DL Homework 2

João Tomás de Almeida Santos Antunes Gomes
106204

Maxence Jacques Marcel Bomo
116832

Gonçalo da Costa Miranda Teixeira de Jesus
96864

Abstract

This report summarizes the contributions of each group member: João worked on Question 1, Gonçalo on Questions 2.1 and 2.2, and Maxence on Question 2.3.

1 Question 1

1.1 Simple Convolutional Network

Implementation Details

We implemented a Convolutional Neural Network (CNN) to classify images from the BloodMNIST dataset. The architecture follows the specifications:

- **Conv Block 1**: 3 input channels $\rightarrow$ 32 output channels, kernel 3×3, stride 1, padding 1 + ReLU.
- **Conv Block 2**: 32 $\rightarrow$ 64 output channels, kernel 3×3, stride 1, padding 1 + ReLU.
- **Conv Block 3**: 64 $\rightarrow$ 128 output channels, kernel 3×3, stride 1, padding 1 + ReLU.
- **Flatten**: Since no pooling was used, the spatial dimensions remained 28×28. The flattened feature vector has size $128 \times 28 \times 28 = 100,352$.
- **Linear Layers**: A fully connected layer mapping $100,352 \rightarrow 256$ features (ReLU), followed by a final layer mapping $256 \rightarrow 8$ classes.

Training Setup:

- **Optimizer**: Adam ($lr = 0.001$)
- **Loss Function**: `nn.CrossEntropyLoss`
- **Epochs**: 200
- **Batch Size**: 64

Table 1: Comparison of Logits vs. Softmax

Metric	**No Softmax**	**With Softmax**
Convergence	Fast, stable (≈ 0.0)	Stalled (≈ 1.5)
Test Accuracy	$\approx$ **93.25%**	$\approx$ **68.78%**
Stability	Stable	Unstable (Spikes)

Comparison: With vs. Without Softmax Layer

We conducted two experiments to verify the correct usage of the loss function:

1. **Without Softmax (Logits)**: The model outputs raw scores. `nn.CrossEntropyLoss` applies `LogSoftmax` internally.
2. **With Softmax**: The model applies `Softmax` before the loss function. This results in `LogSoftmax(Softmax(x))`.

Discussion of Results and "Spikes": The model trained with the **Softmax layer** performed significantly worse (69% vs 93%).

- **Optimization Failure**: The loss stalled at ≈ 1.5 because the double application of Softmax restricts the inputs to the loss function to the interval [0,1]. The optimizer struggles because gradients vanish or become distorted near the boundaries.
- **The Spikes**: The "With Softmax" accuracy graph shows violent drops. This occurs because the optimizer builds momentum trying to force the Softmax output beyond 1.0. When weights shift slightly, predictions collapse (e.g., becoming uniform), causing accuracy to plummet before recovery.

1.2 Impact of MaxPool2d

Implementation Changes

We modified the network by adding a `nn.MaxPool2d(kernel_size=2, stride=2)`

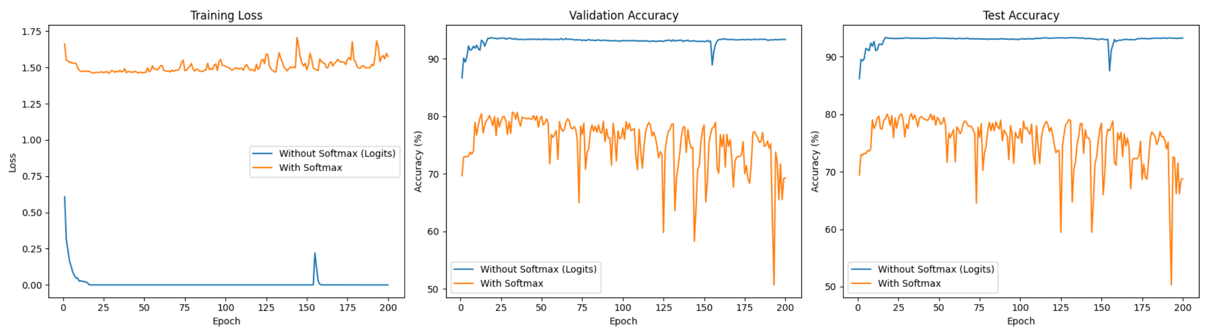

Figure 1: Training loss and accuracy for the simple CNN.

layer after every ReLU activation in the convolutional blocks.

