PatchTST: Turning 64 Words Into a Time Series Prediction

Dong-Keon Kim Follow 9 min read · Jun 25, 2025

2

How segmenting time series into semantic patches reshapes multivariate forecasting with Transformers

Introduction

About

The author in this literature introduces PatchTST, **a Transformer-based architecture tailored for long-term forecasting of multivariate time series.** It proposes two key innovations: **treating time series as sequences of patches** rather than individual time steps, and **applying a channel-independent design** where each univariate series is modeled separately but shares the same Transformer backbone. These choices significantly **reduce computational complexity** and **enhance the model's ability to capture local temporal semantics.** PatchTST not only improves forecasting accuracy over state-of-the-art baselines but also excels in self-supervised representation learning and transferability. Extensive experiments across various datasets validate its effectiveness in both supervised and pretraining settings.

Background

The use of Transformers in time series forecasting has grown rapidly, inspired by their success in NLP and vision. However, their applicability remains debated. Despite innovations like Informer and FEDformer that reduce attention complexity or incorporate signal decomposition, **recent evidence shows that even simple linear models can outperform these architectures**, challenging their practical value in this domain.

One underutilized idea in time series Transformers is patching — standard in vision (e.g., ViT) and speech — but often neglected in favor of point-wise inputs that lack semantic context. Although models like Autoformer and Triformer make partial use of patch-level representations, **they do not fully treat patches as fundamental modeling units capable of capturing local semantics.**

Another overlooked design choice is **channel-independence.** While typical Transformer models mix features across channels, prior work with CNNs and linear models has shown that **modeling univariate series separately can be more effective.** Furthermore, **self-supervised learning for time series has not yet fully leveraged the potential of masked modeling in Transformers,** often relying on contrastive approaches that are less aligned with forecasting goals.

Method

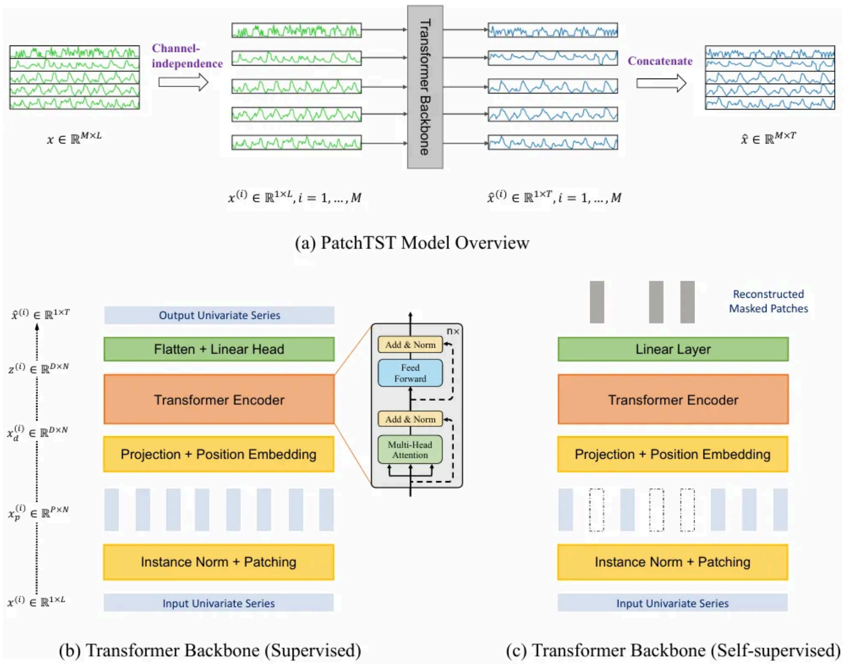


Fig 1. Proposed PatchTST architecture. (Source)

The proposed PatchTST framework is designed for long-term multivariate time series forecasting and self-supervised representation learning. The method is built upon two fundamental architectural choices: (1) **the segmentation of univariate time series into patches**, and (2) **the use of a shared Transformer encoder applied independently across each channel.** The overall architecture, including both supervised and self-supervised paths, is illustrated in **Figure 1**.

