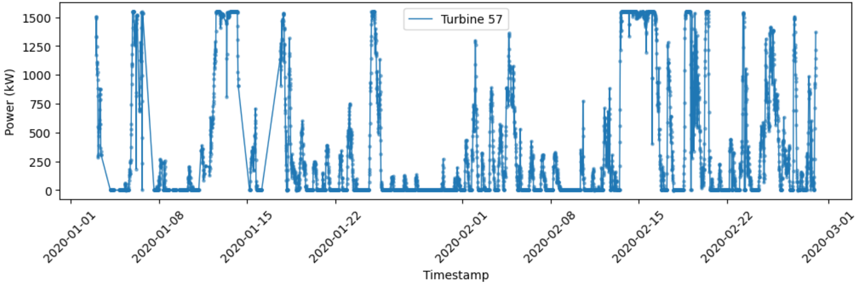
Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

TimeTransformer

Daniel Dold
HTWG Konstanz
Institute for Optical Systems

What are timeseries?



?

Timeseries Methods

- ARIMA
- CNN
- RNN/ LSTM

But why not use transformers for timeseries?

Image from: https://wallpaperaccess.com/full/1595505.jpg

Transformer architecture

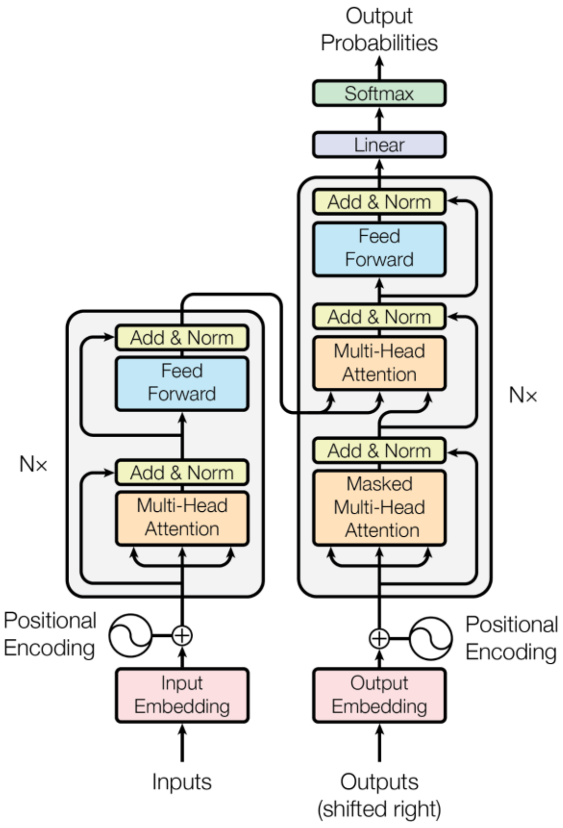
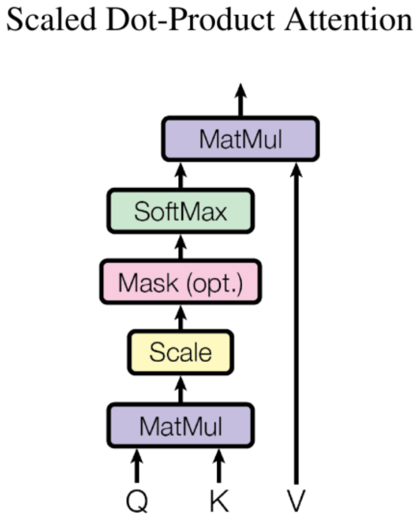
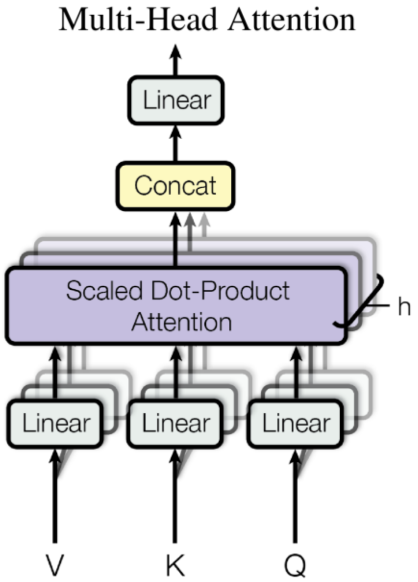


Figure 1: The Transformer - model architecture.





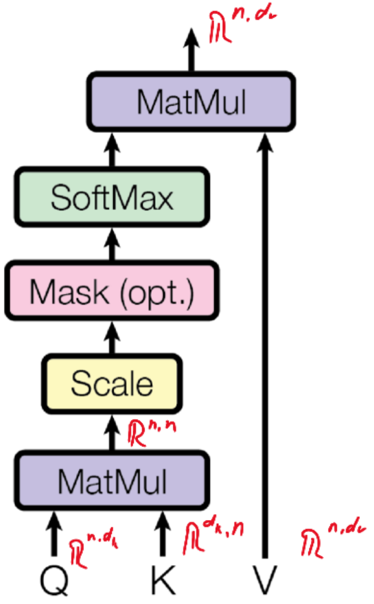
$$\text{att} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{att}_1, \dots, \text{att}_h)W^O$$

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Transformer architecture (shapes)

