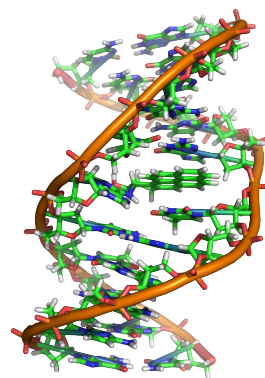


LSTM for gene fusions classification - #3

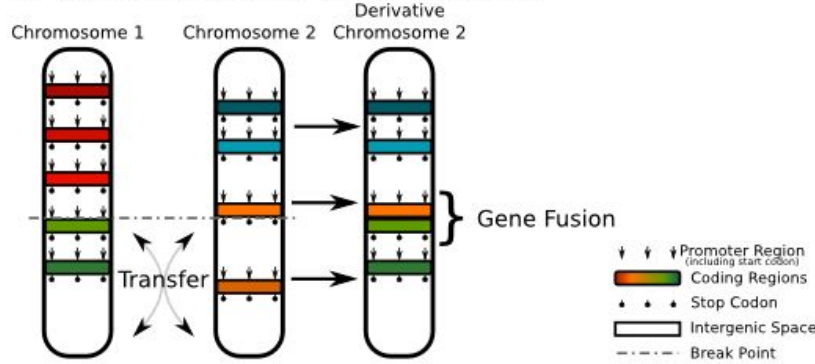
Bioinformatics Project, A.Y. 2018/2019



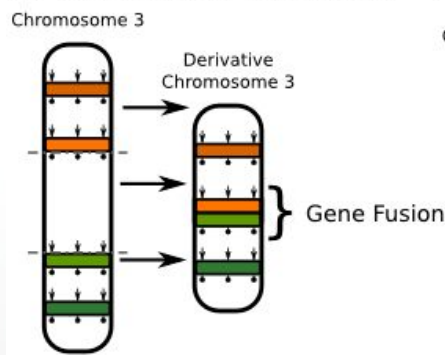
Stefano Brilli, s249914@studenti.polito.it
Michele D'Amico, s246501@studenti.polito.it
Francesco Chiarlo, s253666@studenti.polito.it

Gene fusions

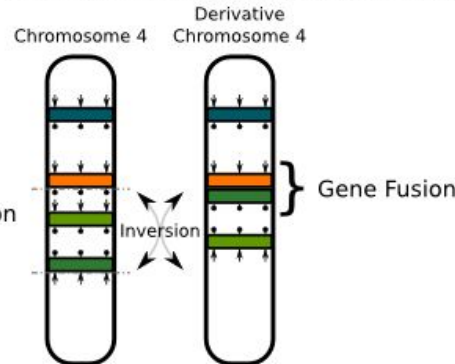
A. Chromosomal Translocation



B. Interstitial Deletion



C. Chromosomal Inversion



Gene fusions are the result of the union of **two different DNA regions**, either in a laboratory or naturally.

Under some conditions, this could result in **cancer development**.

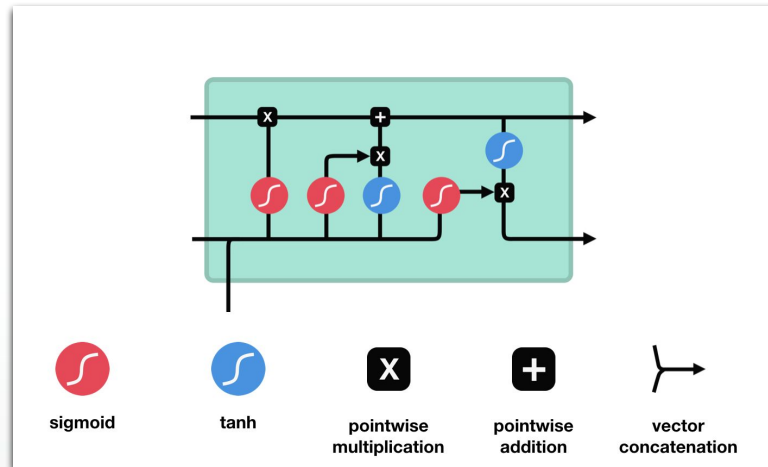
Two common scenarios are:

- a fusion that leads to a new protein with a different function from the original two genes (breakpoint in coding region).
- a fusion in which a strong promoter is associated with an oncogenic function providing a deregulation of oncogene protein production (breakpoint in a non coding region)

Long Short Term Memory

LSTM networks:

- solve the vanishing gradient problem that affects the traditional RNNs by introducing a self loop internal recurrence
- provide a system of gating units that controls the signal and gradient flow
- are capable of learning long term dependencies in sequences

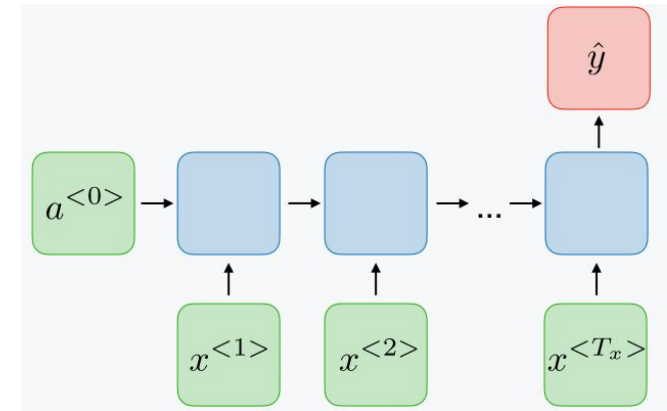


The task

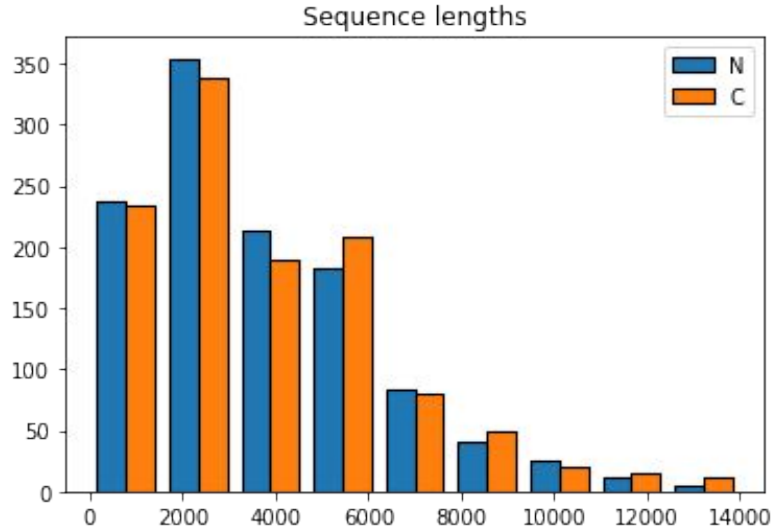
The aim of this task is the classification of fused genes according to their own sequences. A “many-to-one” RNN architecture is suited for such a task.

