Hotdog / not hotdog

Jules Belveze
s182291

Abstract

In 2017, the HBO show *Silicon Valley* released an AI application that identifies hotdogs and not hotdogs. This report provides an overview of the methods used in order to implement our own convolutional neural network to perform this binary classification.

1 Introduction

This report aims to detail the creation's process of a convolutional neural network that can classify images into two different classes: *"hotdog"* and *"not hotdog"*. To do so we use a dataset containing images from ImageNet and we implement the classifier using the Python library *PyTorch*. The architecture of our CNN is a derived version of an *AlexNet* (Krizhevsky 2012) neural network. After various optimizations our CNN is able to reach $85,2\%$ of accuracy.

2 Background

2.1 Convolutional neural network

Convolutional neural networks are a class of deep neural networks that assign importance to various aspects in an image and are able to differentiate one from the other. The main idea, as opposed to a fully-connected neural network, is to reduce the number of parameters involved and reuse the weights. It is composed of a various number of layers composed by: a convolutional layer, which extract features, followed by an activation layer that introduces non-linearity and a pooling layer reducing the number of parameters.
AlexNet is a convolutional neural network that contains eight layers, where the first five are convolutional layers and the last three are fully connected layers. In order to improve the performance and stability of our classifier we will add a batch normalization, which is not used in *AlexNet*.

2.2 Optimizers

To determine the model parameters an optimization algorithm is needed, in order to minimize the loss function. All the different algorithms are based on gradient descent. We try three different techniques: stochastic gradient descent (Herbert Robbins 1951), adaptive moment estimation (Diederik P. Kingma 2014) and RMSprop.
The first one is using the cost gradient of one example at each iteration instead of using the sum of the cost gradient of all examples. In addition, it uses momentum in order to take better steps adapting to the loss surface. RMSprop uses the squared gradients to scale the learning rate combined with an adapted learning rate for each parameter. Finally Adam can be seen as a combination of both SGD and RMSprop.

2.3 Data augmentation

Regarding the fact that neural networks have a lot of parameters we need to provide them a consequent amount of data in order to get good performance. Our dataset is composed of 3909 images: 2047 used for the train set and 1862 for the test set. In order to be more accurate in our classification we perform data augmentation to "generate more pictures".

3 Methods

3.1 Data preprocessing

Since all the images have different ratios and resolution we decide to crop them from the center. We randomly crop images from the train set and crop from center the ones in the test set. In fact, we assume that each image is decently centered and thus that most of the information contained is in the center. Then we normalize each image in the range $[-1, 1]$ in order to reduce skewness.
Due to the small size of our dataset we perform data augmentation. However, in order to avoid training our model on meaningless images we only perform random horizontal flips and random cropping on images.

3.2 Neural network

As described in 2.1 our neural network was inspired by AlexNet. It contains seven layers: four convolutional layers and three fully connected ones. We also used a batch normalization layer after the first convolution layer. In order to ensure that for any parameter the network produces activation with the desired contribution we insert the batch normalization between the convolution layer and the activation layer. We use the rectified linear unit as our activation function to benefit from its sparse activation and faster training. Finally, we try both maximum and average pooling.

3.3 CNN's parameters tuning

As explained in section 2.2 we try three different optimization techniques: Adam, SGD, RMSprop and compare them. To be able to compare them we first tune the learning rate for each optimizer independently.

4 Results

Fig.1 is showing the evolution of the accuracy with respect to the learning rate using Adam on 10 epochs.

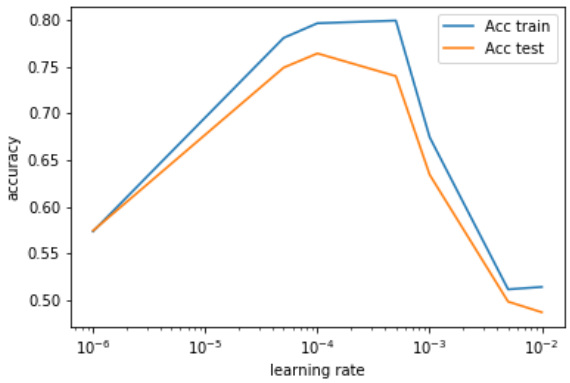

Figure 1: Error with respect to learning rate using Adam optimizer

As we can see in the above chart the optimal learning rate for the Adam optimizer is 10^{-4}. We process in the same way to get the optimal learning rate for the RMSprop and SGD and then we plot the training and testing accuracy for each of them in order to choose the one that performs best.

