***Image Classifier***

***Barel Kantsepolsky***

***Snir Yefet***

***Eilon Tsadok***

**Abstract:**

In this project, our task is to develop an algorithm to classify images of horses and zebras. We used a dataset which we found on Kaggle.com. We mainly investigated two approaches to address this problem. The first one is a traditional model. We used tensor flow 1.x and took every pixel of the image as feature for our model. For the second approach, we used Deep Convolutional Neural Networks (CNN) to learn features of images and trained Backpropagation (BP) Neural Networks. We tried various experiments to improve our performance on the test dataset, and finally got the best accuracy of 91.02% by the second approach.

**Introduction:**

**Motivation and Background**

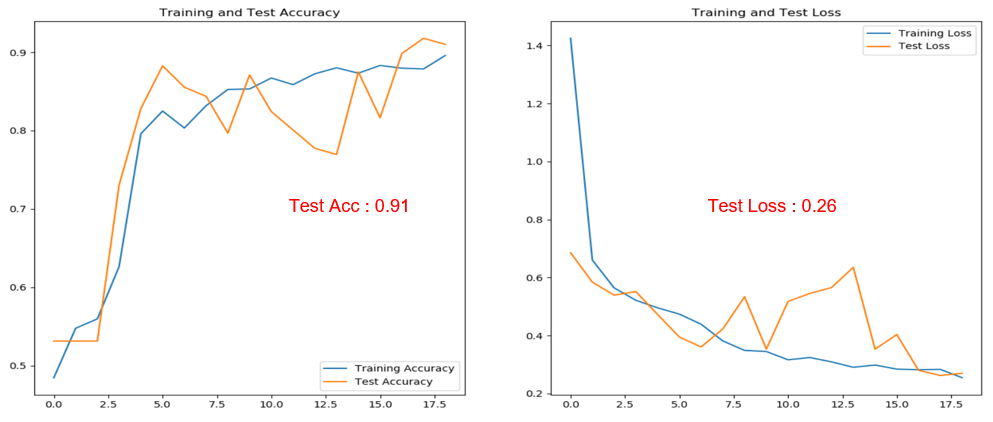
The Horse vs Zebra classifier problem which we found on Kaggle caught our eye the most among all kind of challenges there like Cats. Vs Dogs and more well-known classifiers. The problem of distinguishing images of horses and zebras is unique and challenging for many reasons. It is easy for humans, but evidence suggests that horses and zebras are particularly difficult to tell apart automatically since they share common features like similar shape and size. Furthermore, not many people have worked or are working on constructing machine learning classifiers to address this problem which make it even more challenging since we did not have many works to compare with. Among the works we saw were classifier based on color features, an SVM classifier based on a combination of color and texture features. In our project, we also would like to solve this problem and achieve higher performance. We tried different strategies which will be Explained extensively during the article.

**Project Description:**

The best approach in this project was using Convolutional Neural Network (CNN) after implementing some improvements learned from previous attempts.

The final model structure:

* 3 convolutional layers with max pooling and activation function of ReLU.
* a dropout layers in order to overcome overfitting
* fully connected layer with ReLU as activation function.
* fully connected layer with sigmoid as activation function.

The result of the final model on the test pictures:

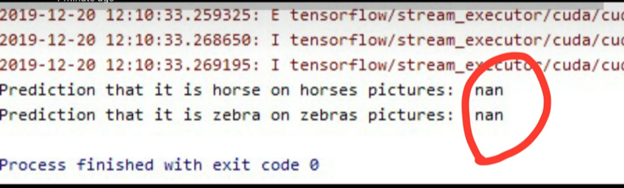
**Previous Attempts:**

**First Approach**

At first, we built the code learned in class of simple Logistic Regression while using our Dataset which contains many pictures of horses and zebras split to train and test sections.

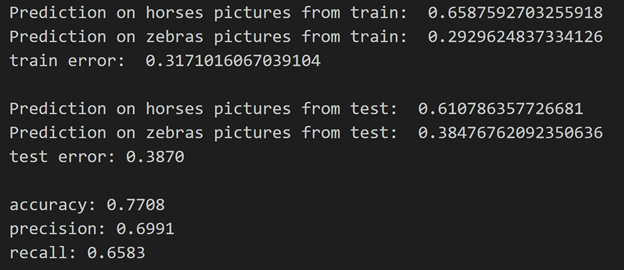
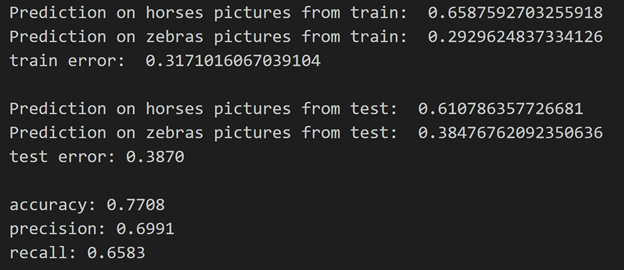
**Common Issues**

In the process a few problems had to be solved. First, each picture contains large amount of pixel multiplied the huge amount of pictures in our dataset multiplied 3 (Since every picture is represented as RGB ) causing long computing time. Secondly, dealing with different scale of numbers computing resulting “nan” values.

