A decoder-only foundation model for time-series forecasting

...

Das et al., ICML, 2024

Presented By: Vance Ayscue & Nick Ochsner

Necessary Definitions

What is a time-series?

- A sequence of values indexed over time

What is zero-shot forecasting?

- Predicting future values on unseen datasets
- No retraining or fine-tuning required

Motivation

- Why Forecasting Matters
  - Why use a foundation model for time-series forecasting?
- Challenges:
  - No natural tokenization (unlike NLP)
  - Data is heterogeneous and sparse
  - Limited large-scale datasets
  - Prior Approaches
    - Statistical Models (ARIMA)
    - Deep Learning
    - LLM-based forecasting - inefficient
- Need:
  - General-purpose forecasting model

Introduction and Problem Definition

- Can a large pretrained model on time-series data learn temporal patterns that transfer zero-shot to previously unseen forecasting datasets?
- Time-series forecasting = core problem across domains
- Current methods:
  - Dataset-specific
  - Require retraining
- Goal:
  - Single pretrained model towards general forecasting
- With a zero-shot forecaster we want to predict future time points:
  - We learn a model f such that:
  - $f : (y_{1:L}) \rightarrow \hat{y}_{L+1:L+H}$
  - L is the context length and H is the forecasted horizon size.

Core Contribution

- Prior Work
  - Many forecasting approaches train separate models for each dataset.
  - These methods are typically local, supervised, and tailored to a specific domain or task.
  - As a result, they often require retraining and do not generalize well across datasets.
- What's New
  - This paper proposes a single pretrained forecasting model.
  - The model is designed to generalize across multiple datasets without retraining.
  - It operates in a zero-shot setting, enabling direct application to new data.

Time-Series Specific Design Choices

Key design choices in TimesFM:

- Decoder-only transformer architecture
- Patching (grouping time points into tokens)
- Larger output patches than input patches
- Patch masking for variable context lengths

These design choices enable efficient and flexible zero-shot forecasting.

Time-Series Specific Design: Decoder Only Model

What is a Decoder-Only Model? And why use it?

- A decoder-only model auto-regressively predicts future values using only past observations.
- Predicts the next patch from all previous patches (LM-style)
- Trained in decoder-only mode, unlike many prior models
- Enables flexible zero-shot forecasting for unknown output/horizon lengths

Time-Series Specific Design: Patching and Longer Output Patches

What is Patching?

- Split time series into fixed-size patches (p).
- Instead of single time points, use multiple consecutive values.
- Natural analogue to tokens in LMs
- Each patch is a vector of numbers (not text).

Why do this?

- Shortens sequence length → more efficient.
- Helps capture local/short-term patterns.

Output Patch Design

- Output patch length = h | (h > p)

Key idea:

- Predict multiple future time steps at once

Why?

- Prior work shows full-horizon prediction is more accurate
- Reduces number of autoregressive steps

Tradeoff:

- Horizon unknown → use larger output patches instead of full prediction

Time-Series Specific Design: Patch Masking

Problem:

- Model would only learn context lengths that are multiples of p

Solution:

