Machine Learning

Assignment 2 - Image Classification

Albert Cerfeda

Contents

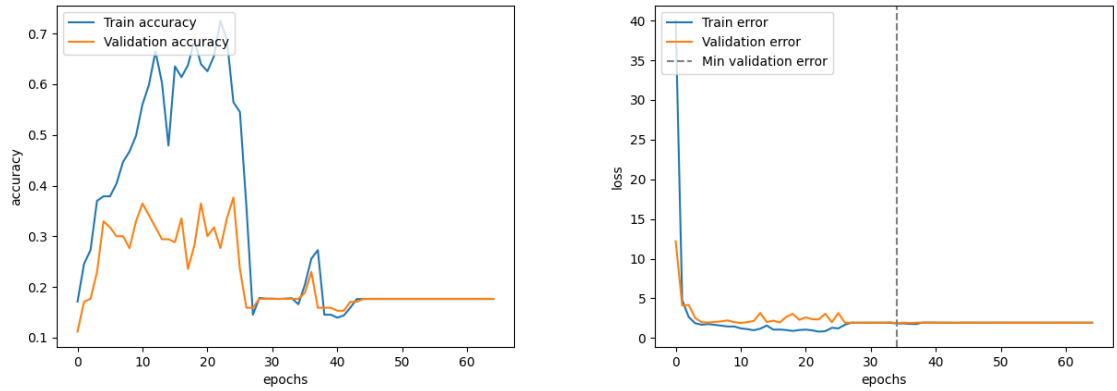
1 June 2023
Università della Svizzera italiana
Faculty of Informatics
Switzerland

1 Tasks

In this assignment we are tasked in building neural networks with the goal of correctly classifying images from a dataset containing images of 7 different species of felines. The dataset is composed of 1697 samples and each image has 223×224 resolution. The images are in color so they have 3 channels.

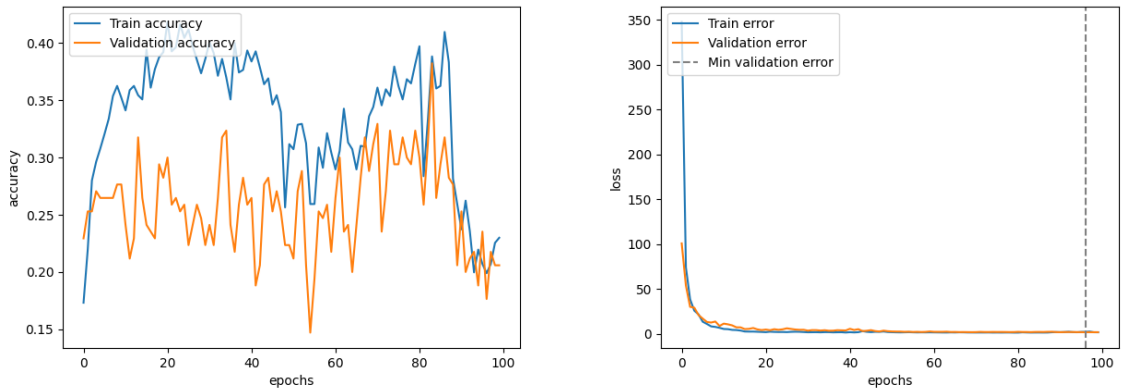
1.1 Task 1

In the first task we build a Fully Connected Feed Forward Neural Network (FFNN) for solving the classification problem.



(a) Training accuracy and validation accuracy (b) Error rate

Figure 1: Basic T1 model metrics during training



(a) Training accuracy and validation accuracy (b) Error rate

Figure 2: Bonus T1 model metrics during training

We split the available dataset as such: 80% train set, 10% test set and 10% test set.
We utilize the early stopping technique to avoid overfitting i.e we stop the training when the validation loss does not improve for a predefined amount of times.

Basic T1 model
Test Error: **1.86941** , Test Accuracy: **0.17647**
The basic T1 model respects the requirements of the exercise: no more than 3 layers, all layers are Dense, utilize the ReLu activation function for every neuron.

