A Multi-Task Learning Approach to Linear Multivariate Forecasting

Liran Nochumsohn **Hedi Zisling** **Omri Azencot**
Ben-Gurion University of the Negev

Abstract

Accurate forecasting of multivariate time series data is important in many engineering and scientific applications. Recent state-of-the-art works ignore the inter-relations between variates, using their model on each variate independently. This raises several research questions related to proper modeling of multivariate data. In this work, we propose to view multivariate forecasting as a *multi-task learning* problem, facilitating the analysis of forecasting by considering the angle between task gradients and their balance. To do so, we analyze linear models to characterize the behavior of tasks. Our analysis suggests that tasks can be defined by grouping similar variates together, which we achieve via a simple clustering that depends on correlation-based similarities. Moreover, to balance tasks, we scale gradients with respect to their prediction error. Then, each task is solved with a linear model within our *MTLinear* framework. We evaluate our approach on challenging benchmarks in comparison to strong baselines, and we show it obtains on-par or better results on multivariate forecasting problems. Code is available at https://github.com/azencot-group/MTLinear.

1 INTRODUCTION

Time series forecasting (TSF) with deep learning leverages neural networks to model and predict sequential data over time, enabling accurate predictions and insights into future trends. Its importance lies in its ability to handle complex temporal dependencies and patterns, making it invaluable for tasks such as financial forecasting, resource planning, and demand prediction in various industries. While most existing TSF approaches are based on N-BEATS and the Transformer [29, 39], a recent work finds simple linear layers to be highly effective [44, 19]. However, linear models are naturally limited, and thus, current efforts focus on developing nonlinear approaches where state-of-the-art (SOTA) techniques incorporate the linear module as the final decoder layer [27, 49].

Generally, TSF frameworks are designed to accept univariate and multivariate temporal data. In the multivariate case, time series data has multiple dimensions per sequence sample, whereas univariate data is one dimensional. Remarkably, while TSF is assumed to benefit from the inter-relations underlying multivariate information [9, 34], recent methods opt to handle each variate independently [44, 27]. Nevertheless, dominant variate signals may disproportionally affect forecast performance, especially in limited linear models. Several research questions arise from the latter straightforward observation: how to model different variates? how to treat similar vs. dissimilar variates? how to balance the contribution of variates in the context of forecasting?

Towards addressing the above questions, we interpret time series forecasting with deep learning through the lens of multi-task learning (MTL) [46]. MTL trains a single model to perform multiple related tasks simultaneously, leveraging shared representations to improve performance across tasks, enable knowledge transfer, and facilitate efficient learning. Our view of TSF as MTL is based on the assumption that *similar variates should be modeled similarly, and dissimilar variates encode different forecasting tasks.* When considered as separate tasks within MTL formalism, we can harness tools, observations, and the general advances in MTL to better solve time series forecasting problems.

Particularly, a recent work [42] formulates some of the challenges underlying multi-task learning with respect to the *tragic triad*. The authors advocate that conflicting gradients, varying gradient magnitudes, and highly curved loss manifolds hinder MTL methodologies. Further, identifying which tasks can learn together is crucial to effective multi-task learning [7]. For instance, assigning conflicting tasks to a separate set of

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

network weights was suggested in [11]. In our study, we explore how to group variates together, and how to weigh different variate groups during training. To our knowledge, deep TSF from a multi-task learning perspective has received only limited attention.

Given the success of linear models in TSF, we are motivated to study the use of one linear model per similar variates group, and to balance dissimilar variate groups based on their dominance. Specifically, we analyze the gradients of linear modules, and we observe the factors affecting the norm of the gradients and their direction. Based on our derivative analysis, we propose to group variates by their Pearson Correlation Coefficient, encoding linear relationships. Then, we construct a multi-head linear model (MTLinear), where each head learns from a separate variate group. Finally, we scale the gradients using the variate's characteristics. Our contributions can be summarized as follows:

- We suggest to interpret time series forecasting as multi-task learning, where similar variates are grouped together, and each group forms a separate task. Variate grouping and per group balancing is inspired by our gradients analysis of linear TSF.
- We propose a simple, efficient and effective multi-head linear network (**MTLinear**) for solving multivariate time series forecasting tasks.
- We extensively evaluate our method on challenging TSF benchmarks, and compare our results to state-of-the-art (SOTA) models, showing that MTLinear is a strong standalone technique and it can be considered as a building block module for TSF.

2 BACKGROUND

Multi-task learning (MTL) aims to optimize a set of weights θ that simultaneously minimize k different tasks, each corresponding to a different loss function $L_i(\theta)$. A typical single-objective function that combines all tasks takes the following form:

$$\theta^* = \arg\min_\theta L(\theta) := \arg\min_\theta \frac{1}{k}\sum_i^k L_i(\theta) \ , \quad (1)$$

where all tasks are averaged together to form a single loss $L(\theta)$ that can be directly minimized via gradient descent. In this setting, each loss $L_i(\theta)$ is associated with a gradient $g_i := \nabla L_i(\theta)$. The total gradient reads

$$\nabla L(\theta) = \frac{1}{k}\sum_i^k g_i = \frac{1}{k}\sum_i^k \nabla L_i(\theta) \ . \quad (2)$$

Following [42], we detail fundamental challenges inherent to multi-task learning, termed as the tragic triad. First, *conflicting gradients* arise when the inner product between a pair of task gradients g_i, g_j is negative, i.e., $g_i^T g_j < 0$. Geometrically, it means that their angle is greater than $\pi/2$, and therefore, optimizing θ is carried by non-aligned directions. Second, when a set of gradients have a large *gradient magnitude difference*, larger gradients may shadow weaker gradients. This issue becomes especially problematic when gradients are also conflicting, leading to long and unstable training [30]. Finally, *highly curved loss landscapes* negatively affect the optimization of neural networks [18, 15, 16].

Multivariate TSF deals with predicting future values of multiple inter-related variates over the time domain. Particularly, we are given in the multivariate case k inter-related components $x_1, x_2, \ldots, x_k \in \mathbb{R}^l$ of length l, whereas the time series is univariate if $k = 1$. In TSF we predict the corresponding future values, denoted by $y_1, y_2, \ldots, y_k \in \mathbb{R}$. Formally, TSF solves

$$\theta^* = \arg\min_\theta F(\theta) := \arg\min_\theta \frac{1}{k}\sum_i^k F_i(\theta) \ , \quad (3)$$

where the loss for forecasting tasks is the mean squared error (MSE), i.e., $F_i(\theta) := [M(x_i, \theta) - y_i]^2 \in \mathbb{R}$ with $M(\cdot, \cdot)$ being a parametric model such as a deep neural network or a linear module.

Linear forecasting with Linear, NLinear, and DLinear in [44] includes a single linear layer. While extremely simple, these models have achieved remarkable results in comparison to many Transformer-based approaches. Yet, they fall short with respect to recent SOTA techniques [27, 49]. Notably, the latter works often include in their architectures a linear decoder, similar in structure and role to the linear layers proposed in [44]. Formally, a standard linear layer is defined as,

$$\tilde{Y}^T = X^T\Theta + b \ , \quad (4)$$

where $X = [x_1, x_2, \ldots, x_k] \in \mathbb{R}^{l\times k}$ is the input look-back, $\tilde{Y} = [y_1, y_2, \ldots, y_k] \in \mathbb{R}^{h\times k}$ is the forecast horizon output, and $\Theta = [\theta_1, \theta_2, \ldots, \theta_h] \in \mathbb{R}^{l\times h}$ and $b \in \mathbb{R}^h$ are the model weights and bias, respectively. We denote by $Y \in \mathbb{R}^{h\times k}$ the ground-truth horizon values.

3 RELATED WORK

Multivariate time series forecasting. Significant efforts have been dedicated to modeling and predicting sequential information using deep neural networks [8]. In particular, recurrent neural networks (RNNs) such as LSTM and GRU with their gating mechanisms obtained groundbreaking results on various vision and language tasks [13, 5] across generative [25, 24, 32] and augmentation [17, 28] tasks, among many others.

Alas, not until recently, pure deep models were believed to be unable to outperform non-deep or hybrid tools [29]. Nevertheless, within a span of two years, pure deep methods based on RNNs [33], feedforward networks [29], and the Transformer [47] have appeared, demonstrating competitive results and setting a new SOTA bar for long-term TSF.

Following these breakthroughs, numerous works have emerged, most of them are based on the Transformer architecture [40, 48, 45]. Recently, a surprising work [44] has shown remarkable TSF results with a simple single-layer linear model, competing and even surpassing the best models for that time. Notably, the linear model applied weights along the time axis, in contrast to the more conventional practice of applying weights along the variate axis. Subsequently, new SOTA techniques [27, 41] and a recent foundation model for time series [49] have integrated a similar linear module as their final decoder to attain better forecasts.

Multi-task learning. Improving learning on multiple tasks follows three different categories: *gradient manipulation*, *task grouping*, and *architecture design*. Manipulating gradients is done by imposing weights [10], shift gradient directions [42], and add an optimization step to balance gradients or resolve conflicts [3, 35, 20]. In task grouping, previous works attempted to cluster different tasks based on similarity measures, thereafter assigning them to separate models [43, 37, 7, 36]. Lastly, multiple forms of architecture design have been introduced to support MTL including hard-parameter sharing [38], soft-parameter sharing [23, 14], and mixing solutions [11]. For time series data, a multi-gate mixture of experts for classification was suggested in [22], whereas a soft-parameter sharing RNN-based model was proposed in [4]. However, most of the multi-task techniques mentioned above focus mostly on vision problems. Consequently, little attention has been given to analyzing and mitigating the multi-task learning challenges for time series problems, particularly TSF.

4 METHOD

We analyze linear forecasting (Sec. 4.1), motivating our variate grouping (Sec. 4.2), gradient scaling 4.3, and yielding an efficient and effective TSF model (Sec. 4.4).

The following preliminary example motivates our perspective of TSF as MTL. We compare gradient conflicts vs. variate correlation by training a vanilla TSF model, and counting the number of conflicts between gradients after every epoch. We used the DLinear [44] and PatchTST [27] TSF models and ETTm2 and Weather datasets. The results are shown in Fig. 1, where every line plot shows the conflicts number for a pair of variates. The correlation (corr) per pair is detailed in the legend, where $|\text{corr}| \approx 1$ and $|\text{corr}| \approx 0$ denote strong and weak linear correlations, respectively. Clearly, variates with a strong linear relationship (negative or positive) have fewer conflicts and vice versa, consistent across both DLinear and PatchTST architectures.

4.1 Linear Analysis

To motivate our approach, we present below a gradient analysis of the linear model in Eq. (4). We begin by noting that since Θ is applied in the temporal domain, every forecast horizon $j = 1, \ldots, h$ is obtained with a separate set of weights, i.e., $\theta_j \in \mathbb{R}^l$ and $b_j \in \mathbb{R}$. Indeed, we have that $X^T\Theta = [X^T\theta_1, X^T\theta_2, \ldots, X^T\theta_h] = [\tilde{Y}_1, \tilde{Y}_2, \ldots, \tilde{Y}_h]$, where $\tilde{Y}_j$ represents the j-th row of $\tilde{Y}^T$, and we incorporated the bias b_j in θ_j, for all j. Thus, the MSE loss corresponding to Eq. (4) is given by

$$F(\Theta) = |X^T\Theta - Y^T|_F^2 := \sum_{j=1}^{h}\sum_{i=1}^{k} \frac{(x_i^T\theta_j - y_{j,i})^2}{kh} \ , \quad (5)$$

where $Y \in \mathbb{R}^{h\times k}$ are the true values, and we split every $X^T\theta_j$ to individual coordinates via $x_i^T\theta_j$. The objective in Eq. (5) is quadratic, and thus, its gradient is linear in Θ. Given the above discussion, F can be viewed either as a function $F(\Theta) : \mathbb{R}^{(l+1)\times h} \to \mathbb{R}$ or as a sum of functions $F(\theta_j) : \mathbb{R}^{l+1} \to \mathbb{R}$ for $j = 1, \ldots, h$. The gradient for the latter form reads

$$\nabla_{\theta_j} F(\theta_j) = \frac{1}{k}\sum_{i=1}^{k} 2x_i(x_i^T\theta_j - y_{j,i}) \ . \quad (6)$$

We provide the full derivation in App. A. The expression in Eq. (6) highlights that multivariate TSF and MTL are closely related, in the sense that the k variates represent k different tasks sharing the same set of weights θ_j. Thus, multivariate forecasting may potentially face similar challenges as multi-task learning.