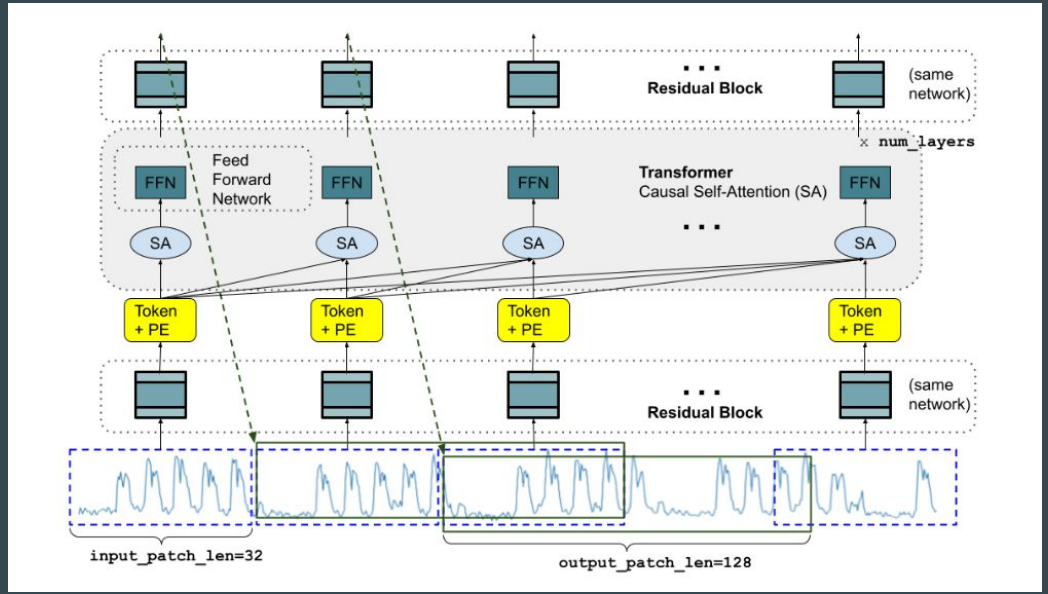
- Randomly mask parts of patches, starting from the beginning, during training

Result:

- Model learns all possible context lengths
- Improves generalization

Model Architecture

- Input Layers (patch embedding)
- Stacked Transformer Layers
- Output Layers (prediction)
- Loss Function
- Training Strategy
- Inference Procedure



Model Architecture: Input Layers

We pass in input: $y_{1:L}$ and binary padding mask: $m_{1:L}$

- 1 in $m_{1:L}$ designates that corresponding input should be ignored

Input Layers

- Time-series split into patches
- Each patch processed by a residual block
- The Residual Block is just a Multi-layer Perceptron (MLP) block with one hidden layer and a skip connection.
- Converted into embedding vectors
- Positional encoding added

Output:

- Sequence of tokens for transformer: t_j

$y_{1:L}$ for patch j is: $\tilde{y}_j = y_{p(j-1)+1:pj}$

Similar for $\tilde{m}_j$

$$t_j = \text{InputResidualBlock}(\tilde{y}_j \square (1 - \tilde{m}_j)) + PE_j$$

Model Architecture: Stacked Transformers

Stacked Transformer Layers

- Multi-head self-attention
- Causal attention (only attends to past)
- Feedforward layers

$o_j = \text{StackedTransformer}((t_1, \dot{m}_1), \cdots, (t_j, \dot{m}_j))$

- $\dot{m}_j$ is the masking indicator
- If the patch has any non-masked time-points, the corresponding token isn't masked
- Fully masked patches aren't utilized

Purpose:

- Learn temporal dependencies across patches

Model Architecture: Output Layers

Output Layers

- Each transformer output token is mapped to predictions
- Uses a residual block

$$\hat{y}_{pj+1:pj+h} = \text{OutputResidualBlock}(o_j)$$

Key idea:

- Each token predicts h future time steps
- h being the output patch size

Model Architecture: Loss Function

Loss Function

- Mean Squared Error (MSE)

$$\text{TrainLoss} = \frac{1}{N}\sum_{j=1}^{N} \text{MSE}\left(\hat{y}_{pj+1:pj+h},\ y_{pj+1:pj+h}\right)$$

Why?

