

From Current to Real: Deep Learning for Real-Time Segmentation of Nanopore Signals

Maxime Rochkoulets^{1,2}, Sara Bakić^{1,3}, Mile Šikić¹

¹ Genome Institute of Singapore, Agency for Science, Technology and Research

² KU Leuven

³ National University of Singapore



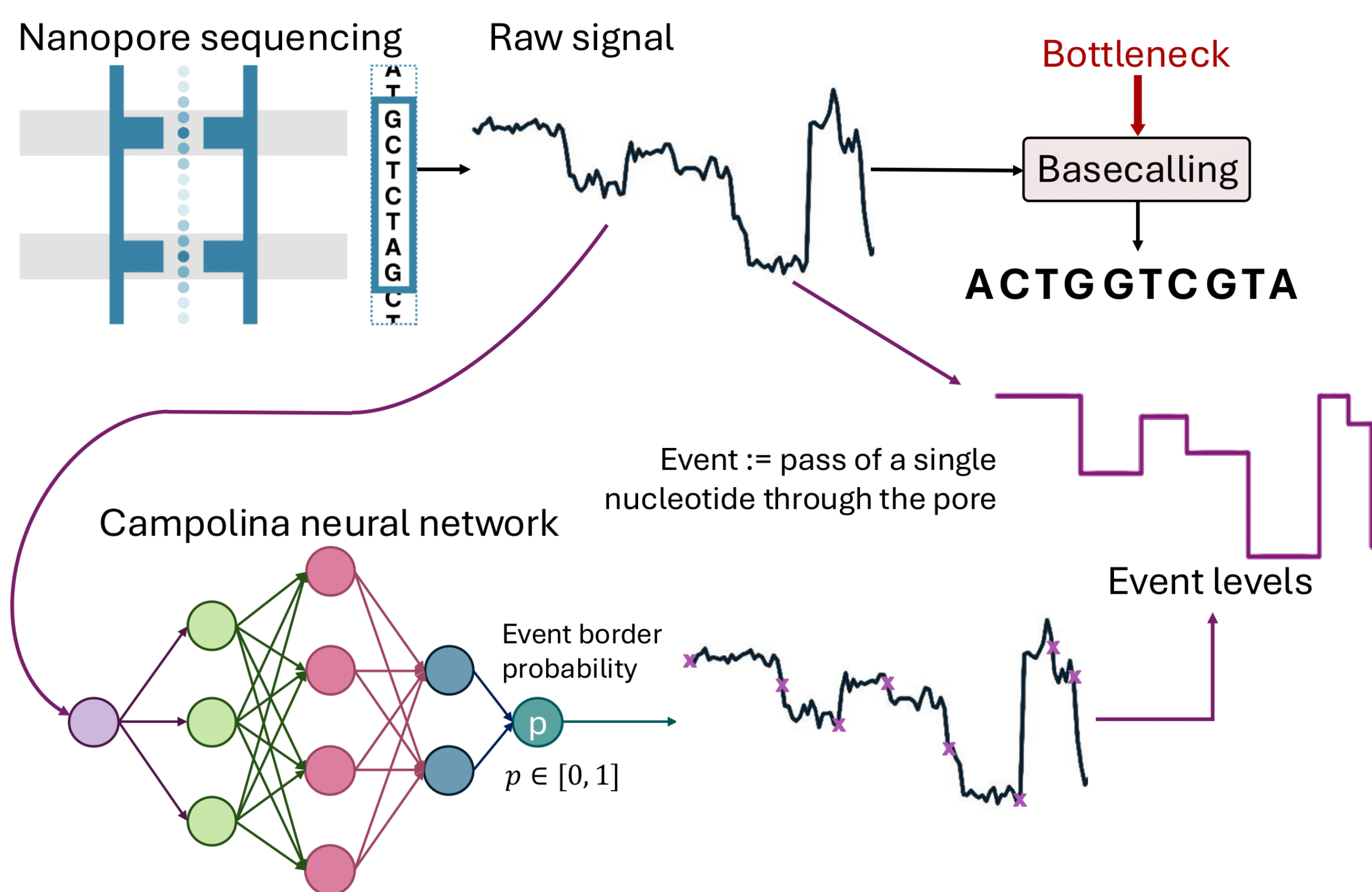
Campolina Paper