T1 model for the bonus task

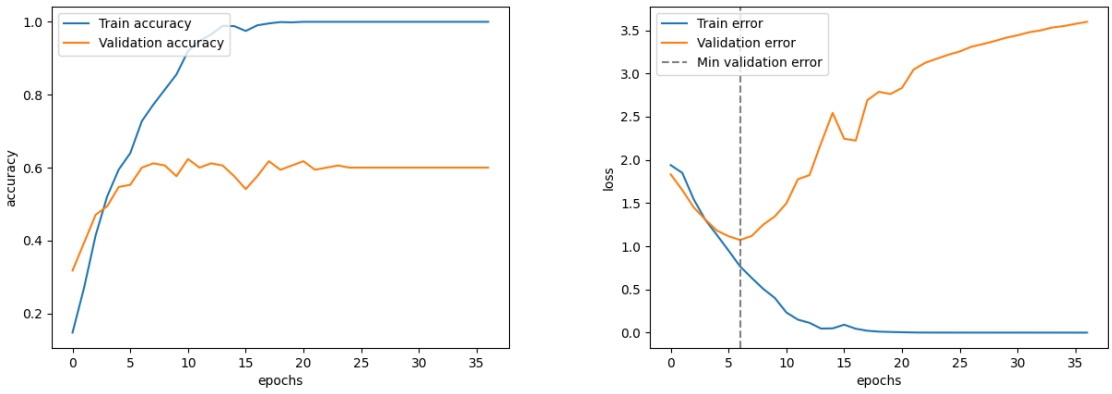
Test Error: **2.04282** , Test Accuracy: **0.20000**
The T1 model for the bonus task is the same as the basic T1 model but with an additional Dense layer.

From our results we can see that our feed-forward neural networks are performing quite badly. This highlights the fact FCNNs are not very good at multi-class image classification for a number of reasons. The most important thing they are lacking is awareness of spatial continuity. Since every pixel is a separate feature, CNNs struggle a lot when trying to recognize patterns in image data. That is because the relative distance and spacing of the pixels is not taken into account. As a demonstration if were to evaluate the performance on the *training set* we would get very good, biased results as expected, but were we to slightly so shift each image or perform any kind of manipulation on them, and we would see the error rise. Also in FCNNs since every layer is fully connected to the next, we have a very high number of parameters, rendering our model more computationally expensive and generally more prone to overfitting.

In our second model made for the bonus task, we do not have restrictions anymore on the input layer dimensionality of the FCNN. Instead of using the raw image pixel values as features, we have to perform some degree of feature extraction. We extract some features that can tell the neural network as much as possible on the nature of each image in the dataset, such as: mean and variance over the whole image, mean and variance for each channel, max and min values over the whole image, max and min values for each channel, ratios between statistics for each channel and image histograms.
Although we may are under the impression that we have richer, more descriptive data now, that is really not the case. By computing such metrics and not including the individual pixel values, we have more ambiguous data as every metric is computed by aggregating the image. In other words, images with the same mean or variance may not be the same image.
Our test results in fact show that our T1-BONUS model does not perform better. As a proof of what I stated above, I trained the same model on the pixel values *plus* the extracted features, and the test error decreased significantly to about **1.5**. A possible approach to further improve the feature extraction would be to compute the fourier transform of each image. That is because most felines in the dataset have distinguishing traits based on their striped fur. Such alternating pattern on their fur would be noticeable in the fourier transform and therefore identifiable by the FCNN.

1.2 Task 2

In this task we build a Convolutional Neural Network (CNN) for solving the classification problem. CNNs use convolutional layers which take into account the spatial structure of input images and learn local features, making them more effective at detecting patterns in images.

