DL Homework 2

Gonçalo Santos
116212
goncalopikin@gmail.com

João Botas
116326
email@domain

Tiago Fontes
117387
tiagofontes99@gmail.com

Abstract

This homework project has as its main objective applying deep neural networks (DNN) architectures learned during the course classes. In the first exercise, a CNN model will be applied to an image classification task, In the second problem, there are thee parts, corresponding to a regression task with RNA binding protein interaction prediction. The first one, two models architectures will be chosen to be applied to the RNA sequence dataset with the objective of solving the regression problem of assigning an affinity value for the RNA sequence. In the second part, an attention mechanism will be applied to verify if it enhances the performance and generalisation capability of one of the first part chosen models. The third part will address a new approach for a multi-protein binding affinity task.

The contribution of each member of the group (n.º 28) were the following:

- Tiago Fontes: 1 and 2.3
- Gonçalo Santos: 2.1 and 2.3
- João Botas: 2.2

1 Question 1

In this section, we address points 1.1 and 1.2 of the homework assignment, which require the implementation of a convolutional neural network (CNN) whose architecture follows the structure specified in the project guidelines. The objective of this experiment is to train the proposed CNN on the BloodMNIST dataset in order to classify samples into eight distinct classes.

Four different models were trained and evaluated according to the experimental protocol defined in the guidelines. These models differ in the inclusion or exclusion of max-pooling layers, as well as in the presence or absence of a Softmax layer at the network output. The performance of each model was subsequently compared using validation and test accuracy metrics.

Figure 1 presents the validation accuracy as a function of the epoch number. A significant difference in convergence behaviour can be observed between models trained with and without an explicit Softmax layer.

This behaviour can be explained by the loss function used during training. The CrossEntropyLoss already incorporates a log-softmax operation internally. Consequently, applying an explicit Softmax layer in the forward pass introduces a redundant transformation, which negatively affects gradient computation. This redundancy leads to slower convergence and, in some cases, unstable training behaviour or failure to converge as shown by the models as they tend to have an unstable training pattern and diciculty to converge fully.

As shown in Figure 2, the training loss is presented as a function of the epoch number. This representation enables a direct comparison of the learning dynamics and convergence speed of the different models. It is evident that all models are capable of converging and adapting to the training data, although with noticeable differences in their convergence behaviour.

However, these differences do not generally translate into a significant degradation in validation performance, as models trained with and without a Softmax layer achieve comparable validation accuracy. An exception is observed for the model trained without max-pooling and with a Softmax layer, which reached a maximum validation accuracy of 0.8306 at epoch 35, after which its performance progressively degraded, indicating reduced generalization capability.

Additional differences can be observed when comparing models trained with and without max-pooling layers. Models incorporating max-pooling significantly reduced the overall training time. Specifically, the model without max-pooling and without a Softmax layer required approximately 619.88 seconds to complete training, whereas the

corresponding model with max-pooling completed training in approximately 236.70 seconds.

This phenomenon can be explained by the use of max-pooling layers, which reduce the spatial dimensions of the feature maps by retaining only the most relevant activations within each kernel region. As a result, the number of features and trainable parameters is significantly reduced, leading to lower computational complexity and faster training. Despite this reduction, the predictive performance remains comparable to that of the model without max-pooling, indicating that max-pooling improves efficiency without substantially affecting accuracy.

Despite the differences in training time, the validation accuracy achieved by the models was comparable for the configurations with max-pooling and without a Softmax layer, as well as for the models trained without max-pooling and without a Softmax layer. The maximum difference between the best- and worst-performing models among these configurations was approximately one percentage point.

In contrast, the model trained without max-pooling and with a Softmax layer exhibited significantly poorer performance, achieving validation accuracy values approximately ten percentage points lower than the remaining models. The highest validation accuracy was obtained by the model incorporating max-pooling without a Softmax layer, which achieved a validation score of 0.95619.

For clarity, training loss and validation accuracy are presented in separate figures, as a form to create a simpler and more understandable plot to visualize and compare between models and the same metrics.

