Advanced tools and packages for handling tabular data and time series data

Md Atik Ahamed

Brief Outline

- Tabular data
- XGBoost
- Transformer
- TransTab
- RNN
- MambaTab
- PatchTST
- iTransformer
- TimeMachine

Tabular Data

- Numerous applications in healthcare, finance, industry, etc.
- Heterogeneous data type unlike images
- Numerical/binary/categorical
- May require incremental learning

Difficulties

- Inability of regular methods in incremental setting
- Memory drawbacks of deep learning models
- Extensive pre-processing
- Performance limitation

XGBoost

- Efficient implementation of gradient boosting
- Widely used in competitions and industry
- Fast and scalable
- Handles large dataset

XGBoost

- **Gradient Boosting:** Sequentially builds decision trees to correct previous errors
- **Key Steps:**
    - Start with initial prediction
    - Train new trees on residuals (errors)
    - Minimize the loss
    - Repeat for several boosting rounds

Installation

```
pip install xgboost
```

XGBoost

```
from xgboost import XGBClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import
train_test_split
```

XGBoost

```
data = load_iris()

X_train, X_test, y_train, y_test =
train_test_split(data['data'], data['target'],
test_size=0.2, random_state=42)
```

XGBoost

```
bst = XGBClassifier(n_estimators=2,
max_depth=2, learning_rate=1,
objective='multi:softmax', num_class=3)

bst.fit(X_train, y_train)

preds = bst.predict(X_test)
```

XGBoost

Advantages:

- Efficient in speed and memory
- Handles missing values
- Parallel computing support (CPU/GPU)

Limitations:

- Complex tuning of hyperparameters
- May overfit with noisy data
- Can be memory-intensive for large datasets
- Not suitable for feature incremental learning

Transformer

Input Hello! My name is

Token

Sequence

Hello	!	My	Name	is
1	2	3	4	5

Transformer

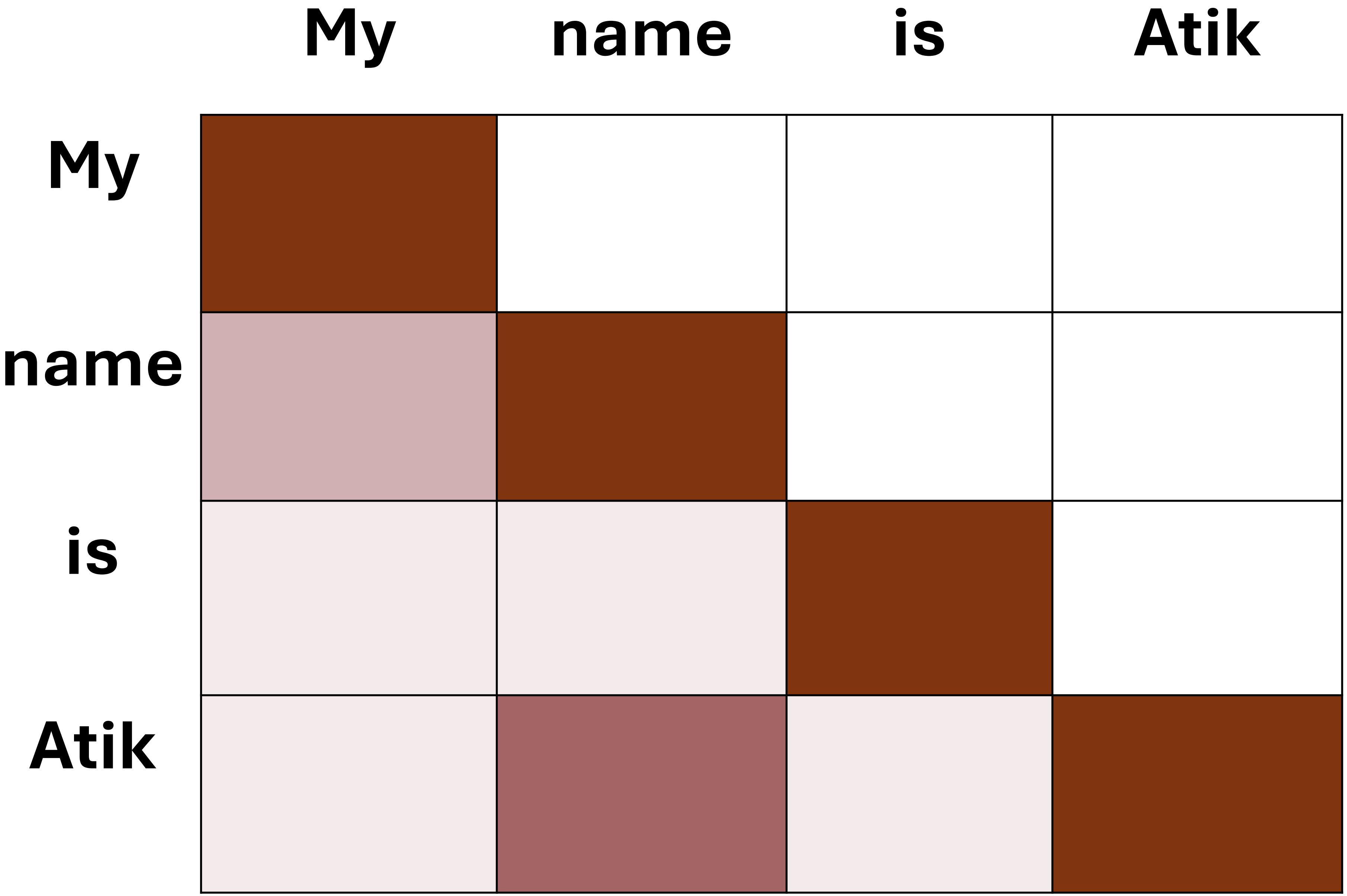
Selective

Individual

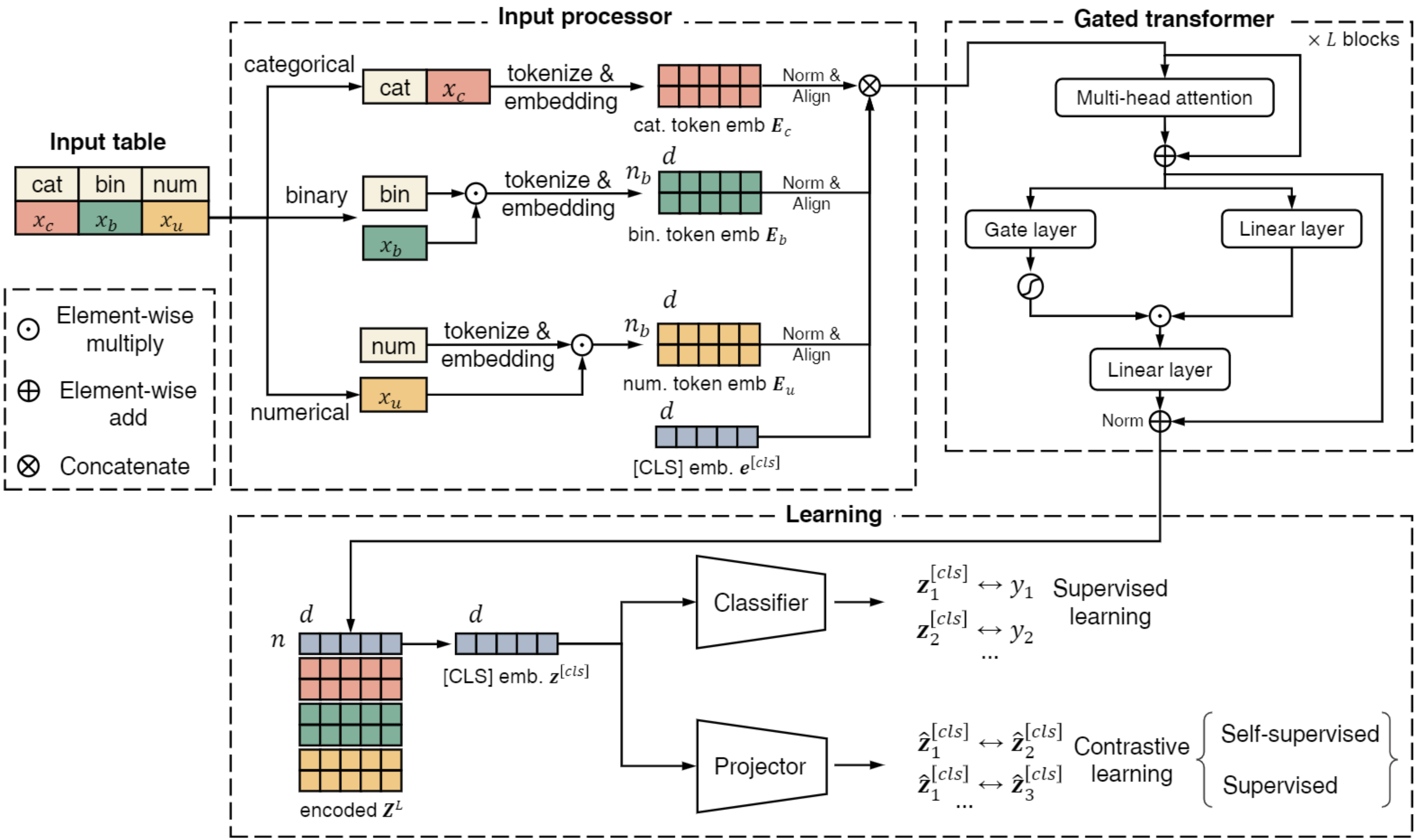
Hello! My name is

Hello! My name is

Transformer



TransTab



Input table
cat
bin
num
x_c
x_b
x_u
Element-wise multiply
Element-wise add
Concatenate
Input processor
categorical
binary
numerical
tokenize & embedding
Norm & Align
cat. token emb E_c
bin. token emb E_b
num. token emb E_u
d
n_b
[CLS] emb. $e^{[cls]}$
Gated transformer
$\times L$ blocks
Multi-head attention
Gate layer
Linear layer
Linear layer
Norm
Learning
n
d
encoded Z^L
[CLS] emb. $z^{[cls]}$
Classifier
Projector
$z_1^{[cls]} \leftrightarrow y_1$
$z_2^{[cls]} \leftrightarrow y_2$
...
Supervised learning
$\hat{z}_1^{[cls]} \leftrightarrow \hat{z}_2^{[cls]}$
$\hat{z}_1^{[cls]} \leftrightarrow \hat{z}_3^{[cls]}$
...
Contrastive learning
Self-supervised
Supervised

