Università degli Studi di Milano
Data Science and Economics

Machine Learning

Ivo Bonfanti
985892

Abstract

The project deals with the application of three different neural networks to perform binary classification on a dataset collecting colored images depicting chihuahuas and muffins. Specifically, the convolutional architectures considered are, in order of complexity, Lenet-5, a simplified version of Alexnet and the full Alexnet. As suspected, performance is inversely related with model's parsimony and the best score is achieved by the full Alexnet architecture. The code is implemented in Pyhton and the libraries used for building, training and testing the networks are mainly TensorFlow and Keras.

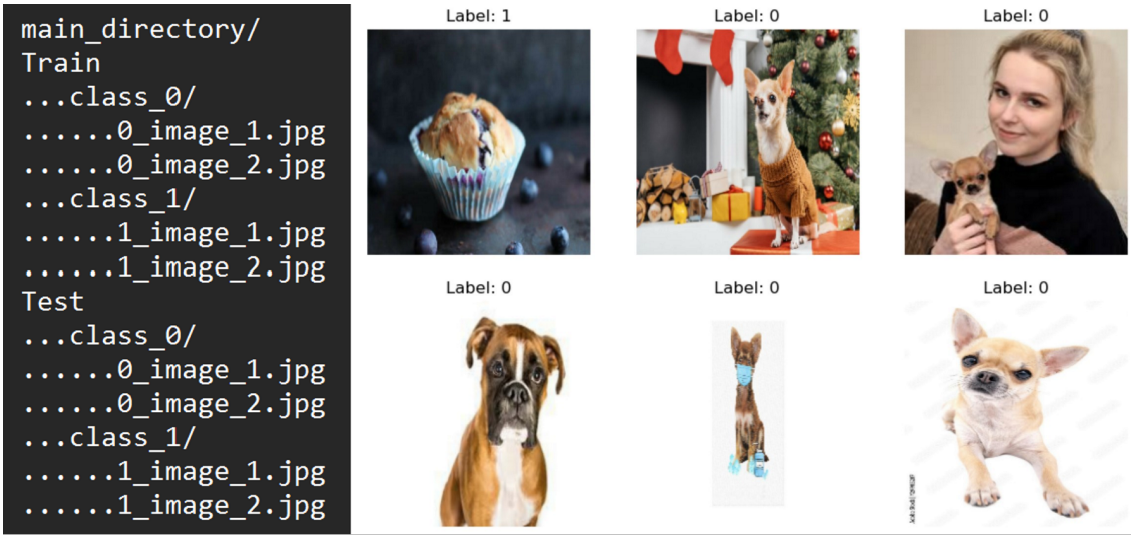
ACADEMIC YEAR 2022-2023

Contents

1 Dataset

The given dataset consists of 5917 colored images of chihuahuas and muffins, being them similar enough to challenge a neural network in the non trivial binary classification task. The pictures are stored hierarchically within folders, where the outer folders make distinction between train and test, while the inner determine the picture's label, 0 for chihuahuas and 1 for muffins. For a better understanding the directory structure is reported in figure 1. Given the way the images are stored, they are uploaded as a TensorFlow dataset by using the function `image_dataset_from_directory` from Keras. Here below a sample of images is provided.

Figure 1: Directory path and head of the dataset.



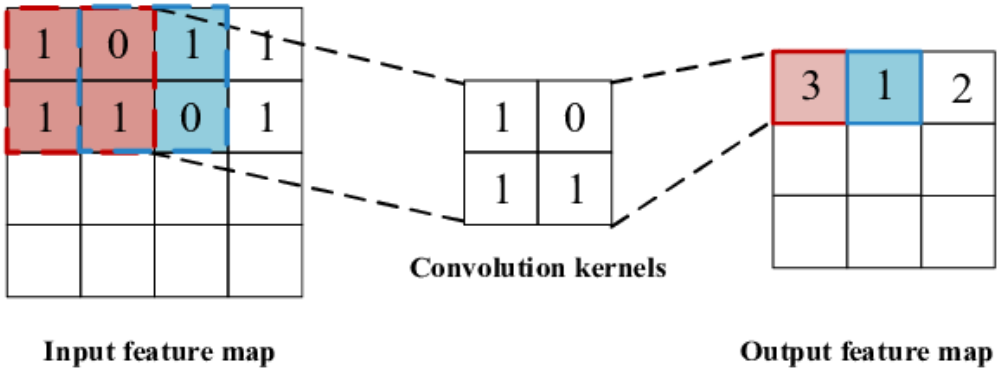
The pictures are uploaded three times, once for each network considered. That is due to their different requirements in terms of resolution and number of channels of the input data.

