An Analysis of Linear Time Series Forecasting Models

William Toner [1 2] Luke Darlow [2]

Abstract

Despite their simplicity, linear models perform well at time series forecasting, even when pitted against deeper and more expensive models. A number of variations to the linear model have been proposed, often including some form of feature normalisation that improves model generalisation. In this paper we analyse the sets of functions expressible using these linear model architectures. In so doing we show that several popular variants of linear models for time series forecasting are equivalent and functionally indistinguishable from standard, unconstrained linear regression. We characterise the model classes for each linear variant. We demonstrate that each model can be reinterpreted as unconstrained linear regression over a suitably augmented feature set, and therefore admit closed-form solutions when using a mean-squared loss function. We provide experimental evidence that the models under inspection learn nearly identical solutions, and finally demonstrate that the simpler closed form solutions are superior forecasters across 72% of test settings. [3]

1. Introduction

Time series forecasting is a crucial challenge across a wide array of domains, where accurate predictions of the future are essential. Key areas such as finance, meteorology, healthcare, cloud infrastructure, and traffic flow management rely heavily on forecasting for decision-making and strategic planning (Wu et al., 2021; Lai et al., 2018; Sloss et al., 2019; Taylor & Letham, 2018; Darlow et al., 2023; Joosen et al., 2023). This has led to significant research efforts to develop effective forecasting models. Deep learning has transformed many fields, most notably in computer vision and language processing, superseding simpler classical models. Following these successes, deep learning has seen increasing usage in time series forecasting. In particular transformer models have been adapted for broader time series forecasting applications (Nie et al., 2022; Zhou et al., 2021; Liu et al., 2021; Wu et al., 2021; Liu et al., 2023; Darlow et al., 2024).

Linear models for forecasting Despite the advantages offered by deep learning, its application to time series forecasting has encountered unique challenges. Recent studies have shown that the performance benefits of deep models for forecasting are often marginal when compared to simpler linear models (Zeng et al., 2023; Li et al., 2023). Linear models are also appealing due to their simplicity, explainability, and efficiency. This is particularly relevant in industries where forecasting models are queried frequently and/or involve high-resolution data, such as in cloud resource allocation (Joosen et al., 2023; Darlow et al., 2023). This has spurred a growing interest in refining linear models: several variants of linear time series forecasting models have emerged, each purporting superiority owing to some architectural difference (summary in Section 2). We show in this research that many of these popular, and often high-performing, linear models are essentially equivalent. By this we mean that the parametric families of functions which they describe, are equal (up to choice of data normalisation). The convexity of least-squares linear regression makes this a significant finding since it implies that all these models should converge to the same optima, given a suitable optimiser.

Outline and Contributions In this paper, we delve into the mathematics of several well-known linear time series forecasting models. We fully characterise the set of functions which are expressible using each architecture. We show, somewhat remarkably, that they are all essentially equivalent: corresponding either to unconstrained or weakly constrained (via feature augmentation) linear regression. The convexity of least-squares linear regression suggests that the behaviour of these models should therefore be virtually indistinguishable. We provide experimental evidence which supports this hypothesis, showing that, in practice, all models tend to the same optima. Furthermore, we show that the closed form solution to least-squares linear regression performs either comparably or better than those trained by

[1]ANC, Department of Informatics, University of Edinburgh, Edinburgh [2]Systems Infrastructure Research, Huawei Research Centre, Edinburgh. Correspondence to: William Toner <w.j.toner@sms.ed.ac.uk>, Luke Darlow <luke.darlow1@huawei.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[3]github.com/sir-lab/linear-forecasting

gradient descent. Our contributions are:

1. Mathematical proofs that several popular linear models for time series forecasting are essentially identical.
2. Experimental evidence that each model indeed tend to the same solution when trained on the same data, differing only in the bias parameter.
3. Quantitative evidence that closed form ordinary least squares (OLS) solutions are typically superior to existing models trained using stochastic gradient descent.

The goal of this paper is to provide a, much-needed, in-depth mathematical analysis of several popular linear time-series forecasting models. We aim to demonstrate that, from a functional and performance point of view, these models are not substantially different to each other and amount to weakly constrained linear regression.

2. Related Work

Zeng et al. (2023) asked the important question of whether the transformer architecture (Vaswani et al., 2017) had utility for time series forecasting. Their work introduced two models, namely **DLinear** (Section 3.1.1) and **NLinear** (Section 3.5.1), that have become widely used baselines for other research in time series forecasting (Darlow et al., 2024; Nie et al., 2022; Liu et al., 2023). Their work served to show that linear models are comparable, and sometimes better, than complex transformer architectures.

Reversible instance normalisation (Kim et al., 2021) (RevIn) is a feature normalisation technique that typically improves time series forecasting. It operates by standardising input features (zero mean, unit standard deviation) before passing these through a given model, and reversing this standardisation process as a final step (with an optional learnable affine transformation). We unpack the mathematics of how various modes of instance normalisation constrain the underlying model class in Section 3.2.

Li et al. (2023) revisited long-term time series forecasting by exploring the impact of RevIn and channel independence (CI). CI for linear models implies learning distinct models for each variate in a given dataset. They proposed **RLinear** – a linear mapping that uses RevIn – and tested the impact of CI, showing how for some datasets (usually with a higher number of channels and/or complexity) CI improves generalisation. We define RLinear in Section 3.4.1.

Xu et al. (2023) recently proposed **FITS**, a linear time series model that operates in frequency space and includes an optional high-frequency filtering component to reduce the model footprint. FITS first computes the real discrete Fourier transform (RFT), applies a complex linear map, and inverts the result back into the time domain. We define FITS in Section 3.1.2. The performance of FITS is impressive, obtaining at or near state of the art (SoTA) under its optimal hyperparameter settings.

3. Analysis of Linear Time Series Forecasting Models

For the purpose of this paper we refer to a 'model class' as the parametric set of functions induced by a model architecture. For example, a single layer linear neural network with no hidden layer has the model class $\vec{x} \mapsto A\vec{x} + \vec{b}$, where the dimensions of A and $\vec{b}$ are as appropriate. We call this '**Linear**' for the remainder of this paper. In this section we define the task of forecasting with a linear model. We then analyse the widely used DLinear (Section 3.1.1) and the recent SoTA FITS architectures (Section 3.1.2). We prove mathematically that these models are equivalent to linear regression in that they have the same model class.

We then define and discuss several invertible data normalisation strategies employed for time series forecasting in Section 3.2. These normalisation strategies yield additional linear model variants, namely RLinear, NLinear, and FITS+IN (i.e., FITS with instance normalisation, as per Xu et al. (2023)). We show how each choice of feature normalisation restricts the model class. This allows us to categorise all linear model variants into only 3 similar but distinct classes.

3.1. Notation

The following notations are used throughout this paper:

- L: Context length (time steps in the input sequence).
- c: Number of channels (distinct time series).
- T: Forecast horizon (future time steps predicted).
- $\vec{x}$: Context vector (historical data), $\vec{x} \in \mathbb{R}^{L\times c}$.
- $\vec{y}$: Target vector (values to be predicted), $\vec{y} \in \mathbb{R}^{T\times c}$.

The models we look at in Section 3 do not explicitly use cross-channel information in their predictions. By this we mean that the i^{th} channel of the target is predicted only using the i^{th} channel of the context. Therefore, for improved clarity, we consider the case of $c = 1$ (univariate), with $\vec{x} \in \mathbb{R}^L$.

A **Forecast Model** is a function $f : \mathbb{R}^L \to \mathbb{R}^T$ that generates a forecast vector $\vec{y}$ from a given context vector $\vec{x}$. A **Parametric Model** is a family of forecast models; $f(x; \theta)$ indexed by a parameter $\theta \in \Theta$. [4] The set of functions obtainable by different parameter settings, $\{f(x; \theta)|\theta \in \Theta\}$, is called the **Model Class** which we denote with an $\mathcal{M}(\cdot)$.

[4] Θ is some set, typically $\mathbb{R}^M$.

For example, the model class for **unconstrained Linear Regression**, denoted as $\mathcal{M}(\text{Linear})$, includes all functions of the form $\vec{x} \mapsto W\vec{x} + \vec{b}$, where W is any weight matrix from $\mathbb{R}^{T\times L}$ and $\vec{b}$ is any bias vector from $\mathbb{R}^T$. If specific conditions are imposed on W and $\vec{b}$, the resulting set is termed **constrained Linear Regression**.

3.1.1. DLinear

DLinear Model Definition: Let $\vec{x} \in \mathbb{R}^L$ be a context vector. DLinear works by decomposing $\vec{x}$ into a 'trend' and 'seasonal' components. The trend component is defined by taking a moving average of the components of $\vec{x}$. The seasonal component is given by the residual $\vec{x}_{\text{seasonal}} := \vec{x} - \vec{x}_{\text{trend}}$. The moving average is padded so that it preserves the dimensionality of x. One then takes $\vec{x}_{\text{trend}}$ and $\vec{x}_{\text{seasonal}}$ and passes these though separate learnable linear layers.

Lemma 3.1 (DLinear Model Class)**.** *Let $\mathcal{M}(DLinear)$ denote the DLinear model class, i.e. the set of functions $f : \mathbb{R}^L \to \mathbb{R}^T$ which can be represented as a DLinear model. $\mathcal{M}(DLinear)$ is precisely equal to the space of affine linear functions. That is, all functions of the form $A\vec{x} + \vec{b}$ may be expressed as a DLinear model and vice versa.*

Proof. Following our definition, any DLinear model can be written as $B\vec{x}_{\text{seasonal}} + C\vec{x}_{\text{trend}} + \vec{c} + \vec{d}$ where $B, C \in \mathbb{R}^{T\times L}$, $\vec{c}, \vec{d} \in \mathbb{R}$ are the weight matrices and biases of DLinear's two linear layers. This can be expressed as $B(\vec{x} - \vec{x}_{\text{trend}}) + C(\vec{x}_{\text{trend}}) + \vec{c} + \vec{d} = B(\vec{x} - D\vec{x}) + C(D\vec{x}) + \vec{c} + \vec{d} = (B - BD + CD)\vec{x} + \vec{c} + \vec{d}$ where D is the (square) matrix corresponding to a padded moving average (See Appendix D for an explanation). Thus we have shown that any DLinear model may be expressed in the form $A\vec{x} + \vec{b}$. It remains to show the converse, that is, any affine linear map is expressible in the form of a DLinear model.

Let $A\vec{x} + \vec{b}$ be some arbitrary affine linear map. We claim that $A\vec{x} + \vec{b}$ can be expressed in the form $(B - BD + CD)\vec{x} + \vec{c} + \vec{d}$. By setting e.g. $\vec{c} = \vec{b}$, $\vec{d} = 0$ we match the bias terms. By setting $B = C = A$ we match the weight matrices □

$\mathcal{M}$(DLinear) = $\mathcal{M}$(Linear)

3.1.2. FITS

FITS Model Definition: Let $\vec{x} \in \mathbb{R}^L$ be a context vector. FITS applies the Real (discrete) Fourier Transform (RFT) to $\vec{x}$. This maps $\vec{x}$ to a complex vector of length $\lfloor L/2 \rfloor + 1$. Next one applies a learnable complex linear map with output dimension $\lfloor (L + T)/2 \rfloor + 1$. After this one applies the inverse RFT to map to $\mathbb{R}^{L+T}$.

