TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting

Vijay Ekambaram
vijaye12@in.ibm.com
IBM Research
Bangalore, India

Arindam Jati*
arindam.jati@ibm.com
IBM Research
Bangalore, India

Nam Nguyen*
nnguyen@us.ibm.com
IBM Research
Yorktown Heights, NY, USA

Phanwadee Sinthong
gift.sinthong@ibm.com
IBM Research
Yorktown Heights, NY, USA

Jayant Kalagnanam
jayant@us.ibm.com
IBM Research
Yorktown Heights, NY, USA

ABSTRACT

Transformers have gained popularity in time series forecasting for their ability to capture long-sequence interactions. However, their memory and compute-intensive requirements pose a critical bottleneck for long-term forecasting, despite numerous advancements in compute-aware self-attention modules. To address this, we propose **TSMixer**, a lightweight neural architecture exclusively composed of multi-layer perceptron (MLP) modules. TSMixer is designed for multivariate forecasting and representation learning on patched time series, providing an efficient alternative to Transformers. Our model draws inspiration from the success of MLP-Mixer models in computer vision. We demonstrate the challenges involved in adapting Vision MLP-Mixer for time series and introduce empirically validated components to enhance accuracy. This includes a novel design paradigm of attaching online reconciliation heads to the MLP-Mixer backbone, for explicitly modeling the time-series properties such as hierarchy and channel-correlations. We also propose a Hybrid channel modeling approach to effectively handle noisy channel interactions and generalization across diverse datasets, a common challenge in existing patch channel-mixing methods. Additionally, a simple gated attention mechanism is introduced in the backbone to prioritize important features. By incorporating these lightweight components, we significantly enhance the learning capability of simple MLP structures, outperforming complex Transformer models with minimal computing usage. Moreover, TSMixer's modular design enables compatibility with both supervised and masked self-supervised learning methods, making it a promising building block for time-series Foundation Models. TSMixer outperforms state-of-the-art MLP and Transformer models in forecasting by a considerable margin of 8-60%. It also outperforms the latest strong benchmarks of Patch-Transformer models (by 1-2%) with a significant reduction in memory and runtime (2-3X).

Accepted in the Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), Research Track. Delayed release in arXiv to comply with the conference policies on the double-blind review process. This work has been submitted to the KDD peer-review process on Feb, 02, 2023 [1].

*Both authors contributed equally to this research.

KDD '23, August 6–10, 2023, Long Beach, CA, USA
2023.

1 INTRODUCTION

Multivariate time series forecasting is the task of predicting the future values of multiple (possibly) related time series at future time points given the historical values of those time series. It has widespread applications in weather forecasting, traffic prediction, industrial process controls, *etc.* This decade-long problem has been well studied in the past by several statistical and ML methods [2], [15]. Recently, Transformer [29]-based models are becoming popular for long-term multivariate forecasting due to their powerful capability to capture long-sequence dependencies. Several Transformer architectures were suitably designed for this task in last few years including Informer [37], Autoformer [30], FEDformer [30], and Pyraformer [20]. However, the success of the Transformer in the semantically rich NLP domain has not been well-transferred to the time series domain. One of the possible reasons is that, though positional embedding in Transformers preserves some ordering information, the nature of the permutation-invariant self-attention mechanism inevitably results in temporal information loss. This hypothesis has been empirically validated in [34], where an embarrassingly simple linear (DLinear) model is able to outperform most of the above-mentioned Transformer-based forecasting models.

Furthermore, unlike words in a sentence, individual time points in a time series lack significant semantic information and can be easily inferred from neighboring points. Consequently, a considerable amount of modeling capacity is wasted on learning point-wise details. PatchTST [22] has addressed this issue by dividing the input time series into patches and applying a transformer model, resulting in superior performance compared to existing models. However, PatchTST employs a pure channel[1] independence approach which does not explicitly capture the cross-channel correlations. PatchTST has demonstrated that channel independence can enhance performance compared to channel mixing, where channels are simply concatenated before being fed into the model. This simple mixing approach can lead to noisy interactions between channels in the initial layer of the Transformer, making it challenging to disentangle them at the output. CrossFormer [3], another recent effort on Patch Transformers with an improved channel mixing technique, also faces this issue (see Appendix Table 9). Therefore, there is a necessity to explicitly model channel interactions to seize opportunistic accuracy improvements while effectively reducing the high

[1]Throughout the text, we will use "channel" to denote the individual time series in a multivariate data (i.e. a multivariate time series is a multi-channel signal).

volume of noisy interactions across channels. In addition, though PatchTST reduces the timing and memory overhead via patching, it still uses the multi-head self-attention under the hood, which is computationally expensive even when it is applied at the patch level.

Recently, a series of multi-layer perceptron (MLP) models under the umbrella of "MLP-Mixers" have been proposed in the computer vision domain [19, 25, 27]. These models are lightweight and fast, and achieve comparable or superior performance to vision Transformer models while completely eliminating the need for computing intense multi-head self-attention [25]. Moreover, MLP-Mixer by its default architecture does not disturb the temporal ordering of the inputs which makes it a natural fit for time-series problems and resolves the concerns raised in DLinear [34]. At this point, we ask the following important question - *Can MLP-Mixer models yield superior performance for multivariate time series forecasting? If so, what are the required time series customizations?*

Towards this, we show that the adoption of MLP-Mixer for time series is *not* trivial, i.e., just applying the vanilla MLP-Mixer with some input and output shape modification will not make it a powerful model, and will have suboptimal performance compared to the state-of-the-arts. Hence, we propose **TSMixer**, a novel MLP-Mixer architecture for accurate multivariate time series forecasting. Like PatchTST, TSMixer is also patching-based and follows a modular architecture of learning a common *"backbone"* to capture the temporal dynamics of the data as a patch representation, and different *"heads"* are attached and finetuned based on various downstream tasks (Ex. forecasting). Backbone is considered task-independent and can learn across multiple datasets with a masked reconstruction loss while the heads are task and data-specific.

Key highlights of TSMixer are as follows:

(1) TSMixer is a patching-based, lightweight neural architecture designed solely with MLP modules that exploits various inherent time-series characteristics for accurate *multivariate forecasting and representation learning*.
(2) TSMixer proposes a novel design paradigm of attaching & tuning online reconciliation[2] heads to the MLP-Mixer backbone that significantly empowers the learning capability of simple MLP structures to outperform complex Transformer models while using less computing resources. This study is the first of its kind to highlight the benefits of infusing online reconciliation approaches in the prediction head of Mixer style backbone architectures for time-series modeling.
(3) Specifically, TSMixer proposes two novel online reconciliation heads to tune & improve the forecasts by leveraging the following intrinsic time series properties: *hierarchical patch-aggregation* and *cross-channel correlation*.
(4) For effective *cross-channel modeling*, TSMixer follows a novel "Hybrid" channel modeling by *augmenting a channel independent backbone with a cross-channel reconciliation head*. This hybrid architecture allows the backbone to generalize across diverse datasets with different channels, while the reconciliation head effectively learns the channel interactions specific to the task and data. This approach effectively handles noisy interactions across channels, which is a challenge in existing patch channel-mixing methods.
(5) For effective *long sequence modeling*, TSMixer introduces a simple Gated Attention[3] that guides the model to focus on the important features. This, when augmented with Hierarchical Patch Reconciliation Head and standard MLP-Mixer operations enables effective modeling of long-sequence interaction and eliminates the need for complex multi-head self-attention blocks.
(6) Finally, the modular design of TSMixer enables it to work with both supervised and masked self-supervised learning methodologies which makes it a potential building block for time-series foundation models [7].

We conduct a detailed empirical analysis on 7 popular public datasets, wherein. TSMixer outperforms all existing benchmarks with *extremely reduced training time and memory usage*. Snapshot (relative MSE improvements) of the primary benchmarks are as follows:

- TSMixer outperforms DLinear by a considerable margin of 8% (Table 2).
- TSMixer marginally outperforms PatchTST by 1-2% but with a significant 2-3X reduction in training time and memory usage. (Table 2,3,6,7).
- TSMixer outperforms all other state-of-the-arts by a significant margin of 20-60%.

2 RELATED WORK

Transformer-based time series models: The success of Transformers [29] in the NLP domain has inspired time series researchers to come up with TS Transformer-based models. Due to the quadratic time and memory complexity of Transformer architectures, one major difficulty in adopting NLP Transformers for time series tasks lies in processing much longer input sequence lengths (L). To address this, several compute-aware time series Transformer models such as Informer [37], Autoformer [30], Pyraformer[20] and FEDformer [38] have been proposed, which modifies the self-attention mechanisms to achieve $O(L)$ or $O(L\log(L))$ complexity.

Patch-based time series models: The above-mentioned works mostly feed granular-level time series into the Transformer model, hence, they focus on learning attention over every time point. In contrast, PatchTST [22] and CrossFormer [3] enable patching before feeding it to the Transformers for learning representation across patches. PatchTST follows the channel independence approach and in contrast, CrossFormer follows the channel-mixing approach.

MLP-Mixer in vision domain: Recently, multi-layer perceptron's (MLP) strength has been reinvented in the vision domain [19, 25, 27]. The idea of [25] is to transform the input image through a series of permutation operations, which inherently enforces mixing of features within and across patches to capture short and long-term dependencies without the need for self-attention blocks. MLP-Mixer attains similar performance as CNN and Transformer. To extend upon, in gMLP [19], authors propose to use MLP-Mixer

[2]This "reconciliation" is different from the standard reconciliation in hierarchical forecasting [15]. Here, reconciliation targets patch-aggregation and cross-channel correlation and it is done online during training. We call it "online" as it is learned as part of the overall loss computation and not as a separate offline process.

[3]Gated Attention proposed in this paper is different from the Spatial Gating Unit proposed in gMLP [19]

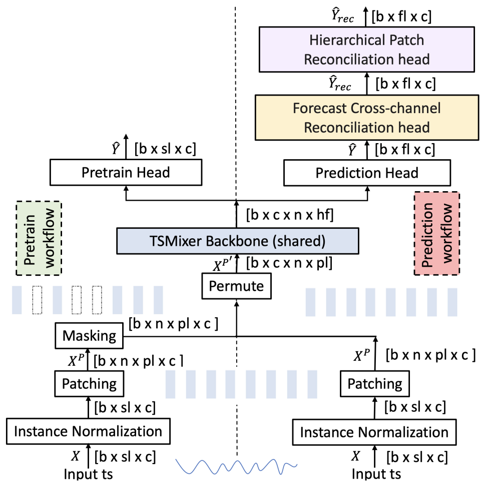


Figure 1: High-level model architecture.

with Spatial Gating Unit (SGU)[3] and ResMLP [27] proposes to use MLP-Mixer with residual connections.