2 Convolutional neural network

A convolutional neural network (CNN) is a class of deep neural networks specifically designed for processing visual data which over the time demonstrated to be very effective in tasks such as object recognition, image segmentation or image classification. The main building block of a CNN is, as the name suggests, the convolution layer which is what distinguishes them from other deep learning models. It is made of kernels (or filters) of learnable parameters which convolve with the image thus returning a feature map (or activation map). Each component of the activation map, computed as the dot product between the region of the image considered and the filter, can be thought to be the output of a neuron. Each feature map is responsible for detecting a specific feature, or combination

of features, and it is associated with a specific kernel of trainable weights. For the sake of clarity the process is described in figure 2[1].

Figure 2: Convolutional process.



The kernel mechanism benefits of weights sharing since the same filter convolves over different sets of pixel values of the input, also known as receptive fields. This allows the network to learn translation invariant features as will be discussed further in the following sections. Thus, with respect a fully connected layer which connects each neuron in one layer to every neuron in the next layer without considering the spatial structure of the input, the number of weights is significantly lower and so is the computational cost.

While the firsts convolution layers focus on local regions capturing low level features, the deeper convolution layers are able to learn more complex patterns, or high level features, such as edges and shapes. Instead fully connected networks lack the ability to capture spatial hierarchies efficiently since they consider each input feature as independent. Such a hierarchical structure is inspired by the human visual and perceptual capacities.

What instead is shared by the two types of models is the learning algorithm called Back-propagation. It is the process in charge of updating the weights during training to minimize the error between the target and the predicted outcome. The magnitude of the update depends on the learning rate, while the direction upon the gradient of the loss function.

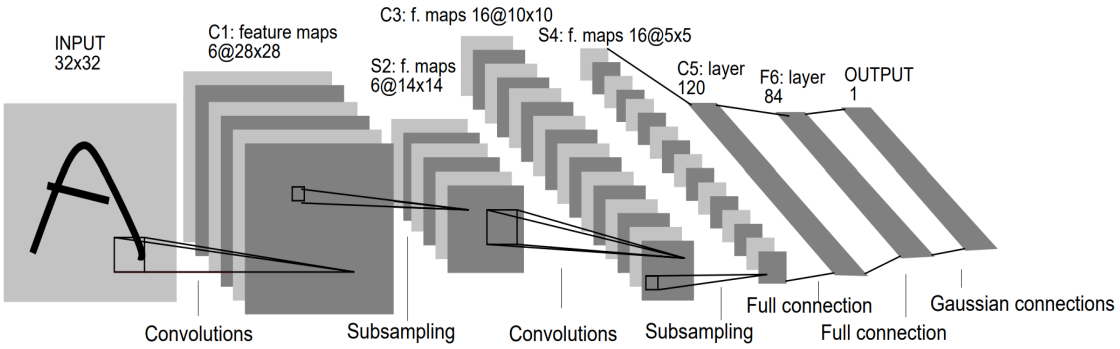
The analysis investigates how the performance, i.e. the accuracy for the classification task[1], changes with the complexity of the network, thus the number of parameters to be learned defined by its architecture. With this objective in mind three different network architectures are fitted to the above mentioned dataset. These are, in order of complexity, Lenet-5, a simplified version of Alexnet and the full Alexnet. The choice fell on them due the desire to understand the machanisms behind two very popular models which, at their time, pushed further the state of the art of computer vision. Furthermore, being them quite light models, their application is reasonable for the available computational capacity.

[1]The metrics used is the 0-1 loss.

3 Lenet-5

LeNet-5 is a convolutional neural network that was introduced by Yann LeCun and his collaborators in 1998. It is one of the pioneering CNN architectures and was designed for handwritten digit recognition tasks. It is made of 5 layers, including the input one, each learning a set of parameters. The input is a 32×32 pixel grayscale image, thus with only one channel. The first convolution layer has 6 32×32 features maps, with filter size of 5×5. The sub-sampling method used in the second layer is the average pooling[2], which outputs 6 feature maps of size 14×14. It follows another convolution layer with 16 kernels and another average pooling layer with the same characteristics of the previous one. The output is flattened and fed to the last 3 layers which are fully connected layers, the first counting 120 neurons, the second 84 and the last, being the output layer, only 1. The activation function used in the fully connected layers is the sigmoid[3], while for those convolution is the hyperbolic tangent[4]. The architecture, counting 60941 parameters, is illustrated in figure 3[2].

