DEEP-LEARNING-BASED SYSTEM FOR IMAGE CLASSIFICATION

MATTEO CIARROCCHI

September 17, 2022

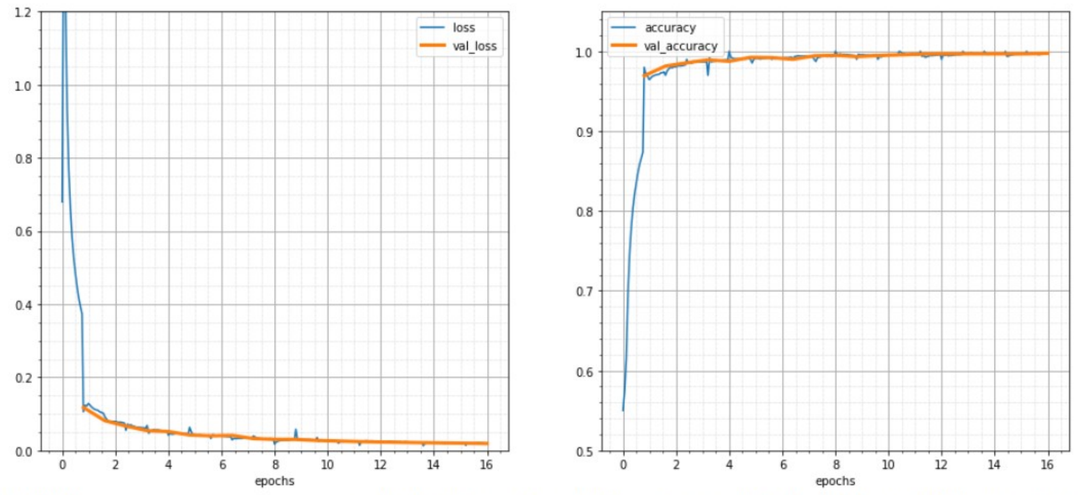
Contents

1 Declaration of independent work

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

2 Data selection and pre-processing

The project topic consists of creating a deep-learning-based system to discriminate images of the Comics Faces dataset into real and comic faces. The dataset is composed by 20000 instances, evenly split into two classes. It has been taken entirely and mainly analyzed using Tensorflow and Keras. The dataset is stored in a public GCS bucket which I have created using a TFRecordFile format in order to enable the parallel TPU implementation. The file has been realized through the code reported in the apposite section of the google Colab notebook, downloading faces and comics data and encoding them in the aforementioned file format. Images are resized from original $(1024, 1024)$ to a $(64, 64)$ dimensions, in order to avoid the explosion of parameters and fasten the computation. Besides, images are decoded with three channels instead of converting them to grayscale. Furthermore, image data are normalized using the tf.cast method dividing each value by 256 and obtaining, in this way, all the values belonging to the [0,1] range. Once uploaded, data are shuffled and then split into train-test



```
128/128 [==============================] - 5s 42ms/step - loss: 0.0196 - accuracy: 0.9968 - val_loss: 0.0189 - val_accuracy: 0.9975
```

Figure 1: Model output

with a 5:1 ratio. Training dataset is further split into smaller training and validation set, in order to evaluate models step by step and reserve test data for purely final observation of performances.
In addition, resulting datasets are stored in the cache to speed up the data retrieval in the training process, data are shuffled with reshuffling implementation at each iteration performed with the repeat argument in order to create an infinite dataset. In the end, data are batched with an initial value set to 100 and the argument "prefetch" is applied to the data with an AUTOTUNE attribute to fasten operations in a dynamic way by preparing successive elements while the previous ones are processed.

3 Theoretical framework

Thereafter, the report is structured as follows: section 4 describes the baseline model implementation, section 5 focuses on tuning of hyperparamter batch size based on the previous model structure, section 6 increases the complexity of the model implementing deep learning through a multi-layer Perceptron model, section 7 introduces regularization, section 8 implements a convolutional neural network, section 9 discusses the scalability of the proposed solution and reproducibility, and section 10 discusses experimental results and final remarks.