GitHub

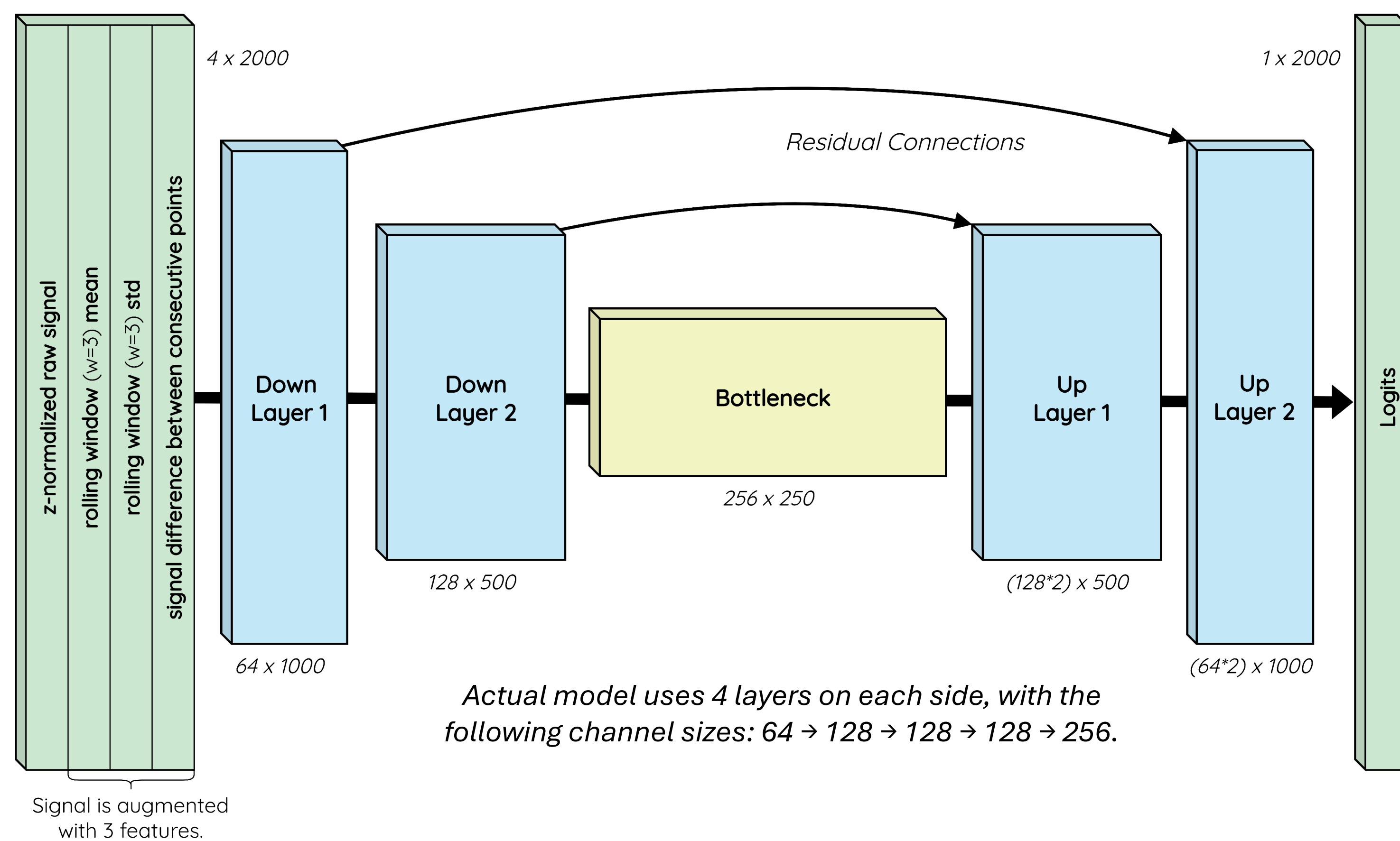
INTRODUCTION

This work is based on “Campolina: A Deep Neural Framework for Accurate Segmentation of Nanopore Signals”, from S. Bakić, K. Friganović, B. Hooi and M. Šikić. We use the same framework, dataset and evaluation pipeline, and focus on improving their proposed model.

