UNDER REVIEW

RESCIENCE C

Editorial

[Re] A Time Series is Worth 64 Words: Long-term Forecasting with Transformers

Bernardo Perrone Ribeiro[1,2] **and Camile Lendering**[1,2]

[1]Faculty of Computer and Information Science, University of Ljubljana, Večna pot 113, Ljubljana 1000, Slovenia – [2]Universitat Pompeu Fabra, Tànger, 122-140 08018 Barcelona

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
05 May 2025

Published
–

DOI
–

1 Introduction

Time series forecasting remains an important yet challenging task for machine learning models due to the inherent complexity of capturing temporal dependencies and handling long-range interactions efficiently. While Transformer-based approaches have significantly advanced time series forecasting, their computational complexity and inherent permutation invariance in self-attention may limit their sensitivity to temporal locality, potentially offsetting their benefits. [1] Nie et al. (2023) proposed PatchTST, a model that addresses these limitations by segmenting multivariate time series into subseries-level patches and utilizing a channel-independent Transformer architecture. By processing time series data as localized patches, PatchTST retains semantic information, reduces computational complexity, and allows modeling of longer historical sequences, achieving superior forecasting accuracy compared to previous state-of-the-art Transformer-based methods and linear models. [2]

In this replication study, we aim to reproduce and validate the claimed advantages of PatchTST, particularly its efficiency in handling longer historical data and its accuracy in multivariate time series forecasting tasks. Furthermore, we investigate its generalizability by applying self-supervised pre-training and transfer learning methods as described in the original paper. This replication will provide insights into the reproducibility and reliability of PatchTST's reported improvements and evaluate its practical effectiveness across different forecasting datasets.

2 Replication Summary

We developed an independent implementation of PatchTST based on the descriptions in the original paper [2], while using the provided code base as a reference in areas where the paper did not provide sufficient detail. Our implementation largely follows the experimental protocols described in the original work, including dataset selection, hyperparameter settings, and evaluation metrics (MSE and MAE). Since the authors conducted extensive experiments, including with larger model variants, we occasionally had to reduce the batch size (e.g., in ablation studies) to fit within our computational constraints (NVIDIA H100 80GB VRAM). Overall, we successfully reproduced the main supervised forecasting results for most datasets, but encountered difficulties replicating the ILI dataset results and the self-supervised/transfer learning performance reported in the original paper. Our code implementation is publicly available.

Correspondence should be addressed to Camile Lendering (cl53387@student.uni-lj.si)
The authors have declared that no competing interests exists.
Code is available at https://github.com/b-rbmp/PatchTSTReproduction.

3 Methods

The original paper tackles the problem of multivariate time-series forecasting, where given a collection of multivariate time series samples with lookback window L : $(x_1, ..., x_L)$ where each x_t at time step t is a vector of dimension M, the goal is to forecast T future values $(x_{L+1}, ..., x_{L+T})$.

3.1 PatchTST Structure

The model uses a vanilla Transformer encoder as the backbone of the network, with the following main characteristics:

- **Channel-Independence during the Forward Process**: The input $(x_1, ..., x_L)$ is split into M univariate series $x^{(i)} \in \mathbb{R}^{1\times L}$ and each of them is passed independently through the Transformer encoder, which outputs predictions $\hat{x}^{(i)} = (\hat{x}^{(i)}_{L+1}, ..., \hat{x}^{(i)}_{L+T}) \in \mathbb{R}^{1\times T}$.
- **Patching and Attention Complexity Reduction**: Each input univariate time series $x^{(i)}$ is divided into patches, which reduces the number of input tokens from L to approximately L/S, where L is lookback window and S the stride. This represents a quadractic reduction in the memory usage and computational complexity of the attention map by a factor of S.
- **Vanilla Transformer Encoder**: The architecture uses a vanilla Transformer encoder that maps the observed signals to latent representations. The patches are first projected into a latent space of dimension D via a trainable linear projection $\mathbf{W}_p \in \mathbb{R}^{D\times P}$ and a learnable additive position encoding $\mathbf{W}_{\text{pos}} \in \mathbb{R}^{D\times N}$ that is applied to maintain the temporal order of the patches:

$$\mathbf{x}_d^{(i)} \;=\; \mathbf{W}_p\,\mathbf{x}_p^{(i)} \;+\; \mathbf{W}_{\text{pos}},$$

where $\mathbf{x}_d^{(i)} \in \mathbb{R}^{D\times N}$ is then fed into the Transformer encoder.

Next, each head $h = 1, \ldots, H$ in the multi-head attention module transforms $\mathbf{x}_d^{(i)}$ into query, key, and value matrices:

$$\mathbf{Q}_h^{(i)} \;=\; \big(\mathbf{x}_d^{(i)}\big)^T\mathbf{W}_h^Q, \quad \mathbf{K}_h^{(i)} \;=\; \big(\mathbf{x}_d^{(i)}\big)^T\mathbf{W}_h^K, \quad \mathbf{V}_h^{(i)} \;=\; \big(\mathbf{x}_d^{(i)}\big)^T\mathbf{W}_h^V,$$

where $\mathbf{W}_h^Q, \mathbf{W}_h^K \in \mathbb{R}^{D\times d_k}$ and $\mathbf{W}_h^V \in \mathbb{R}^{D\times D}$.