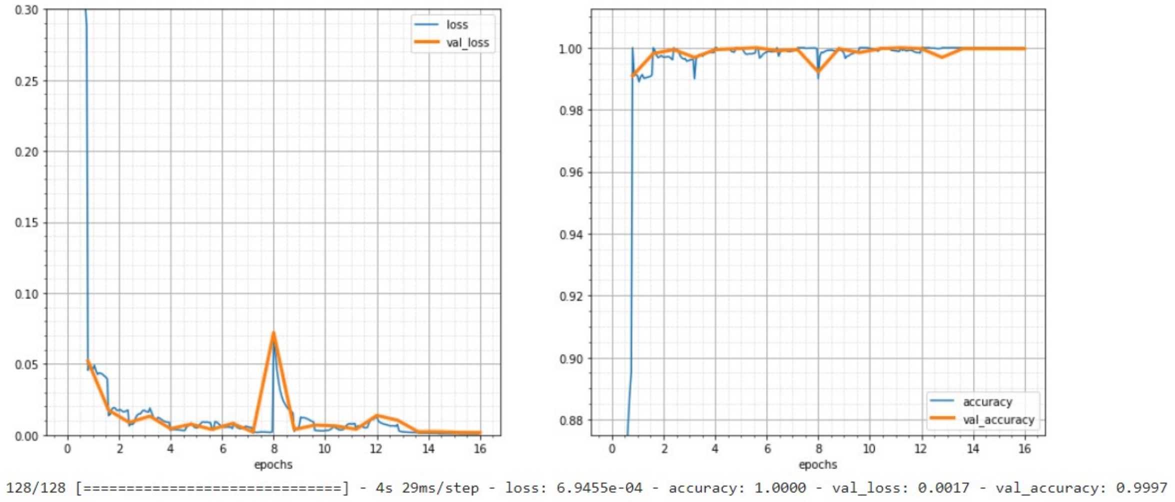
4 Baseline model

The first model is implemented specifying a 3D input layer of dimensions (64, 64, 3) representing resized image dimension and the number of image colours. The layer is flattened and an output dense layer is stacked with a sigmoid activation function. Having only one layer, the model does not address the non-linearity features of the data and does not capture the deep and abstract nature of data. This baseline model is compiled adopting a stochastic gradient descent as optimizer, binary cross-entropy and accuracy as metric. The model is fit, evaluated on validation data and it performs accurately despite its simplicity, as reported in figure 1.

Training and validation curves are tightly related in both loss and accuracy graphs, overfitting is not present as well as underfitting, but performances can be further increased with additional implementations.

5 Hyperparamter tuning

In order to obtain higher performances, an hyperparamter selection over batch size is performed using a train-validation method. The search for best value is performed over a batch size ranging in [80, 100, 125, 150] and best performing parameter is equal to 100, even though validation accuracies are close.



```
128/128 [==============================] - 4s 29ms/step - loss: 6.9455e-04 - accuracy: 1.0000 - val_loss: 0.0017 - val_accuracy: 0.9997
```

Figure 2: Multi-layer Perceptron training

”Number of epochs” hyperparamter has been always set equal to 20. This value is big enough for this set of data, since model predictors are able to reveal their pattern in terms of underfitting or overfitting. Besides, loss and accuracy curves flatten out after a number of epochs smaller than 20, such that the improvement of each epoch becomes negligible.

6 Multi-layer Perceptron

The models implemented so far are shallow because composed by only one layer and this does not allow the predictor to understand the intrinsic non-linearity of data. A proper deep learning encompasses an higher number of layers in order to train predictors according to highly complex function through the combination of inputs and weights and their modelling across layers. A multi-layer Perceptron preserves the structure of the previous models, introducing hidden layers with rectified linear activation function instead of sigmoid in order to avoid vanishing gradient drawback. The implemented model is composed by 2 dense hidden layers, respectively with 200 and 60 nodes for a total amount of trainable parameters of 2,469,921. Thereafter, another more performing optimizer is adopted for the model: Adam. Respect to gradient stochastic descent where the learning rate is constant for all the weight updates, Adam tailors learning rates for different parameters based on each gradient peculiarity.

As reported in 2, the predictor achieves better results: training loss curve approaches to 0 and training accuracy achieves perfect classification. Validation values are encouraging as well, even though it is possible to improve the overall performance.

7 Regularization

Regularization or dropout consists of pruning the dense layer connections skipping with a probability parameter the connection between one node and another of the successive layer. The considered model of this example is the same of the previous section, although dropout with rate 0.25 is introduced immediately before the output layer. The performance is not encouraging, since the curves becomes really wavy and all the values pinpoint an inefficient implementation. Training losses increase and accuracy decrease as expected since there is not overfitting in the Perceptron model depicted in figure 2.