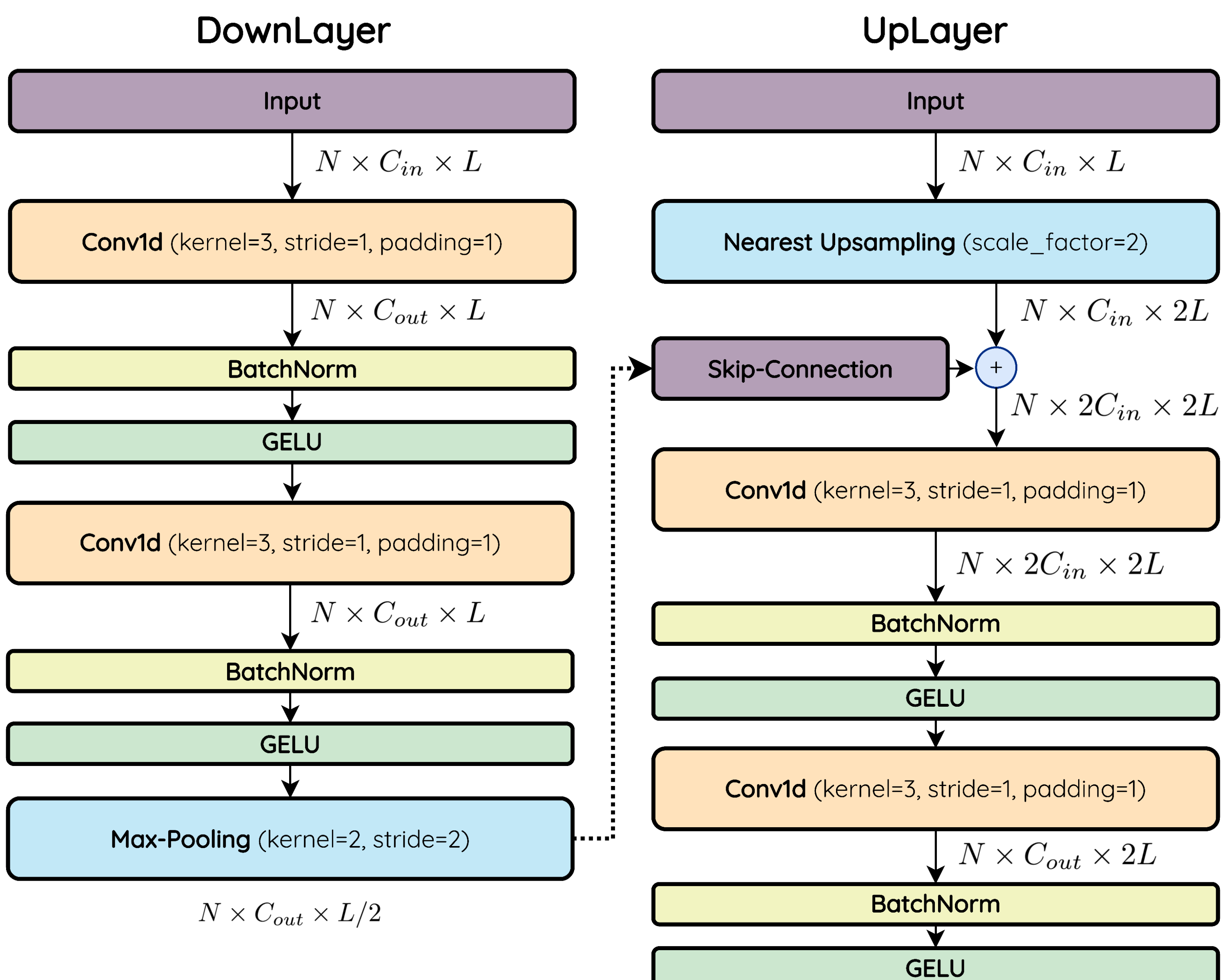


ARCHITECTURE

The 3 objectives of such a model are **accuracy**, **speed** (low inference time) and **size** (low hardware requirements). Although it may seem counterintuitive at first, we decided to use a **U-Net**, a fully convolutional architecture traditionally used for 2D image segmentation.