Installation

```
pip install git+https://github.com/RyanWangZf/transtab.git
```

https://github.com/RyanWangZf/transtab

Usage

```python
import transtab
allset, trainset, valset, testset, cat_cols,
num_cols, bin_cols \
= transtab.load_data('credit-approval')
```

Usage

```
model = transtab.build_classifier(cat_cols,
num_cols, bin_cols)

training_arguments = {
   'num_epoch':5,
   'eval_metric':'val_loss',
   'eval_less_is_better':True,
   'output_dir':'./checkpoint'
 }
```

Usage

```
transtab.train(model, trainset, valset,
**training_arguments)

ypred = transtab.predict(model, df_x)
```

Limitations

- Manual feature identification
- Large number of parameters
- Scalability issue

Transformer

Length L

Sequence

Hello	!	My	Name	is	Atik
1	2	3	4	5	6

New token

.
7

Previously generated token

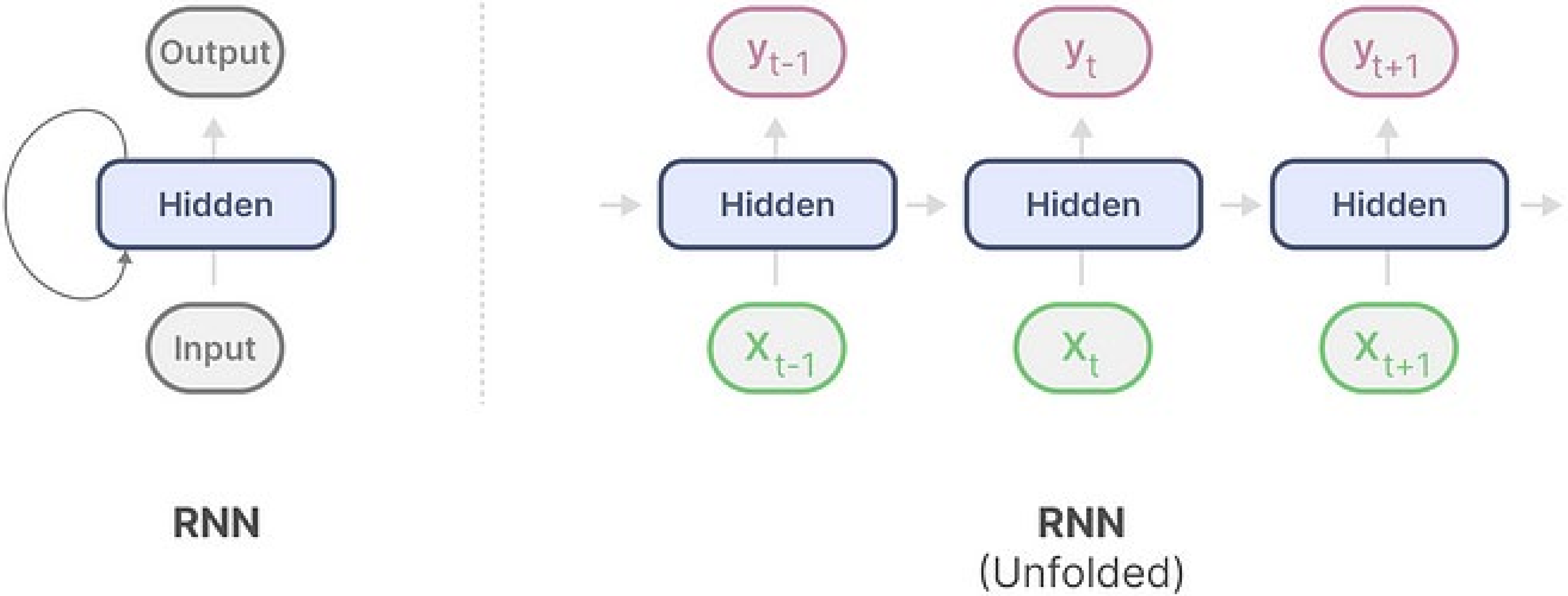
L^2 computations

Transformer

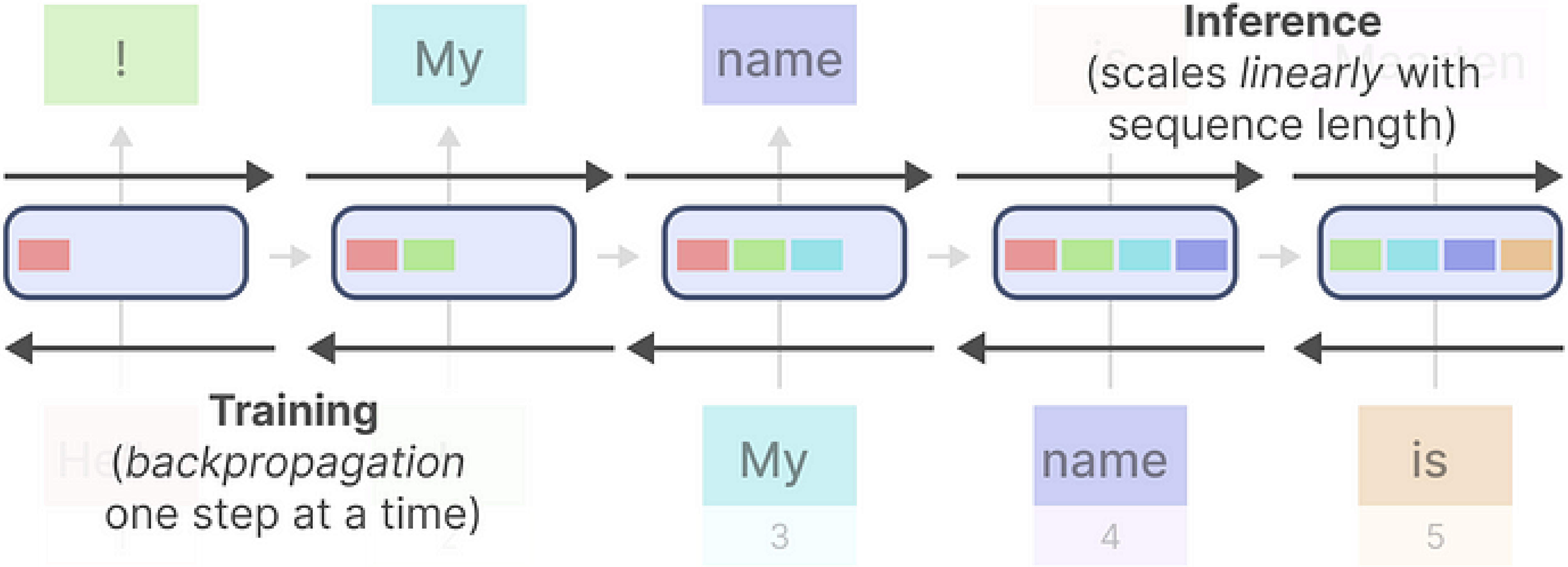
Fast Training

Slow Inference

RNN


Output
Hidden
Input
RNN
y_{t-1}
y_t
y_{t+1}
Hidden
Hidden
Hidden
X_{t-1}
X_t
X_{t+1}
RNN
(Unfolded)

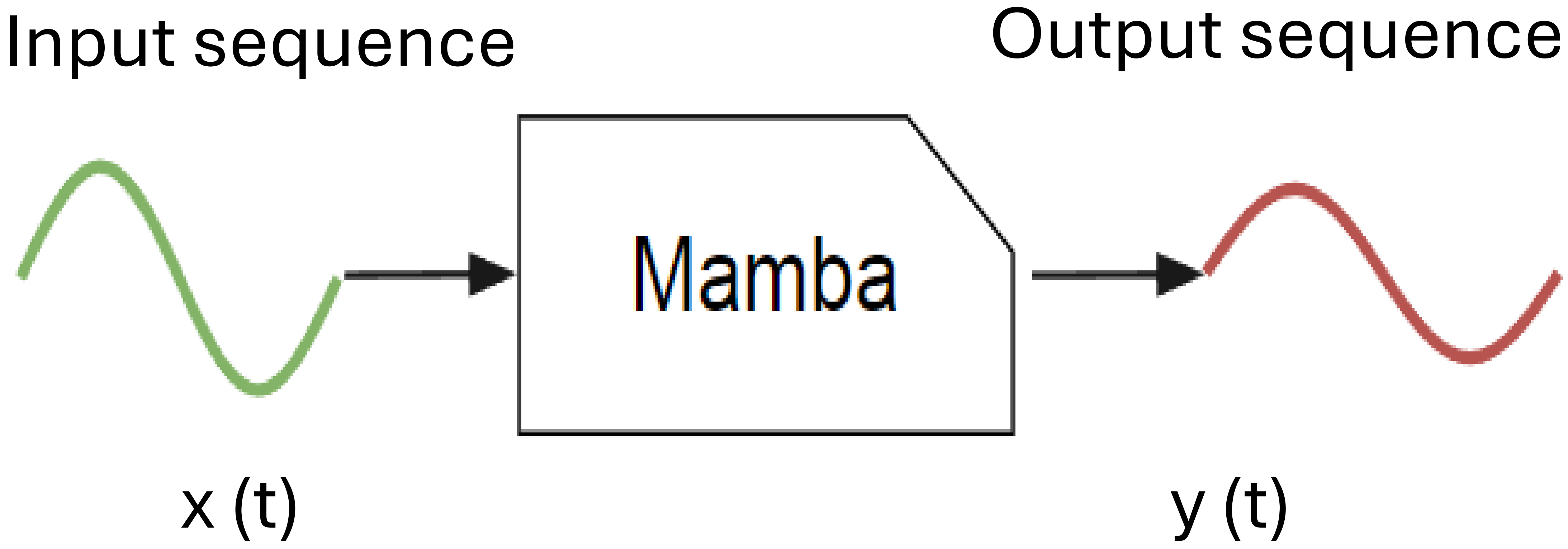
RNN


!
My
name
Inference
(scales *linearly* with sequence length)
Training
(*backpropagation* one step at a time)
My
3
name
4
is
5

