Artificial Intelligence COM2028
Coursework Poster

Paula Rodríguez Jou 6650398

Department of Computer Science, University of Surrey

INTRODUCTION

The aim of this project is to produce a model that does an automatic classification of images of a human kidney tissue cell. The training set that will be used for this project is 150,000 images that were extracted from the Broad Bioimage Benchmark Collection (BBBC) image sets and each image was revised into 28 x 28 gray-scale. The chosen model is a convolutional neural network (CNN) which is a type of supervised learning and is the most common when it comes to classifying images.

ANALYSIS OF THE PROBLEMS

The 150,000-image dataset is divided into 8 different classes of different weight named 0-7, as can be seen in Figure 1. Each corresponds to a type of cell. In the type of model mentioned above the network learns to optimize the filters through automated learning and its basic unit is known as a neuron and it's based on the human ones.

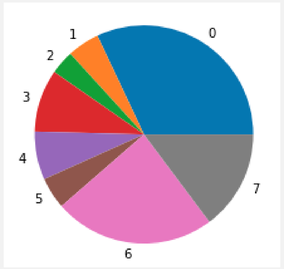


Figure 1

The development environment used is Google Colab and all the models were programmed in Python. In my first attempt, the model had 3 convolutional layers of sizes 16, 32 and 64. After training the model had a 0.67 test accuracy but a 0.50 validation accuracy and as can be seen in the wide distance between both values in figure 2 there was a clear case of overfitting.

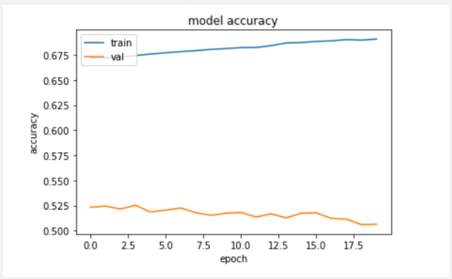


Figure 2

To solve this I applied regularization techniques, which consisted of adding dropout and max pooling layers to the final model. Other technical issues I had to solve were the sometimes long running times when executing the code and for that, I used BatchNormalization layers throughout the model so it would require less epochs to achieve the desired accuracy.

IMPLEMENTATION OF THE METHODS

The final model consisted of 9 convolutional layers grouped in groups of three with filter sizes of 32, 64 and 128 respectively as it can be seen in Figure 3. These layers had the ReLU activation method, and each trio was accompanied by their respective BatchNormalization, MaxPooling and Dropout layers.

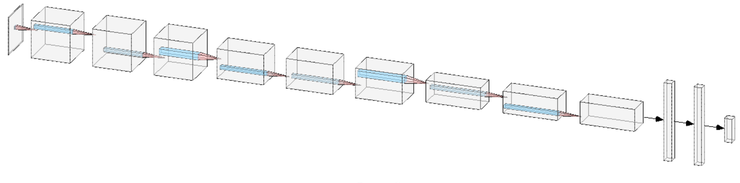


Figure 3

The BatchNormalization layer oversaw standardizing the inputs to a layer for each mini-batch and, by stabilizing the learning process, reducing the training times. The MaxPooling layer would down-sample an input representation reducing its dimensionality and allowing for assumptions to be made about features contained in not as important parts and the Dropout layer would drop or ignore some of the layer outputs, so the layer looks ang gets treated like a layer with a different number of nodes and connectivity to the prior layer. The final layers contain a layer of Flatten as the input can't be in cubic shape and three more Dense layers, the first two with 1056 units and the last with 8 units. All hidden layers use ReLU as the activation function as in modern neural networks it's the default recommendation and for the output layers, and I used softmax because there are 8 mutually exclusive classes.

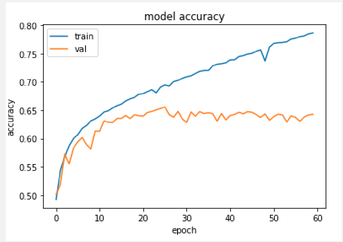


Figure 4

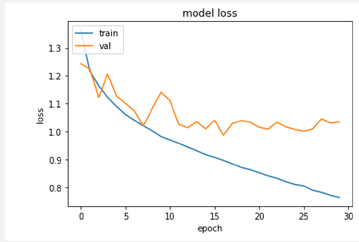


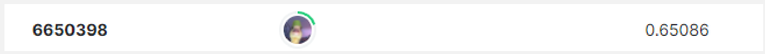
Figure 5

As can be seen in Figure 4, these changes approximated the values of the test accuracy and validation accuracy, thus solving the overfitting problems. Even so, as you can see in the graph, the validation values fluctuated slightly, so I decided to use a model checkpoint that would save the model that achieved the best performance so far. In figure 5 it can be seen that the loss of the model is quite close to zero, which indicates that the model is close to perform a perfect training.

RESULTS AND CONCLUSIONS

The final model reached the desired 65% accuracy for the coursework submission, even so after exploring more methods for image classification I have come to the conclusion that perhaps if I had used data augmentation techniques the accuracy of the model could still be higher since, as can be seen in figures 4 and 5, despite the fact that the model starts its training well, as the epochs go by, the two lines begin to separate, which is once again a sign of overfitting. Data augmentation techniques provide more data to the database since when applying changes such as rotating an image, so the model thinks that it is a new image. This would be beneficial for the training as I would be having a larger database and as it was seen at the beginning in the Figure 1 there are not an equal number of pictures for each type of cell so this would also count as a weight adjustment. Figure 6 show a picture of a few cells and the cell types they got assigned by the model. In summary, despite having fulfilled the objective of the work, which was to reach an accuracy of 65 percent, my model was not perfect, and the implementation of more techniques could be the solution to finish perfecting it.

Figure 6



Final accuracy result on Kaggle.

REFERENCES

1.https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/
2.https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a
3.https://machinelearningmastery.com/check-point-deep-learning-models-keras/
4.https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/
5.https://alexlenail.me/NN-SVG/AlexNet.html
6.https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/
7.https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/
8.https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529#:~:text=Examples%20of%20CNN%20in%20computer,i.e%2C%20weights%2C%20biases%20etc