MLP based sequence / time series models:[4] Similar efforts have recently been put in the time series domain. Zhang *et. al.* [36] proposed LightTS that is built upon MLP and two sophisticated downsampling strategies for improved forecasting. The DLinear model [34] also employs a simple linear model after decomposing the time series into trend and seasonality components. DLinear questions the effectiveness of Transformers as it easily beats Transformer-based SOTAs. Li *et. al.* [18] proposed MLP4Rec, a pure MLP-based architecture for sequential recommendation task, to capture correlations across time, channel, and feature dimensions.

3 METHODOLOGY

3.1 Notations

Throughout the paper, we use the following variable names: $\boldsymbol{X}_{c\times L}$: a multivariate time series of length L and c channels or c time series, $sl \leq L$: input sequence length, fl: forecast sequence length (a.k.a. horizon), b: batch size, n: number of patches, pl: patch length, hf: hidden feature dimension, ef: expansion feature dimension, nl: number of MLP-Mixer layers, $\mathcal{M}$: learned DNN model, op: number of output forecast patches, cl: context length, spl: patch length for cross-channel forecast reconciliation head, $\hat{H}$: patch-aggregated prediction, H: patch-aggregated ground truth, $\hat{Y}_{\text{rec}}$: actual base prediction, Y_{rec}: base ground truth, sf: scale factor. To ensure better understanding, we provide the shape of a tensor as its subscript in the text, and in square brackets in the architectural diagrams. We denote a linear layer in the neural network by $\mathcal{A}(\cdot)$ for compactness.

The multivariate forecasting task is defined as predicting future values of the time series given some history:

$$\hat{\boldsymbol{Y}}_{fl\times c} = \mathcal{M}\left(\boldsymbol{X}_{sl\times c}\right). \tag{1}$$

The ground truth future values will be denoted by $\boldsymbol{Y}_{fl\times c}$.

[4]Comparison with [9] is not included, as it was released in arXiv one month after the start of the KDD 2023 peer-review process [1], where our current paper has been submitted and got accepted.

3.2 Training methodologies

TSMixer has two training methodologies: supervised and self supervised. The supervised training follows the "prediction" workflow as shown in the right part of Figure 1. First, the input history time series goes through a sequence of transformation (normalization, patching, and permutation). Then, it enters the TSMixer backbone which is responsible for the main learning process. The prediction head converts the output embeddings of the backbone to the base forecasts, $\hat{Y}$. The model can be trained to minimize the mean squared error (MSE) of the base forecasts: $\mathcal{L}\left(Y,\hat{Y}\right) = \left\|Y-\hat{Y}\right\|_2^2$. We introduce two extra *online* forecast reconciliation heads which, if activated, can tune the base forecasts and produce more accurate forecasts by leveraging cross-channel and patch-aggregation information. When any or both of these reconciliation heads are activated, a customized MSE-based objective function is employed on the tuned forecasts. A detailed discussion is provided in Section 3.3.7.

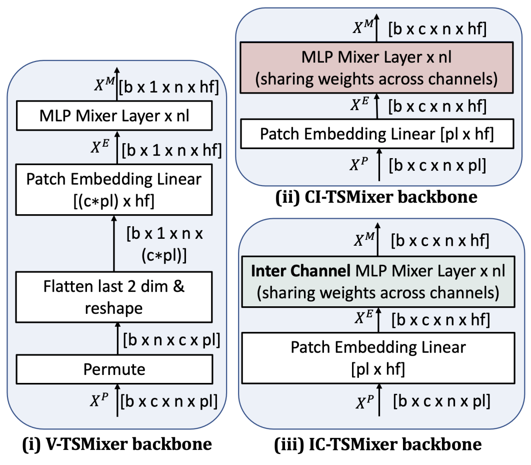
The self-supervised training is performed in two stages. First, the model is pretrained (see the "pretrain" workflow in Figure 1) with a self-supervised objective. Then, the pretrained model is finetuned through the "prediction" workflow for a supervised downstream task. Self-supervised pretraining has been found to be useful for a variety of NLP [10], vision [6], and time series tasks [22]. Similar to BERT's [10] masked language modeling (MLM) in the NLP domain, we employ a masked time series modeling (MTSM) task. The MTSM task randomly applies masks on a fraction of input patches, and the model is trained to recover the masked patches from the unmasked input patches. Other input transformations in the pretrain workflow are the same as in the prediction workflow. The MTSM task minimizes the MSE reconstruction error on the masked patches. The modular design of TSMixer enables it for either supervised or self-supervised training by only changing the model head (and keeping backbone the same).

3.3 Model components

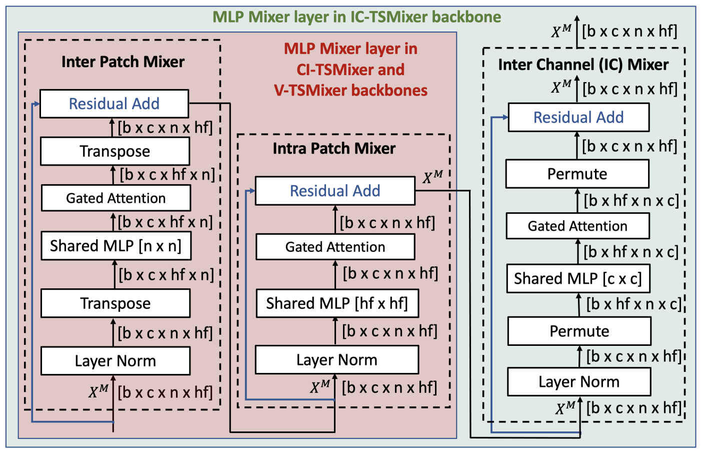
Here we discuss the modeling components that are introduced to a vanilla MLP-Mixer to have improved performance. The high-level architecture is shown in Figure 1. For stochastic gradient descent (SGD), each minibatch, $\boldsymbol{X}^{B}_{b\times sl\times c}$, is populated from $\boldsymbol{X}$ by a moving window technique. The forward pass of a minibatch along with its shape is shown in Figure 1

3.3.1 ***Instance normalization.*** The input time series segment goes through reversible instance normalization (RevIN) [17]. RevIN standardizes the data distribution (i.e., removes mean and divides by the standard deviation) to tackle data shifts in the time series.

3.3.2 ***Patching.*** Every univariate time series is segregated into overlapping / non-overlapping patches with a stride of s. For self-supervised training flow, patches have to be strictly non-overlapping. The minibatch $\boldsymbol{X}^{B}_{b\times sl\times c}$ is reshaped into $\boldsymbol{X}^{P}_{b\times n\times pl\times c}$, where pl denotes the patch length, and n is the number of patches (hence, $n = \lfloor(sl - pl)/s\rfloor + 1$). The patched data is then permuted to $\boldsymbol{X}^{P'}_{b\times c\times n\times pl}$ and fed to the TSMixer backbone model. Patching reduces the number of input tokens to the model by a factor of s, and



(a) Different backbones in TSMixer



(b) MLP Mixer layer architecture

Figure 2: Different backbones in TSMixer and the architecture of the mixer layers.

hence, increases the model runtime performance significantly as compared to standard point-wise Transformer approaches [22].

3.3.3 *TSMixer backbone.*

Three possible backbones are shown in Figure 2a. The vanilla backbone (V-TSMixer) flattens the channel and patch dimensions ($c \times pl$) on the input before passing to the next layer. This approach is commonly followed in Vision MLP Mixing techniques and hence acts as a vanilla baseline. We propose two new types of backbone: channel independent backbone (CI-TSMixer) and inter-channel backbone (IC-TSMixer). They differ in their MLP mixer layer architectures. CI-TSMixer backbone is inspired from the PatchTST [22] model, in which the MLP Mixer layer is shared across channels which forces the model to share learnable weights across the channels. This results in a reduction of model parameters. Moreover, CI-TSMixer enables us to employ TSMixer for self-supervised modeling over multiple datasets each having a different number of channels, which V-TSMixer cannot. In IC-TSMixer, an extra inter-channel mixer module is activated in the backbone to explicitly capture inter-channel dependencies. All these backbones will be compared with extensive experimentation.

Note that all the TSMixer backbones start with a linear patch embedding layer (see Figure 2a). It transforms every patch independently into an embedding: $\boldsymbol{X}^E_{b\times c\times n\times hf} = \mathcal{A}\left(\boldsymbol{X}^{P'}\right)$. The weight and bias of $\mathcal{A}(\cdot)$ are shared across channels for CI-TSMixer and IC-TSMixer backbones, but *not* for V-TSMixer. Since channels are completely flattened in V-TSMixer, $\mathcal{A}(\cdot)$ in V-TSMixer does not have any notion of multiple channels (i.e. c = 1).

3.3.4 *MLP Mixer layers.*

TSMixer backbone stacks a set of mixer layers like encoder stacking in Transformers. Intuitively, each mixer layer (Figure 2b) tries to learn correlations across three different directions: (i) between different patches, (ii) between the hidden feature inside a patch, and (iii) between different channels. The former two mixing methods are adopted from the vision MLP-Mixer , while the last one is proposed particularly for multivariate time series data. The **inter patch mixer** module employs a shared MLP (weight dimension = $n \times n$) to learn correlation between different patches. The **intra patch mixer** block's shared MLP layer mixes the dimensions of the hidden features, and hence the weight matrix has a dimension of $hf \times hf$. The proposed **inter channel mixer** (weight matrix size = $c \times c$) mixes the input channel dimensions, and tries to capture correlations between multiple channels in a multivariate context. This inter-channel mixer has been proposed in MLP4Rec [18] for the event prediction problem and we investigate its applicability for the time-series domain. Please note that inter-channel mixer block is included only in the IC-TSMixer backbone and not with the CI-TSMixer and V-TSMixer backbones (Figure 2b). The input and output of the mixer layers and mixer blocks are denoted by $\boldsymbol{X}^M_{b\times c\times n\times hf}$. Based on the dimension under focus in each mixer block, the input gets reshaped accordingly to learn correlation along the focussed dimension. The reshape gets reverted in the end to retain the original input shape across the blocks and layers. All of the three mixer modules are equipped with an MLP block (Appendix Figure. 6a), layer normalization [4], residual connection [14], and gated attention. The former three components are standard in MLP-Mixer while the gated attention block is described below.