Architecture Impact:

- **Dimensionality Reduction**: Spatial resolution is halved at each block ($28 \to 14 \to 7 \to 3$).
- **Parameter Count**: The input to the first linear layer is reduced from $100,352$ to $1,152$.
  - Q1.1 Parameters (FC1): ≈ 25.6 Million.
  - Q1.2 Parameters (FC1): ≈ 0.3 Million.

Analysis of MaxPooling Impact

We repeated the experiments (Logits vs Softmax) with the new architecture.

1. Effectiveness (Accuracy)

- **Logits (Best)**: Accuracy improved from **93.25%** (Q1.1) to **94.39%** (Q1.2).
- MaxPooling introduces **translation invariance** and acts as a regularizer, reducing overfitting.

2. Efficiency (Training Time & Compute)

- **Training Time**: Q1.1 took $\approx$ **1210s**, while Q1.2 took $\approx$ **765s** ($\approx$ **37% reduction**).
- **Computational Cost**: The **98% reduction** in the first dense layer's weights drastically reduces memory usage and gradient computation time.

3. Stability (Why did the spikes disappear?) In the Q1.2 "With Softmax" experiment, the violent spikes disappeared (though accuracy was still lower at $\approx 85.9\%$).

- **Reason**: Q1.1 had ≈ 25M parameters; instability propagates explosively.
- **Constrained Space**: Q1.2 has only ≈ 300k parameters. This constraint stabilizes the optimization landscape, preventing wild swings even with the broken gradient.

Conclusion

1. **Softmax vs Logits**: Never apply Softmax before `CrossEntropyLoss`. It creates instability and degrades accuracy.
2. **MaxPooling**: Highly beneficial. Increases effectiveness (+1.14% Acc) and efficiency (98% fewer parameters).

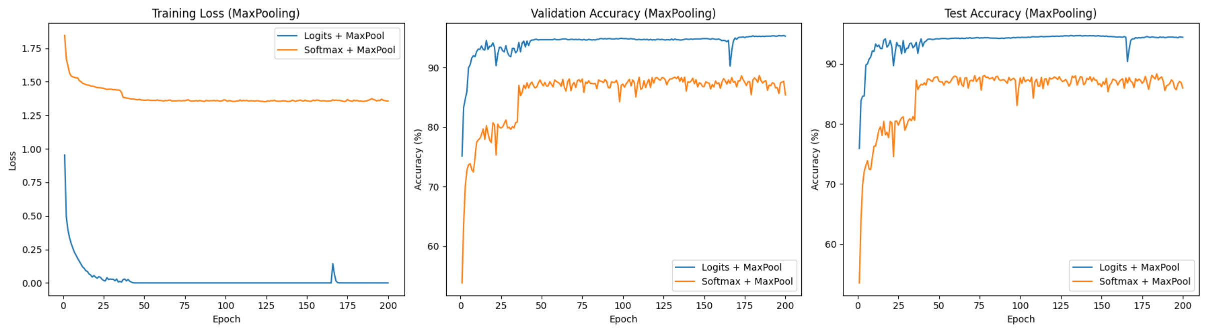

Figure 2: Training Loss, Validation and Test Accuracy with MaxPooling (Q1.2)

2 Question 2

2.1

Model Selection and Justification

To predict the binding affinity between the RBFOX1 protein and RNA sequences, we selected two complementary Deep Learning architectures: a 1D CNN and a Bi-LSTM. The CNN was chosen for its translation invariance and efficiency in local feature extraction, enabling the detection of consensus motifs (k-mers) regardless of their position within the sequence, which are subsequently refined by a downstream MLP classifier. Conversely, the Bi-LSTM was implemented to capture the sequential nature and global context of the data; by processing information bidirectionally, the model infers long-term dependencies and structural relationships between distant nucleotides, overcoming the memory limitations of traditional RNNs. We prioritized these architectures over more complex attention-based models (e.g., Transformers) because, given the short sequence length (41 nt), CNNs and LSTMs offer the optimal trade-off between computational efficiency and predictive capacity, avoiding the unnecessary complexity and overfitting risks associated with larger models.

