Protein Subcellular Localization

Project Report

Moh-Ouali Amrouni
Rayane BELKALEM

February 13, 2026

Contents

1 Goal 2

2 Constraints and environment 2

3 Data 2

4 Pipeline 2

5 Embedding choice and compute budget 2

6 Model architecture 3

7 Training 3

8 EDA summary 3

9 Results 6

10 Limitations and future work 7

11 Reproducibility 7

12 Conclusion 8

1 Goal

We predict the subcellular localization of prokaryotic proteins (6 classes) from FASTA sequences. The project requires an architecture more advanced than a plain MLP, using embeddings, a Transformer fusion block, a BiLSTM refinement, and a classification MLP.

2 Constraints and environment

Experiments were run on an ASUS TUF 15, mostly on CPU. This creates memory and time constraints for embedding extraction. To stay within limits we:

- cap sequence length (`max_len`) to avoid OOM;
- keep ProstT5 3Di in CPU mode with disk offload;
- cache HF models locally to avoid re-downloads;
- tune conservative batch sizes for stability.

3 Data

Data comes from DeepLocPro in FASTA format. Headers follow:

```
>PROTEIN_ID|LOCATION|GRAM_TYPE|PARTITION
```

We generate `metadata.csv` and leakage-safe train/val/test splits (stratified by default).

4 Pipeline

1. **Metadata preparation** from FASTA.
2. **Split creation** with leakage control.
3. **Embedding extraction**:
   - ESM-C (300M) as main embedding (dim 960).
   - ProstT5 3Di (dim 1024) as required second embedding.
4. **Training** of Transformer + BiLSTM + MLP model.
5. **Evaluation** on test split (Accuracy, Macro-F1, MCC).

5 Embedding choice and compute budget

The assignment requires ESM-C + ProstT5 3Di. ProstT5 is expensive on CPU, so we used disk offload and a batch size of 1 for stable extraction.

To keep extraction feasible, we set `max_len=1000`. This keeps most sequences while remaining practical on CPU.

6 Model architecture

Our model uses *embeddings* → *Transformer* → *BiLSTM* → *MLP*:

- linear projection of both embeddings into a common space;
- encode modality tokens with a small Transformer encoder;
- process encoded tokens with BiLSTM and attention pooling;
- fuse pooled LSTM output with `[CLS]` before MLP classification.

Main configuration

Parameter	Value
Embedding 1	ESM-C 300M (dim 960)
Embedding 2	ProstT5 3Di (dim 1024)
Pooling	meanpool
max_len	1000
ESM batch	16
Embed2 batch	1

Table 1: Main extraction and training configuration.

7 Training

We use focal loss with class weighting for imbalance and early stopping on Macro-F1.

8 EDA summary

The dataset has 11,906 sequences. Main columns: `protein_id`, `sequence`, `label`, `gram_type`, `partition`, `split`, `seq_len`. No missing values observed.

Class distribution

Class	Count
Cytoplasmic	6885
Cytoplasmic Membrane	2535
Extracellular	1077
Outer Membrane	756
Periplasmic	566
Cell Wall	87

Table 2: Class distribution (EDA).

Sequence length stats

- mean: 438.4, std: 289.0

- min: 8, median: 399, max: 5627
- P10: 146, P25: 264, P75: 557, P90: 787, P95: 878, P99: 1296

For CPU runs, `max_len` between 600 and 1000 is recommended. With `max_len=1000`, most sequences are kept while compute remains manageable.

Split sizes

Train: 8333, Val: 1191, Test: 2382.