Model Structure

PatchTST is designed for long-term forecasting of multivariate time series by rethinking how temporal data is tokenized and processed. Given a multivariate input sequence of length L, represented as $(\mathbf{x}_1,\ldots,\mathbf{x}_l)$, where each $\mathbf{x}_t \in \mathbb{R}^M$, the model first splits the input into M univariate series. Each univariate time series $\mathbf{x}_{1:l}^{(i)}=(x_1^{(i)},\ldots,x_l^{(i)})$ is normalized using instance normalization to mitigate distribution shift across samples.

Next, each channel is segmented into patches of fixed length P and stride S, resulting in N patches:

$$N = \left\lfloor \frac{L-P}{S} \right\rfloor + 2$$

Each patch sequence $\mathbf{x}_p^{(i)} \in \mathbb{R}^{P\times N}$ is linearly projected into a latent dimension D via a learnable matrix $W_p \in \mathbb{R}^{D\times P}$, and positional encodings $W_{\text{pos}} \in \mathbb{R}^{D\times N}$ are added to form the input tokens:

$$\mathbf{x}_d^{(i)} = W_p \mathbf{x}_p^{(i)} + W_{\text{pos}}$$

The tokens are passed through a shared Transformer encoder that uses multi-head self-attention and feed-forward layers, as shown in **Figure 1b**. For each attention head hhh, the attention output is computed as:

$$\text{Attention}(Q_h^{(i)}, K_h^{(i)}, V_h^{(i)}) = \text{Softmax}\left(\frac{Q_h^{(i)}(K_h^{(i)})^T}{\sqrt{d_k}}\right) V_h^{(i)}$$

where the *queries, keys,* and *values* are learned via linear projections from the input.

After encoding, the output representations $\mathbf{z}^{(i)} \in \mathbb{R}^{D\text{x}N}$ are flattened and passed through a linear head to produce the predicted values:

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

The final objective is to minimize the mean squared error (MSE) across all channels:

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

This architecture is significantly more efficient than standard Transformers, as **the patching mechanism reduces the attention map complexity from** $O(L^2)$ to $O(N^2)$, **allowing for longer input sequences. Figure 1a** provides a visual summary of the full architecture.

Representation Learning

In addition to supervised forecasting, PatchTST supports self-supervised pretraining via **masked patch reconstruction**, inspired by masked autoencoders in NLP and vision. The model uses **non-overlapping patches** to e**nsure that masked regions can be cleanly separated from observed ones**. A subset of patches is selected uniformly at random and zero-masked.

The same encoder is used, but the forecasting head is replaced with a linear decoder that reconstructs the original values of the masked patches:

$$\hat{\mathbf{x}}_p^{(i)} = W_{\text{dec}} \mathbf{z}^{(i)}$$

Note that the weight $W \in \mathbb{R}^{P\text{x}D}$ is a learned decoder matrix.

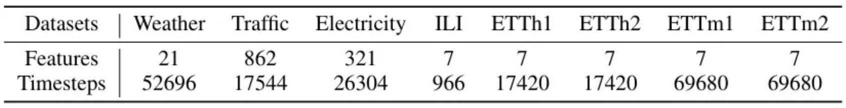
Get Dong-Keon Kim's stories in your inbox

This pretraining approach **encourages the encoder to learn abstract and transferable representations of temporal patterns**, which are later fine-tuned for downstream forecasting tasks. Notably, **the shared encoder design enables pretraining on datasets with different channel counts**, as the weight-sharing mechanism maintains compatibility across domains (see **Figure 1c**).

Experiment

To validate the effectiveness of PatchTST, the authors conducted **extensive experiments across a diverse set of multivariate time series datasets**, comparing the model against state-of-the-art baselines in both supervised and self-supervised settings. The experimental analysis was designed to **highlight improvements in forecasting accuracy**, **efficiency**, and **generalization.** It also explores the value of the proposed patching and channel-independence mechanisms via ablation studies.

Dataset



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

Table 1. Statistics of popular datasets for benchmark. (Source)

The evaluation is carried out on eight well-established public datasets frequently used for benchmarking time series forecasting:

- **Weather: 21** channels, **52,696** timesteps — large-scale climate data.
- **Traffic: 862** channels, **17,544** timesteps — road occupancy rate.
- **Electricity: 321** channels, **26,304** timesteps — power consumption.
- **ILI (Influenza-like Illness): 7** channels, **966** timesteps — medical time series.
- **ETT (Electricity Transformer Temperature):** Four variants:
- **ETTh1 & ETTh2:** Hourly frequency, **7** channels, **17,420** timesteps.
- **ETTm1 & ETTm2:** Minutely frequency, **7** channels, **69,680** timesteps.