A scaled dot-product attention then produces the output $\mathbf{O}_h^{(i)} \in \mathbb{R}^{D\times N}$. Concretely,

$$\big(\mathbf{O}_h^{(i)}\big)^T \;=\; \text{Attention}\Big(\mathbf{Q}_h^{(i)}, \mathbf{K}_h^{(i)}, \mathbf{V}_h^{(i)}\Big) \;=\; \text{Softmax}\Big(\tfrac{\mathbf{Q}_h^{(i)}\,\mathbf{K}_h^{(i)\top}}{\sqrt{d_k}}\Big)\,\mathbf{V}_h^{(i)}.$$

Each multi-head attention block also includes BatchNorm layers and a feed-forward network with residual connections. After processing, the Transformer outputs a representation $\mathbf{z}^{(i)} \in \mathbb{R}^{D\times N}$. Finally, a flatten operation followed by a linear head produces the prediction

$$\hat{\mathbf{x}}^{(i)} \;=\; \big(\hat{x}^{(i)}_{L+1}, \ldots, \hat{x}^{(i)}_{L+T}\big) \;\in\; \mathbb{R}^{1\times T}.$$

- **Multi-Channel Loss Function**: Mean Squared Error is used as the loss, measuring the difference between the prediction and the ground truth. The overall objective averages the loss in each channel over the M univariate series, such that:

$$\mathcal{L} \;=\; \mathbb{E}_x\left[\frac{1}{M}\sum_{i=1}^{M}\Big\|\hat{x}^{(i)}_{L+1:L+T} \;-\; x^{(i)}_{L+1:L+T}\Big\|^2\right].$$

- **Instance Normalization**: To mitigate the distribution shift effect between training and testing data, Instance Normalization [3, 4] is performed by normalizing each time series $x^{(i)}$ with zero mean and unit standard deviation before patching, and then de-normalizing the output prediction.

3.2 Representation Learning

Self-supervised representation learning extracts high-level features from unlabeled data. In the paper, PatchTST is used to learn representations for multivariate time series that can be transferred to forecasting tasks. The approach uses a masked autoencoder framework, successful in natural language [5] and computer vision [6], by randomly removing parts of the input sequence and training the model to recover them.

The authors note that earlier methods, which mask individual time steps, risk trivial interpolation and overfitting when mapping a latent representation of size $L \times D$ to an output of size $M \times T$. In contrast, PatchTST mitigates these issues by dividing the input into regular, non-overlapping patches. The Transformer encoder (the same as in the supervised setting) is used without its prediction head; instead, a $D \times P$ projection layer is added. A random subset of patch indices is masked (set to zero), and the network is trained with MSE loss to reconstruct the missing patches. This design supports a unified latent representation even when the number of time series differs between pre-training and downstream tasks.

4 Experiments

We repeated all experiments performed by the original authors, additionally bootstrapping the experiments with a number of experiments of 5 (reporting the average metrics), except for the Ablation Study, due to computational constraints. Performance was evaluated on 8 time series datasets commonly used for benchmarking, namely Weather, Traffic, Electricity, ILI, and 4 ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2). The number of features and time steps for each data set can be seen in Table 1.

Table 1. Summary of benchmarking datasets.

Datasets	Weather	Traffic	Elect.	ILI	ETTh1	ETTh2	ETTm1	ETTm2
Features	21	862	321	7	7	7	7	7
Timesteps	52696	17544	26304	966	17420	17420	69680	69680

Two versions of PatchTST are employed, PatchTST/64 with 64 input patches and look-back window of 512 and PatchTST/42 with 42 input patches and look-back window of 336. We split the experiments into five groups:

1. Supervised Training: PatchTST is tasked with the traditional long-term supervised time series forecasting task, with prediction length $T \in \{24, 36, 48, 60\}$ for the ILI dataset and $T \in \{96, 192, 336, 720\}$ for the other datasets. Both versions of PatchTST use a patch length of 16 and a stride of 8.
2. Self-Supervised: PatchTST undergoes masked self-supervised pre-training with non-overlapping patches, a masking ratio of 40%, input sequence of 512 and patch size of 12, resulting in 42 patches. After pre-training for 100 epochs, we perform supervised training with the same predictions lengths as the previous section applying two techniques: linear probing and end-to-end fine-tuning. With linear probing, the model head is trained for 20 epochs while freezing all other layers, and with end-to-end fine-tuning, we apply linear probing for 10 epochs, and then

unfreezing all other layers and training for 20 more epochs. The self-supervised experiments are performed with the Weather, Traffic and Electricity datasets

3. Transfer Learning: The same procedure of the Self-Supervised experiments is performed, but pre-training on the Electricity dataset and fine-tuning on the Traffic and Weather datasets with the same two techniques.
4. Representation Learning on ETTh1: Following the original paper, the self-supervised pre-training and transfer learning protocols (as described in items 2 and 3) were specifically repeated for the ETTh1 dataset. For the transfer learning part on ETTh1, the model was pre-trained on the Traffic dataset.
5. Ablation Study: We verify the same ablation studies as shown in the original paper, such as observing the effects of patching and channel-independence by testing the model without both, with only patching and with only channel-independence, and also the effect of different look-back windows ({24, 48, 96, 192, 336, 720}) on two different prediction horizons ({96, 720}). The patching and channel-independence ablations are run with a reduced number of epochs of 20 due to the time complexity when patching is disabled for the Traffic and Electricity datasets.

All experiments were conducted on NVIDIA H100 GPUs, with 80GB of VRAM.

5 Results

In this section, the results for all the experiments conducted in the original paper are presented in tables, where the difference between the results found in the original paper and this reproduction are shown next to each value. Reported values are the average taken with 5 bootstrapped experiments.

