Clasificación de símbolos musicales

Este documento servirá para anotar las ideas, etapas y pruebas que se realizan en el desafío de la asignatura **Desafíos de Programación** de forma cronológica

Servirá para que el profesor realice un seguimiento del trabajo durante el curso y se valorará en la calificación final.

-------------------------------

Bajamos a 15 épocas para realizar las pruebas de una forma más rápida.

Test score:1.15 accuracy: 63.10%

* Cambiamos MaxPooling2D por AveragePooling2D. - No funciona.
* Cambiamos el optimizador de Adadelta a Nadam. Sube a 75.38%
* Cambiado batch\_size a 16. Sube a 78.30%
* Padding=’same’. Mejora hasta 82.29%
* Realizado el aumentado de datos, con 45º de inclinación. Nota: Igual 45º es mucho, Juanra ha sugerido bajar a un rango de 10º-20º.
* Leido que la activación ReLu ha demostrado mejores resultados últimamente. <https://riunet.upv.es/bitstream/handle/10251/64848/TrabajoFinalMasteJavierJorgeCano.pdf?sequence=1>
* Optimizadores (estudio): <http://ruder.io/optimizing-gradient-descent/index.html>
* Best Practiques Keras: <https://github.com/fchollet/keras/issues/898>
* <https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f>
* <https://www.quora.com/Intuitively-how-does-batch-size-impact-a-convolutional-network-training>
* <https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>

Usando un optimizador de primer orden, porque son más rápidos (en teoría). Usaremos SGD con momentum, funciona bastante bien.

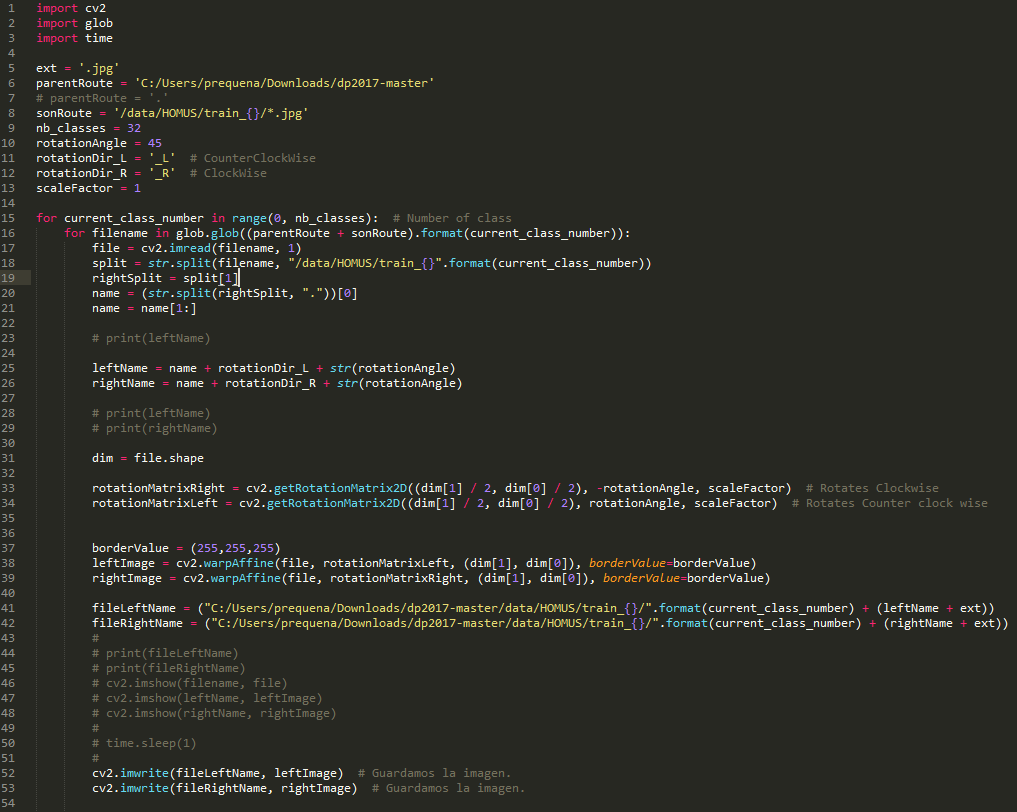
<https://www.analyticsvidhya.com/blog/2016/10/tutorial-optimizing-neural-networks-using-keras-with-image-recognition-case-study/>