Remark: As proposed by Xu et al. (2023), FITS optionally includes a low-pass filter (LPF) to discard high frequencies components. Our initial experiments showed that utilising a LPF results in a degradation in performance – this is confirmed by analysing the settings of FITS that yield high-performance. Thus, we analyse FITS without the LPF.

Remark: Unlike other models, FITS outputs both a forecast and a reconstruction of the context vector. The forecast may be obtained by discarding the first L components output by the model.

Theorem 3.2 (FITS Model Class)**.** *Let $\mathcal{M}(FITS)$ denote the FITS model class, i.e. the set of functions $f : \mathbb{R}^L \to \mathbb{R}^T$ which can be represented as a FITS model. When $L \geq T - 2$, $\mathcal{M}(FITS)$ is precisely equal to the space of affine linear functions $A\vec{x} + \vec{b}$.*

Proving Theorem 3.2 is somewhat involved. Importantly, as a combination of a Fourier transform, a complex linear map, an inverse Fourier transform, FITS is a composition of linear maps and is therefore expressible in the form $A\vec{x} + \vec{b}$. The proof in Appendix A.1 shows when $L \geq T - 2$ that both A and $\vec{b}$ are entirely unconstrained. This is significant since all the settings in Xu et al. (2023) utilise a context larger or equal to the prediction horizon T.

$\mathcal{M}$(DLinear) = $\mathcal{M}$(Linear) = $\mathcal{M}$(FITS)
(when $L \geq T - 2$ and without a low-pass filter)

3.2. Invertible Data Normalisations

Invertible instance-wise feature normalisation has been recently adopted for time-series forecasting. 'Instance normalisation' (in the context of time series) was proposed by Kim et al. (2021). In this section we cover three such mechanisms: Instance Norm (IN), Reversible Instance Norm (RevIN), and *NowNorm* (NN) which is the name we give to the normalisation scheme implemented by NLinear. For clarity, RevIN and IN are identical except for the learnable affine mapping of RevIN – we mark this distinction because the optional learnable affine map is often not used (e.g., FITS). We look at how each normalisation restricts the model class when used in conjunction with linear models.

3.3. Instance Norm

Definition 3.3 (Instance normalisation)**.** Given a context vector $\vec{x}$ and a target vector $\vec{y}$, **instance normalisation** (IN) for each data instance involves normalizing $\vec{x}$ by its mean $\mu(\vec{x})$ and standard deviation $\sigma(\vec{x})$, applying a model f on the normalized $\vec{x}'$, and inversely transforming the prediction $\hat{y}$ back to the original scale. Formally, this is expressed as:

$$\begin{aligned} \vec{x}' &= \frac{\vec{x} - \mu(\vec{x})}{\sigma(\vec{x}) + \varepsilon}, \\ \hat{y} &= f(\vec{x}'), \\ \hat{y}_{\text{out}} &= \hat{y} \cdot (\sigma(\vec{x}) + \epsilon) + \mu(\vec{x}), \end{aligned}$$

where ε is a small constant for numerical stability.

Lemma 3.4 (Linear+IN)**.** *Let $\mathcal{M}(ILinear)$ represent the set of forecast models that can be expressed as a linear layer combined with instance normalization (Definition 3.3). $\mathcal{M}(ILinear)$ is equal to the set of functions $f : \mathbb{R}^L \to \mathbb{R}^T$ expressible in the form $\tilde{A}\vec{x} + \vec{b}\sigma(\vec{x})$. $\tilde{A}$ is a matrix with each row summing to 1, and $\sigma(\vec{x})$ is the standard deviation of $\vec{x}$.*

Proof. Let $\vec{x} \in \mathbb{R}^L$ be a context vector. Let f be a forecast model obtained by applying a linear layer after IN. If $A, \vec{b}$ are the weight matrix and bias of the linear layer then we have $f(\vec{x}) = \vec{\mu}(\vec{x}) + \sigma(\vec{x})(A(\frac{\vec{x}-\vec{\mu}(\vec{x})}{\sigma(\vec{x})}) + \vec{b})$. Here $\vec{x} - \vec{\mu}(\vec{x})$ denotes the subtraction of the mean $\mu(\vec{x})$ from every component of $\vec{x}$. We can expand this expression out to obtain simply $\vec{\mu}(\vec{x}) + A(\vec{x} - \vec{\mu}(\vec{x})) + \sigma(\vec{x})\vec{b}$. The $T-$dimensional vector $\vec{\mu}(\vec{x})$ of means can be written as a matrix multiplication $B_T\vec{x}$ where B_m denotes a matrix of shape $m \times L$ populated exclusively by $\frac{1}{L}$'s. Using this notation we have:

$$\begin{aligned} f(\vec{x}) =& \vec{\mu}(\vec{x}) + A(\vec{x} - \vec{\mu}(\vec{x})) + \sigma(\vec{x})\vec{b} \\ =& (B_T + A - AB_L)\vec{x} + \sigma(\vec{x})\vec{b} \end{aligned}$$

Thus f can be written in the form $\tilde{A}\vec{x} + b\sigma(\vec{x})$, it remains only to demonstrate that $B_T + A - AB_L$ satisfies the condition that the rows sum to one. And, conversely that any matrix of shape $T \times L$ whose rows sum to one can be written in this form. Begin by noting that AB_L can be written as: .

$$AB_L = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1L} \\ A_{21} & A_{22} & \dots & A_{2L} \\ & \dots\dots\dots & & \\ A_{L1} & A_{L2} & \dots & A_{TL} \end{bmatrix} \begin{bmatrix} 1/L & 1/L & \dots & 1/L \\ 1/L & 1/L & \dots & 1/L \\ & \dots\dots\dots & & \\ 1/L & 1/L & \dots & 1/L \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{L}\sum_{i=1}^{L} A_{1i} & \frac{1}{L}\sum_{i=1}^{L} A_{1i} & \dots & \frac{1}{L}\sum_{i=1}^{L} A_{1i} \\ \frac{1}{L}\sum_{i=1}^{L} A_{2i} & \frac{1}{L}\sum_{i=1}^{L} A_{2i} & \dots & \frac{1}{L}\sum_{i=1}^{L} A_{2i} \\ & \dots\dots\dots\dots\dots\dots & & \\ \frac{1}{L}\sum_{i=1}^{L} A_{Li} & \frac{1}{L}\sum_{i=1}^{L} A_{Li} & \dots & \frac{1}{L}\sum_{i=1}^{L} A_{Li} \end{bmatrix}$$

Therefore the ijth element of $(B_T + A - AB_L)$ may be written as $\frac{1}{L} + A_{ij} - \frac{1}{L}\sum_{k=1}^{L} A_{ik}$. It follows that the sum of row i of $(B_T + A - AB_L)$

$$\begin{aligned} &= \sum_{j=1} \Big(\frac{1}{L} + A_{ij} - \frac{1}{L}\sum_{k=1}^{L} A_{ik}\Big) \\ &= 1 + \sum_{j=1}^{L} A_{ij} - \frac{L}{L}\sum_{k=1}^{L} A_{ik} \\ &= 1. \end{aligned}$$

Conversely we wish to show that any T by L matrix C whose rows sum to one can be written in the form $(B_T + A - AB_L)$. Let C be such a matrix. One may easily show that in fact C may be expressed as $C = B_T + C - CB_L$, thus we may let $A = C$. □

3.4. Reversible Instance Normalisation

A second more general form of data normalisation is known as Reversible Instance Norm (RevIN) (Kim et al., 2021). This normalisation is designed to allow a forecasting model to handle shifts in the temporal distribution over time. Li et al. (2023) showed that a simple linear model using RevIN is able to outperform most deep models on standard datasets.

Definition 3.5 (Reversible Instance Normalisation)**.** Given a context vector $\vec{x}$ and a target vector $\vec{y}$, **Reversible Instance Normalisation** (RevIN) for each data instance involves a two-step normalization process. First, $\vec{x}$ is normalized by its mean $\mu(\vec{x})$ and standard deviation $\sigma(\vec{x})$. Subsequently, an affine transformation with parameters α and β is applied, followed by the application of a forecasting model f on the transformed $\vec{x}'$. The process is then reversed to retrieve the prediction in the original scale. Formally, this is expressed:

$$\begin{aligned} \vec{x}' &= \frac{\vec{x} - \mu(\vec{x})}{\sigma(\vec{x}) + \varepsilon}, \\ \vec{x}'' &= \frac{\vec{x}' - \beta}{\alpha}, \\ \hat{y} &= f(\vec{x}''), \\ \hat{y}' &= \alpha\hat{y} + \beta, \\ \hat{y}_{\text{out}} &= \hat{y}' \cdot (\sigma(\vec{x}) + \varepsilon) + \mu(\vec{x}). \end{aligned}$$

3.4.1. RLinear

RLinear is a linear model using RevIN (Li et al., 2023).

Lemma 3.6 (RLinear)**.** *Let $\mathcal{M}(RLinear)$ denote the RLinear model class, i.e. the set of functions $f : \mathbb{R}^L \to \mathbb{R}^T$ which can be represented as an RLinear model. $\mathcal{M}(RLinear)$ is precisely equal to the space of functions $\tilde{A}\vec{x} + \vec{b}\sigma(\vec{x})$ where the rows of $\tilde{A}$ each sum to 1 and where $\sigma(\vec{x})$ denotes the standard deviation of the context vector $\vec{x}$.*

Proof. Let f be arbitrary RLinear model (i.e., a forecast model obtainable by the composition of a linear layer and reversible instance norm). If $A, \vec{c}$ are the weight matrix and bias of the linear layer then

$$f(\vec{x}) = \mu(\vec{x}) + \sigma(\vec{x})\left(\beta + \alpha(AR(\vec{x}) + c)\right)$$
$$\text{where } R(\vec{x}) := \frac{1}{\alpha}\left(\frac{\vec{x} - \mu(\vec{x})}{\sigma(\vec{x})} - \beta\right)$$

We can expand this out to obtain the following .

$$f(\vec{x}) = (\mu(\vec{x}) + A\vec{x} - A\mu(\vec{x})) + \beta\sigma(\vec{x}) + \alpha c\sigma(\vec{x}) - A\beta\sigma(\vec{x}).$$

As per the proof on Lemma 3.4 we can write the vector of means $\mu(\vec{x})$ as $B_T\vec{x}$ where B_T is a $T \times L$ matrix populated by $\frac{1}{L}$'s. Therefore $f(\vec{x})$ can be expressed as

$$\begin{aligned} f(\vec{x}) =& \tilde{A}\vec{x} + \sigma(\vec{x})\vec{b} \\ \text{where } \tilde{A} =& B_T + A - AB_L \\ \text{and } \vec{b} =& \beta + \alpha c - A\beta \end{aligned}$$

As in the proof of Lemma 3.4, $B_T + A - AB_L$ are precisely the set of matrices where each row sums to one. It is left therefore to demonstrate that $\vec{b}$ can be any vector in $\mathbb{R}^T$. Since we are free in our choice of β, α, c then we can let $\beta = 0, \alpha = 1$ and $\vec{c}$ be any desired arbitrary vector in $\mathbb{R}^T$. This concludes the proof. □

IN and **RevIN** impose the constraints: (1) the rows of the weight matrix must sum to 1; (2) the bias is scaled by the standard deviation of the instance.