5.1 Supervised Training

Results for the Supervised Training experiments can be seen in Table 2. For almost all datasets, the results fall in line with the original paper, with some variability in the last decimal point. The exception is ILI, where due to the different prediction lengths, an adjustment has to be made for the input length and also the patch size and stride to keep the same number of patches of PatchTST/64 and PatchTST/42. Surprisingly, we were unable to find any details regarding this change in the original paper, but we did find a training script on the official repository, which sets the input length to 104, patch size to 24 and stride to 2. Nevertheless, our model performed significantly worse than their findings, which could imply that their training procedure is different from the one made available, as many commentors have pointed out on the GitHub repository, also finding it difficult to reproduce these results.

Table 2. Multivariate long-term forecasting results with supervised PatchTST/64 and PatchTST/42. All values are shown as *Mean (Δ = Change from the original paper)*.

Dataset	Horizon	PatchTST/64		PatchTST/42	
		MAE	MSE	MAE	MSE
Weather					
	96	0.202 ($\Delta = +0.004$)	0.151 ($\Delta = +0.002$)	0.204 ($\Delta = +0.005$)	0.155 ($\Delta = +0.003$)
	192	0.242 ($\Delta = +0.001$)	0.195 ($\Delta = +0.001$)	0.243 ($\Delta = 0$)	0.198 ($\Delta = +0.001$)
	336	0.283 ($\Delta = +0.001$)	0.246 ($\Delta = +0.001$)	0.284 ($\Delta = +0.001$)	0.250 ($\Delta = +0.001$)
	720	0.332 ($\Delta = -0.002$)	0.312 ($\Delta = -0.002$)	0.335 ($\Delta = 0$)	0.319 ($\Delta = -0.001$)
Traffic					
	96	0.249 ($\Delta = 0$)	0.360 ($\Delta = 0$)	0.250 ($\Delta = -0.001$)	0.367 ($\Delta = 0$)
	192	0.256 ($\Delta = 0$)	0.379 ($\Delta = 0$)	0.258 ($\Delta = -0.001$)	0.386 ($\Delta = +0.001$)
	336	0.264 ($\Delta = 0$)	0.391 ($\Delta = -0.001$)	0.265 ($\Delta = 0$)	0.398 ($\Delta = 0$)
	720	0.293 ($\Delta = +0.007$)	0.438 ($\Delta = +0.006$)	0.285 ($\Delta = -0.002$)	0.432 ($\Delta = -0.002$)
Electricity					
	96	0.223 ($\Delta = +0.001$)	0.129 ($\Delta = 0$)	0.223 ($\Delta = +0.001$)	0.130 ($\Delta = +0$)
	192	0.242 ($\Delta = +0.002$)	0.148 ($\Delta = +0.001$)	0.241 ($\Delta = +0.001$)	0.148 ($\Delta = 0$)
	336	0.259 ($\Delta = 0$)	0.163 ($\Delta = 0$)	0.259 ($\Delta = 0$)	0.165 ($\Delta = +0.002$)
	720	0.291 ($\Delta = +0.001$)	0.199 ($\Delta = +0.002$)	0.293 ($\Delta = +0.002$)	0.203 ($\Delta = +0.001$)
ILI					
	24	0.966 ($\Delta = +0.353$)	2.001 ($\Delta = +0.682$)	0.904 ($\Delta = +0.090$)	1.843 ($\Delta = +0.321$)
	36	0.971 ($\Delta = +0.101$)	1.975 ($\Delta = +0.396$)	0.960 ($\Delta = +0.126$)	1.946 ($\Delta = +0.516$)
	48	1.030 ($\Delta = +0.215$)	2.210 ($\Delta = +0.657$)	1.007 ($\Delta = +0.153$)	2.142 ($\Delta = +0.469$)
	60	0.974 ($\Delta = +0.186$)	2.082 ($\Delta = +0.612$)	0.985 ($\Delta = +0.123$)	2.054 ($\Delta = +0.525$)
ETTh1					
	96	0.409 ($\Delta = +0.009$)	0.380 ($\Delta = +0.010$)	0.413 ($\Delta = +0.014$)	0.391 ($\Delta = +0.016$)
	192	0.433 ($\Delta = +0.004$)	0.417 ($\Delta = +0.004$)	0.430 ($\Delta = +0.009$)	0.423 ($\Delta = +0.009$)
	336	0.442 ($\Delta = +0.002$)	0.423 ($\Delta = +0.001$)	0.436 ($\Delta = 0$)	0.430 ($\Delta = -0.001$)
	720	0.465 ($\Delta = -0.003$)	0.443 ($\Delta = -0.004$)	0.461 ($\Delta = -0.005$)	0.441 ($\Delta = -0.008$)
ETTh2					
	96	0.339 ($\Delta = +0.002$)	0.275 ($\Delta = +0.001$)	0.342 ($\Delta = +0.006$)	0.282 ($\Delta = +0.008$)
	192	0.382 ($\Delta = 0$)	0.341 ($\Delta = 0$)	0.384 ($\Delta = +0.005$)	0.344 ($\Delta = +0.005$)
	336	0.386 ($\Delta = +0.002$)	0.330 ($\Delta = +0.001$)	0.388 ($\Delta = +0.008$)	0.333 ($\Delta = +0.002$)
	720	0.423 ($\Delta = +0.001$)	0.378 ($\Delta = -0.001$)	0.422 ($\Delta = 0$)	0.382 ($\Delta = +0.003$)
ETTm1					
	96	0.346 ($\Delta = 0$)	0.293 ($\Delta = 0$)	0.342 ($\Delta = 0$)	0.289 ($\Delta = -0.001$)
	192	0.371 ($\Delta = +0.001$)	0.335 ($\Delta = +0.002$)	0.370 ($\Delta = +0.001$)	0.333 ($\Delta = +0.001$)
	336	0.391 ($\Delta = -0.001$)	0.364 ($\Delta = -0.005$)	0.393 ($\Delta = +0.001$)	0.368 ($\Delta = +0.002$)
	720	0.421 ($\Delta = +0.001$)	0.412 ($\Delta = -0.004$)	0.424 ($\Delta = 0$)	0.419 ($\Delta = -0.001$)
ETTm2					
	96	0.256 ($\Delta = 0$)	0.166 ($\Delta = 0$)	0.254 ($\Delta = -0.001$)	0.165 ($\Delta = 0$)
	192	0.298 ($\Delta = +0.002$)	0.222 ($\Delta = -0.001$)	0.294 ($\Delta = +0.002$)	0.222 ($\Delta = +0.002$)
	336	0.329 ($\Delta = 0$)	0.278 ($\Delta = +0.004$)	0.329 ($\Delta = 0$)	0.277 ($\Delta = -0.001$)
	720	0.383 ($\Delta = -0.002$)	0.360 ($\Delta = -0.002$)	0.383 ($\Delta = -0.002$)	0.365 ($\Delta = -0.002$)