The simplicity of the linear model and its gradients allows to characterize the direction and scale of gradients. Based on Eq. (6), we arrive at the following two straightforward observations: 1) The *direction* of the gradient is governed by $x_i \in \mathbb{R}^{l+1}$; and 2) The gradient's *scale* is affected by $2(x_i^T\theta_j - y_{j,i}) \in \mathbb{R}$. Using these observations, we also consider the relation of different tasks, i.e., the angle and balance between the gradients associated with different variates x_a and x_b.

To this end, the first observation reveals that each task is updated simply along the direction of its corresponding variate x_i. Thus, the *angle* between gradients is equivalent to the angle α between variates x_a and x_b, which is given by their cosine similarity, $\cos_{x_a,x_b}(\alpha) = x_a^T x_b/(|x_a| \cdot |x_b|)$. The cosine similarity

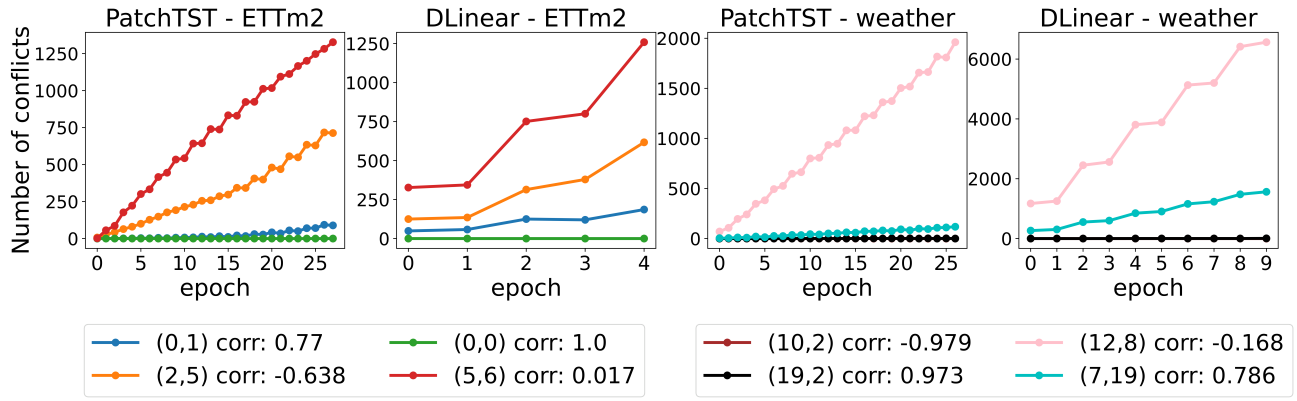

Figure 1: The total number of conflicts as a function of epochs. Colored lines represent variate pairs. Pairs with a higher absolute correlation (shown in legend) tend to have fewer conflicts during training.

is closely related to the *Pearson Correlation Coefficient* (PCC) [6]. Indeed, PCC and cosine similarity coincide for centered data. Thus, we interchangeably use these terms below. In our work, we consider high absolute similarity, i.e., $|\cos_{x_a,x_b}(\alpha)| \approx 1$, to correspond to a strong relationship between tasks, whereas low PCC, $|\cos_{x_a,x_b}(\alpha)| \approx 0$, means weakly-related variates.

The second observation details what impacts the gradient norm. Suppose some variates are inherently more difficult to forecast than others due to rapid distribution shifts with the lack of clear seasonality or trend, thus achieving a larger error $e_{i,j} = |x_i^T\theta_j - y_{j,i}|$. Clearly, in such cases the gradient will dominate the optimization process due to its large magnitude. In other words, gradients associated with variates with higher forecast uncertainty have generally larger gradients, see Fig. 5. Based on the above analysis and corresponding observations and towards addressing multi-task learning issues related to the tragic triad, we propose in what follows a simple, efficient and effective method that takes into account the correlation of variates and the error $e_{i,j}$.

4.2 Variate grouping

Based on Sec. 3, one common approach to deal with some of the challenges underlying MTL is to assign non-conflicting tasks to separate sets of weights [7, 11, 36]. A direct implementation of this idea in TSF will result in an independent model per variate, which may be highly demanding computationally for data with many variates (App. C.4). An optional middle-ground toward an efficient framework is to assign separate weights for *a-priori clustered variates*. Namely, we identify similar variates, clustering them into groups with separate network weights. Importantly, forecasting is different from multi-task learning problems in, e.g., vision, where task relationships can not be necessarily extracted directly from the input. In comparison, the objective in TSF is to accurately predict the future horizon, which shares fundamental features with the input lookback, such as trend, seasonality, periodicity, and data distribution. Moreover, our analysis in Sec. 4.1 shows that tasks' alignment can be deduced in practice by estimating the correlation between different variate vectors.

Our a-priori clustering and weights assignment method follows three simple steps: 1) Compute the absolute-value correlation matrix for variates, defined via $R_X = (|\cos_{x_a,x_b}(\alpha)|)$ for $a, b = 1, \ldots, k$. 2) Perform agglomerative clustering on R_X, based on a threshold $\bar{\alpha}$, encoding the maximum angle between two variates. Note that variates with $|\alpha| < \bar{\alpha}$ are grouped together. Sec. 5.2 justifies empirically grouping variates with a strong negative correlation, as they share a similar optimization trajectory. 3) Finally, for each variates cluster, assign a separate linear-based neural network (Sec. 4.4).

4.3 Gradient manipulation

Even though dissimilar variates are assigned to different sets of weights, the risk of having a subset of variates dominating the optimization process within each group still prevails. Thus, we introduce a *gradient magnitude penalty* $w^a_{i,j} \in \mathbb{R}^+$ that incorporates the error $e_{i,j}$ of its associated gradient. The penalty $w^a_{i,j}$ multiplies the loss for every admissible i, j, and the new loss is formally given by $F_W(\Theta) = |X^T\Theta - Y^T|^2_{W^a}$, where

$$|X^T\Theta - Y^T|^2_{W^a} = \sum_j^h \sum_i^k w^a_{i,j}(\mathbf{x}_i^T\theta_j - y_{i,j})^2 \ , \quad (7)$$

where $|A|^2_{W^a} = \text{trace}[A^T \odot \sqrt{W^a}(\sqrt{W^a})^T \odot A]$, $W^a = (w^a_{i,j}) \in \mathbb{R}^{k\times h}$, and the weights $w^a_{i,j}$ are defined as follows,

$$w^a_{i,j} = \frac{1}{(K_j \cdot H_i)^a}, \ K_j = \sum_{i=1}^k \frac{e_{i,j}}{k}, \ H_i = \sum_{j=1}^h \frac{e_{i,j}}{h}. \quad (8)$$

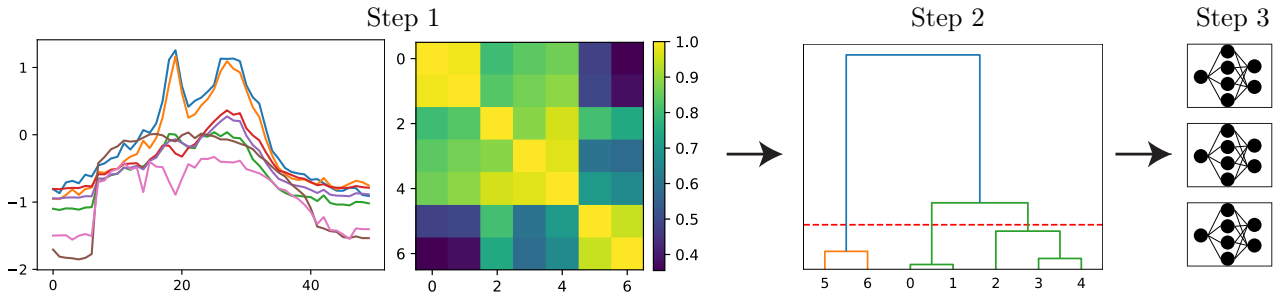

Figure 2: Our pipeline consists of three steps: estimating variate correlations, variate clustering, and assigning a linear module per group. The resulting framework, **MTLinear** solves multivariate TSF effectively.

Table 1: Multivariate forecasting results of MTLinear (ours) compared to other strong baselines. A **bold** and underlined notation represent the best and second-best scores, respectively.

Dataset	MTDLinear		MTNLinear		iTransformer		PatchTST		Crossformer		DLinear		FEDformer		Autoformer	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	0.399	0.402	0.403	0.402	0.407	0.410	**0.387**	**0.400**	0.513	0.495	0.403	0.407	0.448	0.452	0.588	0.517
ETTm2	0.284	0.334	**0.279**	**0.320**	0.288	0.332	0.281	0.326	0.757	0.610	0.350	0.401	0.304	0.349	0.327	0.371
ETTh1	0.456	0.441	0.444	**0.429**	0.454	0.448	0.469	0.454	0.529	0.522	0.456	0.452	**0.440**	0.460	0.496	0.487
ETTh2	0.453	0.447	**0.373**	**0.397**	0.383	0.406	0.387	0.407	0.942	0.684	0.559	0.515	0.436	0.449	0.450	0.459
ECL	0.198	0.286	0.204	0.283	**0.178**	**0.270**	0.205	0.290	0.244	0.334	0.212	0.300	0.214	0.327	0.227	0.338
Exchange	**0.290**	**0.377**	0.410	0.422	0.360	0.403	0.366	0.404	0.940	0.707	0.354	0.414	0.518	0.429	0.613	0.539
Traffic	0.621	0.380	0.624	0.372	**0.428**	**0.282**	0.481	0.304	0.550	0.304	0.624	0.383	0.609	0.376	0.628	0.379
Weather	**0.238**	0.295	0.249	**0.276**	0.258	0.278	0.258	0.280	0.258	0.315	0.265	0.317	0.309	0.360	0.338	0.382
ILI	2.234	0.995	**1.964**	**0.902**	2.738	1.098	2.421	1.011	3.386	1.236	2.616	1.090	2.846	1.144	3.006	1.161
Average	0.575	0.440	**0.550**	**0.423**	0.610	0.436	0.584	0.431	0.902	0.579	0.649	0.475	0.680	0.483	0.741	0.515
1^{st} Count	2	1	**3**	**5**	2	2	1	1	0	0	0	0	1	0	0	0

Essentially, $w^a_{i,j}$ addresses dominant scales along the horizon axis and the variates axis. Specifically, K_j is the mean error of different variates for the same horizon j, whereas H_i is the mean error of different horizons for a given variate i. Thus, $w^a_{i,j}$ balances both means when they attain high magnitudes. The parameter a controls the intensity of our penalty. During training, we treat $w^a_{i,j}$ as a constant scalar (i.e., a computational graph leaf), thus Eq. (7) avoids additional gradient computations. Consequently, the computational complexity of our gradient manipulation is $\mathcal{O}(1)$, unlike other methods [3, 42, 20] whose complexity is $\mathcal{O}(k)$. This procedure is applied individually to each group's linear model.