- Standard for point forecasting

$$\text{MSE}(\hat{y}, y) = \frac{1}{h}\sum_{i=1}^{h}\left(y_{pj+i} - \hat{y}_{pj+i}\right)^2$$

Extension:

- Can support probabilistic forecasting (quantile loss)

Model Architecture: Training

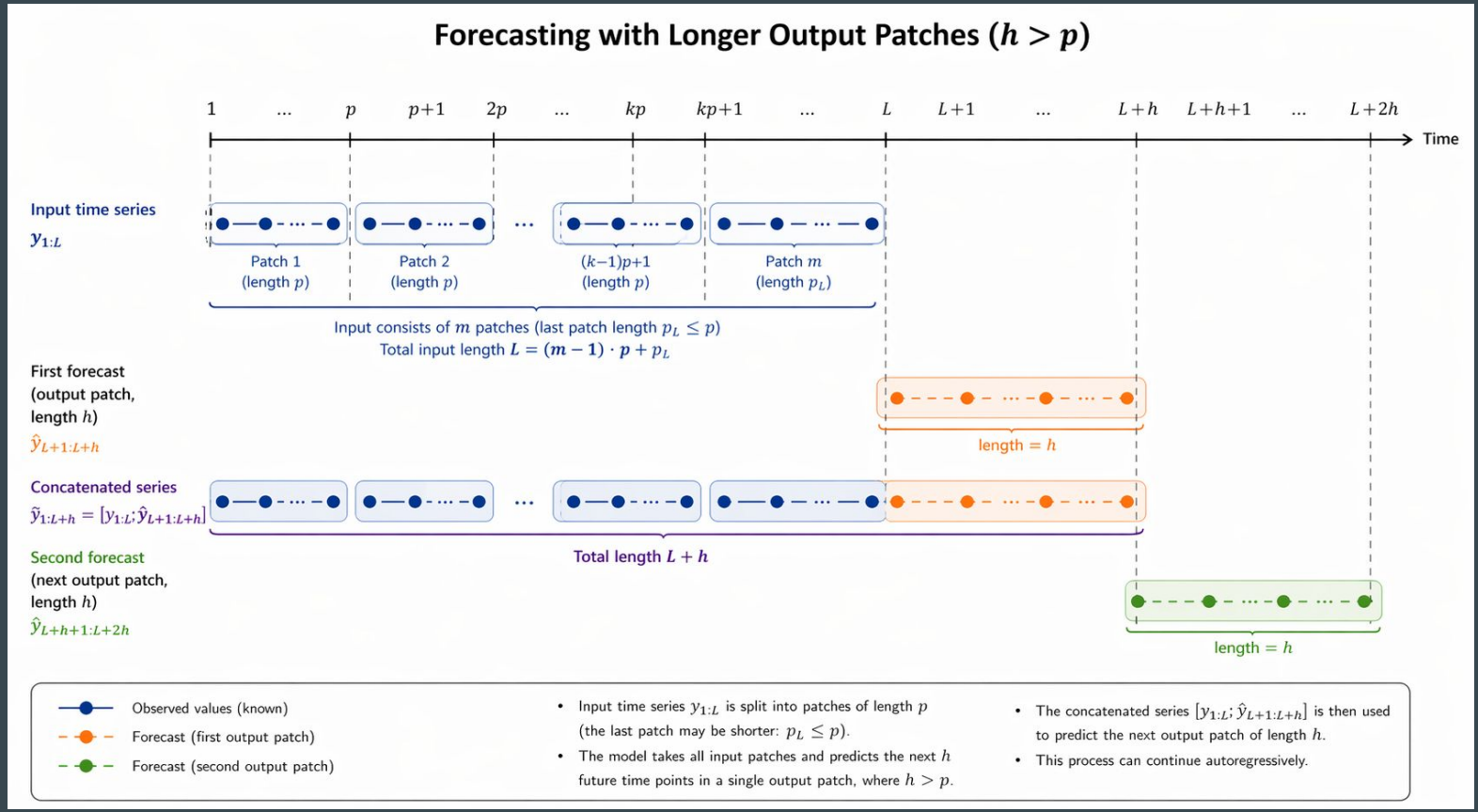
Training Strategy

- Mini-batch gradient descent
- Decoder-only training
- Random patch masking (to ensure learning of different context lengths)

