Woof woof ruff au au arf, ruff woof*

Giuliano R Pinheiro
(RA 108579)

1. Introduction

Of all fields that benefit from machine learning applications, it feels like none can be said more benefited than computer vision. In the last few years, there has been an incresing interest over the field as computing power increased to a point industry giants started to offer these for free. Don't get me wrong, they virtually always have, but there is a difference between having enough hardware to keep a web server up and hardware capable of intense computing.

Since AlexNet came out in 2012[1] the attention has turned to the power of Deep Neural Networks (DNNs). Mr. Krizhevsky's at al. network caused such a fuss computer vision is now dominated by the offspring of their creation, and most applications still go towards computer vision. Applications range across a wide range of subjects, like face detection and identification, movement tracking, self-driving cars, you name it. Three years ago a competition posted in data science website Kaggle asked how good could be a model to identify individual plankton species.

This assignment asks a similar question: how good can a model identify individual breeds of dogs? For this task, given a specific dataset, I employ Deep Learning models and explore how they evolve and accomplish the task using some configurations of classification ends, optimizers and other hyperparameters.

This work was implemented in Python using libraries `numpy`, `matplotlib`, `tensorflow`, `keras`, `pandas` and `scikit-learn`.

2. State of the Art

Under the scope of this report's proposal, there are some state of the art network architectures worth mentioning: VGG, ResNet and Inception.

The VGG architecture[1] is composed of either 16 or 19 convolutional layers, which gives them the names VGG16 and VGG19, and are among the most preferred networks to get image descriptors from. It took 2-3 weeks of GPU power to train it, and there is a caveat: it is no trivial task to let its 138 million parameters train.

Residual Neural Network, or ResNet[2], features "skip connections"[2] and makes heavy use of batch normalisation. Skip connection in this architecture refers to a mechanism capable of carrying the input over between small groups of layers as the identity function. It is theorised this helps avoid the vanishing gradient problem DNNs suffer from.

Last but not necessarily least, there is the base architecture I work with, Inception v3[3]. This architecture employs a block it calls *the inception module* and, moreover, there is actually an *Inception Hypothesis* it bases itself on.

Convolution layers are supposed to learn not only spatial correlations in the data, but also cross-channel correlations. All at once. The Inception Hypothesis states that these correlations are sufficiently decoupled[3] one could learn (or sufficiently approximate) them separately by using spatial and cross-channel convolutions and combined them, instead of trying to do it all at once, jointly.

3. Dataset and preprocessing pipeline

The dataset is comprised of 83 classes with 100 examples per class. All images are RGB but they do not agree on any dimension other than that. Their content is not limited to the dog in question, several of them being in fact shots including other objects like humans or other dogs in the background (sometimes this is arguable, other dogs appear quite in foreground). Also some examples' breeds can even be contested (see Figure 1), but they are a minority and I do not think the model could be mistrained by those.

For validaton, the set was comprised of the same classes but it was slightly unbalanced, having at least 50 images per class but sometimes twice or thrice that amount. Other than that, both sets of images shared the same qualities.

As the deep learning models used in this assignment were all trained on ImageNet, some preprocessing was required. Images went by an augmentation pipeline which

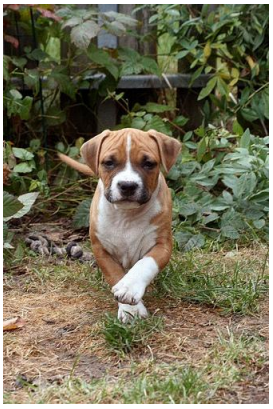
*Dog Breed Classification: CNNs to the rescue!

[1] AlexNet wasn't the first convolutional network, that goes to LeNet in 1998, but it was good enough on ImageNet it got the spotlight.

[2] ResNet was not the first to apply such an invention. The Highway Network introduced "gated connections" which act very alike to ResNet's skip connections. It was noted that the Highway Net solution space *contains* ResNet but it performs no better than it. This contradicts the expectation but hey, given we're speaking of DNNs, what doesn't?

[3] You might see a parallel with sepparable kernels in traditional convolutions, and you would be on the right track.

