Lecture 8 - Time Series Modelling

Advanced Machine Learning

Miroslav Čepek, Zdeněk Buk, Rodrigo Da Silva Alves, Vojtěch Rybář, Petr Šimánek

FIT CTU

13. 4. 2023

Agenda

- Classical Approaches & Fourier transform
- Recurrent Neural Networks
- Convolutional Networks for Time Series
- Transformers for Time Series

- Classification for whole signal or sections
- Anomaly detection
- Forecasting

Signal Matching - Cross-Correlation

- How similar two signals are given the translation.
- How well signal matches given pattern.

$$(f \star g)(\tau) = \int_{-\infty}^{+\infty} f(t)g(t+\tau)dt$$

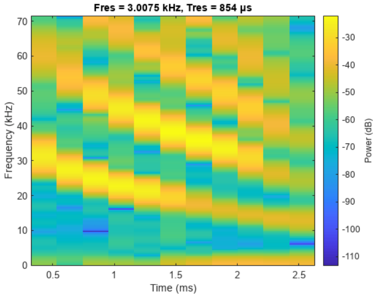
- Slide a pattern - in this context called window - across signal and identify spots where the window and signal matches the best.
- Translation invariance.
- *Where are the heart beats in EEG signal?*

Signal Matching - Dynamic Time Warping

- Similarity between signals varying in speed.
- Signals are expected to start at the same time and are expected to have the same amplitude (values).
- Limited invariance to translation.
- Algorithm tries to match individual samples from both signals - where in either signal, I have to add/remove/modify samples to get the complete match?
- The idea is similar as in Levenshtein's Edit Distance or Needleman-Wunsch Algorithm for matching sequences.
- *Is this signal a heart beat?*

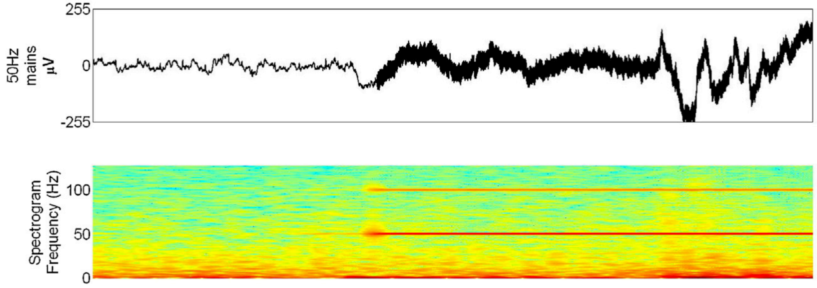
Fourier Transform Reminder

- The math transformation between time and frequency domain.
- Approximates a segment of a time signal with serie of sine/cosine waves with defined frequencies and varying amplitudes.
- Estimated frequency amplitudes and phase shifts can be used as latent representation of the signal.



Matlab Documentation

Fourier Transform Reminder (2)

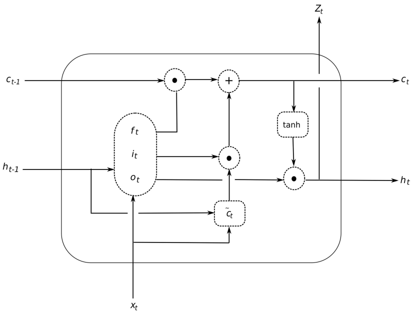


- PRO: Fast and Efficient algorithms to compute.
- CONS: Time locality vs frequency resolution tradeoff.

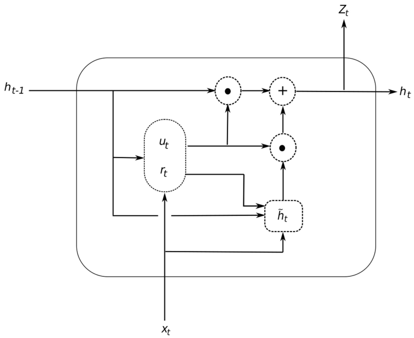
Shayan Motamedi-Fakhr et al.: Signal processing techniques applied to human sleep EEG signals—A review

RNN - Usual Suspects

- Long Short-Term Memory(LSTM)
- Gated Recurrent Units (GRU)



LSTM Cell

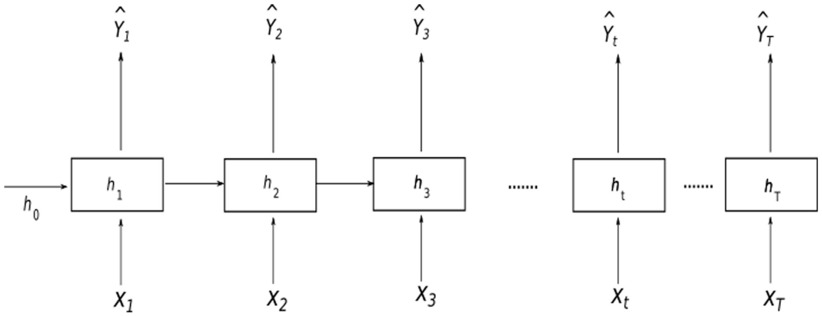


GRU Cell

Recurrent Neural Networks for Time Series Forecasting: Current status and future directions