4.4 Multi-task Linear Model

The individual linear models mentioned in Sec. 4.2 are added to a training framework we call **MTLinear**. MTLinear is a single multi-head linear layer with c heads, each corresponding to a group of variates. Importantly, the separate heads can be trained in parallel, reducing the overall computational footprint of our approach. Each head of MTLinear is based on the linear models proposed in [44]. Specifically, we focus on the DLinear and NLinear baselines. DLinear decomposes the time series to a trend component and a remainder component, where each component is handled by a separate set of weights. The trend is extracted with a standard average pooling kernel. NLinear introduces a "normalization" pre-processing mechanism where the last values of the lookback are subtracted from the series before the forward pass, and are added back in when computation finishes. Our overall pipeline for time series forecasting is illustrated in Fig. 2.

5 EXPERIMENTS

Details regarding the datasets, models, experimental setup and implementation notes are provided in App. D. We tested our approach based on DLinear and NLinear vs. SOTA nonlinear Transformer models. The results in Tab. 1 show the MSE and MAE scores. Each score represents the average of four forecast horizons $h \in \{24, 36, 48, 60\}$ and 36 input length for ILI, as well as $h \in \{96, 192, 336, 720\}$ and 96 input length for the remaining. We use a similar format also in Tabs. 2, 3, and 4. Our results indicate that MTLinear has an overall superior performance with a best global MSE and MAE corresponding to **0.550** and **0.423** with MTNLinear, and second best MSE **0.575** with MTDLinear. In general, we outperform the transformer models, obtaining 11 top scores whereas iTransformer and PatchTST account for only 6. The full results are presented in Tab. 7, along with an extension to the 336 lookback length where we also compare to the original

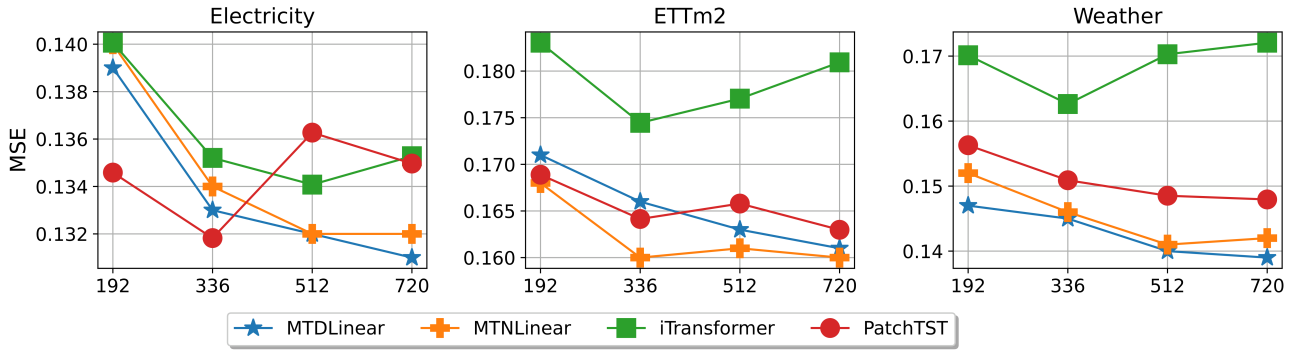

Figure 3: MSE results for different lookback lengths with a forecast horizon of 96.

Table 2: Average MSE scores ablating the components of MTLinear. See text.

	MTDLinear			
Dataset	Ours	Penalty	Grouping	Baseline
ETTm1	**0.399**	0.402	**0.399**	0.406
ETTm2	**0.284**	**0.284**	0.323	0.323
ETTh1	**0.456**	0.464	**0.456**	0.471
ETTh2	**0.453**	**0.453**	0.503	0.498
ECL	**0.198**	0.209	0.199	0.209
Exchange	0.289	**0.278**	0.312	0.309
Traffic	**0.621**	0.626	0.622	0.625
Weather	**0.238**	0.264	0.239	0.267
ILI	**2.234**	**2.234**	2.769	2.728

	MTNLinear			
Dataset	Ours	Penalty	Grouping	Baseline
ETTm1	**0.403**	0.407	0.404	0.410
ETTm2	**0.280**	0.285	0.281	0.286
ETTh1	0.443	**0.442**	0.444	0.446
ETTh2	**0.373**	0.374	0.374	0.374
ECL	**0.204**	0.215	**0.204**	0.214
Exchange	0.410	**0.366**	0.411	0.378
Traffic	**0.624**	0.625	**0.624**	**0.624**
Weather	**0.249**	0.272	**0.249**	0.273
ILI	**1.965**	**1.965**	2.254	2.213

PatchTST and GPT4TS results reported in Tab. 8.

5.1 Ablation of MTLinear Components

The proposed MTLinear method incorporates variate clustering and a gradient penalty, which, although separate components, work synergistically to address the tragic triad issues. Specifically, clustering helps manage gradient conflicts, while scaling addresses the varying gradient magnitudes. In the following experiment, we evaluate the effect of each component individually and in combination, with results presented in Tab. 2. Our ablation study reveals that in some datasets, using only one component may provide minimal benefit or even degrade performance. However, the combined approach consistently outperforms, with both components together yielding superior results in most cases.

5.2 Longer Lookbacks and Grouping Criteria

Longer lookbacks. Constraining a forecast model to a fixed lookback, such as 96, reduces its robustness and adaptability to different problems with varying lookback lengths. Here, we evaluate MTLinear's performance across different lookbacks—192, 336, 512, and 720—and compare it with iTransformer and PatchTST. As shown in Fig. 3, MTLinear variants consistently outperform iTransformer across all cases, with particularly strong results for the 720 lookback. While PatchTST proves competitive, MTLinear surpasses its performance across all lookbacks and datasets, except for Electricity at 192 and 336 lookbacks. Overall, MTLinear consistently improves as the lookback length increases, a trend not typically observed with transformer-based models [44].

Grouping criteria. We now examine the effect of various angles, $\bar{\alpha} \in \pi/2, \pi/3, \pi/4, \pi/6, 0$, each representing a maximum cosine similarity threshold for grouping variates. Specifically, $\bar{\alpha} = 0$ enforces strict clustering, with one variate per cluster, while $\bar{\alpha} = \pi/2$ results in a single model across all variates, as groups are formed based on absolute correlation values. In Fig. 4, we plot MTLinear's performance for each $\bar{\alpha}$ on the x-axis, with corresponding MSE scores on the y-axis. Our findings indicate that $\bar{\alpha}$ impacts datasets differently, likely due to varying degrees of the tragic triad issues. Weather and ETTm1 perform well with most groupings, suggesting that a shared model does not eliminate gradient conflicts. Conversely, for ILI, $\bar{\alpha} = \pi/2$ proves optimal. Additionally, model size plays a role, as groupings with $\bar{\alpha} > 0$ yield a more compact model—especially relevant for datasets with numerous variates. Tab. 6 shows that this approach can reduce model size by more than half while enhancing performance. Notably, in each sub-plot of Fig. 4, several MTLinear configurations outperform iTransformer, indicating that linear forecasters remain effective in TSF

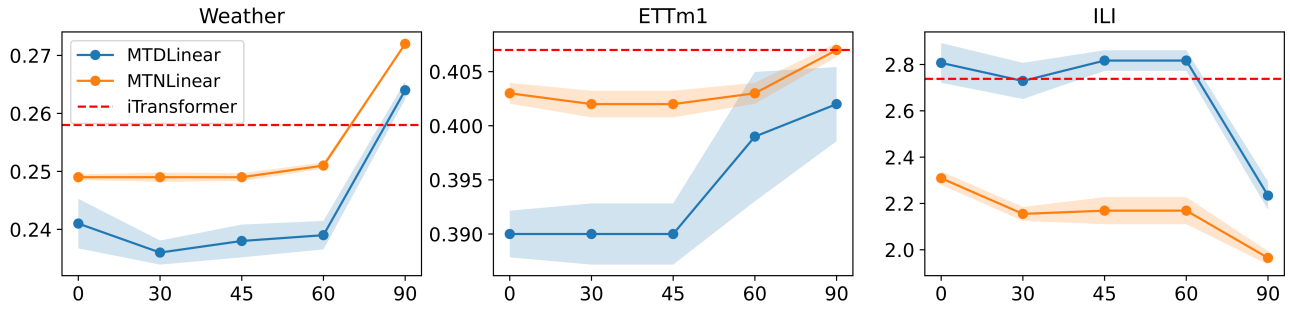

Figure 4: MSE measures for different clustering $\bar{\alpha}$. The red dashed line is the mean for iTransformer. The results suggest that MTLinear is comparable or better in comparison to iTransformer.

Table 3: A comparison of gradient manipulation techniques to MTLinear with DLinear NLinear.

	DLinear						
Dataset	Ours	GradNorm	CoV-W	PCgrad	CAGrad	Nash-MTL	Baseline
ETTm1	**0.399**	0.472	0.406	0.411	0.406	0.406	0.406
ETTm2	**0.284**	0.447	0.316	0.302	0.306	0.325	0.323
Exchange	**0.289**	0.379	0.311	0.325	0.375	0.375	0.309
Weather	**0.244**	0.280	0.268	0.266	0.267	0.267	0.267
ILI	**2.234**	4.850	2.600	2.443	2.350	2.780	2.728

	NLinear						
Dataset	Ours	GradNorm	CoV-W	PCgrad	CAGrad	Nash-MTL	Baseline
ETTm1	**0.403**	0.438	0.410	0.414	0.414	0.413	0.410
ETTm2	**0.280**	0.299	0.285	0.284	0.284	0.286	0.286
Exchange	0.410	0.419	0.379	0.371	**0.357**	**0.357**	0.378
Weather	**0.249**	0.306	0.274	0.273	0.273	0.274	0.273
ILI	**1.965**	2.540	2.174	2.150	1.987	2.201	2.213

Table 4: MTLinear based on different linear modules.

	MTLinear			MTRLinear			MTDLinear			MTNLinear		
Dataset	Ours	Baseline	% Imp	Ours	Baseline	% Imp	Ours	Baseline	% Imp	Ours	Baseline	% Imp
ETTm1	**0.397**	0.413	3.87%	**0.400**	0.413	3.15%	**0.399**	0.406	1.72%	**0.403**	0.410	1.71%
ETTm2	**0.291**	0.324	10.19%	**0.280**	0.286	2.1%	**0.284**	0.323	12.07%	**0.280**	0.286	2.1%
ETTh1	**0.456**	0.464	1.72%	0.445	0.445	0.0%	**0.456**	0.471	3.18%	**0.443**	0.446	0.67%
ETTh2	**0.471**	0.489	3.68%	**0.376**	0.377	0.27%	**0.453**	0.498	9.04%	**0.373**	0.374	0.27%
ECL	**0.198**	0.209	5.26%	**0.203**	0.214	5.14%	**0.198**	0.209	5.26%	**0.204**	0.214	4.67%
Exchange	**0.284**	0.289	1.73%	0.370	**0.359**	-3.06%	**0.289**	0.309	6.47%	0.410	**0.378**	-8.47%
Traffic	**0.621**	0.625	0.64%	0.623	0.623	0.0%	**0.621**	0.625	0.64%	0.624	0.624	0.0%
Weather	**0.241**	0.268	10.07%	**0.244**	0.272	10.29%	**0.238**	0.267	10.86%	**0.249**	0.273	8.79%
ILI	**2.320**	2.858	18.82%	**2.148**	2.423	11.35%	**2.234**	2.728	18.11%	**1.965**	2.213	11.21%

and merit inclusion in new methods. Finally, we study the correlation-conflict similarity in Fig. 6, finding that the correlation and conflict matrices are similar for DLinear and PatchTST but not for Autoformer. This difference may stem from PatchTST and DLinear's use of weight-per-time linear layers, unlike Autoformer.

