Dog Breed Classification

Wallace S. Loos

September 11, 2018

1 Introduction

The dog breed classification problem has been hosted by Kaggle. It consists in finding the correct breed for a given dog. The low intra-variance presented among some dog breeds and the class imbalance, make this classification problem a challenge. Besides that, the dataset provided by Kaggle, has only 10222 images, for 120 different breeds. In this report we present three different architectures to solve this multiclass classication problem and also an approach to handle the class imbalance.

2 Multiclass Classification

The goal of the classification problem is to take an input $\mathbf{x}$ and assign it to one of k discrete classes [1]. When we have only two classes, it is called a binary classification, and when we have more than two classes, it is called multiclass classification. For each input $\mathbf{x}$, only one class is assigned to it, the classes are mutually exclusive, differently from a multilabel classification problem. A common issue related with multiclass classification is the class imbalance. The imbalance dataset can impact the performance either in classical machine learning approaches or modern machine learning approaches [2]. In the next section we propose a way to address this problem.

2.1 Class Imbalance Problem

Our dataset has an uneven distributions. There are 120 samples in the majority class, and 66 samples in the minority class. We have a linear imbalance class distribution. There are two main approaches to handle class imbalance. The first one operate over the distribution of the classes (undersampling or oversampling). The second one operate over the training algorithm. In this report we adopted the first category. Every class was oversampled until they had the same number of samples of the majority class.

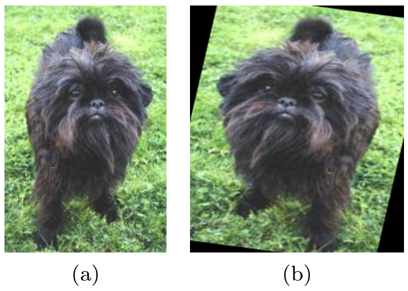

Figure 1: Oversampling the dataset: (a) before, and (b) after the transformations.

To oversample the dataset three transformations were performed: horizontal flip, rotation, and scale, as presented in Figure 1. In the end of this process, our dataset increased from 10222 to 15120 samples. The oversampled dataset is created separately from the original dataset, then we analyzed both original and oversampled dataset for comparison purposes.

3 Network Architecture

In this section we present three different architecture to solve the dog breed problem: 1) Transfer learning (inceptionV3) + MLP, 2) Bottleneck Feature (inceptionV3) + Logistic Regression, and 3) Bottleneck Feature (inceptionV3) + Support Vector Machine (SVM), as presented in Table 1.

	Architecture
1	Transfer Learning + MLP
2	Bottleneck Features + Logistic Regression
3	Bottleneck Features + SVM

Table 1: Three different architecture proposed.

3.1 Transfer Learning

Deep Convolutional Neural Network (CNN or ConvNet) has been used widely in computer vision tasks [5, 8]. For a good performance, usually deep CNN's need large training datasets. Since our dataset is small, we cannot train a deep CNN from scratch. However, we can pretrain our CNN on a very large dataset. This is called *transfer learning*. To overcome the limited dataset we pretrain our CNN on Imagenet dataset. Imagenet contains more than 2 million images with 1000 categories [3]. Our CNN use the inceptionv3 architecture [6], followed by a convolutional layer (3x3x64), dropout (rate = 0.5), bacth normalization and two fully connected layers (FC). The first one followed by a rectified linear unit (ReLU) and the output layer with a sofmax-function.

3.2 Bottleneck Features

The other architectures use the CNN as a feature extractor. We remove the last fully-connected layer, and then we use the features to train a classifier. We performed an average-pooling in the last convolutional layer of the inceptionv3 architecture. The classifiers used were: logistic regression and support vector machine (SVM). To visualize the features extracted by the CNN, we reduce the dimensionality to two dimensions using the algorithm t-SNE [7]. The result of this visualization is presented in Figure 2.