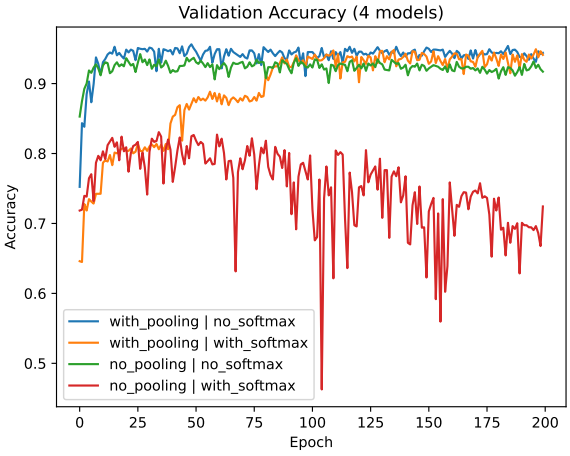

Figure 1: Validation accuracy

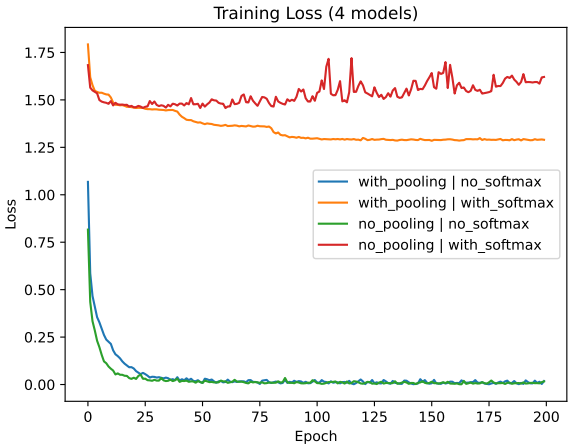

Figure 2: Training loss

2 Question 2

2.1

The chosen DNN architectures were LSTM and CNN models because the task involved assigning an affinity value to the RNA sequence. The LSTM is designed to process sequence inputs and capture long range sequence context. The CNNs are designed with convolutional filters to extract local patterns, invariant to position, making them effective for finding patterns in the sequences of this task, such as the motifs "UAGAC" for example (Teragawa and Wang, 2023). Moreover, according to Teragawa and Wang (Teragawa and Wang, 2023), convolutional kernels have been widely demonstrated to be effective feature extraction tools for handling sequence data.

We chose the LSTM architecture over standard RNNs because, although the sequences are relatively short, standard RNNs suffer from the vanishing gradient problem due to their limited memory mechanisms. On the other hand, LSTMs were designed to solve this problem through their gating mechanism (GeeksforGeeks, 2025).

We decided not to implement a Transformer architecture because, although an encoder-only Transformer could be applicable, it would require positional encodings and a larger dataset to learn the attention mechanisms effectively without overfitting. Moreover, a Transformer without positional encodings looks at the whole sequence at the same time, so it can't detect relationships (affinity) between nucleotides. Furthermore, learning this would take a lot of computational overhead.

2.1.1 First model.

The first selected model is a Bidirectional LSTM (BiLSTM). Since the whole RNA input sequence is already known and the goal is a regression task rather than an autoregressive task (not trying to predict the next nucleotide), the BiLSTM is well suited. Moreover, compared to the unidirectional LSTM, enhances the capability of the model by running two LSTMs in parallel, one processing the sequence from the forward direction and the other from the backward direction, achieving better relationships between nucleotides because it uses the previous nucleotide (past) and the next nucleotide (future) to understand the context of the data. The only drawback is that the training is slower due to the processing of sequences in both directions and requires more memory due to additional parameters (GeeksforGeeks, 2025).

The optimization strategy employed was GridSearch, with the following hyperparameters chosen for tuning:

- learning rate: [0.001, 0.0005]
- dropout: [0, 0.25]
- number of units (width): [64, 128]
- number of layers (depth): [1, 2]

The best selected checkpoint achieved a Spearman correlation of 0.6502 with the hyperparameters configuration: `'num_layers': 2, 'width': 128, 'dropout': 0, 'lr_rate': 0.001`

The learning rate range was selected based on the balance between convergence speed and training stability. The dropout values were selected to explore whether their regularisation and, thus, generalisation effects are better or not. Given the dataset medium size, we decided to constrain the network architecture with 1-2 layers and 64-128 hidden units, avoiding a powerful network that could overfit and an underfitting network.