5.2 Self-Supervised

For self-supervised linear probing and end-to-end fine-tuning experiments, our implementation of PatchTST/42 performed somewhat worse (10-30%) than the reported results, while the same was not observed in the supervised version. In fact, differently from their findings, the supervised training consistently beat the two self-supervised training procedures. We believe that the number of epochs for pre-training (100) might not have been enough, as the models were still improving their validation losses before

they were stopped and the finetuning stage was started.

Table 3. Multivariate long-term forecasting results with self-supervised PatchTST/42. All values are shown as *Mean (Δ = Change from the original paper)*.

Dataset	Horizon	Finetuning		Linear Probing		Supervised	
		MAE	MSE	MAE	MSE	MAE	MSE
Weather							
	96	0.216 (Δ = +0.023)	0.156 (Δ = +0.012)	0.225 (Δ = +0.016)	0.163 (Δ = +0.005)	0.204 (Δ = +0.005)	0.155 (Δ = +0.003)
	192	0.256 (Δ = +0.020)	0.202 (Δ = +0.012)	0.259 (Δ = +0.010)	0.205 (Δ = +0.002)	0.243 (Δ = 0)	0.198 (Δ = +0.001)
	336	0.292 (Δ = +0.012)	0.251 (Δ = +0.007)	0.292 (Δ = +0.007)	0.251 (Δ = 0)	0.284 (Δ = +0.001)	0.250 (Δ = +0.001)
	720	0.344 (Δ = +0.009)	0.327 (Δ = +0.007)	0.343 (Δ = +0.007)	0.322 (Δ = +0.001)	0.335 (Δ = 0)	0.319 (Δ = −0.001)
Traffic							
	96	0.356 (Δ = +0.112)	0.527 Δ = +0.175	0.360 (Δ = +0.066)	0.540 (Δ = +0.141)	0.250 (Δ = −0.001)	0.367 (Δ = 0)
	192	0.360 (Δ = +0.107)	0.529 (Δ = +0.158)	0.363 (Δ = +0.065)	0.542 (Δ = +0.130)	0.258 (Δ = −0.001)	0.386 (Δ = +0.001)
	336	0.369 (Δ = +0.112)	0.564 (Δ = +0.183)	0.367 (Δ = +0.061)	0.559 (Δ = +0.134)	0.265 (Δ = 0)	0.398 (Δ = 0)
	720	0.500 (Δ = +0.218)	0.780 (Δ = +0.355)	0.389 (Δ = +0.083)	0.634 (Δ = +0.209)	0.285 (Δ = −0.002)	0.432 (Δ = −0.002)
Electricity							
	96	0.286 (Δ = +0.065)	0.165 (Δ = +0.039)	0.291 (Δ = +0.054)	0.170 (Δ = +0.032)	0.223 (Δ = +0.001)	0.130 (Δ = 0)
	192	0.299 (Δ = +0.061)	0.182 (Δ = +0.037)	0.302 (Δ = +0.050)	0.185 (Δ = +0.029)	0.241 (Δ = +0.001)	0.148 (Δ = 0)
	336	0.314 (Δ = +0.058)	0.199 (Δ = +0.035)	0.322 (Δ = +0.057)	0.207 (Δ = +0.037)	0.259 (Δ = −0.002)	0.165 (Δ = −0.002)
	720	0.347 (Δ = +0.056)	0.241 (Δ = +0.077)	0.356 (Δ = +0.059)	0.251 (Δ = +0.043)	0.293 (Δ = +0.002)	0.203 (Δ = +0.001)

5.3 Transfer-Learning

We ran transfer learning experiments by pretraining PatchTST/42 on the Electricity dataset, then fine-tuning or applying linear probing on the Weather and Traffic datasets. Table 4 shows the results compared to both the original paper and our supervised baseline.

For the Weather dataset, our transfer learning results are close to those reported in [2], with only small differences (Δ). This suggests that transfer from Electricity to Weather works reasonably well. Still, as with our self-supervised experiments, both fine-tuning and linear probing gave higher MAE and MSE than our own supervised training for Weather (see Table 4).