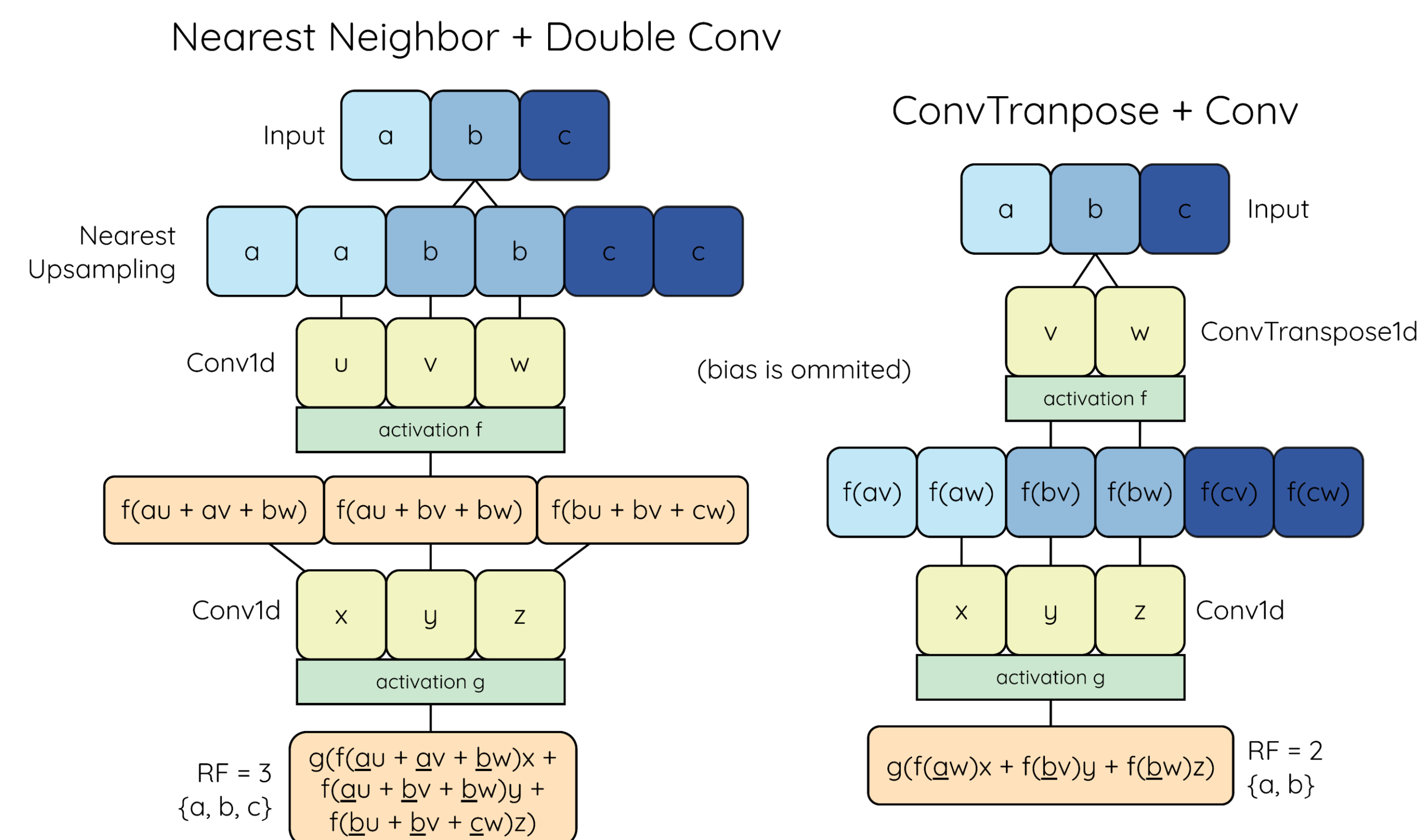


In the encoder part of the model, the input is halved at each step, allowing us to stack more parameters in the middle layers without overloading VRAM. In the upscaling part, the resolution is symmetrically doubled until we reach the same size as the initial input. Skip connections allow us to compensate for the information lost during encoding, and ease the optimization process.



Choosing the right upscaling strategy. There are many ways to upsample in CNNs, the default approach uses *Transposed Convolutions*, it can be opposed to “dummy” upscaling layers such as *Nearest-Neighbor Upsampling*, that do not contain any learnable weights.

In our case, we used nearest upsampling followed by two convolutions, **this requires C_{in}^2 more parameters per layer** (a roughly 25% increase in the number of weights for the decoder part) **but also leads to more representation power**, as illustrated below.



Finally, by controlling the receptive field size (147 at bottleneck), we can capture long term dependencies while keeping a complexity lower than other architectures such as Transformers (quadratic in the input size) or RNNs (temporal dependency), allowing for the parallel processing of large batches. **Our model can predict for arbitrary length sequences without loss in accuracy**, as long as they are not smaller than its receptive field.

TRAINING

We use a training setup similar to Campolina, with samples of shorter length (2000 instead of 6000) and a smaller learning rate for reducing training variance. For the training objective, we use the same 3-components loss function.

$$\mathcal{L} = \alpha \cdot FL + \beta \cdot HL + \gamma \cdot CL$$

FL = Focal Loss; HL = Huber Loss; CL = Consecutive Loss; $\alpha = 5000$; $\beta = 0.05$; $\gamma = 0.1$

The model is trained for 100 epochs on a single A100 GPU, with a batch size of 1024.

EVALUATION

Using the same pipeline for evaluating segmentation quality, we see that our approach outperforms the previous Campolina model in almost all metrics...