Proposed architectures:

- on DNA sequences
 - LSTM
 - bidirectional LSTM with convolution
- on translated proteins
 - LSTM
 - bidirectional LSTM with attention
 - bidirectional LSTM with convolution

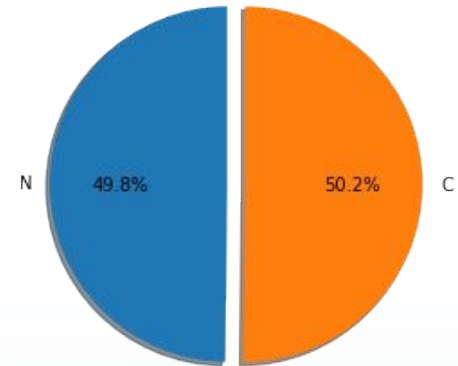


Class Distribution



```
bin_dfs.Sequences.duplicated().value_counts()  
False    2229  
True       79  
Name: Sequences, dtype: int64
```

Sequences are quite equally distributed in dataset and both classes have almost the same length distribution.

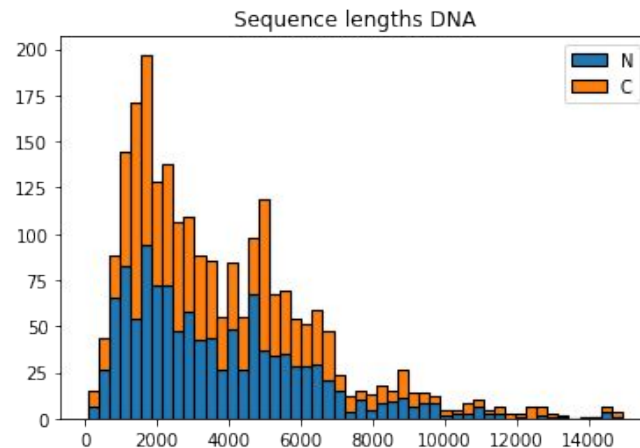


Dealing with long DNA sequences

Some DNA sequences are extremely long and can make learning difficult.

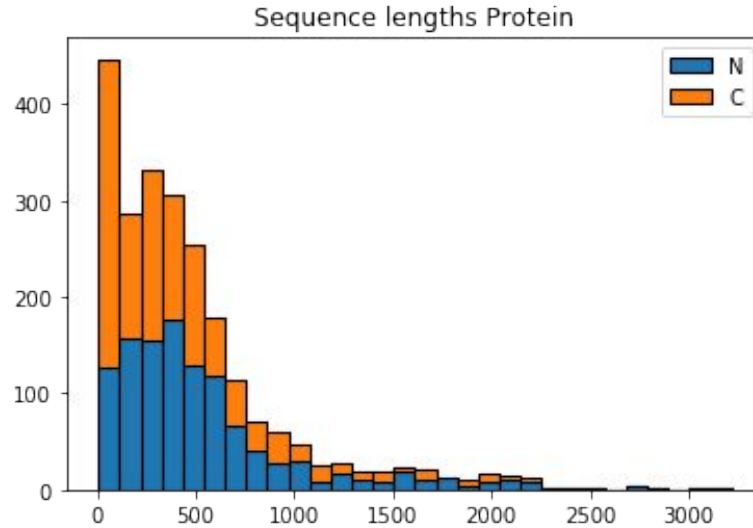
Different approaches were tried in order to tackle such problem:

- using coarser sequence representation as a result of convolution and max pooling
- sequence truncation despite the breakpoint inclusion uncertainty
- DNA translation into amino acids



```
count      2308.000000
mean       3770.457106
std        2628.655730
min         56.000000
25%        1690.750000
50%        3079.000000
75%        5158.250000
max        14945.000000
Name: Sequences, dtype: float64
```

Protein data



```
count    2308.000000
mean     471.215771
std      463.889156
min       7.000000
25%     159.000000
50%     354.000000
75%     593.250000
max     3219.000000
Name: Protein_length, dtype: float64
```

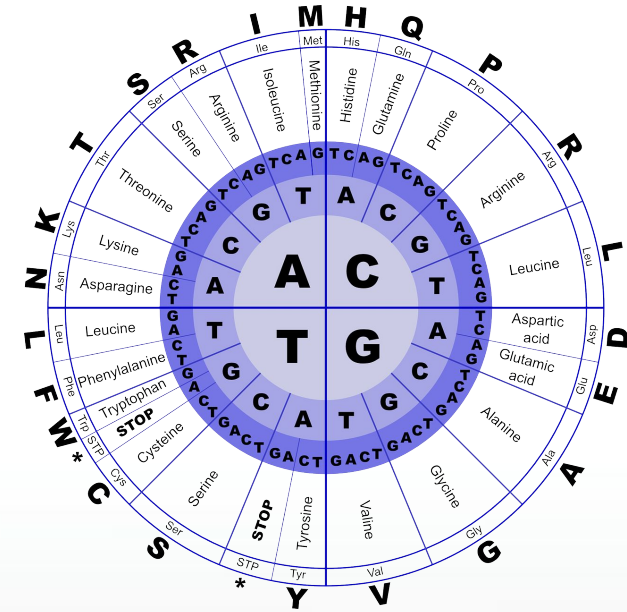
```
bin_dfs.Translated_sequences.duplicated().value_counts()
False      1385
True         923
Name: Translated_sequences, dtype: int64
```

Translating triplets into amino acids until the stopping codon allow us to shrink considerably the sequence length, nevertheless it, nevertheless dramatically increases the number of duplicates.

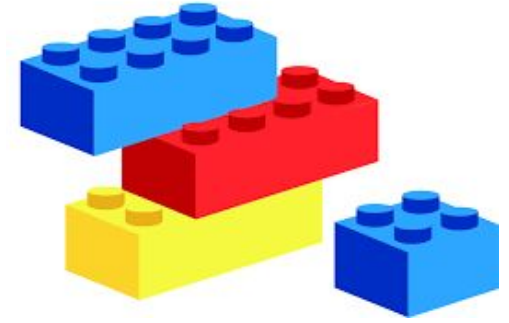
Handling the data

- We trained our models with both raw and translated DNA;
 - Bin {1,2,3,4} $\rightarrow X^{\text{train}}, y^{\text{train}}$
 - Bin {1,2,3} $\rightarrow X^{\text{subtrain}}, y^{\text{subtrain}}$
 - Bin {4} $\rightarrow X^{\text{val}}, y^{\text{val}}$
 - Bin {5} $\rightarrow X^{\text{test}}, y^{\text{test}}$

Polypeptides were extracted by reading nucleotidic triplets until the first stop codon, using *Biopython* library.