Key insight:

- Masking allows training on all possible context lengths

Model Architecture: Inference



Pretraining and Dataset information

- Datasets
  - Google Trends
  - Wiki pageviews
  - Synthetic Data
  - Other real-world data (M4 dataset, hourly and 15min Electricity and hourly traffic dataset)
  - 10 min granularity Weather dataset
- We train on a mixture of these datasets to optimize performance

Metrics

Evaluation Metrics

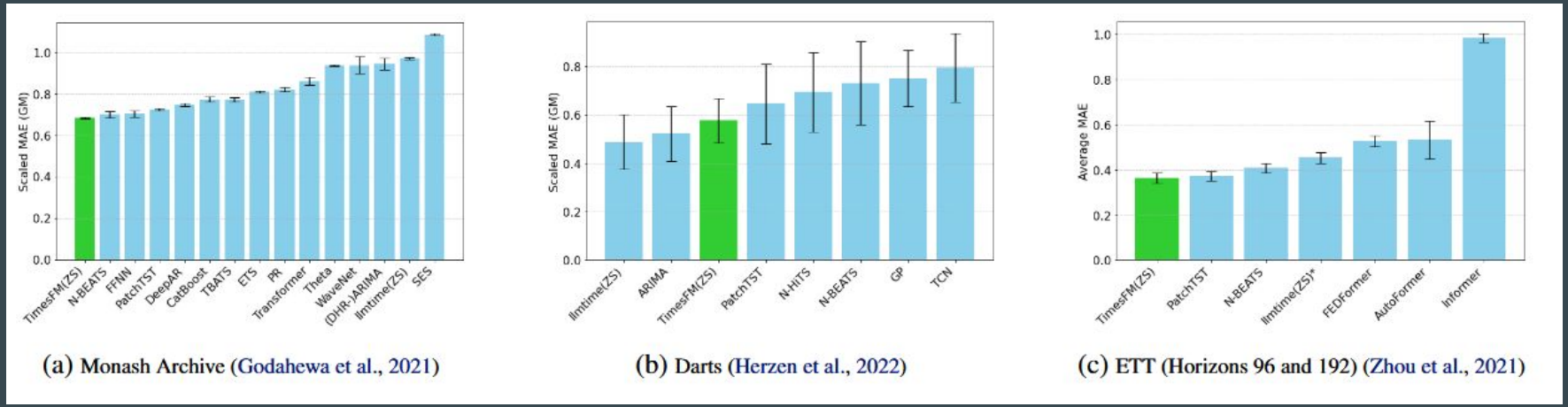
- MAE (Mean Absolute Error)
- Scaled MAE (normalized across datasets)

Aggregation:

- Geometric mean used for robustness

Results

- Achieves a lower error than other models
- Zero-shot performance on three new unseen dataset
  - Monash, Darts, and Informer dataset
- No retraining, fine-tuning, dataset-specific training



(a) Monash Archive (Godahewa et al., 2021)

(b) Darts (Herzen et al., 2022)

(c) ETT (Horizons 96 and 192) (Zhou et al., 2021)