Scaled Dot-Product Attention



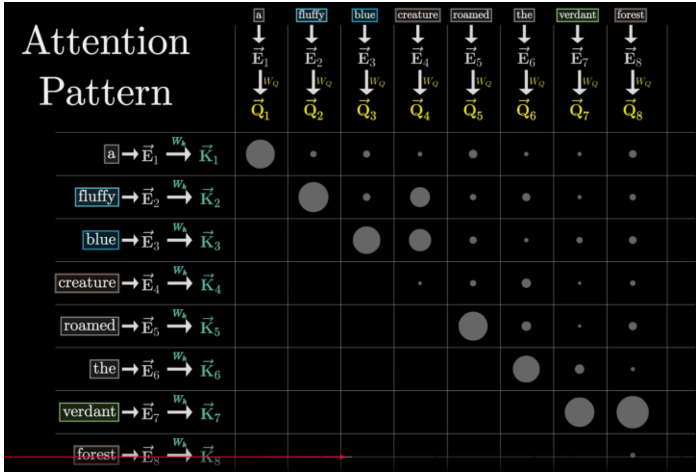
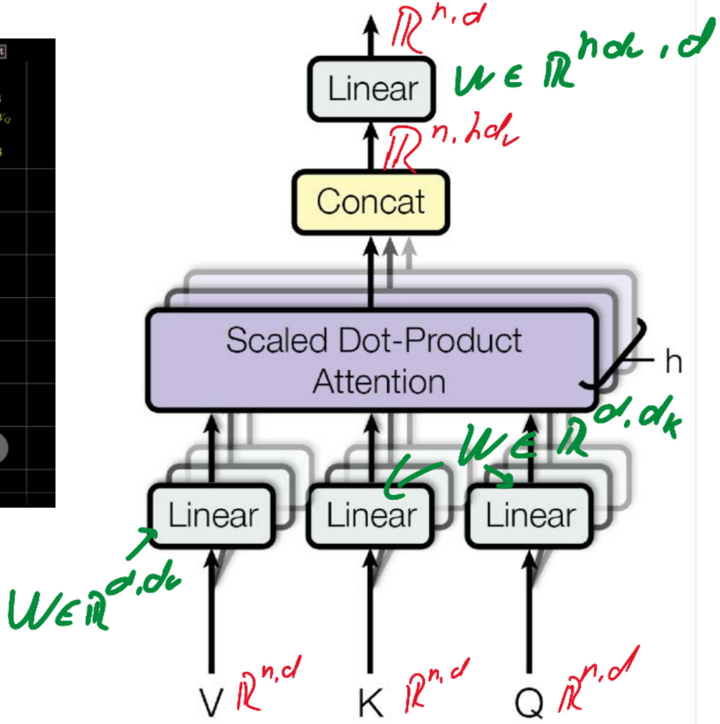


Image from:
https://www.youtube.com/watch?v=eMlx5fFNoYc&t=791s

Multi-Head Attention



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Jax example

```python
from flax import linen as nn
import jax

# Flax Transformer Example
class SimpleFlaxTransformer(nn.Module):
    num_heads: int = 2
    qkv_features: int = 8 # (h*dk)
    out_f: int = 3

    @nn.compact
    def __call__(self, x, mask=None):
        x = nn.MultiHeadDotProductAttention(num_heads=self.num_heads, qkv_features=self.qkv_features, out_features=self.out_f)(x, mask=mask)
        return x

# Example input
key = jax.random.PRNGKey(0)
x = jax.random.normal(key, (2, 20, 9))  # (batch, seq, in_features)
model = SimpleFlaxTransformer()
params = model.init(key, x)

# Create a mask (e.g., mask out last 5 positions in the sequence)
# Shape: (batch, num_heads, seq_len, seq_len)
mask = jnp.ones((x.shape[0], model.num_heads, x.shape[1], x.shape[1]), dtype=jnp.float32)
mask = mask.at[:, :, -5:].set(0)

output = model.apply(params, x, mask=mask)
print("Flax Transformer input shape:", x.shape)
print("Flax Transformer output shape:", output.shape)

print("Flax Transformer parameters:")
for k,v in jax.tree_util.tree_flatten_with_path(params['params'])[0]:
    name = f"{k[1].key}/{k[2].key}"
    print(f"\t{name},\t{v.shape}")
```

✓ 0.1s

$\Rightarrow d_k = 4$

```
Flax Transformer input shape: (2, 20, 9)
Flax Transformer output shape: (2, 20, 3)
Flax Transformer parameters:
        key/bias,       (2, 4)
        key/kernel,     (9, 2, 4)
        out/bias,       (3,)
        out/kernel,     (2, 4, 3)
        query/bias,     (2, 4)
        query/kernel,   (9, 2, 4)
        value/bias,     (2, 4)
        value/kernel,   (9, 2, 4)
```

d, d_k

h, d_k, d

Timeseries Metrics

- Root mean square error (RMSE)
- Mean absolute percentage error (MAPE)
  - $\text{MAPE} = \frac{1}{NH}\sum_{i=1}^{N}\sum_{t=T+1}^{T+H}\frac{|y_{i,t}-f_{i,t}|}{|y_{i,t}|}$
- Mean absolute scaled error (MASE)
  - $\text{MASE} = \frac{1}{NH}\sum_{i=1}^{N}\sum_{t=T+1}^{T+H}\frac{|y_{i,t}-f_{i,t}|}{a_i}, a_i = \frac{1}{T-m}\sum_{t=m+1}^{T}|y_{i,t} - y_{i,t-m}|$

- Continuous ranked probability score (CRPS)
  - $\text{CRPS} = \int \left(F(x) - I(y \leq x)\right)^2 dx$
- Weighted quantile loss (WQL)
  - $\text{WQL} = \frac{2}{\sum_{i=1}^{N}\sum_{t=T+1}^{T+H}|y_{i,t}|}\sum_{i=1}^{N}\sum_{t=T+1}^{T+H}\sum_{q}\rho_q\left(y_{i,t}, f_{i,t}^{q}\right)$

    $\rho_q\left(y_{i,t}, f_{i,t}^{q}\right) = \begin{cases} q\left(y_{i,t} - f_{i,t}^{q}\right) & , y_{i,t} \leq f_{i,t}^{q} \\ (1-q)\left(y_{i,t} - f_{i,t}^{q}\right) & , y_{i,t} > f_{i,t}^{q} \end{cases}$

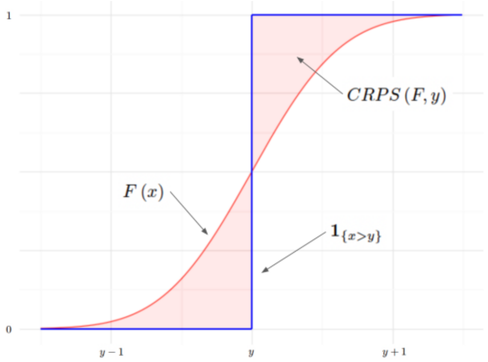


Image from: https://towardsdatascience.com/crps-a-scoring-function-for-bayesian-machine-learning-models-dd55a7a337a8/

https://auto.gluon.ai/stable/tutorials/timeseries/forecasting-metrics.html#autogluon.timeseries.metrics

Positional encoding

- Vanilla Positional Encoding (Absolute encodings)
  - Like sine/cosine embedding
  - Were unable to fully exploit the important features of time series data
- Learnable Positional Encoding
  - Adaptable to specific task
- Timestamp Encoding
  - Using timestamp information + additional lernable embeding layer
  - Used by Informer, Autoformer and FEDformer
- Rotary Position Embeddings (RoPE) (Su et al., 2024)
- Learned binary attention biases (Yang et al., 2022)

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., & Liu, Y. (2024). RoFormer: Enhanced transformer with Rotary Position Embedding. Neurocomputing, 568, 127063. https://doi.org/10.1016/j.neucom.2023.127063

Yang, J., Gupta, A., Upadhyay, S., He, L., Goel, R., & Paul, S. (2022). TableFormer: Robust Transformer Modeling for Table-Text Encoding. In S. Muresan, P. Nakov, & A. Villavicencio (Hrsg.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (S. 528–537). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.40

Timeseries Benchmark

- GIFT-Eval Time Series Forecasting
  - 15 univariate and 8 multivariate datasets,
  - covering 7 domains and 10 frequencies, totaling 144,000 time series and 177 million observations

Table 1: Property comparisons of various forecasting benchmarks.