3.3.5 *Gated attention (GA) block.*

Time series data often has a lot of unimportant features that confuse the model. In order to effectively filter out these features, we add a simple Gated Attention [5] after the MLP block in each mixer component. GA acts like a simple gating function that probabilistically upscales the dominant features and downscales the unimportant features based on its feature values. The attention weights are derived by: $\boldsymbol{W}^A_{b\times c\times n\times hf} = \text{softmax}\left(\mathcal{A}\left(\boldsymbol{X}^M\right)\right)$. The output of the gated attention module is obtained by performing a dot product between the attention weights and the hidden tensor coming out of the mixer modules: $\boldsymbol{X}^G = \boldsymbol{W}^A \cdot \boldsymbol{X}^M$ (Appendix Figure. 6b). Augmenting GA with standard mixer operations effectively guides the model to focus

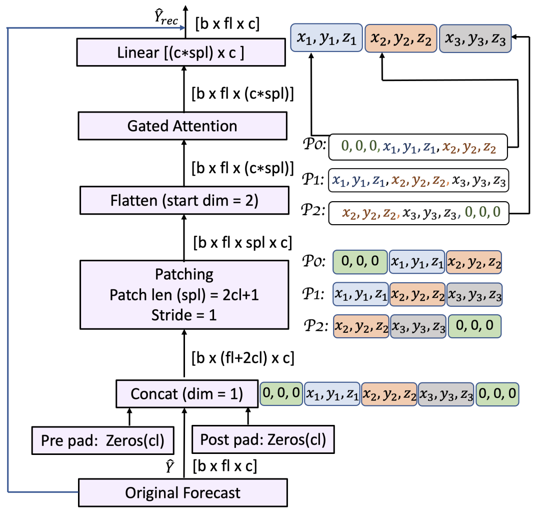


Figure 3: Cross-channel forecast reconciliation head.

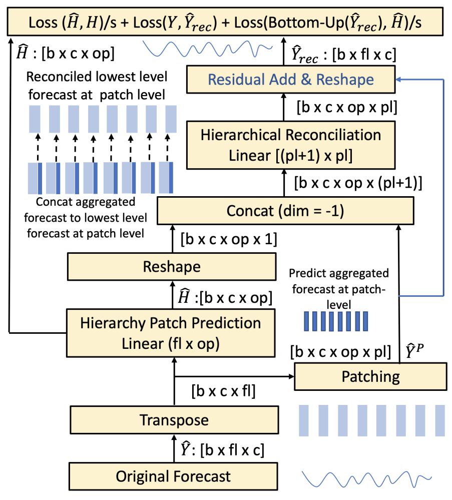

Loss ($\hat{H}$, H)/s + Loss(Y, $\hat{Y}_{rec}$) + Loss(Bottom-Up($\hat{Y}_{rec}$), $\hat{H}$)/s
$\hat{H}$: [b x c x op]
$\hat{Y}_{rec}$: [b x fl x c]
Reconciled lowest level forecast at patch level
Residual Add & Reshape
[b x c x op x pl]
Hierarchical Reconciliation Linear [(pl+1) x pl]
[b x c x op x (pl+1)]
Concat aggregated forecast to lowest level forecast at patch level
Concat (dim = -1)
[b x c x op x 1]
Reshape
Predict aggregated forecast at patch-level
$\hat{H}$:[b x c x op]
Hierarchy Patch Prediction Linear (fl x op)
[b x c x op x pl] $\hat{Y}^P$
Patching
[b x c x fl]
Transpose
$\hat{Y}$: [b x fl x c]
Original Forecast

Figure 4: Online hierarchical patch reconciliation head.

on the important features leading to improved long-term interaction modeling, without requiring the need for complex multi-head self-attention.

3.3.6 ***Model heads.*** Based on the training methodology (i.e. supervised or self-supervised), we either add prediction or pretrain head to the backbone. Both heads employ a simple Linear layer with dropout after flattening the hidden features across all patches (Appendix Figure. 7). By default, heads share the same weights across channels. The output of the prediction head is the predicted multivariate time series ($\hat{Y}_{b\times fl\times c}$), while the pretrain head emits a multivariate series of the same dimension as the input ($\hat{X}_{b\times sl\times c}$).

3.3.7 ***Forecast online reconciliation.*** Here we propose two novel methods (in the prediction workflow, see Figure 1) to tune the original forecasts, $\hat{Y}$, based on two important characteristics of time series data: inherent temporal hierarchical structure and cross-channel dependency. Any or both of them can be activated in our TSMixer model to obtain reconciled forecasts.

Cross-channel forecast reconciliation head: In many scenarios, the forecast for a channel at a certain time might be dependent on the forecast of another channel at a different point in time on the horizon. For example, in retail domain, sales at a future time point might be dependent on the discount patterns around that time. Hence, we introduce a cross-channel forecast reconciliation head which derives an objective that attempts to learn cross-dependency across channels within a local surrounding context *in the forecast horizon.* Figure 3 demonstrates its architecture.

First, every forecast point is converted into a patch (of length *spl*) by appending its pre and post-surrounding forecasts based on the context length (*cl*). Then, each patch is flattened across channels and passed through a gated attention and linear layer to obtain a revised forecast point for that patch. Thus, all channels of a forecast point reconcile its values based on the forecast channel values in the surrounding context leading to effective cross-channel modeling. Residual connections ensure that reconciliation does not lead to accuracy drops in scenarios when the channel correlations are very noisy. Since the revised forecasts have the same dimension as the original forecasts, no change to the loss function is required. From experiments, we observe that the "hybrid" approach of having a channel-independent backbone augmented with a cross-channel reconciliation head provides stable improvements as compared to other channel-mixing approaches. Also, this architecture helps in better backbone generalization as it can be trained with multiple datasets with a varying number of channels while offloading the channel-correlation modeling to the prediction head (which is task and data-dependent).

Online hierarchical patch reconciliation head:[5]. Time series data often possess an inherent hierarchical structure, either explicitly known (e.g., hierarchical forecasting datasets [15]), or as an implicit characteristic (e.g., an aggregation of weather forecasts for seven days denotes an estimate of weekly forecast, an aggregation of sales forecast over all stores in a state denotes state-level sales, and so on). In general, aggregated time series have better predictability and a good forecaster aims at achieving low forecast error in all levels of the hierarchy ([15, 16, 21]). Here, we propose a novel method to automatically derive a hierarchical patch aggregation loss (online during training) that is minimized along with the granular-level forecast error. Figure 4 shows the architecture. The original forecast $\hat{Y}$ is segregated into *op* number of patches, each of length *pl*. We denote this as $\hat{Y}^P$. Now, $\hat{Y}$ is also passed through a linear layer to predict the hierarchically aggregated forecasts at the patch-level: $\hat{H}_{b\times c\times op} = \mathcal{A}\left(\hat{Y}_{b\times c\times fl}\right)$. Then, we concatenate $\hat{Y}^P$ and $\hat{H}$ at the patch level and pass it through another linear transformation to obtain reconciled granular-level forecast: $\hat{Y}_{\text{rec}}$. Thus, the granular-level forecasts get reconciled at a patch level based on the patch-aggregated forecasts leading to improved granular-level forecasts. Residual connections ensure that the reconciliation

[5]This "reconciliation" is different from the reconciliation in hierarchical forecasting [15]. Here, hierarchy is defined as an aggregation over a patch, and the reconciliation is done online during training.

does not lead to accuracy drops in scenarios when predicting aggregated signals become challenging. Now, the hierarchical patch aggregation loss is calculated as follows:

$$\mathcal{L}_{\text{hier}} = \frac{1}{sf}\mathcal{L}\left(\hat{H}, H\right) + \mathcal{L}\left(Y, \hat{Y}_{\text{rec}}\right) + \frac{1}{sf}\mathcal{L}\left(\text{BU}\left(\hat{Y}_{\text{rec}}\right), \hat{H}\right) \quad (2)$$

where, Y is ground-truth future time series, H is the aggregated ground-truth at patch-level, BU refers to bottom-up aggregation of the granular-level forecasts to obtain the aggregated patch-level forecasts [15], and sf is scale factor. For MSE loss, $sf = (pl)^2$. More intuitively, this loss tries to tune the base forecasts in a way such that they are not only accurate at the granular-level, but also accurate at the aggregated patch-level. Note that a pre-defined dataset-specific hierarchical structure can be enforced here, but it is left for future studies.

Datasets	Weather	Traffic	Electricity	ETTH1	ETTH2	ETTM1	ETTM2
Features	21	862	321	7	7	7	7
Timesteps	52696	17544	26304	17420	17420	69680	69680

Table 1: Statistics of popular datasets for benchmark.

4 EXPERIMENTS

4.1 Experimental settings

4.1.1 ***Datasets.*** We evaluate the performance of the proposed TSMixer model on 7 popular multivariate datasets as depicted in Table 1. These datasets have been extensively used in the literature [22][34][30] for benchmarking multivariate forecasting models and are publically available in [35]. We follow the same data loading parameters (Ex. train/val/test split ratio) as followed in [22].

4.1.2 ***Model Variants.*** A TSMixer variant is represented using the naming convention "*BackboneType*"-TSMixer("*Enhancements*"). "*BackboneType*" can either be Vanilla (V), or Channel Independent (CI), or Inter Channel (IC). "*Enhancements*" can be a combination of Gated Attention (G), Hierarchical Patch Reconciliation head (H), and/or Cross-channel Reconciliation head (CC). Common variants are:

- **CI-TSMixer**: Channel Independent TSMixer with no enhancements
- **CI-TSMixer(G,H)**: Channel Independent TSMixer with Gated Attention and Hierarchy Reconciliation head
- **CI-TSMixer(G,H,CC)**: Channel Independent TSMixer with Gated Attention, Hierarchy and Cross-Channel Reconciliation head
- **CI-TSMixer-Best**: Best of top performing TSMixer variants [i.e. CI-TSMixer(G,H) and CI-TSMixer(G,H,CC)]
- **V-TSMixer**: Vanilla TSMixer
- **IC-TSMixer**: Inter-Channel TSMixer

Other model variants can be formed using the same naming convention. Unless stated explicitly, the cross-channel reconciliation head uses the default context length of 1.