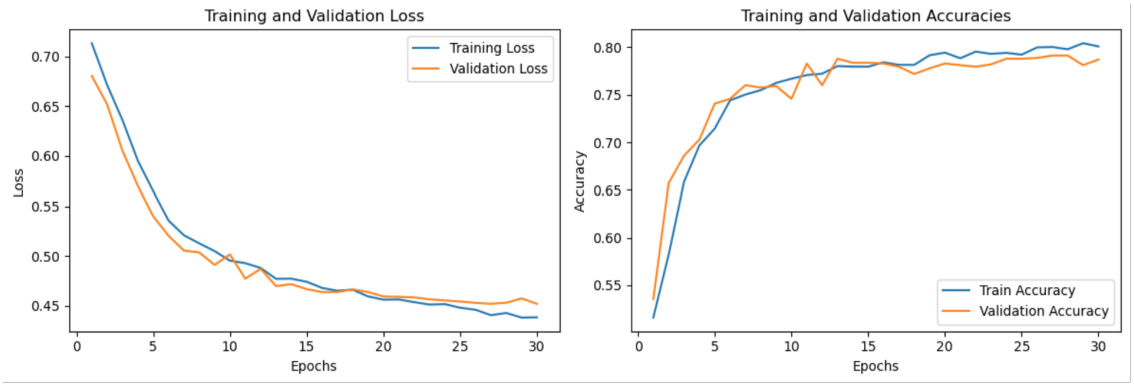
Figure 3: Lenet-5 architecture.



The batch size, the learning rate and the optimizer are tuned using the `RandomSearchCV` function from Sklearn. The 5 folds cross validation applied on the training set returned a batch size of 32, a learning rate of 0.0001 and the Adam optimizer[5] as the best combination of hyperparameters, with an average score across the folds of 77.75%. The best model is finally trained with 30 epochs[6] on the full training set, setting a patience of 8 through the method `earlystopping` from Keras to stop the training before the model eventually overfits. Results are shown in figure 4.

[2]A pooling operation that computes the average value for patches of a feature map, thus creating a downsampled feature map.
[3]$\sigma(z) = \frac{1}{1+e^{-z}}$
[4]$\tanh(z) = 2\sigma(2z) - 1$
[5]The Adaptive moment estimation optimizer is claimed to converge faster than stochastic gradient descent but also to generalize worse in most cases.
[6]One complete pass of the entire training dataset through the learning algorithm.

Figure 4: Lenet-5 results.



As expected the task seems too complex for such a simple architecture, indeed the training accuracy barely goes beyond the 80% threshold, nevertheless, given the lightness of the model and its fast convergence, it might be still useful for some applications. The highest accuracy scored on the test set is 79.14%.

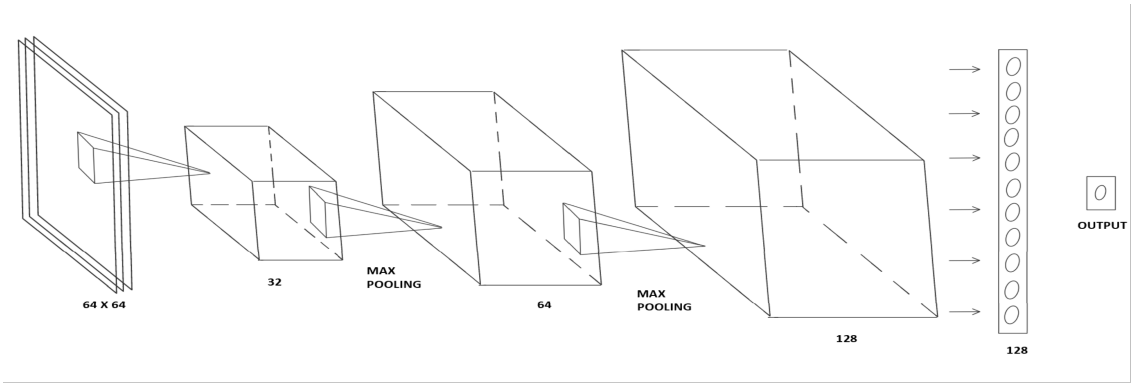
4 Alexnet Simplified Version

The next architecture tested is a simplified version of the classic Alexnet which will be described in detail in the next section. This version takes in input 64×64 colored rgb images, thus with three channels. This means that the input is no more two-dimensional but a three-dimensional tensor instead. This time the presence of the colors and the higher resolution make the pictures easy for the human eye to distinguish.