Hyperparameter Tuning

Regarding hyperparameter optimization, we first selected a CNN kernel size of 12. Given that the RBFOX1 consensus motif ('UGCAUG') has a length of 6, a kernel of 12 allows the model to capture the core motif together with sufficient flanking context. For the training duration, we observed that the recurrent architecture converged slower than the convolutional one; therefore, we extended the LSTM training to 50 epochs to ensure full convergence, while the CNN required only 25. In terms of model capacity, we increased the LSTM hidden dimension to 64, as this added complexity yielded better generalization than the initial 32 units. In contrast, for the CNN, we maintained 32 filters, as increasing this number resulted in overfitting without performance gains. Finally, we experimented with different learning rates but retained the default 0.001, as lower values did not yield significant improvements.

2.1.1 First model.

Describe the first model, the hyperparameters you chose to vary, and the hyperparameters of the selected checkpoint. Provide requested plots.

Performance Analysis

The comparative analysis reveals that the Bi-LSTM model outperformed the CNN architecture, achieving a higher peak Spearman correlation (≈ 0.6485 versus ≈ 0.577). While we initially hypothesized that the CNN would be highly effective due to the strong local dependence of RBFOX1 binding on the specific 'UGCAUG' motif, the results indicate that local pattern matching alone is insufficient for optimal prediction. This outcome suggests that binding affinity is significantly modulated by the global sequence context and potential RNA secondary structures; unlike the CNN, which is restricted to a local receptive field defined by its kernel size, the bidirectional LSTM successfully captured these long-range dependencies, processing the sequence as a holistic structure rather than a collection of isolated fragments.