Figures

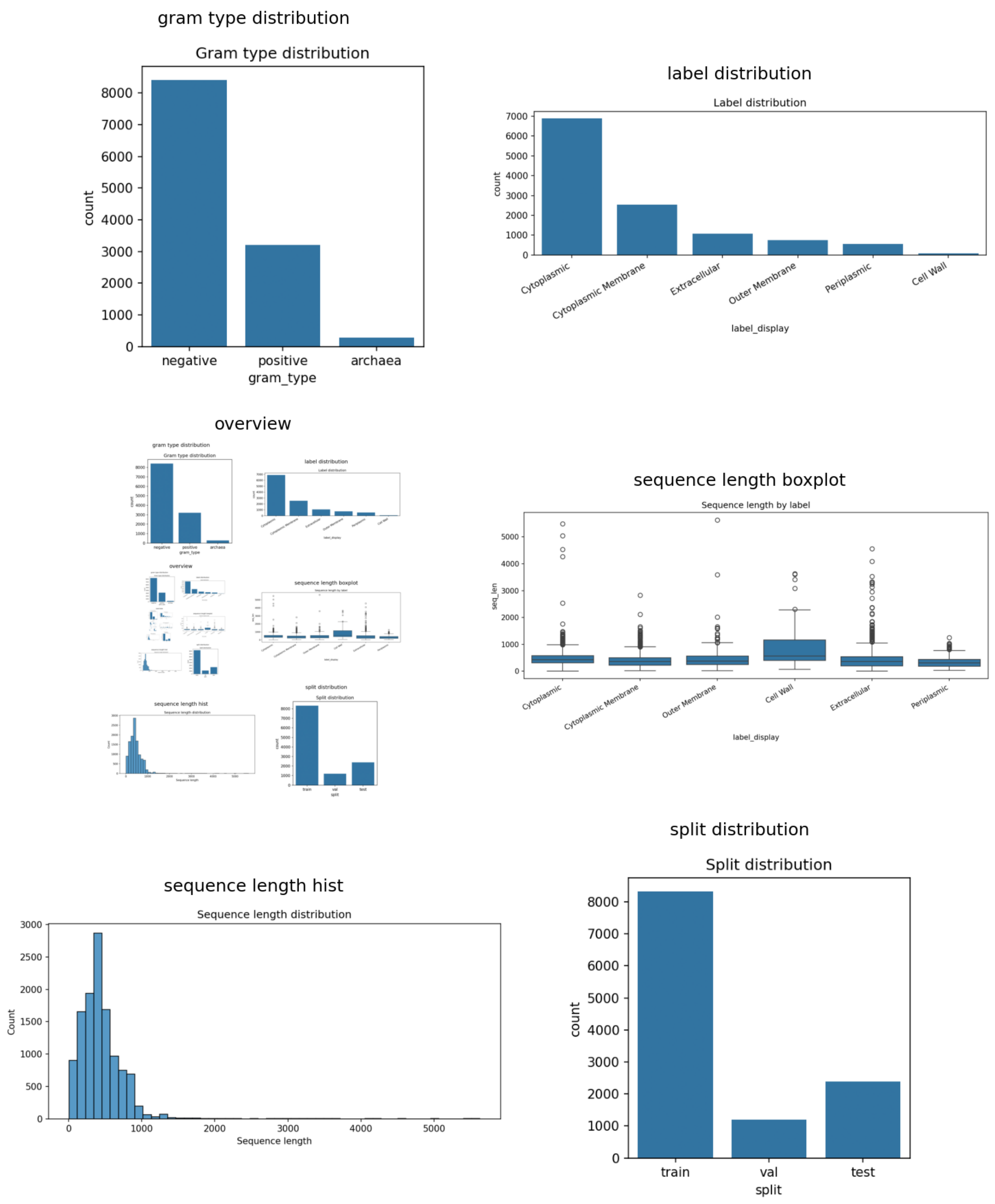

Figure 1: Overview: labels, lengths, splits.

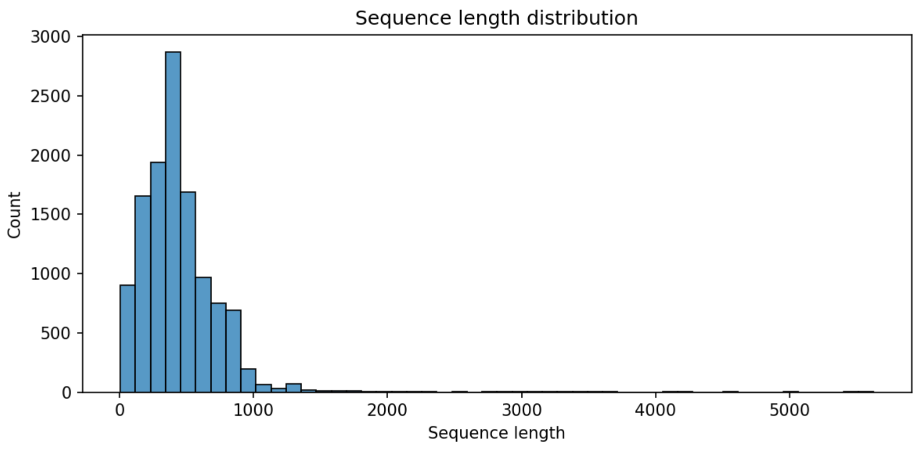

Figure 2: Sequence length distribution.