<https://wiseodd.github.io/techblog/2016/06/22/nn-optimization/>

<https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>

<https://becominghuman.ai/understanding-and-coding-inception-module-in-keras-eb56e9056b4b>

<https://www.researchgate.net/publication/285164623_An_Introduction_to_Convolutional_Neural_Networks>



***Código correspondiente al aumentado de datos con OpenCV.***

Pruebas de como quedan las imágenes

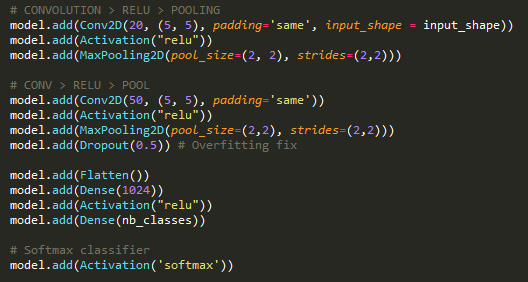
Normal :  L20:  R20:  Para la primera imágen.

Cómo dijo Juanra, se ha bajado el ángulo de rotación a 20º, donde L corresponde al sentido contrario a las agujas del reloj, y R corresponde al sentido de las agujas del reloj.

Con aumentado de datos, se queda en un total de 36969 train samples y 4071 test samples.

El aumentado de datos funciona mejor, se ha comprobado con varias pruebas.