On the Traffic dataset, results were less consistent. We saw much larger differences from the original paper's numbers, with Δ values notably higher for both fine-tuning and linear probing. Errors in our runs were higher, and transfer learning performed worse than our supervised baseline. This suggests that, at least in our setup, transfer learning didn't help when moving from Electricity to the more complex Traffic dataset, which has higher dimensionality and variability.

Table 4. Multivariate long-term forecasting results with transfer-learning PatchTST/42. All values are shown as *Mean (Δ = Change from the original paper)*.

Dataset	Horizon	Finetuning		Linear Probing		Supervised	
		MAE	MSE	MAE	MSE	MAE	MSE
Weather							
	96	0.216 (Δ = +0.021)	0.155 (Δ = +0.010)	0.221 (Δ = +0.005)	0.160 (Δ = +0.003)	0.204 (Δ = +0.005)	0.155 (Δ = +0.003)
	192	0.255 (Δ = +0.012)	0.198 (Δ = +0.005)	0.258 (Δ = +0.006)	0.202 (Δ = −0.003)	0.243 (Δ = 0)	0.198 (Δ = +0.001)
	336	0.293 (Δ = +0.013)	0.248 (Δ = +0.004)	0.296 (Δ = +0.007)	0.251 (Δ = −0.002)	0.284 (Δ = +0.001)	0.250 (Δ = +0.001)
	720	0.344 (Δ = +0.007)	0.320 (Δ = −0.001)	0.347 (Δ = +0.011)	0.326 (Δ = +0.006)	0.335 (Δ = 0)	0.319 (Δ = −0.001)
Traffic							
	96	0.355 (Δ = +0.082)	0.501 (Δ = +0.113)	0.421 (Δ = +0.133)	0.610 (Δ = +0.210)	0.250 (Δ = −0.001)	0.367 (Δ = 0)
	192	0.367 (Δ = +0.090)	0.531 (Δ = +0.131)	0.440 (Δ = +0.147)	0.644 (Δ = +0.232)	0.258 (Δ = −0.001)	0.386 (Δ = +0.001)
	336	0.370 (Δ = +0.090)	0.549 (Δ = +0.141)	0.419 (Δ = +0.112)	0.638 (Δ = +0.213)	0.265 (Δ = 0)	0.398 (Δ = 0)
	720	0.484 (Δ = +0.174)	0.754 (Δ = +0.307)	0.439 (Δ = +0.122)	0.668 (Δ = +0.211)	0.285 (Δ = −0.002)	0.432 (Δ = −0.002)

5.4 Representation Learning

This section reports results for representation learning on the ETTh1 dataset. We evaluate both self-supervised learning (pretraining and fine-tuning/probing on ETTh1) and

transfer learning (pretraining on Traffic, then fine-tuning/probing on ETTh1). Full results are shown in Table 5.

For the self-supervised setting, our results show small positive deviations (Δ) from those reported in the original paper across all horizons. While slightly worse overall, the reproduced results are generally consistent with the original.

In contrast, transfer learning (Traffic $\rightarrow$ ETTh1) shows larger discrepancies, especially for short horizons ($T \in 24, 48$), where both MAE and MSE are noticeably higher. The deviations are smaller at longer horizons, indicating better agreement with the original findings in those cases.

Table 5. Multivariate long-term forecasting results with self-supervised and transfer-learning PatchTST/42. All values are shown as *Mean (Δ = Change from the original paper)*.

Dataset	Horizon	Transfer-Learning		Self-Supervised	
		MAE	MSE	MAE	MSE
ETTh1					
	24	0.400 ($\Delta = +0.038$)	0.356 ($\Delta = +0.044$)	0.380 ($\Delta = +0.011$)	0.328 ($\Delta = 0$)
	48	0.414 ($\Delta = +0.036$)	0.381 ($\Delta = +0.042$)	0.406 ($\Delta = +0.021$)	0.373 ($\Delta = +0.019$)
	168	0.448 ($\Delta = +0.011$)	0.432 ($\Delta = +0.008$)	0.447 ($\Delta = +0.023$)	0.431 ($\Delta = +0.012$)
	336	0.457 ($\Delta = -0.015$)	0.444 ($\Delta = -0.028$)	0.463 ($\Delta = +0.017$)	0.452 ($\Delta = +0.007$)
	720	0.492 ($\Delta = -0.015$)	0.481 ($\Delta = -0.027$)	0.492 ($\Delta = +0.014$)	0.488 ($\Delta = +0.001$)

5.5 Uni-variate Supervised Training

We also reproduced the univariate forecasting experiments on the four ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2), focusing on predicting the 'oil temperature' (OT) target variable. Table 6 presents our reproduction results alongside the difference compared to the results reported in the original paper. The reproduction accuracy aligns closely with the original results, with most deviations being minor.

Comparison with NLinear — While the original PatchTST paper included comparisons to linear models like DLinear [1], it did not report on NLinear, which often performs better on univariate forecasting tasks. To provide additional context for our univariate results, we compare PatchTST with NLinear using results reported in [1]. Table 7 shows this comparison on the ETT datasets.

As shown, NLinear often matches or outperforms both PatchTST/64 and PatchTST/42, especially at longer horizons on ETTh1 and consistently across horizons on ETTm2. This highlights that, in univariate settings, simpler linear models can rival or surpass Transformer-based approaches.