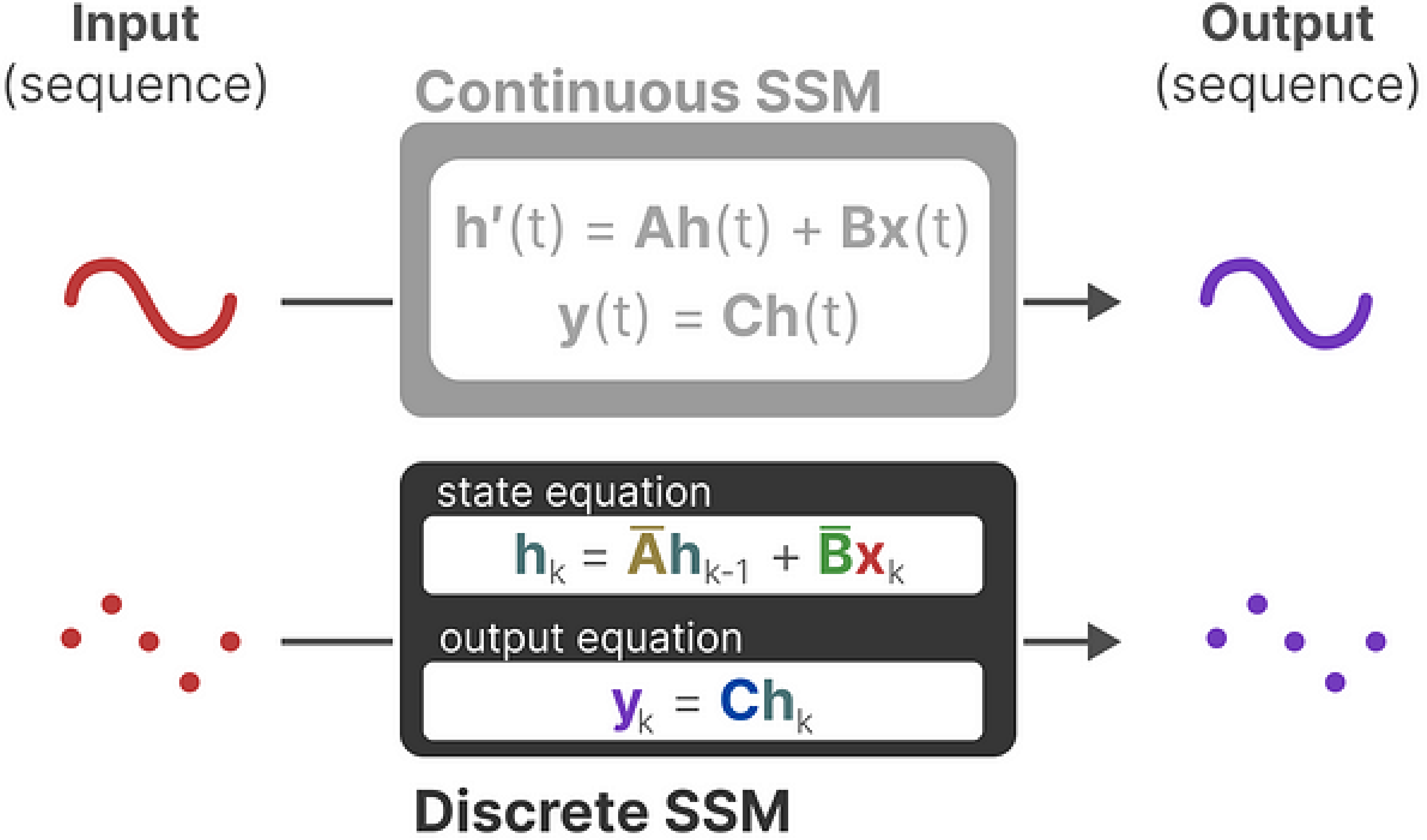
Transformers vs RNNs

	Training	Inference
Transformers	**Fast**	**Slow**
RNNs	**Slow**	**Fast**

Mamba



SSM (State Space Machine)