To expedite the training convergence, the Adam optimizer was used. Furthermore, a fixed weight decay of 0.0001 was applied to penalise large weights in the model, mitigating overfitting. The training was set to 30 epochs, which was sufficient for achieving convergence and comparing the performance of the models.

The training and validation losses during the training are presented in the next figure:

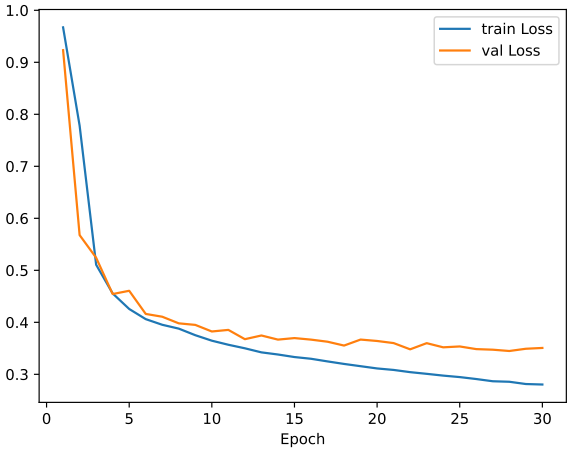

Figure 3: LSTM model - Train and Validation loss over epochs

2.1.2 Second model.

The second chosen model was the CNN designed for 1D and not 2D, since the input is a sequence and not an image.

The optimization strategy employed was GridSearch, with the following hyperparameters chosen for tuning:

- learning rate: [0.001, 0.0005]
- dropout: [0, 0.25]
- number of units (width): [64, 128]
- number of layers (depth): [1, 2]

The best selected checkpoint achieved a Spearman correlation of 0.5850 with the hyperparameter configuration: `'num_layers': 2, 'width': 128, 'dropout': 0.25, 'lr_rate': 0.001`

The same hyperparameters from the LSTM model were used by the CNN model, with the difference that the hyperparameter 'number of layers' pertains to the convolutional layers and not to the fully connected layers. The same optimizer and weight decay from the first model were maintained. The choice of activation function was ReLU over sigmoid and tanh, because it is not a saturated function, mitigating the vanishing gradient.

The training and validation losses during the training are presented in the next figure:

2.1.3 Comparative results.

As expected, the LSTM model outperformed the CNN model, demonstrating higher performance results without overfitting. As we can see in the figure 3, the validation loss follows the training loss

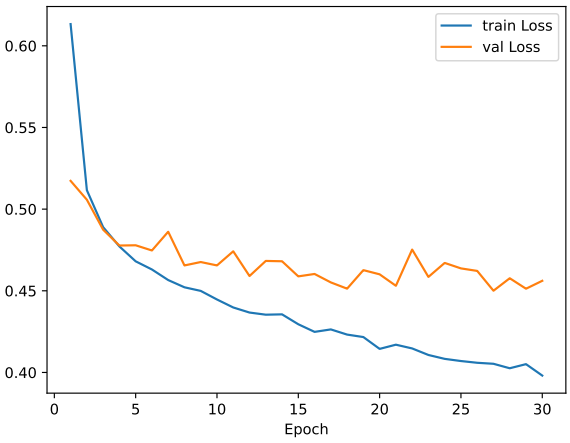

Figure 4: CNN - Train and Validation loss over epochs

in a downward direction over epochs, achieving a significantly lower validation loss (around 0.35) compared to the CNN (around 0.46). On the other hand, the CNN model overfitted, as shown in figure 4, the training loss is minimised over the epochs, but the validation loss stops decreasing consistently and starts to slightly diverge at epoch 17.

As mentioned before, the architecture of the BiLSTM can take context from the entire sequence (past and future nucleotides), which makes it well suited for this task. This shows that the global context over the sequences obtained by the LSTM was more relevant than the local context obtained by the CNN model for predicting binding affinity in this dataset.

