**Minutes**

24/11/22

* Looked through Google Colab - looked at current model I had
* Noticed the way my current padding meant that I had redundant 0’s around the value extracted by the kernel. I didn’t need to pad my columns, as the result after the first convolution will be a 1D image.
* Examples used was a 21x100 protein sequence for explanation purposes.
* Discussed conv2D output:
  + Conv2D(1, 3, 21, padding=10) over the 21x100 sequence gave 3 1x100 sequences – the 3 channels.
  + Now when is passed to the next layer, the kernel will cover all of the channels at once, because the kernel is of size 1x21x3:
  + (kernel seen in red going across all channels (RGB for example)
  + Each layer has unique weights, so here we have [1x21x3 (+1 for the bias)] 64 weights at each kernel placement (for each amino acid).
* Discussed PyTorch convolution input:
  + Conv2D – tensor[1,1,20,100] - [batch, channels, height, width]
  + Conv1D – tensor[1,1,100] - [batch, channels, width]
  + Use of squeeze/unsqueeze to change dimensions
* With conv2D we would need to use an asymmetric (1,21) kernel for our middle convolution layers as our tensor is of 1xLENGTH shape
* However,
  + As the sequence is 1D, we are representing each amino acid as a 20-Dimensional object, making it seem we have a 20x100 sequence image.
  + We could represent each amino acid as a separate channel in the input. The sequence will be 1D, and we will have a 1x100 sequence image, this time with 20 channels.
  + This lets our window size vary – could be 15, could be 21- arbitrary.
    - Not constrained to having window size to suit specific image shape.
    - Better – there are only 20 amino acids and the 21st was introduced to count as a buffer (like a 0 padding) - would only really be used if attaching sequences together and using this 21st amino acid as a ‘spacer’ - for efficient use of GPU.
  + So we can use a 1D convolution now – conv1D(1, 20, 100) - 20 channels
  + This behaviour is the same as the 2D kernel, but a bit more flexible. Less complex solution.
* Would be a good idea to discuss the development of this change in the dissertation. Going from previous 2D method to new approach now.
* 1D representation with 20 channels:
  + So if the 0 index of data is the first sequence, then data[0,:] returns our vectorised sequence.
  + Then data[0, 0] - will be a tensor the length of the sequence, with 1 if the amino acid at this position is ‘A’ and 0 otherwise. Same for the other channels, but for their amino acid.
* Discussed large models:
  + Treatment of multiple convolutions and activation functions as ‘one convolution’.
    - They treat these as separate modules.
  + These are complex high performing nets.
  + They utilise skip layers – allowing these very deep networks.
  + ResNET.
  + Quite state of the art.
  + Could be nice to also replicate idea – skip connections – residual neural networks.

Goals:

* Post current notebook to GitHub. Despite refactoring the implementation, this will be good to look back at and reflect on.
* Change amino acid map.
* Refactor dataset class and network to use the shape of this data.