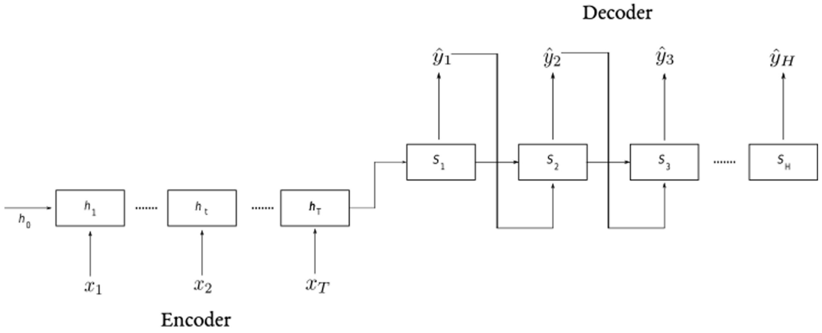
RNN Stacked Architecture

- Feed samples (events) into RNN and get back predictions as $\hat{Y}_i$. The predictions can be either for every sample or at the end.



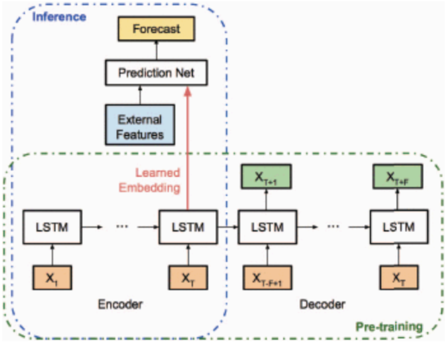
Sequence to Sequence Approach

- Variation on Sequence to Sequence model known from NLP field.
- Used for forecasting as well for sample-wise classification.

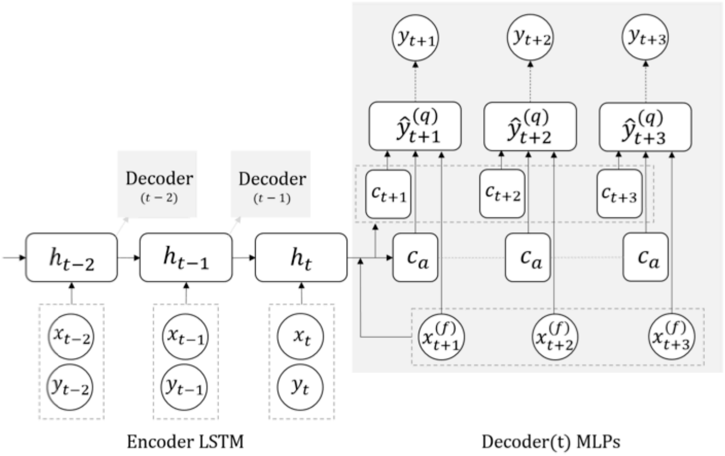


Sequence to Sequence model as Feature Extractor

- First train Sequence to Sequence model to correctly predict the future signal.
- The second step is to take frozen encoder, get the sequence embeddings and train the prediction (fully connected) network.
- The prediction network has additional inputs.



