Monkey CNN Classification

Alessandro Carnazza CALES2805@GMAIL.COM

1. Model Description

The following project manage to built the best possible classifier using CNN to predict the Monkey species from images of the animals. In the present section I describe the best performing model I built.

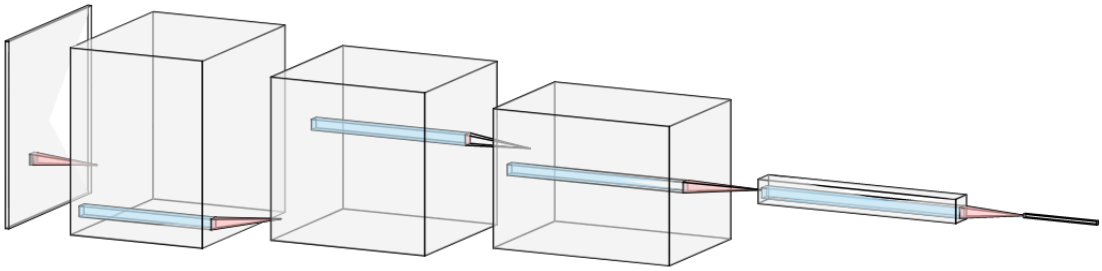

Figure 1: A simple representation of the proposed model.

It consist of 4 2D convolutional Layers and a convolutional classifier. The number of feature maps extracted in the layers using kernel 3x3, padding 1 and no stride, are 64, 128, 256, 512. After each Convolution I used ReLU as activation function. In order to reduce the dimension of the feature maps and speed up the computation retaining only the relevant information I used Max Pooling, with a 2x2 kernel, after the second and the third layers. Immediately after the fourth layer and before the classifier, the feature maps are passed through an Adaptive Max Pooling step which reduce the map size in 2x2. Finally using Cross Entropy the output is classified in 10 different classes. (A simple representation of the model can be seen in Figure 1.)

2. Dataset

The data I used are taken from a pubblic Domain dataset shared on the online community, Kaggle. It consist of 1369 images of monkey, labelled according to their species (10 different species) divided in training (1098 images) and test set (272 images). Further details, like species names and number of images for species, are presented in Table 1.

The images, as taken from the source, presented very heterogeneous dimension and shape, there were both squared images and rectangular, the latter with sparse ratio between height and width. To make the images homogeneous and to avoid distortion and loss of information, I choose to use the function *resize_contain* from the package *resizeimage*. This

Label	Latin Name	Common Name	Train Images	Test Images
n0	alouatta_palliata	mantled_howler	131	26
n1	erythrocebus_patas	patas_monkey	139	28
n2	cacajao_calvus	bald_uakari	137	27
n3	macaca_fuscata	japanese_macaque	152	30
n4	cebuella_pygmea	pygmy_marmoset	131	26
n5	cebus_capucinus	white_headed_capuchin	141	28
n6	mico_argentatus	silvery_marmoset	132	26
n7	saimiri_sciureus	common_squirrel_monkey	142	28
n8	aotus_nigriceps	black_headed_night_monkey	133	27
n9	trachypithecus_johnii	nilgiri_langur	132	26

Table 1: Latin and common Species' names and Number of images for training and test set for each classes.

function perform a resize of the original images, applying a background of a specified colour (I used white), to the images with different shape with respect to the one given as input. I gave as input the parameters 224x224, so it create squared images adding the background to the rectangular images, in their shorter side. (Figure 2). I chose to perform this operation in a preprocessing step, in order to decrease the dataset size and make it easier to share.

Figure 2: A sample of 32 images after the resize

I drawn at random the 10% of images from the training data in order to create a validation step, and I created a check step, in order to have the possibility to look at the data distribution inside the classes, so I set a seed which gave a distribution as uniform as possible.

3. Training procedure

Some transformation have been implemented, to perform *Data Augmentation*, in particular I applied horizontal and vertical random flip with probability of 33% and a normalization with parameters mean ad standard deviation equals to $[0.5, 0.5]$. The loss was computed using the standard multi-class cross entropy. All the models have been trained from scratch for 100 epochs on google Colab using the provided GPU, except for the models which clearly was overfitting, reaching 0 in the training Loss and showing unsatisfactory losses and accuracy for the validation set, which were interrupted earlier. Batch size was set to 10 for all the models, a low level chosen because of the saturation of the colab RAM. Adam is chosen as optimizer algorithm using a learning rate of $1 * 10^{-4}$.

4. Experimental Results

In order to find the best possible classifier, I tested several models performing an ablation study. Starting from a deeper and more complex model, with 5 convolutional layers and 1 dense layer, I obtained a validation accuracy of 68,89%, and a training accuracy of 98%. This values made me guess that the model was slightly overfitting, so I decided to simplify the model. So, I removed one convolutional layer, and I noticed an improvement in the accuracy of the model to 73,43%, I decided then to remove other layers in order to continue to reduce the overfit. Against mine expectations the model with only 3 convolutional layers and a fully connected layer presented a decline in validation accuracy, 68,79%, so I decided to remove instead the fully connected layer and classifier. So i built some model with only convolutional layers and a convolutional classifier, these models performed better. Starting from a model with 5 conv layers and reducing them until 3 i found models with accuracy over 80%. The best model, the one presented in the first section, reached 86.36% of validation accuracy so I chose it. It's performance on the accuracy of the test set are 83,21%.

Model	TrA	VaA	TeA	TrL	VaL	TeL
5 Conv layers + 1FCL + Classifier	98.38%	68.89%	64.29%	0.0449	2.1603	2.057
4 Conv layers + 1FCL + Classifier	100%	73.43%	70.71%	0.0004	1.5538	1.7709
3 Conv layers + 1FCL + Classifier	99.69%	68.79%	68.21%	0.0186	1.9332	1.7014
5 Conv layers + Classifier	97.35%	85.35%	82.14%	0.0873	0.7442	0.6843
3 Conv layers + Classifier	94.49%	82.63%	78.93%	0.1879	0.6994	0.6672
Best model (4 Conv layers + Classifier)	96.33%	86.36%	83.21%	0.1044	0.5585	0.6119

Table 2: Test, training and validation performance of the models.