3.5. NowNorm

Definition 3.7 (Now-Normalisation)**.** Given a context vector $\vec{x}$ and a target vector $\vec{y}$, **NowNorm** (NN) involves normalising the context so that x_L, the most-recent value of $\vec{x}$, is zero. Explicitly; $\vec{x}_{\text{norm}} := \vec{x} - (x_L, x_L, \ldots, x_L)$. Next we apply a forecasting model f on the normalized $\vec{x}_{\text{norm}}$, before adding x_L back on to each component of the output. Formally, this is expressed as:

$$\begin{aligned}\vec{x}_{\text{norm}} &= \vec{x} - x_L,\\ \hat{y} &= f(\vec{x}_{\text{norm}}),\\ \hat{\vec{y}}_{\text{out}} &= \hat{y} + x_L.\end{aligned}$$

3.5.1. NLinear

Nlinear is a linear model using NN (Zeng et al., 2023).

Lemma 3.8 (NLinear)**.** *Let $\mathcal{M}$(NLinear) denote the NLinear model class, i.e. the set of functions $f : \mathbb{R}^L \to \mathbb{R}^T$ which can be represented as a NLinear model. $\mathcal{M}$(NLinear) is precisely equal to the space of linear functions $\tilde{A}\vec{x} + \vec{b}$ where the rows of $\tilde{A}$ each sum to 1.*

Proof. By Definition 3.7, an NLinear model can be written

$$(x_L, x_L, \ldots, x_L) + A\vec{x}_{\text{norm}} + \vec{b} \tag{1}$$

Where $A, \vec{b}$ are the weight matrix and bias terms of NLinear's linear layer and $\vec{x}_{\text{norm}}$ is the normalised context vector. If we let B_m denote an m by m matrix with 1's in the final column and zeros elsewhere. Then we can write Equation 1 equivalently as $B_T\vec{x} + A(\vec{x} - B_L\vec{x}) + \vec{b} = (B_T + A - AB_L)\vec{x} + \vec{b}$. We claim that the rows of the matrix $B_T + A - AB_L$ sum to one. Begin by noting that AB_L has the following form: .

$$AB_L = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1L} \\ A_{21} & A_{22} & \ldots & A_{2L} \\ & \ldots\ldots\ldots & & \\ A_{L1} & A_{L2} & \ldots & A_{TL} \end{bmatrix} \begin{bmatrix} 0 & 0 & \ldots & 1 \\ 0 & 0 & \ldots & 1 \\ & \ldots & & \\ 0 & 0 & \ldots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & \ldots & \sum_{i=1}^{L} A_{1i} \\ 0 & 0 & \ldots & \sum_{i=1}^{L} A_{2i} \\ & \ldots\ldots\ldots & & \\ 0 & 0 & \ldots & \sum_{i=1}^{L} A_{Li} \end{bmatrix}$$

Therefore $B_T + A - AB_L$ may be expressed as follows: .

$$\begin{bmatrix} A_{11} & A_{12} & \ldots & 1-\sum_{i=1}^{L-1} A_{1i} \\ A_{21} & A_{22} & \ldots & 1-\sum_{i=1}^{L-1} A_{2i} \\ & \ldots\ldots\ldots\ldots & & \\ A_{L1} & A_{L2} & \ldots & 1-\sum_{i=1}^{L-1} A_{Li} \end{bmatrix} \tag{2}$$

It is clear to see that the rows of this matrix sum to one as claimed. Moreover, any matrix whose rows sum to 1 may be written as Equation 2. Since the bias $\vec{b}$ is unconstrained then we conclude our proof. □

NowNorm imposes the same weight matrix constraint as **IN** and **RevIN**, but does not constrain the bias.

Integrating the insights from Lemma 3.1 and Theorem 3.2 with the analyses presented in this subsection, we establish the following equivalences among the model classes:

$$\mathcal{M}(\textbf{DLinear+IN}) = \mathcal{M}(\textbf{Linear+IN}) = \mathcal{M}(\textbf{FITS+IN}) = \mathcal{M}(\textbf{RLinear}) \approx \mathcal{M}(\textbf{NLinear})$$

4. Discussion

Model Class	Variants	Normalisation	Constraints
$A\vec{x} + \vec{b}$	Linear, DLinear, FITS	None	None
$\tilde{A}\vec{x} + \vec{b}$	NLinear	NowNorm	Rows sum to one
$\tilde{A}\vec{x} + \vec{b}\sigma(\vec{x})$	RLinear	RevIn	Rows sum to one Bias coupled with σ
	FITS+IN	Instance	

Table 1. A summary of the model classes for the DLinear, FITS, RLinear, NLinear and Linear models. Here $\tilde{A}$ denotes a matrix whose rows must each sum to one and $\sigma(\vec{x})$ denotes the standard deviation of the components of the context vector.

Our analysis is summarised in Table 1. When $L \geq T - 2$ FITS and DLinear are functionally equivalent to unconstrained linear regression. In Section 3.2 we looked at the model classes for linear models which use one of the standard normalisation procedures for time series analysis. We saw how using normalisation slightly alters the model class. For example, NLinear is equivalent to restricted linear regression wherein the rows of the weight matrix must sum to 1. We showed that Linear+IN, Linear+RevIN (RLinear (Li et al., 2023)), and FITS+IN (i.e., the setting in (Xu et al., 2023)) are equivalent to each other, and differ from NLinear in that the bias is parameterised as $\vec{b}\sigma(\vec{x})$. Perhaps most importantly, each model class can be reformulated as unconstrained linear regression on an augmented feature set, and are solvable in closed form.

Convexity Each of the models we have discussed train using a mean-squared error (MSE) loss function (Xu et al., 2023; Zeng et al., 2023). Linear regression with a mean-squared loss function is a convex optimisation problem. By this we mean that the training loss is a convex function of the parameters. A consequence of convexity is that there exists a unique global optima which minimises the training loss (uniqueness requires that the training data is full rank). Significantly, this means that given the same training data, these models should converge to the same solution, via a suitable optimisation procedure.

Closed Form An important property of least-squares linear regression is that it admits a closed-form solution. A recap of how one computes a closed-form solution for linear regression and closed form solutions for the three model classes in Table 1 may be found in Appendix D.2. In Section 5 we refer to the closed form solutions as the ordinary least-squares (OLS) models, and we will determine how each model fairs against this closed form approach.

Remark: FITS has two separate training modes. In mode 1 the model is trained by mean-squared error (MSE) between the forecast and the target. In mode 2 an additional term is added to the loss which is the MSE between the context vector and FITS reconstruction. Empirically both settings have similar performance (Xu et al., 2023). In our analysis and experiments we consider mode 1 only.

For each model class in Table 1, the least-squares optima may be found in **closed form**.

We have hypothesised that the convexity of least-squares linear regression means that each model should converge to the same solution given the same data. Nevertheless, given that each model architecture yields a different parameterisation and initialisation, this still leaves open the possibility that early stopping may impact generalisation. Next, we explore how the parameterisation of FITS has the effect of inducing a much lower learning rate on the bias term compared to that of the weight matrix.

4.1. The FITS Bias-Term

In Theorem 3.2 we showed that any FITS model can be written in the form $A\vec{x} + \vec{b}$. Moreover, we showed how one may obtain $A, \vec{b}$ given the weight matrix and the bias of the complex linear layer (Appendix A.7). Specifically, if $\vec{c}$ denotes the complex bias of the complex linear layer then $\vec{b} = iRFT(\vec{c})$ where iRFT denotes the inverse discrete Fourier transform (Definition A.2). It is important to consider what the implications are of parameterising the bias term in this way, rather than simply parameterising $\vec{b}$ directly.

Consider representing the complex vector $\vec{c}$, of dimension $\frac{L+T}{2}$, as a $(T + L)$-dimensional real vector. This is achieved by separating the real and imaginary components of $\vec{c}$. Given that the iRFT is a linear mapping, it follows that $\vec{b}$ and $\vec{c}$ are interconnected through the equation $\vec{b} = M\vec{c}$, where M is a specific matrix derived from the iRFT. Critically, the matrix M plays a pivotal role in determining the effective learning rate for the bias in our linear model. For instance, small values within M imply that adjustments in $\vec{c}$ induce only minor changes in $\vec{b}$. Due the choice of normalisation used for the RFT, the entries of the matrix M are of the order $\sim \frac{1}{\sqrt{L}}$ and FITS manifests this exact phenomena. A detailed breakdown of this may be found in Appendix C.

5. Experiments

In Section 5.1 we demonstrate that the models discussed in this paper tend toward their corresponding closed form solutions. In Section 5.2 we test and compare each model across 8 benchmarking datasets, and show how the **closed form solution is usually superior**.

5.1. Convergence

Comparison of Learned Weight Matrices Figure 1 visualises the internal weight matrices for 4 trained linear model variants plus the closed-form solution (denoted **OLS+IN**). The models shown are RLinear, NLinear, DLinear+IN, FITS+IN (the SoTA variant of FITS from (Xu et al., 2023)), and OLS+IN. Each model is trained for 50 epochs[5] on the ETTh1 dataset with a context of 720 and a prediction horizon length of 336. The weight matrices are then extracted and visualised using the same colour scale. In all cases the learned matrices are near identical. The similarity of the weight matrices for Linear+IN, RLinear, FITS+IN and OLS+IN is precisely in line with our hypothesis and matches the theory and discussion from previous sections. Note that while NLinear lies is a slightly different model class (See Table 1), the learned matrix is still near identical.

Plotting the learned matrices as in Figure 1 requires us to first convert each trained model into the form $f(\vec{x}) = A\vec{x} + \vec{b}$. To do this we note that $f(\vec{0}) = A\vec{0} + \vec{b} = \vec{b}$. Thus, the bias can be found by passing the zero vector into the trained model. We can determine A in a similar manner. Let $\vec{e_i}$ denote the i^{th} coordinate vector, that is $\vec{e_i}$ is the vector which is 1 at position i and zero elsewhere. Then $f(\vec{e_i}) = A\vec{e_i} + \vec{b} = A_{\cdot,i} + \vec{b}$ where $A_{\cdot,i}$ is the i^{th} column of A. Hence, given that we have already computed the bias term, we may derive A simply by passing through each coordinate vector $\vec{e_i}$ and subtracting $\vec{b}$. The procedure for extracting the weight matrices for models of the form $A\vec{x} + \vec{b}\sigma(x)$ is similar and is discussed in Appendix G.1.

[5] except for OLS which is solved using an SVD solver in Scikit-learn (Pedregosa et al., 2011).