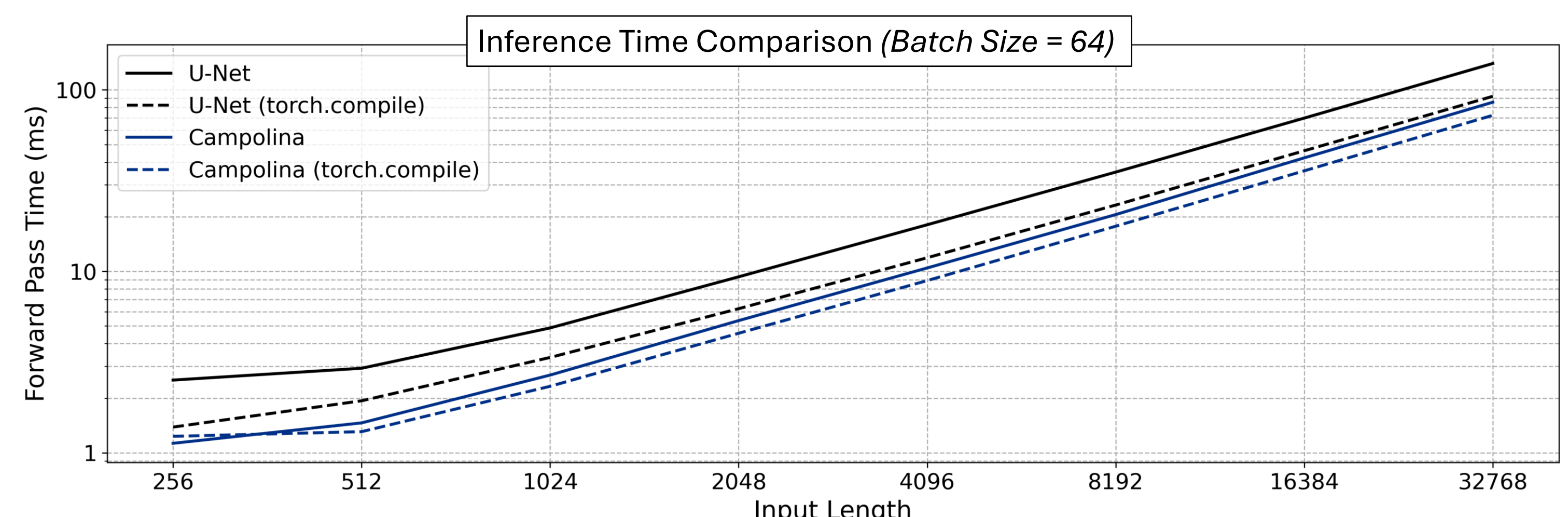
R10.4.1 Zymo D6322	Jaccard		CD ↓	Alignment Ratio			Pearson's r	
	Naive ↑	Expand ↑		Match ↑	Insert ↓	Delete ↓	Full ↑	Match ↑
Scrappie R	0.23	0.38	10.18	0.51	0.13	0.36	0.91	0.94
Campolina	0.54	0.65	5.36	0.67	0.08	0.25	0.96	0.97
U-Net	0.57	0.68	4.79	0.72	0.08	0.20	0.96	0.97

...and performs better when paired with existing real-time frameworks.

Downstream Task: Zymo Multiclass Classification (Classify 7 Microbial Species)						
Algorithm	Segmentation	Accuracy	Precision	Recall	F1	Unclassified Rate
Sigmoni	Scrappie	83.05%	84.53%	82.88%	83.17%	0.00%
	Campolina	92.79%	92.99%	92.79%	92.79%	0.00%
	U-Net	93.90%	94.04%	93.90%	93.90%	0.00%
RawHash2	Scrappie	87.54%	97.50%	87.74%	92.33%	10.69%
	Campolina	90.00%	97.19%	90.24%	93.56%	7.74%
	U-Net	91.56%	97.63%	91.70%	94.55%	6.71%

INFERENCE

Our fully convolutional model scales linearly in the input length. It is larger (2.9M params) than the original Campolina model (1M), but **more than 50% of its weights are located near the bottleneck layer**, where the input tensor has already been halved 3 or 4 times. Finally, we use the PyTorch JIT compiler `torch.compile` to reduce CUDA overhead.



CONCLUSION

We showed that a fully convolutional and lightweight U-Net architecture, traditionally used for image segmentation, is well suited for real-time processing of raw nanopore signal sequences. It outputs high quality segmentation and improves the performance of existing frameworks, while maintaining a fast inference time for both small and large inputs.