5.3 Variants of Gradient Manipulation

We compare our approach with robust gradient manipulation baselines, including GradNorm [3] and PCGrad [42], which adjust gradients per task to address magnitude and conflict issues. Additionally, we benchmark against Cov-Weighting (CoV-W) [10], which emphasizes tasks with higher trailing variance, assuming that reduced variance indicates a satisfied loss. We also include CAGrad [20] and Nash-MTL [26]. Exploring different scaling schemes helps assess whether the variance of $e_{i,j}$ impacts results beyond the error alone. As shown in Tab. 3, MTLinear outperforms these methods while being more efficient, requiring only a single backward pass compared to the k passes needed by PCGrad and GradNorm. We further analyze the link between error term $e_{i,j}$ and gradient magnitude in Fig. 5, where we find a strong correlation across the Weather, ETTm2, and ILI datasets, with higher

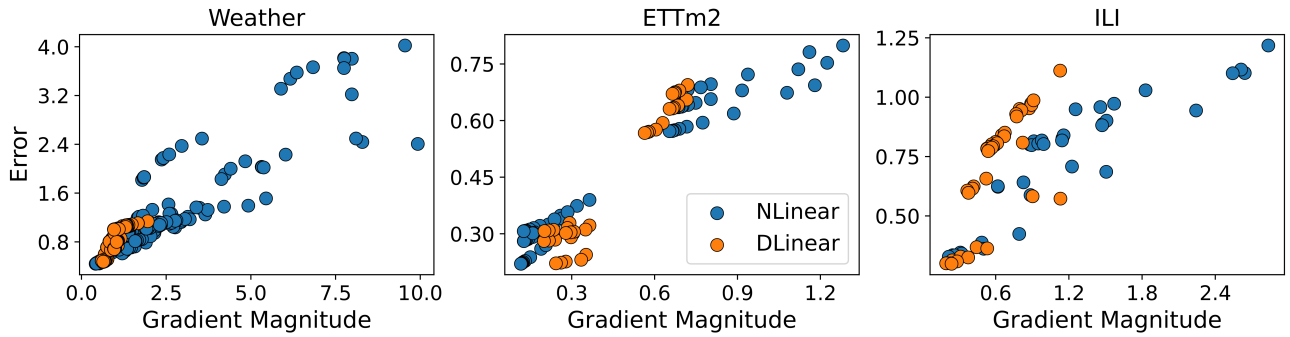

Figure 5: We plot the error $e_{i,j}$ of a given loss vs. its gradient's magnitude. These results highlight the clear positive correlation between the two for both DLinear and NLinear.

errors corresponding to larger gradients—motivating our penalty in Sec. 4.3.

5.4 Model Variations and Linear Probing

The MTLinear method is versatile and can integrate with linear layers beyond NLinear or DLinear, including a standard linear layer, to boost performance. We present these results in Tab. 4, where MTLinear-based approaches consistently improve baseline MSE scores by up to 18% for ILI and 10% across other datasets.

In TSF transfer learning with foundation models and Transformer architectures, a common approach is linear probing, where only the final linear layer is fine-tuned on a pre-trained model [27, 49, 2]. To assess the transfer capabilities of TSF models, we conducted an experiment where several baselines were first trained on a source dataset and then fine-tuned on a target dataset. The baselines include MTLinear, MTLinear Probing, Zero-shot, and PatchTST. MTLinear Probing refers to PatchTST with our MTLinear module replacing its decoder, while Zero-shot involves no fine-tuning. MSE results are presented in Tab. 5, with the top table showing results for Electricity as the source dataset with Weather and ETTm2 as targets, and the bottom table displaying Weather as the source dataset with Electricity and ETTm2 as targets. All experiments used a 336 lookback, as in PatchTST's original setup.

Table 5: Linear probing with MTLinear (MTLi) using PatchTST on source datasets Electricity (top) and Weather (bottom). The model is then tested on target datasets. See text for details.

Dataset	MTLi	MTLi Probing	Zero-shot	PatchTST
ECL*	**0.161**	**0.161**	0.162	0.162
Weather	**0.22**	0.226	0.326	0.229
ETTm2	**0.254**	0.258	0.344	0.257

Dataset	MTLi	MTLi Probing	Zero-shot	PatchTST
Weather*	**0.22**	0.224	0.231	0.231
ECL	**0.161**	0.165	0.867	0.162
ETTm2	**0.254**	**0.254**	0.313	0.257

Our conclusions are as follows: 1) MTLinear consistently outperforms all transfer learning configurations, including PatchTST, on the evaluated datasets; 2) Training PatchTST with MTLinear probing further enhances results, evidenced by MSE reductions for both Electricity and Weather; and 3) MTLinear probing applied to non-pre-trained PatchTST yields significant improvements, bringing linear fine-tuning results close to, and in some cases surpassing, fully trained PatchTST outcomes. However, this configuration still does not match the performance of a standard MTLinear setup without PatchTST.

6 CONCLUSION

In this work, we considered the task of multivariate time series forecasting. While several strong existing works treat different variates independently, we offered to exploit their inter-relations. We do so by viewing multivariate forecasting as a multi-task learning problem, allowing to consider forecasting through the tragic triad challenges. Our analysis of linear models and their gradients suggest that variates are optimized along the direction of the variate, and scaled proportionally to the prediction error. Based on our analysis, we propose to group variates together if the angle between them is small, and to balance variate groups using their error. Ultimately, each variate group is viewed as an independent task, solved using a single linear module, and combined into an optimization framework we named MTLinear. Our approach shows competitive results in several benchmarks in comparison to strong baseline methods. While our method effectively utilizes inter-related variate information, its structure limits the exploitation of non-correlated cross-variate information. This is significant, as different variates—regardless of

their correlation—may contain unique information that could enhance the overall predictive performance. Another limitation is our reliance on a specific linear layer type, such as DLinear or NLinear. We believe that a unified framework combining these or incorporating additional layer types would further strengthen the MTLinear approach.

In the future, we plan to investigate the incorporation of the multi-head MTLinear as a decoder in state-of-the-art methodologies. Additionally, we will explore the effect of learning the clusters during training, instead of computing them as a pre-processing step. Lastly, we wish to address the limitations of our work. In general, we believe that further studying the inter-relations between variates of real-world time series data is important, and our work is a first step toward achieving that goal.

Acknowledgments

This research was partially supported by the Lynn and William Frankel Center of the Computer Science Department, Ben-Gurion University of the Negev, an ISF grant 668/21, an ISF equipment grant, and by the Israeli Council for Higher Education (CHE) via the Data Science Research Center, Ben-Gurion University of the Negev, Israel.

References

[1] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[2] C. Chang, W.-Y. Wang, W.-C. Peng, and T.-F. Chen. LLM4TS: Aligning Pre-Trained LLMs as Data-Efficient Time-Series Forecasters. *arXiv preprint arXiv:2308.08469*, 2023.

[3] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018.

[4] Z. Chen, E. Jiaze, X. Zhang, H. Sheng, and X. Cheng. Multi-task time series forecasting with shared attention. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 917–925. IEEE, 2020.

[5] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1724–1734. ACL, 2014.

[6] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009.

[7] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn. Efficiently Identifying Task Groupings for Multi-Task Learning. *Advances in Neural Information Processing Systems*, 2021.

[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT press, 2016.

[9] C. W. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

[10] R. Groenendijk, S. Karaoglu, T. Gevers, and T. Mensink. Multi-Loss Weighting with Coefficient of Variations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1469–1478, 2020.

[11] S. Guangyuan, Q. Li, W. Zhang, J. Chen, and X.-M. Wu. Recon: Reducing Conflicting Gradients From the Root For Multi-Task Learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[12] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The Elements of Statistical Learning: Data mining, Inference, and Prediction*, volume 2. Springer, 2009.

[13] S. Hochreiter and J. Schmidhuber. Long Short-term Memory. *Neural computation*, 1997.

[14] K. Ishihara, A. Kanervisto, J. Miura, and V. Hautamaki. Multi-task learning with attention for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2902–2911, 2021.

[15] I. Kaufman and O. Azencot. Data representations' study of latent image manifolds. In *International Conference on Machine Learning*, pages 15928–15945. PMLR, 2023.

[16] I. Kaufman and O. Azencot. Analyzing deep transformer models for time series forecasting via manifold learning. *Transactions on Machine Learning Research, TMLR*, 2024.

[17] I. Kaufman and O. Azencot. First-order manifold data augmentation for regression learning. In *Forty-first International Conference on Machine Learning, ICML*, 2024.

[18] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[19] Z. Li, S. Qi, Y. Li, and Z. Xu. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*, 2023.

[20] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu. Conflict-Averse Gradient Descent for Multi-task Learning. *Advances in Neural Information Processing Systems*, 2021.

[21] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

[22] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1930–1939, 2018.

[23] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[24] I. Naiman, N. Berman, I. Pemper, I. Arbiv, G. Fadlon, and O. Azencot. Utilizing image transforms and diffusion models for generative modeling of short and long time series. *Advances in Neural Information Processing Systems*, 2024.

[25] I. Naiman, N. B. Erichson, P. Ren, M. W. Mahoney, and O. Azencot. Generative modeling of regular and irregular time series data via koopman vaes. In *The Twelfth International Conference on Learning Representations,ICLR*, 2024.

[26] A. Navon, A. Shamsian, I. Achituve, H. Maron, K. Kawaguchi, G. Chechik, and E. Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning, ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 16428–16446. PMLR, 2022.

[27] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

[28] L. Nochumsohn, M. Moshkovitz, O. Avner, D. Di Castro, and O. Azencot. Beyond data scarcity: A frequency-driven framework for zero-shot forecasting. *arXiv preprint arXiv:2411.15743*, 2024.

[29] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations, ICLR*, 2020.

[30] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32, 2019.

[32] P. Ren, R. Nakata, M. Lacour, I. Naiman, N. Nakata, J. Song, Z. Bi, O. A. Malik, D. Morozov, O. Azencot, et al. Learning physics for unveiling hidden earthquake ground motions via conditional generative modeling. *arXiv preprint arXiv:2407.15089*, 2024.

[33] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[34] T. Schreiber. Measuring information transfer. *Physical review letters*, 85(2):461, 2000.

[35] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[36] X. Song, S. Zheng, W. Cao, J. Yu, and J. Bian. Efficient and Effective Multi-task Grouping via Meta Learning on Task Combinations. *Advances in Neural Information Processing Systems*, 2022.

[37] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

[38] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3614–3633, 2021.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Advances in Neural Information Processing Systems*, 2021.

[41] W. Xue, T. Zhou, Q. Wen, J. Gao, B. Ding, and R. Jin. Make transformer great again for time series forecasting: Channel aligned robust dual transformer. *arXiv preprint arXiv:2305.12095*, 2023.

[42] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient Surgery for Multi-Task Learning. *Advances in Neural Information Processing Systems*, 2020.

[43] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.

[44] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[45] Y. Zhang and J. Yan. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In *The Eleventh International Conference on Learning Representations*, 2022.

[46] Y. Zhang and Q. Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

[47] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, 2021.

[48] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *International Conference on Machine Learning*. PMLR, 2022.