5.6 Ablation Study

We conducted two sets of ablation experiments following the original paper: one investigating the contributions of Patching (P) and Channel-Independence (CI), and another examining the effect of varying look-back window lengths (L).

Patching and Channel-Independence Effects — To assess the impact of Patching (P) and Channel Independence (CI), we ran ablation experiments by selectively removing these components and comparing the results to the full PatchTST/42 model. For a fair comparison, all models in Table 8, including the full version (P+CI), were trained for only 20 epochs. This is a reduced training budget compared to the 100 epochs used for the main supervised results in Table 2. It's worth noting that the original paper [2] may have compared

Table 6. Reproduction results for PatchTST univariate forecasting. For each dataset and forecast horizon, the table shows the MSE and MAE from our reproduction along with the difference (Δ = reproduction - original).

Dataset	Horizon	PatchTST/64		PatchTST/42	
		MSE	MAE	MSE	MAE
ETTh1					
	96	0.0601 ($\Delta = +0.001$)	0.1904 ($\Delta = +0.001$)	0.056 ($\Delta = +0.001$)	0.181 ($\Delta = +0.002$)
	192	0.077 ($\Delta = +0.003$)	0.218 ($\Delta = +0.003$)	0.072 ($\Delta = +0.001$)	0.207 ($\Delta = +0.002$)
	336	0.081 ($\Delta = -0.005$)	0.218 ($\Delta = +0.002$)	0.086 ($\Delta = +0.005$)	0.231 ($\Delta = +0.006$)
	720	0.097 ($\Delta = -0.01$)	0.245 ($\Delta = -0.009$)	0.099 ($\Delta = -0.01$)	0.247 ($\Delta = -0.02$)
ETTh2					
	96	0.137 ($\Delta = +0.006$)	0.290 ($\Delta = +0.006$)	0.136 ($\Delta = +0.007$)	0.289 ($\Delta = +0.007$)
	192	0.180 ($\Delta = +0.009$)	0.338 ($\Delta = +0.009$)	0.175 ($\Delta = +0.007$)	0.333 ($\Delta = +0.002$)
	336	0.182 ($\Delta = +0.011$)	0.347 ($\Delta = +0.011$)	0.194 ($\Delta = +0.009$)	0.358 ($\Delta = +0.007$)
	720	0.232 ($\Delta = +0.009$)	0.387 ($\Delta = +0.007$)	0.241 ($\Delta = +0.017$)	0.400 ($\Delta = +0.007$)
ETTm1					
	96	0.026 ($\Delta = 0.000$)	0.122 ($\Delta = -0.001$)	0.026 ($\Delta = 0.000$)	0.121 ($\Delta = 0.000$)
	192	0.040 ($\Delta = 0.000$)	0.152 ($\Delta = +0.001$)	0.039 ($\Delta = 0.000$)	0.150 ($\Delta = 0.000$)
	336	0.053 ($\Delta = 0.000$)	0.174 ($\Delta = 0.000$)	0.052 ($\Delta = -0.001$)	0.173 ($\Delta = 0.000$)
	720	0.073 ($\Delta = 0.000$)	0.206 ($\Delta = 0.000$)	0.074 ($\Delta = 0.000$)	0.207 ($\Delta = 0.000$)
ETTm2					
	96	0.065 ($\Delta = 0.000$)	0.187 ($\Delta = 0.000$)	0.065 ($\Delta = 0.000$)	0.187 ($\Delta = +0.001$)
	192	0.093 ($\Delta = 0.000$)	0.232 ($\Delta = +0.001$)	0.094 ($\Delta = 0.000$)	0.230 ($\Delta = -0.001$)
	336	0.120 ($\Delta = -0.001$)	0.265 ($\Delta = -0.001$)	0.120 ($\Delta = 0.000$)	0.264 ($\Delta = -0.001$)
	720	0.171 ($\Delta = -0.001$)	0.322 ($\Delta = 0.000$)	0.171 ($\Delta = 0.000$)	0.321 ($\Delta = -0.001$)

Dataset	Horizon	PatchTST/64		PatchTST/42		NLinear	
		MSE	MAE	MSE	MAE	MSE	MAE
ETTh1							
	96	0.059	0.189	0.055	0.179	**0.053**	**0.177**
	192	0.074	0.215	0.071	0.205	**0.069**	**0.204**
	336	**0.076**	**0.220**	0.081	0.225	0.081	0.226
	720	0.087	0.236	0.087	0.232	**0.080**	**0.226**
ETTh2							
	96	0.131	0.284	0.129	0.282	**0.129**	**0.278**
	192	0.171	0.329	**0.168**	0.328	0.169	**0.324**
	336	**0.171**	**0.336**	0.185	0.351	0.194	0.355
	720	**0.223**	**0.380**	0.224	0.383	0.225	0.381
ETTm1							
	96	0.026	0.123	**0.026**	**0.121**	0.026	0.122
	192	0.040	0.151	0.039	0.150	**0.039**	**0.149**
	336	0.053	0.174	0.053	0.173	**0.052**	**0.172**
	720	**0.073**	**0.206**	0.074	0.207	**0.073**	0.207
ETTm2							
	96	0.065	0.187	0.065	0.186	**0.063**	**0.182**
	192	0.093	0.231	0.094	0.231	**0.090**	**0.223**
	336	0.121	0.266	0.120	0.265	**0.117**	**0.259**
	720	0.172	0.322	0.171	0.322	**0.170**	**0.318**

Table 7. Comparison of original PatchTST univariate forecasting results (MSE and MAE) from [2] with the NLinear baseline from [1] on ETT datasets for prediction horizons $T \in \{96, 192, 336, 720\}$. This table highlights the performance of PatchTST relative to a strong linear model not featured in the original paper's main comparisons. Best results for each metric and horizon are shown in **bold**.

ablated 20-epoch results against a 100-epoch full model, which raises some methodological concerns.