Our pipeline



1. Searching hyperparameters

Holdout phase (done for each hyperparameter)

- Train on $(X^{\text{subtrain}}, y^{\text{subtrain}})$ until either the maximum number of epochs is reached or the early stopping callback is invoked

2. Select best model:

Training phase:

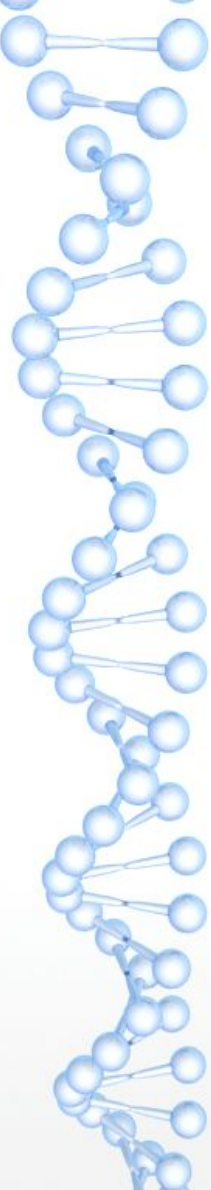
- Finetune the chosen model on X^{train} until the loss on y^{val} is lower than the one reached on y^{subtrain} at the end of the holdout phase

Test phase

- Test the model on X^{test}

One-Hot Encoding (Part 1)

- In **One-Hot Encoding** we represent each nucleotide by a row vector containing only one non-zero value in the corresponding column of the element represented.
- For a N length sequence of DNA and C possible values for each step the encoding will produce a $N \times C$ tensor.




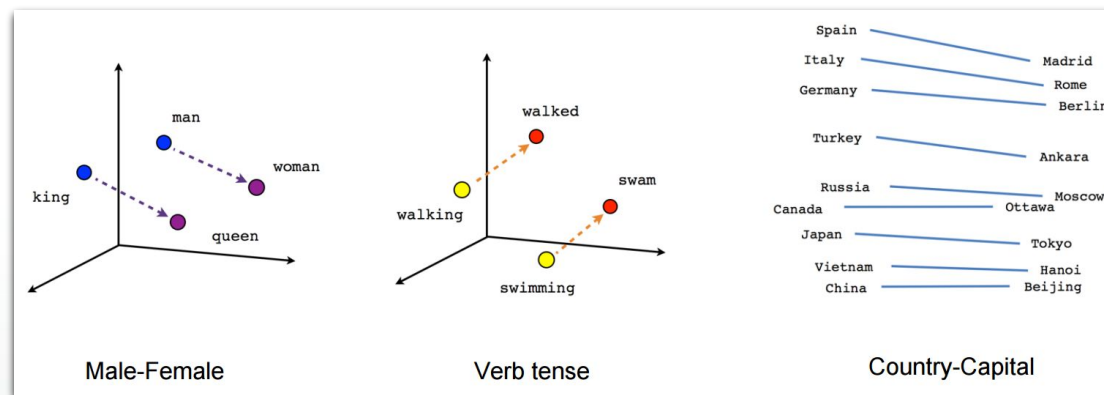
Vocabulary		OneHot				
PAD		1	0	0	0	0
A		0	1	0	0	0
T		0	0	1	0	0
C		0	0	0	1	0
G		0	0	0	0	1

Figure: one-hot encoding applied for DNA bases plus PAD character.

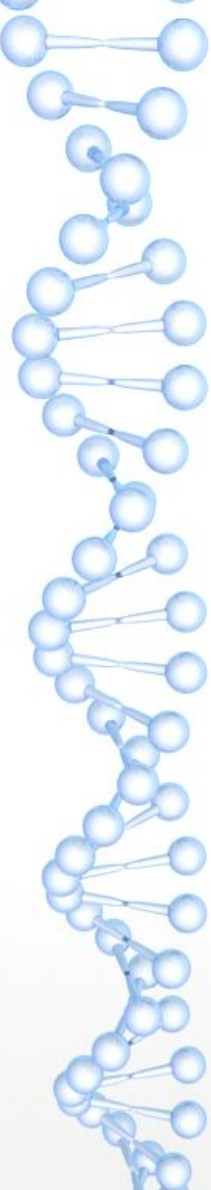
Embedding (Part 1)

- **Word embeddings** maps a language, e.g. DNA or polypeptide sequences, into a geometric space.
- In the embedding space, typically Euclidean:
 - data represent a domain dependent structure
 - vector arithmetic reflects the semantic.
 - we can extract semantic similarities between “words” in vocabulary by evaluating distance between them.



Embedding (Part 2)

- In our analysis word embedding is gradually built up from training by incoming gradients.
- The representation will be more data specific compared to one-hot encoding



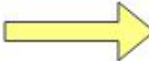
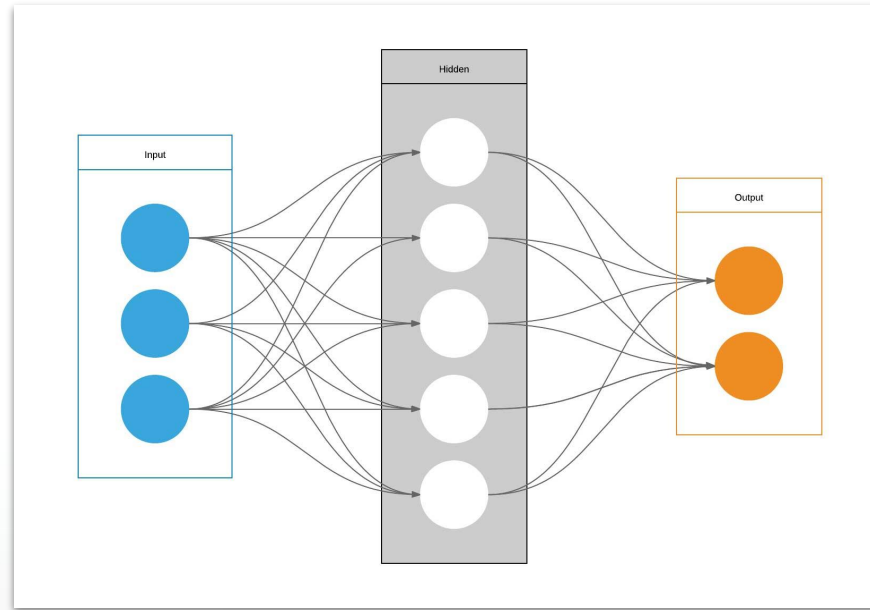
Vocabulary		Embedding				
PAD		1.2	-0.1	4.3	3.2	1.8
A		0.4	2.5	-0.9	0.5	-2.3
T		2.1	0.3	0.1	0.4	4.5
C		-0.3	0.4	1.5	3.0	-7.3
G		0.8	10.0	0.5	1.2	1.7

Figure: embedding strategy applied on DNA bases plus PAD symbol.