[49] T. Zhou, P. Niu, L. Sun, R. Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

A Linear Analysis Proof

Below, we provide the full derivation for the gradient provided in Eq. (6). We recall that $F(\Theta)$ can be viewed as a function $F(\Theta) : \mathbb{R}^{l+1\times h} \to \mathbb{R}$ or as a sum of functions $F(\theta_j) : \mathbb{R}^{l+1} \to \mathbb{R}$. After flattening, the gradient of the first function is an object of size $l + 1 \cdot h$, where every $l + 1$ elements correspond to a particular gradient for $F(\theta_j)$, $j = 1, \ldots, h$. Thus, it is sufficient to derive the gradient of $F(\theta_j)$, as we detail below.

$$
\begin{aligned}
F(\Theta) &= \frac{1}{kh}\sum_{j=1}^{h}\sum_{i=1}^{k}(x_i^T\theta_j - y_{j,i})^2 \ , \quad F(\theta_j) = \frac{1}{k}\sum_{i=1}^{k}(x_i^T\theta_j - y_{j,i})^2 \\
F(\theta_j + \delta\theta_j) - F(\theta_j) &= \frac{1}{k}\sum_{i=1}^{k}(x_i^T(\theta_j + \delta\theta_j) - y_{j,i})^2 - (x_i^T\theta_j - y_{j,i})^2 \\
&=^{*} \frac{1}{k}\sum_{i=1}^{k} x_i^T(\theta_j + \delta\theta_j)[x_i^T(\theta_j + \delta\theta_j) - y_{j,i}] - x_i^T\theta_j(x_i^T\theta_j - y_{j,i}) \\
&= \frac{1}{k}\sum_{i=1}^{k} x_i^T(\theta_j + \delta\theta_j)(x_i^T\theta_j - y_{j,i}) + x_i^T(\theta_j + \delta\theta_j)(x_i^T\delta\theta_j - y_{j,i}) - x_i^T\theta_j(x_i^T\theta_j - y_{j,i}) \\
&=^{**} \frac{1}{k}\sum_{i=1}^{k} x_i^T\delta\theta_j(x_i^T\theta_j - y_{j,i}) + x_i^T\delta\theta_j(x_i^T\theta_j - y_{j,i}) \\
&= \frac{2}{k}\sum_{i=1}^{k}\delta\theta_j^T x_i(x_i^T\theta_j - y_{j,i}) = \delta\theta_j^T\nabla_{\theta_j}F(\theta_j) \ ,
\end{aligned}
$$

where the starred pass is where we leave only elements that depend on θ_j, and the double starred pass is due to eliminating non first-order in $\delta\theta_j$ elements.

B Convergence Proof

In what follows, we provide a straightforward proof for the convexity of our optimization, detailed in Eq. (7). Then, under certain mild conditions that are satisfied by our problem, stochastic gradient descent (SGD) is guaranteed to converge [1]. We recall that per cluster, our loss take the form of $|X^T\Theta - Y^T|^2_{W^a}$, where $X \in \mathbb{R}^{l\times k}, Y \in \mathbb{R}^{h\times k}, \theta \in \mathbb{R}^{l\times h}$, and $W^a \in \mathbb{R}^{h\times k}$. The norm $|A|_{W^a} := \text{trace}(A^T \odot W^a A)$, where $\odot$ is an element-wise multiplication operation. To prove that Eq. (7) is convex, we will show that its Hessian is a fixed, semi-positive definite (SPD) matrix. First, we observe that since each element $W^a_{ij} \geq 0$ by Eq. (8), then it holds that $|A|^2_{W^a} = \text{trace}[A^T \odot \sqrt{W^a}(\sqrt{W^a})^T \odot A]$, where $\sqrt{A}$ is the element-wise square-root of the matrix A. Additionally, we denote by A_W a matrix scaled by $\sqrt{W^a}$, i.e., $A_W = (\sqrt{W^a})^T \odot A$. Then, it follows that

$$
\begin{aligned}
|X^T\Theta - Y^T|^2_{W^a} &= \text{trace}\left[(X^T\Theta - Y^T)^T \odot \sqrt{W^a}(\sqrt{W^a})^T \odot (X^T\Theta - Y^T)\right] \\
&= \text{trace}\left[(\Theta^T X_W - Y_W)(X_W^T\Theta - Y_W^T)\right] \ .
\end{aligned}
$$

We differentiate with respect to Θ under the trace$(\cdot)$ operation and obtain

$$
\begin{aligned}
\frac{\partial}{\partial\Theta}|X^T\Theta - Y^T|^2_{W^a} &= \frac{\partial}{\partial\Theta}\text{trace}\left[(\Theta^T X_W - Y_W)(X_W^T\Theta - Y_W^T)\right] \\
&= 2X_W(X_W^T\Theta - Y_W^T) \ ,
\end{aligned}
$$

which follows from properties of the trace$(\cdot)$ and computing the derivative per element. Taking another derivative yields

$$
\frac{\partial}{\partial\Theta}\left[2X_W(X_W^T\Theta - Y_W^T)\right] = 2X_W X_W^T \ ,
$$

which is an SPD matrix as it is the product of a matrix multiplied by the same matrix transposed.

Finally, to guarantee convergence via SGD, we need the gradient $2X_W(X_W^T\Theta - Y_W^T)$ to be a Lipschitz function. Indeed, this property holds with a Lipschitz constant of $L = 2|X_W X_W^T|_F$, where $|\cdot|_F$ is the Frobenius norm. In practice, the data and scaling are bounded and thus $L \ll \infty$.

C Variate Grouping

C.1 Correlation and Conflict Matrices

In this section, we present the correlation matrices and conflict matrices for the datasets Weather and ETTm2. In Fig. 6, each cell in the conflict matrices represents the count of all conflicts between two variates occurred during training. A resemblance is apparent mostly for DLinear and PatchTST. One interesting observation is that strong negative correlations between variates are associated with a low number of conflicts.

C.2 Task Affinity Grouping (TAG)

Task affinity grouping (TAG) [7] is a task grouping method, that suggests to select task groups based on the quality of transferability between tasks. In this context, the term *inter-task affinity* encodes the transferability of each task with the remaining tasks during training. TAG proposes two steps. First, obtain a measure of pair-wise task similarity by training a shared model across tasks and observing the contribution of single-task optimization updates on the remaining tasks; the inter-task affinity between tasks is collected and later acts as a measure of task similarity. The next step is calculating the optimal task groups with a selection process that maximizes the total affinity score. When applied to time series forecasting, certain elements in TAG pose serious difficulties: 1) Training the first steps of this framework could impose a great amount of overhead when applied to the dataset Electricity or Weather since they contain 321 and 21 variates (tasks) respectively. 2) After collecting all the inter-task affinities the selection process maximizes the total affinity score. However, this problem is NP-hard, and thus it requires approximation for using a large number of tasks. Therefore, MTLinear (our approach) can be seen as a more practical approach to variate grouping, and model assignment in TSF settings.

C.3 Hierarchical Clustering

In this section, we briefly present the *hierarchical clustering* algorithm [12] we utilize in Sec. 4.2. Hierarchical clustering is a general name for a bottom-up (agglomerative) strategy or top-down (divisive) strategy. In this work, we focus on agglomerative clustering which commences with each individual object forming its distinct group. It then systematically combines objects or groups that are proximate to each other until all groups are

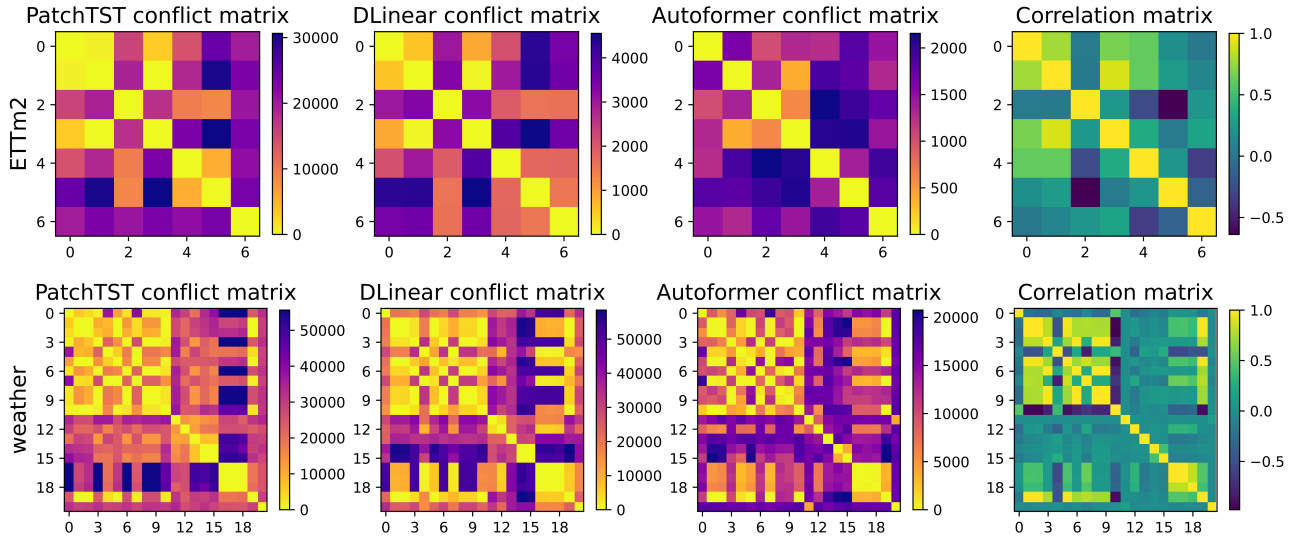

Figure 6: The Pearson correlation matrix (rightmost) and conflict matrices, where each element represents the total number of conflicts seen between two variates during training.

joined into a single entity at the highest level of the hierarchy or until a termination condition is met. Apart from selecting a metric such as Euclidean distance, a *linkage* criterion must be determined. The linkage defines how to measure the similarity between clusters, which can be defined by the closest pair (*single linkage*), farthest pair (*complete linkage*), or the average of the distances of each observation of the two sets (*average linkage*). A tree structure called a dendrogram is often used to represent the process of hierarchical clustering. We show an example in Fig. 7.

In our method, we use the Pearson correlation coefficient as the metric between two clusters (variate groups) and employ complete linkage as our linkage strategy. The following table, Table 6, presents the number of clusters (groups) for each dataset and criteria $\bar{\alpha}$. We denote the distance between groups by $d_{\bar{\alpha}}$, and we define it as follows, $d_{\bar{\alpha}} = 1 - \cos(\bar{\alpha})$. This term expresses the maximal correlation distance in a given cluster.

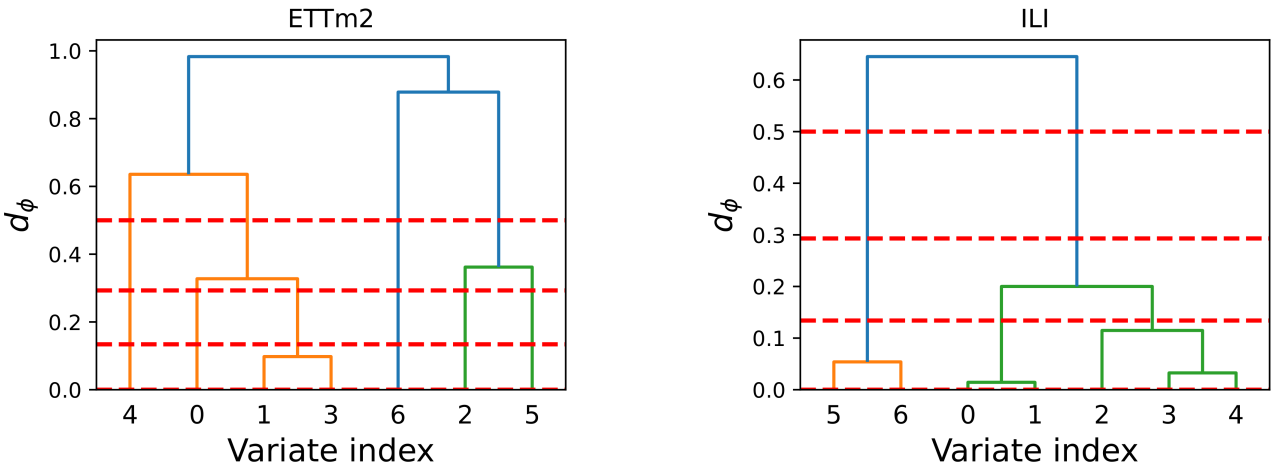

Figure 7: The dendrograms for the ILI and ETTm2 datasets. The red lines represent the cut points $d_{\bar{\alpha}}$ associated with $\bar{\alpha} \in \{\pi/2, \pi/3, \pi/4, \pi/6\}$.