Running the 20-epoch ablations revealed major computational challenges, as shown in Table 8. On the large Traffic dataset, even with fewer epochs, removing patching ("CI" column), channel-independence ("P" column), or both ("Vanilla" column) caused out-of-memory (OOM) errors ("CUDA" entry) on an NVIDIA H100 GPU with 80GB VRAM, even at batch size 1. On the Electricity dataset, removing only channel-independence ("P" column) was feasible within 20 epochs, but removing both ("Vanilla" column) exceeded a 72-hour time limit ("TIMELIMIT" entry). These results highlight the clear efficiency gains from using patching together with channel-independence.

Table 8. Ablation study of patching and channel-independence on PatchTST/42, with both patching and channel independence (P+CI), only channel independence (CI), only patching (P) and neither of them (Vanilla TST). All values are shown as *Mean (Δ = Change from the original paper)*. CUDA in the table means the model runs out of GPU memory; TIMELIMIT indicates that the model took more than 72hrs to finish.

Dataset	Horizon	P+CI		CI		P		Vanilla	
		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
Weather									
	96	**0.204**	**0.154**	0.232 ($\Delta = -0.001$)	0.168 ($\Delta = -0.004$)	0.263 ($\Delta = +0.040$)	0.203 ($\Delta = +0.035$)	0.229 ($\Delta = -0.007$)	0.166 ($\Delta = -0.011$)
	192	**0.242**	**0.196**	0.250 ($\Delta = 0.000$)	0.205 ($\Delta = 0.000$)	0.288 ($\Delta = +0.026$)	0.246 ($\Delta = +0.033$)	0.270 ($\Delta = 0.000$)	0.215 ($\Delta = -0.006$)
	336	**0.282**	**0.247**	0.291 ($\Delta = +0.002$)	0.257 ($\Delta = +0.002$)	0.315 ($\Delta = +0.015$)	0.284 ($\Delta = +0.018$)	0.316 ($\Delta = +0.010$)	0.273 ($\Delta = +0.002$)
	720	**0.335**	**0.321**	0.342 ($\Delta = -0.001$)	0.330 ($\Delta = +0.003$)	0.358 ($\Delta = -0.001$)	0.349 ($\Delta = -0.002$)	0.362 ($\Delta = +0.009$)	0.346 ($\Delta = +0.006$)
Traffic									
	96	0.257	0.370	CUDA	CUDA	CUDA	CUDA	CUDA	CUDA
	192	0.264	0.389	CUDA	CUDA	CUDA	CUDA	CUDA	CUDA
	336	0.270	0.404	CUDA	CUDA	CUDA	CUDA	CUDA	CUDA
	720	0.289	0.433	CUDA	CUDA	CUDA	CUDA	CUDA	CUDA
Electricity									
	96	**0.225**	**0.132**	0.228 ($\Delta = -0.003$)	0.135 ($\Delta = -0.001$)	0.297 ($\Delta = -0.010$)	0.188 ($\Delta = -0.008$)	TIMELIMIT	TIMELIMIT
	192	**0.240**	**0.149**	0.245 ($\Delta = -0.018$)	0.152 ($\Delta = -0.012$)	0.309 ($\Delta = -0.014$)	0.204 ($\Delta = -0.011$)	TIMELIMIT	TIMELIMIT
	336	**0.264**	**0.171**	**0.264** ($\Delta = +0.003$)	**0.171** ($\Delta = +0.003$)	0.317 ($\Delta = -0.021$)	0.214 ($\Delta = -0.014$)	TIMELIMIT	TIMELIMIT
	720	0.304	0.215	**0.297** ($\Delta = +0.006$)	**0.207** ($\Delta = +0.005$)	0.356 ($\Delta = +0.011$)	0.266 ($\Delta = +0.022$)	TIMELIMIT	TIMELIMIT

Analyzing the results in Table 8 for the datasets where training finished shows a few clear points.

For the Weather dataset, the full P+CI model had the best performance (lowest MAE and MSE) across all time steps. Removing channel-independence (P) caused a large drop in accuracy. Removing patching (CI) had a smaller effect. This means channel-independence contributes more than patching, though both are useful.

For the Electricity dataset, the P+CI model also performed well. The CI-only model was close, and sometimes better (e.g., at T=720). This suggests patching matters less here, or CI alone gives most of the benefit. Again, dropping CI (P only) hurt performance significantly.

Effect of Varying Look-back Window — Figure 1 shows how changing the look-back window length (L) affects PatchTST/42's forecasting performance (MSE) on the Weather, Traffic, and Electricity datasets, for prediction horizons $T = 96$ and $T = 720$.

In line with the original paper (Figure 2 [2]), we see that performance generally improves as L increases, from 24 to 336 or 720. This effect is more noticeable at the longer horizon ($T = 720$). These results support the idea that PatchTST/42 can make better use of longer input histories, unlike some earlier Transformer-based models where performance degraded with longer look-back windows.