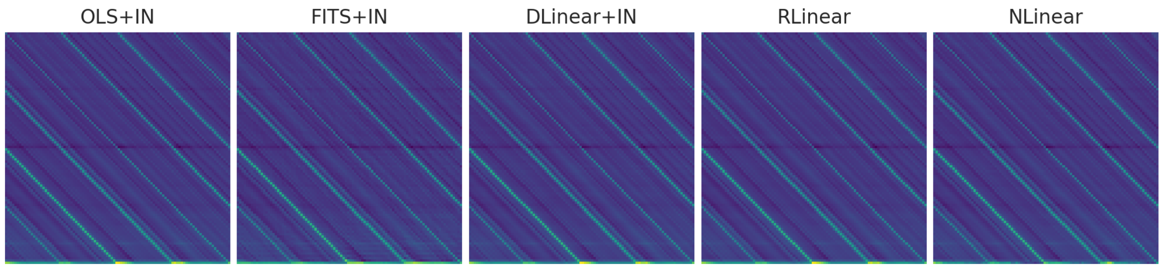

Figure 1. This figure displays the cropped weight matrices after 50 epochs of training for all four models with instance normalization, juxtaposed with their corresponding closed-form solution (extreme left). These show how similar the underlying models are. There are slight differences that affect forecasts to a marginal degree (see Figure 3).

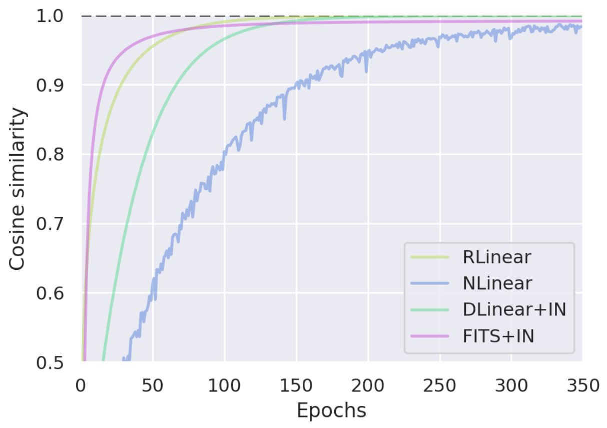

Figure 2. A demonstration of how the model's weight matrices tend to the OLS solution during training. This is a visualised as the cosine similarity between a given model's weight matrix and that determined by the closed form solution.

Cosine similarity during training Figure 2 tracks the cosine similarity (Defined $d(x, y) := \frac{x \cdot y}{||x||_2 \cdot ||y||_2}$) between the above-mentioned 4 models' weight matrices and their OLS counterpart during training. A cosine similarity of one, corresponds to exact equality between the matrices. In line with our hypothesis, all model's weight converge toward the OLS solution. The rapidity of this behaviour differs per model, thus demonstrating that SGD optimisation coupled with each unique parameterisation impacts the particularly route taken and rate of convergence.

Forecasts Figure 3 shows the forecasts from these models after 50 epochs of training. While subtle differences in the models do indeed result in subtle differences in forecasts, there is clear and pervasive similarity between forecasts.

Bias Terms The bias terms for each trained model are visualised in Figure 4. As expected, the models DLinear+IN, RLinear and OLS+IN learn the same bias terms as each other. Notably however, the bias for FITS+IN differs considerably from the other models. Moreover the magnitude of this bias is much smaller. This difference is despite the fact that all these models' classes are equivalent (Table 1). This confirms our analysis from Section 4.1.

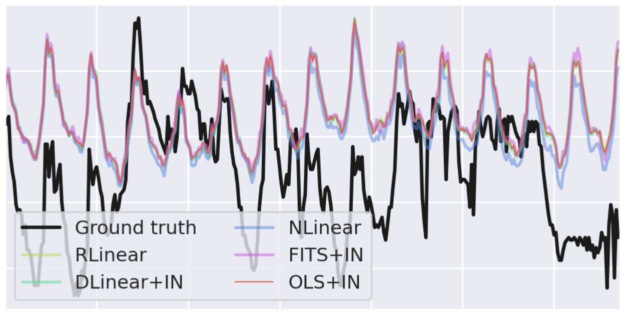

Figure 3. Forecast comparison on ETTh1 with $T = 336$, comparing the 5 models that use instance normalisation.

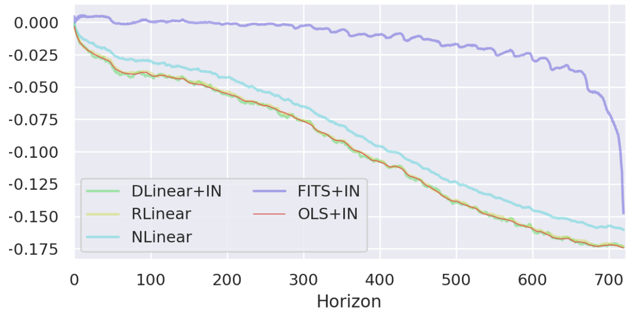

Figure 4. The components of the learned bias vectors ($\vec{b} \in \mathbb{R}^{720}$) plotted for several linear models implementing feature normalisation technique. FITS results clearly in a different bias term.

5.2. Performance

Table 2 presents the Mean Squared Error (MSE) values, accompanied by error bars, for the models evaluated in this study, both with and without instance normalization.[6] In the table, **green** highlighting signifies instances where the corresponding sOrdinary Least Squares (OLS) solution achieves a lower MSE compared to the model being evalu-

[6]We included NLinear in the grouping 'with' instance normalisation, even though the model classes is slightly different.

Table 2. Long-term multivariate forecasting results, showing MSE values for all models investigated in this work. The **green** and **blue** highlighting indicate when the OLS is superior and within 1 standard deviation of a given model, respectively. **Bold**ing indicates the best performing model for a given dataset-horizon combination.

		Methods **without** instance normalisation				Methods **with** instance normalisation				
	T	**OLS**	FITS	DLinear	Linear	**OLS+IN**	FITS+IN	DLinear+IN	RLinear	NLinear
ETTm1	96	**0.306**	0.310 ±0.0005	0.311 ±0.0008	0.314 ±0.0037	0.307	0.309 ±0.0002	0.312 ±0.0008	0.312 ±0.0024	0.319 ±0.0021
	192	**0.335**	0.338 ±0.0008	0.342 ±0.0014	0.343 ±0.0012	0.336	0.338 ±0.0005	0.341 ±0.0014	0.343 ±0.0010	0.346 ±0.0009
	336	**0.364**	0.367 ±0.0003	0.372 ±0.0006	0.374 ±0.0008	0.365	0.367 ±0.0001	0.372 ±0.0006	0.372 ±0.0016	0.378 ±0.0003
	720	**0.413**	0.435 ±0.0010	0.422 ±0.0016	0.426 ±0.0058	0.415	0.417 ±0.0006	0.422 ±0.0016	0.421 ±0.0018	0.424 ±0.0029
ETTm2	96	0.166	0.165 ±0.0003	0.164 ±0.0017	0.163 ±0.0010	**0.162**	**0.162** ±0.0001	0.163 ±0.0011	0.164 ±0.0009	0.164 ±0.0009
	192	0.228	0.225 ±0.0001	0.222 ±0.0023	0.218 ±0.0013	**0.216**	0.217 ±0.0001	0.217 ±0.0004	0.217 ±0.0007	0.217 ±0.0007
	336	0.295	0.291 ±0.0008	**0.267** ±0.0029	0.272 ±0.0021	0.268	0.269 ±0.0000	0.269 ±0.0007	0.270 ±0.0011	0.270 ±0.0011
	720	0.415	0.409 ±0.0004	0.356 ±0.0056	0.362 ±0.0073	**0.349**	0.350 ±0.0002	0.354 ±0.0016	0.354 ±0.0010	0.355 ±0.0010
ETTh1	96	0.376	0.378 ±0.0002	0.380 ±0.0027	0.390 ±0.0016	**0.375**	0.377 ±0.0002	0.379 ±0.0010	0.387 ±0.0006	0.383 ±0.0027
	192	**0.413**	**0.413** ±0.0002	0.424 ±0.0045	0.426 ±0.0029	**0.413**	**0.413** ±0.0002	0.419 ±0.0018	0.415 ±0.0015	0.418 ±0.0016
	336	0.448	0.500 ±0.0014	0.458±0.0104	0.465 ±0.0044	0.445	**0.432** ±0.0008	0.451 ±0.0020	0.450 ±0.0007	0.446 ±0.0006
	720	0.491	0.506 ±0.0062	0.522 ±0.0051	0.512 ±0.0017	0.460	**0.428** ±0.0002	0.470 ±0.0013	0.460 ±0.0074	0.464 ±0.0006
ETTh2	96	0.309	0.307 ±0.0008	**0.276** ±0.0013	0.277 ±0.0119	0.270	0.270 ±0.0001	0.275 ±0.0002	0.272 ±0.0015	0.279 ±0.0020
	192	0.423	0.447 ±0.0019	0.351 ±0.0140	0.351 ±0.0123	**0.331**	**0.331** ±0.0000	0.342 ±0.0025	0.335 ±0.0010	0.343 ±0.0026
	336	0.540	0.566 ±0.0014	0.424 ±0.0139	0.455 ±0.0053	**0.353**	0.354 ±0.0001	0.359 ±0.0062	0.357 ±0.0011	0.383 ±0.0028
	720	0.900	0.971 ±0.0018	0.664 ±0.0400	0.619 ±0.0185	0.380	**0.377** ±0.0001	0.384 ±0.0001	0.384 ±0.0012	0.406 ±0.0054
ECL	96	**0.133**	0.134 ±0.0002	0.134 ±0.0001	**0.133** ±0.0002	**0.133**	**0.133** ±0.0001	0.134 ±0.0001	0.134 ±0.0001	0.134 ±0.0002
	192	**0.147**	0.148 ±0.0001	0.148 ±0.0005	0.148 ±0.0001	0.148	0.148 ±0.0000	0.149 ±0.0000	0.148 ±0.0000	0.149 ±0.0002
	336	**0.162**	0.164 ±0.0002	0.164 ±0.0009	0.163 ±0.0001	0.164	0.164 ±0.0001	0.165 ±0.0001	0.165 ±0.0001	0.165 ±0.0001
	720	**0.197**	0.200 ±0.0001	**0.197** ±0.0034	0.198 ±0.0002	0.203	0.203 ±0.0000	0.205 ±0.0003	0.204 ±0.0001	0.205 ±0.0002
Traffic	96	**0.385**	0.386 ±0.0003	0.387 ±0.0003	0.386 ±0.0005	**0.385**	0.386 ±0.0002	0.387 ±0.0002	0.386 ±0.0004	0.387 ±0.0003
	192	**0.396**	0.397 ±0.0001	0.398 ±0.0001	0.398 ±0.0003	0.397	0.398 ±0.0001	0.399 ±0.0003	0.397 ±0.0004	0.398 ±0.0000
	336	**0.410**	0.411 ±0.0001	0.412 ±0.0001	0.412 ±0.0002	**0.410**	0.411 ±0.0001	0.412 ±0.0005	0.412 ±0.0000	0.412 ±0.0000
	720	0.450	0.450 ±0.0002	0.450 ±0.0006	0.451 ±0.0003	**0.448**	0.449 ±0.0001	0.450 ±0.0002	0.449 ±0.0002	0.451 ±0.0000
Weather	96	0.142	0.144 ±0.0002	0.145 ±0.0017	0.145 ±0.0011	**0.141**	0.142 ±0.0000	0.142 ±0.0006	0.143 ±0.0005	0.144 ±0.0004
	192	0.185	0.188 ±0.0013	0.188 ±0.0029	0.189 ±0.0028	**0.184**	0.185 ±0.0001	0.185 ±0.0008	0.185 ±0.0007	0.187 ±0.0010
	336	0.235	0.238 ±0.0003	0.235 ±0.0004	0.238 ±0.0019	**0.234**	0.236 ±0.0001	0.235 ±0.0003	0.235 ±0.0005	0.235 ±0.0002
	720	**0.304**	**0.304** ±0.0004	0.308 ±0.0005	0.310 ±0.0018	0.307	0.307 ±0.0001	0.310 ±0.0004	0.309 ±0.0006	0.311 ±0.0003
Exchange	96	0.091	0.099 ±0.0009	**0.084** ±0.0003	0.100 ±0.0097	0.086	0.087 ±0.0001	0.085 ±0.0003	0.086 ±0.0006	0.090 ±0.0008
	192	0.217	0.243 ±0.0032	**0.160** ±0.0088	0.161 ±0.0012	0.180	0.183 ±0.0005	0.178 ±0.0028	0.179 ±0.0024	0.187 ±0.0034
	336	0.450	0.498 ±0.0026	**0.315** ±0.0070	0.323 ±0.0126	0.343	0.344 ±0.0011	0.335 ±0.0031	0.334 ±0.0040	0.347 ±0.0024
	720	1.392	1.256 ±0.0083	0.929 ±0.0218	**0.717** ±0.1699	0.992	0.965 ±0.0010	0.920 ±0.0219	0.948 ±0.0082	1.035 ±0.0130

