NANI Ripple detection algorithm

Preprocesado:

1. Downsamplea los datos a una frecuencia de sampleo 1250 Hz (ver `mov_av_downsample` en `utils.py`)
2. Normalizar: calcula el z-score de cada canal (ver `zscore_signal` en `utils.py`)
3. Enventanado con overlapping (0.7) (ver `adapt_input_to_CNN` en `utils.py`)
4. Codificación de los eventos: para cada ventana de lfp calcula el porcentaje de puntos que forman parte de un ripple (ventanas de 50 ms) (ver `adapt_label_to_CNN` en `utils.py`)

CNN

Red convolucional que recibe un fragmento de lfp de 50 ms y 8 canales y predice el porcentaje de la ventana que contiene un ripple.

Para el diseño de la arquitectura de esta red nos hemos basado en YOLOR y UNet. Las capas y parámetros de la red están resumidos en la siguiente tabla (ver `model_builder_prob` en `model_builders.py`):

Capa	**Filtros**	**kernel**	**activación**	**units**	**output_shape**
Conv2	32	(2,2)	relu		(62,8,32)
BatchNormalization					(62,8,32)
ReLU					(62,8,32)
MaxPooling2D		(2,2)			(31,4,32)
Conv2	16	(2,2)	relu		(31,4,16)
BatchNormalization					(31,4,16)
ReLU					(31,4,16)
MaxPooling2D		(2,2)			(15,2,16)
Conv2	8	(3,2)	relu		(15,2,8)
BatchNormalization					(15,2,8)
ReLU					(15,2,8)
MaxPooling2D		(2,2)			(7,1,8)

Conv2	16	(4,1)	relu		(7,1,16)
BatchNormalization					(7,1,,16)
leakyReLU					(7,1,,16)
Conv2	16	(6,1)	relu		(7,1,16)
BatchNormalization					(7,1,,16)
ReLU					(7,1,,16)
Conv2	8	(8,1)			(7,1,8)
BatchNormalization					(7,1,8)
ReLU					(7,1,8)
Flatten					(56)
Dense			sigmoid		1

Detección de ripples:

Por último, para inferir los tiempos de los ripples a partir del output de la red, vamos detectando de manera secuencial aquellas ventanas con un valor mayor o igual a 0.5 (la red predice que al menos un 50% de los puntos en esa ventana pertenecen a un ripple). En los casos en que dos ventanas adyacentes contienen puntos de ripples, inferimos el inicio y final a partir del porcentaje de puntos en cada ventana que son del ripple (ver `get_ripple_times_from_CNN_output` en `utils.py`)