4.1.3 ***Data & Model Configuration.*** By default, the following data and model configuration is used: Input Sequence length sl = 512, Patch length pl = 16, Stride s = 8, Batch size b = 8, Forecast sequence length $fl \in \{96, 192, 336, 720\}$, Number of Mixer layers nl = 8, feature scaler fs = 2, Hidden feature size $hf = fs * pl$ (32), Expansion feature size $ef = fs * hf$ (64) and Dropout do = 0.1. Training is performed in a distributed fashion with 8 GPUs, 10 CPUs and 1000 GB of memory. For ETT datasets, we use a lower hardware and model configuration with high dropout to avoid overfitting, as the dataset is relatively small (nl = 3, do = 0.7, $ngpus$ = 1). Supervised training is performed with 100 epochs. In self-supervised training, we first pretrain the backbone with 100 epochs. After that, in the finetuning phase, we freeze the backbone weights for the first 20 epochs to train/bootstrap the head (also known as *linear probing*), and then, we finetune the entire network (backbone + head) for the next 100 epochs. We choose the final model based on the best validation score. Since overlapping patches have to be avoided in self-supervised methodology, we use reduced patch length and stride with the same size there (i.e. pl = 8, s = 8). This further updates the hidden feature and expansion feature size by fs (i.e. hf = 16, ef = 32) for self-supervised methodology. Every experiment is executed with 5 random seeds (from 42-46) and the mean scores are reported. Standard deviation is also reported for the primary results. We use mean squared error (MSE) and mean absolute error (MAE) as the standard error metrics.

4.1.4 ***SOTA Benchmarks.*** We categorize SOTA forecasting benchmarks into the following categories: **(i) Standard Transformers:** FEDformer [38], Autoformer[30] and Informer [37], **(ii) Patch Transformers:** PatchTST [22] and CrossFormer [3], **(iii) MLPs and Non-Transformers:** DLinear [34], LightTS [36] and S4 [12], **(iv) Self-supervised models:** BTSF [31], TNC [26], TS-TCC [11], CPC [28], and TS2Vec [33].

4.2 Supervised Multivariate Forecasting

In this section, we compare the accuracy and computational improvements of TSMixer with the popular benchmarks in supervised multivariate forecasting.

4.2.1 ***Accuracy Improvements.*** In Table 2, we compare the accuracy of TSMixer Best variant (i.e. CI-TSMixer-Best) with SOTA benchmarks. Since we observe similar relative patterns in MSE and MAE, we explain all the results in this paper using the MSE metric. TSMixer outperforms standard Transformer and MLP benchmarks by a significant margin (DLinear: 8%, FEDformer: 23%, Autoformer: 30% and Informer: 64%). PatchTST (refers to PatchTST/64 in [22]) is one of the strongest baselines, and TSMixer marginally outperforms it by 1%. However, TSMixer achieves this improvement over PatchTST with a significant performance improvement *w.r.t.* training time and memory (Section. 4.2.2). For exhaustive results with the individual best TSMixer variants, refer to Appendix Table 11. Also, Appendix A.4.1 highlights the superior performance of TSMixer with other secondary benchmarks (LightTS, S4 and CrossFormer).

4.2.2 ***Computational Improvements.*** PatchTST is considered the most compute-effective model across all time series Transformers, as patching drastically reduces the input size by a factor of s

Models		**CI-TSMixer-Best**		DLinear		PatchTST		FEDformer		Autoformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTH1	96	**0.368±0.001**	**0.398±0.001**	0.375	0.399	0.370	0.400	0.376	0.419	0.449	0.459	0.865	0.713
	192	**0.399±0.002**	0.418±0.001	0.405	**0.416**	0.413	0.429	0.420	0.448	0.500	0.482	1.008	0.792
	336	**0.421±0.004**	**0.436±0.003**	0.439	0.443	0.422	0.440	0.459	0.465	0.521	0.496	1.107	0.809
	720	**0.444±0.003**	**0.467±0.002**	0.472	0.490	0.447	0.468	0.506	0.507	0.514	0.512	1.181	0.865
ETTH2	96	0.276±0.006	**0.337±0.003**	0.289	0.353	**0.274**	**0.337**	0.346	0.388	0.358	0.397	3.755	1.525
	192	**0.330±0.003**	**0.374±0.001**	0.383	0.418	0.341	0.382	0.429	0.439	0.456	0.452	5.602	1.931
	336	0.357±0.001	0.401±0.002	0.448	0.465	**0.329**	**0.384**	0.496	0.487	0.482	0.486	4.721	1.835
	720	0.395±0.003	0.436±0.003	0.605	0.551	**0.379**	**0.422**	0.463	0.474	0.515	0.511	3.647	1.625
ETTM1	96	**0.291±0.002**	0.346±0.002	0.299	**0.343**	0.293	0.346	0.379	0.419	0.505	0.475	0.672	0.571
	192	**0.333±0.002**	0.369±0.002	0.335	**0.365**	**0.333**	0.370	0.426	0.441	0.553	0.496	0.795	0.669
	336	**0.365±0.005**	**0.385±0.004**	0.369	0.386	0.369	0.392	0.445	0.459	0.621	0.537	1.212	0.871
	720	**0.416±0.002**	**0.413±0.001**	0.425	0.421	**0.416**	0.420	0.543	0.490	0.671	0.561	1.166	0.823
ETTM2	96	**0.164±0.002**	**0.255±0.002**	0.167	0.260	0.166	0.256	0.203	0.287	0.255	0.339	0.365	0.453
	192	**0.219±0.002**	**0.293±0.002**	0.224	0.303	0.223	0.296	0.269	0.328	0.281	0.340	0.533	0.563
	336	**0.273±0.003**	**0.329±0.003**	0.281	0.342	0.274	**0.329**	0.325	0.366	0.339	0.372	1.363	0.887
	720	**0.358±0.002**	**0.380±0.001**	0.397	0.421	0.362	0.385	0.421	0.415	0.433	0.432	3.379	1.338
Electricity	96	**0.129±1e-4**	0.224±0.001	0.140	0.237	**0.129**	**0.222**	0.193	0.308	0.201	0.317	0.274	0.368
	192	**0.146±0.001**	0.242±1e-4	0.153	0.249	0.147	**0.240**	0.201	0.315	0.222	0.334	0.296	0.386
	336	**0.158±0.001**	**0.256±0.001**	0.169	0.267	0.163	0.259	0.214	0.329	0.231	0.338	0.300	0.394
	720	**0.186±0.001**	**0.282±0.001**	0.203	0.301	0.197	0.290	0.246	0.355	0.254	0.361	0.373	0.439
Traffic	96	**0.356±0.002**	**0.248±0.002**	0.410	0.282	0.360	0.249	0.587	0.366	0.613	0.388	0.719	0.391
	192	**0.377±0.003**	0.257±0.002	0.423	0.287	0.379	**0.256**	0.604	0.373	0.616	0.382	0.696	0.379
	336	**0.385±0.002**	**0.262±0.001**	0.436	0.296	0.392	0.264	0.621	0.383	0.622	0.337	0.777	0.420
	720	**0.424±0.001**	**0.283±0.001**	0.466	0.315	0.432	0.286	0.626	0.382	0.660	0.408	0.864	0.472
Weather	96	**0.146±0.001**	**0.197±0.002**	0.176	0.237	0.149	0.198	0.217	0.296	0.266	0.336	0.300	0.384
	192	**0.191±0.001**	**0.240±0.001**	0.220	0.282	0.194	0.241	0.276	0.336	0.307	0.367	0.598	0.544
	336	**0.243±0.001**	**0.279±0.002**	0.265	0.319	0.245	0.282	0.339	0.380	0.359	0.395	0.578	0.523
	720	0.316±0.001	**0.333±0.002**	0.323	0.362	**0.314**	0.334	0.403	0.428	0.419	0.428	1.059	0.741
CI-TSMixer-Best % improvement (MSE)				**8%**		**1%**		**23%**		**30%**		**64%**	

Table 2: Comparing TSMixer with popular benchmarks in supervised long-term multivariate forecasting. The best results are in bold and the second best is underlined. PatchTST results are reported from [22]. All other benchmarks are reported from [34]

(stride) [22]. In contrast, TSMixer not only enables patching but also completely eliminates the self-attention blocks. Hence, TSMixer shows significant computation improvement over PatchTST and other Transformer models. In Table 3, we highlight the speed-up and memory comparison of TSMixer over PatchTST. For analysis, we capture the following metrics: (i) Multiply-Add cumulative operations on the entire data per epoch (MACs), (ii) Number of model parameters (NPARAMS), (iii) Single EPOCH TIME and (iv) Peak GPU memory reached during a training run (MAX MEMORY). For this experiment, we trained TSMixer and PatchTST models in a single GPU node with the same hardware configuration in a non-distributed manner to report the results. To ensure fairness in comparison, we use the exact model parameters of PatchTST and TSMixer which were used in the error metric comparison reported in Table 2. In Table 3, we capture the average improvement of TSMixer over PatchTST for each performance metric across the three larger datasets (Electricity, Weather, Traffic) with fl=96. Since CI-TSMixer is purely MLP based, it significantly reduces the average MACs (by ~4X), NPARAMS & MAX MEMORY (by ~3X), and EPOCH TIME [6] (by ~2X). Even after enabling gated attention and hierarchy reconciliation, CI-TSMixer(G,H) still shows a good reduction in MACs & NPARAMS (by ~3X), and EPOCH TIME & MAX MEMORY (by ~2X). It is important to note that, when cross-channel reconciliation is enabled [i.e CI-TSMixer(G,H,CC)], the number of parameters becomes very high in TSMixer as compared to PatchTST. The reason is that the number of parameters in the cross-channel reconciliation head is dependent on the number of channels in the dataset that leads to this scaling effect. For example, since Electricity and Traffic datasets have a very high number of channels (i.e. 321 and 862 respectively), the number of parameters of the model also scales up, whereas weather (with only 21 channels) did not encounter any such scaling effect. Even PatchTST should show a similar scaling effect if the channel correlation is enabled in it. However, even with increased parameters, CI-TSMixer(G,H,CC) still shows a notable reduction in MACs (by ~3X) and EPOCH TIME & MAX MEMORY (by ~2X). The reason is that parameter scaling affects only the reconciliation head and not the backbone which primarily constitutes the total training time and memory. Thus, TSMixer and its variants can easily produce improved results over PatchTST in significantly less training time and memory utilization.