Table 6: Number of variate groups given $\bar{\alpha}$ for each dataset.

Dataset	$\bar{\alpha} = 0$ (max variates)	$\bar{\alpha} = \pi/6$	$\bar{\alpha} = \pi/4$	$\bar{\alpha} = \pi/3$	$\bar{\alpha} = \pi/2$
ECL	321	175	79	44	1
Traffic	862	862	838	753	1
Weather	21	13	11	10	1
ILI	7	3	2	2	1
ETT	7	6	6	4	1
Exchange	8	8	8	7	1

C.4 Memory Usage

Since MTLinear relies on duplicated versions of a single linear model variation, the memory usage of each model can be easily computed according to the number of selected groups. Therefore, Tab. 6 also describes the memory complexity in units of a single linear model variation memory usage. For example, MTLDinear for Electricity $\bar{\alpha} = \pi/6$ consists of 175 times the memory of a single DLinear model. While this number may seem large, in practice, it is part of a single module implemented in Pytorch where all linear components are stacked such that only the depth of one layer remains. Another benefit of MTLinear is an effective and efficient implementation of quasi-channel independent modeling, where the full memory usage of a channel independent approach ($\bar{\alpha} = 0$) is replaced with a smaller and utilized representation.

D Experimental Details

Datasets. The proposed method is extensively evaluated on seven common benchmark datasets from different sectors: industrial, weather, energy, and health. The **Electricity** (ECL) dataset includes the hourly electricity consumption data of 321 customers spanning from 2012 to 2014. **Weather** is a meteorological dataset recorded every 10 minutes throughout the entire year of 2020, featuring 21 meteorological indicators such as air temperature and humidity. The **ILI** dataset includes weekly recorded data on influenza-like illness patients from the Centers for Disease Control and Prevention of the United States, spanning from 2002 to 2021. The **ETT** dataset comprises data gathered from electricity Transformers, encompassing load, and oil temperature readings recorded at 15-minute intervals. **ETTh2**, **ETTh1**, **ETTm2**, and **ETTm1** form different interval representations, 2 hours, 1 hour, 30 minutes and 15 minutes, respectively.

Baselines. We selected SOTA and prominent models as the benchmark baselines in our experiments: PatchTST [27], iTransformer [21], Crossformer [45], FEDformer [48], Autoformer [40], GPT4TS [49] as well as linear based models Linear, DLinear, NLinear, RLinear [44, 19]. The baseline results shown in Tabs. 1, and App. 7 are comprised of the reported results in [21] except for ILI, where the results for Crossformer, FEDformer, and Autoformer were imported from the original paper and the latter was reproduced. In Tab. App. 8 the results for PatchTST, GPT4TS, DLinear, and NLinear are the original reported results. The remaining experiment tables and figures are reproduced based on the original implementation and hyper-parameters.

Experimental setting. For all experiments we take the average score of three different seeds. For each seed, we perform grid search and select the setting with the best validation score. The given grid search includes $\bar{\alpha} \in \{\pi/2, \pi/3, \pi/4, \pi/6\}$ and $a \in \{1, 2\}$ for the grouping and penalty parameters, respectively. The other reported results rely on the original implementation and hyperparameters. Most experiments use the standard lookback l of 96, unless mentioned otherwise.

MTLinear implementation details. MTLinear and its multi-head linear modules are implemented in Pytorch [31]. Similar to the other baselines, we use early stopping. However, we deploy a multi-early stopping scheme, hence, each model's training ends at its own time, thus lifting another form of dependency on other model groups. The maximal number of epochs is set to 20, and learning-rate and batch size are set to 0.01 and 32, respectively, for all configurations. For the MTLinear setting and other reproduced results, we used RTX 4090 24GB GPU.

E Sensitivity Parameter a Ablation

In this section we provide experiments ablating the effect of a on forecasting, in Fig. 8. We consider the values a = [0,1,2]. In most cases, a higher a leads to better results with exceptions. It is also shown that when $\alpha = 90$ the a has a stronger impact on the results, but when variates are grouped that difference is decreased. This behavior can be explained by the fact that very different variates also share large gradient scale differences. In practice, a=2 is selected in most cases after hyper-parameter search in the main results.

F Extended Main Results

This table is the extended version of Tab. 7 with all 4 horizons presented separately. Additionally, we added a table that compares MTLinear and PatchTST [27] with an input lookback of 336, and GPT4TS [49] in Table 8.

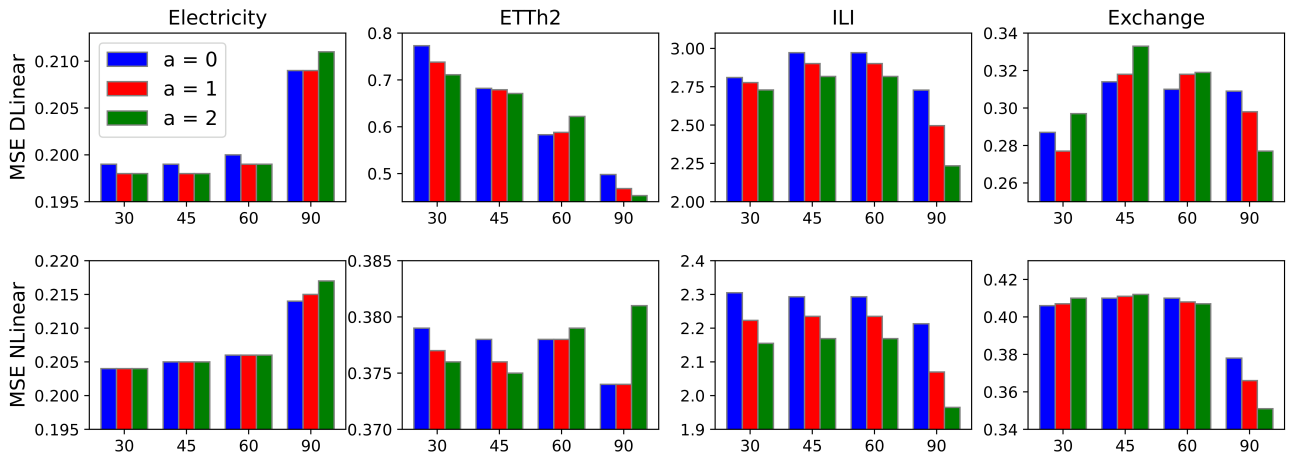

Figure 8: A performance comparison between different values of the sensitivity parameter $a \in \{0, 1, 2\}$ with respect to each $\bar{\alpha} \in \{30, 45, 60, 90\}$.

Table 7: Multivariate forecasting results of MTLinear (ours) compared to other strong baselines of different classes. Each score represents one of four forecast horizons $h \in \{24, 36, 48, 60\}$ and 36 input length for ILI, as well as $h \in \{96, 192, 336, 720\}$ and 96 input length for the remaining. A **bold** and underlined notation represent the best and second-best scores, respectively.

Dataset		MTDLinear		MTNLinear		iTransformer		PatchTST		Crossformer		DLinear		FEDformer		Autoformer	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.337	**0.363**	0.341	0.369	0.334	0.368	**0.329**	0.367	0.404	0.426	0.345	0.372	0.379	0.419	0.505	0.475
	192	0.379	0.388	0.381	0.387	0.377	0.391	**0.367**	**0.385**	0.450	0.451	0.380	0.389	0.426	0.441	0.553	0.496
	336	0.412	0.414	0.413	**0.409**	0.426	0.420	**0.399**	0.410	0.532	0.515	0.413	0.413	0.445	0.459	0.621	0.537
	720	0.468	0.445	0.478	0.443	0.491	0.459	**0.454**	**0.439**	0.666	0.589	0.474	0.453	0.543	0.490	0.671	0.561
ETTm2	96	0.179	0.264	**0.175**	**0.254**	0.180	0.264	**0.175**	0.259	0.287	0.366	0.193	0.292	0.203	0.287	0.255	0.339
	192	0.245	0.308	**0.240**	**0.296**	0.250	0.309	0.241	0.302	0.414	0.492	0.284	0.362	0.269	0.328	0.281	0.340
	336	0.306	0.350	**0.301**	**0.335**	0.311	0.348	0.305	0.343	0.597	0.542	0.369	0.427	0.325	0.366	0.339	0.372
	720	0.407	0.415	**0.402**	**0.393**	0.412	0.407	**0.402**	0.400	1.730	1.042	0.554	0.522	0.421	0.415	0.433	0.432
ETTh1	96	0.386	0.396	0.387	**0.393**	0.386	0.405	0.414	0.419	0.423	0.448	0.386	0.400	**0.376**	0.419	0.449	0.459
	192	0.441	0.426	0.439	**0.421**	0.441	0.436	0.460	0.445	0.471	0.474	0.437	0.432	**0.420**	0.448	0.500	0.482
	336	0.490	0.455	0.476	**0.441**	0.487	0.458	0.501	0.466	0.570	0.546	0.481	0.459	**0.459**	0.465	0.521	0.496
	720	0.506	0.488	**0.472**	**0.460**	0.503	0.491	0.500	0.488	0.653	0.621	0.519	0.516	0.506	0.507	0.514	0.512
ETTh2	96	0.300	0.345	0.288	**0.336**	**0.297**	0.349	0.302	0.348	0.745	0.584	0.333	0.387	0.358	0.397	0.346	0.388
	192	0.390	0.405	**0.375**	**0.388**	0.380	0.400	0.388	0.400	0.877	0.656	0.477	0.476	0.429	0.439	0.456	0.452
	336	0.500	0.482	**0.412**	**0.423**	0.428	0.432	0.42	0.433	1.043	0.731	0.594	0.541	0.496	0.487	0.482	0.486
	720	0.623	0.555	**0.418**	**0.440**	0.427	0.445	0.431	0.446	1.104	0.763	0.831	0.657	0.463	0.474	0.515	0.511
ECL	96	0.183	0.268	0.185	0.265	**0.148**	**0.240**	0.181	0.270	0.219	0.314	0.197	0.282	0.193	0.308	0.201	0.317
	192	0.183	0.271	0.186	0.268	**0.162**	**0.253**	0.188	0.274	0.231	0.322	0.196	0.285	0.201	0.315	0.222	0.334
	336	0.196	0.286	0.201	0.283	**0.178**	**0.269**	0.204	0.293	0.246	0.337	0.209	0.301	0.214	0.329	0.231	0.338
	720	0.231	0.318	0.243	**0.317**	**0.225**	**0.317**	0.246	0.324	0.280	0.363	0.245	0.333	0.246	0.355	0.254	0.361
Exchange	96	**0.084**	0.202	0.085	**0.201**	0.086	0.206	0.088	0.205	0.256	0.367	0.088	0.218	0.148	0.278	0.197	0.323
	192	**0.173**	0.300	0.183	0.302	0.177	**0.299**	0.176	**0.299**	0.470	0.509	0.176	0.315	0.271	0.315	0.300	0.369
	336	0.306	0.412	0.355	0.428	0.331	0.417	**0.301**	**0.397**	1.268	0.883	0.313	0.427	0.460	0.427	0.509	0.524
	720	**0.595**	**0.595**	1.015	0.758	0.847	0.691	0.901	0.714	1.767	1.068	0.839	0.695	1.195	0.695	1.447	0.941
Traffic	96	0.648	0.396	0.647	0.383	**0.395**	**0.268**	0.462	0.295	0.522	0.290	0.650	0.396	0.587	0.366	0.613	0.388
	192	0.594	0.365	0.598	0.359	**0.417**	**0.276**	0.466	0.296	0.530	0.293	0.598	0.370	0.604	0.373	0.616	0.382
	336	0.601	0.368	0.606	0.362	**0.433**	**0.283**	0.482	0.304	0.558	0.305	0.605	0.373	0.621	0.383	0.622	0.337
	720	0.640	0.393	0.644	0.382	**0.467**	**0.302**	0.514	0.322	0.589	0.328	0.645	0.394	0.626	0.382	0.660	0.408
Weather	96	0.159	0.221	0.166	**0.211**	0.174	0.214	0.177	0.218	**0.158**	0.230	0.196	0.255	0.217	0.296	0.266	0.336
	192	**0.202**	0.268	0.212	**0.252**	0.221	0.254	0.225	0.259	0.206	0.277	0.237	0.296	0.276	0.336	0.307	0.367
	336	**0.259**	0.318	0.268	**0.294**	0.278	0.296	0.278	0.297	0.272	0.335	0.283	0.335	0.339	0.380	0.359	0.395
	720	**0.332**	0.373	0.349	**0.346**	0.358	0.347	0.354	0.348	0.398	0.418	0.345	0.381	0.403	0.428	0.419	0.428
ILI	24	2.246	0.993	**2.126**	**0.928**	2.754	1.103	2.390	0.999	3.041	1.186	2.398	1.040	3.228	1.260	3.483	1.287
	36	2.233	0.989	**1.914**	**0.886**	2.707	1.074	2.331	0.994	3.406	1.232	2.646	1.088	2.679	1.080	3.103	1.148
	48	2.102	0.962	**1.795**	**0.867**	2.610	1.068	2.488	1.033	3.459	1.221	2.614	1.086	2.622	1.078	2.669	1.085
	60	2.357	1.036	**2.023**	**0.927**	2.881	1.147	2.475	1.018	3.640	1.305	2.804	1.146	2.857	1.157	2.770	1.125
Average		0.575	0.440	**0.550**	**0.422**	0.611	0.436	0.584	0.431	0.902	0.579	0.649	0.475	0.681	0.483	0.741	0.515
1^{st} Count		6	2	**12**	**23**	9	9	7	4	1	0	0	0	3	0	0	0