(a) Arguably not a boxer



(b) Definetly not a pitbull

Figure 1: Examples of probably misplaced pictures in the dataset (these are from the validation set): a is under Boxer and b is under Pitbull

applied the following transformations using uniform probabilities over each of their values:

- Brightness scaling to interval $[0, 1]$
- Rotation between $[-10, 10]$ degrees
- Horizontal shift of up to 10% of the pixels
- Vertical shift of up to 10% of the pixels
- Horizontal flip
- Random zoom factor in range $[0.8, 1.2]$

Note preprocessed images are not saved, all image transformations are done on the fly by keras and these images are discarded after each batch is fed to the network. It is fairly common to also standardise images by ImageNet's mean and standard deviation vectors, but this was not executed during this work.

4. A Baseline Model via Transfer Learning

To obtain a baseline for comparison, I tuned an Inception v3 model pretrained on Imagenet by removing the top layers (i.e. the bottleneck feature layer), freezing **all** remaining layers and adding a new set of layers for classification for the 84 classes of the dataset. These layers were:

Flattening layer to linearise the output of the network

Dropout layer with $p = 0.25$ probability of dropout

A Fully-Connected layer

A Softmax layer with as many outputs as classes, i.e. 83

The model had 10,879,059 trainable parameters. It was trained over 13 epochs using 32-image batches, SGD as the optimization method with learning rate 10^{-4}, momentum 0.9 and decay 10^{-6}. It attained 0.28 loss and 0.91 accuracy during validation. Training was stopped as soon as the training error started to get smaller than the validation error. In this case, at the same time the accuracy followed the same rule.

5. Proposed Solution and Experimental Results

5.1. Deep Choices

The proposed solution employs the Inception v3 architecture with pretrained weights on ImageNet, with its classification block replaced by an equivalent configuration, but limited to our number of classes:

Global Average Pooling 2D layer, a srategy proposed in [4] to replace the fully connected layer in DNNs. But, as for the creativity of the research community, tt is also commonly used as a means to radically reduce dimensionality

A Fully-Connected layer

A Softmax layer with as many outputs as classes, i.e. 83

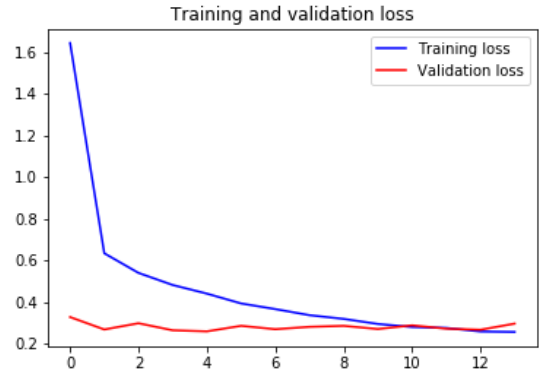
The training was split between two machines: one local with a GeForce GTX1070 with 8 GB of memory and another one as an instance on Google Colaboratory, allegedly on a Tesla K80. Google Colab is a shared environment so I assume their resources are also shared between its users[4]. Several base architectures with frozen layers or not and optimization methods and its parameters were tried, some attaining similar classification performance and error. Thus, I went for the simplest, with least parameters. That corresponds to the model listed above.

Final optimizer was Adam, with learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, no decay. Training ran for 14 epochs (but the model at use is the one at the 10th epoch) using 32 image batches. Figure 2 shows how training went. After 10 epochs the classifier was ready, its categorical crossentropy about 0.30 for both training and validation. At that moment, accuracy was around 0.91 for both training and validation.

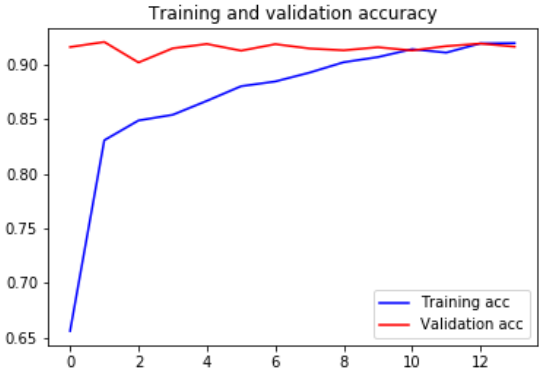
5.2. A Second Thought on the Classification End

Several changes were tried for fine tuning Inception. Those listed had good results given the right choice of opimizer hyper parameters but not always for all fine tuning degrees (read: layers) of freedom. A "common" double

[4]Given that I got similar performance with both systems, that is a very reasonable assumption.