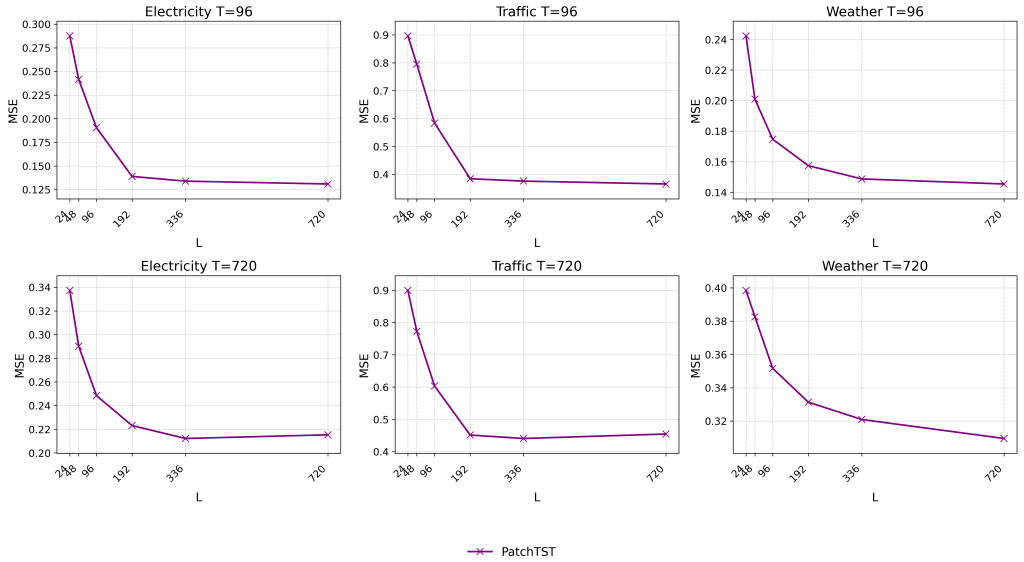

Figure 1. Forecasting performance (MSE) of the supervised PatchTST/42 model with varying look-back windows (L) on three datasets: Electricity, Traffic, and Weather. Results are presented for look-back windows $L \in \{24, 48, 96, 192, 336, 720\}$ across two prediction horizons, $T = 96$ (top row) and $T = 720$ (bottom row). Lower MSE values indicate better forecasting accuracy.

6 Discussion

We reproduced PatchTST [2] and found that its main ideas: patching and channel independence, help improve forecasting accuracy across several standard benchmarks (Weather, Traffic, Electricity, ETT). Our supervised results (Table 2, Table 6) are mostly consistent with those reported in the original paper, confirming the strength of the core architecture. That said, we also identified several methodological and practical issues worth discussing.

First, the ablation studies in the original paper raise concerns. The authors ran the ablation experiments under very limited settings (e.g., 20 epochs, batch size 1) for large datasets [2]. We faced similar resource constraints and couldn't train these ablated variants for the full number of epochs (Section 5.6). These shortened runs may overstate the benefits of PatchTST's components, especially when compared to baseline models that need longer training. In addition, the paper compares these limited ablation runs to full 100-epoch results from the main model, which complicates fair evaluation. We were also unable to run some configurations (e.g., Vanilla TST on Electricity/Traffic) due to memory and time limits, even with a high-end GPU. This suggests there might be undocumented implementation choices or optimizations in the original code. For future ablation work, we recommend using the same training settings across all models.

Second, some experiments were hard to reproduce. Our supervised results on ILI (Table 2) differ substantially from the original, likely because ILI needs specific hyperparameter tuning that wasn't fully documented. Other researchers have reported similar issues. We also got worse results in the self-supervised (Table 3) and transfer learning setups (Table 4, Table 5). This suggests that the pretraining strategies might be more sensitive to implementation details than expected, or they may need more than the reported 100 epochs to outperform plain supervised training.

Third, while PatchTST often beats older Transformer models as reported, its gains over strong linear baselines like DLinear and NLinear are often small, especially on univariate forecasting tasks (Table 7). This raises a question: is the added complexity of

PatchTST worth it, when simpler and faster models can give similar results on many datasets?

7 Conclusion

This study presented a reproduction effort for the PatchTST model [2]. While we successfully replicated the core supervised forecasting performance claimed for most benchmark datasets, confirming the general effectiveness of the patching and channel-independence approach, significant challenges were encountered.

Specifically, results for the ILI dataset and the self-supervised/transfer learning experiments proved difficult to reproduce based on the available information, highlighting potential gaps in the original paper's reported methodology or sensitivity to unspecified implementation details. Furthermore, our investigation confirmed methodological concerns regarding the original ablation study comparisons. The performance advantages of PatchTST over strong linear baselines were also found to be marginal in several univariate settings.

Overall, our findings confirm the value of PatchTST's architectural concepts but also highlight the critical importance of complete methodological transparency, rigorous ablation protocols, and thorough benchmarking against simple baselines for ensuring the reproducibility and practical assessment of complex time series models.

References

1. A. Zeng, M. Chen, L. Zhang, and Q. Xu. "Are transformers effective for time series forecasting?" In: **Proceedings of the AAAI conference on artificial intelligence**. Vol. 37. 9. 2023, pp. 11121–11128.
2. Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. "A time series is worth 64 words: Long-term forecasting with transformers." In: **arXiv preprint arXiv:2211.14730** (2022).
3. D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Instance normalization: The missing ingredient for fast stylization." In: **arXiv preprint arXiv:1607.08022** (2016).
4. T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo. "Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift." In: **International Conference on Learning Representations**. 2022.
5. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: **Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)**. 2019, pp. 4171–4186.
6. K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. "Masked autoencoders are scalable vision learners." In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. 2022, pp. 16000–16009.