RNN and SSM

Timestep 0

$$h_0 = \bar{B}x_0$$

$$y_0 = Ch_0$$

Timestep -1 does not exist so Ah_{-1} can be ignored

Timestep 1

$$h_1 = \bar{A}h_0 + \bar{B}x_1$$

$$y_1 = Ch_1$$

State of **previous** timestep

State of **current** timestep

Timestep 2

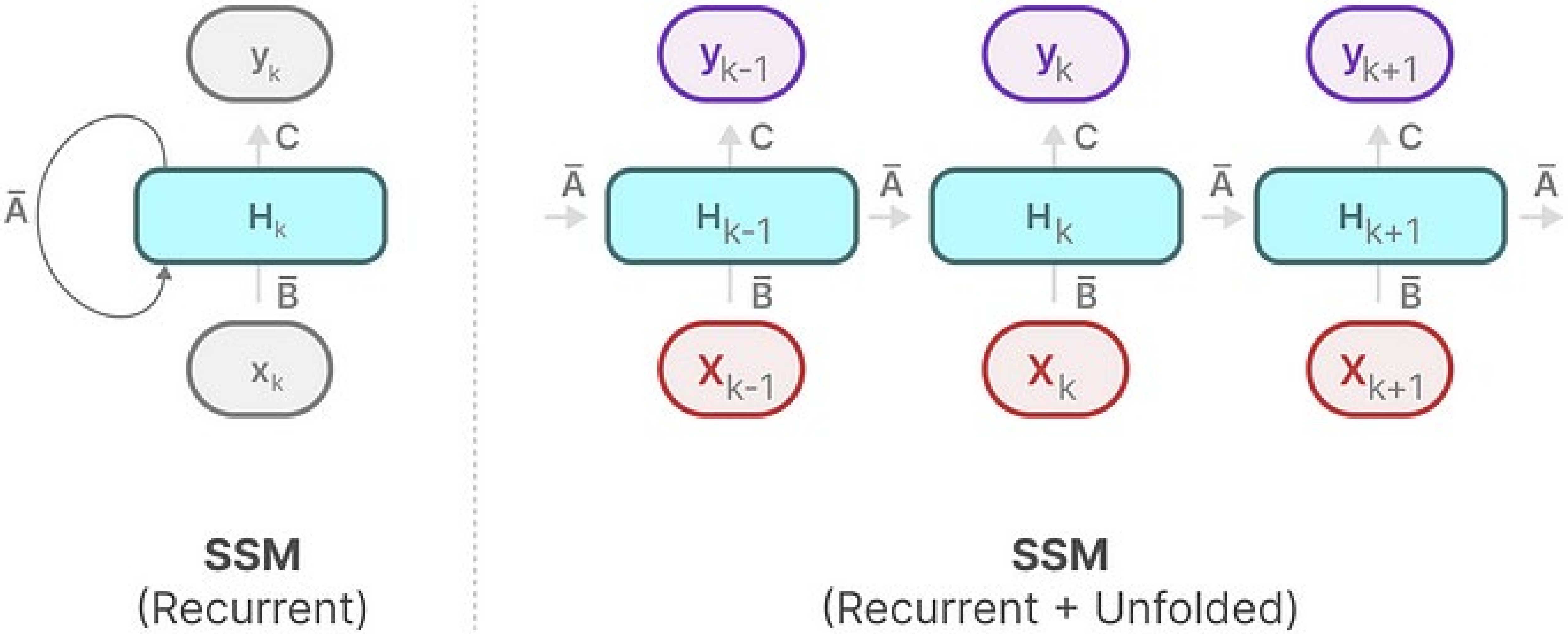
$$h_2 = \bar{A}h_1 + \bar{B}x_2$$

$$y_2 = Ch_2$$

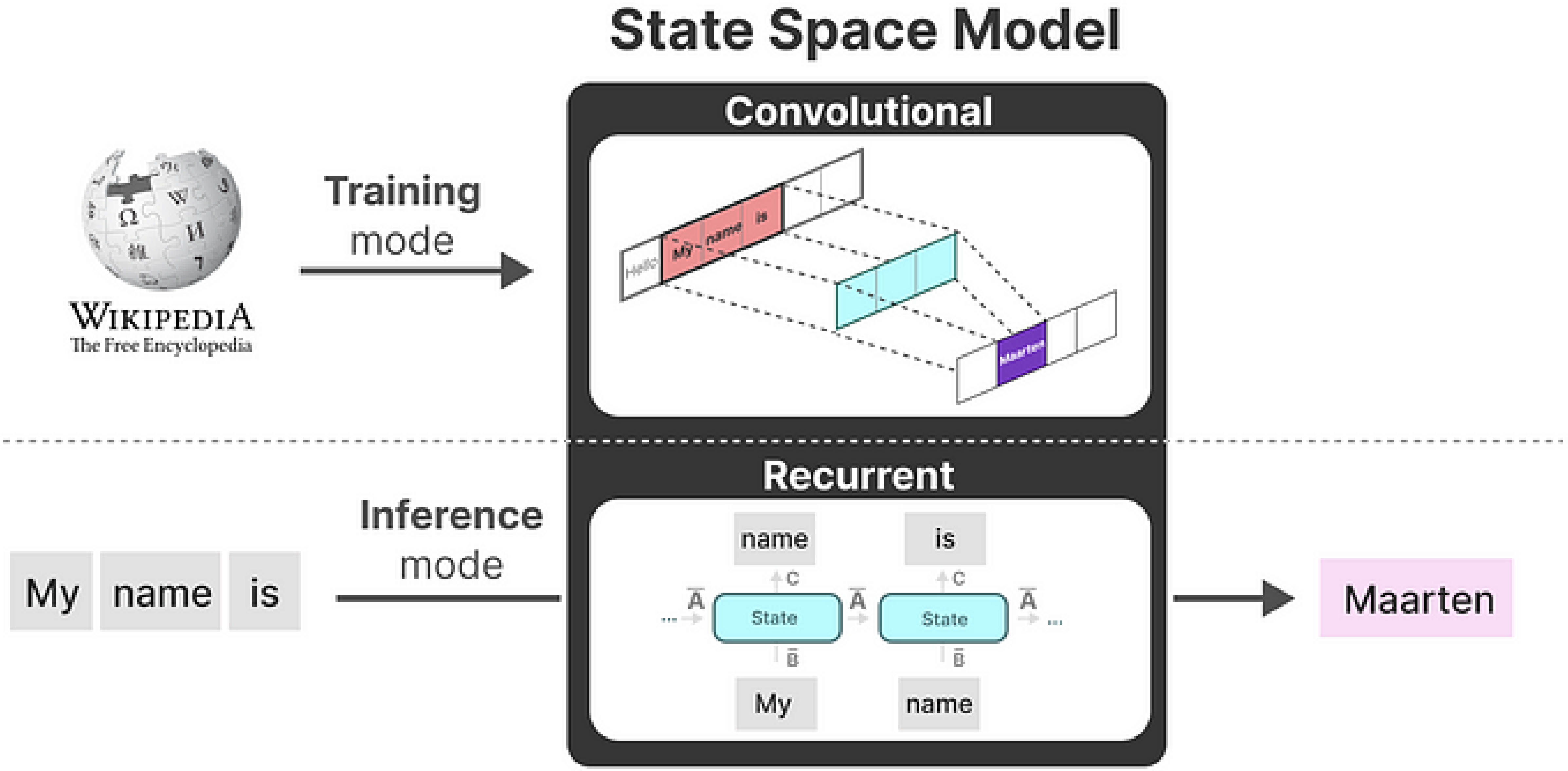
State of **previous** timestep

State of **current** timestep

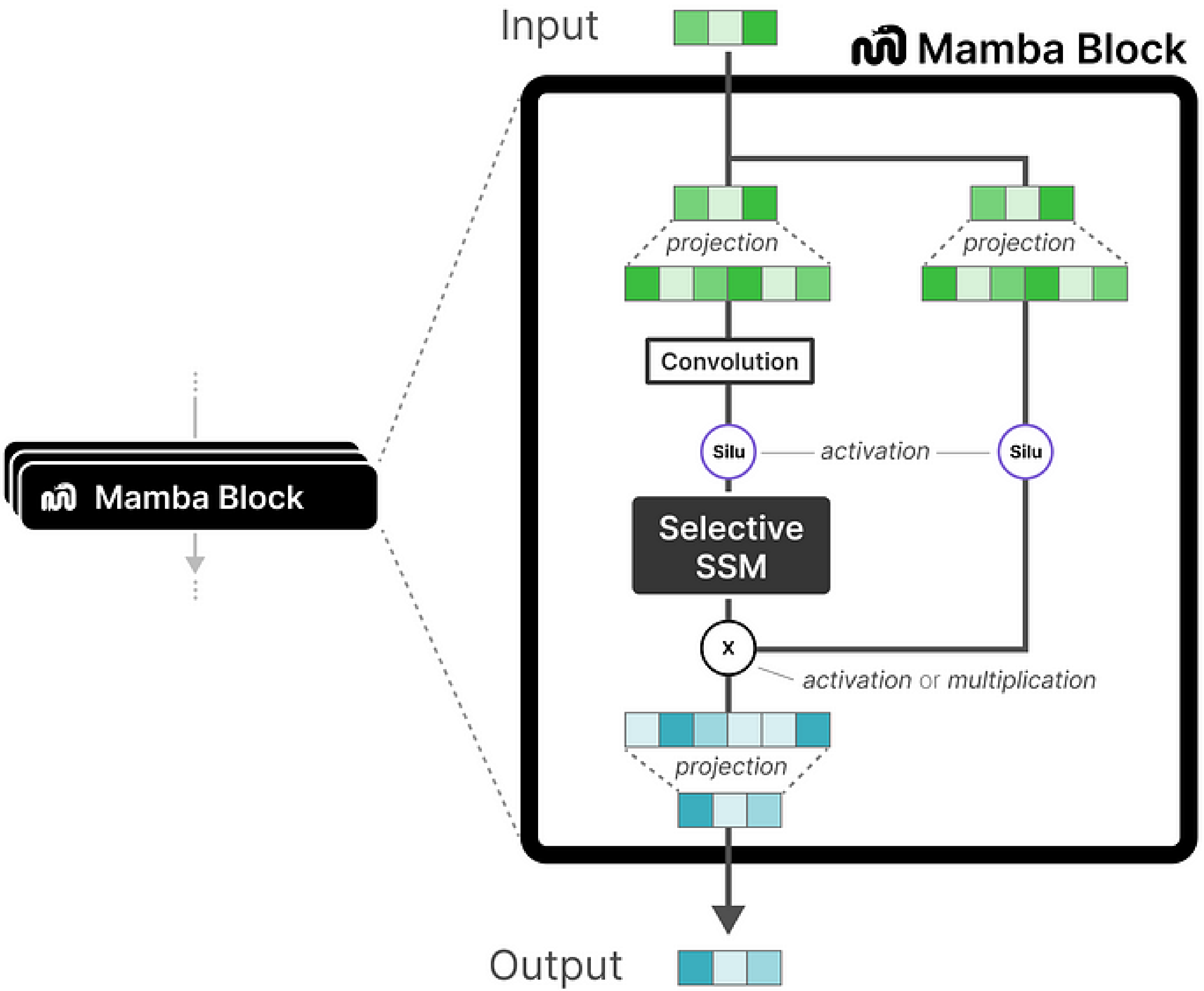
RNN and SSM


y_k
C
$\bar{A}$
H_k
$\bar{B}$
x_k
SSM
(Recurrent)
y_{k-1}
y_k
y_{k+1}
C
C
C
$\bar{A}$
H_{k-1}
$\bar{A}$
H_k
$\bar{A}$
H_{k+1}
$\bar{A}$
$\bar{B}$
$\bar{B}$
$\bar{B}$
x_{k-1}
x_k
x_{k+1}
SSM
(Recurrent + Unfolded)

Training and Inference

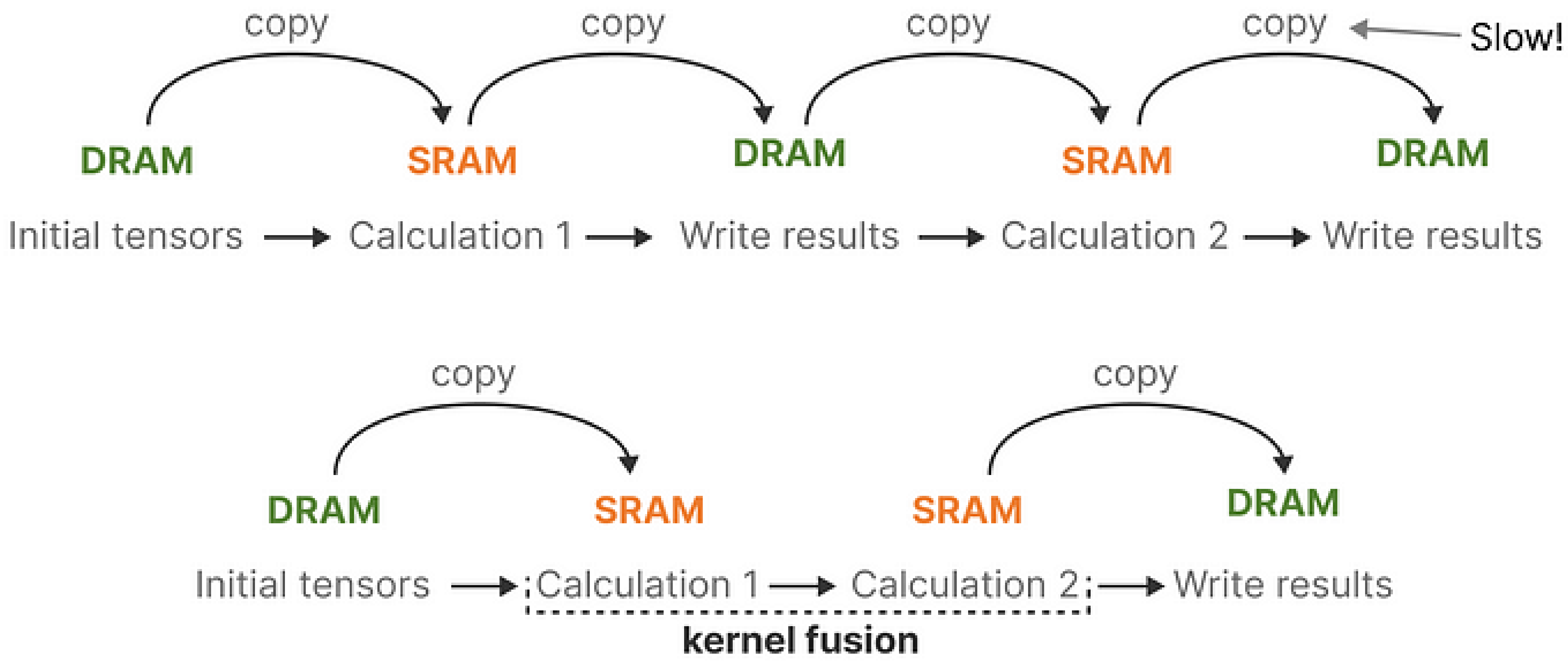


Mamba

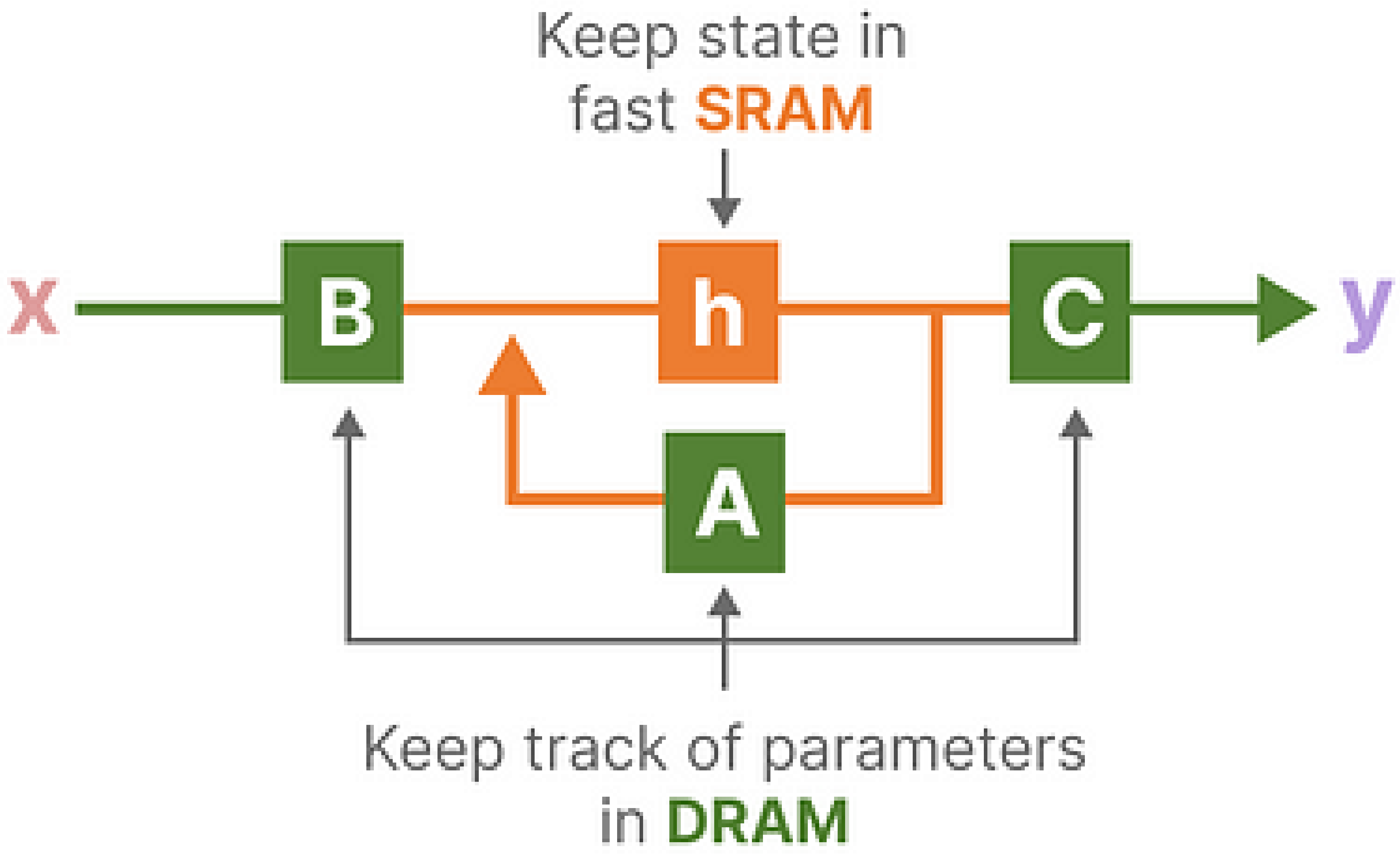


Mamba Block
Input
Mamba Block
projection
projection
Convolution
Silu
activation
Silu
Selective
SSM
x
activation or multiplication
projection
Output

