# Report: Muffin vs Chihuahua

Randellini Mirto, Rapetta Luigi

## 1 Introduction

### 1.1 Muffin vs Chihuahua Problem

Image recognition applications have improved lately thanks to the recent technological advancements of neural networks. A classic problem in image recognition is binary classification. We will focus on classification of muffin and chihuahua pictures. This is interesting because the two subjects can be hard to distinguish at low resolutions. The state of the art of this technology allows us to solve the problem with good accuracy.

In this report we present our solution, which consists of the application of three network architectures: (i) fully connected neural network; (ii) convolutional neural network; (iii) vision transformer. The Keras framework was used for programming the algorithms. The results provided by the three neural networks will be shown and discussed in the following sections.

### 1.2 Work Environment

The work environment of choice is Google Colab. It's a cloud computing service that offers access to a series of GPUs and an easy-to-setup Python 3 environment. The latter feature seemed to fit our necessities the most.

### 1.3 Training the model

Our dataset is composed of jpeg images that we convert to RGB matrices to be fed into the neural networks. A neural network can be expressed as a parametric function

$$F_\theta(x) \approx y$$

that approximates the dataset $\{(x_i, y_i)\}_{i=1}^n$, where each $x_i$ is an image and $y_i$ is its class label, and $\theta$ is a set of weights and biases. The model is trained by using back-propagation with a binary cross-entropy loss and AdamW optimizer [7].

### 1.4 Data Pre-processing

To improve the accuracy of the networks, a series of transformations are applied to the images of the data set: (i) data rescaling; (ii) random horizontal flip; (iii) random rotation; (vi) random zoom.

Data rescaling changes the image data, consisting of RGB triplets of values between 0 and 255, to fit the range of values between 0 and 1. This transformation is always applied and makes the data more treatable by the network. Random transformations, on the other hand, try to augment the data set by adding more spatial variety to the images, which is helpful to further generalize the features learned by the network.

Some examples of random transformations can be seen in Figure 1. Data Pre-processing is applied by the first hidden layer in every network proposed in this report.

Figure 1: Random rotation and flip applied to an image.

## 1.5 Hyperparameter Tuning

An important aspect of improving the predictive algorithm is the choice of hyperparameters. In our solution, we adopted the *Weights & Biases* framework for tuning a selected set of hyperparameters.

### 1.5.1 Weights & Biases

Weights & Biases is a machine-learning framework that facilitates the tuning of hyperparameters. Its application in this project is justified by how much it automatizes the tuning task (compared to Keras), and by monitoring utilities accompanied by a vast offering of graphs.

Tuning is done with a *sweep*, a series of training runs to improve validation accuracy by changing the hyperparameter configuration, following a predefined strategy. The sweep can be configured by specifying the hyperparameters to tune, as well as non-changing parameters. Among the latter, there are (i) the number of learning epochs; (ii) early stopping parameters.

The sweep can be monitored by a dashboard accessible via a browser. The accuracy values of each run get recorded and associated with the hyperparameter configuration that led to such results.

### 1.5.2 Chosen Hyperparameters

The chosen hyperparameters to tune are the following: (i) initial learning rate; (ii) dropout rate; (iii) weight decay rate. The initial learning rate is crucial because it determines the step size for updating model parameters, impacting both convergence speed and stability during gradient-based optimization. The dropout rate stops the propagation of activation signals from the previous layer to the next with a certain probability. This is a regularization technique, useful to limit overfitting [2]. The weight decay rate is a parameter of the loss component that punishes larger weight values.

Hyperparameter values for each configuration are chosen at random within specified distributions. The initial learning rate and the weight decay rate are sampled from a log-uniform distribution ranging from $10^{-5}$ to $10^{-2}$. Dropout rate is sampled uniformly from the following set of values: {0.1, 0.2, 0.3, 0.4, 0.5, 0.7}.

Batch size was initially considered as a hyperparameter. Following our tests, since a seemingly optimized batch size did not considerably improve accuracy, we opted for removing it from the set of hyperparameters to optimize. Batch size was fixed at a value of 32.