[6] MACs and EPOCH TIME are highly correlated and in general create a similar relative impact across models. However, since PatchTST involves high parallelism across the attention heads, we observe a different relative impact *w.r.t.* MACs and EPOCH TIME.

4.3 Component & Design Choice Analysis

In this section, we study the impact of various key components & design choices followed in the TSMixer.

Metric	Data	Patch TST	CI-TSMixer (Avg. Imp)		CI-TSMixer (G,H) (Avg. Imp)		CI-TSMixer (G,H,CC) (Avg. Imp)	
MACs (T)	Electricity	147.879	37.062	**(4X)**	46.44	**(3.2X)**	48.566	**(3X)**
	Traffic	260.367	65.25		81.77		91.78	
	Weather	19.709	4.94		6.19		6.21	
NPARAMS (M)	Electricity	1.174	0.348	**(3.4X)**	0.4	**(2.9X)**	1.648	**(1.2X)**
	Traffic	1.174	0.348		0.4		9.33	
	Weather	1.174	0.348		0.4		0.405	
EPOCH TIME (min)	Electricity	36.22	15.56	**(2X)**	20.43	**(1.6X)**	20.5	**(1.5X)**
	Traffic	64.04	27.49		36.08		36.51	
	Weather	5.2	3.08		4.13		4.17	
MAX MEMORY (GB)	Electricity	6.14	2.25	**(2.7X)**	2.9	**(2.1X)**	2.94	**(2X)**
	Traffic	8.24	3.03		3.89		4.15	
	Weather	0.451	0.165		0.21		0.211	

Table 3: Computational Improvement. nX denotes n times average improvement across datasets (Avg. Imp).

	FL	V-TSMixer	CI-TSMixer	IC-TSMixer	CI-TSMixer (G,CC-Best)
ETTH1	96	0.449	0.375	0.379	**0.373**
	192	0.485	0.411	0.416	**0.407**
	336	0.504	0.437	0.437	**0.43**
	720	0.573	0.465	0.471	**0.454**
ETTH2	96	0.369	0.284	0.291	**0.269**
	192	0.391	0.353	0.345	**0.330**
	336	0.403	0.365	0.361	**0.359**
	720	0.475	0.406	0.419	**0.393**
Weather	96	0.159	0.150	0.150	**0.146**
	192	0.207	0.195	0.196	**0.194**
	336	0.256	**0.246**	0.248	**0.246**
	720	0.330	0.323	0.338	**0.317**
% improvement over V-TSMixer			**11.5%**	**10.7%**	**13.5%**

Table 5: Channel mixing technique comparison (MSE).

	FL	V-TSMixer	CI-TSMixer	CI-TSMixer (G)	CI-TSMixer (H)	CI-TSMixer (G,H)
ETTH1	96	0.449	0.375	0.375	0.377	**0.368**
	192	0.485	0.411	0.408	0.410	**0.399**
	336	0.504	0.437	0.433	0.431	**0.421**
	720	0.573	0.465	0.454	0.457	**0.444**
ETTM1	96	0.328	0.305	**0.296**	0.301	0.297
	192	0.372	0.336	0.334	0.338	**0.333**
	336	0.405	0.375	0.366	0.376	**0.365**
	720	0.457	0.425	0.419	0.422	**0.416**
Weather	96	0.159	0.150	0.149	0.151	**0.148**
	192	0.207	0.195	0.195	0.195	**0.193**
	336	0.256	0.246	0.246	0.246	**0.243**
	720	0.330	0.323	0.317	0.321	**0.317**
% improvement over V-TSMixer			**9.5%**	**10.5%**	**10%**	**11.5%**

Table 4: Effect of CI, Gated Attention and Hierarchy Reconciliation over Vanilla TSMixer (MSE).

4.3.1 *Effect of CI, Gated Attention & Hierarchy Patch Reconciliation.*

Table 4 depicts the improvements of various enhancement components in TSMixer over V-TSMixer in three datasets: ETTH1, ETTM1, and Weather (for space constraint). V-TSMixer represents the vanilla model where all channels are flattened and processed together (similar to vision MLP-Mixer). CI-TSMixer outperforms V-TSMixer by 9.5% by introducing channel independence (CI) in the backbone instead of channel flattening. By further adding gated attention (G) and hierarchy reconciliation (H) together [i.e. CI-TSMixer(G,H)], we observe an additional 2% improvement leading to a total of 11.5% improvement *w.r.t.* V-TSMixer. On analysis with all datasets (Appendix Table 12), we observe that CI-TSMixer(G,H) outperforms V-TSMixer by an avg. of 19.3%. In general, adding 'G' and 'H' together leads to more stable improvements in CI-TSMixer as compared to just adding 'G' or 'H'.

4.3.2 *Hybrid Channel Modelling Approach.*

In Table 5, we compare various channel-mixing techniques and highlight the benefits of "hybrid" channel modeling followed in TSMixer. In summary, CI-TSMixer outperforms V-TSMixer by 11.5%, and by adding cross-channel reconciliation head (CC), the accuracy further improves by 2% leading to an overall improvement of 13.5% for CI-TSMixer(G,CC-Best) (i.e. channel independent backbone with CC head) Context Length (cl) is an important hyperparameter in the CC head, which decides the surrounding context space across channels to enable reconciliation, and this parameter has to be decided based on the underlying data characteristics. For this experiment, we varied cl from 1 to 5 and selected the best which is depicted as CI-TSMixer(G,CC-Best) in Table 5. For more exhaustive results on various cl and all datasets, refer to the Appendix Table. 13. Moreover, we observe from Table 5 that CI-TSMixer(G,CC-Best) performs better compared to the IC-TSMixer which applies cross-channel correlation inside the backbone. In addition, CrossFormer [3] proposes an alternative patch cross-channel correlation approach which TSMixer significantly outperforms by 30% (Appendix Table. 9). Thus, the "hybrid" channel modeling approach of having a *channel-independent backbone augmented with a cross-channel reconciliation head* is relatively robust to the noisy interactions across the channels. Also, from the architectural perspective - this approach helps in pretraining the backbone on multiple datasets with a varying number of channels. This cannot be achieved trivially with channel-mixing backbones.

4.4 Forecasting via Representation Learning

In this section, we compare TSMixer with popular benchmarks on multivariate forecasting via self-supervised representation learning.

4.4.1 *Accuracy Improvements.*

In Table 6, we compare the forecasting results of TSMixer with self-supervised benchmarks. For BTSF, TNC, TS-TCC and CPC - we report the results from [31]. For self-supervised PatchTST, we report ETTH1 from [22] and calculated it for Weather as it was unavailable. We use the fl space commonly reported across [31] and [22]. In the self-supervised workflow, the TSMixer backbone is first pre-trained to learn a generic patch representation, and then the entire network (backbone + head) is finetuned for the forecasting task. By training TSMixer with this approach, we observe from Table. 6 that CI-TSMixer-Best achieves a significant improvement (50-70% margin) from existing forecasting benchmarks learned via self-supervision (such as BFSF, TNC,

	FL	CI-TSMixer -Best	PatchTST	BTSF	TNC	TS-TCC	CPC
ETTH1	24	**0.314**	0.322	0.541	0.632	0.653	0.687
	48	**0.343**	0.354	0.613	0.705	0.720	0.779
	168	**0.397**	0.419	0.640	1.097	1.129	1.282
	336	**0.424**	0.445	0.864	1.454	1.492	1.641
	720	**0.453**	0.487	0.993	1.604	1.603	1.803
Weather	24	0.088	**0.087**	0.324	0.484	0.572	0.647
	48	0.114	**0.113**	0.366	0.608	0.647	0.720
	168	**0.177**	0.178	0.543	1.081	1.117	1.351
	336	**0.241**	0.244	0.568	1.654	1.783	2.019
	720	**0.319**	0.321	0.601	1.401	1.850	2.109
CI-TSMixer-Best % improvement			**2.3%**	**54.2%**	**71.7%**	**73.3%**	**75.8%**

Table 6: Forecasting via Representation Learning (MSE)

TS-TCC, CPC). Similar trends were also observed with TS2Vec [33] (Appendix Table 10). Also, CI-TSMixer-Best beats self-supervised PatchTST by a margin of 2%. However, as explained in Section. 4.2.2, we achieve this improvement over PatchTST with a significant reduction in time and memory. For a drill-down view of CI-TSMixer-Best, refer to Appendix Table. 14

4.4.2 *Pretrain strategies in Self Supervision.*

In Table. 7, we do a detailed analysis of the self-supervised approach on 3 large datasets (Electricity, Traffic, and Weather) with 3 different pretrain data creation strategies as follows: (i) SAME (same primary data for pretrain and finetune), (ii) ALL (pretrain with all data [ETT, Electricity, Traffic and Weather] and finetune with the primary data), (iii) TL (transfer learning: pretrain with all data except primary data and finetune with the primary data). Since we learn across multiple datasets, we use a bigger model size for this experiment (i.e. $nl = 12$, $fs = 3$) for better modeling capacity. From Table. 7, we observe that all three considered data strategies work equally well with marginal variations across them. However, the benefits of transfer learning across multiple datasets are not very notable, as we observe in other domains (like vision and text). On average, CI-TSMixer-Overall-Best(SS) [which indicates the best result across the considered data strategy variants] shows improved performance *w.r.t.* self-supervised PatchTST (reported from [22]) and supervised TSMixer by 1.5%. Thus, enabling self-supervision before the forecasting task helps in improving forecast accuracy. Furthermore, it helps in faster convergence for downstream tasks.

4.4.3 *Patch Representations.*

To understand the semantic meaning of the learned patch representations, we randomly choose 5 patch embeddings (i.e. output of TSMixer backbone after pretraining) from ETTH1 and fetched its nearest 50 embeddings to form 5 clusters in the patch embedding space (Figure. 5(b)). We then fetch their associated patch time series and plot in Figure. 5(a). From the figure, we observe that nearby patch representations highly correlate to the patch time series of similar shapes and patterns, thereby learning meaningful patch representations that can effectively help in the finetuning process for various downstream tasks. Please refer to Appendix Figure. 8 for visualizations on more datasets.