These datasets represent a mix of large and small, dense and sparse multivariate time series, enabling robust comparison across varied temporal dynamics (**Table 1**).

Model Set-up

Two model configurations are proposed for PatchTST:

- **PatchTST/42:**
  Look-back window L = **336**, number of patches = **42**

- **PatchTST/64:**
  Look-back window L = 512, number of patches = **64**

Both models use:

- Patch length P = **16**, stride S = **8**
- Input normalization via **instance norm**
- **Loss function:** Mean Squared Error (MSE)
- Transformer encoder with **shared weights across channels**

For self-supervised experiments:

- Patches are **non-overlapping**
- 40% masking ratio
- Pretrained for **100** epochs, followed by:
  Linear probing (20 epochs) or Fine-tuning (**10** linear + **20** full)

Comparison

PatchTST is compared against a range of strong baselines, including:

- **Transformer-based models:**
  FEDformer, Autoformer, Informer, Pyraformer, LogTrans
- **Non-Transformer baseline:**
  DLinear (a linear model shown to outperform Transformers in prior work)
- **Self-supervised methods:**
  TS2Vec, TNC, BTSF, TS-TCC

For fairness, all Transformer baselines are tested with various look-back windows, and the best result is retained. Baseline results are primarily drawn from this work, with some reimplementation as needed.

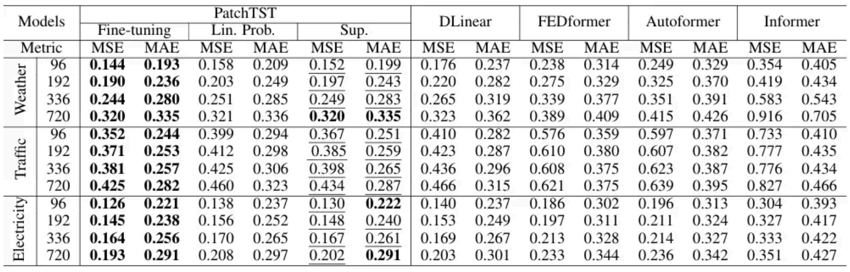
Evaluation Metrics

Two standard metrics are used for evaluation:

- **Mean Squared Error (MSE):**
  Measures the average of squared differences between predicted and true values. Sensitive to large errors, it emphasizes the importance of precision.
- **Mean Absolute Error (MAE):**
  Calculates the average magnitude of the prediction errors without considering their direction. It is more robust to outliers.