Multi-Quantile Recurrent Forecaster - Variation on Sequence to Sequence Model



A Multi-Horizon Quantile Recurrent Forecaster

Clockwork RNN

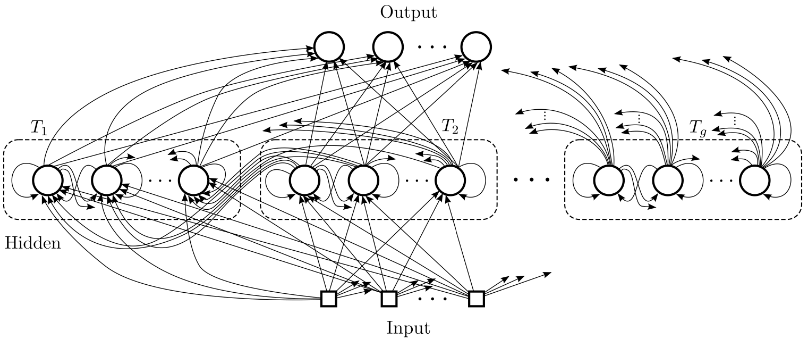


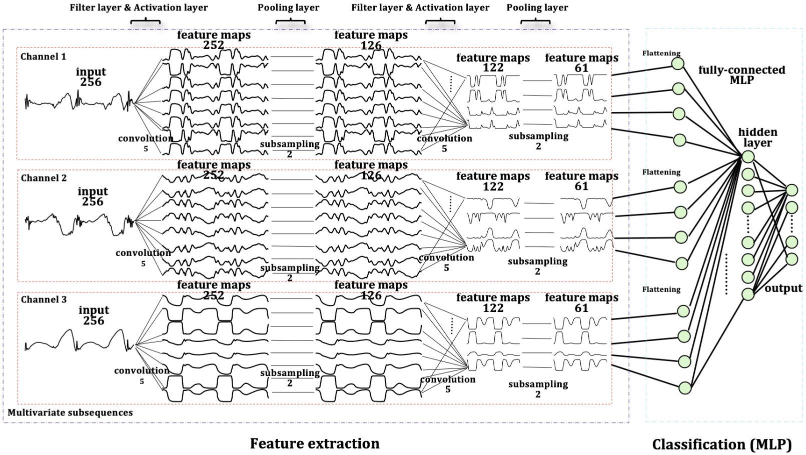
Figure 1. CW-RNN architecture is similar to a simple RNN with an input, output and hidden layer. The hidden layer is partitioned into g modules each with its own clock rate. Within each module the neurons are fully interconnected. Neurons in faster module i are connected to neurons in a slower module j only if a clock period $T_i < T_j$.

A Clockwork RNN

Convolution for Time Series

- Convolution as feature extractor in time series.
- Use 1-D convolution for univariate signal.
- Each convolution filter extract particular type of feature.
- The head on top of extractor can perform any task - classification, regression, anomaly detection or forecasting.

Simplest Approach using CNN



Zheng, Yi, et al. "Time series classification using multi-channels deep convolutional neural networks." (2014)

Variable Filter (Hard Attention)

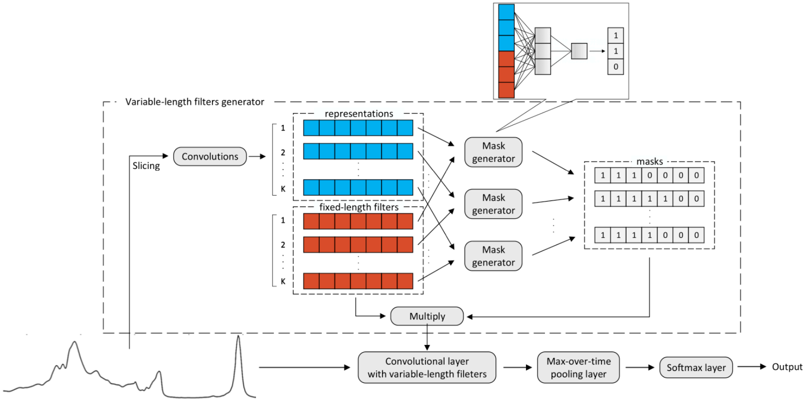


FIGURE 3. General architecture of the Dynamic Multi-Scale Convolutional Neural Network (DMS-CNN). The core of DMS-CNN is variable-length filters generator, which is used to generate a set of filters with different lengths conditioned on the input time series and the randomly initialized filters. Specifically, we first use convolution to obtain the embedded representations of the input time series, which are indicated with the blue lattices. Then, the embedded representations and randomly initialized filters are used to generate a set of filters with different lengths, and we use the red lattices to indicate the randomly initialized filters. Finally, the filters of the conventional CNN are replaced by the variable-length filters to capture features with different time-scales in each time series.

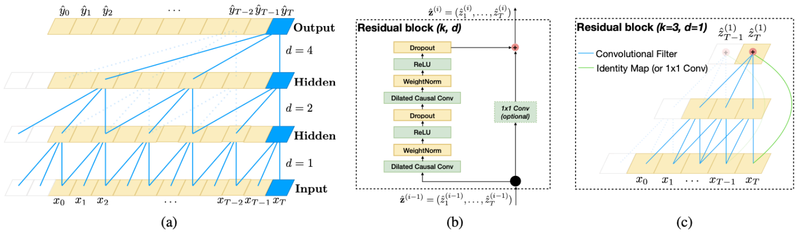
Dynamic Multi-Scale Convolutional Neural Network for Time Series Classification (2020)

Convolution for Forecasting - Temporal Convolutional Networks

- Temporal Convolutional Networks (TCNs) are a special case of 1D convolutional neural network tailored to work well with time series.
- TCNs leverages two principles:
  1. Network outputs the sequence of the same length as input.
  2. Architecturally, no leakage possible from the future into the past.
- Ad $1)$ – TCNs leverage a 1D fully-convolutional network architecture with each hidden and output layer of the same length as the input layer and zero padding.
- Ad $2)$ – TCN uses causal convolutions, convolutions where an output at time t is convolved only with elements from time t and earlier in the previous layer.