Benchmark \ Property	Data			Forecasting Task		Evaluation	
	Freq. Range	Num. of Domain	Pretraining data	Num. of var.	Pred. Len.	Benchmark Methods	Prob. Forecasting
Monash (Godahewa et al., 2021)	Secondly ∼ Yearly	7	No	Uni	Short	Stat./DL	No
TFB (Qiu et al., 2024)	Minutely ∼ Yearly	6	No	Uni/Multi	Short	Stat./DL	No
LTSF (Zeng et al., 2022)	Minutely ∼ Weekly	5	No	Multi	Long	Stat./DL	No
BasicTS+ (Shao et al., 2023)	Minutely ∼ Daily	3	No	Multi	Short/Long	Stat./DL	No
GIFT-Eval (our work)	Secondly ∼ Yearly	7	Yes	Uni/Multi	Short/Long	Stat./DL/FM	Yes

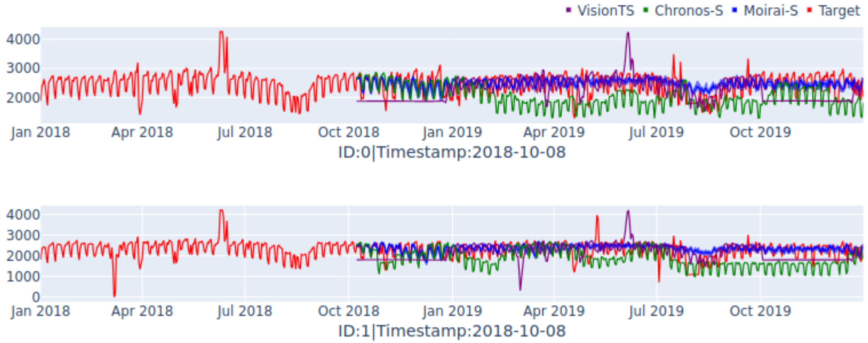
Table 3: GIFT-Eval Test data statistics aggregated by domain.

Domain	Econ/Fin	Energy	Healthcare	Nature	Sales	Transport	Web/CloudOps	Grand Total
# Series	99,974	2,036	1,036	32,618	3,717	1,341	3,524	144,246
# Obs	25,266,415	74,119,755	129,408	3,154,921	671,707	38,028,955	16,610,251	157,981,412

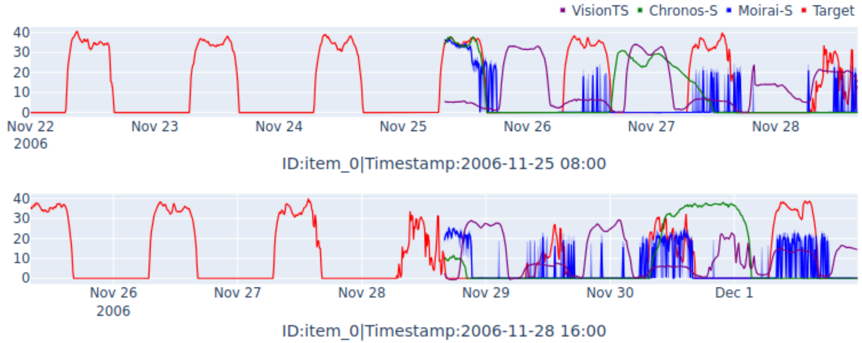
Aksu, T., Woo, G., Liu, J., Liu, X., Liu, C., Savarese, S., ... & Sahoo, D. (2024). GIFT-Eval: A Benchmark for General Time Series Forecasting Model Evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models*.

Timeseries Benchmark

GIFT-Eval some examples



(c) Foundation model forecasts sampled on *M_DENSE* daily dataset with long prediction length.



(d) Foundation model forecasts sampled on *Solar* ten–minutely dataset with medium prediction length.

Aksu, T., Woo, G., Liu, J., Liu, X., Liu, C., Savarese, S., ... & Sahoo, D. (2024). GIFT-Eval: A Benchmark for General Time Series Forecasting Model Evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models*.