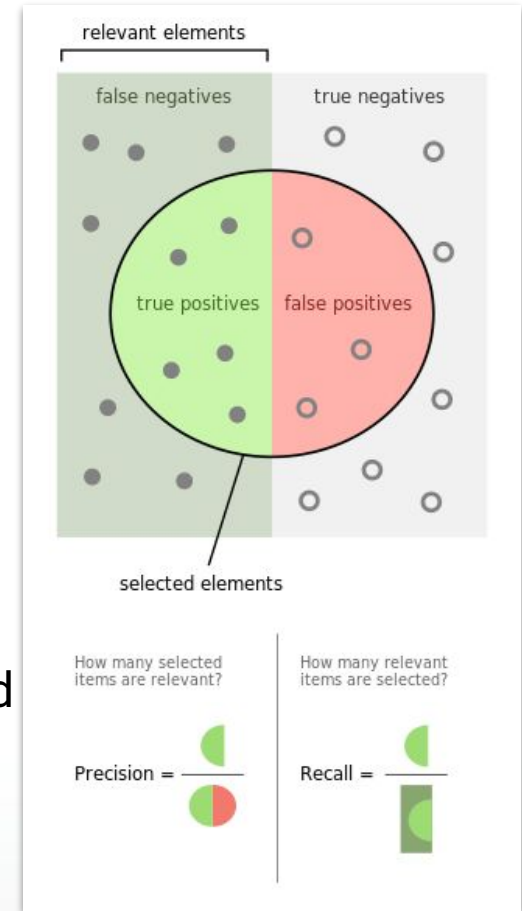
Architectural insights



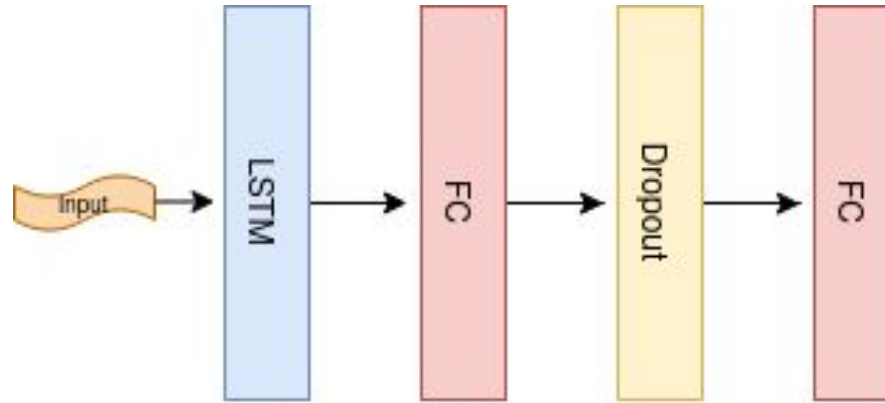
Evaluation

Metrics we used:

- **Precision (P)** $\rightarrow TP / (TP + FP)$
- **Recall (R)** $\rightarrow TP / (TP + FN)$
- **F1-score** $\rightarrow 2 * P * R / (P + R)$
- **Accuracy** $\rightarrow (TP + TN) / (TP + TN + FP + FN)$
- F1-score evaluation rather than accuracy
- It penalizes extreme values of P, and FP & FN more relevant than TN

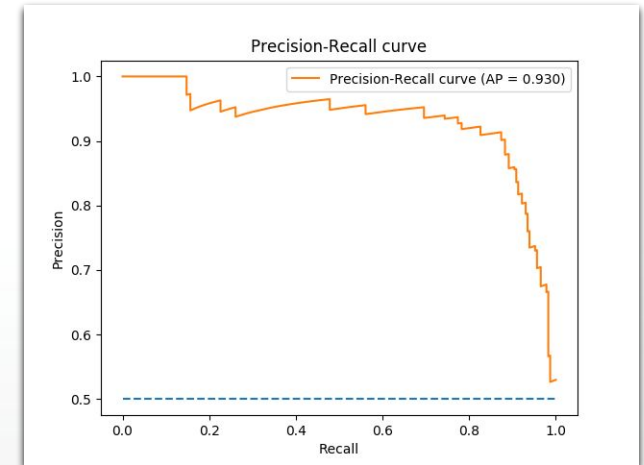


Unidirect LSTM (DNA)



Results on the test set (0.5 threshold)

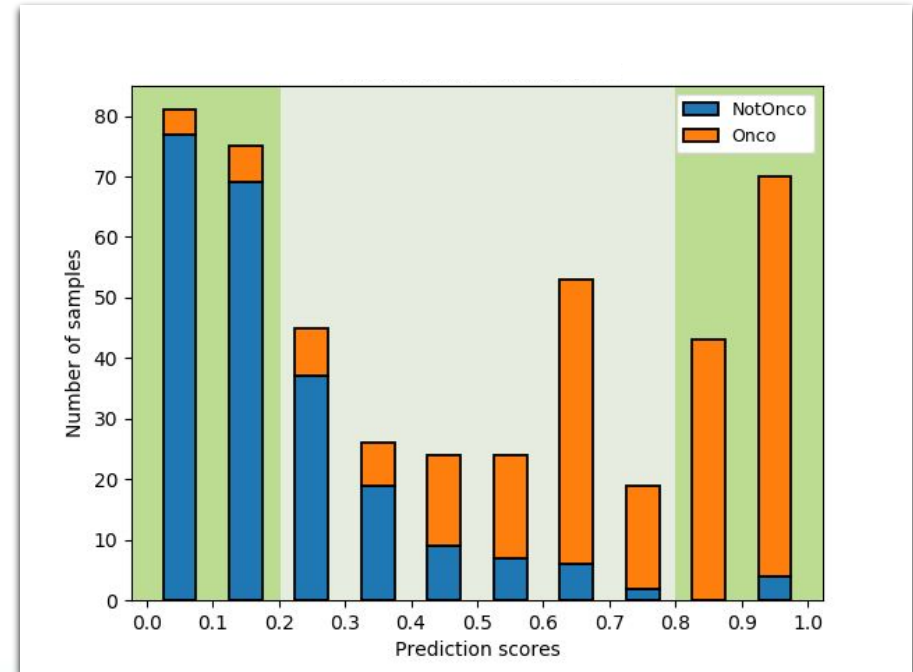
- Loss: **0.386**
- Accuracy: **0.871**
- F1-score: **0.865**
- Precision: **0.919**
- Recall: **0.828**
- AP: **0.930**
- TN/FP/FN/TP: **211/19/40/190**



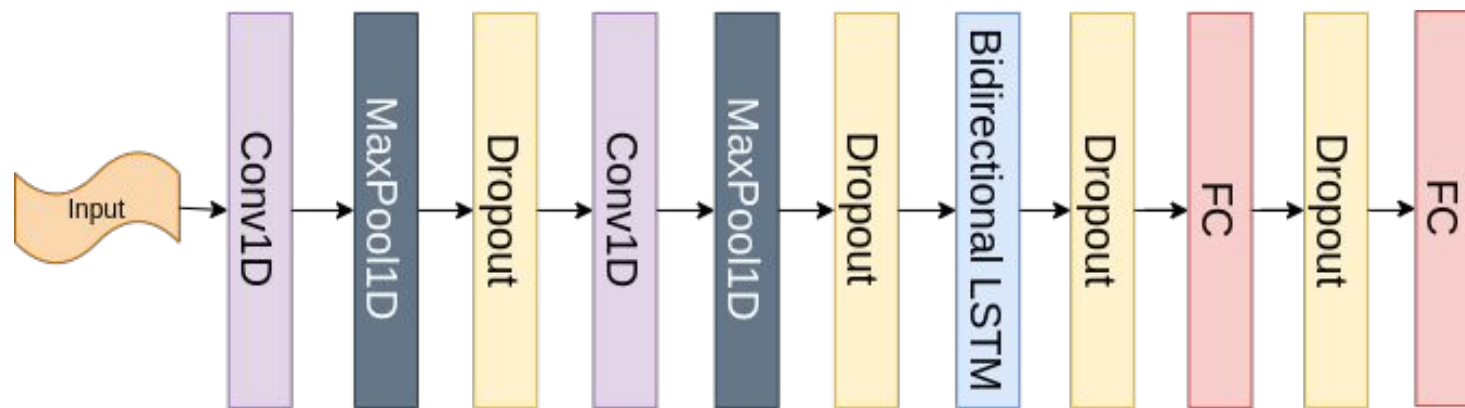
Unidirect LSTM (DNA) - 2

Results on the test set by using a 0.8 threshold:

- Samples classified: **58.47%**
- Accuracy: **0.948**
- Precision: **0.965**
- Recall: **0.916**
- F1-score: **0.940**

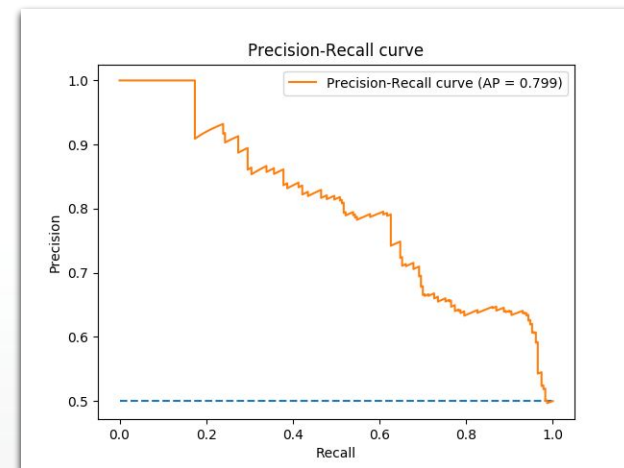


Bidirectional LSTM (DNA)



Results on the test set (0.5 threshold)

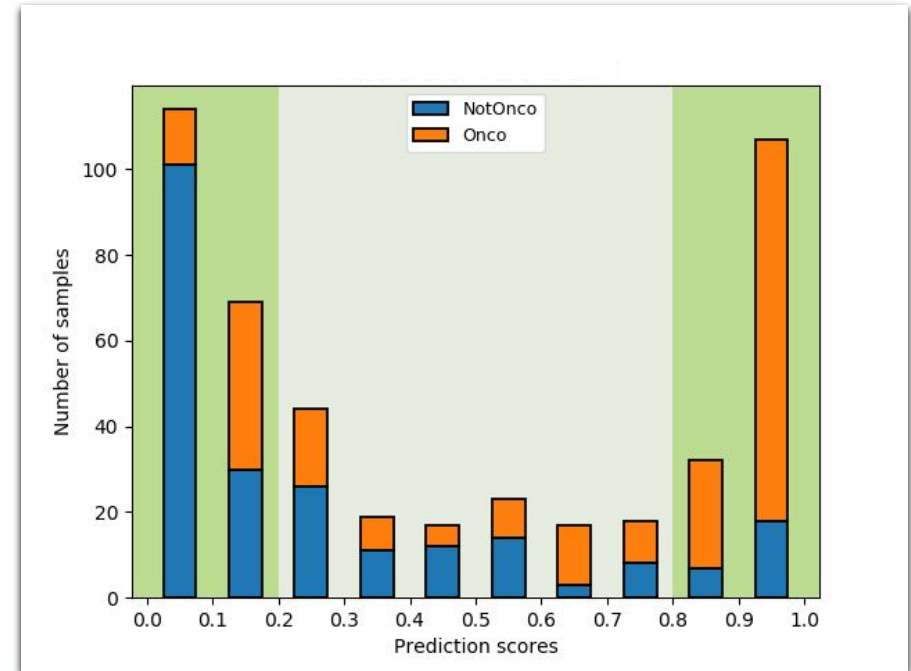
- Loss: **0.706**
- Accuracy: **0.711**
- F1-score: **0.677**
- Precision: **0.750**
- Recall: **0.633**
- AP: **0.799**
- TN/FP/FN/TP: **180/50/83/147**



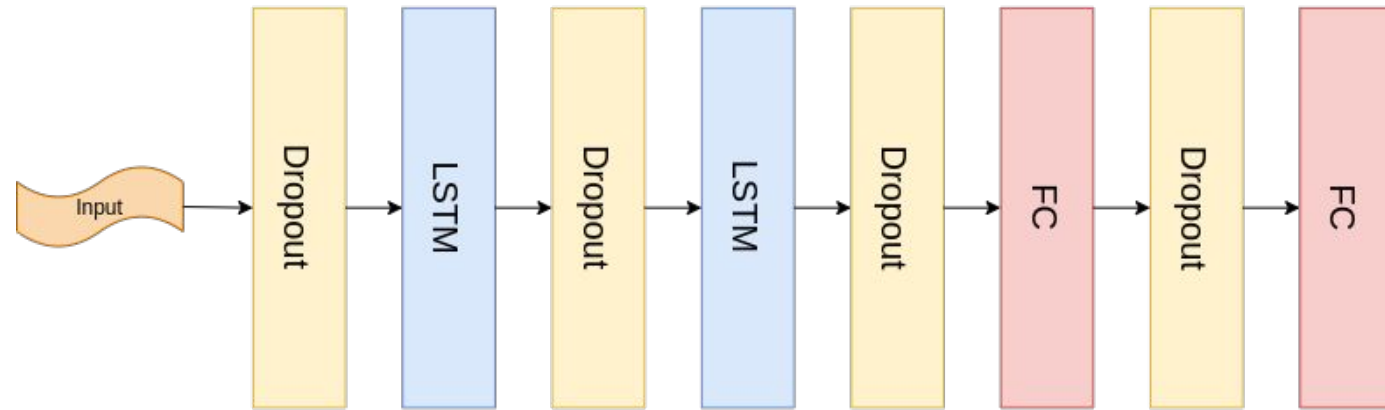
Bidirectional LSTM (DNA) - 2

Results on the test set by using a 0.8 threshold:

- Samples classified: **70.00%**
- Accuracy: **0.761**
- Precision: **0.820**
- Recall: **0.687**
- F1-score: **0.748**

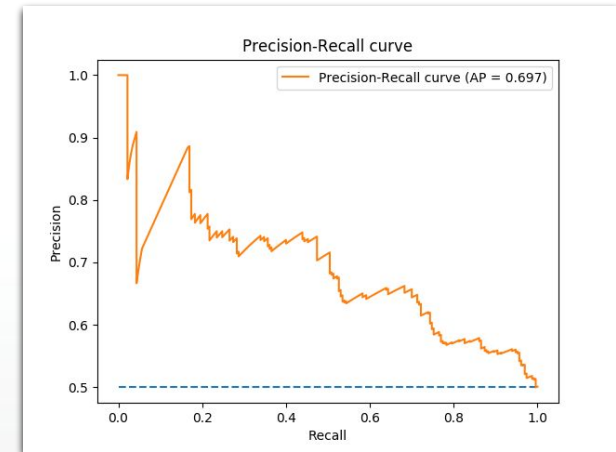


Unidirectional LSTM (proteins)