Results

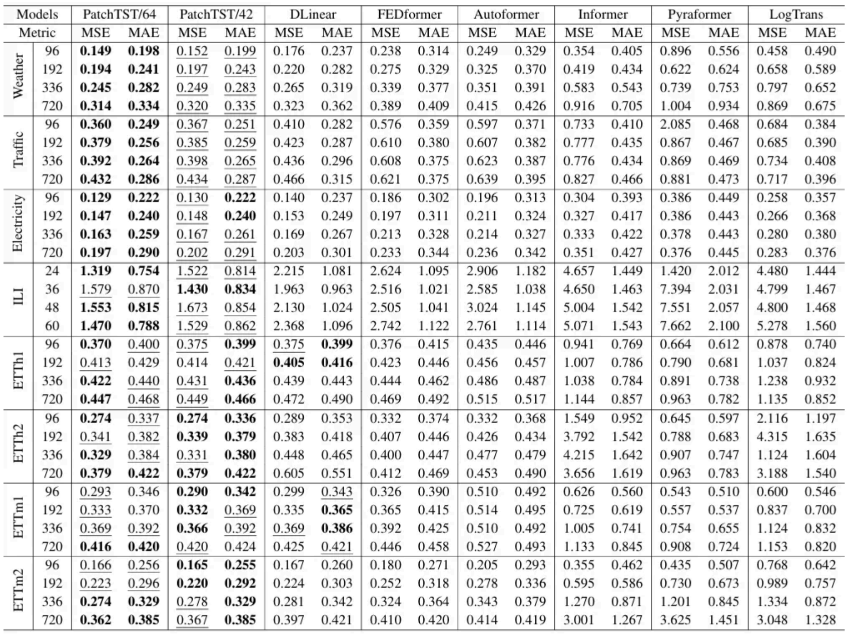
Long-Term Time Series Forecasting



Models		PatchTST						DLinear		FEDformer		Autoformer		Informer	
		Fine-tuning		Lin. Prob.		Sup.									
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.144**	**0.193**	0.158	0.209	0.152	0.199	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405
	192	**0.190**	**0.236**	0.203	0.249	0.197	0.243	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434
	336	**0.244**	**0.280**	0.251	0.285	0.249	0.283	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543
	720	**0.320**	**0.335**	0.321	0.336	**0.320**	**0.335**	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705
Traffic	96	**0.352**	**0.244**	0.399	0.294	0.367	0.251	0.410	0.282	0.576	0.359	0.597	0.371	0.733	0.410
	192	**0.371**	**0.253**	0.412	0.298	0.385	0.259	0.423	0.287	0.610	0.380	0.607	0.382	0.777	0.435
	336	**0.381**	**0.257**	0.425	0.306	0.398	0.265	0.436	0.296	0.608	0.375	0.623	0.387	0.776	0.434
	720	**0.425**	**0.282**	0.460	0.323	0.434	0.287	0.466	0.315	0.621	0.375	0.639	0.395	0.827	0.466
Electricity	96	**0.126**	**0.221**	0.138	0.237	0.130	**0.222**	0.140	0.237	0.186	0.302	0.196	0.313	0.304	0.393
	192	**0.145**	**0.238**	0.156	0.252	0.148	0.240	0.153	0.249	0.197	0.311	0.211	0.324	0.327	0.417
	336	**0.164**	**0.256**	0.170	0.265	0.167	0.261	0.169	0.267	0.213	0.328	0.214	0.327	0.333	0.422
	720	**0.193**	**0.291**	0.208	0.297	0.202	**0.291**	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427

Table 2. Multivariate long-term forecasting results with self-supervised PatchTST. Note that the best results are in bold and the second best are underlined. (Source)

The supervised forecasting results are shown in **Table 2.** Across all datasets and prediction horizons, both PatchTST/42 and PatchTST/64 outperform existing Transformer-based models and the strong linear baseline DLinear. Notably, on large datasets such as Weather, Traffic, and Electricity, PatchTST/64 achieves an average of 21.0% lower MSE and 16.7% lower MAE compared to the best Transformer baseline. For example, on the Weather dataset with a prediction horizon of 336, PatchTST/64 scores an MSE of 0.245 compared to FEDformer's 0.339. The performance gains are consistent across short (96) and long (720) horizons.