Ablation Studies

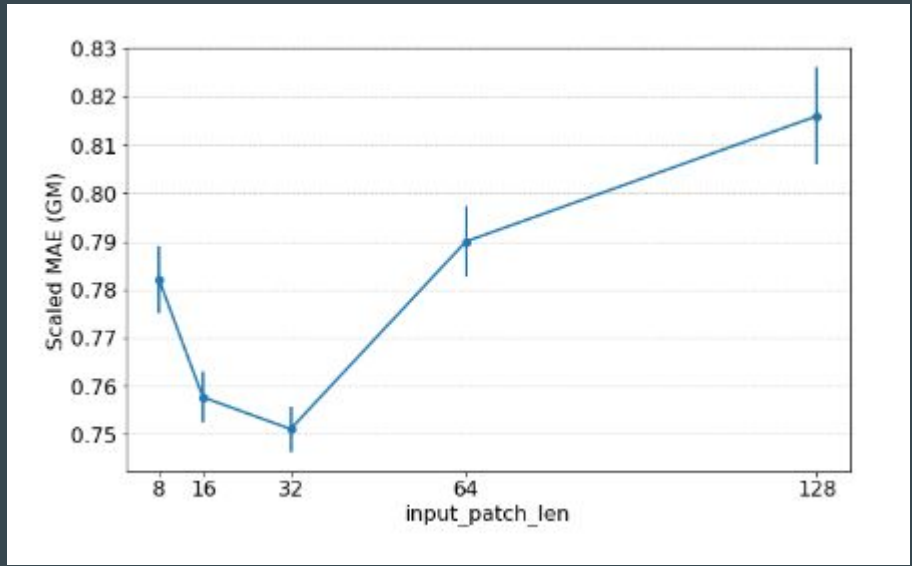
Removing and altering specific aspects and components of the model and seeing if they are needed and provide meaningful performance

- Model Scaling
- Autoregressive decoding
- Input patch length
- Ablation on the datasets

Ablation Studies: Input Patch Length

Input Patch Length Ablation

- Best performance around p = 16–32
- Too small → inefficient
- Too large → loss of detail



Tradeoff:

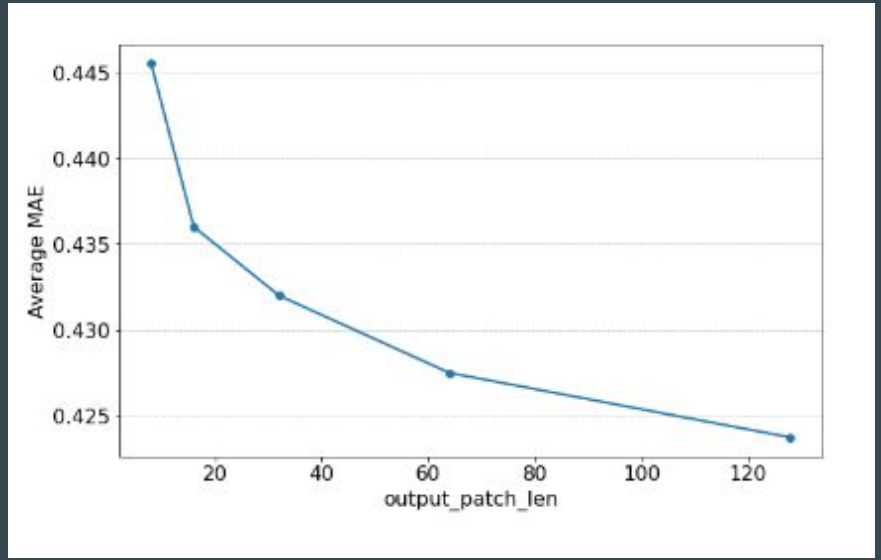
Efficiency vs accuracy

Ablation Studies: Auto-regressive Decoding

How important is the output patch length being greater than input?

To test this:

- Predict 512 time-steps into the future for the ETT datasets
- Test on different output patch lengths
- We see average MAE decrease as output patch length increases