Efficient Implementation



Efficient Implementation



Transformers vs RNNs vs Mamba

	Training	Inference
Transformers	Fast	Slow
RNNs	Slow	Fast
Mamba	Fast	Fast

Installation

https://github.com/state-spaces/mamba

pip install mamba-ssm

Some requirements

Linux
NVIDIA GPU
PyTorch 1.12+
CUDA 11.6+

https://learn.microsoft.com/en-us/windows/wsl/about

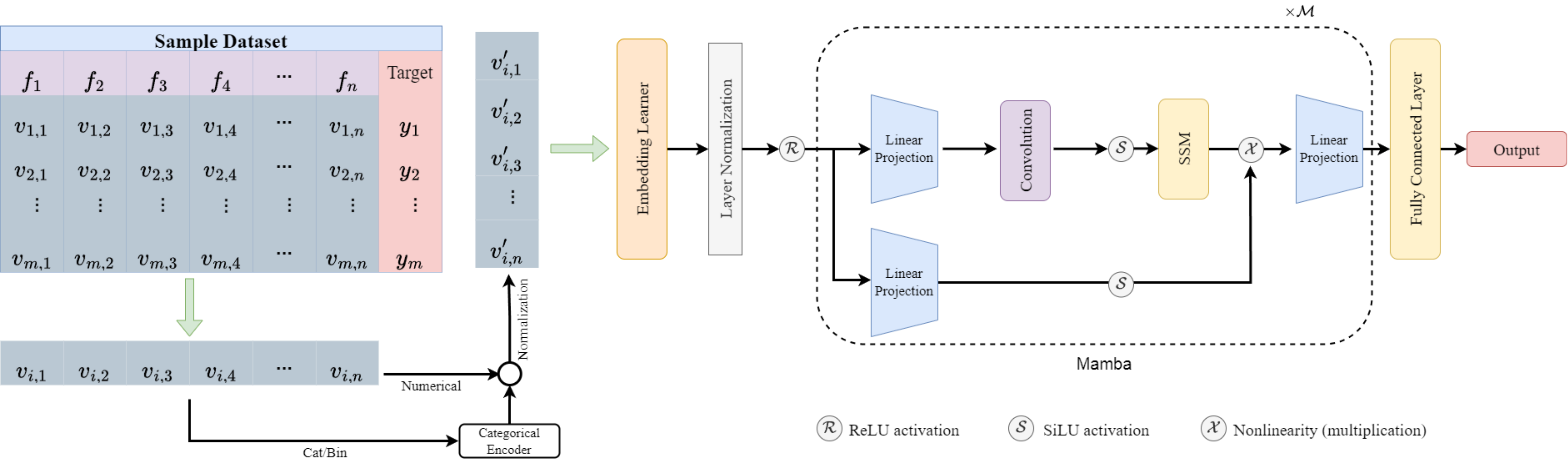
Mamba

```python
import torch
from mamba_ssm import Mamba
batch, length, dim = 2, 64, 16
x = torch.randn(batch, length, dim).to("cuda")
model = Mamba(
    d_model=dim, # Model dimension d_model
    d_state=16,  # SSM state expansion factor
    d_conv=4,    # Local convolution width
    expand=2,    # Block expansion factor
).to("cuda")
y = model(x)
assert y.shape == x.shape
```

MambaTab

- Extremely small model size and number of learning parameters
- Linear scalability
- End-to-end training and inference with minimal data wrangling
- Superior performance
- Adaptable to multiple learning schema

MambaTab


Sample Dataset
Target
Embedding Learner
Layer Normalization
Linear Projection
Convolution
SSM
Linear Projection
Linear Projection
Fully Connected Layer
Output
Mamba
Numerical
Normalization
Categorical Encoder
Cat/Bin
ReLU activation
SiLU activation
Nonlinearity (multiplication)

MambaTab

```
pip install torch==2.1.1 torchvision==0.16.1
pip install causal-conv1d==1.1.1
pip install mamba-ssm
```

MambaTab files

- MambaTab.py
- README.MD
- config.py
- feature_incremental.py
- supervised_mambatab.py
- train_val.py
- utility.py

https://github.com/Atik-Ahamed/MambaTab

```python
config={
    'DATASET_NAME':'credit_approval',
    'SEED':15,
    'BATCH':100,
    'LR':0.0001,
    'EPOCH':1000,
    'REPRESENTATION_LAYER':32,
    'ssl_epochs':100,
    'ssl_corruption':0.5,
    'ssl':False,
    'device':'cuda'
}
```

Data availability

Dataset	URL
credit-g	https://www.openml.org/search?type=data&status=active&id=31
credit-approval	https://archive.ics.uci.edu/ml/datasets/credit+approval
dress-sales	https://www.openml.org/search?type=data&status=active&id=23381
adult	https://www.openml.org/search?type=data&status=active&id=1590
cylinder-bands	https://www.openml.org/search?type=data&status=active&id=6332
blastchar	https://www.kaggle.com/datasets/blastchar/telco-customer-churn
insurance-co	https://archive.ics.uci.edu/ml/datasets/Insurance+Company+Benchmark+%28COIL+2000%29
1995-income	https://www.kaggle.com/datasets/lodetomasi1995/income-classification

Table 13, TransTab

Dataset format

- Dataset should be in .csv format.
- Header row should be the first row in the .csv file.
- Target column should be the last column in the .csv file.
- Rename the file to data_processed.csv and place it in the datasets/X folder, where X can be `dress`, `cylinder`, etc.

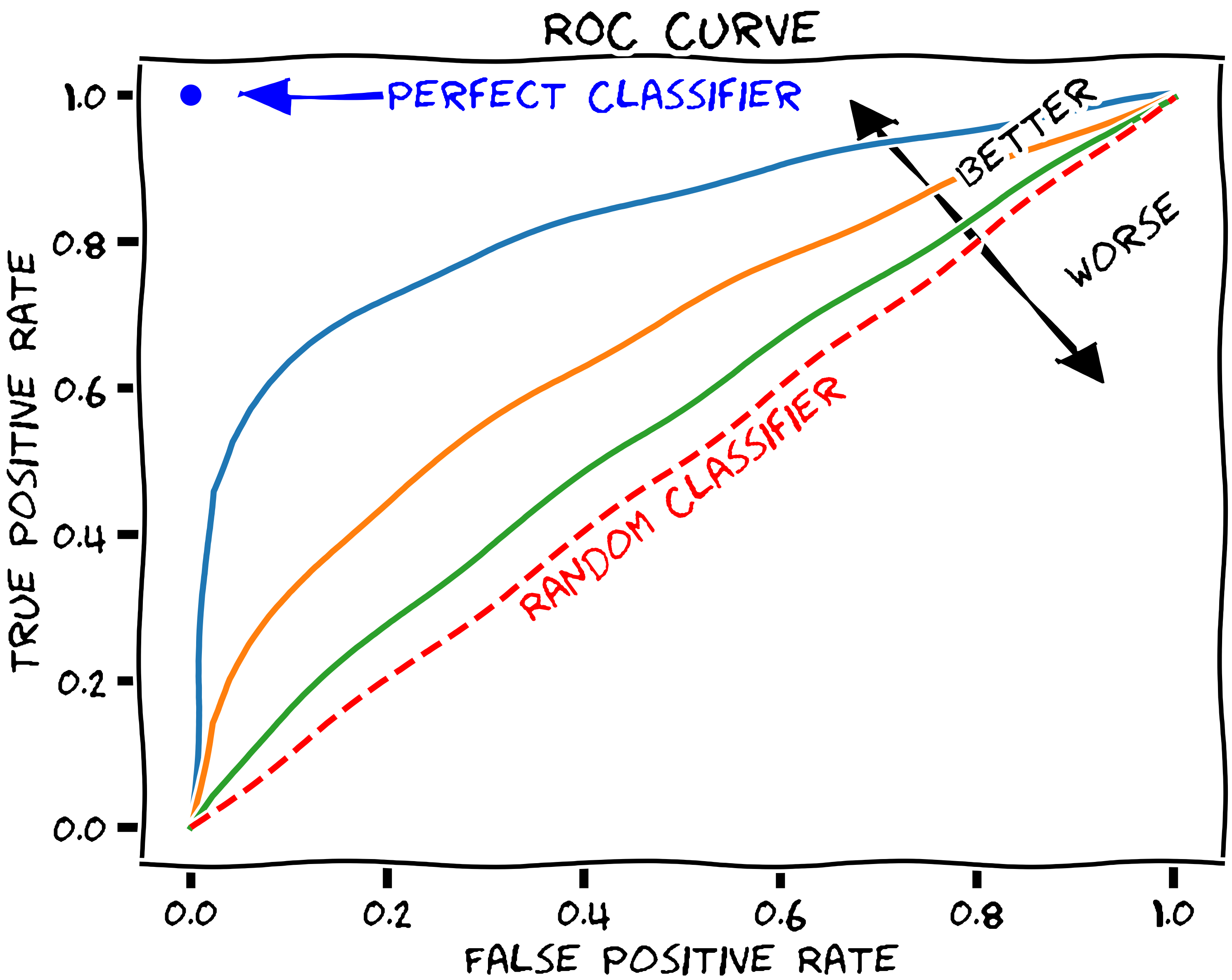
Credit-Approval Dataset

1	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
2	b	30.83	0	u	g	w	v	1.25	t	t	1	f	g	202	0	+
3	a	58.67	4.46	u	g	q	h	3.04	t	t	6	f	g	43	560	+
4	a	24.5	0.5	u	g	q	h	1.5	t	f	0	f	g	280	824	+
5	b	27.83	1.54	u	g	w	v	3.75	t	t	5	t	g	100	3	+