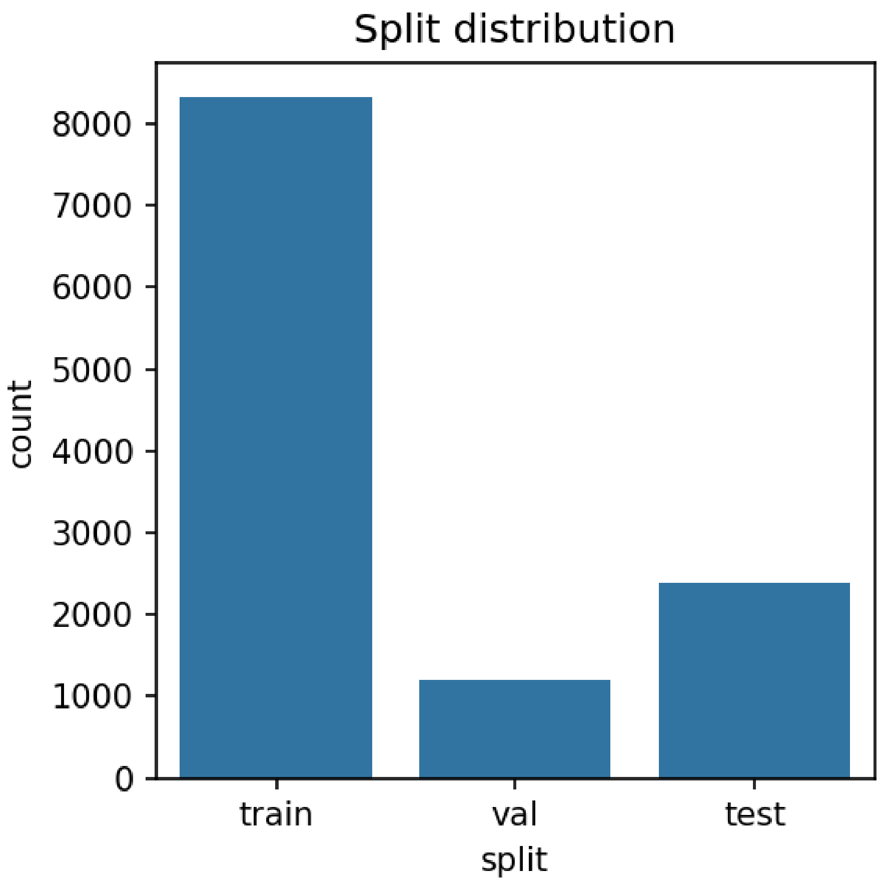

Figure 3: Train/Val/Test split.

9 Results

Run configuration: ESM-C + ProstT5 3Di (meanpool), `max_len=1000`, CPU execution.

Global metrics

Run	Accuracy	Macro F1	MCC
Evaluation	0.8741	0.8005	0.8227

Table 3: Global test metrics.

Per-class metrics

Class	Precision	Recall	F1
Cytoplasmic	0.901	0.846	0.873
Cytoplasmic Membrane	0.500	0.750	0.600
Periplasmic	0.933	0.925	0.929
Outer Membrane	0.776	0.839	0.806
Extracellular	0.706	0.818	0.758
Cell Wall	0.857	0.818	0.837

Table 4: Per-class precision/recall/F1.

Figure not available: run evaluation then `make eda`.

Figure 4: Global metrics.

Figure not available: run evaluation then `make eda`.

Figure 5: Per-class F1.

10 Limitations and future work

- Increase `max_len` or use a GPU to retain more sequence information.
- Quantify the exact gain of BiLSTM versus pure Transformer fusion.
- Explore deeper Transformer fusion or alternative pooling strategies.

11 Reproducibility

Main commands:

```
python scripts/prepare_metadata.py --fasta data/raw/graphpart_set.fasta --output data
python scripts/prepare_splits.py --metadata data/processed/metadata.csv --output data
python -m src.embeddings.fetch_embeddings --embed2_backend prostt5 --max_len 1000 --p
python scripts/train.py --config configs/default.yaml
python scripts/evaluate.py --checkpoint results/checkpoints/best_model.pt --config co
```

12 Conclusion

The end-to-end pipeline is operational under CPU constraints and uses a custom *embeddings + Transformer + BiLSTM + MLP* architecture beyond the original paper.