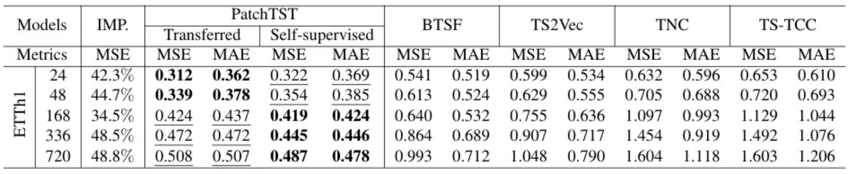


Models		PatchTST/64		PatchTST/42		DLinear		FEDformer		Autoformer		Informer		Pyraformer		LogTrans	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.149**	**0.198**	0.152	0.199	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405	0.896	0.556	0.458	0.490
	192	**0.194**	**0.241**	0.197	0.243	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434	0.622	0.624	0.658	0.589
	336	**0.245**	**0.282**	0.249	0.283	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543	0.739	0.753	0.797	0.652
	720	**0.314**	**0.334**	0.320	0.335	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705	1.004	0.934	0.869	0.675
Traffic	96	**0.360**	**0.249**	0.367	0.251	0.410	0.282	0.576	0.359	0.597	0.371	0.733	0.410	2.085	0.468	0.684	0.384
	192	**0.379**	**0.256**	0.385	0.259	0.423	0.287	0.610	0.380	0.607	0.382	0.777	0.435	0.867	0.467	0.685	0.390
	336	**0.392**	**0.264**	0.398	0.265	0.436	0.296	0.608	0.375	0.623	0.387	0.776	0.434	0.869	0.469	0.734	0.408
	720	**0.432**	**0.286**	0.434	0.287	0.466	0.315	0.621	0.375	0.639	0.395	0.827	0.466	0.881	0.473	0.717	0.396
Electricity	96	**0.129**	**0.222**	0.130	**0.222**	0.140	0.237	0.186	0.302	0.196	0.313	0.304	0.393	0.386	0.449	0.258	0.357
	192	**0.147**	**0.240**	0.148	**0.240**	0.153	0.249	0.197	0.311	0.211	0.324	0.327	0.417	0.386	0.443	0.266	0.368
	336	**0.163**	**0.259**	0.167	0.261	0.169	0.267	0.213	0.328	0.214	0.327	0.333	0.422	0.378	0.443	0.280	0.380
	720	**0.197**	**0.290**	0.202	0.291	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427	0.376	0.445	0.283	0.376
ILI	24	**1.319**	**0.754**	1.522	0.814	2.215	1.081	2.624	1.095	2.906	1.182	4.657	1.449	1.420	2.012	4.480	1.444
	36	1.579	0.870	**1.430**	**0.834**	1.963	0.963	2.516	1.021	2.585	1.038	4.650	1.463	7.394	2.031	4.799	1.467
	48	**1.553**	**0.815**	1.673	0.854	2.130	1.024	2.505	1.041	3.024	1.145	5.004	1.542	7.551	2.057	4.800	1.468
	60	**1.470**	**0.788**	1.529	0.862	2.368	1.096	2.742	1.122	2.761	1.114	5.071	1.543	7.662	2.100	5.278	1.560
ETTh1	96	**0.370**	0.400	0.375	**0.399**	0.375	**0.399**	0.376	0.415	0.435	0.446	0.941	0.769	0.664	0.612	0.878	0.740
	192	0.413	0.429	0.414	0.421	**0.405**	**0.416**	0.423	0.446	0.456	0.457	1.007	0.786	0.790	0.681	1.037	0.824
	336	**0.422**	0.440	0.431	**0.436**	0.439	0.443	0.444	0.462	0.486	0.487	1.038	0.784	0.891	0.738	1.238	0.932
	720	**0.447**	0.468	0.449	**0.466**	0.472	0.490	0.469	0.492	0.515	0.517	1.144	0.857	0.963	0.782	1.135	0.852
ETTh2	96	**0.274**	0.337	**0.274**	**0.336**	0.289	0.353	0.332	0.374	0.332	0.368	1.549	0.952	0.645	0.597	2.116	1.197
	192	0.341	0.382	**0.339**	**0.379**	0.383	0.418	0.407	0.446	0.426	0.434	3.792	1.542	0.788	0.683	4.315	1.635
	336	**0.329**	0.384	0.331	**0.380**	0.448	0.465	0.400	0.447	0.477	0.479	4.215	1.642	0.907	0.747	1.124	1.604
	720	**0.379**	**0.422**	**0.379**	**0.422**	0.605	0.551	0.412	0.469	0.453	0.490	3.656	1.619	0.963	0.783	3.188	1.540
ETTm1	96	0.293	0.346	**0.290**	**0.342**	0.299	0.343	0.326	0.390	0.510	0.492	0.626	0.560	0.543	0.510	0.600	0.546
	192	0.333	0.370	**0.332**	0.369	0.335	**0.365**	0.365	0.415	0.514	0.495	0.725	0.619	0.557	0.537	0.837	0.700
	336	0.369	0.392	**0.366**	0.392	0.369	**0.386**	0.392	0.425	0.510	0.492	1.005	0.741	0.754	0.655	1.124	0.832
	720	**0.416**	**0.420**	0.420	0.424	0.425	0.421	0.446	0.458	0.527	0.493	1.133	0.845	0.908	0.724	1.153	0.820
ETTm2	96	0.166	0.256	**0.165**	**0.255**	0.167	0.260	0.180	0.271	0.205	0.293	0.355	0.462	0.435	0.507	0.768	0.642
	192	0.223	0.296	**0.220**	**0.292**	0.224	0.303	0.252	0.318	0.278	0.336	0.595	0.586	0.730	0.673	0.989	0.757
	336	**0.274**	**0.329**	0.278	**0.329**	0.281	0.342	0.324	0.364	0.343	0.379	1.270	0.871	1.201	0.845	1.334	0.872
	720	**0.362**	**0.385**	0.367	**0.385**	0.397	0.421	0.410	0.420	0.414	0.419	3.001	1.267	3.625	1.451	3.048	1.328