ated. Conversely, **blue** highlighting denotes cases where the differences are within one standard deviation.

Table 2 shows that the linear models are generally outperformed by their corresponding OLS solution (72% of settings). It is interesting that the OLS solution usually outperforms those trained with SGD and early stopping, particularly given that the OLS solutions are purely linear regression (not ridge or lasso regression), meaning that there is no regularisation. The comparably strong performance of the closed-form solution on larger datasets (ECL, Traffic, and Weather) suggests that a linear model may not have sufficient representational capacity in this setting. [7]

Conversely, FITS performs particularly well on the hourly ETT dataset (ETTh1 and h2). We believe that the reason for this is owed to the fact that these datasets are small, such that overfitting can occur rapidly. Since FITS inadvertently imposes a restriction on the bias parameter (see Section 4.1 and Figure 4), it is less prone to this overfitting restriction.

OLS solutions were superior across 23 of 32 (72%) settings.

6. Conclusion

Simple linear models are often on par, or better, than complex or deep models for time series forecasting. Thus, much energy has thus been spent on 'modernising' linear regression for time series forecasting: modelling separately trends and residuals (DLinear), applying some form of instance normalisation (RLinear, NLinear), or by processing in Fourier space (FITS). We have shown in this paper that, from a functional standpoint, these alterations barely deviate these models from standard linear regression. We demonstrated empirically that these model behave and perform similarly to each other and generally worse than their closed-form solutions. A full discussion of the limitations and future of this work may be found in Appendix G.4.

[7]Further discussion and comparisons may be found in Appendices G.3, G.2 and Appendix F respectively.

Impact Statement

This paper presents work whose goal is to advance the field of time series forecasting. The scope of the paper is narrow, focusing on analysing the structures of various popular linear forecasting models. While there are many potential societal consequences of our work, none which we feel must be specifically highlighted here. A list of limitations of our work can be found in Section G.4. To ensure reproducibility, the code to fit and evaluate OLS solutions in this paper can be found here: `github.com/sir-lab/linear-forecasting`.

Acknowledgments

The work presented in this paper was conducted during a research internship at the Systems Infrastructure Research lab, Huawei Edinburgh Research Centre. William Toner is grateful for the support and resources provided by Huawei, which significantly contributed to the research outcomes, and for the supervision and contributions of Luke Darlow. This work has also been supported in part by Adaptive and Neural Computation (ANC) at the School of Informatics, Edinburgh University.

References

Darlow, L. N., Joosen, A., Asenov, M., Deng, Q., Wang, J., and Barker, A. FoldFormer: Sequence folding and seasonal attention for fine-grained long-term FaaS forecasting. In *Proceedings of the 3rd Workshop on Machine Learning and Systems*, pp. 71–77, 2023.

Darlow, L. N., Deng, Q., Hassan, A., Asenov, M., Singh, R., Joosen, A., Barker, A., and Storkey, A. DAM: Towards a foundation model for forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=4NhMhElWqP`.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Joosen, A., Hassan, A., Asenov, M., Singh, R., Darlow, L., Wang, J., and Barker, A. How does it function? characterizing long-term trends in production serverless workloads. In *Proceedings of the 2023 ACM Symposium on Cloud Computing*, SoCC '23, pp. 443–458, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400703874. doi: 10.1145/3620678.3624783. URL `https://doi.org/10.1145/3620678.3624783`.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Li, Z., Qi, S., Li, Y., and Xu, Z. Revisiting long-term time series forecasting: An investigation on linear mapping, 2023.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*, 2021.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting, 2023.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Silva, T. Understanding linear regression using the singular value decomposition, 2024. URL `https://sthalles.github.io/svd-for-regression/`. Online; accessed Day Month Year.

Sloss, B. T., Nukala, S., and Rau, V. Metrics that matter. *Communications of the ACM*, 62(4):88–88, 2019.

Taylor, S. J. and Letham, B. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

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

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

Xu, Z., Zeng, A., and Xu, Q. Fits: Modeling time series with $10k$ parameters. *arXiv preprint arXiv:2307.03756*, 2023.

Zeng, A. Ltsf-linear. https://github.com/cure-lab/LTSF-Linear, 2023.

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

Zhijian, X. Fits. https://github.com/VEWOXIC/FITS, 2023.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

A. Appendix

A.1. FITS

This section is dedicated to fully unpacking the FITS model and proving Theorem 3.2.

FITS Model Definition: Let $\vec{x} \in \mathbb{R}^L$ be a context vector. FITS applies the Real (discrete) Fourier Transform (RFT) to $\vec{x}$. This maps $\vec{x}$ to a complex vector of length $\lfloor L/2 \rfloor + 1$. After this one applies a Low-Pass Filter (LPF), zeroing out the high frequency components. Next one applies a learnable complex linear layer. The output is padded with zeros and the result is passed though the inverse RFT, mapping to $\mathbb{R}^{L+T}$. The result is then scaled by $\frac{L+T}{L}$.

Throughout this section we will assume both the prediction horizon length T and the context length L are even. This will avoid over-cluttered expressions involving the floor functions. Moreover this condition holds for every experiment setting in the original paper (Xu et al., 2023).

Remark: FITS is a state-of-art model. One of the goals of this paper is to understand the superior performance of this model given that it is simply a composition of linear operations. For this reason we restrict our analysis to those settings which give SoTA performance. To this end we ignore the LPF entirely in our subsequent analysis. While the LPF is an effective tool for compressing FITS, it comes with a performance degradation.

In order to fully analyse FITS it is critical to introduce definitions for the Discrete and Real Fourier transforms.

Definition A.1 (Discrete Fourier Transform)**.** Let $\vec{x} \in \mathbb{R}^L$, we define the **Discrete Fourier Transform (DFT)** of $\vec{x}$ as $DFT_L : \mathbb{C}^L \to \mathbb{C}^L$ so that for $j \in \{0, 1, \dots \lfloor L \rfloor\}$

$$DFT_L(\vec{x})_j := \sum_{k=0}^{L-1} e^{\frac{-2\pi i k j}{L}} x_k \tag{3}$$

The DFT can be written in matrix form $DFT_L(\vec{x}) = D_L \vec{x}$ where, if ω denotes the L^{th} root of unity ($\omega := e^{\frac{-2\pi i}{L}}$), then D_L is the matrix:

$$D_L := \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{L-1} \\ & & \dots\dots\dots\dots\dots\dots & & \\ 1 & \omega^{L-1} & \omega^{2(L-1)} & \dots & \omega^{L(L-1)} \end{bmatrix} \tag{4}$$

The DFT is an invertible map and we define the **inverse DFT** (iDFT) by $iDFT(\vec{x}) := D_L^{-1} \vec{x}$ where $D_L^{-1} = \frac{1}{L} D_L^{\star}$.

FITS does not directly employ the DFT. Rather, it employs a closely related transform called the Real Fourier Transform, which we now define.

Definition A.2 (Real Discrete Fourier Transform)**.** Let $\vec{x} \in \mathbb{R}^L$, we define the **Real Discrete Fourier Transform (RFT)** of $\vec{x}$ as $RFT_L : \mathbb{R}^L \to \mathbb{C}^{\lfloor L/2 \rfloor + 1}$ so that for $j \in \{0, 1, \dots \lfloor L/2 \rfloor\}$

$$RFT_L(\vec{x})_j := \sum_{k=0}^{L-1} e^{\frac{-2\pi i k j}{L}} x_k \tag{5}$$

In other words, the RFT and DFT are identical other than the RFT is a truncated version that discards the last $\lfloor L/2 \rfloor - 1$ components. The motivation for this comes from the fact that when $\vec{x}$ is real, then the k^{th} and $(L-k)^{\text{th}}$ components of the DFT are complex conjugates ($DFT(\vec{x})_j = DFT(\vec{x})^{\star}_{L-j}$). For this reason these components contain the same information in that the original signal may be entirely reconstructed from the first $\lfloor L/2 \rfloor + 1$ components via $(Y_0, Y_1, \dots, Y_{\lfloor L/2 \rfloor + 1}) \mapsto iDFT(Re(Y_0), Y_1, Y_2, \dots, Re(Y_{\lfloor L/2 \rfloor}), Y^{\star}_{\lfloor L/2 \rfloor - 1}, \dots, Y^{\star}_1)$. We call this map the **inverse-RFT (iRFT)**.

When $\vec{Y}$ has been obtained by taking the RFT of some real vector then $Y_0, Y_{\lfloor L/2 \rfloor} \in \mathbb{R}$ thus, taking the real part of these components, $Re(Y_0), Re(Y_{\lfloor L/2 \rfloor})$, does nothing. However writing the inverse like this allows us to also take the inverse RFT of complex vectors $\vec{Y} \in \mathbb{C}^{\lfloor L/2 \rfloor}$ which do not lie in the image $RFT(\mathbb{R}^L)$.

We can make the relationship between the DFT and RFT more explicit by defining the following linear map.

