Exploring transfer learning for self-driving car dataset

Paula Kintschev Santana de Moraes
Computer Science Departament – Institute of Mathematics and Statistics
University of São Paulo
São Paulo – SP – Brazil
paulaksm@ime.usp.br

1 Introduction

A core subject in the self-driving cars domain is how to use images taken in real-time to best steer a vehicle in a road. Mapping visual inputs to steering commands can be seen from a machine learning perspective as a regression or classification problem, depending mainly if the control outputs will be discrete e.g. 'turn left', 'go forward' or continuous e.g. steering angle: $+0.014°$. In this experiment, the road following task will be treated as a classification problem, so given an input image the learning model will predict a control command ('up', 'left', 'right').

In this context, how to encode an image as a feature vector is a crucial step. Given an image with dimensions $45 \times 80 \times 3$ one simple way to "extract" features is by flattening it to a single dimension, in this case each pixel is a feature of a $10,800$-dimensional input vector. This image vector can be an input to a Convolutional Neural Network (CNN) or Deep Feed Forward Network (DFF), where the weights for each pixel will be learned in respect to the classification output. To obtain a good performance with an end-to-end learning approach, it is mandatory to have a large dataset of this learning domain. Unfortunately, domain-related datasets are not always available in a good amount or not easily collected. Learning from small datasets can be achieve with transfer learning.

Transfer learning is a learning framework that allows knowledge transfer in a way that the training domain and feature space for a given task in a model may be different from the testing tasks for this same model. One approach to transfer learning is feature-representation-transfer that aims to find a feature representation that minimizes the difference between source domain, in which the model was trained, and target domain, a different domain of interest. In a supervised feature construction the goal is to learn a low-dimensional representation that is shared across related tasks (1). In the context of transfer learning, pre-trained deep learning models, ConvNets, can be used for feature extraction. Since its architecture consists mainly in a stack of convolutional layers followed by fully-connected layers, it can be used as a feature extractor as long as the final layer is removed.

This work aims to (i) explore pre-trained CNNs using a self-driving car dataset, (ii) create a public available repository to generate image embeddings, and (iii) investigate the impact of feature extraction with transfer learning by comparing accuracy of models trained with this pre-processed input vector and with models trained in an end-to-end learning approach.

2 Dataset and Manipulation

The self-driving car dataset (2) used in this experiment is the result of driving a small remote controlled robot car (Figure 1) for 4 hours in an oval paper track (Figure 2), where for each car command 'up', 'left' or 'right' an image was taken and saved. Totalizing roughly 70,000 data points. The dimensions of the RGB images are 45x80. Each image in the dataset is associated with a label that corresponds to a command given to the robot. Figure 3 shows some of the data points.

Since Keras has deep pre-trained models trained on ImageNet, a public repository was created to provide different image embeddings for a quick evaluation of machine learning algorithms to be trained on different representations of the same input data (3).

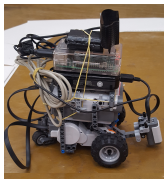

Figure 1: Remote controlled car

Figure 2: Dataset track

(a) Left

(b) Up

(c) Right

Figure 3: Examples of all classes in the dataset

3 Transfer learning vs end-to-end learning

Table 1 shows the results for different embedding techniques, using VGG-16, MobileNet and Xception architectures with their pre-trained weights. Our base line is a convolutional model trained in an end-to-end fashion i.e. without feature engineering. The architectures are represented as lists where 3 is the output layer with softmax activation, so $[200, 3]$ stands for a network with one hidden layer with 100 units. In the convolutional model, tuples represents a convolutional layer of 24 5x5 filters, for example.

Table 1: Results for training simple multilayer perceptrons on different image embbedings comparing to a model trained using an end-to-end approach (highlighted in red)

Embedding	Architecture	Input features	Training instances	Accuracy
VGG-16	[200, 3]	4096	2900	0.797
MobileNet	[200, 3]	1024	2900	0.784
Xception	[200, 3]	2048	2900	0.785
flattening	[(24, 5), (36, 5), (64, 5), 200, 3]	10800	56000	0.80

Analysing Table 1, is clear for this task that when using transfer learning is possible to get similar results to a classical supervised learning training approach using much less data. Also is important noticing the reduced size of the input vector in contrast to working with flat image representations, reducing the total computational cost.

4 Conclusion

Transfer learning has proven to yield good results on feature extraction in a unique domain, presenting similar results of a deep classical training approach using **only** 5% of the original training dataset. It's also worth noticing that since we collected the data, the images passed through the pre-trained networks were never seem before on their trainig phase, showcasing that indeed a transfer of feature knowledge was achieved. Future work resides on using layer visualization techniques to better understand the feature extraction process.

References

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] P. Moraes and F. Salvatore, "Self driving data." `https://github.com/felipessalvatore/self_driving_data/`, 2018.

[3] P. Moraes, "Transfer learning - image embeddings generator." `https://github.com/paulaksm/transfer-learning`, 2018. commit xxxxxxx.