	FL	CI-TSMixer-Best (SS) Model Size: Big			**CI-TSMixer-Overall -Best (SS)**	PatchTST (SS)	CI-TSMixer-Best
		SAME	ALL	TL			Supervised
Electricity	96	0.127	0.128	0.129	0.127	**0.126**	0.129
	192	**0.145**	0.146	0.147	**0.145**	**0.145**	0.146
	336	**0.156**	**0.156**	0.159	**0.156**	0.164	0.158
	720	0.187	**0.185**	0.189	**0.185**	0.193	0.186
Traffic	96	0.350	**0.348**	0.350	**0.348**	0.352	0.356
	192	0.372	**0.370**	0.372	**0.370**	0.371	0.377
	336	**0.379**	0.380	**0.379**	**0.379**	0.381	0.385
	720	**0.420**	0.421	0.422	**0.420**	0.425	0.424
Weather	96	0.144	**0.143**	0.144	**0.143**	0.144	0.146
	192	**0.189**	**0.189**	0.190	**0.189**	0.190	0.191
	336	0.240	**0.239**	0.241	**0.239**	0.244	0.243
	720	**0.311**	0.316	0.319	**0.311**	0.321	0.316
CI-TSMixer-Overall-Best (SS) % Improvement (MSE)						**1.45%**	**1.41%**

Table 7: Self-Supervision(SS) with varying pretrain strategies (MSE). PatchTST (SS) results are reported from [22](SS finetuning mode)

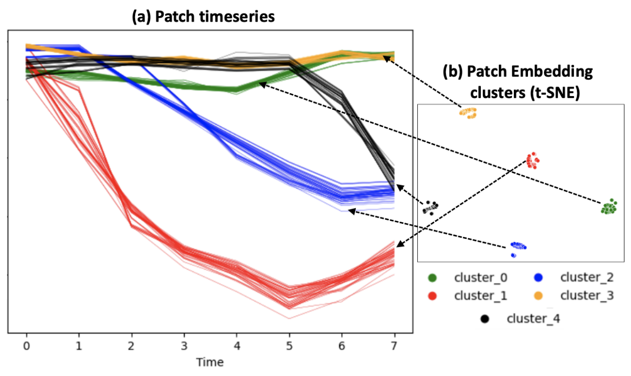


Figure 5: Correlation between Patch time-series and its associated embeddings.

5 CONCLUSIONS AND FUTURE DIRECTIONS

Inspired by the success of MLP-Mixers in the vision domain, this paper proposes **TSMixer**, a purely designed MLP architecture with empirically validated time-series specific enhancements for multivariate forecasting and representation learning. Especially, we introduce a new hybrid architecture of augmenting various reconciliation heads and gated attention to the channel-independent backbone that significantly empowers the learning capability of simple MLP structures to outperform complex Transformer models. Through extensive experimentation, we show that TSMixer outperforms all popular benchmarks with a significant reduction in compute resources. In future work, we plan to extend TSMixer to other downstream tasks (such as classification, anomaly detection, etc.) and also improve the transfer learning capabilities across datasets. We also plan to investigate Swin[13], Shift[32], and other newer Mixer variants[8][24] for its applicability in time-series.

REFERENCES

[1] 2023. KDD 2023 Conference details. (2023). https://kdd.org/kdd2023/call-for-research-track-papers/

[2] Mohanad S Al-Musaylh, Ravinesh C Deo, Jan F Adamowski, and Yan Li. 2018. Short-term electricity demand forecasting with MARS, SVR and ARIMA models using aggregated demand data in Queensland, Australia. *Advanced Engineering Informatics* 35 (2018), 1–16.

[3] Anonymous. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. (2023). https://openreview.net/forum?id=vSVLM2j9eie under review.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[6] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. BEiT: BERT Pre-Training of Image Transformers. In *International Conference on Learning Representations*. https://openreview.net/forum?id=p-BhZSz59o4

[7] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[8] Shoufa Chen, Enze Xie, Chongjian Ge, Runjian Chen, Ding Liang, and Ping Luo. 2021. CycleMLP: A MLP-like Architecture for Dense Prediction. (2021). https://doi.org/10.48550/ARXIV.2107.10224

[9] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. 2023. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053* (2023).

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[11] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. 2021. Time-Series Representation Learning via Temporal and Contextual Contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. 2352–2359.

[12] Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently Modeling Long Sequences with Structured State Spaces. (2021). https://arxiv.org/abs/2111.00396

[13] Jianyuan Guo, Yehui Tang, Kai Han, Xinghao Chen, Han Wu, Chao Xu, Chang Xu, and Yunhe Wang. 2021. Hire-MLP: Vision MLP via Hierarchical Rearrangement. (2021). https://doi.org/10.48550/ARXIV.2108.13341

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[15] R.J. Hyndman and G. Athanasopoulos (Eds.). 2021. *Forecasting: principles and practice*. OTexts: Melbourne, Australia. OTexts.com/fpp3.

[16] Arindam Jati, Vijay Ekambaram, Shaonli Pal, Brian Quanz, Wesley M. Gifford, Pavithra Harsha, Stuart Siegel, Sumanta Mukherjee, and Chandra Narayanaswami. 2022. Hierarchy-guided Model Selection for Time Series Forecasting. https://doi.org/10.48550/ARXIV.2211.15092

[17] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2022. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *International Conference on Learning Representations*. https://openreview.net/forum?id=cGDAkQo1C0p

[18] Muyang Li, Xiangyu Zhao, Chuan Lyu, Minghao Zhao, Runze Wu, and Ruocheng Guo. 2022. MLP4Rec: A Pure MLP Architecture for Sequential Recommendations. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2138–2144. https://doi.org/10.24963/ijcai.2022/297 Main Track.

[19] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. 2021. Pay attention to mlps. *Advances in Neural Information Processing Systems* 34 (2021), 9204–9215.

[20] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2022. Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. In *International Conference on Learning Representations*.

[21] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2022. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting* (2022). https://www.sciencedirect.com/science/article/pii/S0169207021001874 https://doi.org/10.1016/j.ijforecast.2021.11.013.

[22] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. https://doi.org/10.48550/ARXIV.2211.14730

[23] Leslie N. Smith and Nicholay Topin. 2017. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. (2017). https://doi.org/10.48550/ARXIV.1708.07120

[24] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Yanxi Li, Chao Xu, and Yunhe Wang. 2021. An Image Patch is a Wave: Phase-Aware Vision MLP. (2021). https://doi.org/10.48550/ARXIV.2111.12294

[25] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems* 34 (2021), 24261–24272.

[26] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. 2021. Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding. In *International Conference on Learning Representations*. https://openreview.net/forum?id=8qDwejCuCN

[27] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. 2022. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[28] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018). arXiv:1807.03748 http://arxiv.org/abs/1807.03748

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[30] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*.

[31] Ling Yang and Shenda Hong. 2022. Unsupervised Time-Series Representation Learning with Iterative Bilinear Temporal-Spectral Fusion. In *ICML*.

[32] Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. 2021. S^2-MLP: Spatial-Shift MLP Architecture for Vision. (2021). https://doi.org/10.48550/ARXIV.2106.07477

[33] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2022. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8980–8987.

[34] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2022. Are Transformers Effective for Time Series Forecasting? *arXiv preprint arXiv:2205.13504* (2022). https://arxiv.org/pdf/2205.13504.pdf

[35] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2022. Github Repo: Are Transformers Effective for Time Series Forecasting? *arXiv preprint arXiv:2205.13504* (2022). https://github.com/cure-lab/LTSF-Linear

[36] Tianping Zhang, Yizhuo Zhang, Wei Cao, Jiang Bian, Xiaohan Yi, Shun Zheng, and Jian Li. 2022. Less Is More: Fast Multivariate Time Series Forecasting with Light Sampling-oriented MLP Structures. https://doi.org/10.48550/ARXIV.2207.01186

[37] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, Vol. 35. 11106–11115.

[38] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning* (Baltimore, Maryland).

A APPENDIX

A.1 Datasets

We use 7 popular multivariate datasets provided in [22] for forecasting and representation learning. *Weather*[7] dataset collects 21 meteorological indicators such as humidity and air temperature. *Traffic*[8] dataset reports the road occupancy rates from different sensors on San Francisco freeways. *Electricity*[9] captures the hourly electricity consumption of 321 customers. *ETT*[10] (Electricity Transformer Temperature) datasets report sensor details from two electric Transformers in different resolutions (15 minutes and 1 hour). Thus, in total we have 4 ETT datasets: *ETTM1*, *ETTM2*, *ETTH1*, and *ETTH2*.

[7] https://www.bgc-jena.mpg.de/wetter/

[8] https://pems.dot.ca.gov/

[9] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[10] https://github.com/zhouhaoyi/ETDataset

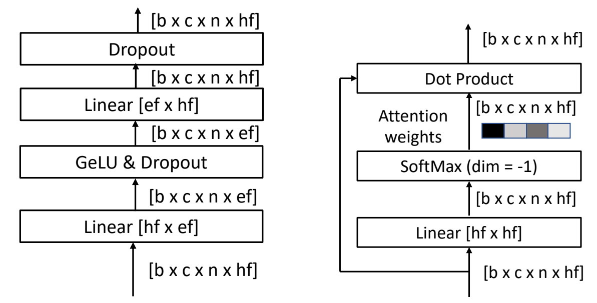
A.2 Supplementary Details

TSMixer follows [23] to determine the optimal learning rate during training for better convergence. Early stopping with patience 10 is applied during training. TSMixer library is built using PyTorch and multi-GPU node training is enabled via Pytorch DDP[11]. TSMixer can easily scale and cater to the needs of large-scale forecasting and can be deployed across multiple nodes in the Kubernetes clusters.

A.3 Supplementary Figures

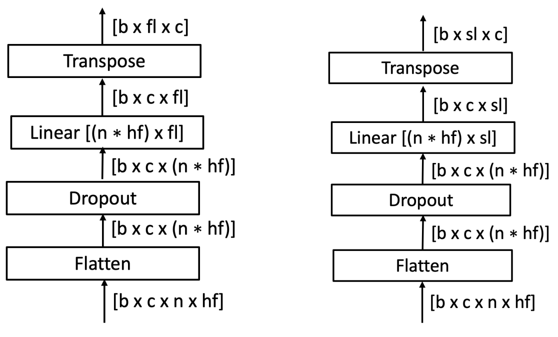
This section covers the supplementary figures that are referred in the paper for better understanding. Figure. 6(a) depicts the standard MLP block used in the mixer layer. Figure. 6(b) explains the gated attention block used in the mixer layer to downscale the noisy features. Figure. 7 highlights the architecture followed in the prediction and pretrain head. In the self-supervised pre-training workflow, pre-train head is attached to the backbone. In other workflows (such as supervised, or finetuning), the prediction head is attached.