3.3 Training

The input image resolution was fixed in (300×300). The dataset was slipt in 80% for training and 20% for test. The validation set was made taking 20% from the traning set. The oversampled dataset was divided in the same way. Both datasets are used during the training. We train the first architecture for about 20 epochs on the training and validation datasets. Throughout training we use a batch size of 64 and optimise the loss function cross entropy,

$$cross\ entropy = \sum_{i=0}^{N} y_i log(\hat{y_i}) \qquad (1)$$

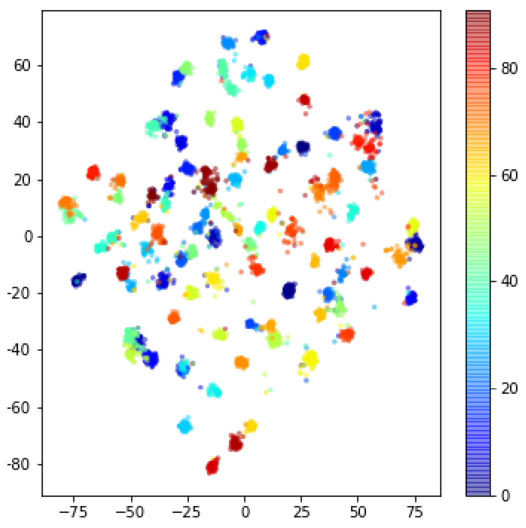

Figure 2: CNN features.

where y_i is the ground truth and the $\hat{y}_i$ is the prediction. The optimizer used was *Adam* [4]. Our final layer predicts the breed dog probabilities. The second architecture was trained using logistic regression with the Limited-memory BFGS optimization algorithm, and the third architecture was trained using SVM with radial basis function kernel (RBF).

4 Result

We present the results of the three architecture proposed evaluating them in the original dataset and in the dataset oversampled. The result are summarized in Table 2 and Table 3. The result obtained from the submission to the Kaggle is presented in Table 4. The cross-entropy was used to compute the error. For the original dataset the second model provided

Architecture	Accuracy	Error
1	0.835	0.564
2	0.892	0.351
3	0.895	0.860

Table 2: Results without oversampling the dataset.

a high accuracy and the lowest error. Despite the highest accuracy of the third model, its error was the largest among the architectures, suggesting a larger number of misclassification. The result obtained from

Architecture	Accuracy	Error
1	0.859	0.451
2	0.951	0.144
3	0.930	0.556

Table 3: Results oversampling the dataset.

the dataset oversampled were better compared with the result from the models trained using the original dataset. The second architecture had, yet, the best result. However when they were applied for the Kaggle test set submission, the results were almost similar achieved previously with the original dataset, as presented in Table 4.

	1	2	3
Dataset	0.584	0.351	0.857
Dataset+Over	0.586	0.354	0.814

Table 4: Results from the submission on Kaggle's test dataset.

5 Discussion

The oversampled dataset kept the performance of the model similar to the original dataset. It wasn't very effective, because the models overfitted when they were trained over the oversampled dataset. When the breeds have different characteristics, as presented in Figure 4, our feature extractor could take distinguish features from them, as presented in Figure 6a. However, the low intra-variance presented by some dogs breed resulted in a large number of misclassification. The Figure 5 presents a case where two different dog breeds are very similar. The features extracted from them are shown in Figure 6b.

