<!-- APPUNTI -->

-Lo scopo del gioco è quello di capire se il DNA che ti vene mostrato è di origine VIRALE (A) oppure NO (B).
-Per identificare potenziali genomi virali nella specie umana si utilizza l'NCBI BLAST, il quale confronta una
        sequenza di genomi conosciuti da un database pubblico e stima quanta similarità condividono. -> quanto
sono
        simili tra di loro? hanno una certa sequenza in comune?
                Dicono sia poco efficace...
-I convulational layer possono essere usati per il pattern matching e non solo per le immagini.
-L'utilizzo dell'average pool aiuta a capire in modo più generale il contesto rispetto che al solo max pool.
-K-MER è pattern frequency.
-Non hanno utilizzato l'addestramento casuale dei pesi per evitare un overfitting troppo alto.
-Hanno trovato i pesi e quindi addestrato la CNN partendo da un modello di partenza e un po' alla volta hanno
aggiustato dove serviva i pesi.
-Utilizzano due processi in parallelo:
        -average: permette di capire la frequenza.
        -max: permette di avere una miglior precisione -> scrematura?
-Addestrameto ad epoche e con Adam con batch size di 128.
-Utilizzo del DROP OUT -> che cos'è?

<!-- IDEE -->

-I dati (secondo Excel) sono una serie di basi azotate che definiscono una sequenza di DNA -> pattern matching?
Inoltre
        i dati sono classificati nella classe 0 (?) e nella classe 1 (?).
-Perché non ragionare in termini probabilistici e mettere un thrashold? -> con che probabilità è un A o un B?
-Si prende una sottostringa delle basi azotate e si analizza quella -> Quanto dev'essere lunga?
-Utilizzo di una RNN (recursive neural network) (C) invece che una CNN (convolutional neural network) (D). Le (D)
        valutano meglio immagini dove la distribuzione spaziale dei dati è più importante. Nel nostro caso
        sarebbe più interessante vedere se un gruppo di basi azotate di (A) sia all'interno di una stringa di
DNA. In
        tal caso una rete (C) potrebbe essere più interessante in quanto da più importanza all'ordine.
-Utilizzare uno strato LSTM (Long-Short Term Memory) (E) o GRU (Gated Recurrent Unit) (F) per gestire le sequenze

        di caratteri. Nello specifico sono:
                -(E): permettono di apprendere da sequenze di dati più lunghe rispetto alle RNN tradizionali.
Esse
                        contengono uno strato interno chiamato CELLA che può memorizzare informazioni a lungo

termine. Utilizza PORTE per gestire i flussi input/output/dimenticanza. Le (E) permettono
di decidere quali informazioni tenere e quali dimenticare.
-(F): sono come le (E) ma hanno una struttura più semplice. Utilizzano due PORTE, una di ripristino
e una di aggiornamento. La CELLA (F) utilizza uno strato nascosto per trasferire le informazioni. Hanno prestazioni simili alle (E).

-Possibile struttura:
-Input: Sequenza di caratteri (ad esempio, lunghezza massima della stringa).
-Manipolazione dell'imput: come trasforamere l'imput alfabetico in un input utilizzabile dalla rete.
-Strato RNN (LSTM o GRU): Cattura le dipendenze sequenziali -> core della rete.
-Strato Fully Connected (Dense): Effettua la classificazione binaria -> necessario per la classificazione.
Si può mettere una sigmoid function per avere un output in termini probabilistici [0,1] + thrashold.

<!-- LINK -->

-LSTM with PyTorch: https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html
-GRU with PyTorch: https://pytorch.org/docs/stable/generated/torch.nn.GRU.html