification weight vectors that we could have performed fine tuning on.

Baselines Due to time constraints and the focus of the project being transfer learning, not much effort was put into baselines beyond the minimum of having comparable results. One possible direction would be to implement better baselines, though this is probably not as fruitful.

Image Preprocessing For efficiency, we chose a single data preprocessing scheme (bounding box cropping, then scaling) early on in our investigation. We had also played around with other ideas, such as scaling and square cropping, or only scaling. These ideas were never fully fleshed out in this project.

Additionally, another direction that had been discussed was the effect of the data images' resolutions on the performance. However, this idea was quickly scrapped when we realized that we didn't have a good way of transferring weights from pre-trained models of different resolutions. A potentially interesting investigation could focus on designing such a technique.

Dataset Quality Initially, our investigation was simply geared towards fine-grained classification, with the data constraint added later after we had already chosen our dataset. Though this gave us some insight into the nature of training on sparse data, the project perhaps could have been improved with more abundant data.

Web demo

For demonstration purposes, we adapted the Caffe web demo for use with our project. This application accepts image input through a URL or uploaded file, and then classifies it using CaffeNet and GoogLeNet. The top-5 results and confidences are then presented to the user. The demo is hosted here:

http://ld-gargantua.stanford.edu:5000/

Acknowledgements

We would like to thank **the CS231N course staff** for their advice and guidance during this project and **Jonathan Krause** for his expertise, data, and patience. We would also like to thank **the CS210 teaching team and Jay Borenstein** for providing dedicated CUDA hardware for the bulk of our computation, **Microsoft Azure** for providing access and complementary credit for their G-series compute-intensive virtual instances, and **the BVLC team and other contributors** for building the Caffe deep learning framework.

References

[1] L. Dlagnekov and S. Belongie: Recognizing cars. UCSD, La Jolla, CA, TR. CS2005-0833, 2005. 1, 2

[2] G. Pearce and N. Pears: Automatic make and model recognition from frontal images of cars. In AVSS, September 2011. 1, 2

[3] V. Petrovic and T. Cootes: Analysis of features for rigid structure vehicle type recognition. In BMVC, 2004.

[4] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2014.

[5] Farrell, R., Oza, O., Zhang, and N., Morariu, V.I., Darrell, T., Davis, L.S.: Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In: ICCV 2011

[6] Angelova, A. and Zhu, S.: Efficient object detection and segmentation for fine-grained recognition. In: CVPR 2013

[7] Szegedy, et al.: Going deeper with convolutions. In: CVPR 2014

[8] Simonyan, K. and Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: ICLR 2015

[9] Krause, J. et al: 3D Object Representations for Fine-Grained Categorization. In: 4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013

[10] Pan, S. and Qiang, Y.: A Survey on Transfer Learning. In: IEEE Transactions on Knowledge and Data Engineering 2010

[11] Krizhevsky, A., Sutskever, I. and Hinton, G.: ImageNet Classification with Deep Convolutional Networks. In: NIPS 2012

[12] Donahue, J. et al.: A Deep Convolutional Activation Feature for Generic Visual Recognition. 2014

Appendix

	Iterations	LR	LR decay
Baseline SVM	50,000	0.001	0.9
Baseline ConvNet	50,000	0.001	0.9
CaffeNet Finetune	100,000	0.001	0.1
CaffeNet PartTune	80,000	0.001	0.1
CaffeNet FullTrain	100,000	0.001	0.1
CaffeNet Scratch	100,000	0.01	0.9
GoogLeNet Finetune	50,000	0.0001	0.96
GoogLeNet PartTune	50,000	0.0001	0.96
GoogLeNet FullTrain	50,000	0.0001	0.96
GoogLeNet Scratch	200,000	0.0001	0.96
VGGNet Finetune	100,000	0.0005	0.1
VGGNet Scratch	250,000	0.01	0.1

Table 2: Hyperparameters for all experiments

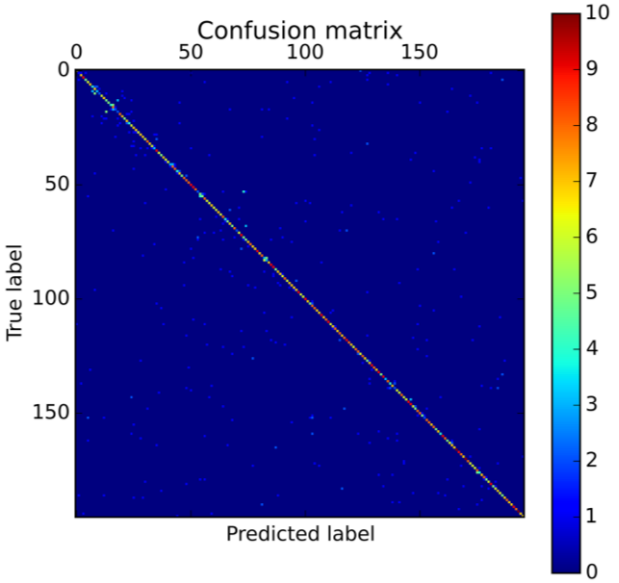

Figure 8: VGGNet confusion matrix