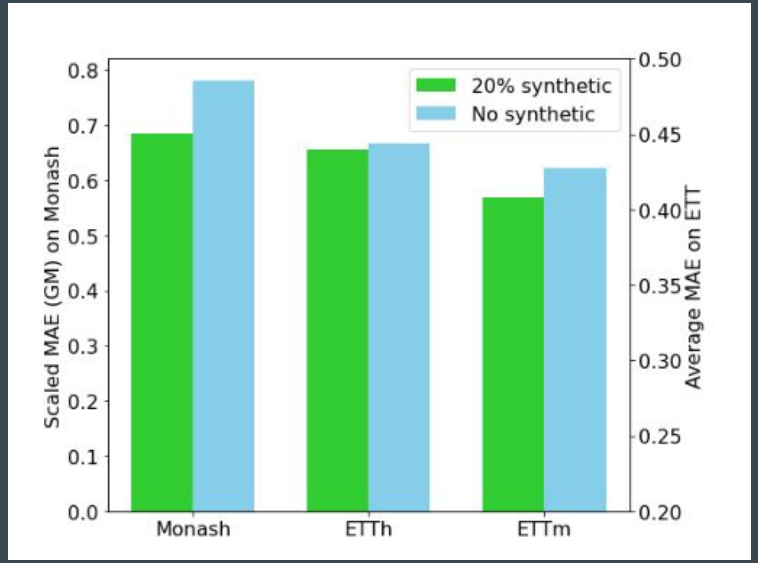
Ablation Studies: Dataset Ablation

Dataset Ablation

- Removing synthetic data reduces performance

Key insight:

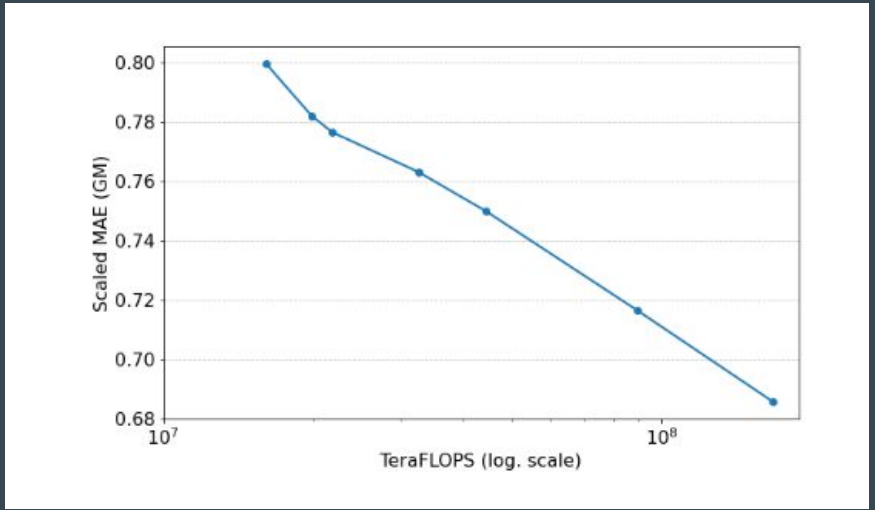
- Synthetic data improves generalization to rare patterns including:
  - Rare periodic patterns
  - Across frequencies
  - Patterns not present in real data



Ablation Studies: Scaling

Scaling Results

- Larger models → better performance
- Follows LLM-style scaling laws



Example:

17M → 70M → 200M parameters → decreasing error

Strengths and shortcomings of the paper

Strengths:

- Strong zero-shot generalization
- Efficient architecture
- Competitive performance
- Strong empirical results

Shortcomings:

- Limited interpretability and real-world deployment analysis
- No covariate modeling
- Requires large pre-training data

Key Points and Future Directions

- First practical time-series foundation model
- Achieves strong zero-shot forecasting
- Demonstrates generalization across datasets

Future directions:

- Probabilistic forecasting
- Covariate integration
- Prompt tuning for time-series

Questions?

References

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. International conference on learning representations, 2022.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? Proceedings of the AAAI conference on artificial intelligence, 2023.

Herzen, J., L¨assig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Van Pottelbergh, T., Pasieka, M., Skrodzki, A., Huguenin, N., et al. Darts: User-friendly modern machine learning for time series. The Journal of Machine Learning Research, 23(1):5442–5447, 2022.

Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., and Montero-Manso, P. Monash time series forecasting archive. arXiv preprint arXiv:2105.06643, 2021.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI conference on artificial intelligence, 2021.