Definition A.3. Define the projection $\Pi_L : \mathbb{C}^L \to \mathbb{C}^{\frac{L}{2}+1}$

$$\Pi_L(Y_0, Y_1, \ldots, Y_{L-1}) := (Y_0, Y_1, \ldots, Y_{\frac{L}{2}})$$

Let $\vec{Y} = (Y_0, Y_1, \ldots, Y_{\frac{L}{2}})$. Then, the inverse map $\Pi_L^{-1} : \mathbb{C}^{\frac{L}{2}+1} \to \mathbb{C}^L$ is defined as

$$\Pi_L^{-1}(\vec{Y}) = (Re(Y_0), Y_1, \ldots, Re(Y_{\frac{L}{2}}), Y^{\star}_{\frac{L}{2}-1}, \ldots, Y^{\star}_1)$$

Example: $\Pi_{L+T}^{-1}(Y_0, Y_1, Y_2, Y_3) = (\frac{Y_0+Y_0^{\star}}{2}, Y_1, Y_2, \frac{Y_3+Y_3^{\star}}{2}, Y_2^{\star}, Y_1^{\star})$

Remark A.4. Using this transformation one may express $RFT_L = \Pi_L \circ D_L$ and $iRFT_L = D_L^{-1} \circ \Pi_L^{-1}$

Remark A.5. For any $\vec{Y} \in DFT_L(\mathbb{R}^L)$ one may confirm that $\Pi_L^{-1} \circ \Pi_L = id_L$. Likewise, for any $\vec{Y} \in RFT_L(\mathbb{R}^L)$ one may confirm that $\Pi_L \circ \Pi_L^{-1} = id_{\frac{L}{2}+1}$

Having explicitly defined the discrete and discrete real Fourier transforms we are ready to begin the process of proving Theorem 3.2 which we restate below.

Theorem A.6 (FITS)**.** *Let $\mathcal{M}(FITS)$ denote the FITS model class, i.e. the set of functions $f : \mathbb{R}^L \to \mathbb{R}^T$ which can be represented as a FITS model. When $L \geq T - 2$, $\mathcal{M}(FITS)$ is precisely equal to the space of affine linear functions $A\vec{x} + \vec{b}$.*

As a composition of affine linear operation, FITS is itself an affine linear model. As a result any FITS model may be expressed in the form $A\vec{x} + \vec{b}$. The remainder of this section is dedicated to showing that when $L \geq T - 2$ that A and $\vec{b}$ are unconstrained meaning that FITS model class is equivalent to unconstrained linear regression. Before this we present a prescription, showing how one may obtain $A, \vec{b}$, given the complex bias and weight matrix from FITS's linear layer.

Lemma A.7. *Let $f : \mathbb{R}^L \to \mathbb{R}^{L+T}$ be some FITS model. Let $W, \vec{c}$ be the weight matrix and bias of the complex linear layer in this model. Then we can express f as a real affine linear map $f(x) = A\vec{x} + \vec{b}$ where $A = D_{L+T}^{-1} \circ \Pi_{L+T}^{-1} W \circ \Pi_L \circ D_L$ and $\vec{b} = iRFT(\vec{c})$*

Proof. As discussed before, as a composition of affine linear operation, FITS is itself an affine linear model. As a result any FITS model may be expressed in the form $A\vec{x} + \vec{b}$. One may recover the bias by applying f to the zero vector by noting that $f(\vec{0}) = A\vec{0} + \vec{b} = \vec{b}$. f is a composition of the RFT, the complex affine map $\vec{x} \mapsto W\vec{x} + \vec{c}$ and an iRFT. Since the RFT maps zero to zero then $f(\vec{0}) = iRFT(W\vec{0} + \vec{c}) = iRFT(\vec{c})$ as desired.

We know therefore that $A\vec{x} = A\vec{x} + \vec{b} - \vec{b} = iRFT(W\vec{z} + \vec{c}) - iRFT(\vec{c})$ where $\vec{z} := RFT(\vec{x})$. Using the linearity of the iRFT we have $A\vec{x} = iRFT(W\vec{z} + \vec{c} - \vec{c}) = iRFT(W\vec{z}) = iRFT \circ W \circ RFT\vec{x}$. By writing the RFT and iRFT in terms of the DFT and the operator Π as in Remark A.4 concludes our proof of Lemma A.7. □

Proof Structure for Theorem 3.2: Our goal is to demonstrate that for $L \geq T - 2$, any affine map $\vec{x} \mapsto A\vec{x} + \vec{b}$ can be represented using a FITS architecture. We achieve this by characterizing the set of matrices representable within a FITS framework. The detailed characterization is presented in Lemma A.8 and Lemma A.9, which will be introduced subsequently. In Lemma A.8, we introduce a specific set of linear maps, illustrating their formulation as complex matrix multiplications and detailing the process for deriving the corresponding matrix from the linear map. Lemma A.9 then ties these concepts directly to the FITS architecture, demonstrating how the linear map type discussed is integral to FITS. This establishes a comprehensive characterization of matrices expressible via FITS. Following these lemmas, we will prove that for $L \geq T - 2$, this characterization includes all affine transformations $\vec{x} \mapsto A\vec{x} + \vec{b}$.

In order to prove Theorem 3.2 we must introduce the following set of complex matrices.

Lemma A.8. *Let $\mathcal{A}$ denote the set of linear maps $T : DFT(\mathbb{R}^{\mathbb{L}}) \to iDFT(\mathbb{R}^{L+T})$ which can be expressed as a composition $T = \Pi_{L+T}^{-1} \circ W \circ \Pi_L$ where W is some $\left(\frac{L+T}{2} + 1\right)$ by $\left(\frac{L}{2} + 1\right)$ complex matrix. We claim that each T can be expressed as* *a complex matrix multiplication $T(W) : DFT(\mathbb{R}^L) \to iDFT(\mathbb{R}^{L+T})$ where $T(W)$ is derived from W as follows:*

$$(T(W))_{ij} = \left\{ \begin{array}{lr} Re(W_{ij}), & i \in \{0, \frac{L+T}{2}\}, j = 0, \frac{L}{2} \\ \frac{1}{2}(W_{ij}), & i \in \{0, \frac{L+T}{2}\}, 0 < j < L/2 \\ \frac{1}{2}(W^{\star}_{i,L-j}), & i \in \{0, \frac{L+T}{2}\}, j > L/2 \\ W_{ij}, & 0 < i < \frac{L+T}{2}, j \leq \frac{L}{2} \\ 0, & 0 < i < \frac{L+T}{2}, j > \frac{L}{2} \\ W^{\star}_{L+T-i,j}, & i \notin \{0, \frac{L+T}{2}\}, j = 0 \\ W^{\star}_{L+T-i,L-j}, & \textit{otherwise} \end{array} \right\} \tag{6}$$

For example let $T = 2$ and $L = 4$ and let W be an arbitrary complex 4×2 matrix. Then one has:

$$T(W) = (\Pi^{-1}_{L+T}(W\Pi_L)) = \begin{bmatrix} Re(W_{00}) & \frac{W_{01}}{2} & Re(W_{02}) & \frac{W^{\star}_{01}}{2} \\ W_{10} & W_{11} & W_{12} & 0 \\ W_{20} & W_{21} & W_{22} & 0 \\ Re(W_{30}) & \frac{W_{31}}{2} & Re(W_{32}) & \frac{W^{\star}_{31}}{2} \\ W^{\star}_{20} & 0 & W^{\star}_{22} & W^{\star}_{21} \\ W^{\star}_{10} & 0 & W^{\star}_{12} & W^{\star}_{11} \end{bmatrix} \tag{7}$$

Remark: In the statement of Lemma A.8, $\Pi^{-1}_{L+T} \circ W \circ \Pi_L$ denotes the application of Π^{-1}_{L+T} to each column of W and applying Π_L to each row. The order of these operations makes no difference since Π_L is a projection.

Proof. Let W be some arbitrary complex matrix of dimension $\left(\frac{L+T}{2} + 1\right)$ by $\left(\frac{L}{2} + 1\right)$. Let T be the linear map defined on the domain $DFT(\mathbb{R}^{\mathbb{L}})$ formed from the composition $\Pi^{-1}_{L+T} \circ W \circ \Pi_L$. We begin by assuming that there exists a complex $T + L$ by L matrix $T(W)$ which is equivalent to this linear map and derive it's structure. At the end we then show that it is indeed equivalent to the linear map T.

As a projection onto the first $1 + \frac{L}{2}$ components, Π_L can be written as an $(\frac{L}{2} + 1) \times L$ matrix where the ii^{th} entry of Π_L is 1 and all other entries are zero. Right composing W by Π_L yields a single matrix equivalent to appending $\frac{L}{2} - 1$ columns of zeros to the right of W. That is:

$$(\Pi W)_{ij} = \left\{ \begin{array}{lr} W_{ij}, & \text{for } j \leq \frac{L}{2} + 1 \\ 0, & \text{otherwise} \end{array} \right\} \tag{8}$$

For example, in the case $L = 4, T = 2$;

$$\begin{aligned} W \circ \Pi_L &= \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \\ W_{41} & W_{42} & W_{43} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} W_{11} & W_{12} & W_{13} & 0 \\ W_{21} & W_{22} & W_{23} & 0 \\ W_{31} & W_{32} & W_{33} & 0 \\ W_{41} & W_{42} & W_{43} & 0 \end{bmatrix} \end{aligned}$$

Now, let $B = (W \circ \Pi_L)$ be some complex $(\frac{T+L}{2} \times L)$ matrix. One may similarly write $\Pi^{-1}_{L+T} \circ B$, as a single matrix $D \in \mathbb{C}^{((T+L)\times L)}$. Using the definition of Π^{-1}_{T+L} (Definition A.3) we can derive the entries of the matrix D. We do this by noting that the matrix D must satisfy $D\vec{Y} = \Pi^{-1}_{L+T} \circ B$ for any $\vec{Y} \in DFT(\mathbb{R}^L)$. By equating the components $(D\vec{Y})_i = (\Pi^{-1}_{L+T} \circ B)_i$, one may deduce the matrix D in terms of B.

Case 1: Let $i = 0, \frac{L+T}{2}$.

$(D\vec{Y})_i = (\Pi_{L+T}^{-1} B\vec{Y})_i := Re((B\vec{Y})_i) = \frac{(B\vec{Y})_i + (B\vec{Y})_i^\star}{2}$. Therefore,

$$
\begin{aligned}
(D\vec{Y})_i = \sum_{j=0}^{L-1} D_{ij} Y_j &= \frac{1}{2}\Bigg(\sum_{j=0}^{L-1} B_{ij} Y_j + B_{ij}^\star Y_j^\star \Bigg) \\
&= \left(\frac{B_{i0} + B_{i0}^\star}{2} \right) Y_0 + \sum_{j=1}^{L-1} \left(\frac{B_{ij} + B_{i,L-j}^\star}{2} \right) Y_j
\end{aligned}
$$

As this holds for all $\vec{Y} \in DFT(\mathbb{R}^L)$, we may conclude; $D_{i0} = Re(B_{i0}), D_{i,L/2} = Re(B_{i,L/2})$ and otherwise; $D_{ij} = \left(\frac{B_{ij} + B_{i,L-j}^\star}{2} \right)$.

Case 2: Let $0 < i < \frac{L+T}{2}$.

By the definition of Π_{L+T}^{-1} we have $(D\vec{Y})_i = (\Pi_{L+T}^{-1} B\vec{Y})_i = (B\vec{Y})_i$. Since this holds for all $\vec{Y}$ we must have $D_{ij} = B_{ij}$ for all j.

Case 3: Let $i > \frac{L+T}{2}$.