Additionally, one intrinsic limitation of CNNs that use max global pooling is the loss of spatial relative information. Although they can detect that the motif is present, they lose the notion of its position and the spatial relationships within multiple motifs in the RNA sequence. Therefore, a motif could only have a greater influence at a certain position in the sequence, or if it is connected to or distanced from other specific motifs. A solution for this problem could be using the combination of Bi-LSTM with the CNN architecture (Hassanzadeh and Wang, 2016).

2.2

To extend the LSTM architecture, we incorporated an attention pooling mechanism applied on the hidden states produced by the bidirectional LSTM. After the RNA sequence is processed by the BiLSTM, the model outputs a sequence of hidden representations, one for each nucleotide position and then we apply attention to compute a weighted combination of all hidden states.

Was implemented a multi-head pooling attention, where multiple attention heads operate in parallel and each of them, potentially, learns to focus on different regions or patterns within the sequence, such as, which nucleotide positions are more relevant for predicting binding affinity, assigning higher weights to biologically meaningful regions such as binding motifs and their surrounding context. After that, their outputs are concatenated to form the final representation and give the final value for the binding affinity.

2.2.1 Results.

The attention variant that was chosen was the dot product attention, and we chose that, because in this problem, we want to predict a single scalar value from a RNA sequence, and not generate an output sequence, and besides that, this specific variation is also a simple but very effective form of attention. About the number of heads used, we think that maybe if we use 4 to 8 heads, the results would be better, and perhaps more that 8 heads would be over complex and saturated. To choose de number we performed the train and validation with various number of heads: $1; 2; 4; 8$.

2.2.2

Since, as we saw before, the curve of train loss and validation loss, are close to each other, we think that applying attention to the LSTM wouldn't perform mutch better than without attention.

2.2.3

So, for the implementation, was used the LTSM model and the hyperparameters chosen previously, so that, we could select the number of heads.

After, the process of training and validation, we compared the plot of the training and validation loss in function of the number of epochs, for the each number of heads, with the plot without attention, and we saw that for a number of heads of 4 and 8, the curve of validation loss was generally a little lower than the one without attention, and the curve for the 8 heads comparatively with the one for the 4 heads, seems to generalize a little better. Besides that, we can see that increasing the number of heads beyond a certain number, doesn't performed better.

Therefore, we ended up choosing a number of 8 heads for the attention:

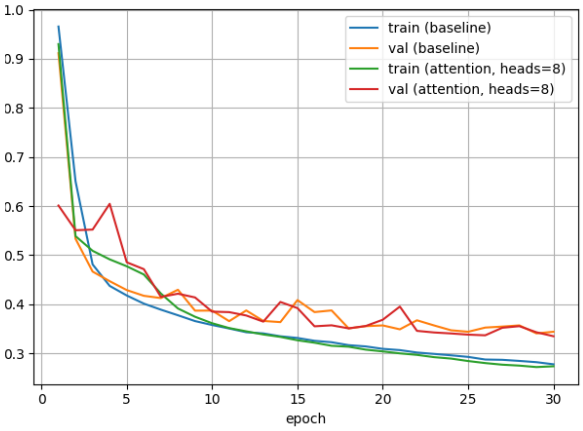

Figure 5: BiLSTM vs BiLSTM + attention (train/val loss).

2.2.4

Finally, we compared the performance on the test set before and after the incorporation of attention, resulting in this values:

Model	Spearman value
BiLSTM	0.6466
BiLSTM + Attention	0.6551

Table 1: Spearman value for the LSTM with and without attention on the test set.

Lastly, we can conclude that the model with attention performs a slightly better than the model without attention.

2.3

2.3.1 Proposed modifications

To develop a model that generalises to multiple RNA-binding proteins, certain aspects would need to be modified.

The dataset must be changed to include sequences for all available proteins. As the problem suggests, the data itself would now include two inputs, the RNA sequence and the identity (ID) of the corresponding protein, allowing the model to predict different affinities for the same RNA sequence depending on the target protein. The objective remains the same: predicting a scalar binding affinity. Therefore, the labels remain the same, but they should also be normalised (Z-scored) because the scale for each protein could be different.