Results on the test set (0.5 threshold)

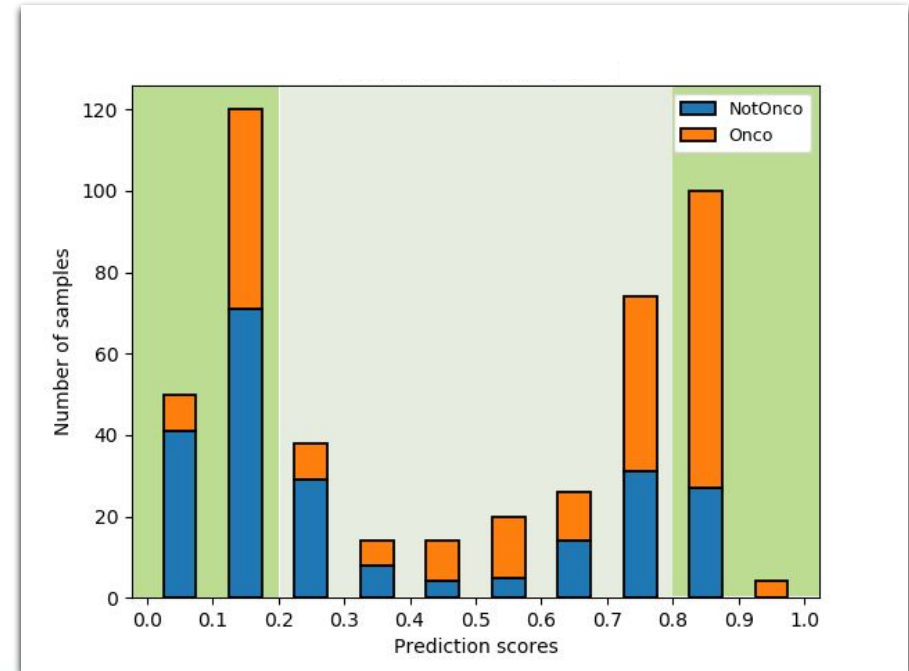
- Loss: **0.708**
- Accuracy: **0.652**
- F1-score: **0.643**
- Precision: **0.680**
- Recall: **0.634**
- AP: **0.697**
- TN/FP/FN/TP: **153/77/83/147**



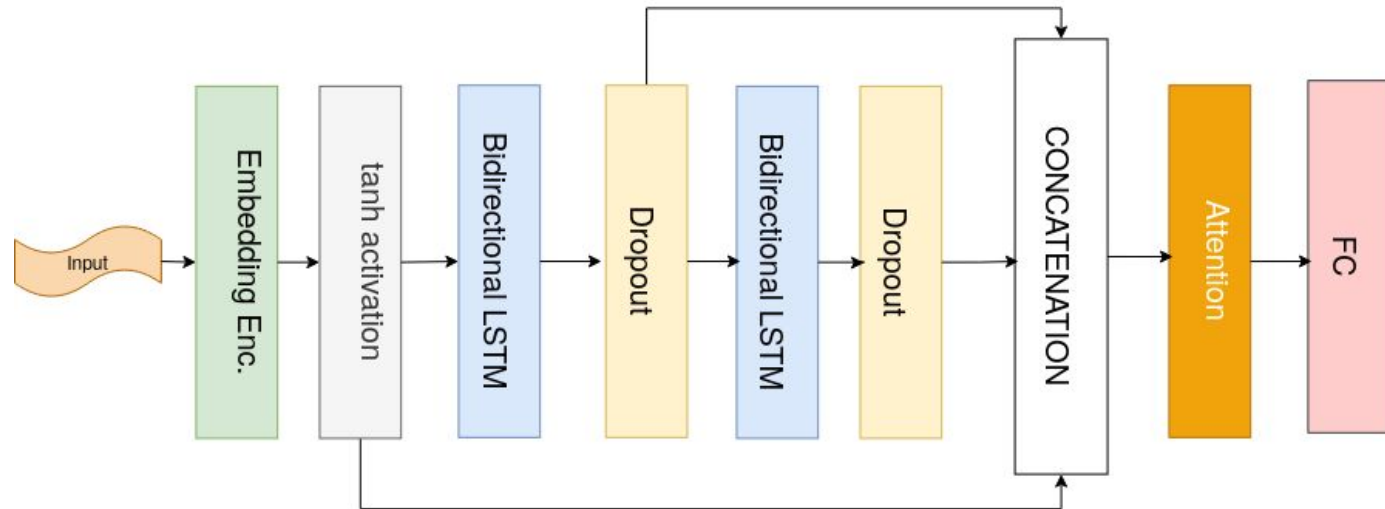
Unidirectional LSTM (proteins) - 2

Results on the test set by using a 0.8 threshold:

- Samples classified: **59.57%**
- Accuracy: **0.690**
- Precision: **0.740**
- Recall: **0.570**
- F1-score: **0.644**

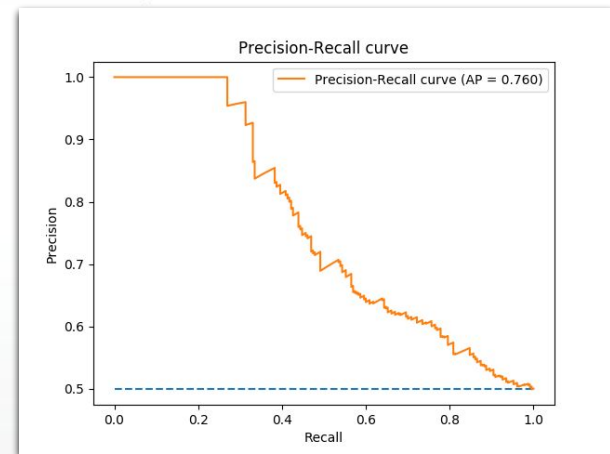


Bidirectional LSTM (Proteins) - Model A



Results on the test set (0.5 threshold)