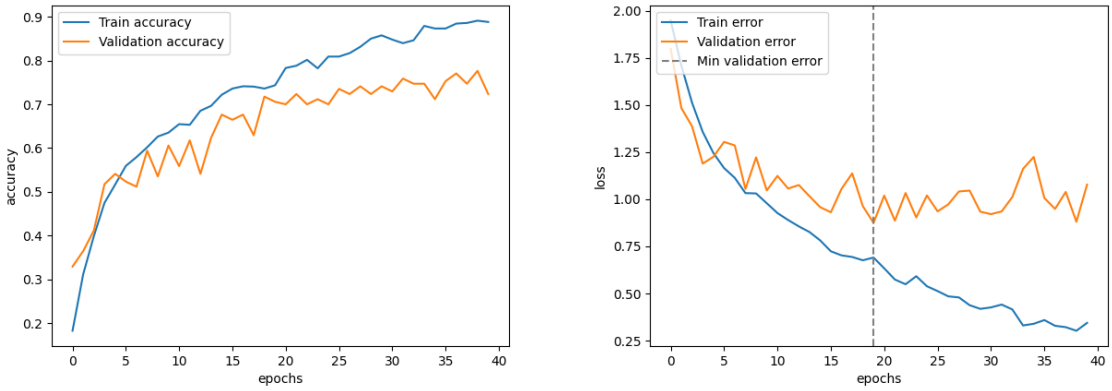


(a) Training accuracy and validation accuracy

(b) Error rate

Figure 3: The T2 model metrics during training

The models respect the requirements of the exercise: utilize 3 convolutional layers, 3 pooling layers and 3 dense layers (output layer included).



(a) Training accuracy and validation accuracy

(b) Error rate

Figure 4: *Augmented* T2 model metrics during training

Basic T2 model
Test Error: **0.93583** , Test Accuracy: **0.62353**

***Augmented* T2 model**
Test Error: **0.77835** , Test Accuracy: **0.74706**

Bonus T2 model
Test Error: **0.775862** , Test Accuracy: **0.741176**

When training with images, we have to take into account the variability of images. For example, our model has to be capable of classifying an image of a lion as such, independently from other image factors such as exposure, rotation, scale, saturation ecc. Even a slight change in the image can make it unrecognizable to our model. Figure 3 shows the training and validation accuracy and error rate for the basic T2 model during training. In order to make our model more stable, we perform *Data Augmentation*. That is, we augment our dataset by introducing many slight variations of our original image dataset. By expanding our dataset obtain a better model that yields better results. Notice how the model improves at a slower rate during training [Figure 4a]. It is possible that it is due to the fact that our dataset is more "diluted". Normally, the model would train on a much smaller and concise dataset with which the model learns quickly the main differences between the output classes. With the augmented dataset instead, the model will learn to thoroughly classify the data with better accuracy, but more extensive training is required for the model to train on every input class as the dataset is much larger now.

For the bonus task, we perform grid-search for finding the best hyperparameters to tune our model to. When building the model, I defined two main hyperparameters that affect the structure and behaviour of the model: the *learning rate* and the *network downscale*. I introduced the network downscale hyperparameter for trying to reduce nedlessely large model sizes as much as possible while retaining the best prediction accuracy. For example a network downscale of 2 will halve the neurons for each layer by 2. I tried using `GridSearchCV`, but deprecated modules plus other errors made me go for a simple for loop.

The grid parameters are the following: *learning rate* can be chosen between 0.001, 0.0001 and 0.00005, while the *network downscale* can be either 0.8, 1, 1.5, 2 or 3. Notice that a network downscale of 0.8 results in a network 20% bigger.
After running the grid search, we obtain the best configuration of hyperparameters, i.e

$$\text{Learning rate} : 0.0001, \text{Network downscale} : 2$$

By comparing the three models, we see how each of them is performing ever so slightly better, but not in a significative manner. The augmented model has better prediction capabilities than the basic model, but is extremely slow in training because of the augmented dataset. The Bonus model found through grid search is faster to the basic model, and also performs the best out of all the other models.

The models for trained for this assignment can be downloaded at the following link:
https://drive.google.com/drive/folders/1eA_jhffm-WtpzWtJAXAQplc8prVoZ708?usp=drive_link