8 Convolutional neural network

Another model is used to discriminate human-comics faces of the dataset: convolutional neural network. This deep learning abstracts data using high dimensional layers and it is able to spot features of image

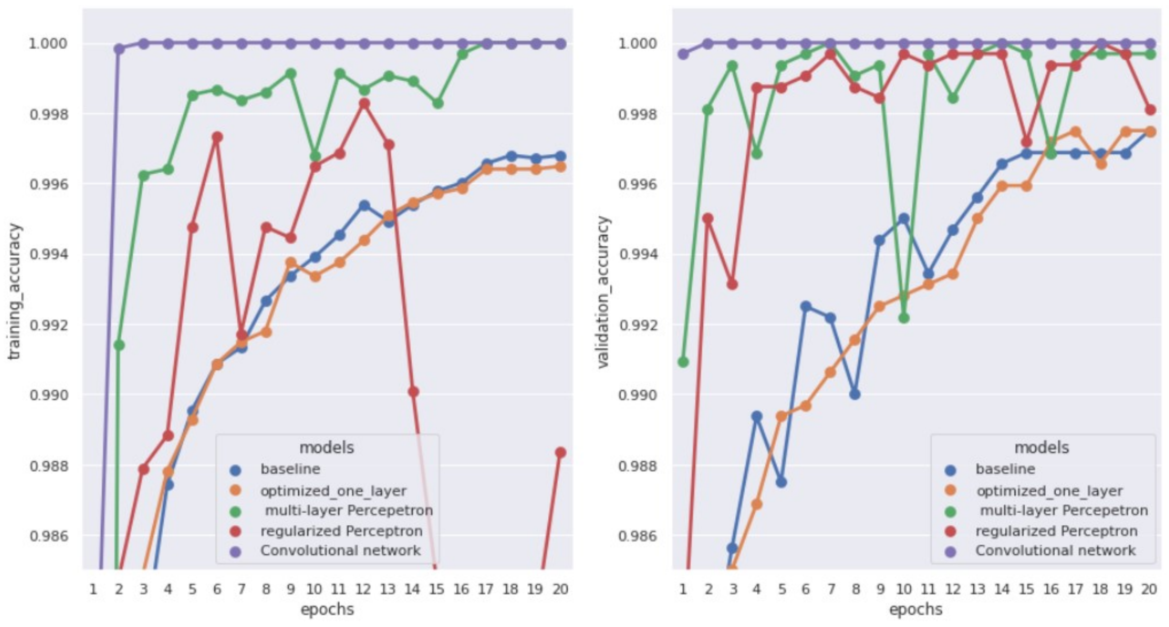


Figure 3: Training and validation accuracy

data like patterns in the image. The model is structured on the definition of several parameters: the kernel size, specifying height and width of the convolutional window, strides, which defines the movement of the window along the dimensions and filters, defining the dimensionality of the output space. The model is defined with 2 convolutional layers with kernel size respectively 3 and 6 and filters 12 and 24, strides constant to 2, rectified linear unit activation functions and a dense layer with 200 nodes stacked on top of the convolutional layers before the output layer. Performance of this model achieves best values, because the learning curves are steeper achieving accuracy equal to 1 in both training and validation values and are preserved constant as the number of epoch increases, while training and validation losses are lower than previous models.

9 Scalability of the proposed solution and reproduciblity of results

The proposed solution simulates scalability with input data implementing a Tensor Processor Unit (TPU) distribution strategy. This strategy is easily implemented in Google Colab, where there are available Cloud TPUs with a standard configuration composed by 8 devices for implementing distributed strategy. Larger configurations can be set with an higher number of TPUs, if available, in order to scale up with data size. The requirement is that data must be available on GCS bucket in order to make them accessible to the TPUs, therefore the dataset has been created as reported in section 2.
Perfect reproducibility of the results is not guaranteed. Indeed, parallel computation introduces source of randomness in the analysis even setting the seeds. Nevertheless, results are always similar. In order to prove it, the models have been saved and stored in GCS bucket and they can be used on test data in the last section of the notebook in order to be compared with the results of the runtime. The reported discussion of section 10 about evaluation on the test data, depicted in figure 5 and 6, refers to these models.