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling

Temporal Convolutional Networks Illustration

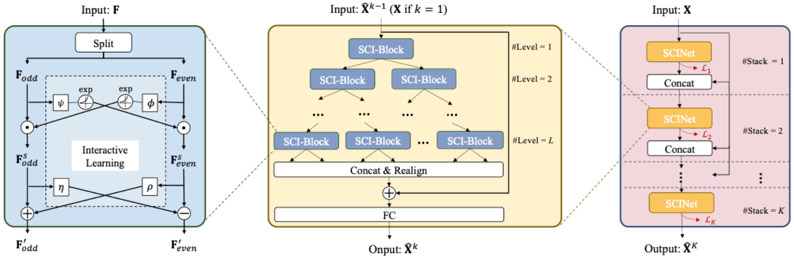


$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 ... x_{T-2} x_{T-1} x_T
(a)
$\hat{\mathbf{z}}^{(i)} = (\hat{z}_1^{(i)}, \dots, \hat{z}_T^{(i)})$
Residual block (k, d)
Dropout
ReLU
WeightNorm
Dilated Causal Conv
Dropout
ReLU
WeightNorm
Dilated Causal Conv
1x1 Conv (optional)
$\hat{\mathbf{z}}^{(i-1)} = (\hat{z}_1^{(i-1)}, \dots, \hat{z}_T^{(i-1)})$
(b)
Residual block (k=3, d=1)
$\hat{z}_{T-1}^{(1)}$ $\hat{z}_T^{(1)}$
Convolutional Filter
Identity Map (or 1x1 Conv)
x_0 x_1 ... x_{T-1} x_T
(c)

Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.

- **TCN = 1D FCN + causal convolutions**
- Almost reuse 30 years old Time Delay Neural Network architecture. Only update is to use zero padding to ensure equal sizes of all layers.
- Note that in order to leverage long history, you need **1)** deep network of many layers *or* **2)** large convolution filters and big dilation step.

SCINet - Reimagined Convolution on Time Series



(a) SCI-Block (b) SCINet (c) Stacked SCINet

SCINet: Time Series Modeling and Forecasting with Sample Convolution and Interaction

SCI-Block

- Splits signal into odd/even samples – coarser temporal resolution but preserve most information of the original sequence.
- Use separate convolution kernels to extract independently features form odd/even sub-sequences.
- Combine features of odd/even subsequences:
  - $F^s_{odd} = F_{odd} \bigodot \exp \phi(F_{even})$
  - $F^s_{even} = F_{even} \bigodot \exp \psi(F_{odd})$
- The F^s_{odd} and F^s_{even} are then projected into output feature space as element-wise multiplication of the original half-signal and projected other half to keep important information in the picture.
  - $F'_{odd} = F_{odd} + \rho F^s_{even}$
- This way we increase knowledge horizon of the output half signal, without loosing information.

SCINet

- Each level of SCIBlocks shortens the length of the sequence by half (or increase the reach).
- In this, similar to dilation in TCNs.
- At the end of every SCIBlock the sequence is realigned and concatenated back into a single sequence.
- One can stack multiple SCINets to achieve better forecasting accuracy (and capture more complex intra-dependancies) at cost of more complex architecture, more weights and lengthier training.

SCINet - result Improvement

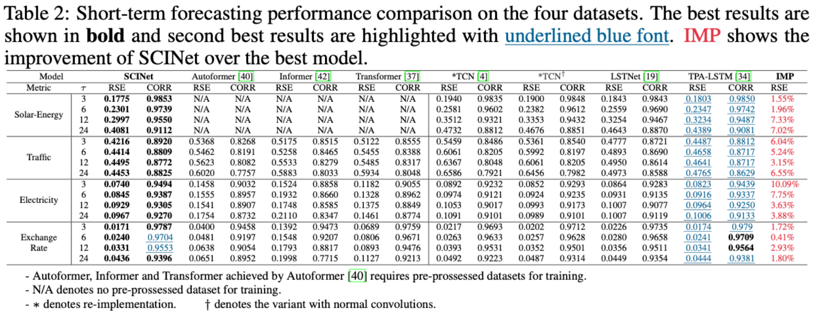


Table 2: Short-term forecasting performance comparison on the four datasets. The best results are shown in **bold** and second best results are highlighted with underlined blue font. IMP shows the improvement of SCINet over the best model.