Figure. 8 shows the correlation between patch time series and its associated embeddings in different datasets.



(a) MLP block in MLP-Mixer **(b) Gated attention in ours**

Figure 6: MLP and Gated Attention



(a) Prediction head **(b) Pretrain head**

Figure 7: Model heads for the two workflows.

Data	FL	**CI-TSMixer-Best**	LightTS	S4
ETTH1	336	**0.421**	0.466	1.407
	720	**0.444**	0.542	1.162
ETTH2	336	**0.357**	0.497	1.98
	720	**0.395**	0.739	2.65
Weather	336	**0.243**	0.527	0.531
	720	**0.316**	0.554	0.578
CI-TSMixer-Best % improvement			**33.2%**	**66.4%**

Table 8: LightTS and S4 MSE Benchmarking - Baseline results reported from [36] and [12]

Data	FL	CrossFormer	CI-TSMixer	**CI-TSMixer-Best**
ETTH1	336	0.44	0.437	**0.421**
	720	0.519	0.465	**0.444**
Electricity	336	0.323	0.165	**0.158**
	720	0.404	0.204	**0.186**
Traffic	336	0.53	0.388	**0.385**
	720	0.573	0.426	**0.424**
Weather	336	0.495	0.246	**0.243**
	720	0.526	0.323	**0.316**
% Improvement over CrossFormer			**31.35%**	**33.50%**

Table 9: CrossFormer MSE Benchmarking - Baseline results reported from [3]

A.4 Supplementary results

This section explains the supplementary results which are not reported in the main paper due to space constraints.

A.4.1 ***Benchmarking LightTS, S4, CrossFormer and TS2Vec.*** This section compares and contrasts TSMixer with LightTS [36], S4 [12], CrossFormer [3] and TS2Vec [33]. Considering the space constraints and the lower performance of these benchmarks as compared to our reported primary benchmarks (like PatchTST, DLinear), we mention these in the appendix instead of the main paper. Table. 8 and Table. 9 compare and contrast TSMixer with LightTS, S4 and CrossFormer in a supervised workflow. Table. 10 compares and contrasts TSMixer with TS2Vec in the self-supervised workflow. Since the baseline papers reported the results in a different forecast horizon (fl) space, we report their comparison in separate tables as per the commonly available forecast horizons.

A.4.2 ***Detailed Analysis.*** In this section, we provide more detailed benchmarking results as follows:

- Table. 11 shows the drill-down view of CI-TSMixer-Best on supervised multivariate forecasting.
- Table. 12 highlights the effect of CI, Gated Attention and Hierarchy Reconciliation over Vanilla TSMixer on all datasets.
- Table. 13 depicts the MSE analysis of various channel mixing techniques with different context lengths.
- Table. 14 highlights the MSE drill-down view of CI-TSMixer-Best for self-supervised multivariate forecasting.

[11] *https://pytorch.org/tutorials/beginner/dist_overview.html*

	FL	CI-TSMixer -Best	TS2Vec
ETTH1	24	**0.314**	0.599
	48	**0.343**	0.629
	168	**0.397**	0.755
	336	**0.424**	0.907
	720	**0.453**	1.048
Electricity	336	**0.16**	0.349
	720	**0.187**	0.375
	% Improvement		**50.7%**

Table 10: TS2Vec MSE Benchmarking - Baseline results reported from [33]

Models		PatchTST		DLinear		CI-TSMixer(G,H)		CI-TSMixer(G,H,CC)		CI-TSMixer-Best	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTH1	96	0.37	0.4	0.375	0.399	**0.368**	**0.398**	**0.368**	**0.398**	**0.368**	**0.398**
	192	0.413	0.429	0.405	**0.416**	**0.399**	0.418	0.4	0.418	**0.399**	0.418
	336	0.422	0.44	0.439	0.443	**0.421**	**0.436**	0.422	0.437	**0.421**	**0.436**
	720	0.447	0.468	0.472	0.49	**0.444**	**0.467**	0.45	**0.467**	**0.444**	**0.467**
ETTH2	96	**0.274**	**0.337**	0.289	0.353	0.276	**0.337**	0.276	0.339	0.276	**0.337**
	192	0.341	0.382	0.383	0.418	0.335	0.377	**0.33**	**0.374**	**0.33**	**0.374**
	336	**0.329**	**0.384**	0.448	0.465	0.369	0.406	0.357	0.401	0.357	0.401
	720	**0.379**	**0.422**	0.605	0.551	0.409	0.447	0.395	0.436	0.395	0.436
ETTM1	96	0.293	0.346	0.299	**0.343**	0.297	0.348	**0.291**	0.346	**0.291**	0.346
	192	**0.333**	0.37	0.335	**0.365**	**0.333**	0.369	0.334	0.369	**0.333**	0.369
	336	0.369	0.392	0.369	0.386	**0.365**	**0.385**	0.367	0.387	**0.365**	**0.385**
	720	**0.416**	0.42	0.425	0.421	**0.416**	**0.413**	0.421	0.415	**0.416**	**0.413**
ETTM2	96	0.166	0.256	0.167	0.26	**0.164**	**0.255**	0.165	**0.255**	**0.164**	**0.255**
	192	0.223	0.296	0.224	0.303	**0.219**	**0.293**	0.225	0.299	**0.219**	**0.293**
	336	0.274	**0.329**	0.281	0.342	**0.273**	0.33	**0.273**	**0.329**	**0.273**	**0.329**
	720	0.362	0.385	0.397	0.421	**0.358**	**0.38**	0.361	0.383	**0.358**	**0.38**
Electricity	96	**0.129**	**0.222**	0.14	0.237	**0.129**	0.224	**0.129**	0.225	**0.129**	0.224
	192	0.147	**0.24**	0.153	0.249	0.148	0.242	**0.146**	0.242	**0.146**	0.242
	336	0.163	0.259	0.169	0.267	0.164	0.259	**0.158**	**0.256**	**0.158**	**0.256**
	720	0.197	0.29	0.203	0.301	0.201	0.292	**0.186**	**0.282**	**0.186**	**0.282**
Traffic	96	0.36	0.249	0.41	0.282	**0.356**	**0.248**	0.369	0.264	**0.356**	**0.248**
	192	0.379	**0.256**	0.423	0.287	**0.377**	0.257	0.393	0.278	**0.377**	0.257
	336	0.392	0.264	0.436	0.296	**0.385**	**0.262**	0.406	0.285	**0.385**	**0.262**
	720	0.432	0.286	0.466	0.315	**0.424**	**0.283**	0.445	0.304	**0.424**	**0.283**
Weather	96	0.149	0.198	0.176	0.237	0.148	0.198	**0.146**	**0.197**	**0.146**	**0.197**
	192	0.194	0.241	0.22	0.282	0.193	**0.24**	**0.191**	**0.24**	**0.191**	**0.24**
	336	0.245	0.282	0.265	0.319	**0.243**	**0.279**	0.244	0.28	**0.243**	**0.279**
	720	**0.314**	0.334	0.323	0.362	0.317	**0.333**	0.316	**0.333**	0.316	**0.333**

Table 11: Drill-down view of CI-TSMixer-Best for supervised multivariate forecasting. CI-TSMixer-Best is defined as the best of CI-TSMixer(G,H) and CI-TSMixer(G,H,CC), wherein. based on the considered dataset - either CI-TSMixer(G,H) or CI-TSMixer(G,H,CC) outperforms the existing benchmarks. CI-TSMixer(G,H,CC) uses context length = 1 for this experiment. PatchTST results are reported from [22] and DLinear from [34]

Models		V-TSMixer		CI-TSMixer		CI-TSMixer(G)		CI-TSMixer(H)		CI-TSMixer(G,H)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTH1	96	0.449	0.462	0.375	0.401	0.375	0.4	0.377	0.405	**0.368**	**0.398**
	192	0.485	0.484	0.411	0.426	0.408	0.422	0.41	0.428	**0.399**	**0.418**
	336	0.504	0.497	0.437	0.445	0.433	0.439	0.431	0.445	**0.421**	**0.436**
	720	0.573	0.534	0.465	0.478	0.454	0.47	0.457	0.477	**0.444**	**0.467**
ETTH2	96	0.369	0.412	0.284	0.342	**0.276**	0.339	0.283	0.341	**0.276**	**0.337**
	192	0.391	0.428	0.353	0.384	0.338	0.379	0.344	0.38	**0.335**	**0.377**
	336	0.403	0.44	0.365	**0.403**	**0.362**	0.404	0.376	0.408	0.369	0.406
	720	0.475	0.481	0.406	0.442	0.415	0.451	**0.396**	**0.434**	0.409	0.447
ETTM1	96	0.328	0.372	0.305	0.354	**0.296**	0.349	0.301	0.351	0.297	**0.348**
	192	0.372	0.397	0.336	0.374	0.334	0.37	0.338	0.372	**0.333**	**0.369**
	336	0.405	0.416	0.375	0.398	0.366	0.387	0.376	0.394	**0.365**	**0.385**
	720	0.457	0.445	0.425	0.417	0.419	0.415	0.422	0.417	**0.416**	**0.413**
ETTM2	96	0.195	0.282	0.172	0.263	0.175	0.264	0.168	0.259	**0.164**	**0.255**
	192	0.26	0.326	0.221	0.294	0.221	0.294	0.224	0.296	**0.219**	**0.293**
	336	0.341	0.373	**0.273**	**0.328**	0.277	0.333	0.276	0.331	**0.273**	0.33
	720	0.437	0.432	**0.357**	0.384	0.365	0.385	0.36	0.383	0.358	**0.38**
Electricity	96	0.212	0.323	0.13	0.224	**0.129**	**0.223**	0.13	0.226	**0.129**	0.224
	192	0.21	0.319	**0.148**	0.242	**0.148**	**0.241**	**0.148**	0.243	**0.148**	0.242
	336	0.224	0.332	0.165	**0.259**	0.165	**0.259**	0.165	0.26	**0.164**	**0.259**
	720	0.251	0.352	0.204	0.293	**0.201**	**0.291**	0.204	0.295	**0.201**	0.292
Traffic	96	0.618	0.386	0.358	0.251	**0.355**	**0.246**	0.357	0.251	0.356	0.248
	192	0.624	0.385	0.379	0.26	**0.377**	**0.257**	0.378	0.26	**0.377**	**0.257**
	336	0.63	0.38	0.388	0.265	0.387	0.264	0.387	0.265	**0.385**	**0.262**
	720	0.66	0.389	0.426	0.286	0.427	0.285	0.425	0.286	**0.424**	**0.283**
Weather	96	0.159	0.216	0.15	0.202	0.149	0.199	0.151	0.202	**0.148**	**0.198**
	192	0.207	0.257	0.195	0.244	0.195	0.242	0.195	0.243	**0.193**	**0.24**
	336	0.256	0.294	0.246	0.284	0.246	0.281	0.246	0.283	**0.243**	**0.279**
	720	0.33	0.344	0.323	0.338	**0.317**	**0.333**	0.321	0.337	**0.317**	**0.333**
% MSE improvement over V-TSMixer				**18%**		**18.5%**		**18.1%**		**19.3%**	

Table 12: Effect of CI, Gated Attention and Hierarchy Reconciliation over Vanilla TSMixer on all datasets. CI-TSMixer outperforms V-TSMixer by 18% and by adding gated attention (G) and Hierarchy Reconciliation head (H), the accuracy further improves by 1.3%, leading to a total of 19.3% improvement. It is important to note that, we observe stable improvements when (G) and (H) are used together instead of just (G) or (H).