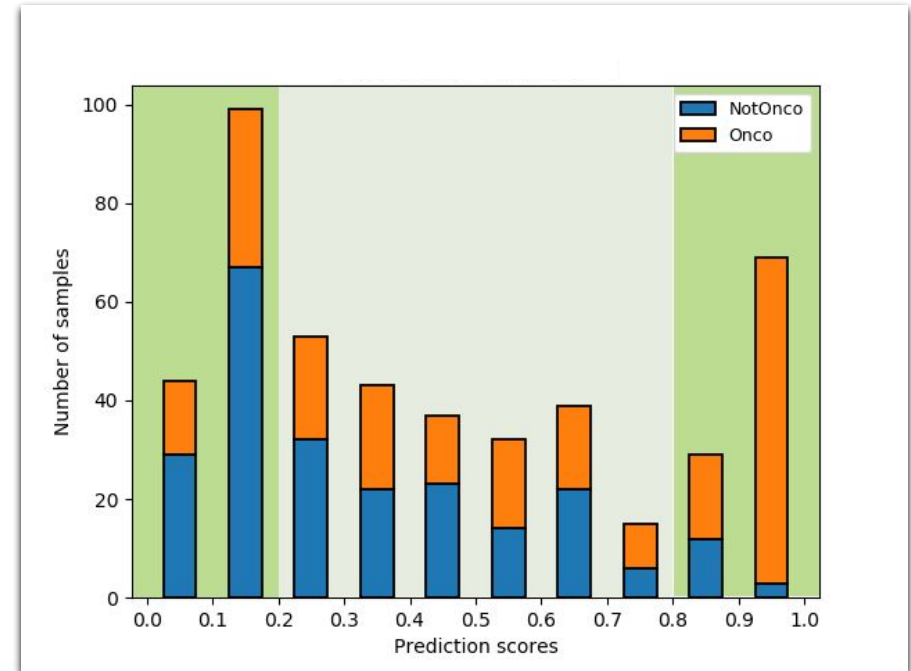
- Loss: **0.729**
- Accuracy: **0.652**
- F1-score: **0.610**
- Precision: **0.709**
- Recall: **0.551**
- AP: **0.760**
- TN/FP/FN/TP: **173/57/103/127**



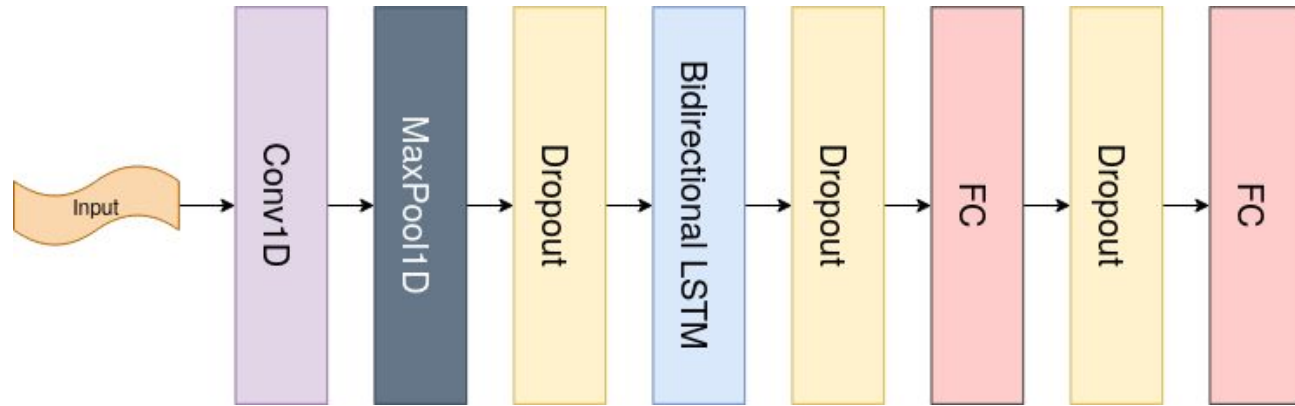
Bidirectional LSTM (Proteins) - Model A - 2

Results on the test set by using a 0.8 threshold:

- Samples classified: **52.39%**
- Accuracy: **0.743**
- Precision: **0.847**
- Recall: **0.638**
- F1-score: **0.728**

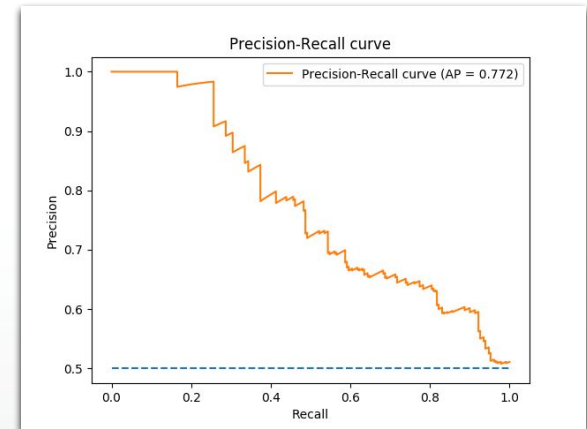


Bidirectional LSTM (Proteins) - Model B



Results on the test set (0.5 threshold)

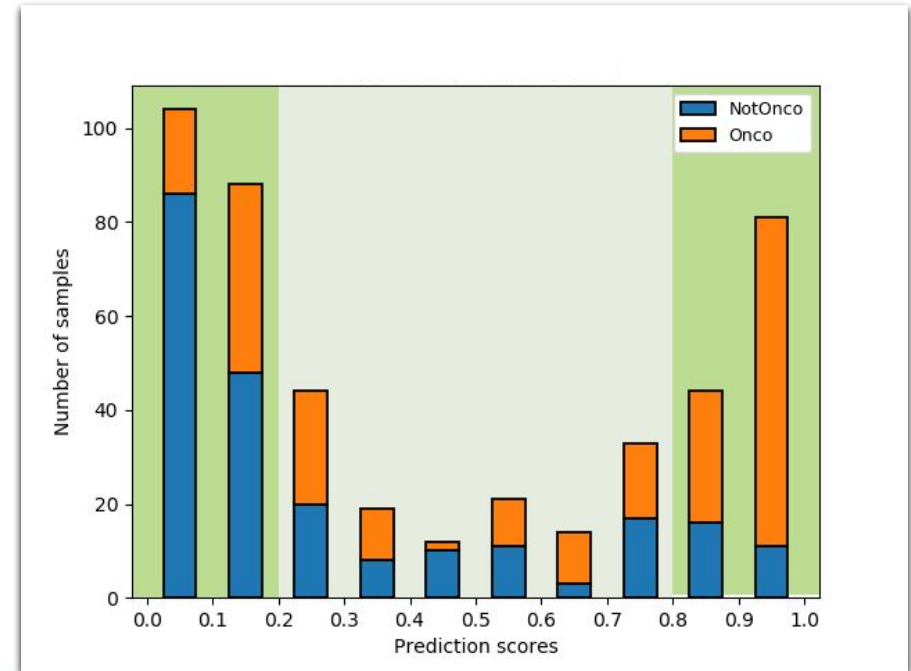
- Loss: **0.708**
- Accuracy: **0.667**
- F1-score: **0.615**
- Precision: **0.722**
- Recall: **0.584**
- AP: **0.772**
- TN/FP/FN/TP: **172/58/95/135**



Bidirectional LSTM (Proteins) - Model B - 2

Results on the test set by using a 0.8 threshold:

- Samples classified: **68.91%**
- Accuracy: **0.732**
- Precision: **0.784**
- Recall: **0.628**
- F1-score: **0.698**



Dealing with a small dataset

- Small datasets are more prone to be overfitted
- Necessity to limit the model's tendency to learn noise.
- Fewer model parameters allow to better generalize patterns found in data





Handling overfitting issue

Regularization techniques used to reduce the error arising from overfitting:

- L1 & L2 regularizations
- Dropout
- Small batch size
- Small learning rates
- Early stopping
- Learning rate decay



Conclusions

- LSTM networks tend to overfit when trained on a small dataset as the one used. Generalization is possible only by constraining parameters with a strong regularization.
- Longer sequences mean longer training time. We tried to overcome that by employing different approaches, such as conv+pooling or truncation. We obtain worse results compared to full-length sequences.
- Unfortunately, we did not have any direct comparison between the same architecture with LSTM and bidirectional-LSTM counterpart, because we focused more on a broad architectures exploration.
- As we can see DNA model on average perform better than protein ones. We guess that, because of the huge amount of duplicates in the protein dataset, the models based on them overfitted data.

