Computer Vision Lab Working Notes
Car Model identification with bi-linear models

Roberto Labadie Tamayo

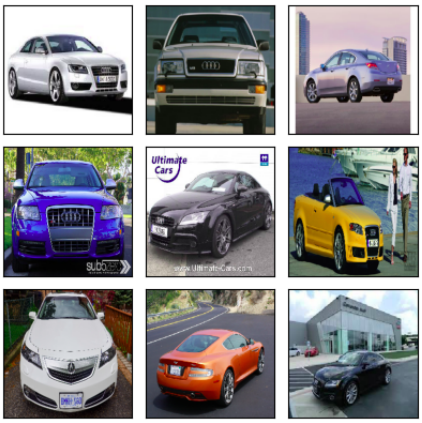
April 2023

Given a dataset $\mathcal{D} = \{X, Y\}$ with X_i an image shaped as NxMxH, belonging exclusively to class $Y_i \in C$, our task is to propose a model $\mathcal{F}$ that minimizes the expected error defined as:

$$e = 1 - \frac{\sum_{x_i, y_i \in T} \delta(y_i = \mathcal{F}(x_i))}{|\mathcal{T}|}$$

where T is an evaluation dataset with the same nature as D.

As a particular instance of this image classification task, we deal with a dataset, where each image maps into 250x250 RGB pixels a car belonging to some brand.



From here, we face a fine-grained multi-class classification problem of brand identification. This work relies on the use of Deep Convolutional Neural Networks, specifically Bi-Linear modelation (Lin et al., 2015), on top of the Pytorch framework.

The labels in the training and test sets are distributed as illustrated in Figure 1, where, as we can see there exist a pretty fair balance among the classes examples, nevertheless besides the error, in these working notes we report the macro-f1 measurement for each result.

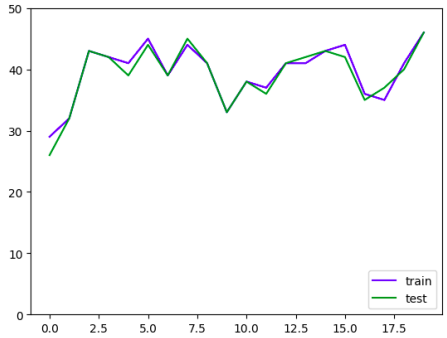


Figure 1: Data Distribution.

Based on previous work, we normalize the inputs of the model assuming a standard deviation of 255 for each channel values. Also, data augmentation is employed by horizontally flipping the images in a stochastic way and substituting the image with randomly cropped patches padded by reflection to obtain 250x250 pixels images.
The studied approaches are based on using a single neural network as an encoder for simplicity. This encoder receives the same image twice to obtain two dense tensors of features and them apply the bi-linear pooling operation. To this end, and to simulate encoders which learn different feature representations, for each of the two mappings obtained with our encoder, the 50% of the features are randomly dropped before bi-linear pooling.
Two strategies are tested to avoid loosing the base knowledge of the pre-training phase. The first is based on freezing the pre-trained encoder at the beginning and in some point unfreezing it, and the second consist just on employing the strategy proposed in the Universal Language Model Fine-Tuning (ULMFiT) (Howard and Ruder, 2018) for tuning pre-trained models in a gradual unfreezing-discriminative fashion.
The latter involves setting a different learning rate for each encoder layer, increasing it while the neural network gets deeper. This is:

$$\alpha_i = \alpha_0 + i \times \frac{(\alpha_d - \alpha_0)}{d}$$

where α_i is the learning rate for the i^{th} layer,d is the depth of the encoder module,and α_0 and α_d are preset lower and upper bounds of the learning rate respectively, in this case $1e-5$ and $1e-4$.
This dynamic learning rate keeps the most information from the pre-training at shallow layers and biases the deeper ones to learn about the faced downstream task.

Regarding the hyperparameters of the models, Adaptive Momentum Estimation (Adam) based optimizer (Kingma and Ba, 2014) is employed on the optimization process with a batch size of 64 examples.
As the encoder module, we employed a pre-trained model of DenseNet121 (Huang et al., 2016). For all of the approaches on top of this encoder, we simply stack a classification layer of 20 neurons in charge of predicting the probability of an image belonging to some car brand.

Using DenseNet121 as encoder and unfreezing it at the 60^{th} epoch of 150, with a plane learning rate of $1e-4$ we obtain an error of 0.314 which is equivalent to 68.6% of accuracy and 0.689 points of macro-f1. Taking advantage of the soft learning rate employed, we also try to reverse the freezing process, i.e., taking into account the soft lr, the amount of bad gradient coming from the stacked classification layer is not too aggressive so we first fine-tune the whole model and at the 60^{th} epoch we freeze the encoder module to specialize the classification layer.
The latter slightly under-performed the results yield by the above approach with an error of 0.323 and macro-f1 of 0.681.

Employing a dynamic learning rate to avoid freezing and unfreezing the neural network yield much more stable results and robustness in less epochs of training, with **0.815** points of macro-f1 and **0.185** of error in just 34 epochs.
We also tried to combine this approach with freezing the encoder module at the middle of the training process, to this end, we also add one cycle learning rate policy proposed on (Smith and Topin, 2017) which was in charge of scheduling the learning rate of the classification layer once the encoder module was freeze. This approach, although had a better performance compared with the experiments where non-dynamic learning rate was used, yield a slightly poorer performance with 0.807 points of macro-f1 and 0.193 points of error, again, just using 50 epochs of training. The learning rate evolution of the classification layer is depicted in the following Figure.

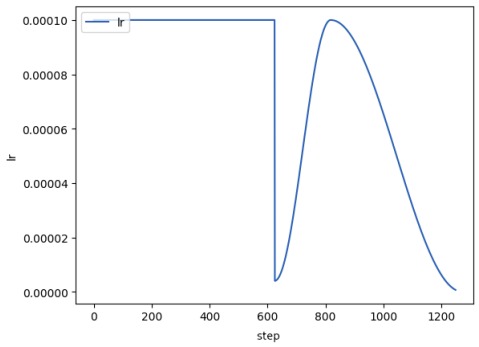


Figure 2: Learing rate evolution of the classification layer

The code for reproducing these models is provided with these working notes.

Bibliography

Jeremy Howard and Sebastian Ruder. 2018. http://arxiv.org/abs/1801.06146 Universal language model fine-tuning for text classification.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016. http://arxiv.org/abs/1608.06993 Densely connected convolutional networks. *CoRR*, abs/1608.06993.

Diederik P. Kingma and Jimmy Ba. 2014. https://doi.org/10.48550/ARXIV.1412.6980 Adam: A method for stochastic optimization.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. 2015. https://doi.org/10.1109/ICCV.2015.170 Bilinear cnn models for fine-grained visual recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1449–1457.

Leslie N. Smith and Nicholay Topin. 2017. http://arxiv.org/abs/1708.07120 Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120.