10 Results

The implemented deep-learning system is able to discriminate real vs comic faces with satisfactory results with all the models: as reported in figure 3, training and validation accuracy are generally greater than 0.99, while multi-layer perceptron and convolutional models perfectly classify training

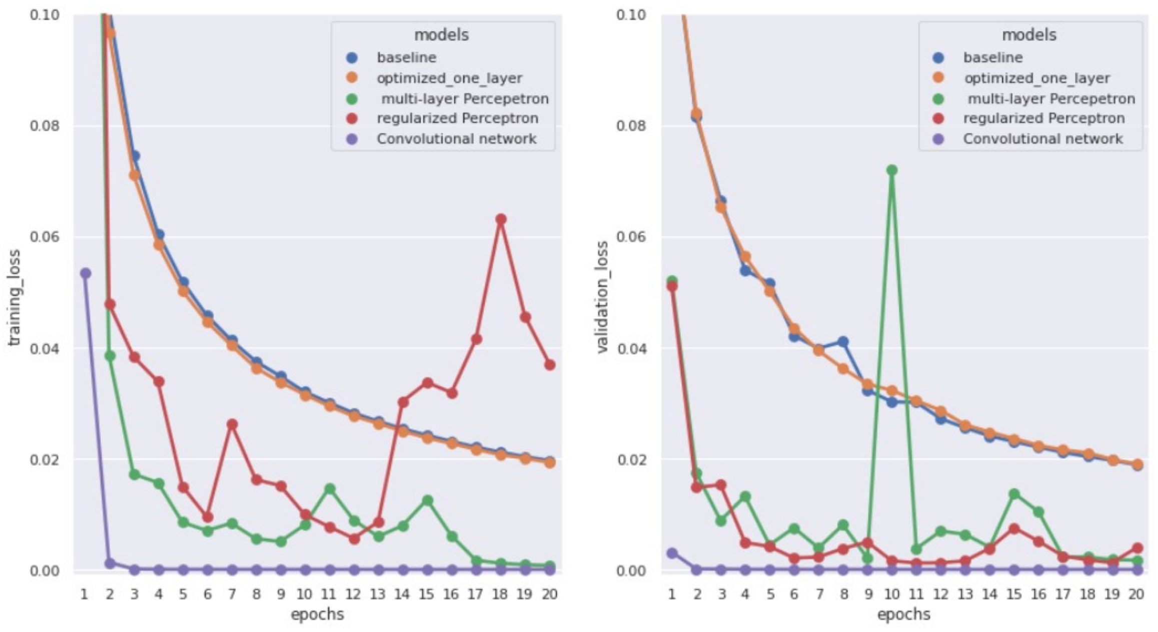


Figure 4: Training and validation losses

and validation instances. Nevertheless, this latter one outperforms the first one: the training process is much faster than the counterpart and it achieves an optimal accuracy in a lower number of epochs. Moreover, this argument is sustained by figure 4 as well, where the optimal model in terms of training and validation losses is clearly depicted.

Therefore, convolutional neural network is the best approach in order to implement a deep-learning system for faces discrimination according to results on validation data.
In order to check the consistency of the findings, a final evaluation of the models is performed using a test dataset of 4000 images which is unknown to the models. Results are reported in figure 5: accuracy pattern is consistent with the validation outcomes and the convolutional model perfectly discriminate images, while simpler models misclassify some instances, which are depicted in figure 6.
Nevertheless, deeper analysis is possible to further check the consistence of this result:

- images have been resized from a $(1024, 1024)$ to $(64, 64)$ dimensions. The number of pixels has been strongly decreased in order to speed up the code, sacrificing additional information inside the discarded pixels and better image visualization. Even though faces are perfectly discriminated with this setting, re-running the analysis with an higher number of pixels might result in a lower loss and therefore in a better model.

- throughout the analysis, one model has been constructed for each type of network structure or variation, corresponding to different sections of this report. In order to further analyze the dataset, it is possible to further vary the model structure as increasing number of layers or implement batch normalization.

- moreover, considering the degree of randomness in the results, it is possible to run the models several times and to consider an average of the outcomes. This has the effect to make the result more robust and to approach true capability of specified models.

In conclusion, this analysis recommends to adopt a convolutional neural network to implement a deep-learning based system in order to discriminate real vs comic faces.