Table 3. Multi variate long-term forecasting results with supervised PatchTST. (Source)

These results confirm that **the proposed patching and channel-independent architecture improves both modeling efficiency and accuracy, especially as the receptive field increases.** The supervised forecasting results are shown in **Table 3.** Across all datasets and prediction horizons, both PatchTST/42 and PatchTST/64 outperform existing Transformer-based models and the strong linear baseline DLinear. Notably, on large datasets such as Weather, Traffic, and Electricity, PatchTST/64 achieves an average of 21.0% lower MSE and 16.7% lower MAE compared to the best Transformer baseline. For example, on the Weather dataset with a prediction horizon of 336, PatchTST/64 scores an MSE of 0.245 compared to FEDformer's 0.339. The performance gains are consistent across short (96) and long (720) horizons.

Representation Learning



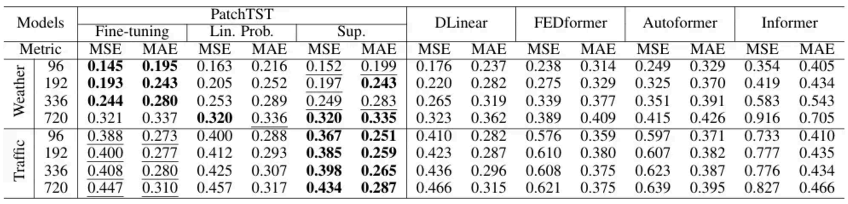
Models		IMP.	PatchTST				BTSF		TS2Vec		TNC		TS-TCC	
			Transferred		Self-supervised									
Metrics		MSE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	42.3%	**0.312**	**0.362**	0.322	0.369	0.541	0.519	0.599	0.534	0.632	0.596	0.653	0.610
	48	44.7%	**0.339**	**0.378**	0.354	0.385	0.613	0.524	0.629	0.555	0.705	0.688	0.720	0.693
	168	34.5%	0.424	0.437	**0.419**	**0.424**	0.640	0.532	0.755	0.636	1.097	0.993	1.129	1.044
	336	48.5%	0.472	0.472	**0.445**	**0.446**	0.864	0.689	0.907	0.717	1.454	0.919	1.492	1.076
	720	48.8%	0.508	0.507	**0.487**	**0.478**	0.993	0.712	1.048	0.790	1.604	1.118	1.603	1.206

Table 4. Representation learning methods comparison outcomes. (Source)

To evaluate PatchTST in a self-supervised setting, the model is pretrained using masked patch reconstruction and then fine-tuned or probed for forecasting. **Table 4** shows that self-supervised PatchTST (especially with fine-tuning) outperforms all baselines, including its own supervised variant. On the Electricity dataset with a 336-step horizon, fine-tuned PatchTST achieves an MSE of 0.164, which is better than the supervised result (0.167) and significantly better than FEDformer (0.213).

These findings suggest that **pretraining with masked patch modeling allows PatchTST to learn generalizable representations, improving downstream forecasting** even when labeled data is limited.

Transfer Learning



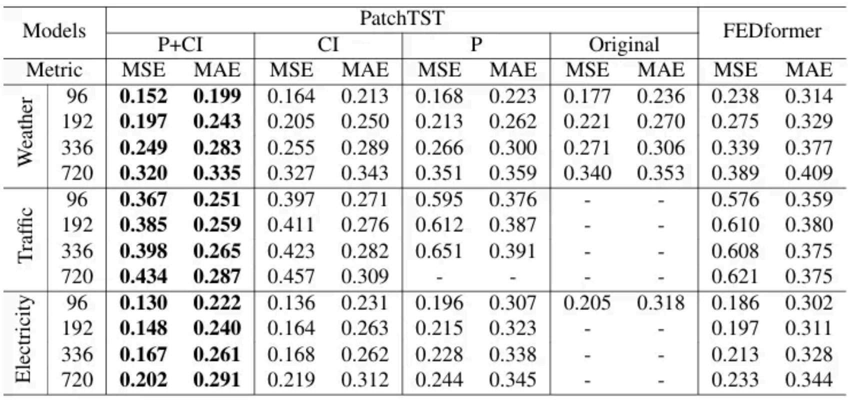
Models		PatchTST						DLinear		FEDformer		Autoformer		Informer	
		Fine-tuning		Lin. Prob.		Sup.									
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.145**	**0.195**	0.163	0.216	0.152	0.199	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405
	192	**0.193**	**0.243**	0.205	0.252	0.197	**0.243**	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434
	336	**0.244**	**0.280**	0.253	0.289	0.249	0.283	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543
	720	0.321	0.337	**0.320**	0.336	**0.320**	**0.335**	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705
Traffic	96	0.388	0.273	0.400	0.288	**0.367**	**0.251**	0.410	0.282	0.576	0.359	0.597	0.371	0.733	0.410
	192	0.400	0.277	0.412	0.293	**0.385**	**0.259**	0.423	0.287	0.610	0.380	0.607	0.382	0.777	0.435
	336	0.408	0.280	0.425	0.307	**0.398**	**0.265**	0.436	0.296	0.608	0.375	0.623	0.387	0.776	0.434
	720	0.447	0.310	0.457	0.317	**0.434**	**0.287**	0.466	0.315	0.621	0.375	0.639	0.395	0.827	0.466