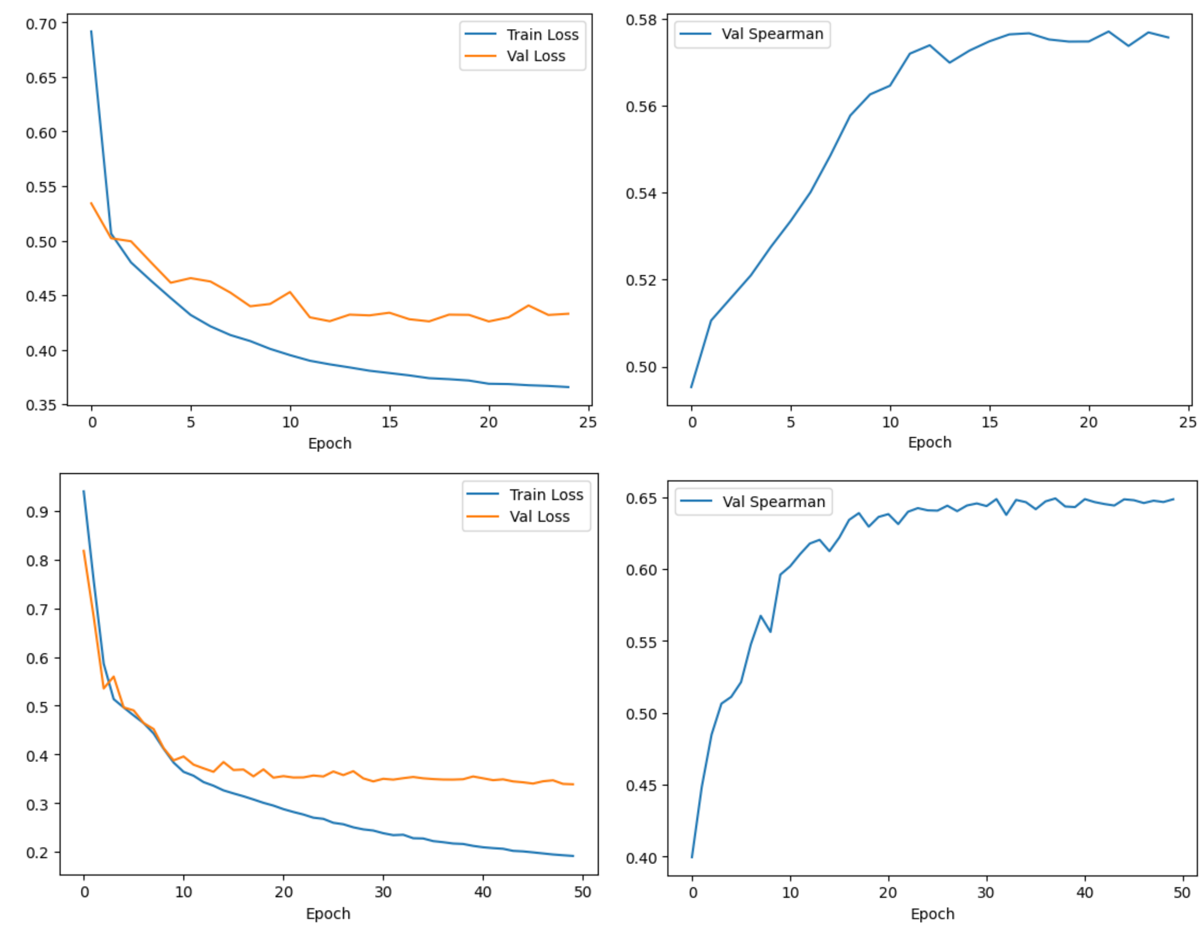

Figure 3: **Training curves for CNN and Bi-LSTM.**
Top row (CNN, Kernel=12, Filters=32). The model converges quickly, with validation loss stabilizing around 0.43 and Spearman correlation reaching a plateau of approximately 0.577 by epoch 15.
Bottom row (Bi-LSTM, Hidden=64, Epochs=50). The validation loss decreases steadily, and the Spearman correlation achieves a superior peak of approximately 0.650 around epoch 45.

2.2 Attention Mechanism

Implementation and Justification

To extend the Bi-LSTM architecture, we implemented an Attention Pooling mechanism rather than a multi-headed self-attention block. Given that the binding of RBFOX1 is driven by specific, localized motifs (e.g., 'UGCAUG'), a complex multi-head architecture would likely introduce unnecessary parameters and overfitting risks for such short sequences (41 nt). Instead, we employed a single attention head that computes a scalar importance score for each time step using a learnable linear projection followed by a Softmax normalization. This allows the model to calculate a context vector as a weighted sum of the LSTM hidden states, effectively learning to 'highlight' the relevant motif positions while suppressing the background noise of non-interacting nucleotides.

Expecations

We hypothesized that integrating an attention mechanism would primarily impact the efficiency and stability of the training process rather than strictly increasing the upper bound of the Spearman correlation. By providing the network with a direct mechanism to focus on the binding motif, we expected the model to converge significantly faster, as the gradient signal could flow directly to the relevant inputs without degrading over time steps. Furthermore, we anticipated a reduction in the final validation loss, as the ability to assign near-zero weights to irrelevant sequence padding allows the model to filter out noise that might otherwise confuse a standard LSTM, leading to more confident and precise affinity predictions.