Using Def. A.3 we may derive the following which holds for all $\vec{Y} \in DFT(\mathbb{R}^L)$:

$$
\begin{aligned}
(D\vec{Y})_i &= (\Pi_{L+T}^{-1} B\vec{Y})_i = (B\vec{Y})_{L+T-i}^\star \\
\implies (\sum_{j=0}^{L-1} D_{ij} Y_j) &= \sum_{j=0}^{L-1} B_{L+T-i,j}^\star Y_j^\star \\
&= (\sum_{j=1}^{L-1} B_{L+T-i,j}^\star Y_{L-j}) + B_{L+T-i,0}^\star Y_0
\end{aligned}
$$

Since $\vec{Y} \in DFT(\mathbb{R}^L)$ we know that $Y_0, Y_{L/2} \in \mathbb{R}$ and otherwise $Y_j = Y_{L-j}^\star$. It follows therefore that $D_{i0} = B_{L+T-i,0}^\star$ and $D_{ij} = B_{L+T-i,L-j}^\star$ for $j > 0$.

Below we summarise our findings, writing a general expression for the ijth component of $D = \Pi_{L+T}^{-1} B$

$$
(\Pi_{L+T}^{-1} B)_{ij} = \left\{ \begin{array}{ll} Re(B_{ij}), & i \in \{0, \frac{L+T}{2}\}, j = 0 \\ \frac{1}{2}(B_{ij} + B_{i,L-j}^\star), & i \in \{0, \frac{L+T}{2}\}, j \neq 0 \\ B_{ij}, & 0 < i < \frac{L+T}{2} \\ B_{L+T-i,j}^\star, & i \notin \{0, \frac{L+T}{2}\}, j = 0 \\ B_{L+T-i,L-j}^\star, & \textit{otherwise} \end{array} \right\} \tag{9}
$$

We may combine Equation 9 with the earlier Equation 8 to establish a general form for the ijth element of $\Pi_{L+T}^{-1} W \Pi_L$:

$$
T(W)_{ij} := (\Pi_{L+T}^{-1} W \Pi_L)_{ij} = \left\{ \begin{array}{ll} Re(W_{ij}), & i \in \{0, \frac{L+T}{2}\}, j = 0, \frac{L}{2} \\ \frac{1}{2}(W_{ij}), & i \in \{0, \frac{L+T}{2}\}, 0 < j < L/2 \\ \frac{1}{2}(W_{i,L-j}^\star), & i \in \{0, \frac{L+T}{2}\}, j > L/2 \\ W_{ij}, & 0 < i < \frac{L+T}{2}, j \leq \frac{L}{2} \\ 0, & 0 < i < \frac{L+T}{2}, j > \frac{L}{2} \\ W_{L+T-i,j}^\star, & i \notin \{0, \frac{L+T}{2}\}, j = 0 \\ W_{L+T-i,L-j}^\star, & \textit{otherwise} \end{array} \right\} \tag{10}
$$

This is precisely the characterisation given in Equation 10.

Existence Proof: We have demonstrated the necessary structure for a complex matrix $T(W)$ that is equivalent to the linear map $\Pi_{L+T}^{-1} \circ W \circ \Pi_L$. The task now is to prove that such a complex matrix representation, $T(W)$, indeed exists.

The map $\Pi_{L+T}^{-1} \circ W \circ \Pi_L$, being real-linear, can be represented by a real matrix M when considering its domain, $DFT(\mathbb{R}^L)$, as a real vector space of dimension $2L$. The domain $DFT(\mathbb{R}^L)$ consists of complex vectors $(z_0, z_1, \ldots, z_{L/2}, \ldots, z_{L-1})$, with z_0 and $z_{L/2}$ real, and $z_i = z_{L-i}^*$ for other indices, indicating complex conjugate pairs.

To transition from a real to a complex matrix representation, we exploit the structure of these complex vectors by expressing the real and imaginary parts of z_i as $\text{Re}(z_i) = \frac{z_i + z_i^*}{2}$ and $\text{Im}(z_i) = \frac{z_i - z_i^*}{2}$, respectively. This approach allows for the real matrix M, defined in terms of the real and imaginary components, to be reformulated as a complex matrix, thus confirming the existence and formulation of $T(W)$ as a complex matrix multiplication. □

Lemma A.9. *Any FITS model can be expressed in the form $\vec{x} \mapsto A\vec{x} + \vec{b}$ where $A \in D_{L+T}^{-1} \circ \mathcal{A} \circ D_L$ and $\vec{b} \in \mathbb{R}^{L+T}$. Here $\mathcal{A}$ denotes the set of matrices introduced in Lemma A.8. Conversely, if $\vec{x} \mapsto A\vec{x} + \vec{b}$ is affine linear map such that $A \in D_{L+T}^{-1} \circ \mathcal{A} \circ D_L$, then there exists a functionally equivalent FITS model.*

Proof. We reiterate, that as a sequence of affine linear operations, FITS is a real affine linear model $\mathbb{R}^L \to \mathbb{R}^{L+T}$. It follows that any FITS model can be expressed in the form $A\vec{x} + \vec{b}$ for some choice of $A \in \mathbb{R}^{(L+T)\times L}$ and $\vec{b} \in \mathbb{R}^L$. It remains to show that for any FITs model, A can be selected from the family $D_{L+T}^{-1} \circ \mathcal{A} \circ D_L$.

We showed in Lemma A.7 that, if $W, \vec{c}$ are the weights matrix and bias of the complex linear layer in the FITS model then $\vec{b} = iRFT(\vec{c})$ and $A = D_{L+T}^{-1} \circ \Pi_{L+T}^{-1} W \circ \Pi_L \circ D_L$.

Putting this together, we have

$$FITS(\vec{x}; W, \vec{c}) = D_{L+T}^{-1}(\Pi_{L+T}^{-1} W \Pi_L) D_L \vec{x} + iRFT(\vec{c})$$

which can be compactly expressed as

$$FITS(\vec{x}; W, \vec{c}) = D_{L+T}^{-1} B D_L$$

where $B = \Pi_{L+T}^{-1} W \Pi_L : DFT(\mathbb{R}^L) \to iDFT(\mathbb{R}^{L+T})$ belongs to $\mathcal{A}$ as desired.

Since W can be any complex matrix then the converse also holds in that any linear map $\vec{x} \mapsto A\vec{x}$ where $A \in D_{L+T}^{-1} \mathcal{A} D_L$ must be equivalent to a FITS model.

It remains to show that any bias $\vec{b} \in \mathbb{R}^{L+T}$ can be expressed in the form $iRFT(\vec{c})$ where $\vec{c} \in \mathbb{C}^{\frac{L+T}{2}+1}$. This follows from the bijectivity of the DFT. Specifically, we can obtain any $\vec{b}$ by letting $\vec{c} := RFT(\vec{b})$. Then $iRFT(RFT(\vec{b})) = iDFT \circ \Pi_{L+T}^{-1} \circ \Pi_{L+T} \circ DFT(\vec{b}) = \vec{b}$ by Remark A.5.

□

Having proved Lemma A.9 and the more technical Lemma A.8 we are now ready to prove Theorem 3.2.

Proof of Lemma 3.2

Proof. We showed in Lemma A.9 that every FITS model $\vec{x} \mapsto FITS(\vec{x}; W, \vec{c})$ can be written in the form $x \mapsto A\vec{x} + \vec{b}$ where $A \in \mathbb{R}^{(L+T)\times L}$, $\vec{b} \in \mathbb{R}^{L+T}$. Moreover we showed how one may obtain $A, \vec{b}$ from $W, \vec{c}$ via $A = D_{L+T}^{-1} \Pi_{L+T}^{-1} W \Pi_L D_L$ and $\vec{b} = iRFT\vec{c}$. FITS outputs both a forecast and a reconstruction of the context. Consequently we may decompose $A = \begin{bmatrix} A_L \\ A_T \end{bmatrix}$ where $A_T \in \mathbb{R}^{T\times L}$ is the matrix which produces a forecast from the context vector. We have already seen in Lemma A.9 that FITS imposes no restriction on our bias term $\vec{b}$. Our claim is that additionally, when $L \geq T - 2$, any real $T \times L$ matrix A_T by be attained an appropriate selection of W. If we define the operator $P : \mathbb{R}^{(T+L)\times L} \to \mathbb{R}^{T\times L}$ by $P\left(\begin{bmatrix} A_L \\ A_T \end{bmatrix}\right) = A_T$ then we can formulate this claim as

$$L \geq T - 2 \implies \mathbb{R}^{T\times L} \subseteq P \circ D_{L+T}^{-1} \circ \mathcal{A} \circ D_L$$

Since D_L is bijective this may be equivalently be written as

$$\mathbb{R}^{T\times L}\circ D_L^{-1}\subseteq P\circ D_{L+T}^{-1}\circ\mathcal{A}$$

Note that we already have the reverse inclusion $P\circ D_{L+T}^{-1}\circ\mathcal{A}\circ D_L\subseteq\mathbb{R}^{T\times L}$ by Lemma A.9.

We begin by characterising the space of matrices $\mathbb{R}^{T\times L}\circ D_L^{-1}$. This is the space of matrices one gets when you apply an inverse DFT to the rows of all real $T\times L$ matrices. That is, $\mathbb{R}^{T\times L}\circ D_L^{-1}$ is the subset of complex $T\times L$ matrices where each row is in the set $D_L^{-1}(\mathbb{R}^L)$. These are precisely the complex vectors of length L where $v_0, v_{L/2}\in\mathbb{R}$ and where otherwise $v_i=v_{L-i}^\star$. For example, for $T=6, L=4$ the general form of $\mathbb{R}^{T\times L}\circ D_L^{-1}$ can be written as follows where lowercase denotes a real entry.

$$\begin{bmatrix} b_{00} & B_{01} & b_{02} & B_{01}^\star \\ b_{01} & B_{11} & b_{12} & B_{11}^\star \\ b_{02} & B_{21} & b_{22} & B_{21}^\star \\ b_{03} & B_{31} & b_{32} & B_{31}^\star \\ b_{04} & B_{41} & b_{42} & B_{41}^\star \\ b_{05} & B_{51} & b_{52} & B_{51}^\star \end{bmatrix} \tag{11}$$

Using this fact, $\mathbb{R}^{T\times L}\circ D_L^{-1}$ may be alternatively be characterised as the set of complex $T\times L$ matrices where the zeroth and $L/2$th columns, $c_0, c_{L/2}$, are arbitrary real vectors and where otherwise all other columns are arbitrary complex vectors subject to the condition $c_i=c_{L-i}^\star$. Written as a vector space isomorphism this is:

$$\mathbb{R}^{T\times L}\circ D_L^{-1}\cong\mathbb{R}^T\oplus\underbrace{\mathbb{C}^T\oplus\ldots\oplus\mathbb{C}^T}_{(\frac{L}{2}-1)\text{ times}}\oplus\mathbb{R}^T\oplus\underbrace{\mathbb{C}^T\ldots\oplus\mathbb{C}^T}_{(\frac{L}{2}-1)\text{ times}}$$