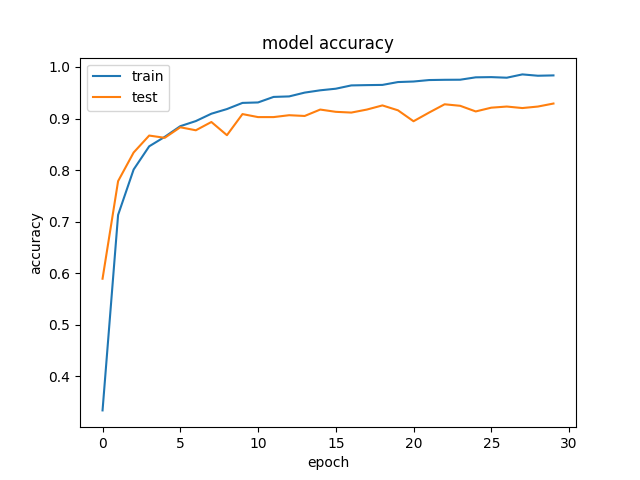
**Red con la siguiente estructura**

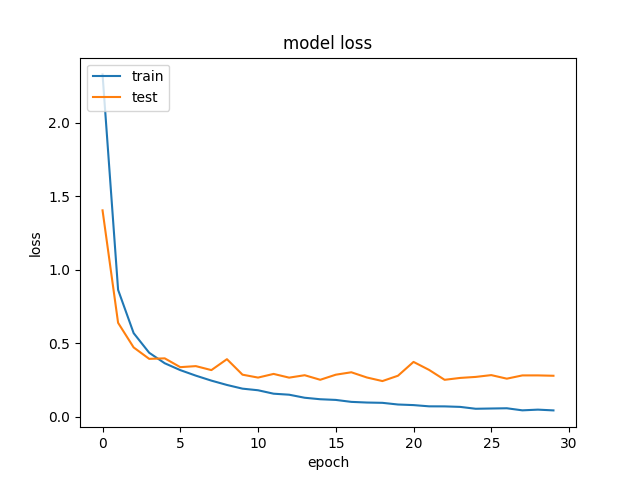


89% Accuracy

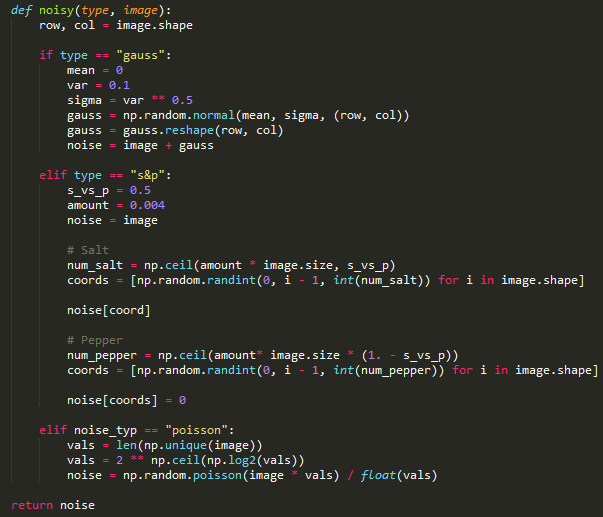
**Se utiliza matplotlib para imprimir los resultados**

Nos guardamos el model accuracy y el model loss, por cada época, y se pueden guardar las imágenes.





**Probado el ruido**



Realizadas pruebas con la red empeorada (10x10, 10 épocas) con y sin ruido.

Normal:  Gauss:  Poisson:  Salt and Pepper: 

|  |  |
| --- | --- |
| Sin Ruido | 93.22% |
| * Gauss | 93.23% |
| * Poisson | 93.20% |
| * Salt and Pepper | 93% |

Comprobado con la red normal (homus\_LeNet5.py), a bajas épocas. ¡Descartado!

Probamos modelo:

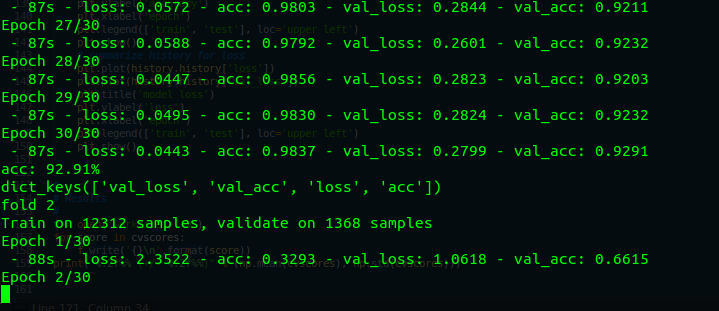
* # CONV > RELU > POOL
* model.add(Conv2D(20, (5, 5), padding='same', input\_shape = input\_shape))
* model.add(Activation("relu"))
* model.add(MaxPooling2D(pool\_size=(2, 2), strides=(2,2)))
* # CONV > RELU > POOL
* model.add(Conv2D(50, (5, 5), padding='same'))
* model.add(Activation("relu"))
* model.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2)))
* model.add(Dropout(0.5)) # Overfitting fix
* model.add(Flatten())
* model.add(Dense(500))
* model.add(Activation("relu"))
* model.add(Dense(nb\_classes))

91.12% - homus\_cnn.h5 (primera guardada)

* Cambiamos padding por border\_mode (https://github.com/fchollet/keras/issues/1984):

93.22% - homus\_cnn2.h5

Crossvalidation kfold 10:





**Análisis estadístico**

He lanzado el test de Wilcoxon con las dos pruebas de CrossValidation que tengo

*x <- c(94.95614035087719, 95.00487329434698, 94.76120857699804, 94.49317738791423, 95.07797270955166, 94.83430799220272, 94.56627680311891, 94.49317738791423, 94.6150097465887, 94.49317738791423) # CV + datos*

*y <- c(91.81286549707602, 93.12865497076024, 92.9093567251462, 87.79239769567523, 92.17836257309942, 92.25146195344757, 92.03216374269006, 92.17836257309942, 92.47076023391813, 91.59356725146199)*

*# Queremos comprobar si los resutados de 'x' son mejores que los de 'y' sabiendo que las pruebas*

*# se comparan por pares (paired=TRUE)*

*wilcox.test(x,y, paired=TRUE, alternative="greater")*

Determinamos, que pValue = 0.002945, lo cual nos indica que el vector X es mejor que el Y, lo cual nos dice que la prueba de CrossValidation con el aumentado de datos funciona mejor.