**ChannelGate: A Gating Network to Route Ion-Channel Data towards Optimal Deep-Learning Models**

**PI Prof Richard Barrett-Jolley, Musculoskeletal Biology, University of Liverpool, RBJ@liverpool.ac.uk**

*Context:* Over the past several years, RBJs group have developed an electrophysiological analysis software tool called Deep-Channel. Deep-Channel makes sense of complex electrical signals from proteins by using artificial intelligence, we were the first to develop and publish this entire approach (https://doi.org/10.1038/s42003-019-0729-3). Traditional methods are slow and manual, but Deep-Channel rapidly and accurately detects activity in noisy data. This speeds up this aspect of research into diseases and drug development 100s of times, removing human bottlenecks and enabling breakthroughs in molecular biology.

Since its initial release however, a challenge has emerged. The models do not generalise as well as hoped across different families of proteins and recording set-ups beyond that of the original training data. This is a common (if not universal) problem with deep-learning models in biology. Our present solution is to ask the user to choose from several pre-trained models to find the one that matches the data best. We have conceived a far more sophisticated approach, but are yet to code, pilot or implement it. The approach is to bolt on an initial (trained) "ChannelGate" model that analyses the data, e.g., using a convolutional neural network (CNN), and classifies the input into one of several types. This directs the pre-processor to select the ideal model from the available set, or even exit nicely with a "none of these" response – which would allow the user to contact us about training additional models.

The core of this project "ChannelGate" is to develop such a pre-processing AI model, and implement it first in python and then in our existing web-app proving ground. The longer-term plan would then be to embed ChannelGate into our full web-app and allow an even wider range of users to analyse their data in an ever more automated manner.

*HPC: To train each model takes about 1 week with a single desktop/GPU. To fully train several competitor models on a PC is impractical. Our workflow will be basic model checking on PC final training on – HPC.*

WP 1: Data curation and problem specification. The first week will be induction discussion, defining goals, and workstation set-up etc, but after this we expect the curation of a starting dataset to take less than a day.

WP 2: python implementation of a simple model. To initially code our existing "default" (a CNN classifier) it should be easy and we would expect metrics and images within the day, given "first few epochs".

WP 3 embed our new pilot into an existing python package. Compare outputs to benchmarks. Then progress to longer runs, further datasets, and more complex models.

WP4 Load and run these models on the HPC to allow us to train with more parameter variants, and for longer. Selection of "ideal model/s".

WP5 Embed in our deep-channel web-app using javascript.tf. The model needs to be converted from a tensorflow HDF to a json structure and we will now have a completed pilot for longer term evaluation by the wider team.