**Our Solutions**

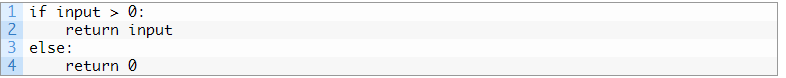
1. For making computing time more efficient we converted the images represented as RGB into grayscale.
2. In order to overcome “ nan “ values problem one should normalized his dataset. Normalization makes training less sensitive to the scale of features, so we can better solve for coefficients.

Notice

* It’s a recommendation to try both approaches grayscale and RGB. compare the result . color may influence result for the good.
* If we didn't scale our inputs in a way that resulted in similarly-ranged feature, sharing wouldn't happen very easily because to one part of the image weight is a lot and to another it's too small.

(Final results of the simple Logistic Regression Model)

In order to further improve the result two hidden layers have been added using ReLU activation layer. the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.The rectified linear activation function is a piecewise linear function that will output the input directly if is positive, otherwise, it will output zero.



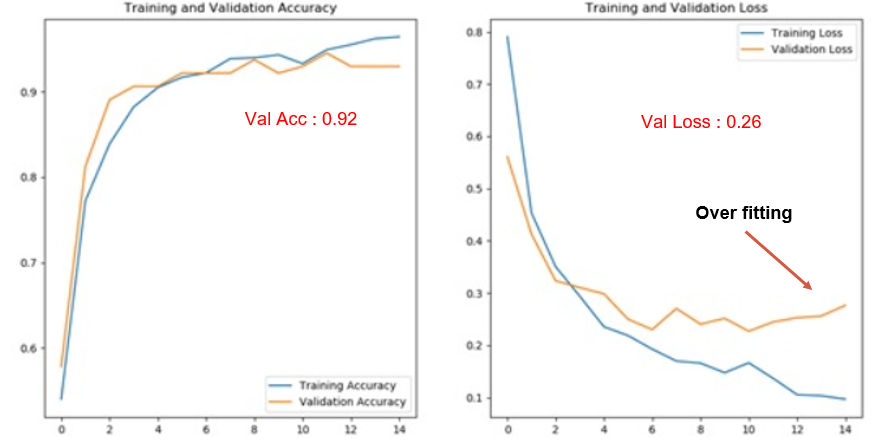
It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

**Second Approach**

Using convolutional neural network while using Dataset which contains many pictures of horses and zebras split to train, validation and test sections.

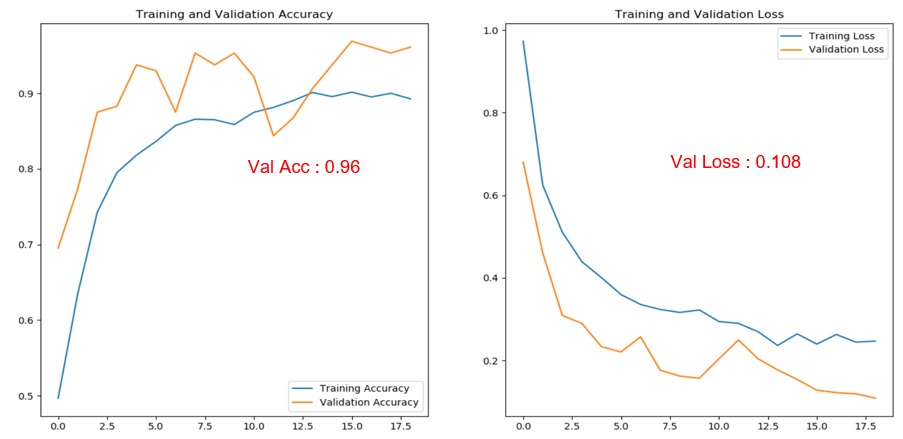
**Common Issues**

From the training and validation result it is noticeable that after epoch number 10 the loss result of the training section continued to decrease while the loss of the validation starts to become higher. Which shows about **overfitting**.



**Our Solutions**

* Insertion of two dropout layers. one before the second convolutional layer, and another one after the third convolutional layer.
* Augment the train section in the Dataset to overcome the overfitting. Each picture become many different pictures. Each of them with a little change from the original picture. Changes like rotation, shifting, zoom in. This helps expose the model to more aspects of the data and generalize better.



**Suggestion for improvements:**

Incase there is consistent decline in the "loss" like in the graph shown above, it’s recommended to try increase the number of epochs until minimum point is reached in order to achieve better results.

**Conclusions and Future Work:**

In this report, we first briefly explained our motivation of this project and showed some background materials. Then, precisely illustrated our task, including the learning task and the performance task. After that, we introduced our solution in detail, mainly including two approaches.

The first approach is a traditional Logistic Regression model, by which learned the classification model from the features as pixels. However, due to poor results of the model (77% accuracy), Second approach had to be taken. Our second model which is a trainable model that applies the CNN to learn features achieve good performance. The model consists of three convolution blocks with a max pooling layer and ReLU activation function in each of them. There's a fully connected layer with 512 units on top of it that is activated by a ReLU activation function. The model outputs class probabilities based on binary classification by the sigmoid activation function. The highest accuracy of this approach is 91.02%.

In the future, we will explore more to achieve better performance. For instance, we will try to change the architecture and parameter settings of the Deep Neural Network based on the feature visualization of different layers’ feature maps. We may also try object localization to eliminate the influence of complicated backgrounds.