Table 5. Transfer learning task results with compared Transformer families. (Source)

The generalizability of PatchTST is further validated thrugh transfer learning experiments (**Table 5**). The model is pretrained on the Electricity dataset and fine-tuned on others. While some degradation is observed compared to in-domain pretraining, PatchTST still outperforms other methods, including supervised Transformer models. For example, when transferred to the Traffic dataset (336-step horizon), PatchTST achieves an MSE of 0.408 compared to Informer's 0.776.

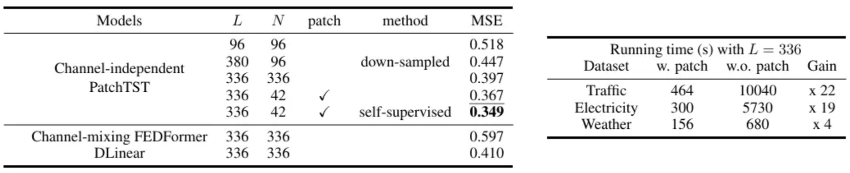
This suggests that representations learned via PatchTST **can be shared across domains with minimal adaptation, saving computational resources and data annotation.**

Ablation Study



Models		PatchTST								FEDformer	
		P+CI		CI		P		Original			
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.152**	**0.199**	0.164	0.213	0.168	0.223	0.177	0.236	0.238	0.314
	192	**0.197**	**0.243**	0.205	0.250	0.213	0.262	0.221	0.270	0.275	0.329
	336	**0.249**	**0.283**	0.255	0.289	0.266	0.300	0.271	0.306	0.339	0.377
	720	**0.320**	**0.335**	0.327	0.343	0.351	0.359	0.340	0.353	0.389	0.409
Traffic	96	**0.367**	**0.251**	0.397	0.271	0.595	0.376	-	-	0.576	0.359
	192	**0.385**	**0.259**	0.411	0.276	0.612	0.387	-	-	0.610	0.380
	336	**0.398**	**0.265**	0.423	0.282	0.651	0.391	-	-	0.608	0.375
	720	**0.434**	**0.287**	0.457	0.309	-	-	-	-	0.621	0.375
Electricity	96	**0.130**	**0.222**	0.136	0.231	0.196	0.307	0.205	0.318	0.186	0.302
	192	**0.148**	**0.240**	0.164	0.263	0.215	0.323	-	-	0.197	0.311
	336	**0.167**	**0.261**	0.168	0.262	0.228	0.338	-	-	0.213	0.328
	720	**0.202**	**0.291**	0.219	0.312	0.244	0.345	-	-	0.233	0.344

Table 6. Ablation study of patching and channel-independence in PatchTST. Here, (CI) indicates "with channel-independence" and (P) denotes "with patching". (Source)



Models	L	N	patch	method	MSE
Channel-independent PatchTST	96	96			0.518
	380	96		down-sampled	0.447
	336	336			0.397
	336	42	✓		0.367
	336	42	✓	self-supervised	**0.349**
Channel-mixing FEDFormer	336	336			0.597
DLinear	336	336			0.410

Running time (s) with $L = 336$			
Dataset	w. patch	w.o. patch	Gain
Traffic	464	10040	x 22
Electricity	300	5730	x 19
Weather	156	680	x 4

Table 7. A case study of multivariate time series forecasting on Traffic dataset. The prediction horizon is 96. (Source)

An ablation study is conducted to isolate the contributions of patching and channel-independence, summarized in **Table 6.** Removing patching or channel-independence from PatchTST significantly degrades performance, with MSE increasing by up to 58% on the Traffic dataset. For instance, the removal of both (i.e., reverting to a standard Transformer) results in MSE of 0.595 at a 96-step horizon, compared to 0.367 for the full PatchTST.

