Image Classification with Convolutional Neural Networks

Chibuzor John Amadi[502623]

[1] University of Pavia, Pavia
[2] University of Milano-Biccoca, Milan
[3] University of Milano-Statale, Milan

Abstract. Convolutional Neural Networks(CNN) have revolutionized the world of computer science, computer vision in particular, with its ability to learn features from pixel data in task such as image classification. CNN is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. Image classification has been a field that showed great potential for deep learning. In this work, we will explore a simple binary classification of images.The loss function to be used for this work is the Binary Cross Entropy, the most suitable for binary classification. The model will be evaluated on the accuracy performance metrics. Subsequently, we we will carryout some experiment to see the strength of the CNN model.

Keywords: Kernel · Binary Cross-Entropy · Accuracy.

1 Introduction

1.1 Dataset

The dataset of our project is based of a collection of images of cats and dogs. This dataset was derived from the website of the Visual Geometry Group in the University of Oxford. This dataset contains 7349 images in total with a distribution of 6349 (2047 cats and 4302 dogs) images to the train set and 1000 (324 cats and 676 dogs) to the test set. All images have been resized to 160 x 160 pixels.

All images were organized in different files for more efficient processing

1.2 Basic Data Pre-processing

We begin by transforming our images from a picture format(png, jpeg) to tensors, transforming to tensor in image classification is an essential step in the deep learning architecture in order to get a unified organized data-frame and optimize the computational load. Normalization on the mean and standard deviation was carried out on the dataset. Typically in image processing normalization is usually calculated on the training dataset and test dataset to maintain consistency. We carried out normalization in batches with a batch size of 64.

2 Model Architecture

2.1 THE CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks(CNNs)[1] in the field of deep learning is renowned for its ability recognised patterns, extract features and achieving relevant outcomes especially in the field of image classification. On this project, the CNN is divided into 2 parts, a feature extractor and a classifier.

The Feature extractor is composed of a sequence of convolutional blocks which includes two convolutional layers, a normalization layer and a ReLU activation function. The other convolution layer in each block we can call, the Strided Convolution, this slides over the input images ass it applies its filter effectively reducing the spatial dimension of the resulting feature maps[2] which we will obtain. This process reduces subsequent computation on the following layers.

The Classifier In the classifier we compute the average feature vector, here we take the averages of the feature maps to a single value, also reducing the number of parameters and the computation in this network. Furthermore we apply a linear layer (full connected layer) to compute a score for each class. The linear layer is then followed by a sigmoid activation function to compute the probability estimate. In classification other activation functions can be considered but for the sake of our project, which is a binary classification, the sigmoid function is sufficient.

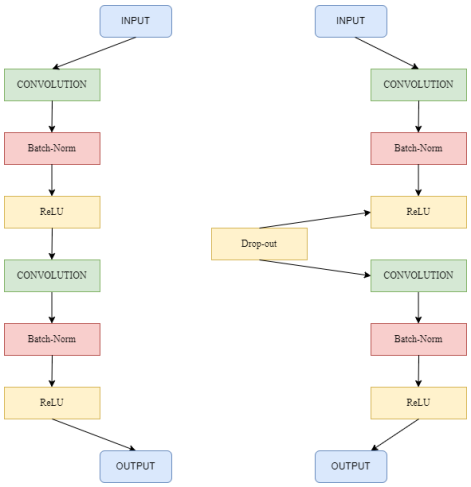

Fig. 1. Model Architecture of the CNN models involved in this project

3 Evaluation

3.1 Experiment details

The loss function of of the model we will be using is the Binary Cross-Entropy(BCE) which is most suitable for binary classification. The BCE loss measures the performance of a classification model whose output is a probability value between 0 and 1, its equation can be defined as

$$LBCE = -ylog(p) - (1-y)log(1-p) \tag{1}$$

To train the CNN model I utilized Adam[3], a variant of the stochastic gradient descent. Adam is an optimization method used to minimize the Binary Cross-Entropy loss, Subsequent to the Adam, some experiments were carried out with other optimization methods as shown in Table 1 1. To improve generalisation and prevent over-fitting of the model I explored the possibility of adding a dropout layer to the model. This dropout layer is placed just before the classifier in the model, and the results are displayed in Table 1 1. The CNN model was evaluated finally on an accuracy performance metrics