Timeseries Benchmark

GIFT-Eval Leaderboard

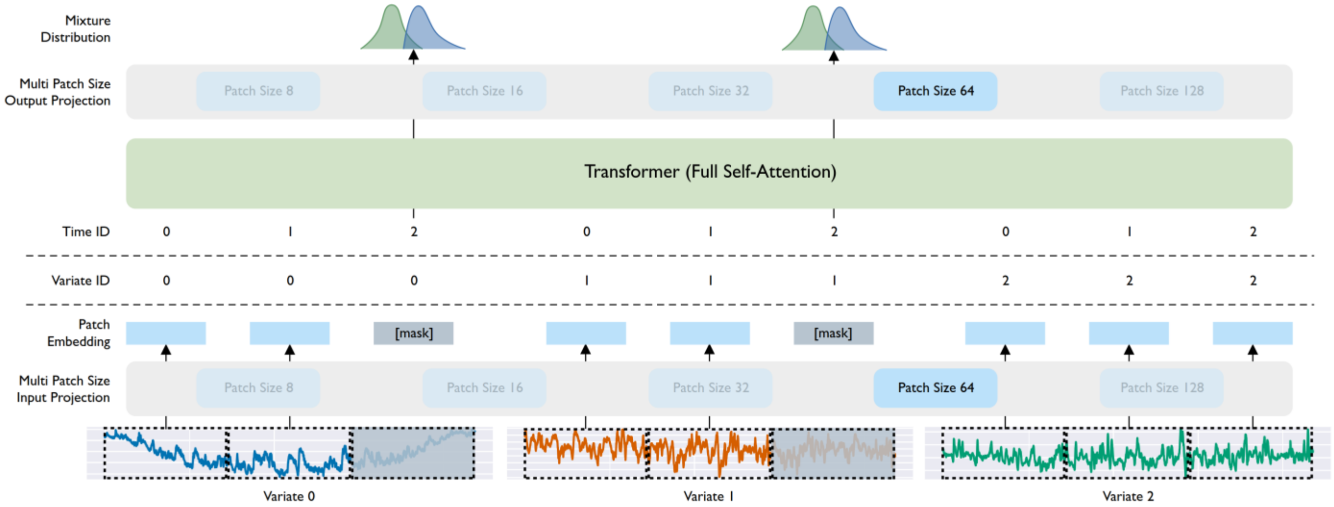
	T	model	MASE	CRPS	Rank
Pretrained	🟢	YingLong_110m	0.726	0.471	10.897
	🟢	TabPFN-TS	0.692	0.46	11.144
	🟢	chronos_bolt_base (code)	0.725	0.485	11.371
	🟢	timesfm_2_0_500m (code)	0.680	0.465	11.526
Fine-tuned	🔶	TEMPO_ensemble	0.773	0.434	11.711
	🟢	YingLong_50m	0.738	0.479	11.866
	🟢	chronos_bolt_small (code)	0.738	0.487	12.423
	🟢	sundial_base_128m	0.673	0.472	13.062
	🔶	TTM-R2-Finetuned (code)	0.679	0.492	13.691
	🟢	Moirai_large (code)	0.785	0.506	14.021
	🟢	Moirai_base (code)	0.809	0.515	14.062
DL	🔷	PatchTST	0.762	0.496	14.247

https://huggingface.co/spaces/Salesforce/GIFT-Eval

Aksu, T., Woo, G., Liu, J., Liu, X., Liu, C., Savarese, S., ... & Sahoo, D. (2024). GIFT-Eval: A Benchmark for General Time Series Forecasting Model Evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models.*

Moirai

- Goal:
  - Universal timeseries forecasting
- Challenges:
  - Time series data is highly heterogeneous
  - Should consider multivariate interactions and take exogenous covariates into account
  - Should capture different kinds of outcome distributions



Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S. & Sahoo, D.. (2024). Unified Training of Universal Time Series Forecasting Transformers. *Proceedings of the 41st International Conference on Machine Learning*, in *Proceedings of Machine Learning Research* 235:53140-53164

Moirai

- Uses patch-based projections
  - „flatten" multivariate time series, considering all variates as a single sequence
  - Opting for a larger patch size to handle high-frequency data, ➔ lower quadratic computation cost of attention
- Instance normalization (Kim et al., 2022) is applied to inputs/outputs
- Rotary Position Embeddings (RoPE)
  - Learned embeddings leads to suboptimal results
- Learned binary attention biases (Yang et al., 2022)
- Mixture of parametric distributions
  - Student's t-distribution
  - Negative binomial distribution
  - Log-normal distribution
  - Low variance normal distribution

$$E_{ij,mn} = (\boldsymbol{W}^Q \boldsymbol{x}_{i,m})^T \boldsymbol{R}_{i-j} (\boldsymbol{W}^K \boldsymbol{x}_{j,n}) + u^{(1)} * \mathbb{1}_{\{m=n\}} + u^{(2)} * \mathbb{1}_{\{m \neq n\}},$$

$$A_{ij,mn} = \frac{\exp\{E_{ij,mn}\}}{\sum_{k,o} \exp\{E_{ik,mo}\}},$$

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S. & Sahoo, D.. (2024). Unified Training of Universal Time Series Forecasting Transformers. *Proceedings of the 41st International Conference on Machine Learning*, in *Proceedings of Machine Learning Research* 235:53140-53164

Instance normalization (Kim et al., 2022)

- Reversible instance normalization for accurate time-series forecasting against distribution shift

- K = variabls, $\mathrm{T}_{\mathrm{x/y}}$ = input/output sequeneze length

$$\hat{x}_{kt}^{(i)} = \gamma_k \left(\frac{x_{kt}^{(i)} - \mathbb{E}_t[x_{kt}^{(i)}]}{\sqrt{\mathrm{Var}[x_{kt}^{(i)}] + \epsilon}} \right) + \beta_k,$$

$$\mathbb{E}_t[x_{kt}^{(i)}] = \frac{1}{T_x} \sum_{j=1}^{T_x} x_{kj}^{(i)} \quad \text{and} \quad \mathrm{Var}[x_{kt}^{(i)}] = \frac{1}{T_x} \sum_{j=1}^{T_x} \left(x_{kj}^{(i)} - \mathbb{E}_t[x_{kt}^{(i)}] \right)^2.$$

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J. H., & Choo, J. (2021, May). Reversible instance normalization for accurate time-series forecasting against distribution shift. *In International conference on learning representations*.

Query-Key Normalization for Transformers

- Normalize over Q, K ith rows

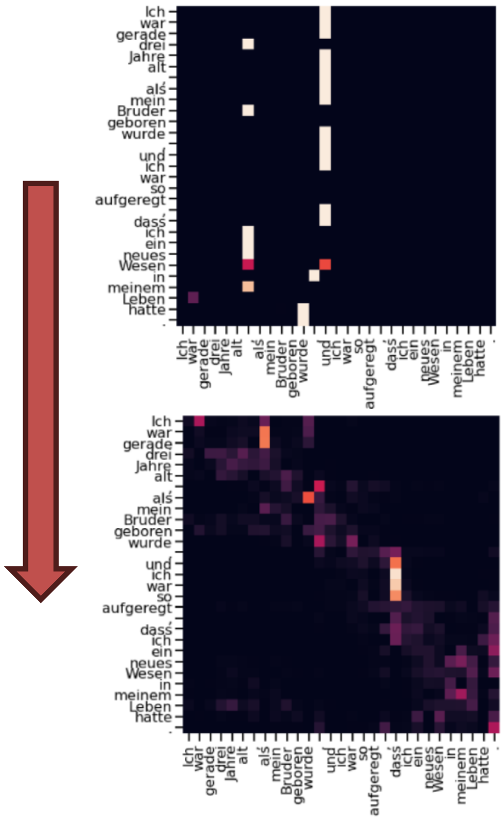
$$\hat{q}_i = \frac{q_i}{||q_i||}$$

- And use learnable scale parameter instead of sqrt(d)