Model		**SCINet**		Autoformer [40]		Informer [42]		Transformer [37]		*TCN [4]		*TCN†		LSTNet [19]		TPA-LSTM [34]		**IMP**
Metric	τ	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE
Solar-Energy	3	**0.1775**	**0.9853**	N/A	N/A	N/A	N/A	N/A	N/A	0.1940	0.9835	0.1900	0.9848	0.1843	0.9843	0.1803	0.9850	1.55%
	6	**0.2301**	**0.9739**	N/A	N/A	N/A	N/A	N/A	N/A	0.2581	0.9602	0.2382	0.9612	0.2559	0.9690	0.2347	0.9742	1.96%
	12	**0.2997**	**0.9550**	N/A	N/A	N/A	N/A	N/A	N/A	0.3512	0.9321	0.3353	0.9432	0.3254	0.9467	0.3234	0.9487	7.33%
	24	**0.4081**	**0.9112**	N/A	N/A	N/A	N/A	N/A	N/A	0.4732	0.8812	0.4676	0.8851	0.4643	0.8870	0.4389	0.9081	7.02%
Traffic	3	**0.4216**	**0.8920**	0.5368	0.8268	0.5175	0.8515	0.5122	0.8555	0.5459	0.8486	0.5361	0.8540	0.4777	0.8721	0.4487	0.8812	6.04%
	6	**0.4414**	**0.8809**	0.5462	0.8191	0.5258	0.8465	0.5455	0.8388	0.6061	0.8205	0.5992	0.8197	0.4893	0.8690	0.4658	0.8717	5.24%
	12	**0.4495**	**0.8772**	0.5623	0.8082	0.5533	0.8279	0.5485	0.8317	0.6367	0.8048	0.6061	0.8205	0.4950	0.8614	0.4641	0.8717	3.15%
	24	**0.4453**	**0.8825**	0.6020	0.7757	0.5883	0.8033	0.5934	0.8048	0.6586	0.7921	0.6456	0.7982	0.4973	0.8588	0.4765	0.8629	6.55%
Electricity	3	**0.0740**	**0.9494**	0.1458	0.9032	0.1524	0.8858	0.1182	0.9055	0.0892	0.9232	0.0852	0.9293	0.0864	0.9283	0.0823	0.9439	10.09%
	6	**0.0845**	**0.9387**	0.1555	0.8957	0.1932	0.8660	0.1328	0.8962	0.0974	0.9121	0.0924	0.9235	0.0931	0.9135	0.0916	0.9337	7.75%
	12	**0.0929**	**0.9305**	0.1541	0.8907	0.1748	0.8585	0.1375	0.8849	0.1053	0.9017	0.0993	0.9173	0.1007	0.9077	0.0964	0.9250	3.63%
	24	**0.0967**	**0.9270**	0.1754	0.8732	0.2110	0.8347	0.1461	0.8774	0.1091	0.9101	0.0989	0.9101	0.1007	0.9119	0.1006	0.9133	3.88%
Exchange Rate	3	**0.0171**	**0.9787**	0.0400	0.9458	0.1392	0.9473	0.0689	0.9759	0.0217	0.9693	0.0202	0.9712	0.0226	0.9735	0.0174	0.979	1.72%
	6	**0.0240**	0.9704	0.0481	0.9197	0.1548	0.9207	0.0806	0.9671	0.0263	0.9633	0.0257	0.9628	0.0280	0.9658	0.0241	**0.9709**	0.41%
	12	**0.0331**	0.9553	0.0638	0.9054	0.1793	0.8817	0.0893	0.9476	0.0393	0.9531	0.0352	0.9501	0.0356	0.9511	0.0341	**0.9564**	2.93%
	24	**0.0436**	**0.9396**	0.0651	0.8952	0.1998	0.7715	0.1127	0.9213	0.0492	0.9223	0.0487	0.9314	0.0449	0.9354	0.0444	0.9381	1.80%

- Autoformer, Informer and Transformer achieved by Autoformer [40] requires pre-prossessed datasets for training.
- N/A denotes no pre-prossessed dataset for training.
- $*$ denotes re-implementation. † denotes the variant with normal convolutions.

Transformers

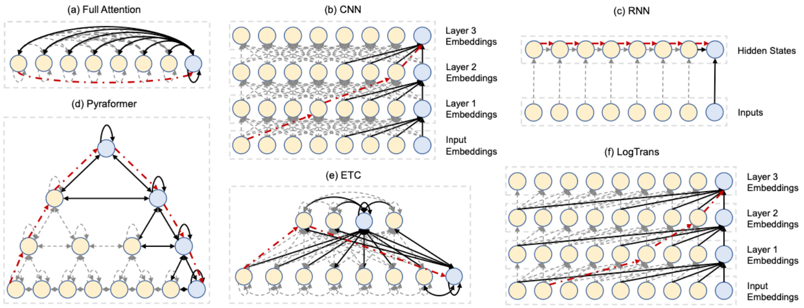
- Transformer is very successful model in NLP - modeling sequences of words.
- Transformers are also faster to infer and train (in terms of simpler parallelisation).
- As usual, the attention mechanism is bottleneck in length of the input serie (limiting the sequence length) – $\mathcal{O}(N^2)$.
- You can use vanilla transformers with some success.
- Typical modifications turn around position encoding
  - Index-Based Positional Encoding
  - Learnable Positional Encoding
  - Timestamp Encoding