The depth of the network does not change with respect to Lenet-5, indeed it is made of 5 learnable layers as well. What changes is its width, thus the number of neurons per layer which increased significantly. The architecture is straightforward, the first convolution layer counts 32 filters of size 3×3, followed by a max pooling[7] layer to reduce the dimension of the activation maps, thus diminishing the number of parameters and speeding up computations. It follows a similar block of convolution and max pooling with the only difference that the convolution layer counts 64 filters, with unchanged dimension. The next layer is a convolution layer with 128 kernels, again with a 3×3 dimension. Once flattened the output is fed to the last two dense layers, respectively counting 128 and 1 output neurons. This time the activation function used in the convolution layers is the rectified linear unit (ReLu)[8], while the dense layers implement the usual sigmoid. The ReLu function, similarly to the hyperbolic tangent or the sigmoid, introduces non linearity allowing the network to learn more complex patterns. However with respect the other two this last it is much more simple, therefore it requires a much lower run time to reach convergence. The overall number of parameters is 2452801, significantly higher than Lenet-5. The architecture is illustrated in figure 5.

[7] A pooling operation that computes the maximum value for patches of a feature map.
[8] $\theta(x) = \max(0, x)$

Figure 5: Simplified Alexnet architecture.



The model best combination of hyperparameters found by the random search cross validation involves the use of a batch size of 64, the Adam optimizer and a learning rate of 0.001. The cross validated score of 86.39% shows a marked improvement over Lenet-5.

However once the model is trained over the whole training set, the error and the accuracy learning curves highlight some problems. Indeed the first epoch training error is huge compared to those following, proving some issues with weights initialization. Moreover the gap between the training and the test accuracies is marked, a clear sign of overfitting and the curves are also very much volatile. To handle the weight initialization issue the pixel values are rescaled from the range [0:255] to [0:1]. The rescaling is considered a good norm when dealing with convolutional neural network and in this case it proved to be effective. Instead to reduce overfitting some data augmentation is performed over the training set. The transformations applied are the random rotation and the horizontal random flip. Those techniques exploits the translation invariance property of CNNs and are intended to artificially increase the variability in the dataset with the aim to improve the generalization of the results during inference time. However the results did not improve much, overfitting still persisted and the accuracy remained highly volatile. Adding more augmentation layers or decreasing the learning rate did not help either. The highest accuracy scored is 90.88% and the results are shown in figure 6.

Figure 6: Simplified Alexnet results.

5 Alexnet

While Lenet-5 results obtained in 1998 on the MNIST dataset for hand-digits recognition were impressive, those achieved by Alexnet[3] in 2012 on the CIFAR-10 dataset with higher resolution images were definitely out of ordinary[4]. Indeed since then deep learning models reigns supreme in the computer vision field.

The Alexnet architecture is massive compared to that of Lenet-5, for both dimensions width and depth. It takes as input rgb images with a resolution of 224×224, slightly higher than before to provide more information to the model. The first convolution layer counts 96 kernels sized 11×11 with a stride[9] of 4 for both dimensions and it is followed by a max pooling layer with a 3×3 pooling window. Then it follows a similar block of convolution and max pooling layer, the first with 256 filters of size 5×5. Next there is a block of three consecutive and equal convolution layers with 384 filters of size 3×3 each. Then the output is max pooled again with a pooling window 3×3 and flattened. Finally there are two blocks made of a dense layer with 4096 neurons and a dropout layer with a rate of 0.5[10]. The last layer as usual is the output one and has a single neuron. The activation function used for the convolution layers is the ReLu while for the dense the sigmoid. In figure 7 the original architecture, counting 60302209 parameters, is illustrated.

Figure 7: Alexnet architecture.

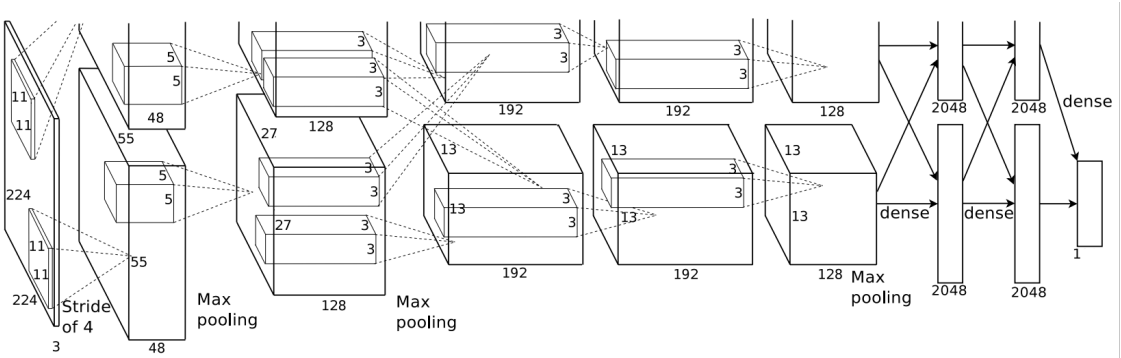


Figure 8: In the original paper all the layers are divided into two to train them on separate GPUs, while today, given the availability of a significantly higher computational power, that is not required anymore.