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \qquad (4)$$

to

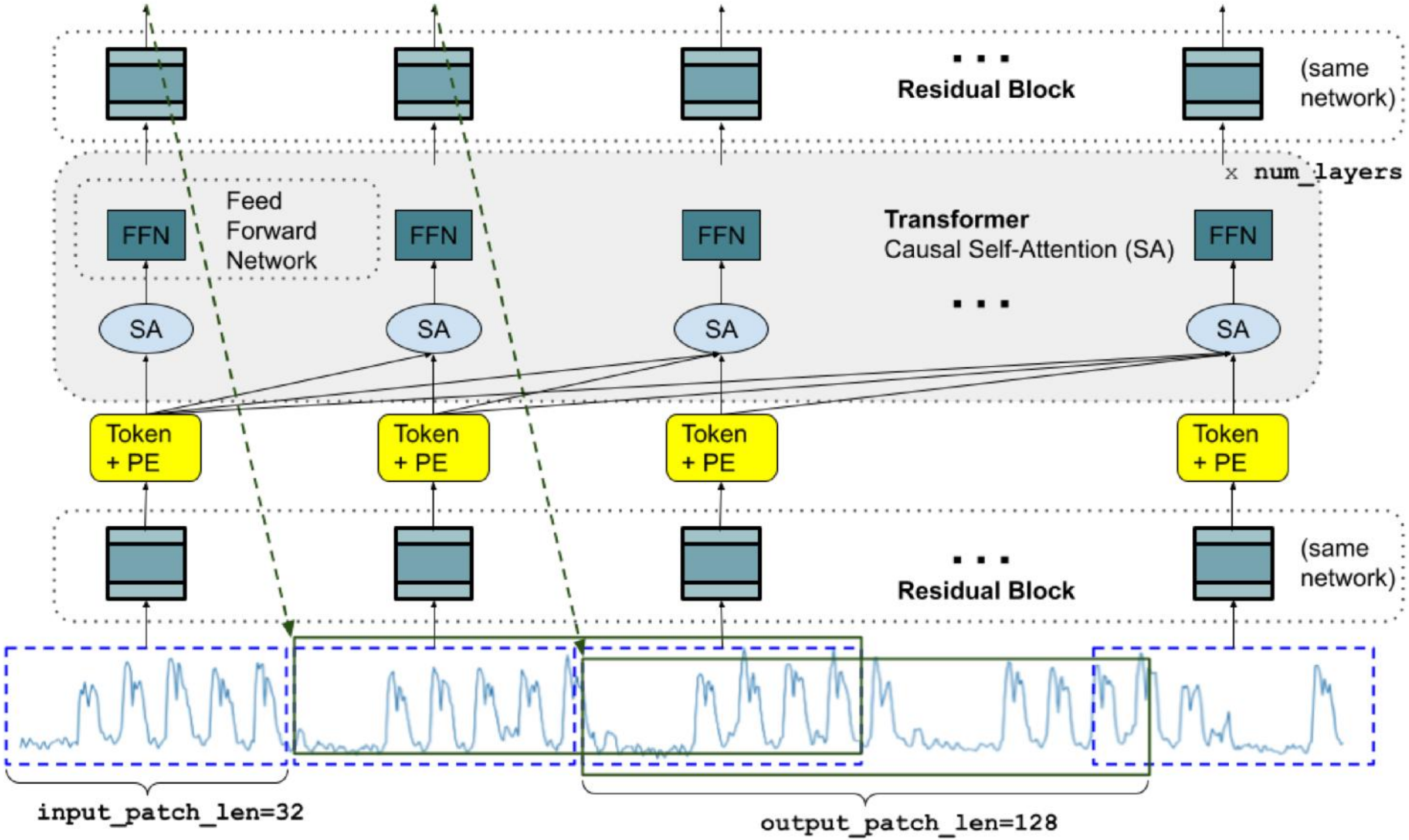
$$\text{softmax}(g * \hat{Q}\hat{K}^T)V \qquad (5)$$



Henry, A., Dachapally, P. R., Pawar, S., & Chen, Y. (2020). Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*.

TimesFM

- Decoder-only model
- Patch based



Das, A., Kong, W., Sen, R., & Zhou, Y. (2024, July). A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*.

TimesFM

- Patch based approach
  - Patching improves performance
  - Longer output patches improves performance
    - decoder can yield better results than auto- regressive decoding on long horizon benchmarks
- Patch Masking
  - Are sampled during training
  - Enables multiple context length
- Vanilla positional encoding
- No probabilistic forecasting
- Trained Dataset
  - Google trends
  - Wiki Pageview statistics
  - synthetic time-series
- Maximum context length of 512
- Reversible instance normalization (Kim et al., 2022)

Das, A., Kong, W., Sen, R., & Zhou, Y. (2024, July). A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*.

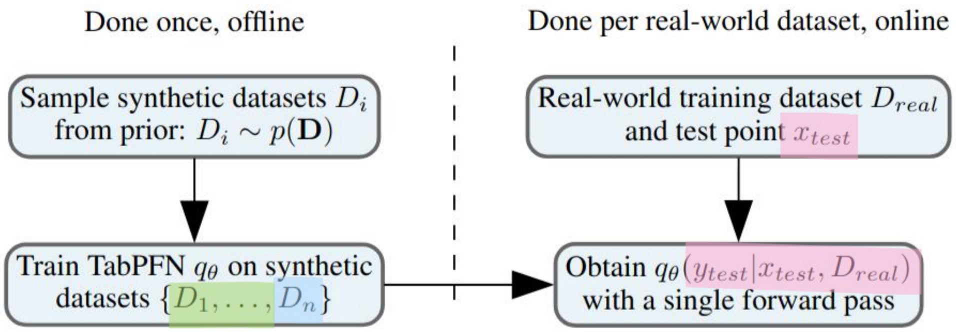
Chronos

- Seeks to transfer NLP transformers to time series with little adaptation.
  - **Ignore time and frequency** information, treating the “time series” simply as a sequence
- Uses **classification** instead of regression
  - Input data is first scaled and then quantized
    - With mean scaling $\widetilde{x}_i = \frac{x_i - m}{s}; m = 0, s = \frac{1}{C}\sum_{i=1}^{C}|x_i|$
    - And Uniform Binning with fixed number of bins
  - **“Probabilistic” output**
- Uses large collection of publicly available datasets and complemented it by a synthetic dataset
  - Synthetic dataset was generated via Gaussian processes
- Uses 8 A100 (40GB) GPUs to train all Chronos models
- **Learnings**:
  - „training/ test loss improves with the model capacity
  - „language model weights are not particularly remarkable in the context of time series forecasting
  - „longer context improves forecast“
  - „Even though the cross entropy is not distance-aware, the model learns to estimate **distributions over neighboring tokens**, and of diverse shapes, including multimodal ones“