Transformers in Time Series: A Survey

Efficient Transformers

- Many efficient Transformers were proposed in NLP as well as in signal processing to to reduce the quadratic complexity of length of the serie.
- Two main approaches
  1. explicitly introduce a sparsity bias into the attention mechanism – LogTrans, Pyraformer
  2. explore the low-rank property of the self-attention matrix – Informer, FEDformer

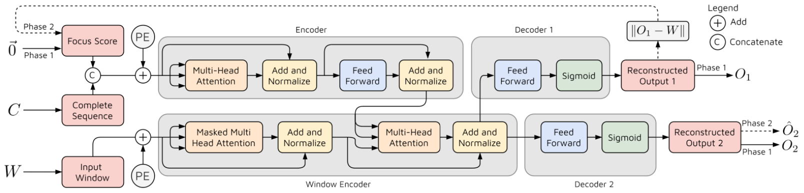
Pyraformer - Pyramid Attention



- Attention based on C-ary trees – attention starts at individual samples. Instead going each-with-each sample, looks at gradually prolonging windows.
- Model developes intra-scale and inter-scale attentions to better capture short and long dependencies in the signal.

Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Series Modeling and Forecasting

TranAD - Advesarial Tranformer for Anomaly Detection



- Two phase inference - with two encoders and decoders.
- Phase 1 - propagate full sequence C and window W and to get reconstructions O_1 and O_2 and **Focus Score**.
- The second pass outputs the final focused reconstruction $\hat{O}_2$.
- Aiming to identify spots where the expected (reconstructed) signal differs from observed.

TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data

TranAD - Advesarial Tranformer for Anomaly Detection

- The focus score generated in the first phase indicates the discrepancy between the reconstructed output and input – modify the attention weights for the second pass.
- The second pass helps to:
  1. emphasise differences between reconstructed and original signal,
  2. prevents false positives by capturing short-term temporal trends
  3. help generalizability and robustness of the adversarial style training

TranAD - Training

- Training is two phase:
  1. In the first phase, both O_1 and O_2 simply learns to predict (reconstruct) the signal.
  2. The second phase represent the adversarial loss - trying to distinguish between original W and candidate reconstruction $\hat{O}_2$.
- The first decoder aims to fool the second decoder by aiming to create a degenerate focus score (a zero vector) by perfectly reconstructing the input — $O_1 = W$.
- This pushes the decoder 2, in this phase, to generate the same output as O_2 which it aims to match the input in phase 1.

$$L_1 = \|O_1 - W\|_2$$

$$L_2 = \|O_2 - W\|_2$$

$$\min_{D1} \max_{D2} \|\hat{O}_2 - W\|_2$$

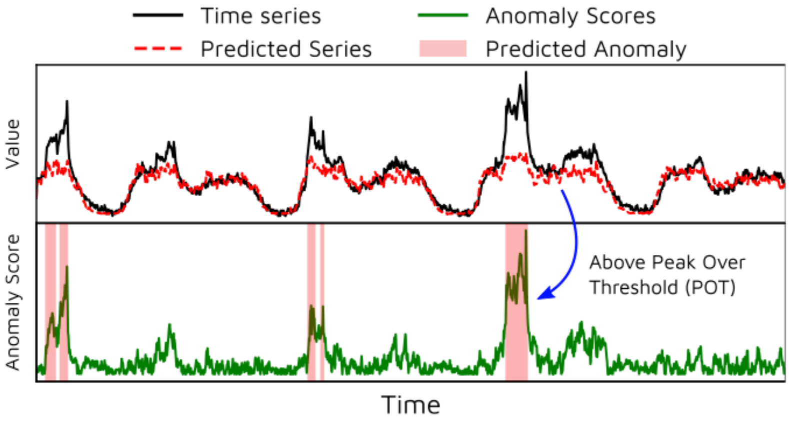


Figure 2: Visualization of anomaly prediction.

TARNet

- Build a transformer to encode the time signal.
- Encoding is passed to head to calculate the desired output.
- The training aims at two tasks - the task and reconstruction.
  - Pass the original signal through transformer encoder and head to obtain the prediction.
  - Mask randomly remove samples (zeros them out) and uses the same transformer encoder and independent head to reconstruct the signal.
- Idea is that learning dependancies and correlation in the signal will help perform the main task better.