Table 1. *Results of CNN Model without a dropout layer and different Optimizers. Note the results of the accuracy is in a percentage value*

Optimizer	Accuracy train	Accuracy test
Adam	84.4	83.1
AdamW	90.1	79.8

Table 2. *Results of CNN Model with a drop-out layer and different Optimizers and different dropout rates. Note the results of the accuracy is in a percentage value*

Optimizer	train @ 0.5	test @ 0.5	train @ 0.3	test @ 0.3	train @ 0.2	test @ 0.2
Adam	76.6	78.4	87.5	85.6	81.2	82.4
AdamW	86.4	82.1	90.1	82.0	90.5	85.6

3.2 Result Analysis

Analysis the results as shown in Table 1 1 derived from the different variations of our CNN model without the dropout layer. From the table we can observe that when the model is trained with the adam optimizer we achieve a result of 84.4 on the training set but its performed by the adamW which produces a score of 90.1 percent. However, the adam optimizer performs better with respect to the test set with a score of 83.1 compared to 79.8 on AdamW. With these

evaluation, considering the model without the drop-out layer, it will be better to utilize the Adam optimizer as it performs better with unseen data, whereas the AdamW heavily overfits[4].

Table 2 2 shows the analysis of the CNN model with a dropout layer included. The main inspiration of this inclusion was to train the model to work better on generalization[4]. The model was also experimented on both the Adam and AdamW optimizer, and the tested on different drop-out rates. From the table we can see that the best results were achieved by Adam optimizer at 0.5 dropout rate and Adam optimizer at 0.2 drop-out rate. The optimal pick will be Adam at 0.2 dropout rate considering it achieved a higher accuracy on the Test.

4 Further Experiments

Experiments details With the optimal models being the CNN with a dropout layer, adam optimizer and drop-out of 0.2 and 0.5. We decided to test the accuracy of these models. A selection of images of other animals were used as the test set. These images are pictures of lions and cheetahs as cats and foxes and hyenas as dogs. The selection of animals was based on the facts that they have similar physical qualities to cats and dogs. Using 100 of these images, we resized each one to 160 x 160 pixels matching the initial training set. The results of the experiment are shown below Table 3 3

Table 3. *Results of optimal CNN Models on a different set of images*

rate	Accuracy train	Accuracy test
0.2	85.1	60.5
0.5	83.2	62.4

Result analysis/Conclusion Experimenting with other images with the top 2 dropout rates on the best CNN model generally over-fitted on the new data. The model on some occasions misclassified and as well classified lions cheetahs as cats and hyenas, foxes as dogs, these animals considerably share a lot of physical attributes and from a biological standpoint it could be understood. However, to improve the classification accuracy of our model, a solution would be to use, clear and precise images for our train-set, which could help the CNN model understand better the exact attribute of each image. Additional Convolutional layers is another possible solution which could be explored. In Conclusion, Image Classification with Convolutional Neural Networks have been achieved. The prospect for this technology with the availability of more computational resources and more data will lead to exciting prospects in the future if deep learning.

Fig. 2. Enter Caption

Fig. 3. *Isolated Images of the misclassified animals*

5 References

1 “Convolutional Neural Networks.” Wikipedia, Wikimedia Foundation, June 2019, en.wikipedia.org/wiki/Convolutional-neural-networkcite-ref-auto3-1-0

2 “Convolutional Neural Networks.” Wikipedia, en.wikipedia.org/wiki/Filter-signal-processing

3 ”Stochastic Gradient Descent.” Wikipedia, en.wikipedia.org/wiki/Stochastic-gradient-descentAdam

4 ”Overfitting” Wikipedia, en.wikipedia.org/wiki/Overfitting