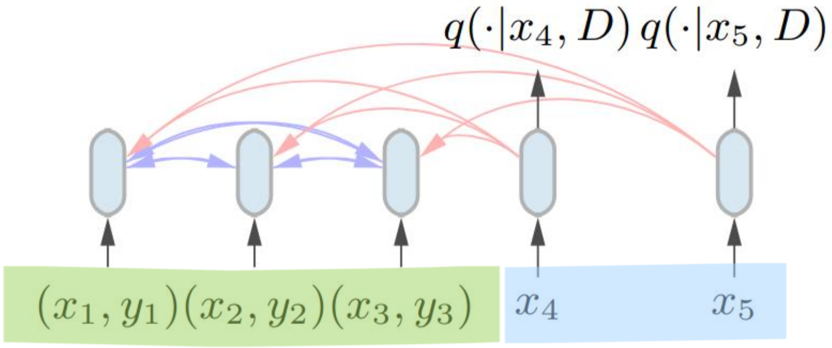
Ansari, A. F., Stella, L., Turkmen, A. C., Zhang, X., Mercado, P., Shen, H., ... & Wang, B. Chronos: Learning the Language of Time Series. *Transactions on Machine Learning Research*.

TabPFN

- Designed for tabular data
- Based on transformer architecture
- Encodes the feature tuple $(x_i, y_i) \in D$ as one token
- Trained on a synthetic data (Prior-Data Fitted Network (PFN) (Müller et al. 2022)
- Performs in-context learning (ICL) with own dataset D_{real}



(a) Prior-fitting and inference



$D_{\text{train\_Syn}} = D_{\text{Syn}} \setminus \mathrm{D}_{\text{test\_Syn}}$ and $D_{\text{test\_Syn}} \subset D$

$D_{\text{Syn}} \coloneqq \{(x_1, y_1), \ldots, (x_n, y_n)\}$

Hollmann, N., Müller, S., Eggensperger, K., & Hutter, F (2023). TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*.

PFN a PPD approximator

- TabPFN approximate the **p**osterior **p**redictive **d**istribution (PPD)

$$q_\theta(y|x,D) \propto \int_\Phi p(y|x,\phi)p(D|\phi)p(\phi)d\phi$$

- θ is optimized trough cross-entropy loss $\min_\theta l_\theta = \min_\theta -q_\theta(y|x,D)$

$$l_\theta = \mathbb{E}_{x,D\sim p(D)}\left[H\big(p(y|x,D), q_\theta(y|x,D)\big)\right]$$

$$l_\theta = \mathbb{E}_{x,D\sim p(D)}\left[KL\big(p(y|x,D), q_\theta(y|x,D)\big)\right] + C$$

- Proof Müller et. Al (2022):

$$\ell_\theta = -\int_{D,x,y} p(x,y,D)\log q_\theta(y|x,D) = -\int_{D,x} p(x,D)\int_y p(y|x,D)\log q_\theta(y|x,D) \quad (3)$$

$$= \int_{D,x} p(x,D)\mathrm{H}(p(\cdot|x,D), q_\theta(\cdot|x,D)) = \mathbb{E}_{x,D\sim p(\mathcal{D})}[\mathrm{H}(p(\cdot|x,D), q_\theta(\cdot|x,D))] \quad (4)$$

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., & Hutter, F (2022). Transformers Can Do Bayesian Inference. In *International Conference on Learning Representations*.

PFN – link to – GP ?

GP

- Hyperparameter define the prior $p(f)$
- Kernel function weights data by similarity to x_{test}
- Predict by conditioning on D and x_{test}

PFN

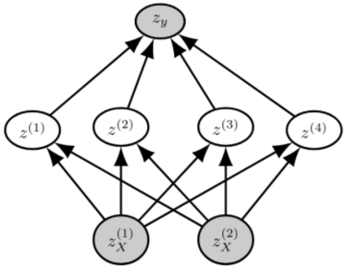
- Learn a prior $p(f)$ by optimizing θ on synthetic data
- θ + attention mimics kernel behavior
- Condition on D and x_{test}, PFN uses θ + attention to weight examples in D relative to x_{test} (uses learned simulated kernel)

Synthetic data fitting phase

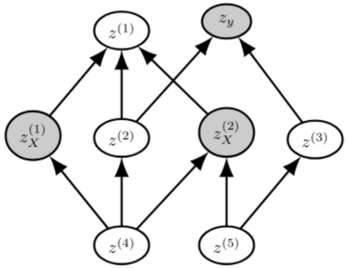
ICL phase

TabPFN Synthetic dataset

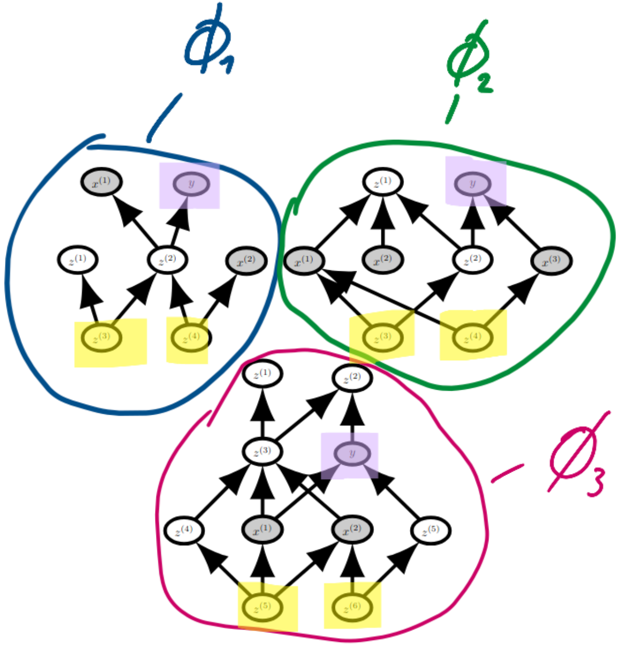
- The synthetic dataset is defined by the prior $p(D) = \mathbb{E}_{\phi \sim p(\phi)}[p(D|\phi)]$ with $\phi \in \Phi$
  - Sample a hypotheses $\phi \sim p(\phi)$
  - Generate synthetic dataset with $\mathrm{D} \sim p(D|\phi)$
    - Sample an input x and generate label $\hat{y}$
    - Splitting the values of $\hat{y}$ into intervals that map to class labels