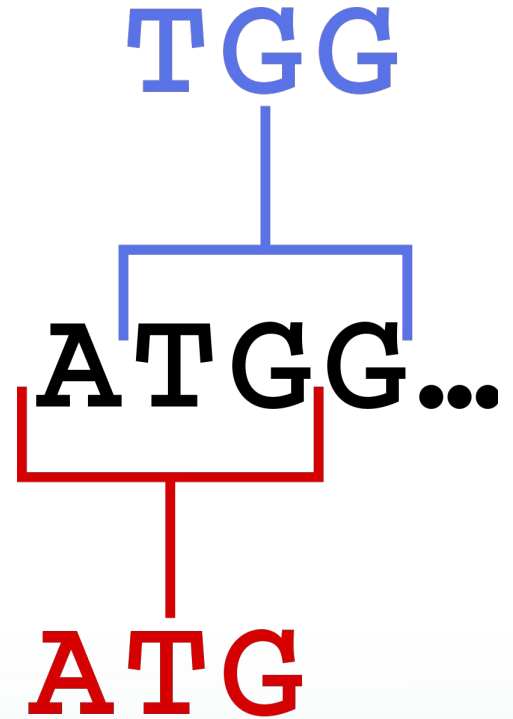


Extensions

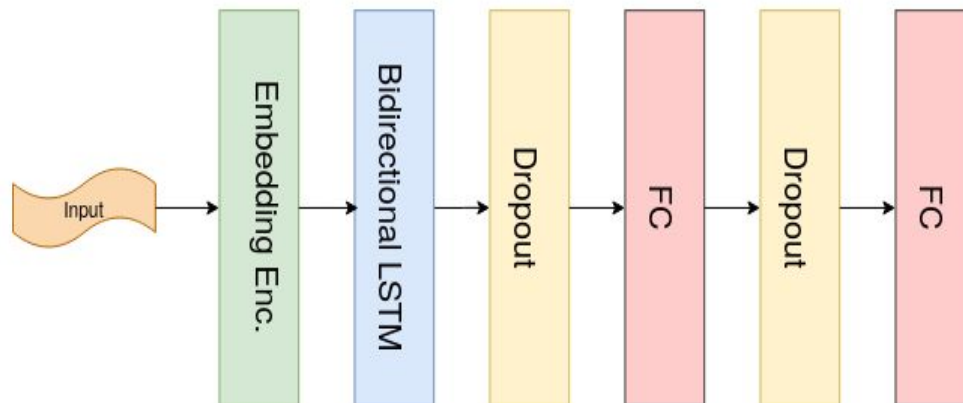
- More recent regularization techniques might be evaluated, such as recurrent dropout (not yet implemented in CUDA).
- It would be worth to evaluate the most performing architecture (model 1) with bidirectional LSTM. This will provide us a clear evidence of the effectiveness of inspecting also reversed sequences.

A further extension: kmers

- Words into the vocabulary: codons instead of single nucleotides
- Bidirectional LSTM architecture
- $K = 3$
- Stride = 1

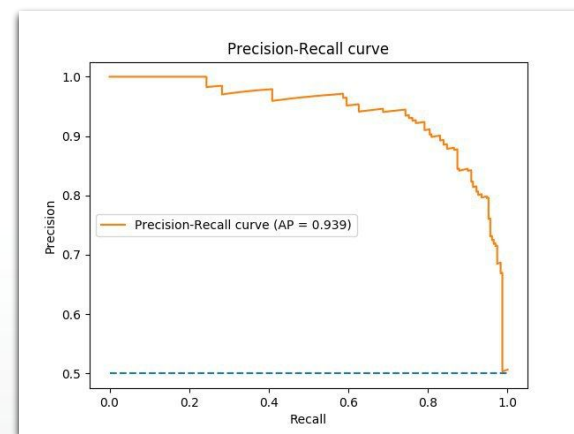


K-mers LSTM



Results on the test set (0.5 threshold)

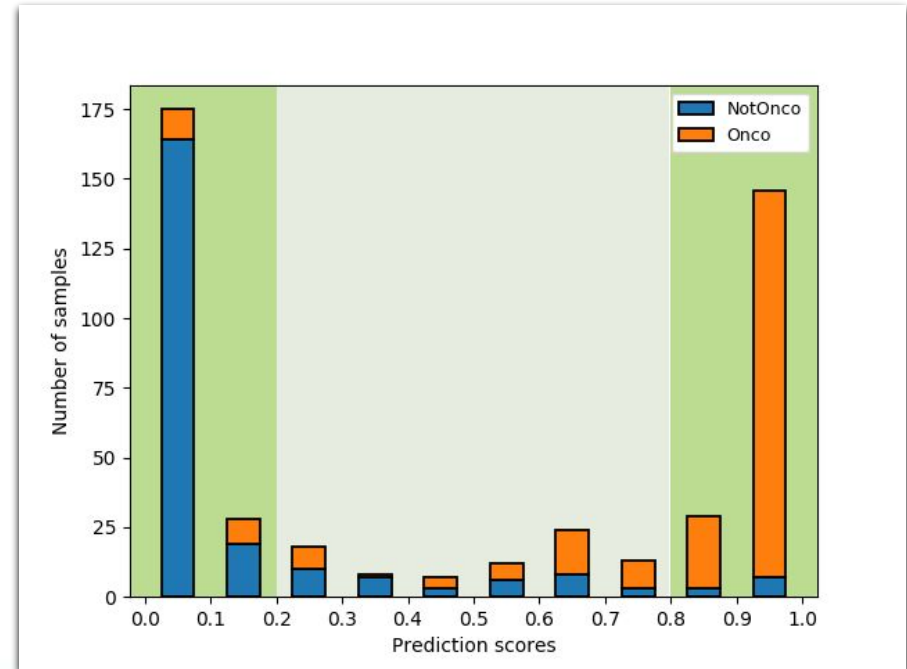
- Loss: **0.445**
- Accuracy: **0.870**
- F1-score: **0.863**
- Precision: **0.0.878**
- Recall: **0.856**
- AP: **0.939**
- TN/FP/FN/TP: **203/27/33/197**



K-Mers LSTM - 2

Results on the test set by using a 0.8 threshold:

- Samples classified: **82.17%**
- Accuracy: **0.920**
- Precision: **0.942**
- Recall: **0.892**
- F1-score: **0.917**





THANKS FOR YOUR ATTENTION!