(a) Loss during training for inception network



(b) Categorical accuracy during training for inception network

Figure 2: Training and validation for Inception v3 model

fully connected setup with softmax activation was the first try. Then, the Global Average Pooling described above was used. Each of these was tried on three distinct layer configurations: no trainable layers, the last inception module made trainable and the last 2 inception modules made trainable[5].

All these configurations optimised by RMSprop with parameters as those used to train Inception net, by SGD optimizer with learning rate 10^{-4}, momentum 0.9 and decay 10^{-6}, and by Adam, as previously described. Most of them did not take much more than 15 epochs to show signs of convergence. All models that were not converging or that were converging too slowly were discarded, retaining only those which did. Those performed very similarly, causing the final model to be chosen by Occam's Razor instead of by clever statistics.

5.3. A Third Though on the Classification End

For the next set of experiments, I decided to let go of the softmax classification end and use bottleneck features directly from the inception network. These are third rank, $(8, 8, 2048)$ dimensional tensors which were reduced by Global Average Pooling to 2048-dimensional vectors which were fed to a pair of SVMs, one linear and one carrying an RBF kernel, which were crossvalidated using a stratified 5-fold scheme and decided the multiclass problem via one-vs-rest. Table 1 shows how this strategy fared. Highest scores are marked with a star.

[5]Keep in mind these are groups of layers.

Model	Accuracy	F1 measure
Linear SVM	0.935 ± 0.005	0.935 ± 0.005
RBF SVM	$0.943 \pm 0.006^{\star}$	$0.942 \pm 0.006^{\star}$

Table 1: Bottleneck features fed directly to SVM classifiers

Model	Accuracy	F1 measure
Linear SVM + PCA	0.926 ± 0.005	0.926 ± 0.005
Linear SVM + LDA	0.973 ± 0.005	$0.973 \pm 0.005^{\star}$
RBF SVM + PCA	0.945 ± 0.004	0.944 ± 0.004
RBF SVM + LDA	$0.971 \pm 0.003^{\star}$	0.971 ± 0.003

Table 2: Bottleneck features fed to SVM classifiers after going through dimensionality reduction. PCA features were reduced to 600 dimensions and LDA features to 68 dimensions. Both retain 95% or more of the total energy.

As the table shows, these are no better than the network's initial result, but were already good. So I decided to give this scheme some more attention and turned to the features themselves. As the previous experiment showed, the features performed not much better than the network, and one of many possible hypothesis is that the classifier is not bad for them, but maybe the space is too big. So I applied PCA and LDA reductions to that space, aiming for at least 95% energy for the reduced feature space. That translated into 600 dimensions for PCA and only 68 for LDA. Crossvalidation by the same 5-fold scheme yielded the results on Table 2. Again, highest scores are marked with a star.

This result points both linear and RBF SVMs using LDA features perform best and my experiments can't statistically differentiate them. But these pretty numbers had standard deviations too low to be trusted. There is an observation that, although it doesn't invalidate the procedure during which the classifiers were trained, turns that table upside down. All classifiers were trained and validated from the same big set of feature vectors (originated from the training set). So far my hypothesis is these are legitimate features and the training, valid. But as Table 3 shows, these classifiers does not survive a different dataset.

The validation set for the neural network was used to ex-

Model	Accuracy	F1 measure
RBF SVM	0.86 ± 0.07	0.89 ± 0.11
RBF SVM + PCA	$0.95 \pm 0.04^{\star}$	0.94 ± 0.03
RBF SVM + LDA	0.94 ± 0.03	0.94 ± 0.03

Table 3: Bottleneck features fed to SVM classifiers after going through dimensionality reduction. This time, though, they were validated in a totally different set of features. Linear SVMs had a similar result.

tract the features that were used to validate this last round of classifiers. They show poorer performance than their predecessors probably because the validation set somehow yields different features from the training set. Both the testing set during cross validation and the bigger validation sets were not seen by the classifiers during training but they obviously differ in some way.