(a) A BNN



(b) An SCM



(c) SCMs sampled from the prior

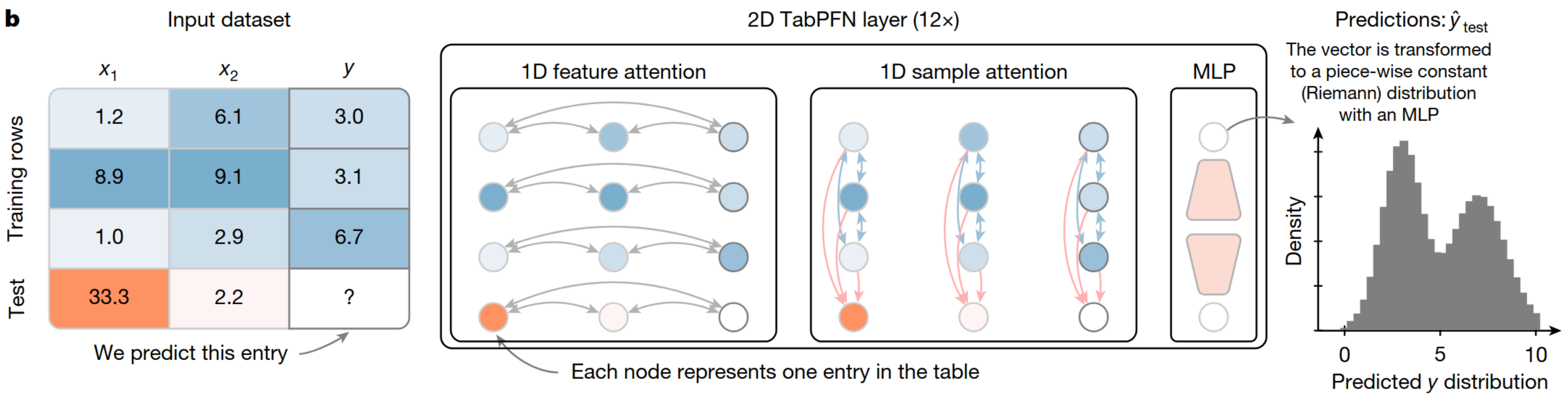
Hollmann, N., Müller, S., Eggensperger, K., & Hutter, F (2023). TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*.

TabPFN

- Handle different numbers of features by zero-padding and scaling
- Training requires 20h on 8 GPUs (Nvidia RTX 2080 Ti)
- TabPFN is much “faster” than methods with comparable performance
- Limitations
  - Size: 1000 data points 100 features 10 classes
  - Not strong with:
    - Categorical features
    - Missing values
    - Uninformative features
  - Only classification
  - High inference time

Hollmann, N., Müller, S., Eggensperger, K., & Hutter, F (2023). TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*.

TabPFN v2



- Improves all the previous limitations
  - Size 10000 datapoints, 500 features, 10 classes
- Caching for inference time
- Can handle missing, uninformative and categorical features

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., & Hutter, F. (2025). Accurate predictions on small data with a tabular foundation model. *Nature, 637(8045)*, 319–326. https://doi.org/10.1038/s41586-024-08328-6

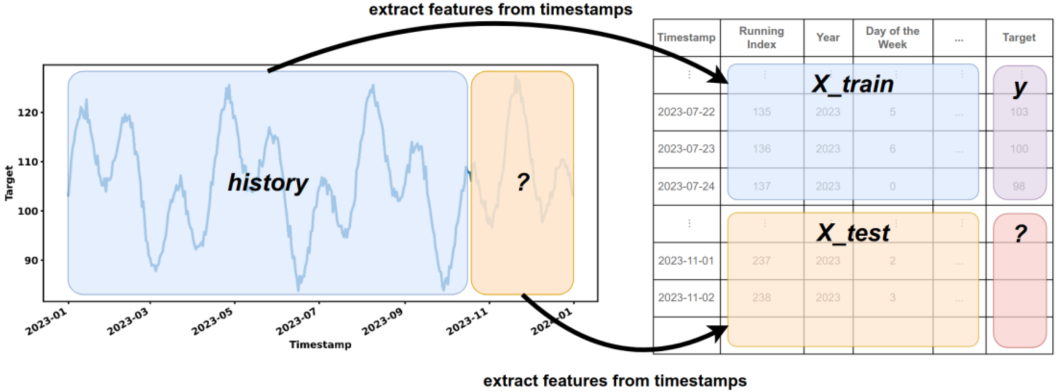
ForecastPFN

- PFN adapted to timeseries data
- Regression instead of classification
  - Optimize **MSE Loss**
- Generated a **synthetic dataset** $\mathrm{p(D)},\ D \coloneqq \{(t_1, y_{t_1}), \dots, (t_n, y_{t_n})\}$
  - Multi- scale seasonal trends
  - Linear and exponential global trends
  - Added Weibull-based noise distribution
- Splitting noise component improves the training speed
  $y_t = \psi(t) + z_t$ with $z_t \sim \mathrm{Weibull}(1, k)$
- **Makes prediction given a timestamp t**
- Special outlier adaption
  - In timeseries conventional scaling techniques cannot be applied
- Benefit: Superior in low data regime
- **Limitation**: Only univariate timeseries and can’t add exogenous features

Dooley, S., Khurana, G. S., Mohapatra, C., Naidu, S. V., & White, C. (2023). ForecastPFN: Synthetically-Trained Zero-Shot Forecasting. *Advances in Neural Information Processing Systems*, 36, 2403–2426.

TabPFN-TS

- TabPFN v2 is sufficiently general, no need for time-series-specific priors
  - No need for ForecastPFN?
- Derive features directly from the timestamps + Automatic Seasonal Features (paper v3)
  - extract several calendar-based features (sin, cos- embedding)
  - Add a running index
  - Automatic features with peaks from FFT (frequency and magnitude)



Hoo, S. B., Müller, S., Salinas, D., & Hutter, F. The Tabular Foundation Model TabPFN Outperforms Specialized Time Series Forecasting Models Based on Simple Features. In *NeurIPS Workshop on Time Series in the Age of Large Models*.