Image size was also considered as a hyperparameter. Yet, it was discarded for the following reasons: (i) accuracy would improve anyway with image size, therefore dedicating computation time to an optimization task that would always converge on the highest possible resolution didn't seem useful; (ii) higher resolutions decrease training speed significantly. Currently, all images in the data set are scaled to a fixed resolution of $96 \times 96$ pixels.

## 1.6 Experimental Methodology

The following sections summarize the flow of the experimental methodology. For brevity's sake, details on setting the environment up and gathering the data set will be omitted.

### 1.6.1 Sweep

Hyperparameter tuning is done with a sweep, as mentioned earlier. The total amount of runs of a sweep is set to 25. Each run is set to have maximum 50 learning epochs. To prematurely finish a run that shows no sign of improvement and to avoid overfitting, the early stopping mechanism is enabled. Early stopping is set to begin after 5 epochs, and its patience (the number of non-improving epochs necessary to trigger early stopping) is set to 3.

After all training runs are completed, the best hyperparameter configuration is retrieved.

### 1.6.2 5-fold Cross Validation

After the sweep is finished, a more precise estimation of the generalization accuracy is computed through a 5-fold CV. The best hyperparameters and the whole dataset are used.

### 1.6.3 Model Evaluation

The last step is the evaluation of the model. After training the model on the training set only, the resulting predictor is tested with examples from the test set, and a final test accuracy value is returned.

## 2 Fully Connected Neural Network

The first network we propose is a Fully Connected Neural Network. Each neuron layer is fully connected with the following. It grants a simple architecture with no inductive bias for images.

This fully connected NN represents our baseline for later comparisons with other architectures.

### 2.1 Architecture Details

The network we propose accepts the input as RGB triplets, which are then augmented and converted into greyscale. The latter step simplifies the image data and makes the network focus on spatial features regardless of color. This was necessary because using three input channels led to an impractical network size for training. The input gets flattened before being processed further.

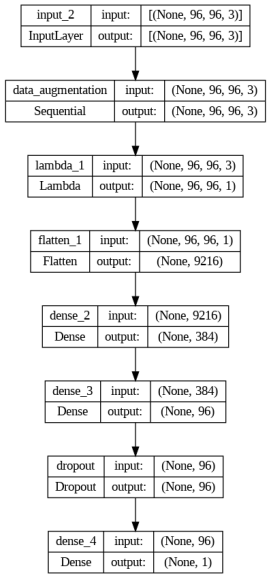


Figure 2: Proposed Fully Connected NN.

Following the input layer there are two dense layers with ReLU activation. The number of neurons for the first is four times the image's length in pixels (96 * 4 = 384). The second layer has a number of neurons equal to the image's edge in pixels (96). The two dense layers are followed by a dropout layer. The aforementioned three layers have been chosen for the

simplicity and performance of the network. Bigger and more layers were initially considered for this network, but they often led to overfitting and slow-to-train models.

The output neuron is set to have a sigmoid activation function.

## 2.2 Results

Sweeps return the highest validation accuracy which stands around 0.67, as shown in Figure 3. The highest value obtained from our experiments is 0.6793.

The accuracy computed by 5-fold CV is 0.6652. The test accuracy computed is 0.6394. A test on two random images is shown in Figure 4.

These results suggest that a fully connected NN is not the best architecture for dealing with image classification problems. The next architectures have been shown to be more effective for this kind of task [6].

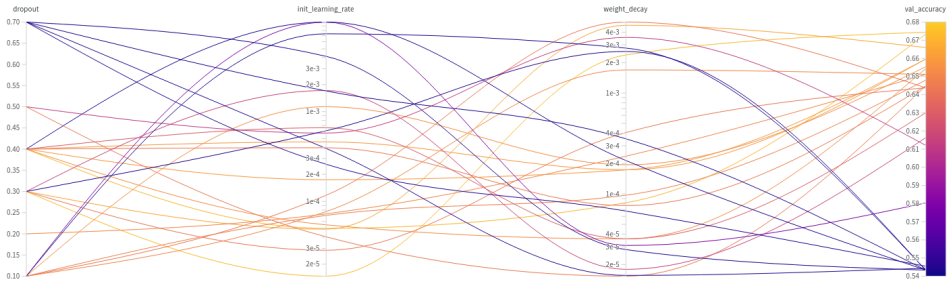


Figure 3: Sweep for the fully connected NN.