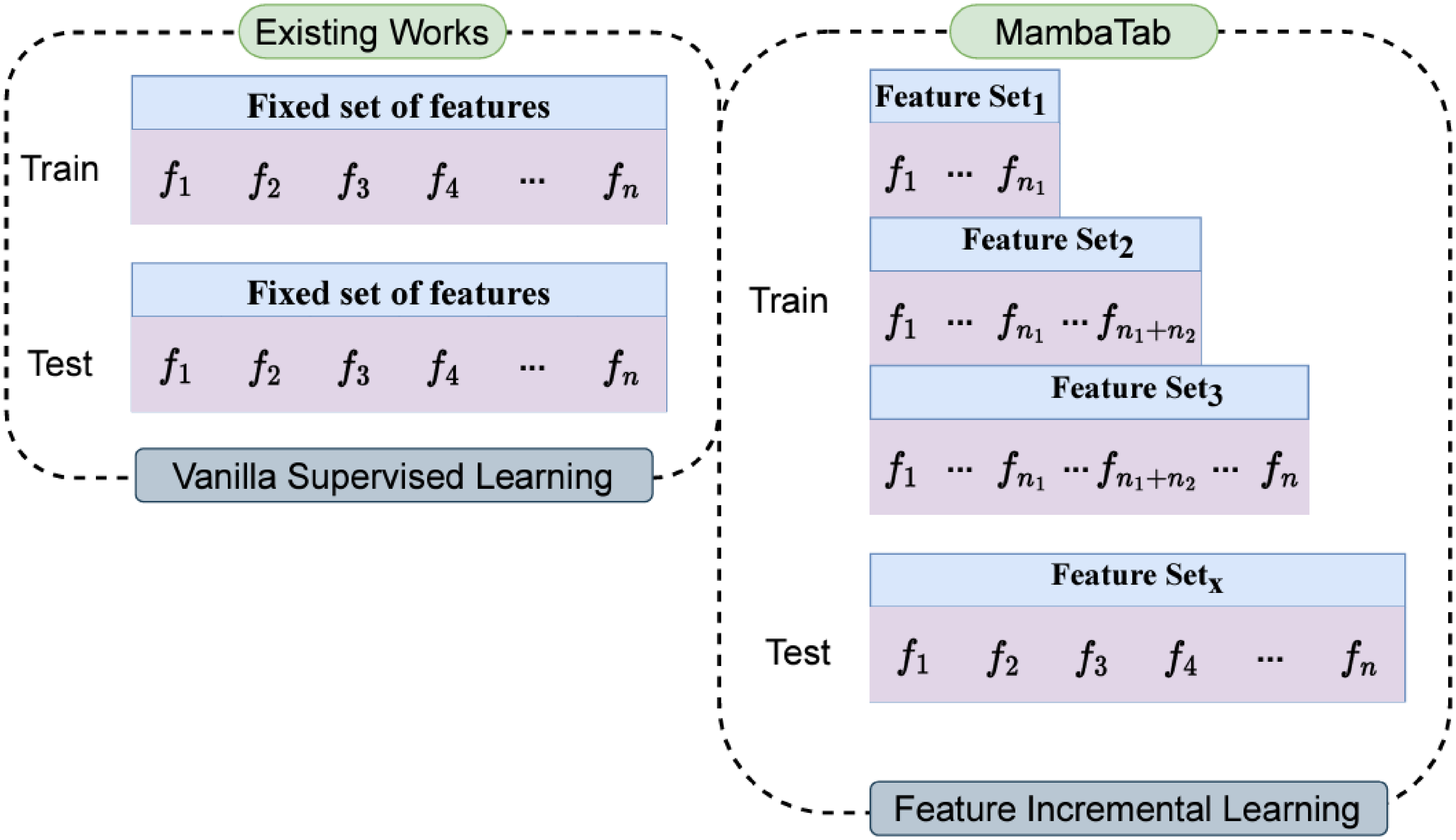
Example output

```
$ python supervised_mambatab.py
Train: (483, 15)
Val: (69, 15)
Test: (138, 15)
  5%|█
AUROC score:  0.9689163295720673
-----------------Complete-----------------
```

AUROC


ROC CURVE
PERFECT CLASSIFIER
BETTER
WORSE
RANDOM CLASSIFIER
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
TRUE POSITIVE RATE
FALSE POSITIVE RATE