Ranak Roy Chowdhury et al. 2022. TARNet: Task-Aware Reconstruction for Time-Series Transformer

TARNet

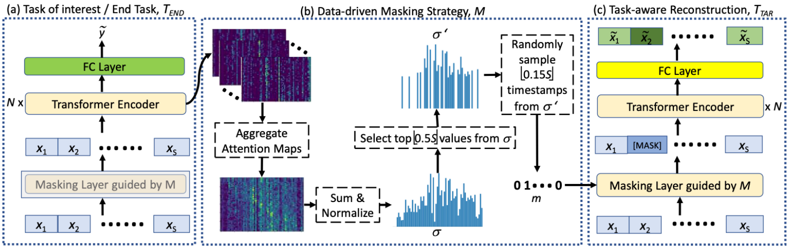

(a) Task of interest / End Task, T_{END}
$\tilde{y}$
FC Layer
N x
Transformer Encoder
x_1 x_2 x_S
Masking Layer guided by M
(b) Data-driven Masking Strategy, M
Aggregate Attention Maps
Sum & Normalize
σ'
σ
Select top $\lfloor 0.5S \rfloor$ values from σ
Randomly sample $\lfloor 0.15S \rfloor$ timestamps from σ'
0 1 • • • 0
m
(c) Task-aware Reconstruction, T_{TAR}
$\tilde{x}_1$ $\tilde{x}_2$ $\tilde{x}_S$
FC Layer
Transformer Encoder
x N
[MASK]
Masking Layer guided by M

Figure 1: TARNet Overview: (a) Task of interest / End Task, T_{END}: Data is mean-standardized, then passed through an Embedding and a Positional Encoding layer (not shown for simplicity), followed by the N-layer Transformer Encoders and Fully Connected (FC) Layer; (b) Data-driven Masking Strategy, M: For every time-series data, we collect attention maps generated by Transformer Encoders in T_{END} and then compute the set of important timestamps to be masked in task-aware reconstruction; and (c) Task-aware Reconstruction, T_{TAR}: Input data are masked at timestamps computed by M and reconstructed. Transformer Encoder parameters are shared between T_{END} and T_{TAR}, but the FC layers are different (highlighted by different colors).

TARNet

- By changing the *FC Layer* at the end of End Task, model can perform any task - classification, regression, prediction, anomaly detection.
- The masking is not performed at random.
- Uses attention from the transformer to identify samples deemed to be important for the end task.
- Create aggregated attention mask (by combining all attention masks from the transformer) and then a random subset of samples with highest attention.