Figure 4: Two random images tested with NN.

# 3 Convolutional Neural Network

The second network we propose is a Convolutional Neural Network. This network architecture is intended for image recognition tasks. The convolution step of a CNN is inspired by the visual cortex of mammals. A neuron of the visual cortex is capable of receiving only a small portion of the total visual signal, bounded within a local region of the field of view. This local region is called *receptive field*. The features a neuron can deduce are observable within its receptive field.

Such a mechanism is reproduced in a CNN. Image data is organized in a matrix, and a neuron of a *convolutional layer* convolutes the data that can be observed within its receptive field, called *patch* in this context. Convolution is comparable to applying a filter to image data, where the weights of the connections between the neurons holding the data and the convolutional neuron represent the filter itself. The *feature map* realized by this convolutional layer describes the deduced features.

Feature maps can be processed further by other convolutional layers, as well as *pooling layers*. These layers reduce the dimensionality of feature maps, following a predefined strategy, and serve the purpose of generalizing spatial features. This mechanism allows a spatial feature to be detected with some degree of flexibility on where it's exactly placed on the image. For instance, the somatic features of a chihuahua can be deduced whether they're placed at the center of the picture or anywhere else.

This architectures scales far better than a fully connected neural network with respect to image dimension.

## 3.1 Architecture Details

As in the previous model, the input is accepted as RGB triplets and passed to the augmentation layer. The first convolution is then applied, which produces a feature map of half the size of the input image where 64 filters are being applied. Data gets normalized relative to its batch.[1] The normalized data is then passed to an activation layer where ReLU is being used.

What follows is a series of convolution, pooling, and activation layers shown in detail in Figure 5. A layer of interest is the *separable convolution* layer. Separable convolution consists of first performing a depthwise spatial convolution (which acts on each input channel separately) followed by a pointwise convolution which mixes the resulting output channels [5]. In

[1] *Batch normalization* applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1 [3].

parallel to the separable convolution a normal convolution is applied and added using a skip connection. This series of convolutions used to be much deeper during development, and similarly to the previous network, it often resulted in mediocre results. Given the small scale of the dataset, we opted for a less deep network. This design choice paid off, as elaborated in the next paragraph.

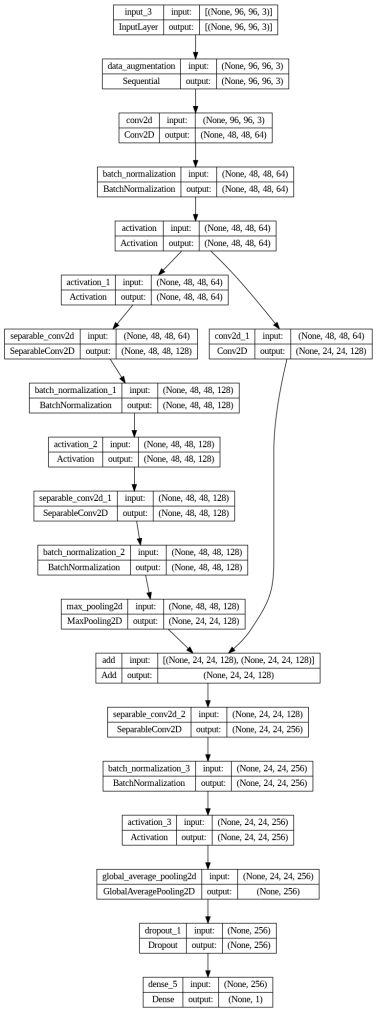


Figure 5: Proposed CNN.