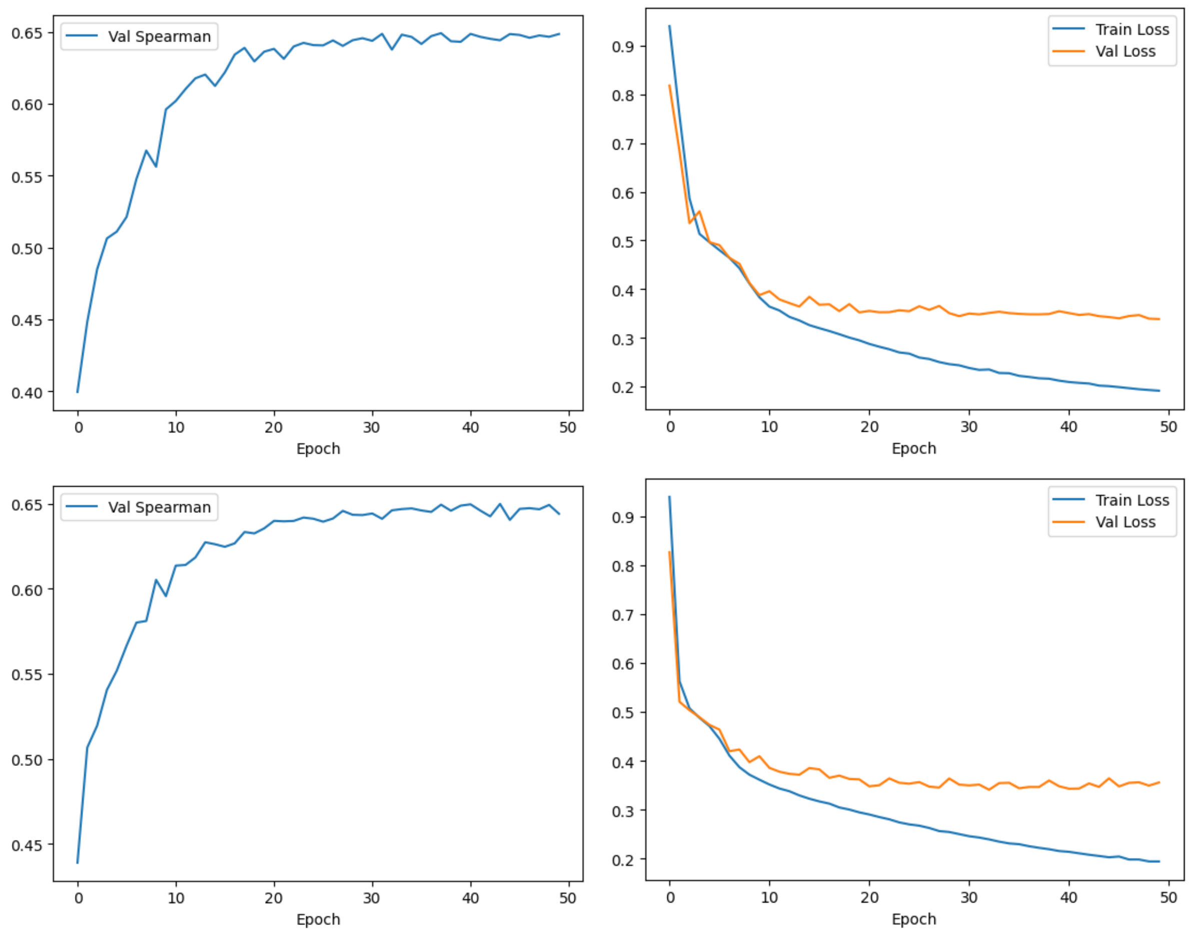

Figure 4: **Visual Comparison (Top: Baseline, Bottom: Attention)**. Note how the Spearman curve in the bottom row (Attention) shoots up immediately in the first 10 epochs, whereas the top row (Baseline) takes a gradual slope. The Loss (Right) also settles at a lower value with Attention.

Results & Comparison

The incorporation of the Attention Pooling mechanism altered the learning dynamics but did not surpass the performance ceiling established by the baseline model.

1. **Convergence Speed (Improved):** The most notable benefit was the acceleration of early learning. The Attention model crossed the significant Spearman correlation threshold of 0.60 as early as **Epoch 9** (0.605), whereas the baseline Bi-LSTM typically required 12-15 epochs to reach this level. This confirms that the attention mechanism successfully directed gradients to the relevant motifs faster.

2. **Peak Performance (Similar):** The maximum Spearman correlation achieved was **0.650** (Epoch 44), which is virtually identical to the baseline model. This strongly suggests that the predictive limit is constrained by the information content of the dataset (aleatoric uncertainty) rather than the model's capacity.

3. **Generalization (Slight Degradation):** Interestingly, the final validation loss for the Attention model was slightly higher (≈ 0.356) compared to the baseline (≈ 0.338), despite the training loss being lower (0.19 vs 0.20). This indicates that the added parameters from the attention layer led to mild **overfitting**, where the model began to memorize noise in the training set rather than generalizing better.

Conclusion: While Attention Pooling accelerated the initial detection of binding motifs, the simpler Bi-LSTM baseline proved to be more robust and equally accurate for this specific dataset and sequence length.

2.3

2.3.1 Proposed modifications

To extend the model to multiple RNA-binding proteins, the data must include pairs of RNA sequences and protein identities or features (e.g., protein IDs, amino-acid sequences, or learned embeddings), while the label remains the binding affinity. The model architecture should be adapted to jointly encode RNA and protein information, for instance using a dual-encoder design where one network processes the RNA sequence and another encodes the protein, followed by a fusion layer and a shared prediction head. The training objective can remain a regression loss over binding affinity, but the evaluation protocol should explicitly assess generalization by holding out entire proteins during validation or testing rather than random RNA–protein pairs.

2.3.2 Anticipated challenges and benefits

A key benefit of this multi-protein setup is improved generalization through shared representations of RNA–protein interactions, enabling accurate predictions for proteins with limited experimental data. However, a major challenge lies in the heterogeneity of binding mechanisms across different proteins, which increases task complexity and may lead to negative transfer if the model capacity or protein representations are insufficient.

3 Collaboration with other teams and use of AI tools

Briefly describe collaboration and AI tool usage here.

4 Conclusions

Brief summary.