The efficiency gain is further visualized in **Table 7**, showing that **patching reduces training time by up to 22x on large datasets, without sacrificing accuracy.**

Varying Look-back Window

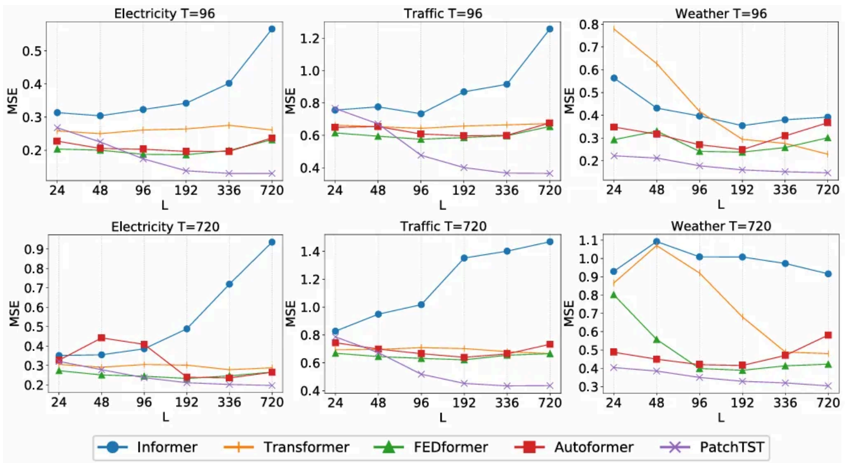


Fig 2. Forecasting performance (MSE) with varying look-back windows on 3 large datasets: Electricity, Traffic

In **Figure 2**, the authors show that unlike other Transformer baselines, PatchTST benefits consistently from increasing the look-back window. For instance, extending LLL from 96 to 336 significantly reduces MSE, whereas FEDformer shows marginal or no improvement. This demonstrates that PatchTST effectively leverages longer temporal contexts due to its reduced attention complexity and local semantic awareness from patching.

These results confirm that **the proposed patching and channel-independent architecture improves both modeling efficiency and accuracy,** especially as the receptive field increases.

Conclusion

This research introduces **a simple yet highly effective approach for Transformer-based long-term time series forecasting** named **PatchTST**. By

shifting from point-wise to patch-based tokens and **adopting a channel-independent architecture**, the model **improves accuracy**, **efficiency**, and **transferability.** Its superiority is evident across supervised, self-supervised, and transfer learning tasks. The work **revitalizes the use of Transformers in time series applications** by **addressing their core limitations without complex modifications.**

Review

This work presents **a well-motivated and elegantly executed reformulation of Transformer models** for time series forecasting. The patching mechanism is conceptually aligned with practices in NLP and vision, bringing semantic locality into the time domain. Meanwhile, c**hannel-independence** introduces **parameter efficiency and robustness**, especially **for large multivariate datasets.** The paper excels in clarity, **provides exhaustive experimental validation**, and d**emonstrates versatility through supervised**, **self-supervised**, and **transfer settings.**

A potential limitation lies in **the assumption of independence across channels**, which **may not hold for datasets with strong inter-channel dependencies.** In addition, while patching reduces sequence length and computational cost, **the choice of patch size and stride might require tuning per dataset.** Further directions could explore adaptive patching, hybrid attention that selectively mixes channels, or extending the model to multivariate outputs for structured forecasting tasks. In conclusion, **PatchTST sets a new state-of-the-art and establishes a compelling design paradigm for future Transformer-based time series models.**

Reference

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

I hope you continue to show much interest! Your follow is a great help!

Paper Review Time Series Forecasting Time Series Analysis Deep Learning Transformers

Written by Dong-Keon Kim

194 followers · 2 following

Follow

I summarize, organize, and review AI papers. I also investigate new or emerging AI technologies.

No responses yet

Write a response

What are your thoughts?