Capstone Project - Dog breed classifier

Riccardo Mattia Galli

Data scientist nanodegree

1 Domain background

The dog breed classifier project belongs to the computer vision domain. One of the problems in this field, which is object recognition, is often addressed via machine learning algorithms trained using deep learning, that has been proven to be particularly efficient. Specifically, deep convolutional neural networks (CNN) have reached state of the art results in learning to classify images starting from a labelled dataset used for training [1]. This neural network architecture differs from the classic multi-layer perceptron (MLP) in two main aspects:

1. While Multi-layer perceptrons (MLPs) only accept flattened vectors as input, CNNs can accept multiple dimensions input which results in preserving the pixel position (in relation to the sorrounding pixels) as information in a 2 dimensional image. Preserving this information gives to the model further context to extract features in a similar fashion to the biological retina receptive fields [2].
2. MLPs architecture is defined by one or more fully connected layers, with learnable parameters. On the other hand, CNNs architecture is defined by layers of learnable convolutional kernels which analyze the whole input (e.g.: image) by sliding across subparts of the input. This results in much less learnable parameters and makes CNNs faster then MLPs in the training.

2 Problem statement

The goal of the project is to build a machine learning model which can:

1. Effectively predict dog breeds from dog images.
2. When a human image is provided, predict the most similar dog breed.
3. Ignore pictures of other classes.

3 Datasets and inputs

The datasets are provided by Udacity and correspond to the following:

1. A dataset containing 13233 human images.
2. A dataset containing 8351 dog images.

The first dataset is used to test an OpenCV implementation of a Haar feature-based cascade classifier [3]. The second dataset is used to train a custom-made CNN classifier to recognize different dog breeds.

4 Solution statement

The first step of the final application is to recognize whether an input image contains a human and/or a dog. In order to achieve this, a ResNet50 model [5] and a pre-trained OpenCV face detector are used sequentially [3]. In a second step, a custom-built CNN classifier infer the dog breed from dog (or human) images. This can be achieved by using two different models. The first is a CNN model built from scratch with Keras [4].The second, given the complexity of the task, is a model built to exploit transfer learning. This model is based on the more powerful ResNet-50 architecture and is expected to have a better accuracy [5]. The second model is then implemented in the final algorithm that outputs the most resembling dog breed for human images, the corresponding dog breed for dog images and an error for other images.

5 Benchmark model

The two main benchmarks regard the models used for dog breed inference. The CNN model built from scratch should achieve an accuracy greater than 1% in the dog breed classification task (chance level). The ResNet-50 based model, on the same task, should achieve an accuracy greater than 60%.

6 Evaluation metrics

The models are evaluated following accuracy. In a classification task, accuracy represent the percentage of correctly classified images out of all the classified images, as shown by equation 1.

$$\frac{\text{correctly classified}}{\text{total classified}} \tag{1}$$

7 Project design

The project design consists in three main components:

1. The custom CNN classifier.
2. The Resnet-50 model used for transfer learning.
3. The final algorithm.

All models are then compiled with categorical cross-entropy as loss function and rmsprop [6] as optimizer.

7.1 Custom-built CNN classifier

The idea behind the architecture of the custom CNN classifier comes from a basic CNN architecture provided from Udacity. The neural network contains multiple hidden layers with convolutional kernels and maxpooling Layers. The number of output feature maps of each convolution increases as the depth of the network does, which is balanced by the subsequent maxpooling layers. This results in each convolution layer extracting more complex features than its predecessor without resulting too computationally heavy. Extracting more complex features from images is essential for exploiting CNN architectures to perform image classification. The two final layers (global average pooling and dense) take the output of the previous hidden layers and computes a probability for each dog breed class.

7.2 ResNet-50 for transfer learning

The idea behind transfer learning is to repurpose an existing and pretrained model for another task. This is possible, in image classification, since the hidden pretrained layers extract higher level features that are common in all types of images, like edges, squares and circles. The only change is in the last linear layer, which is substituted in order to match the current task classes. The training, then, consists of only updating the weights and biases of the last layer to match the new task. The ResNet-50 model is a perfect fit for this task due to its high performance in image classification. In this case the ResNet-50 model is repurposed to perform dog breed inference by changing the last layer to have 133 output neuron following softmax activation, which is the number of dog breeds we want to classify.

7.3 Final algorithm

The final algorithm is a combination of the previously described models. The Resnet-50 and OpenCV human face detector components check whether the input picture contains a human or a dog. If this is the case the algorithm predicts the resembling or actual dog breed with the ResNet-50 transfer model. Otherwise, if neither a human or a dog is detected, the algorithm returns an error.

8 Performance

The CNN model built from scratch achieved an accuracy of 4.2% in the dog breed classification task. The ResNet-50 based model, on the same task, achieved an accuracy of 82.6%.

9 Credits

Udacity for the materials.

References

1. Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
2. Luo, W., Li, Y., Urtasun, R. and Zemel, R., 2016. Understanding the effective receptive field in deep convolutional neural networks. In Advances in neural information processing systems (pp. 4898-4906).
3. Lienhart, R., and Maydt, J., 2002. An extended set of Haar-like features for rapid object detection. Proceedings. International Conference on Image Processing, 1, I-I.
4. Chollet, F., and others, 2015. Keras. GitHub. Retrieved from https://github.com/fchollet/keras.
5. He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
6. Zaheer M., Reddi S. J., Sachan D., Kale S. and Kumar S.. 2018. Adaptive methods for nonconvex optimization. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18) (pp. 9815–9825).