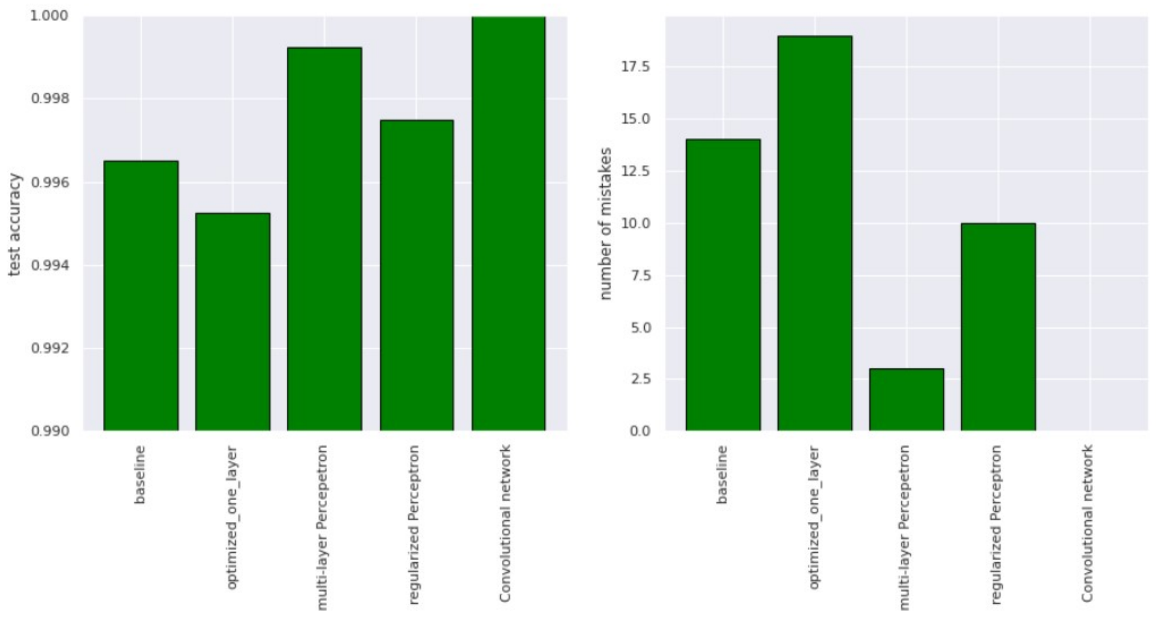


Figure 5: Test accuracy and number of mistakes

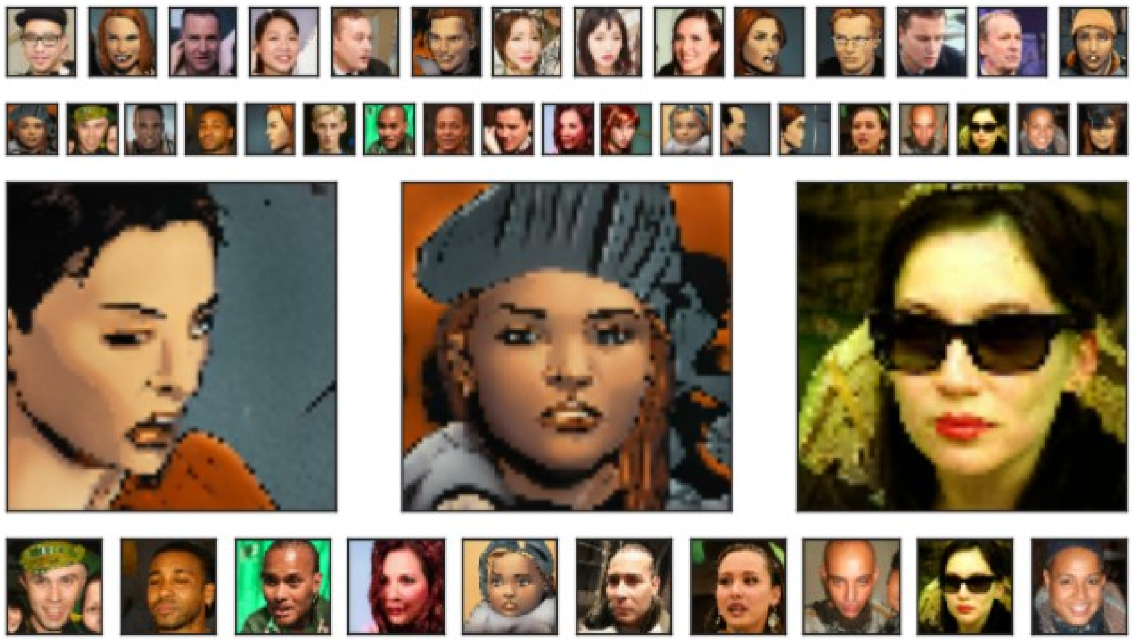


Figure 6: Baseline, optimized-one-layer, Perceptron and regularized Perceptron misclassifed images depicted in line.