### 3.2 Results

Sweeps return the highest validation accuracy which stands around 0.9, as shown in Figure 6. The highest value obtained from our experiments is 0.9049.

The accuracy computed by 5-fold CV is 0.8923. The test accuracy computed is 0.8699.

Compared to a fully connected NN, a CNN seems to be more capable of dealing with image classification tasks. Classification of random images taken from the test set seems more precise and reliable than previously. A test on two random images is shown in Figure 7.

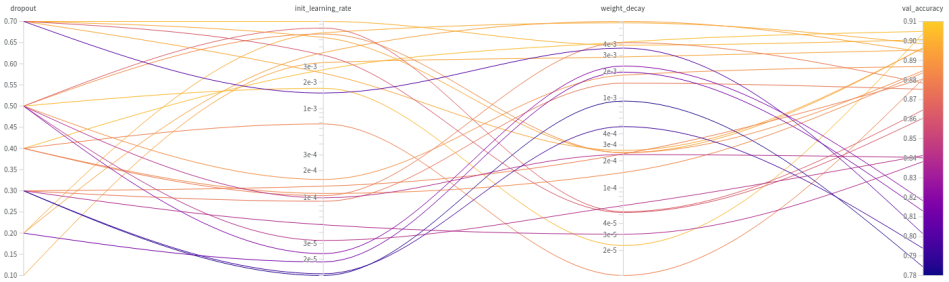


Figure 6: Sweep for the CNN.


Figure 7: Two random images tested with CNN.

# 4 Vision Transformer

The Vision Transformer is the state of the art for image recognition tasks [1]. Literature has shown that a ViT can surpass a CNN in performance on large-scale problems. ViT is a variant of *Transformers* for image recognition.

Transformers were originally designed for natural language processing (NLP), in response to the low performance of RNN on this kind of task.

To handle relations among words that are set far apart in a sentence, transformers implement an *attention* mechanism that relates input words with each other, regardless of their placement.

To handle image recognition tasks, transformers are used with some modifications. The attention mechanism mentioned before is not practical for images due to them being of large size even at low resolution. Therefore, instead of applying attention to the whole image, it gets applied locally to *patches*. The division of an image into patches is shown in Figure 8.

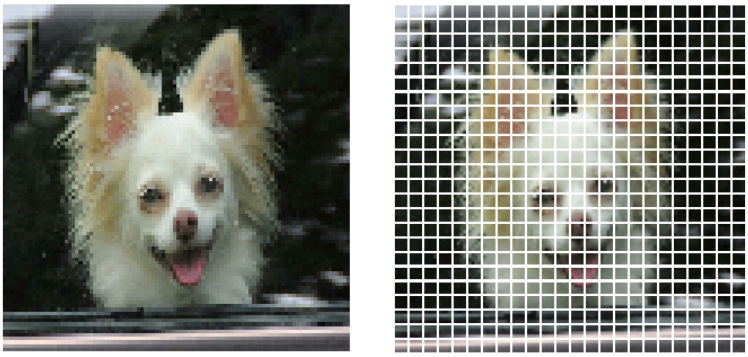

Figure 8: A picture divided into patches.

Patches are then unrolled into a sequence, and enriched with positional information relative to the sequence. Patches are encoded internally as a one-dimensional vector and passed to a transformer, where attention is applied [1].

## 4.1 Architecture Details

The input is taken as RGB triplets, which are immediately augmented, divided into patches, and then encoded. What follows is a pattern of layers where attention is involved, which can be seen repeated three times in Figure 9.

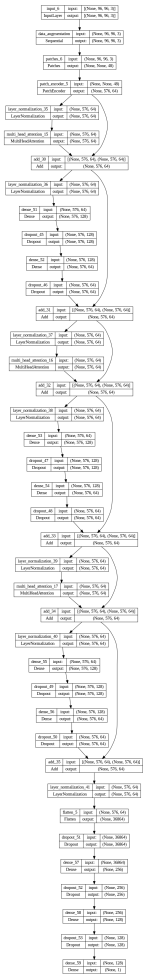

Figure 9: Proposed ViT.

