Machine Learning:

Convolutional Neural Network for hand sign recognition

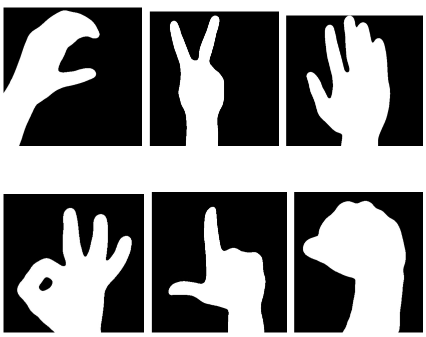
José Ramón Romero Chávez (A01700318)

Tecnológico de Monterrey, Campus Querétaro A01700318@itesm.mx

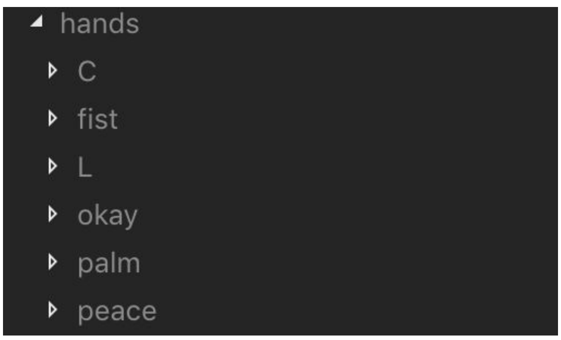
Introduction

The sign language is a powerful tool as it can be interpreted in a universal way, there a lot of applications from helping disabled people to command in the battlefield

Dataset



Our **data set** consist of around 3000 images **labeled** with its corresponding form (C, Peace, Palm, OK, L, Fist) and stored using the following structure



Model Proposal

Using a generic **neuronal network** is not the best of the option when we are leading with images and patterns inside them, an alternative for this will be a **convolutional neural network.**

The key concept behind **Convolutional Neural Networks (CNNs)** is the application of **kernels or filters** and which operate over raw pixel intensities.

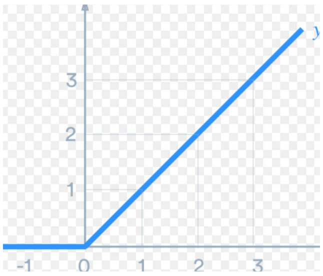
Convolution layers are stacked on top of each other deeper in the network architecture prior to applying a destructive pooling operation

Structure

- Input:
    - The input layer is based on the size of the images (width, height)

- Layers
    - Fully connected layers are used for this implementation, in Keras, they are known as **Dense layers.**
    - Rectified Linear Unit (ReLU), activation function in this network architecture.

$$f(x) = \max(0, x)$$



ReLU is linear (identity) for all positive values, and zero for all negative values. It simplifies the complexity of the calculus inside of the neuronal network

- **A batch size** (30) and its normalization defines the number of samples that will be propagated through the network, in order to learn and pass to the next layer

- **Pooling layers** have a primary function of progressively reducing the size of the input as it goes deeper through the network, (Dimensionality reduction) thye are located between each of the **convolutional layers**

- **Dropout Optimization** is also implemented inside the network, the process consist of disconnecting random variable in order to simplify the model and **evade the probability of an overfitted model** (memorized, useless for generalization). **(In this case 25% of the neurons are randomly dropped=**

- Output:
  - The softmax as activation function in the las layers allow us to layer returns the class probabilities for each label, and classify them according to the one with the highest value

Development

In order to do the proper improvement, the data set was splitted on a **training set,** to build the model, **and testing set, t**o improve and compare, **(80% + 20%)** in a random way, by using also the scikit-learn toolkit.

As the problem handles with different kinds of categories, as a measurement of loss it was used categorical cross entropy, and as optimizer method to reduce it , it was used Stochastic Gradient Descent (SGD), which compared to the common gradient descent it choosest the samples randomly.

Data Augmentation

In order, to improve the result, the data set was increased using keras' ImageDataGenerator, it also will help the to reduce the probability of a model to over and generalize in a better form.

Image augmentation allows us to construct "additional" training data from our existing training data by randomly rotating, shifting, shearing, zooming, and flipping.

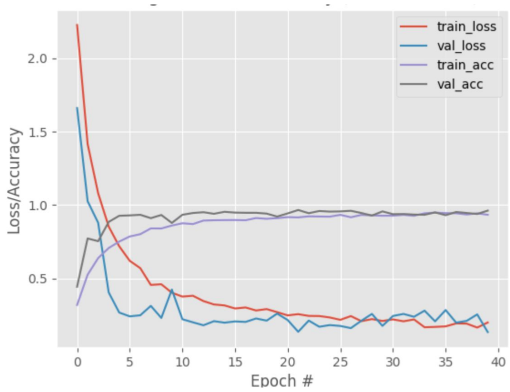
Training

The training process is based on the hyperparameters: **learning rate** (controls the adjustment the weights of our network with respect the loss function), epochs (times of general improvement in the model), batch size and loss generated.

Finally we call model.fit_generator (instead of model.fit), in order to add the data generated as parameter to the model

Finally, we'll evaluate our model, plot the loss/accuracy (the calculated value vs the correct label) curves, and save the model.

As can be seen on the plot, the error starts to decrease from the very first iterations.



Validation Test

The validation test is generated by images captured by camera, the images have been processed using OpenCV, in order to define the contours of the shape formed by the hand using Gaussian Blur, Background masking and Thresholding binarization filters, in order to match the training and test sets

The results seems to be very similar to the ones predicted by the model

For future and improvement for this implementations:

- Add more signs to the data set
- Reduce the area of interest for the frame recovered from the camera
- Improve the background and binarization thresholding

Referencias

A Beginner's Guide to Convolutional Neural Networks (CNNs). (2019). Retrieved from https://skymind.ai/wiki/convolutional-network

A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. (2019). Retrieved from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CS231n Convolutional Neural Networks for Visual Recognition. (2019). Retrieved from http://cs231n.github.io/convolutional-networks/

OpenCV-Python Tutorials — OpenCV 3.0.0-dev documentation. (2019). Retrieved from https://docs.opencv.org/3.0-alpha/doc/py_tutorials/py_tutorials.html

Rosebrock, A. (2019). Keras Tutorial: How to get started with Keras, Deep Learning, and Python - PyImageSearch. Retrieved from https://www.pyimagesearch.com/2018/09/10/keras-tutorial-how-to-get-started-with-keras-deep-learning-and-python/#smallvggnet