November 20th,2019

BOUAYED Aymene Mohammed

# AutoCV competition report

## Data pre-processing:

The datasets I have used are: MNIST, CIFAR-100 and PA100K. I have written my own script to retrieve and preprocess the data.

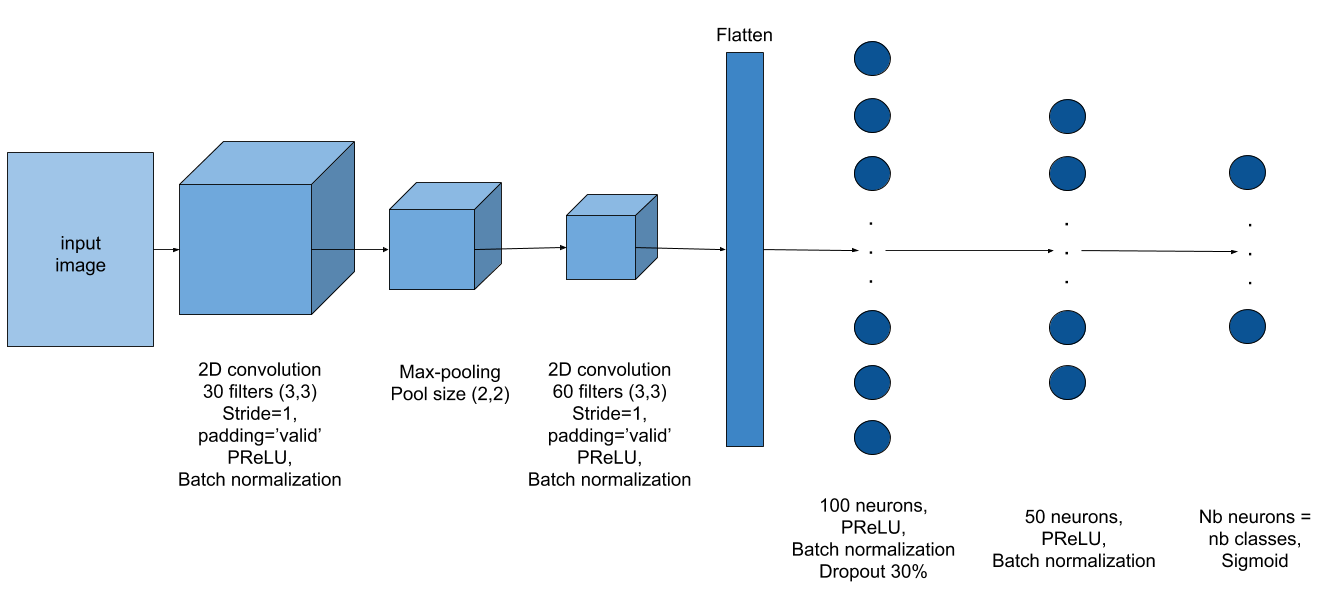
* Retrieving the data: After I have downloaded the 3 datasets, for the MNIST and CIFAR-100 the labels had to be changed from integers into a one hot encoded vector. As for the PA100K dataset, the images had to be put into a list because they didn’t have the same size and couldn’t be fit into an array.
* Pre-processing the data: The main problem was the fact that the images might not have the same dimensions.
  + To tackle this problem, I thought of many approaches like resizing the images to the biggest shape, finding the max and min shapes and resizing to the shape (max+min)/2…etc. The approach that I have settled on is **calculating the mean of the shapes of the images** (summing all the shapes and dividing by the number of images) **and resizing them to this mean.**  This allows to find a weighted middle ground of the shapes. By doing so we **avoid any big transformations or distortions in the shapes on the objects in the picture**.
  + Now after resizing the images, all of them have the same shape and can be put into an array after they have been in a list (because initially they didn’t have the same shape).
  + Also before the training of the CNN, the values of the arrays containing **the images have been normalized to be in the range [0,1] and this by dividing the values by 255.**

## Model architecture:

Since the goal of the competition is to create a model that trains on different datasets (with varying shapes) to handle different kinds of classification tasks (one or multi-class classification tasks), I thought of creating a dynamic CNN with an architecture that changes according to the shape of the input images.

**The idea is that I have a base CNN and I add more layers to this base model according to the shape of the input.**

### The base model I used is summarized as follow:



### Additions to the used base model:

* If the input images’ shape is greater than 400 on the x or y axis, than this image contains a lot of information. Thus, more convolutions can and should be added to extract every piece of intel available. To do so I added a convolutional and a max pooling layer. The former contains 15 filters each with a shape of (7,7), a stride of 1, a valid padding and the PReLU activation function. The latter has a pool size of (3,3).
* If the input images’ shape is greater than 100 and lower than 400 on the x or y axis than I do the same thing as with the images with a shape greater than 400. Except that the shape of the filter is (5,5).
* If the input images are RGB. The information they contain is multiplied by 3 which means there is more information to extract. That is why I added a max-pooling and convolutional layer at the end of the convolutional part of the CNN. The pooling layer has a small pool size of (2,2). And the convolutional layer has 90 filters each with a shape of (3,3), a stride of 1 and a valid padding. The activation function that I used is the PReLU. And I also used batch normalization.
* Now with regards to the dense part, if the input images’ shape is greater than 100 on the x or y axis than the information that arrives to this dense part is big. So to give the neural network more power to understand and process the information it receives, I added a dense layer at the beginning of the dense part of the CNN. This layer contains 200 neurons, a PReLU activation function, batch normalization and a dropout with a rate of 50% to avoid over-fitting.

NB: the architecture of the model needs more fine tuning to achieve greater performance.

## The used training parameters:

* The loss function that has been used for the training is the categorical cross entropy. As for the optimizer, the optimizer of choice was Adam.
* The number of training epochs was initially set to 100 epochs. But using the validation data (20% of the training data and they change from epoch to epoch) the fitting of the model is stopped if the validation loss is greater than the training loss by 0.05.

## Areas to improve on:

* Instead of using a CNN one can use a Capsule network.
* Make a better approach of a dynamic model using percentages.
* Use boosting (like Adaboost) or bagging to increase the accuracy.