Table 8: Multivariate forecasting results of MTLinear (ours) compared to other strong baselines of different classes. Each score represents one of four forecast horizons $h \in \{24, 36, 48, 60\}$ and 104 input length for ILI, as well as $h \in \{96, 192, 336, 720\}$ and 336 input length for the remaining. A **bold** and underlined notation represent the best and second-best scores, respectively.

Class		MTLinear		Transformer		Linear	
Dataset		MTDLinear	MTNLinear	GPT4TS	PatchTST	DLinear	NLinear
ETTm2	96	0.166	**0.160**	0.173	0.165	0.167	0.167
	192	0.222	**0.216**	0.229	0.220	0.224	0.221
	336	0.285	**0.272**	0.286	0.278	0.281	0.274
	720	0.377	**0.367**	0.378	**0.367**	0.397	0.368
ECL	96	0.133	0.134	0.139	**0.130**	0.140	0.141
	192	**0.148**	0.149	0.153	**0.148**	0.153	0.154
	336	**0.164**	0.167	0.169	0.167	0.169	0.171
	720	**0.199**	0.205	0.206	0.202	0.203	0.210
Weather	96	**0.145**	0.146	0.162	0.152	0.176	0.182
	192	**0.187**	0.189	0.204	0.197	0.220	0.225
	336	**0.238**	0.241	0.254	0.249	0.265	0.271
	720	**0.310**	0.320	0.326	0.320	0.323	0.338
ILI	24	2.024	1.611	2.063	**1.522**	2.215	1.683
	36	2.063	1.546	1.868	**1.430**	1.963	1.703
	48	2.086	**1.554**	1.790	1.673	2.130	1.719
	60	2.256	1.735	1.979	**1.529**	2.368	1.819
1^{st} Count		**7**	5	0	6	0	0

F.1 Main results with standard deviation

In Tab. 9 the main results for MTLinear are presented alongside the standard deviation. We should note that the standard deviation for other baseline models is not given here since the corresponding results were taken from other papers, mainly [21].

Table 9: Multivariate forecasting results of MTLinear (ours) with the standard deviation, Each score represents one of four forecast horizons $h \in \{24, 36, 48, 60\}$ and 36 input length for ILI, as well as $h \in \{96, 192, 336, 720\}$ and 96 input length for the remaining.

Dataset		MTDLinear		MTNLinear	
		MSE	MAE	MSE	MAE
ETTm1	96	0.337 ± 0.001	0.363 ± 0.002	0.341 ± 0.002	0.369 ± 0.001
	192	0.379 ± 0.002	0.388 ± 0.005	0.381 ± 0.001	0.387 ± 0.001
	336	0.412 ± 0.003	0.414 ± 0.006	0.413 ± 0.001	0.409 ± 0.001
	720	0.468 ± 0.002	0.445 ± 0.001	0.478 ± 0.001	0.443 ± 0.001
ETTm2	96	0.179 ± 0.0	0.264 ± 0.001	0.175 ± 0.0	0.254 ± 0.0
	192	0.245 ± 0.0	0.308 ± 0.001	0.24 ± 0.0	0.296 ± 0.0
	336	0.306 ± 0.0	0.35 ± 0.002	0.301 ± 0.0	0.335 ± 0.0
	720	0.407 ± 0.005	0.415 ± 0.01	0.402 ± 0.0	0.393 ± 0.0
ETTh1	96	0.386 ± 0.003	0.396 ± 0.004	0.387 ± 0.002	0.393 ± 0.002
	192	0.441 ± 0.005	0.426 ± 0.005	0.439 ± 0.0	0.421 ± 0.0
	336	0.49 ± 0.008	0.455 ± 0.008	0.476 ± 0.001	0.441 ± 0.0
	720	0.506 ± 0.007	0.488 ± 0.006	0.472 ± 0.001	0.46 ± 0.0
ETTh2	96	0.3 ± 0.001	0.345 ± 0.004	0.288 ± 0.001	0.336 ± 0.0
	192	0.39 ± 0.004	0.405 ± 0.004	0.375 ± 0.0	0.388 ± 0.0
	336	0.5 ± 0.017	0.482 ± 0.011	0.412 ± 0.001	0.423 ± 0.0
	720	0.623 ± 0.044	0.555 ± 0.02	0.418 ± 0.002	0.44 ± 0.001
ECL	96	0.183 ± 0.0	0.268 ± 0.0	0.185 ± 0.0	0.265 ± 0.0
	192	0.183 ± 0.0	0.271 ± 0.0	0.186 ± 0.0	0.268 ± 0.0
	336	0.196 ± 0.0	0.286 ± 0.0	0.201 ± 0.0	0.283 ± 0.0
	720	0.231 ± 0.0	0.318 ± 0.001	0.243 ± 0.0	0.317 ± 0.0
Exchange	96	0.084 ± 0.003	0.202 ± 0.003	0.085 ± 0.005	0.201 ± 0.005
	192	0.173 ± 0.01	0.3 ± 0.008	0.183 ± 0.001	0.302 ± 0.002
	336	0.306 ± 0.003	0.412 ± 0.003	0.355 ± 0.003	0.428 ± 0.002
	720	0.595 ± 0.156	0.595 ± 0.06	1.015 ± 0.009	0.758 ± 0.004
Traffic	96	0.648 ± 0.0	0.396 ± 0.0	0.647 ± 0.0	0.383 ± 0.0
	192	0.594 ± 0.0	0.365 ± 0.0	0.598 ± 0.0	0.359 ± 0.0
	336	0.601 ± 0.0	0.368 ± 0.0	0.606 ± 0.0	0.362 ± 0.0
	720	0.64 ± 0.0	0.393 ± 0.0	0.644 ± 0.0	0.382 ± 0.0
Weather	96	0.159 ± 0.001	0.221 ± 0.001	0.166 ± 0.002	0.211 ± 0.001
	192	0.202 ± 0.0	0.268 ± 0.001	0.212 ± 0.0	0.252 ± 0.0
	336	0.259 ± 0.005	0.318 ± 0.006	0.268 ± 0.001	0.294 ± 0.0
	720	0.332 ± 0.005	0.373 ± 0.007	0.349 ± 0.0	0.346 ± 0.0
ILI	24	2.246 ± 0.043	0.993 ± 0.017	2.126 ± 0.027	0.928 ± 0.007
	36	2.233 ± 0.123	0.989 ± 0.049	1.914 ± 0.074	0.886 ± 0.028
	48	2.102 ± 0.058	0.962 ± 0.01	1.795 ± 0.008	0.867 ± 0.005
	60	2.357 ± 0.021	1.036 ± 0.008	2.023 ± 0.007	0.927 ± 0.003

F.2 Gradient manipulation full table results

In this subsection, the full result tables for DLinear 10 and NLinear 11, these results extend Tab. 3 in the main text. Each score represents the average result of three different runs corresponding to different seeds which offer the best validation score after grid search for different parameters.

Table 10: DLinear: Multivariate forecasting results of Gradient manipulation methods, Each score represents one of four forecast horizons $h \in \{24, 36, 48, 60\}$ and 36 input length for ILI, as well as $h \in \{96, 192, 336, 720\}$ and 96 input length for the remaining.

DLinear Dataset	Horizon	GradNorm	Cov-W	PCgrad	CAGrad	Nash-MTL
ETTm1	96	0.384	0.344	0.348	0.345	0.344
	192	0.463	0.381	0.384	0.382	0.382
	336	0.468	0.416	0.417	0.418	0.418
	720	0.573	0.482	0.494	0.480	0.480
ETTm2	96	0.261	0.184	0.181	0.183	0.188
	192	0.374	0.263	0.249	0.255	0.273
	336	0.458	0.337	0.321	0.335	0.363
	720	0.694	0.482	0.458	0.451	0.477
Exchange	96	0.122	0.080	0.087	0.120	0.120
	192	0.229	0.164	0.157	0.207	0.207
	336	0.508	0.289	0.316	0.340	0.340
	720	0.656	0.712	0.740	0.833	0.833
ILI	24	4.765	2.576	2.414	2.365	2.710
	36	6.774	2.517	2.435	2.336	2.835
	48	2.998	2.528	2.347	2.301	2.682
	60	4.862	2.777	2.576	2.396	2.895
Weather	96	0.203	0.196	0.196	0.196	0.197
	192	0.253	0.237	0.236	0.238	0.238
	336	0.301	0.284	0.286	0.285	0.285
	720	0.361	0.354	0.348	0.348	0.348

Table 11: NLinear: Multivariate forecasting results of Gradient manipulation methods, Each score represents one of four forecast horizons $h \in \{24, 36, 48, 60\}$ and 36 input length for ILI, as well as $h \in \{96, 192, 336, 720\}$ and 96 input length for the remaining.

NLinear Dataset	Horizon	GradNorm	Cov-W	PCgrad	CAGrad	Nash-MTL
ETTm1	96	0.408	0.349	0.354	0.351	0.355
	192	0.409	0.388	0.393	0.396	0.389
	336	0.437	0.421	0.424	0.424	0.423
	720	0.499	0.483	0.484	0.486	0.486
ETTm2	96	0.203	0.182	0.181	0.180	0.183
	192	0.256	0.245	0.245	0.245	0.245
	336	0.318	0.306	0.306	0.305	0.306
	720	0.417	0.407	0.406	0.405	0.407
Exchange	96	0.088	0.088	0.088	0.085	0.085
	192	0.190	0.178	0.170	0.175	0.175
	336	0.356	0.330	0.322	0.319	0.319
	720	1.044	0.922	0.904	0.851	0.851
ILI	24	2.933	2.351	2.327	2.191	2.348
	36	2.257	2.088	2.103	1.907	2.136
	48	2.426	2.040	1.999	1.823	2.072
	60	2.546	2.216	2.170	2.028	2.249
Weather	96	0.247	0.195	0.193	0.195	0.195
	192	0.279	0.241	0.239	0.241	0.241
	336	0.311	0.293	0.292	0.293	0.293
	720	0.388	0.365	0.364	0.365	0.366

F.3 PCA of the time series datasets

The figures 9 and 10 in the subsection present the results of Principal Component Analysis (PCA) applied to a time series dataset. The left subplot in each row displays a 2D scatter plot of the first two principal components, highlighting the variance captured in two dimensions. Each point represents a data sample, colored to indicate different observations.