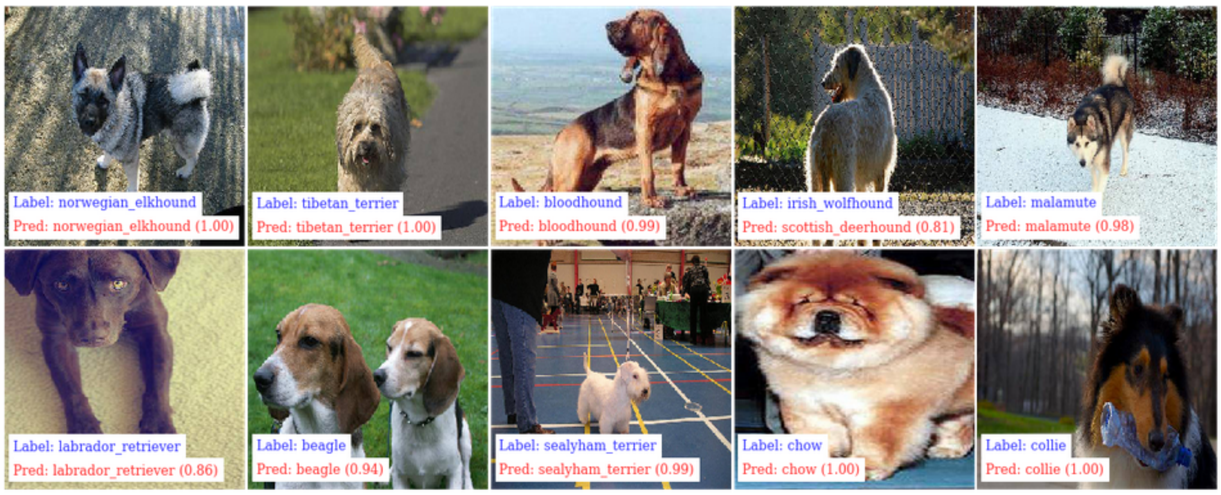

Figure 3: Results obtained from the second architecture.

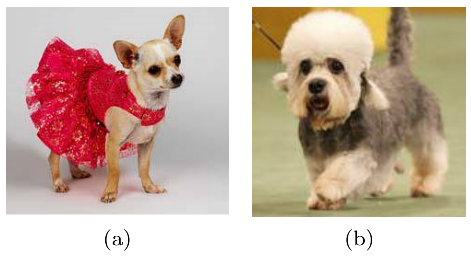

Figure 4: High variance of the dog breed: (a) chihuahua, and (b) dandie dinmont.

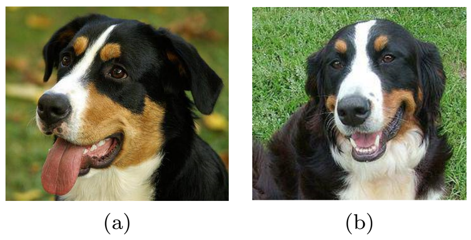

Figure 5: Low variance of the dog breed: (a) appenzeller, and (b) bernese mountain dog.

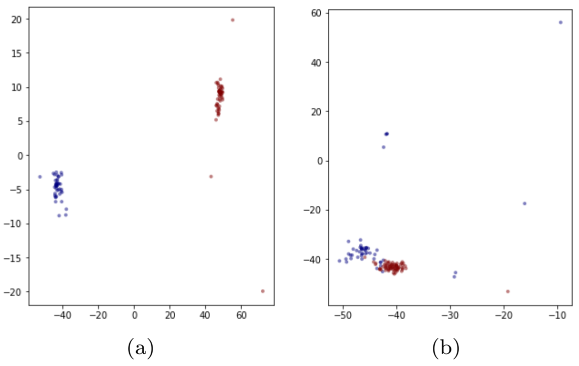

Figure 6: Top 2 t-SNE components of CNN features: (a) Figure 4 and (b) Figure 5.

6 Conclusion

We introduced 3 different architecture to solve the dog breed classification problem. Despite the size of the dataset, the accuracy of our best model was about 90%. A larger dataset and a different strategy to handle the class imbalance could improve our models. The codes are available on github.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *arXiv*, pages 1–23, 2017.

[3] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. pages 1–15, 2014.

[5] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. pages 1–14, 2014.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. 2015.

[7] L J P Van Der Maaten and G E Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[8] Songtao Wu, Shenghua Zhong, and Yan Liu. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, pages 1–17, 2017.