Feature Incremental Learning



Self-supervised Learning

1	3	10	5.5	2	1	90	30

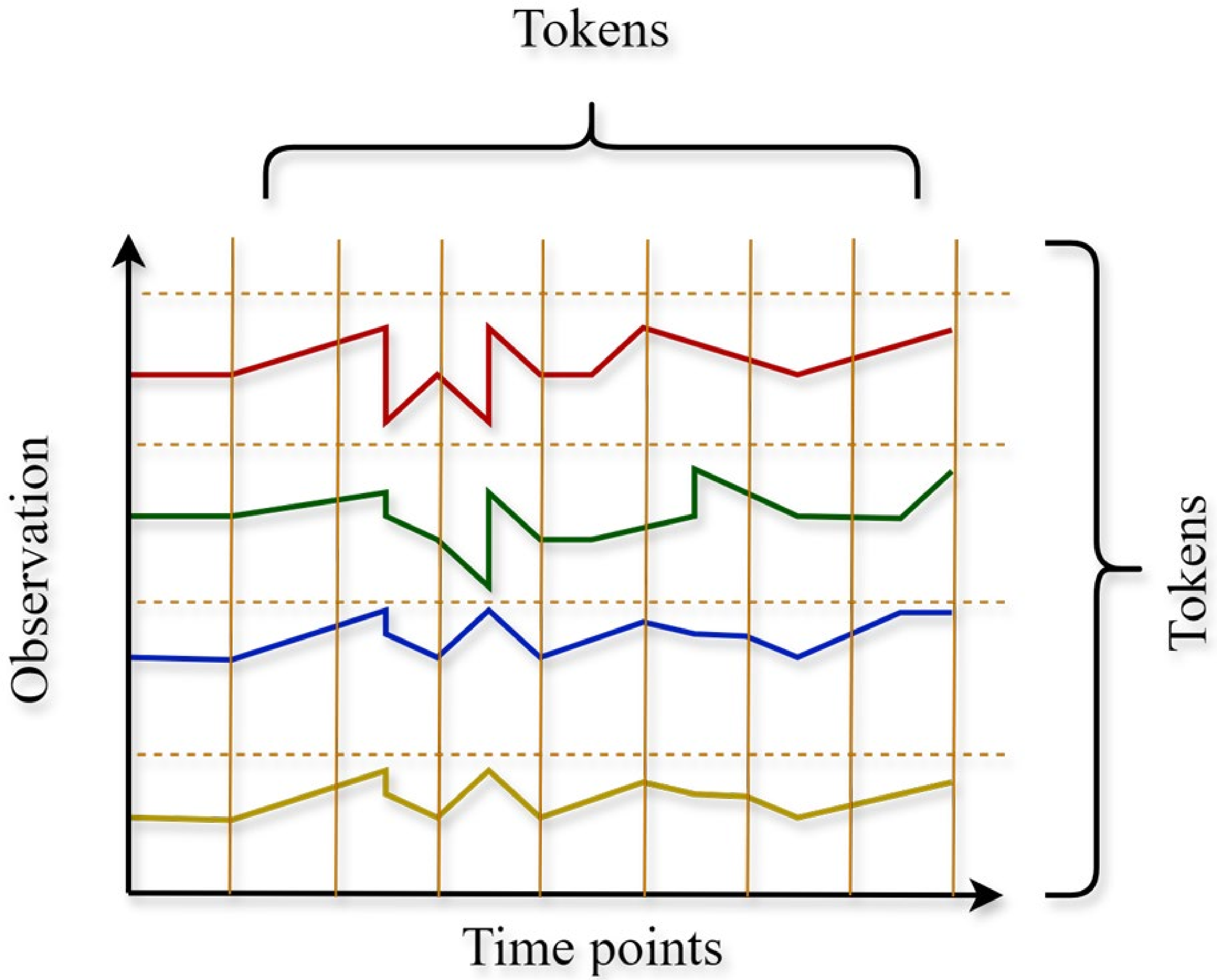
Dropout

1	0	10	0	2	0	90	0

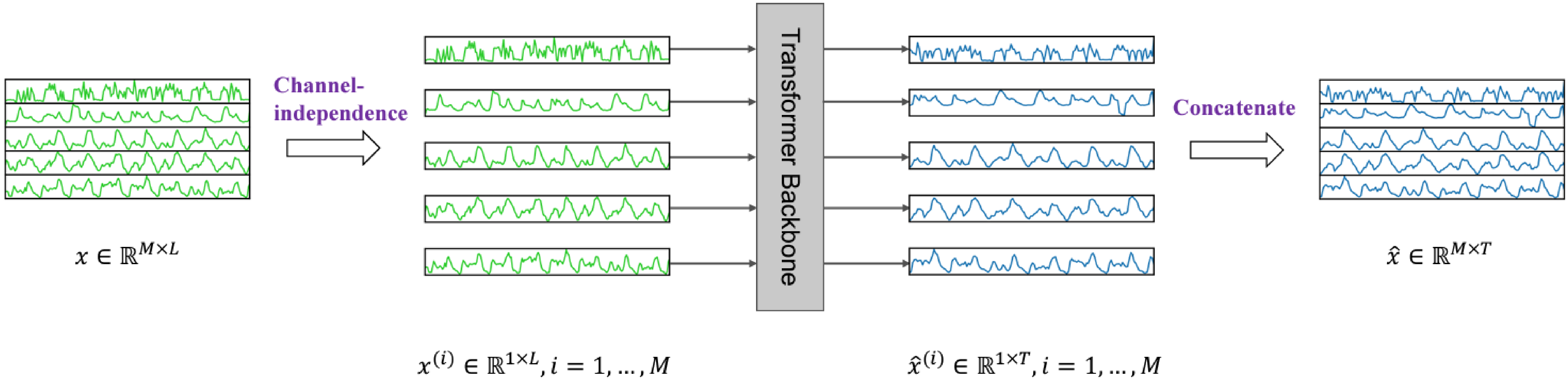
Reconstruction

1	3	10	5.5	2	1	90	30

Time Series

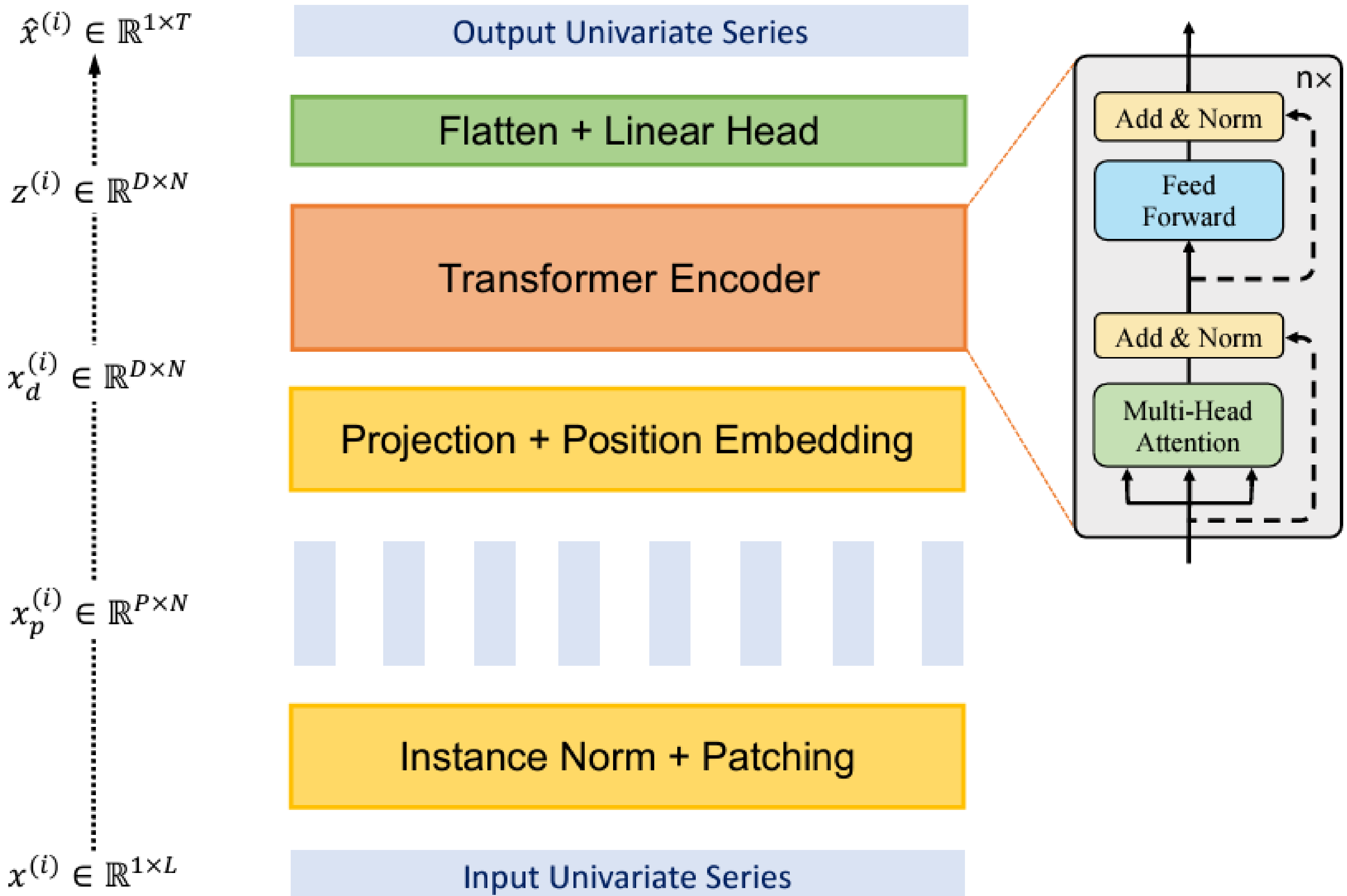


PatchTST



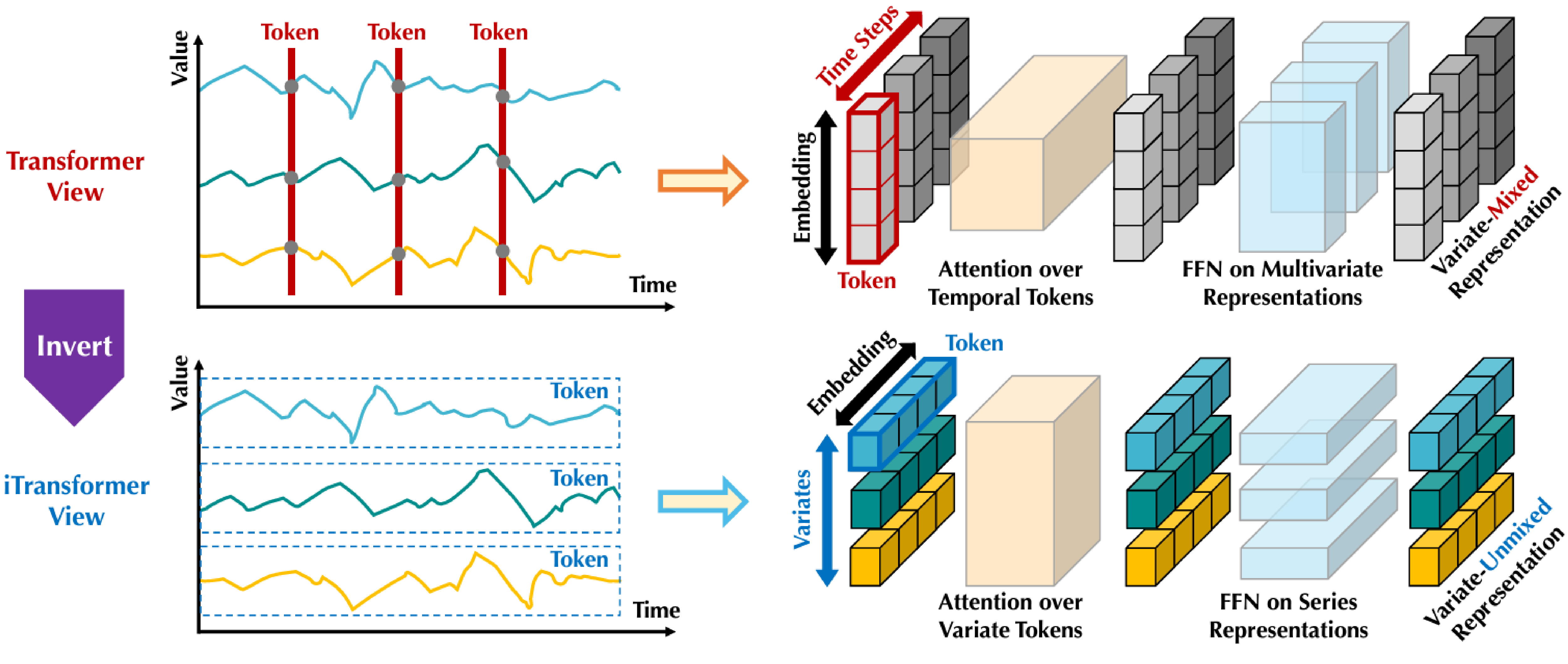
Channel-
independence
Concatenate
Transformer Backbone
$x \in \mathbb{R}^{M \times L}$
$x^{(i)} \in \mathbb{R}^{1 \times L}, i = 1, \dots, M$
$\hat{x}^{(i)} \in \mathbb{R}^{1 \times T}, i = 1, \dots, M$
$\hat{x} \in \mathbb{R}^{M \times T}$

PatchTST

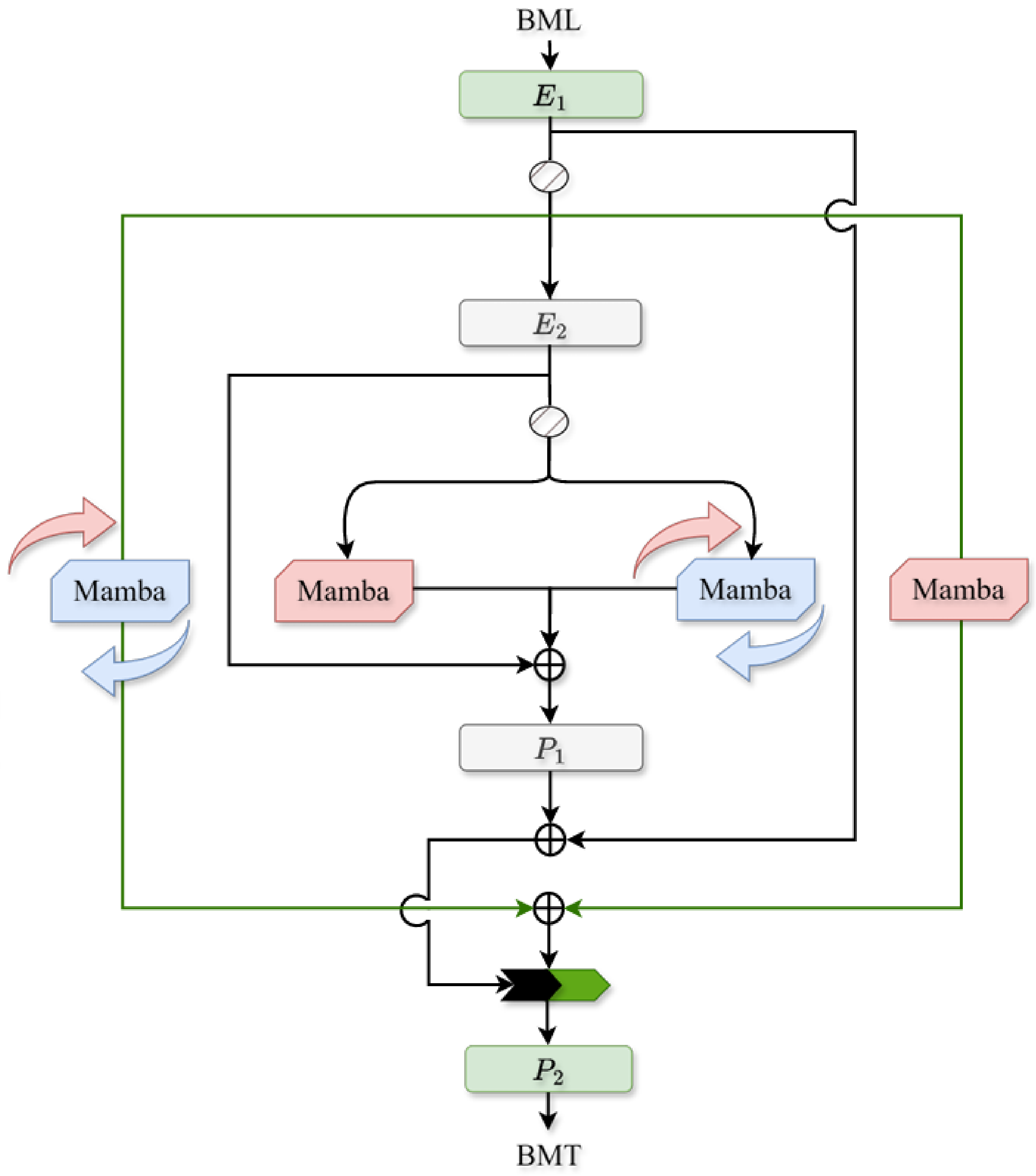


$\hat{x}^{(i)} \in \mathbb{R}^{1\times T}$
$z^{(i)} \in \mathbb{R}^{D\times N}$
$x_d^{(i)} \in \mathbb{R}^{D\times N}$
$x_p^{(i)} \in \mathbb{R}^{P\times N}$
$x^{(i)} \in \mathbb{R}^{1\times L}$
Output Univariate Series
Flatten + Linear Head
Transformer Encoder
Projection + Position Embedding
Instance Norm + Patching
Input Univariate Series
n×
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention

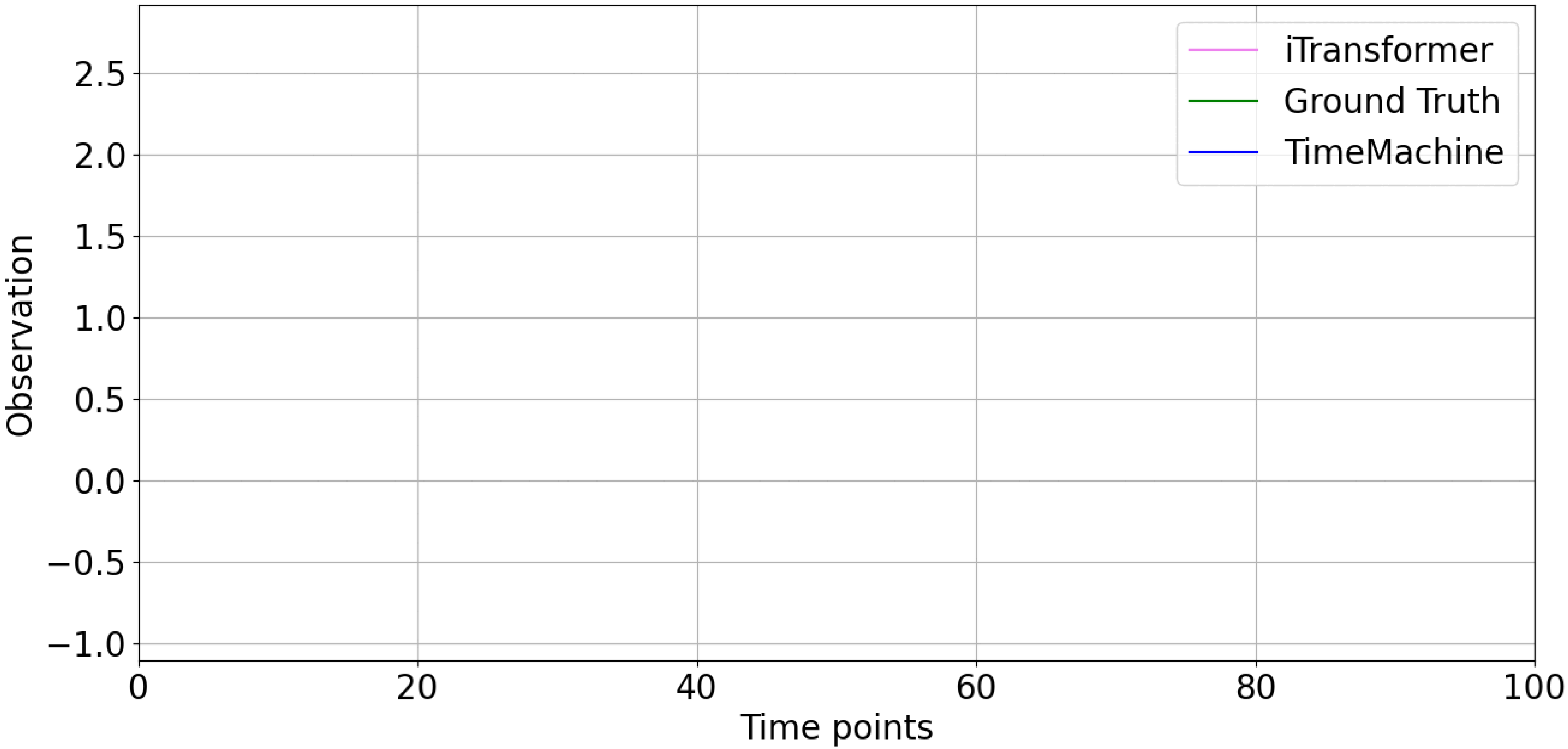
iTransformer



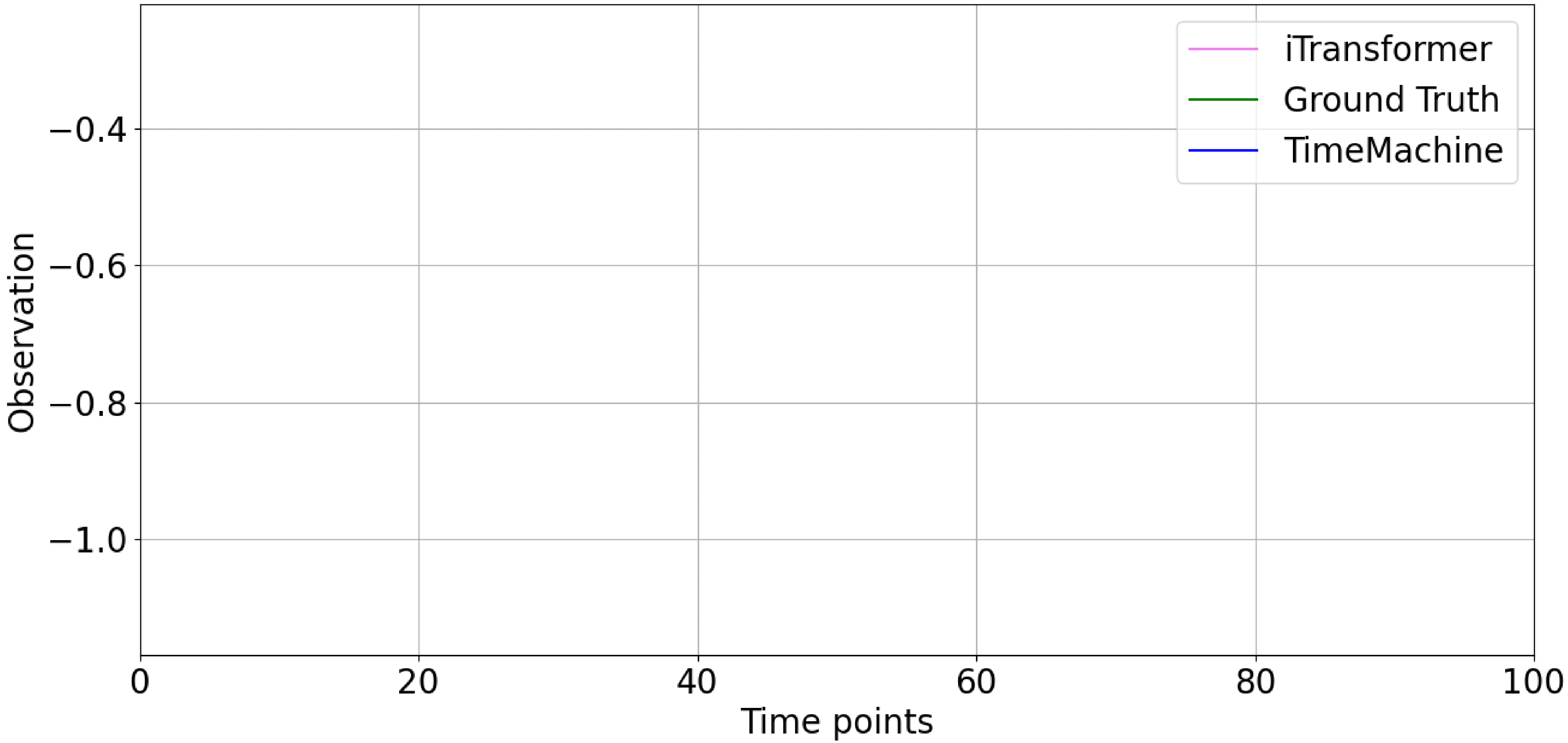
TimeMachine



Qualitative Comparison (Electricity)



Qualitative Comparison (Traffic)



Resources

- https://github.com/thuml/Time-Series-Library
- https://github.com/Atik-Ahamed/TimeMachine
- https://github.com/yuqinie98/PatchTST

References

- Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.
- https://towardsdatascience.com/a-visual-guide-to-mamba-and-state-space-models-8d0d3f7d3ea6
- https://commons.wikimedia.org/wiki/File:Roc-draft-xkcd-style.svg
- Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017).
- Wang, Zifeng, and Jimeng Sun. "Transtab: Learning transferable tabular transformers across tables." Advances in Neural Information Processing Systems 35 (2022): 2902-2915.

References

- Gu, Albert, and Tri Dao. "Mamba: Linear-time sequence modeling with selective state spaces." arXiv preprint arXiv:2312.00752 (2023).
- Ahamed, Md Atik, and Qiang Cheng. "Mambatab: A simple yet effective approach for handling tabular data." arXiv preprint arXiv:2401.08867 (2024).
- Nie, Yuqi, et al. "A time series is worth 64 words: Long-term forecasting with transformers." arXiv preprint arXiv:2211.14730 (2022).
- Liu, Yong, et al. "itransformer: Inverted transformers are effective for time series forecasting." arXiv preprint arXiv:2310.06625 (2023).
- Ahamed, Md Atik, and Qiang Cheng. "Timemachine: A time series is worth 4 mambas for long-term forecasting." arXiv preprint arXiv:2403.09898 (2024).

Thank you