Now it stands to reason the dimensionality reduction caused by LDA might have been too extreme, and PCA gains an edge over LDA for the chosen dimensionality. On the final set of images, the test set, composed of 5,420 images, the appointed best classifier, RBF SVM with PCA features from Inception bottleneck, attained **0.95 accuracy and 0.94 F1 score**, somewhat on par with the training results.

6. Conclusions and Future Work

During this work, several ideas crossed my mind. The first two were relative to the overall ensembling that could be mounted as a means of not only classifying the breeds correctly, but also to drop anything that might be misclassified from the input image that was not a dog. One can defend that if the recognition system is good enough, one can append to its pipeline a detector to sepparate such undesirable content from the input. And that was precisely the idea.

The ensemble would start with an open set recognition system[6] capable of recognising dogs. Simple, maybe naïve approximations could use the output of the classification network to threshold possible rejections, but there are more elaborate efforts on that direction, notoriously by the use of Extreme Value Theory as is the work of Scheirer et al. [5] and directly to neural networks by Bendale and Boult[6].

Another technique that could be explored is Test Time Augmentation (TTA). With a reasonably high recognition accuracy, I just left it out mainly to use processing time trying out different parameterisations instead of trying TTA since it might help that extra bit but not much if the classifier is not good enough[7].

Speaking of augmentation, another improvement could come from preprocessing. As discussed in Section 3, images from this dataset were not of the same dimensions, but the networks used in this work all have an expected receptive field which is a square image. Therefore images going through the augmentation pipeline were resized to that square input size, effectively distorting its contents. An improvement was made, cropping the maximum square possible from the image, sliding it at random in the larger dimension, but it was of little improvement, and sometimes made the result get *worse*. A different idea would be considering better strategies like adding intelligence when chosing the crop position relative to the image so the subject is not partialy (or totally) absent from the resulting crop.

Finally, it was partially tried to release training of some layers to fine tune the best performing model, Inception v3, but they were not as successful as training only the classification layers. It might be the case that fine tuning those layers requires extra control over optimisation parameters like learning rate. These architectures were originally trained using learning rate scheduling and precise values, and those were not tried in this work[7].

The SVM models trained in the bottleneck features proved capable of good scoring on the same task when paired with dimensionality reduction. A not direct conclusion from this is that the network output space might be just too big for this task. Inception was conceived for really big tasks like ImageNet, with not only more than 10 million images for training, but also 1,000 classes. By these numbers alone one can wonder how the number of classes and the output dimension in the bottleneck relate to each other.

Dimensionality reduction was then the cherry atop the cake, pushing the result just enough to make it noticeable for both accuracy and F1 measure, indicating good precision and recall along with the fact datasets were always kept balanced. Still, it remains to be seen if bottleneck features are good as they are before global pooling. That would easily give an 8 classifier ensemble to test with or without dimensionality reduction, but this work will not explore this possibility.

Another alternative for the future is to use more of ensembling results, taking advantage of probability estimates output by the classifiers to produce another score that can range between simple arithmetic mean to more elaborate results like inverse variance weighting. These would have been explored given more time, but for now they will have to wait for the next take on this problem.

[6] An open set problem is that one has when the trained model can be exposed to more classes during prediction than it encountered during training.

[7] To be fair, I tried tuning some parameters and using learning rate scheduling but only up to a point. Given that extra time was a requirement when trying some of these, I could only go so far and results were not as good as the all-layers-frozen models.

Finally, an approach also left unexplored mentions another kind of ensembling: cascading. One could train a cascading classifier trained on the errors of the main classifier (inception net, in this case). Both predictions would be aggregated by using as a weighting parameter the confidence the main classifier has on its prediction. That number could be decided both experimentaly or by regression on a model like the ones mentioned in the last paragraph.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1

[3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. 1

[4] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. 2

[5] Ethan M. Rudd, Lalit P. Jain, Walter J. Scheirer, and Terrance E. Boult. The extreme value machine. *CoRR*, abs/1506.06112, 2015. 4

[6] Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. *CoRR*, abs/1511.06233, 2015. 4

[7] Classifying plankton with deep neural networks. http://benanne.github.io/2015/03/17/plankton.html. Accessed 01/Jun/2018. 4