Models		V-TSMixer	CI-TSMixer	IC-TSMixer	CI-TSMixer(G,CC)					
Channel Context Length					(1)	(2)	(3)	(4)	(5)	(Best)
ETTH1	96	0.449	0.375	0.379	**0.373**	**0.373**	0.377	**0.373**	0.374	**0.373**
	192	0.485	0.411	0.416	0.416	**0.407**	**0.407**	0.409	0.408	**0.407**
	336	0.504	0.437	0.437	0.436	0.435	0.437	0.436	**0.43**	**0.43**
	720	0.573	0.465	0.471	0.482	0.493	0.472	0.465	**0.454**	**0.454**
ETTH2	96	0.369	0.284	0.291	0.27	0.274	0.277	0.275	**0.269**	**0.269**
	192	0.391	0.353	0.345	0.336	0.347	0.333	**0.33**	0.339	**0.33**
	336	0.403	0.365	0.361	0.364	0.363	0.36	**0.359**	0.363	**0.359**
	720	0.475	0.406	0.419	0.4	0.398	0.41	**0.393**	**0.393**	**0.393**
ETTM1	96	0.328	0.305	0.307	**0.3**	0.302	0.304	0.309	0.305	**0.3**
	192	0.372	**0.336**	0.34	0.345	0.345	0.341	0.345	0.338	0.338
	336	0.405	0.375	**0.372**	0.379	0.375	0.375	0.377	0.383	0.375
	720	0.457	0.425	0.428	**0.422**	0.426	0.429	0.426	0.44	**0.422**
ETTM2	96	0.195	0.172	0.169	0.166	0.173	**0.165**	0.166	0.168	**0.165**
	192	0.26	0.221	0.221	0.225	**0.218**	0.22	0.22	0.221	**0.218**
	336	0.341	0.273	0.278	**0.272**	0.284	0.277	0.285	0.276	**0.272**
	720	0.437	**0.357**	0.367	0.359	0.358	0.385	0.362	0.362	0.358
Electricity	96	0.212	**0.13**	0.163	0.139	0.133	0.133	0.133	0.134	0.133
	192	0.21	**0.148**	0.181	0.15	0.15	0.15	0.151	0.15	0.15
	336	0.224	0.165	0.196	0.162	**0.159**	0.161	0.161	0.163	**0.159**
	720	0.251	0.204	0.224	0.197	0.197	**0.194**	**0.194**	**0.194**	**0.194**
Traffic	96	0.618	**0.358**	0.468	0.382	0.384	0.384	0.374	0.387	0.374
	192	0.624	**0.379**	0.483	0.401	0.403	0.401	0.404	0.4	0.4
	336	0.63	**0.388**	0.493	0.411	0.412	0.413	0.413	0.412	0.411
	720	0.66	**0.426**	0.524	0.448	0.457	0.445	0.454	0.451	0.445
Weather	96	0.159	0.15	0.15	0.149	**0.146**	0.148	0.147	0.15	**0.146**
	192	0.207	0.195	0.196	0.196	0.196	**0.194**	0.197	0.197	**0.194**
	336	0.256	**0.246**	0.248	**0.246**	0.251	0.249	0.249	**0.246**	**0.246**
	720	0.33	0.323	0.338	**0.317**	0.318	0.319	0.321	0.319	**0.317**
% improvement over V-TSMixer			**18%**	**13.2%**						**19%**

Table 13: Detailed MSE Analysis of various Channel Mixing techniques with different context lengths. Context length is varied from (1) to (5) and the minimum is selected as (Best) in this table. From the complete data analysis, we observe that CI-TSMixer outperforms V-TSMixer by 18%, and by adding a cross-channel reconciliation head (CC), the accuracy further improves by 1% leading to a total improvement of 19%. In contrast - IC-TSMixer outperforms V-TSMixer but not CI-TSMixer .

	FL	CI-TSMixer -Best	CI-TSMixer (G,H))	CI-TSMixer (G,H,CC))	PatchTST	BTSF
ETTH1	24	**0.314**	0.319	**0.314**	0.322	0.541
	48	**0.343**	0.344	**0.343**	0.354	0.613
	168	**0.397**	**0.397**	0.402	0.419	0.64
	336	**0.424**	**0.424**	0.43	0.445	0.864
	720	**0.453**	**0.453**	0.457	0.487	0.993
Weather	24	0.088	0.09	0.088	**0.087**	0.324
	48	0.114	0.114	0.141	**0.113**	0.366
	168	**0.177**	**0.177**	0.178	0.178	0.543
	336	**0.241**	**0.241**	0.244	0.244	0.568
	720	**0.319**	**0.319**	0.322	0.321	0.601

Table 14: MSE Drill-down view of CI-TSMixer-Best for self-supervised multivariate forecasting. Min of CI-TSMixer(G,H) and CI-TSMixer(G,H,CC) is depicted as CI-TSMixer-Best, wherein. based on the considered dataset - either CI-TSMixer(G,H) or CI-TSMixer(G,H,CC) outperforms the existing benchmarks.

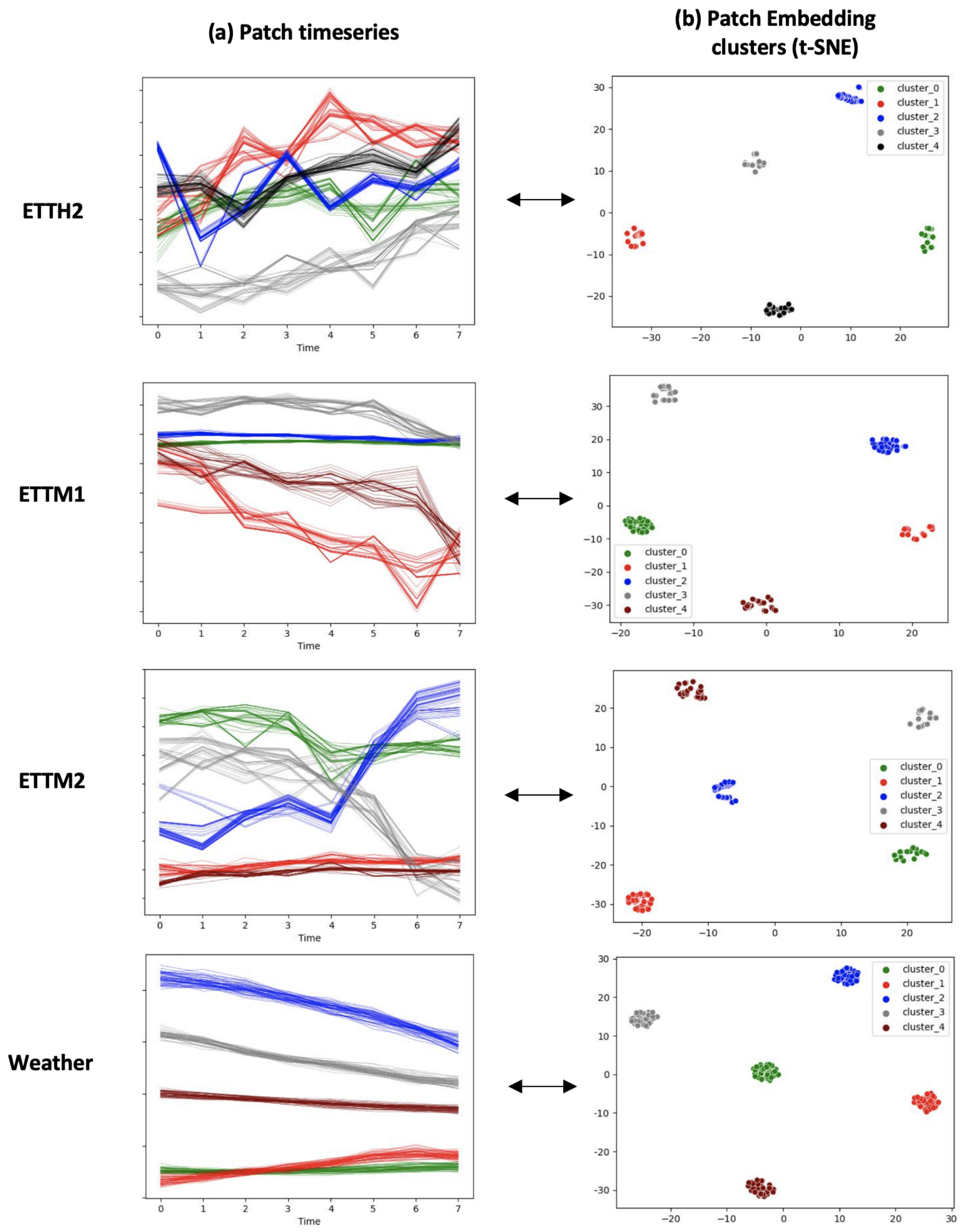


Figure 8: Correlation between Patch time-series and its associated embeddings in multiple datasets. Nearby patch representations highly correlate to the patch time series of similar shapes and patterns, thereby learning meaningful patch representations.