Introduction to Deep Learning Project

Image classification

Narcís Font Massot, Pol Pastells Vilà

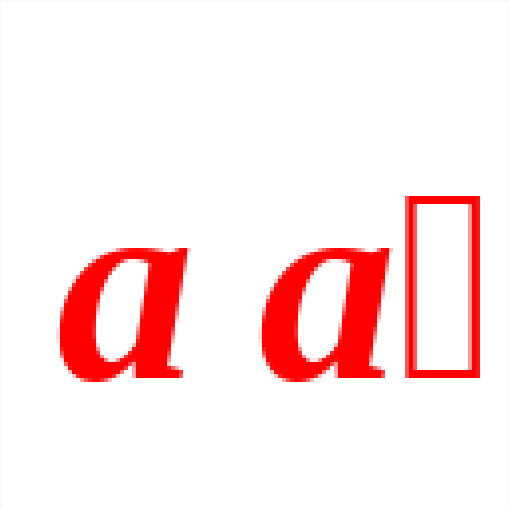
November, 2020

Modelling for Science and Engineering: Research & Innovation

Descriptive Analysis

The training dataset consisted of 2.000 images in 30 different categories, 5 *vowels* $\times$ (5 *accents* + *no accent*)

We notice that a lot of vowels which are supposed to have a bar on top don't actually have it. For example, for image 141:



Prepossessing of the images

We turned all the images into black and white, given that no extra information is given by the colors.

This way the input layer is reduced threefold.

Using the library cv2 this is easily achieved with the line:

cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

Model Description

Four convolutional layers are present. Each consisting of:

1. 32 (3×3) filters with stride 1 and padding to conserve the dimensions
2. Max pooling with a (2×2) kernel and stride 2
3. Dropout layer

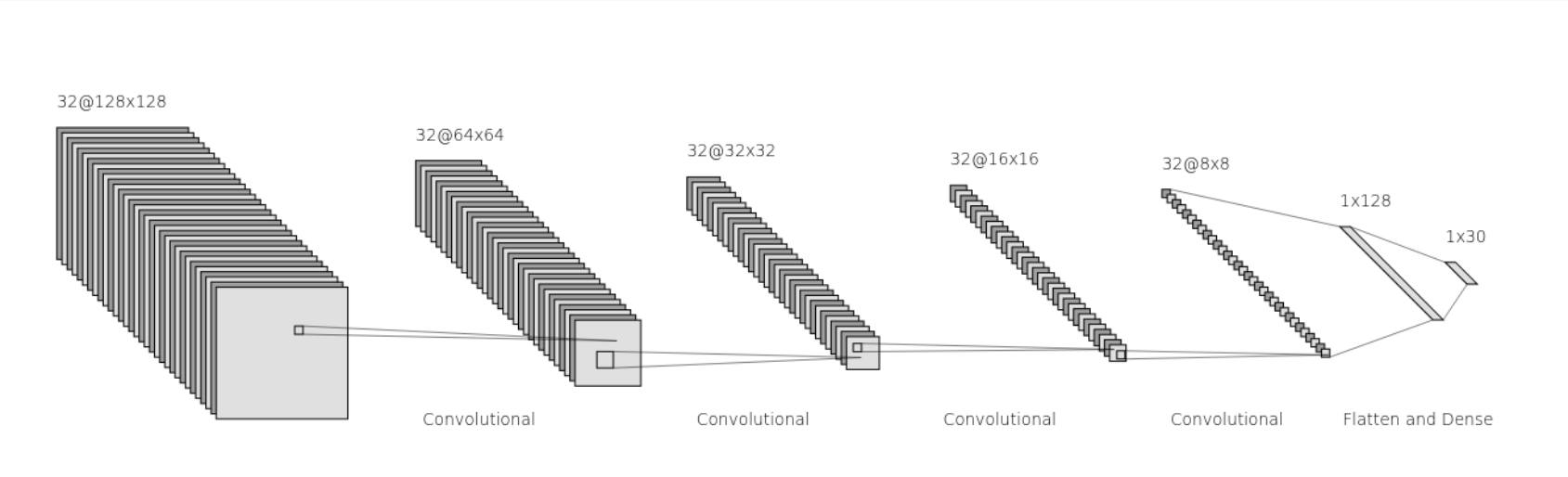
After these comes a fully connected layer of 128 nodes and the final layer with 30 neurons, one for each category.

We've chosen the dropout value to be 0.5 according to the Srivastava et al. 2014 paper's [1] findings.

[1] Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Model Description (2)

Here we show a scheme of our model:



Model Description (3)

All layers use a ReLu activation function, except the last one, using softmax.