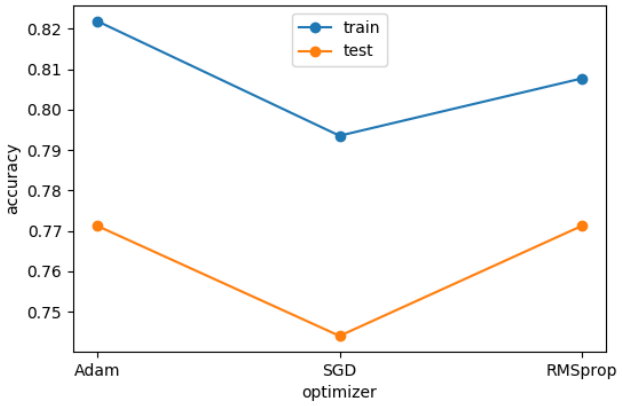

Figure 2: Error with respect to optimizer

As we can see in Fig.2 SGD leads to lower training and testing accuracies. Whereas Adam and RMSprop tends to have the same accuracy on the test even if Adam performs better while training. We thus decide to select Adam as our optimizer.

Now that we have built our neural network we want to analyze how well it is learning. In Fig.3 how the network

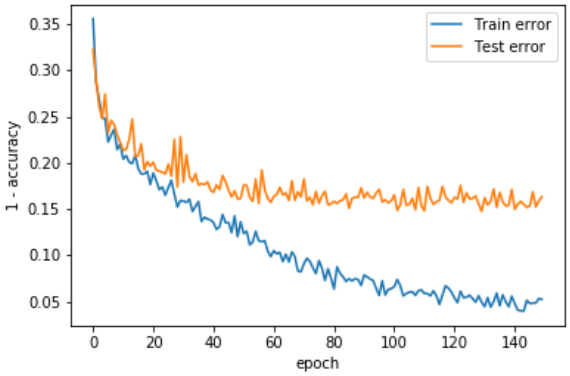

Figure 3: Error with respect to epoch

is learning through 150 epochs. It does correspond to what we would expect from a neural network, that is: the training error is decreasing with respect to the epochs. We have not reached the state of overfitting during our training but we can observe that after 50 epochs, the test error is not significantly decreasing. We finally reach a $85,2\%$ of accuracy on our test set.

Our first network was constituted of two dropout layers between the first and second layer and the second and third in order to avoid overfitting. It turned out that our network was failing capturing patterns in the train set (low accuracy) and removing those two layers increased the accuracy of our classifier without overfitting.

As mentioned above the highest accuracy we were able to reach is 85.2%. In Fig.4 we can see some of the images which were misclassified by our network.

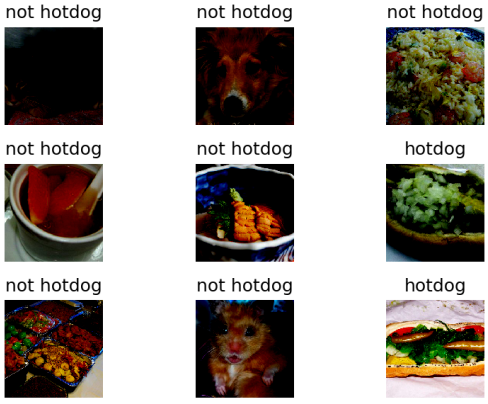

Figure 4: Misclassified images, with expected output above each of them.

It is difficult to find any obvious reasons why those images lead to a false label. However, we can pinpoint two interesting phenomena. First of all, if we look at the top left image we can see that the normalization lead to an almost black image where it is difficult to distinguish something.

Secondly, the middle right image is ambiguous even for a human eye. It is difficult to affirm that this is a hotdog. One of the reason for that might come from cropping. In fact, for big images cropping them from the center will lead to a loss of information.

5 Conclusion

As stated before we managed to reach a $85,2\%$ of accuracy, which lead space for improvement. After looking at the state-of-the-art it is possible to see that most of the people use transfer learning in order to perform better. Another idea could be to build a more sophisticate model.

References

Diederik P. Kingma, J. B. 2014. Adam: A method for stochastic optimization.

Herbert Robbins, S. M. 1951. A stochastic approximation method.

Krizhevsky, A. 2012. Imagenet classification with deep convolutional neural networks.