This time, given the complexity of the network and the very long time required to train it, the hyperparameter tuning via k-folds cross validation is unfeasible and thus skipped. Thanks to the experience gained from the previous outcomes the choice for the optimizer fall on Adam once again, while for the batch size 128 seems fair enough, since a smaller batch would further increase the training time. As for the learning rate is set to a value of 0.0005 as in the original Alexnet paper. Also in this case the pixel values are rescaled to the interval [0, 1] and some data augmentation transformations are applied during training to handle overfitting. Those are random translation, horizontal random flipping, random

[9]It determines how far the kernel moves at each step along each dimension.

[10]It nullifies the contribution of some random neurons while leaving unmodified the others. This is beneficial since it helps to prevent overfitting.

zoom and random rotation. Moreover the number of epochs is increased up to 150 with a patience set to 20. The model is trained on the whole training set and validated on the test set.

The model is run with the help of `Google Colab`, an online Jupyter based notebook granting GPU free access. While the GPU is considered less powerful than the CPU it is specifically designed to handle several tasks at the same time, while the latter copes better with sequential tasks. Such a feature makes it ideal for deep learning models since they make heavy use of linear algebra, this last being characterized by several but relatively simple calculations. Indeed Alexnet trains much faster on Colab, on average one epoch is trained in 20 seconds against the 200 required if run on the local machine. Therefore the platform is exploited to test different configurations of the model. Results are reported in figure 8.

Figure 9: Alexnet results.

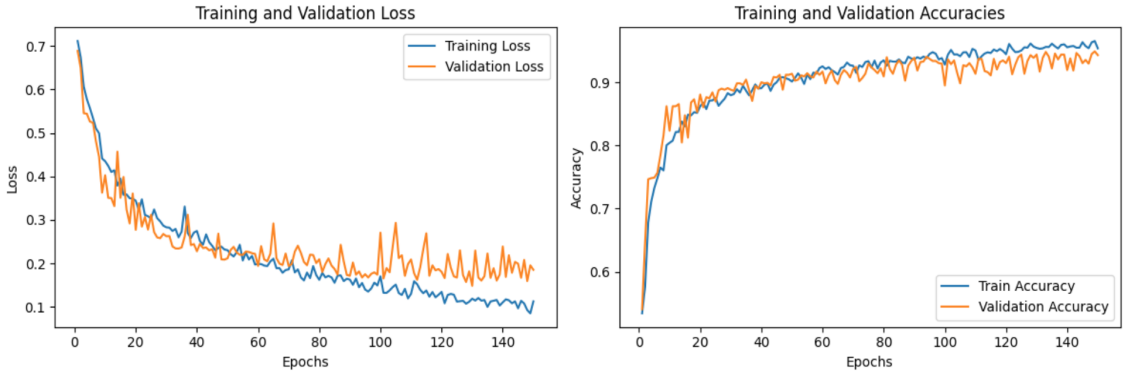


Figure 10

Alexnet moved the accuracy limit slightly further with a best score of 94.93%. However the main improvement with respect its simplified version consists in the reduced volatility of the curves and in a lower level of overfitting, implying a more reliable model.

6 Conclusion

The analysis concerned the application of three different neural network architectures on a dataset of images of muffin and chihuahuas with the ultimate objective to perform binary classification over it. The three deep learning models considered are, in ascending order of size, Lenet-5, a simplified version of Alexnet and the full Alexnet. Each model is built and fit using Keras and TensorFlow Python libraries. To tune the hyperparameters, specifically the learning rate, the optimizer and the batch size, and to obtain the cross validated estimates the Sklearn library is used instead. Moreover, the Google Colab platform is exploited to handle the full Alexnet model. The simplified Alexnet definitely outperformed Lenet-5 in terms of classification accuracy while the full Alexnet improved over its reduced version especially for its reliability on the way it generalizes.

References

[1] Huanna Niu Jiangbo Wang Jiefeng Liang, Tianjun Jing. Two - terminal fault location method of distribution network based on adaptive convolution neural network. *IEEE Access*, 2020.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[3] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems (NeurIPS) conference*, 2012.

[4] Pablo Caceres. The convolutional neural network - theory and implementation of lenet-5 and alexnet, 2020.