The architecture of the model would need to be changed by adding a new branch, turning it into a two-branch architecture. The new branch would be a learnable embedding that processes and maps the identity of the proteins into a vector space. This representation is then concatenated with the output vector of the RNA sequence obtained from the other branch, the LSTM model used in exercise 2.1, for example, and is subsequently passed through a final Multi Layer Perceptron (MLP) so that the model can predict the correct binding affinity, taking into account the corresponding protein.

The training objective remains the Mean Squared Error (MSE). However, for the evaluation metric, the Spearman Correlation would need to be changed to the Mean Spearman Correlation for each Protein because it could be misleading due to different affinity ranges across different proteins. The Mean Spearman Correlation would calculate the correlation for each protein individually and then do the global average. Furthermore, the training and validation split must guarantee that there is data from all proteins to tune the hyperparameters effectively.

According to Teragawa and Wang (Teragawa and Wang, 2023), besides its task was different, for exercises 2.1, 2.2, and 2.3, the RNA encoder representation could be enhanced by replacing one-hot encoding with k-mers (for example, 2-mer or 3-mer) and word embedding vectors. This approach could reduce the dimensionality of the data, thus reducing computational complexity, and the model could capture local contextual information more effectively, where symbols with similar contexts have similar embedding representations (Teragawa and Wang, 2023).

2.3.2 Anticipated challenges and benefits

One potential benefit of training on multiple proteins would be that the model could learn universal structural RNA features that are relevant for all the proteins. This would solve, for example, the unbalanced dataset problem, where proteins with less data would not be ignored by the model during training.

On the other hand, one potential challenge of training is the contrast to the previous benefit mentioned. That is, what the model learns for a specific protein could prejudice the predictions for other proteins, since different RBPs can bind to distinct RNA motifs.

3 Collaboration with other teams and use of AI tools

For exercise 1, the code was based on PB class 8 (CNN), chatGPT was used for guidance and code

correction and simplification to prevent errors before training the model. For exercise 2.1, the code was based on PB classes 8 (CNN) and 9 (RNN and LSTM). However, gemini was used to verify errors in the code, and copilot was used within VSCode to take tips for faster and more efficient coding.

4 Further notes

In exercise 2.1, although the training was stable for the BiLSTM model, a future implementation should incorporate gradient clipping within the training loop. This should be applied before the optimizer step, to limit the norm of the gradient during the backward pass, being robust against exploding gradients. This is needed because the LSTM model mitigates the vanishing gradient but does not solve the exploding gradient (Martins et al., 2025).

5 Conclusions

As a summary, the proposed tasks were completed as expected and were built using theoretical concepts that were learnt during the course and during the homework, by reading papers, searching on the net, and trial and error experiments when coding. In the first exercise, a CNN with max-pooling and without softmax layer demonstrated to be the best architecture for the CNN model, for the respective classification task. In exercise 2.1, the BiLSTM model outperformed the CNN model, with sequential and positional features being better than local features obtained by the CNN model that does not take into account the position of the motifs. In exercise 2.2, the use of multi-head attention enhanced the performance of the BiLSTM model. Finally, in exercise 2.3, a new approach was presented for the multi-protein task, by using the protein ID information as input and two branched model, a BiLSTM to encode the RNA sequence and a learnable embedding layer to encode the protein ID into a dense vector representation.

References

GeeksforGeeks. 2025. Difference between a bidirectional lstm and an lstm. Accessed at: 02-01-2026.

Reza Hassanzadeh and May D. Wang. 2016. Deeperbind: Enhancing prediction of sequence specificities of dna binding proteins. *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*.

André Martins, Chrysoula Zerva, and Mário Figueiredo. 2025. Lecture 8: Recurrent Neural Networks. Deep Learning Course slides, Instituto Superior Técnico. Winter 2025-2026.

Shoryu Teragawa and Lei Wang. 2023. Conf: A deep learning model based on bilstm, cnn, and cross multi-head attention mechanism for noncoding rna family prediction. *Biomolecules*, 13(11).