The right subplot shows a 3D scatter plot of the first three principal components, providing a more comprehensive view of the data's structure. Vector arrows originating from the origin illustrate the direction and magnitude of each principal component, emphasizing the data's spread in three-dimensional space.

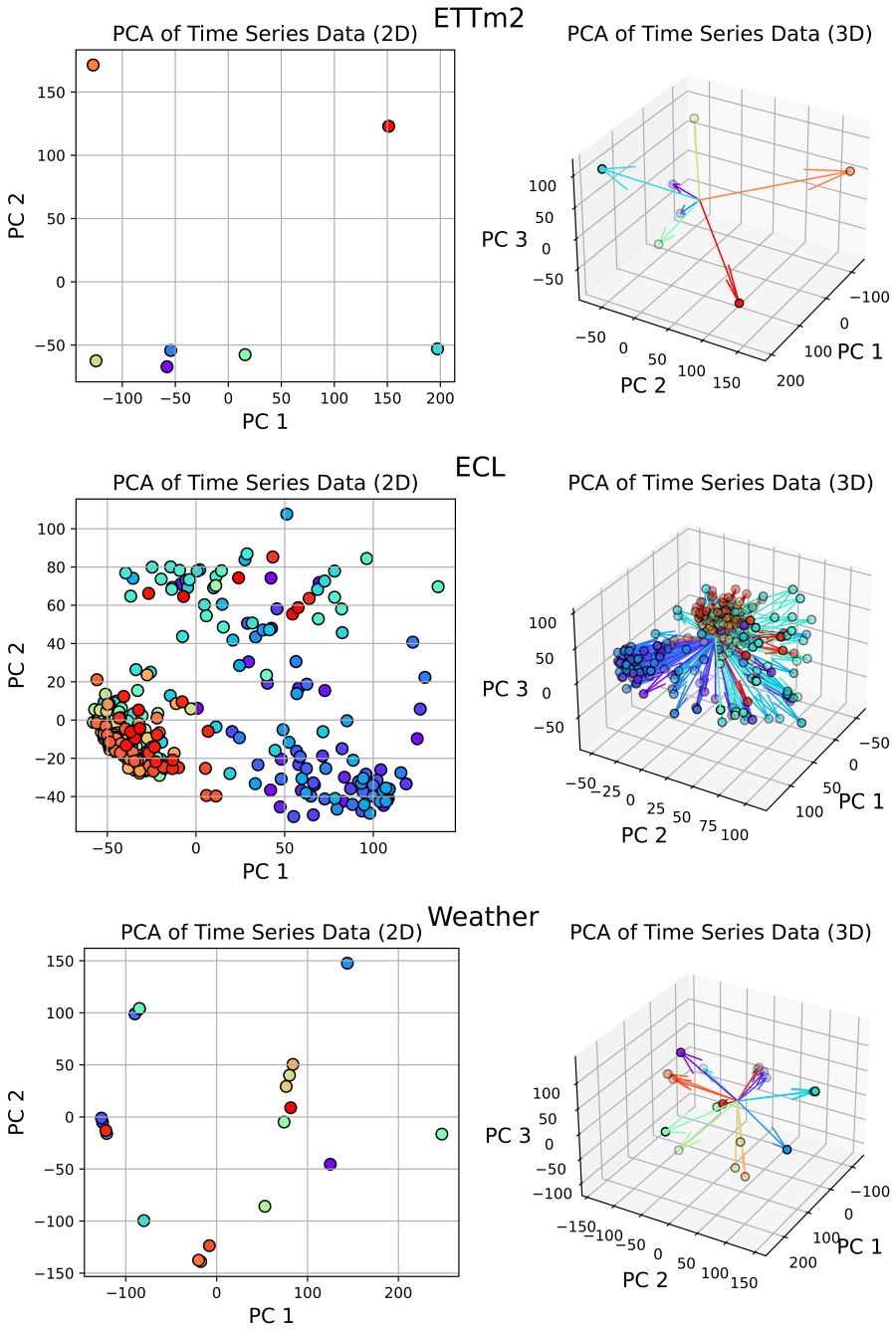

Figure 9: 2D (left) and 3D (right) PCA applied to ETTm2, ECL, and Weather datasets. PCA assists with highlighting variate direction and similarity.

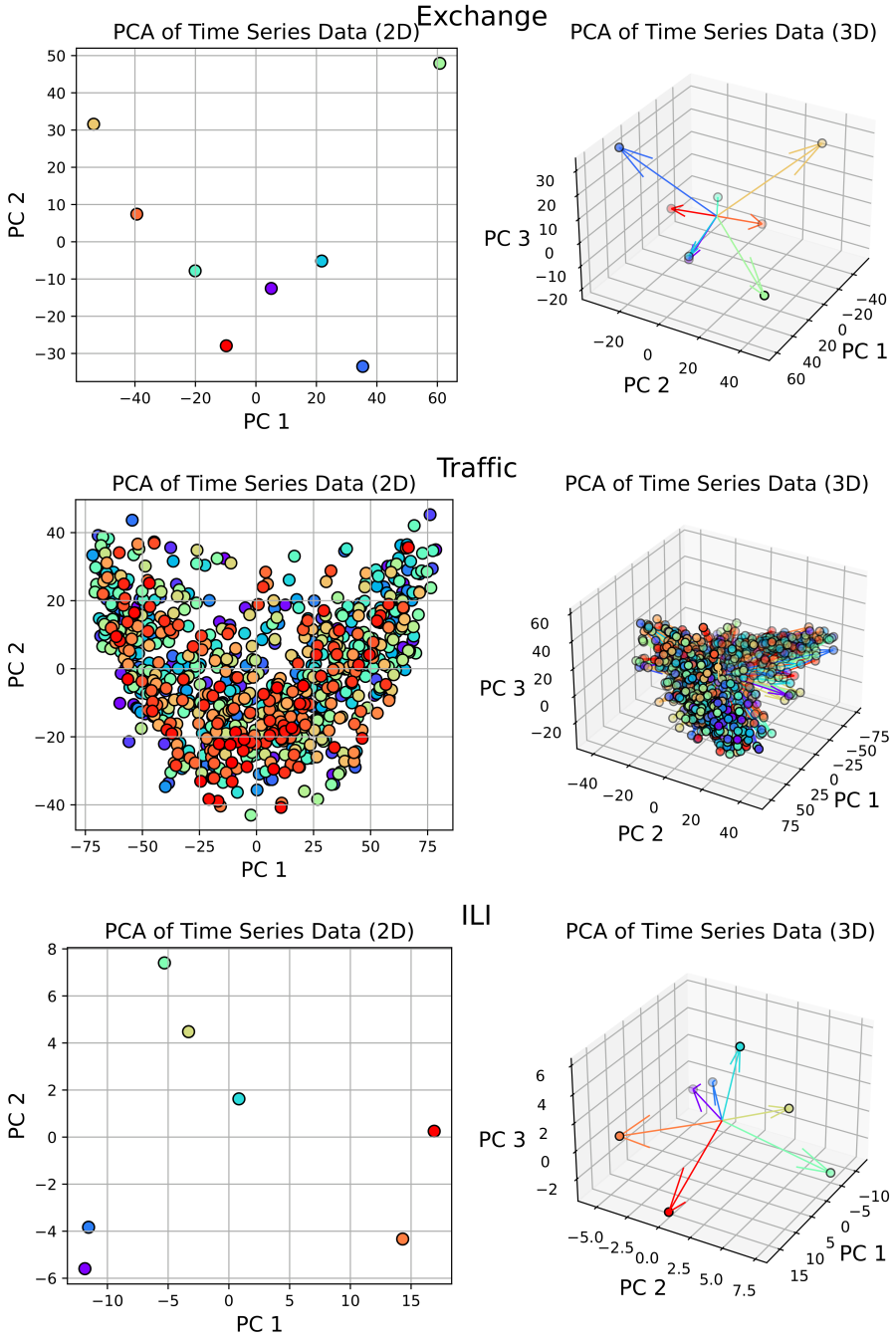

Figure 10: 2D (left) and 3D (right) PCA applied to Exchange, Traffic, and ILI datasets. PCA assists with highlighting variate direction and similarity.

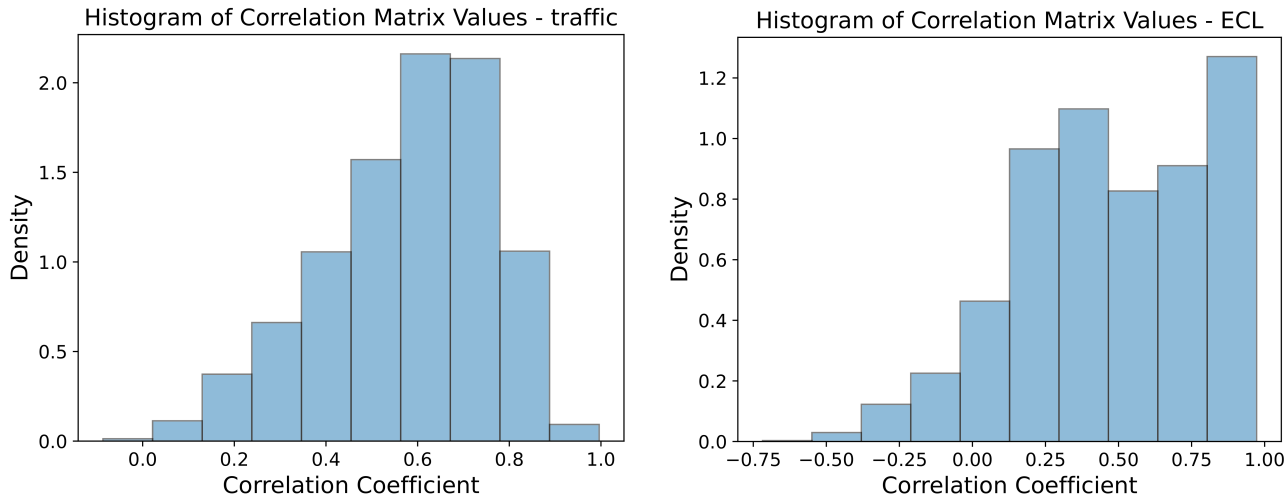

Figure 11: Comparison of the distribution of the correlation coefficient between variates between ECL and Traffic. A skewness closer to 1.0 implies a dataset with many strong linear relationships between variates, therefore, both ECL and Traffic have many variate pairs with a strong linear relationship.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [**Yes**. Mentioned in 2]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [**Yes**. Mentioned in 4 and App. C.4.]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [**No**. The code will be provided upon acceptance.]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [**Yes**. See App. A]
   (b) Complete proofs of all theoretical results. [**Yes**. See App. A]
   (c) Clear explanations of any assumptions. [**Yes**. See App. A]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [**No**. The code will be provided upon acceptance, but all the experimental details are described in App. D.]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [**Yes**. See in App. D.]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [**Yes**. see in App. F]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes. see in App. D.]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [**Yes**. Citation are present for relevant models and datasets.]
   (b) The license information of the assets, if applicable. [**Not Applicable**]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes. See App. D.]
   (d) Information about consent from data providers/curators. [**Not Applicable**]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [**Not Applicable**]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [**Not Applicable**]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [**Not Applicable**]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [**Not Applicable**]