Dataset	ED	MLSTM-FCNs	DTWD	TapNet	DTWI	NS	WEASEL-MUSE	TS-TCC	TNC	ShapeNet	TS2Vec	Rocket	MiniRocket	TST	TARNet
ArticularyWordRecognition	0.970	0.973	0.987	0.987	0.980	0.987	0.990	0.953	0.973	0.987	0.987	**0.993**	**0.993**	0.947	0.977
AtrialFibrillation	0.267	0.267	0.220	0.333	0.267	0.133	0.333	0.267	0.133	0.400	0.200	0.067	0.133	0.533	**1.000**
BasicMotions	0.676	0.950	0.975	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.975	**1.000**	0.975	**1.000**	**1.000**	0.925	**1.000**
CharacterTrajectories	0.964	0.985	0.989	**0.997**	0.969	0.994	0.990	0.985	0.967	0.980	0.995	0.991	0.990	0.971	0.994
Cricket	0.944	0.917	**1.000**	0.958	0.986	0.986	**1.000**	0.917	0.958	0.986	0.972	**1.000**	0.986	0.847	**1.000**
DuckDuckGeese	0.275	0.675	0.600	0.575	0.550	0.675	0.575	0.380	0.460	0.725	0.680	0.500	**0.750**	0.300	**0.750**
EigenWorms	0.549	0.504	0.618	0.489	-	0.878	**0.890**	0.779	0.840	0.878	0.847	0.650	0.790	0.720	0.420
Epilepsy	0.666	0.761	0.964	0.971	0.978	0.957	**1.000**	0.957	0.957	0.987	0.964	0.986	**1.000**	0.775	**1.000**
ERing	0.133	0.133	0.133	0.133	0.133	0.133	0.133	0.904	0.852	0.133	0.874	**0.989**	0.974	0.930	0.919
EthanolConcentration	0.293	0.373	0.323	0.323	0.304	0.236	0.430	0.285	0.297	0.312	0.308	**0.450**	0.430	0.337	0.323
FaceDetection	0.519	0.545	0.529	0.556	-	0.528	0.545	0.544	0.536	0.602	0.501	0.638	0.612	0.625	**0.641**
FingerMovements	0.550	0.580	0.530	0.530	0.520	0.540	0.490	0.460	0.470	0.580	0.480	0.520	0.550	0.590	**0.620**
HandMovementDirection	0.278	0.365	0.231	0.378	0.306	0.270	0.365	0.243	0.324	0.338	0.338	0.486	0.392	**0.675**	0.392
Handwriting	0.200	0.286	0.286	0.357	0.316	0.533	**0.605**	0.498	0.249	0.451	0.515	0.596	0.520	0.359	0.281
Heartbeat	0.619	0.663	0.717	0.751	0.658	0.737	0.727	0.751	0.746	0.756	0.683	0.741	0.771	**0.782**	0.780
InsectWingbeat	0.128	0.167	-	0.208	-	0.160	-	0.264	0.469	0.250	0.466	0.179	0.229	**0.687**	0.137
JapaneseVowels	0.924	0.976	0.949	0.965	0.959	0.989	0.973	0.930	0.978	0.984	0.984	0.978	0.986	**0.995**	0.992
Libras	0.833	0.856	0.870	0.850	0.894	0.867	0.878	0.822	0.817	0.856	0.867	0.906	0.922	0.861	**1.000**
LSST	0.456	0.373	0.551	0.568	0.575	0.558	0.590	0.474	0.595	0.590	0.537	0.635	0.653	0.576	**0.976**
MotorImagery	0.510	0.510	0.500	0.590	-	0.540	0.500	0.610	0.500	0.610	0.510	0.460	0.610	0.610	**0.630**
NATOPS	0.850	0.889	0.883	0.939	0.850	**0.944**	0.870	0.822	0.911	0.883	0.928	0.872	0.933	0.939	0.911
PEMS-SF	0.705	0.699	0.711	0.751	0.734	0.688	-	0.734	0.699	0.751	0.682	0.832	0.809	0.930	**0.936**
PenDigits	0.973	0.978	0.977	0.980	0.939	0.983	0.948	0.974	0.979	0.977	**0.989**	0.981	0.967	0.981	0.976
Phoneme	0.104	0.110	0.151	0.175	0.151	0.246	0.190	0.252	0.207	**0.298**	0.233	0.273	0.291	0.111	0.165
RacketSports	0.868	0.803	0.803	0.868	0.842	0.862	0.934	0.816	0.776	0.882	0.855	0.901	0.868	0.796	**0.987**
SelfRegulationSCP1	0.771	0.874	0.775	0.652	0.765	0.846	0.710	0.823	0.799	0.782	0.812	0.867	0.915	**0.961**	0.816
SelfRegulationSCP2	0.483	0.472	0.539	0.550	0.533	0.556	0.460	0.533	0.550	0.578	0.578	0.555	0.506	0.604	**0.622**
SpokenArabicDigits	0.967	0.990	0.963	0.983	0.959	0.956	0.982	0.970	0.934	0.975	0.988	0.997	0.963	**0.998**	0.985
StandWalkJump	0.200	0.067	0.200	0.400	0.333	0.400	0.333	0.333	0.400	0.533	0.467	0.467	0.333	**0.600**	0.533
UWaveGestureLibrary	0.881	0.891	0.903	0.894	0.868	0.884	0.916	0.753	0.759	0.906	0.906	**0.931**	0.785	0.913	0.878
PAMAP2	0.718	0.949	0.683	0.865	0.769	0.885	0.928	0.942	0.938	0.948	0.941	0.931	0.962	0.948	**0.974**
OpportunityGestures	0.655	0.768	0.762	0.574	0.715	0.689	0.553	0.791	0.821	0.730	0.771	0.813	0.809	0.732	**0.830**
OpportunityLocomotion	0.845	0.900	0.859	0.850	0.868	0.859	0.634	0.881	0.874	0.874	0.842	0.875	0.886	0.907	**0.908**
Occupancy [15]	0.496	0.873	0.517	0.844	0.526	0.817	0.556	0.865	0.828	0.852	0.876	0.832	0.878	0.881	**0.883**
Total best accuracy	0	0	1	2	1	2	5	1	0	2	1	6	4	7	**17**
Average accuracy	0.596	0.651	0.658	0.672	0.675	0.686	0.688	0.692	0.693	0.717	0.722	0.732	0.741	0.745	**0.772**
Ours 1-to-1 Wins	32	26	27	23	31	23	25	28	29	25	24	20	21	20	-
Ours 1-to-1 Draws	0	0	2	2	1	2	3	1	1	2	0	2	4	0	-
Ours 1-to-1 Losses	2	8	5	9	2	9	6	5	4	7	10	12	9	14	-
Mean Rank	12.15	8.79	9.65	7.44	10.44	7.59	7.79	9.03	9.41	5.47	7.18	5.18	4.71	5.74	**4.00**

Are Transformers Effective for Time Series Forecasting?