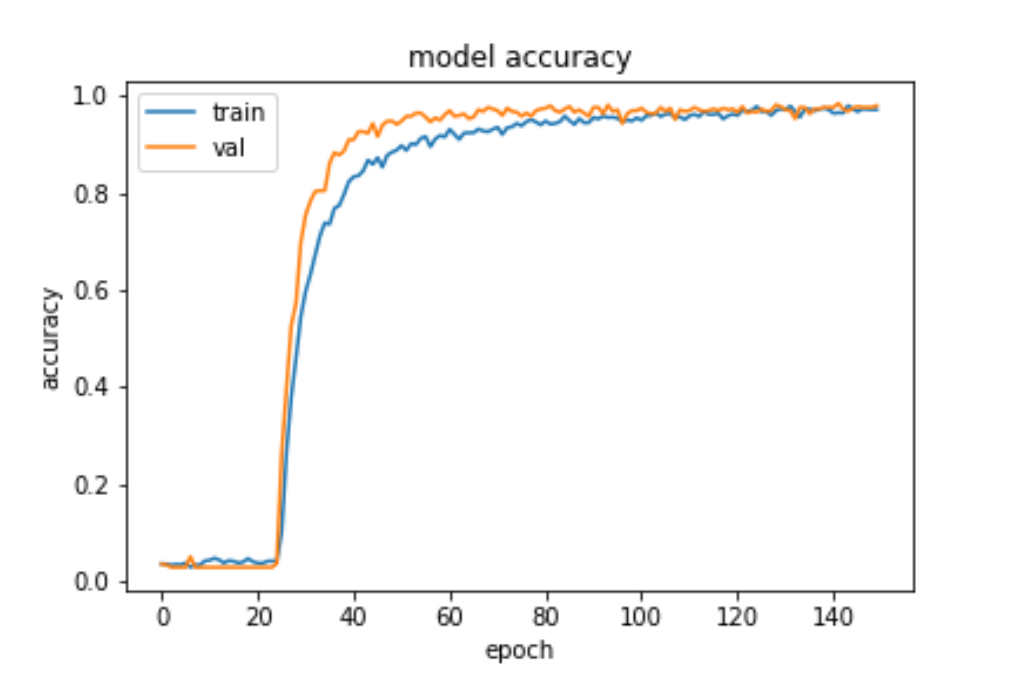
We train this model for 100 epochs with 70% of the trainset for training and 30% for validation.

On the right we show the summary displayed by Keras, with the shape and number of parameters on each layer.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 8, 8, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dense_1 (Dense)              (None, 30)                3870
=================================================================
Total params: 294,206
Trainable params: 294,206
Non-trainable params: 0
_________________________________________________________________
```

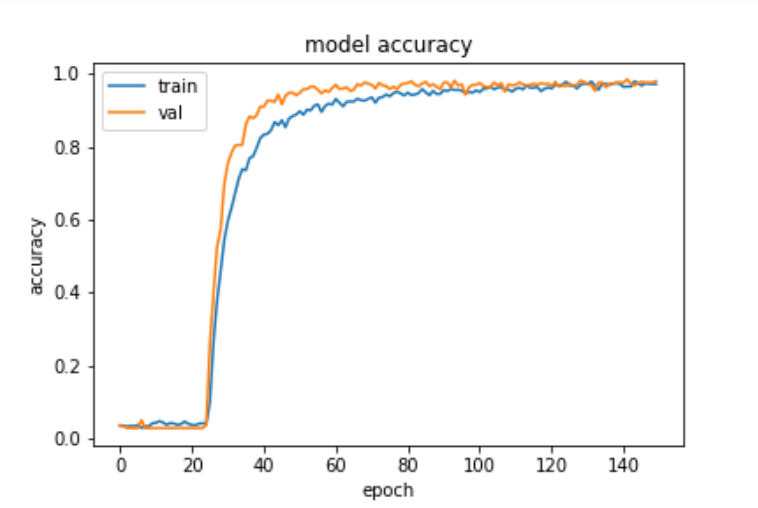
Accuracy

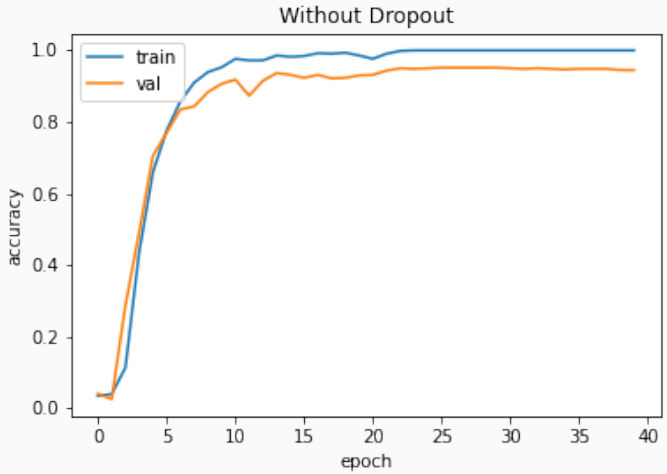
We see that during training it initially takes the Adam optimizer a while to start doing better than random $1/30 \simeq 3.3\%$. But after a while the accuracy for both training and validation gets close to 97%. To avoid overfitting, our final model is only trained during 100 epochs, given that at at that point only the training accuracy is getting better, but not the validation.



Dropout Experiment

We have tested the dropout effect on the CNN.





With dropout, the training and validation accuracy converge to the same value, without dropout this doesn't happen but the convergence is faster. Note that in this last case there is overfitting, but given the similarity of the images it is subtle.

Accuracy Estimation

Having similar training and test accuracy, we expect the accuracy for the testset to be on the same footing. Which gives an estimation of $3\% \times 5000 = 150$ classification errors.

Except that the accuracy given by Keras is not a good measure for how many images are correctly classified.

Testing the result with the entire dataset only 23 images were miss-classified. Therefore we conclude that the real classification error is closer to 1%, smaller than the accuracy error, and expect about 50 classification errors on the testset.

We claim that **our accuracy will be** 98.8%, given the grading strategy, and our confidence on the accuracy being between 97.8 and 99.8%.

Extra Comments

- To import the dataset into Keras, we used a simple bash script that classified all the images into a folder corresponding to its category.
- To be able to play with our model without depending on our (not so fast) personal computers, we've used a colab notebook with a GPU. This way, each epoch took less than one second during training.
- We believe that to improve the accuracy we would need a larger dataset, given that our results are already pretty high.