Input vectors are normalized[2] and fed to the self-attention layer. The output of self-attention is then added to the input vectors. The newly computed vectors are then fed to a series of dense layers with ReLU activation where dropout is applied twice. The processed vectors are then added to the input vectors of the dropout sub-net.

[2]This normalization is done per layer, not batch. *Layer Normalization* normalizes the activations of the previous layer for each given example in a batch independently, rather than across a batch like Batch Normalization. i.e. applies a transformation that maintains the mean activation within each example close to 0 and the activation standard deviation close to 1 [4].

After the repeating pattern of layers, data gets flattened and fed to a series of dropout and dense layers. Each dense layer has progressively fewer neurons until the final output neuron is reached and sigmoid activation is applied.

## 4.2 Results

Sweeps return the highest validation accuracy which stands around 0.88, as shown in Figure 10. The highest value obtained from our experiments is 0.8858.

The accuracy computed by 5-fold CV is 0.8059. The test accuracy computed is 0.8725. A test on two random images is shown in Figure 11.

Compared to the fully connected NN, the new model is more reliable for predicting classes correctly. Scores, however, are very comparable to CNN's.

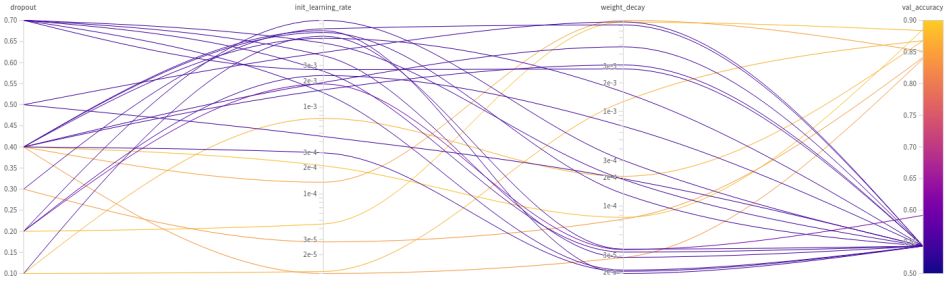


Figure 10: Sweep for the ViT.


Figure 11: Two random images tested with ViT.

# 5 Conclusions

After experimenting with two more advanced architectures, it became apparent that a fully connected NN is not suitable for reliable image classification, even for a simple problem such as the one presented in this report. CNN's and ViT's performance vastly surpasses NN's.

To note an important result, CNN and ViT have very similar performance values, apart from 5-fold CV where CNN usually returns better accuracy estimations. This fact was not expected, since it was assumed ViT would perform better than CNN. It can be argued that ViT might be oversized for such a small dataset, and that for small problems a CNN could be more appropriate since it has a far simpler architecture.

An interesting aspect reported from sweeps is that choosing the initial learning rate influences validation accuracy more than the dropout rate and the weight decay rate. This is noticeable in Figures 3, 6 and 10. With the NN and the ViT, low values for the initial learning rate tend to lead to better results, while it's the opposite with CNN. The same cannot be said for the other two hyperparameters: there's no specific range that is guaranteed to lead to a high validation accuracy. This could be justified by the fact that weight decay and dropout are both regularization strategies and they could be competing with each other.

# 6 References


[1] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[2] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[3] Keras. *BatchNormalization layer*. URL: `https://keras.io/api/layers/normalization_layers/batch_normalization/`.

[4] Keras. *LayerNormalization layer*. URL: `https://keras.io/api/layers/normalization_layers/layer_normalization/`.

[5] Keras. *SeparableConv2D layer*. URL: `https://keras.io/api/layers/convolution_layers/separable_convolution2d/`.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[7] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

## Declaration of originality

We declare that this material, which We now submit for assessment, is entirely our 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 our work. We 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 us or any other person for assessment on this or any other course of study.