TabPFN-TS performance

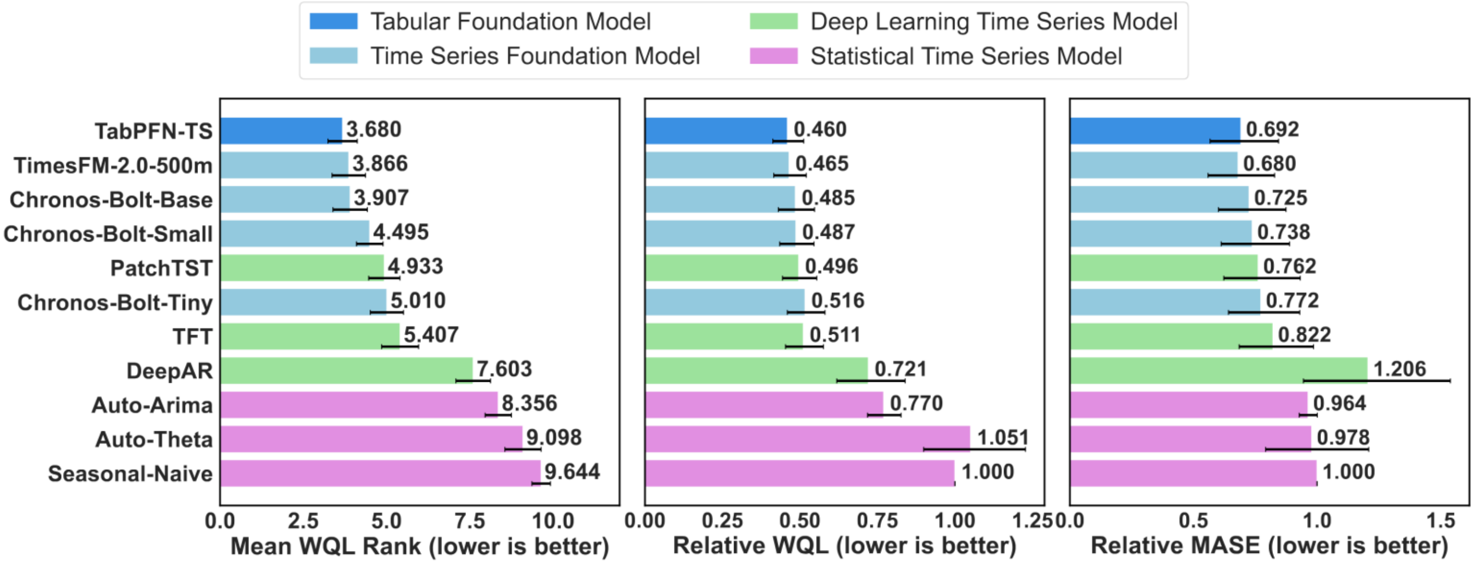
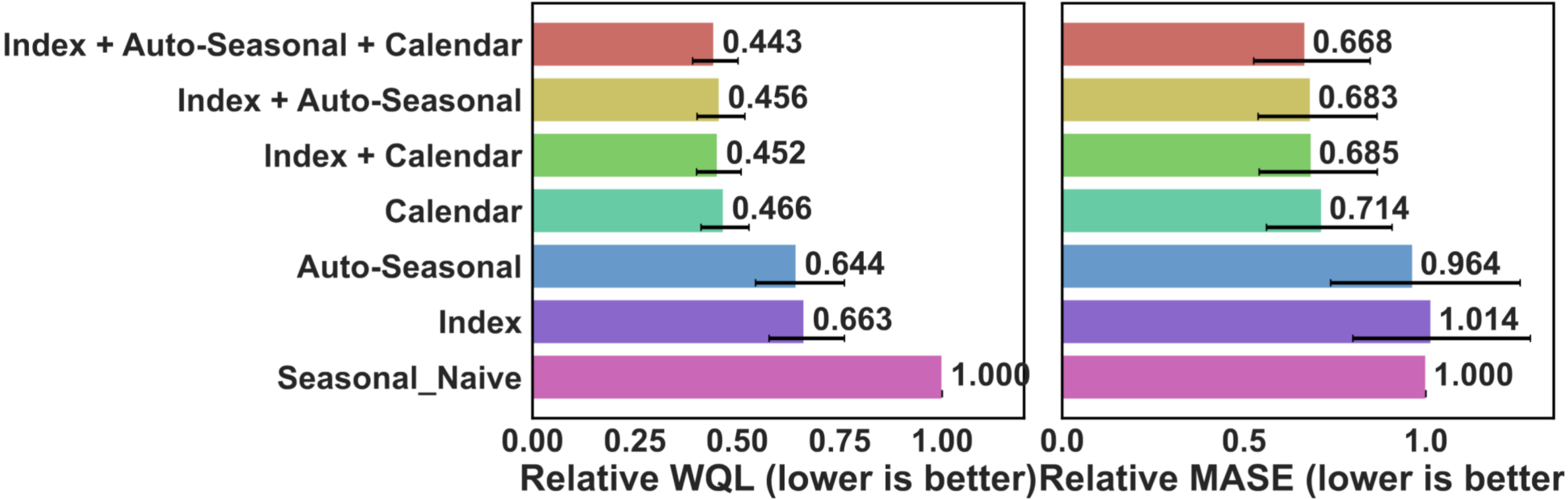


Figure 2: Forecasting performance of TabPFN-TS and baseline models on all 97 GIFT-Eval benchmarking tasks. TabPFN-TS ranks *#1 in probabilistic forecasting* (WQL, both raw and rank) and *#2 in point forecasting* (MASE). WQL and MASE are normalized by Seasonal Naive, and aggregated by geometric mean. Model ranks are aggregated by arithmetic mean. Error bars indicate 95% confidence intervals.

Hoo, S. B., Müller, S., Salinas, D., & Hutter, F. The Tabular Foundation Model TabPFN Outperforms Specialized Time Series Forecasting Models Based on Simple Features. In *NeurIPS Workshop on Time Series in the Age of Large Models*.

TabPFN-TS ablation

Effect of feature generation



Hoo, S. B., Müller, S., Salinas, D., & Hutter, F. The Tabular Foundation Model TabPFN Outperforms Specialized Time Series Forecasting Models Based on Simple Features. In *NeurIPS Workshop on Time Series in the Age of Large Models*.

INSTITUTE FOR OPTICAL SYSTEMS

Thanks for your attention

All images are sourced from the cited footnote unless otherwise specified