Using Lemma A.8 one may characterise $\mathcal{A}$ similarly in terms of its columns. The zeroth and $L/2$th columns are arbitrary vectors in $D_{L+T}(\mathbb{R}^{L+T})$. For $0<i<L/2$ the ith column, c_i, is an arbitrary complex vector of length $T+L$, subject to the condition $c_{ij}=0$ for $j>(T+L)/2$. For $i>L/2$ column c_i satisfies the condition $c_{i0}=c_{L-i,0}$ and otherwise $c_{ij}=c_{L-i,L+T-j}^\star$. In the case $T=2, L=4$ the general form for a matrix in $\mathcal{A}$ can be written as follows where lowercase once again denotes a real entry:

$$\begin{bmatrix} a_{00} & A_{01} & A_{02} & a_{03} & A_{02}^\star & A_{01}^\star \\ A_{10} & A_{11} & A_{12} & A_{13} & 0 & 0 \\ A_{20} & A_{21} & A_{22} & A_{23} & 0 & 0 \\ a_{30} & A_{31} & A_{32} & a_{33} & A_{32}^\star & A_{31}^\star \\ A_{20}^\star & 0 & 0 & A_{23}^\star & A_{22}^\star & A_{21}^\star \\ A_{20}^\star & 0 & 0 & A_{13}^\star & A_{12}^\star & A_{11}^\star \end{bmatrix}$$

We will use the notation $\mathcal{S}$ to represent the space of complex vectors $\vec{v}\in\mathbb{C}^{(L+T)}$ where $v_i=0$ for $i>(T+L)/2$.

$$\mathcal{S}:=\{\vec{v}\in\mathbb{C}^{(L+T)}|v_i=0, i>(T+L)/2\}$$

Using this, one may write $\mathcal{A}$ as a vector isomorphism.

$$\mathcal{A}\cong D_{L+T}(\mathbb{R}^{L+T})\oplus\underbrace{\mathcal{S}\oplus\ldots\oplus\mathcal{S}}_{(\frac{L}{2}-1)\text{ times}}\oplus D_{L+T}(\mathbb{R}^{L+T})\oplus\underbrace{\mathcal{S}\oplus\ldots\oplus\mathcal{S}}_{(\frac{L}{2}-1)\text{ times}}$$

Remark: Note that in both cases $\mathcal{A}$ and $\mathbb{R}^{T\times L}\circ D_L^{-1}$ are completely specified by their first $(L/2)$+1 columns since the for $i>L/2$ column c_i can be determined completely by c_{L-i}

If A_T is some arbitrary matrix in $\mathbb{R}^{T\times L}\circ D_L^{-1}$ we want to show that, when $L\geq T-2$, we can find $W\in\mathcal{A}$ where $P\circ D_{L+T}^{-1}(W)=A$.

We observe that the linear map $P \circ D_{L+T}^{-1} : \mathbb{C}^{L+T} \to \mathbb{C}^T$ operates independently on each column of $\mathcal{A}$. Thus, using the decompositions given above for $\mathcal{A}$ and $D_{L+T}(\mathbb{R}^{L+T})$, we only need to show that:

$$\mathbb{R}^T \subseteq P \circ D_{L+T}^{-1}(D_{L+T}(\mathbb{R}^{L+T}))$$

and

$$\mathbb{C}^T \subseteq P \circ D_{L+T}^{-1}(\mathcal{S})$$

The former of these is trivial since D_{L+T} is a bijection; hence $P \circ D_{L+T}^{-1}(D_{L+T}(\mathbb{R}^{L+T})) = P(\mathbb{R}^{L+T}) = \mathbb{R}^T$. To show the second inclusion, we recollect that we already have $P \circ D_{L+T}^{-1}(\mathcal{S}) \subseteq \mathbb{C}^T$. Hence, we need only to show that the dimension of the space $P \circ D_{L+T}^{-1}(\mathcal{S})$ is greater than T (the dimension of $\mathbb{C}^T$).

We claim that $\dim(P \circ D_{L+T}^{-1}(\mathcal{S})) = \min(T, \frac{T+L}{2} + 1)$. Hence, we have $\dim(P \circ D_{L+T}^{-1}(\mathcal{S})) \geq T \iff \frac{T+L}{2} + 1 \geq T \iff L \geq T - 2$ as required.

In order to demonstrate this claim, note that $P \circ D^{-1}L + T$ can be written as a $T \times (L+T)$ matrix formed by taking the bottom T rows of the matrix $DL + T^{-1}$. Then, due to the structure of $\mathcal{S}$ (namely, that $v_i = 0$ for all $i > \frac{T+L}{2}$), $\dim(P \circ D^{-1}L + T(\mathcal{S}))$ is equal to the rank of the $T \times \left(\frac{T+L}{2} + 1\right)$ submatrix extracted from the bottom left of the $(L+T) \times (L+T)$ matrix $DL + T^{-1}$. If we can show that this submatrix has full row and column rank, then we are done. Let $a := \min\left(\frac{T+L}{2} + 1, T\right)$ and form the squared $a \times a$ matrix by discarding the excess rows or columns. We claim this square matrix has rank a. This follows from the fact that this submatrix is a Vandermonde matrix generated from a root of unity, thus it has a non-zero Vandermonde determinant and is therefore full rank.

$\square$

B. Further Results and Experiments

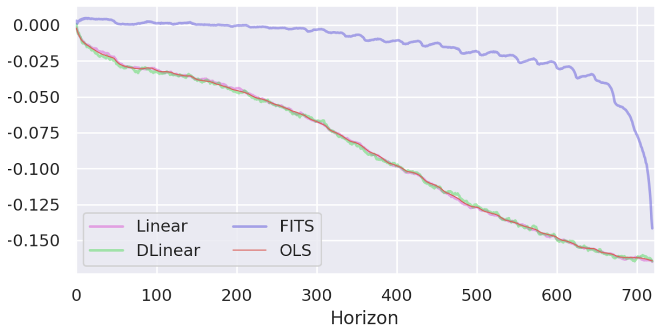

Figure 5. The biases learned by the FITS, Linear, DLinear after being trained on ETTh1 for 50 epochs. We also include the bias learned by the closed-form OLS linear regression. We note that, in line with theory from Section 3, we get the same bias for the DLinear, OLS and Linear models. Notably the bias for FITS is substantially different. This is explained by the choice of normalisation used in the Fourier transform in FITS.

C. FITS Bias Term - Detailed Breakdown

In this section we explain and breakdown Section 4.1 explaining how FITS operates as an almost bias-free model early in training.

In Def A.1 we defined the DFT and its inverse. The definition we use is standard and in line the implementation used in FITS [8]. An alternative definition of the DFT instead defines it as follows:

$$DFT_L(\vec{x})_j := \frac{1}{\sqrt{N}} \sum_{k=0}^{L-1} e^{\frac{-2\pi i k j}{L}} x_k \tag{12}$$

[8] https://github.com/VEWOXIC/FITS/

The inverse DFT is then defined as $\frac{1}{\sqrt{N}}D_L^*$ where D_L is as defined in Eqn. 4.

This second definition is identical, differing only in the choice of normalisation. This alternative definition is referred to as the *Normalised* or *Orthonormallly Normalised* DFT. A key property of the normalised DFT is that it is distance-preserving. By this we mean that $||DFT(\vec{x})||_2 = ||\vec{x}||_2$. On the contrary, the DFT as it is defined in Def. A.1 satisfies $||DFT(\vec{x})||_2 = \sqrt{N}||\vec{x}||_2$ where N is the number of components of the vector $\vec{x}$. In other words, the DFT as defined in Section 3.1.2 stretches distances by a factor of $\sqrt{N}$. The opposite of this is true for the inverse DFT so that $||iDFT(\vec{x})||_2 = \frac{1}{\sqrt{N}}||\vec{x}||_2$.

We saw in Lemma A.7 that any FITS model may be expressed in the form $A\vec{x} + \vec{b}$. Moreover if $\vec{c}$ denotes the bias in FITS's complex linear layer then we may obtain $\vec{b}$ via $\vec{b} = iRFT(\vec{c})$. Since the $iRFT$ is real-linear this means that $\vec{b}$ and $\vec{c}$ are related via a matrix equation $\vec{b} = M\vec{C}$ where $\vec{C}$ is the real vector obtained by splitting $\vec{c}$ into its real and imaginary components $\vec{C} := \begin{bmatrix} Re(\vec{c}) \\ Im(\vec{c}) \end{bmatrix}$

Let us now consider what the ramifications are of learning $\vec{b}$ by stochastic gradient descent (SGD) using the parameterisation $\vec{b} = M\vec{C}$ rather than learning $\vec{b}$ directly. If $\bar{v}$ denotes the derivative of the loss with respect to the variable $\vec{v}$: that is $\bar{v} := \frac{\partial L}{\partial \vec{v}}$ and η denotes our learning rate then the gradient update using a naive parameterisation of $\vec{b}$ is:

$$\vec{b} \mapsto \vec{b} - \eta\bar{b}$$

Conversely, if we let $\vec{b} = M\vec{C}$ and we instead learn $\vec{C}$ by gradient descent. One may show by the chain rule that $\bar{C} := M^T\bar{b}$. Thus, using the same learning rate as before, this induces an update

$$\begin{aligned} \vec{C} &\mapsto \vec{C} - \eta\bar{C} = \vec{C} - \eta M^T\bar{b} \\ \implies \vec{b} &\mapsto \vec{b} - \eta M M^T\bar{b} \end{aligned}$$

Thus, this choice of parameterisation means that we get an update of $MM^T\bar{b}$ where naively we would have an update of $\bar{b}$.

It should be immediately clear, that unless MM^T is approximately distance preserving that we are effectively scaling the learning rate of our bias $\vec{b}$. As we have discussed, because we are using a non-orthonormal normalisation M scales $\vec{c}$ in the order of $\frac{1}{\sqrt{L+T}}$. Put together, this means that MM^T is scaling $\vec{c}$ in the order of $\frac{1}{L+T}$. FITS applies a scaling of $\frac{L+T}{L}$ before outputting the forecast which partially mitigates this. However in conclusion, $\vec{b}$ still has a learning rate approximately $\frac{1}{L}$ times smaller than one would obtain through a naive parameterisation of $\vec{b}$.

As we saw, any FITS model can be expressed in the form $A\vec{x} + \vec{b}$. It is natural to ask whether this phenomena also impacts the weight matrix A. In fact it does not. As a result the issue with the bias cannot simply be resolved by increasing the learning rate as this would result in a learning rate which is too high for learning A. Let briefly sketch the reason why the weight matrix doesn't also have these issues. Crudely speaking the weight matrix W in FITS's complex linear layer and A are related via an expression of the $A = M_1WM_2$ where M_1, M_2 are real matrices corresponding to the iRFT and RFT respectively. If one chooses to normalise the iRFT by $\frac{1}{N}$ this is then offset by the fact that the right multiplication M_2 is unnormalised. In terms of backpropagation rules we have:

$$\bar{W} = M_1^T\bar{A}M_2^T$$

Therefore, whereas under a naive parameterisation we would have an update of $\bar{A}$, FITS gives us an update of $M_1M_1^T\bar{A}M_2^TM_2$. Thus, whatever normalisation standard we use for the RFT; whether we normalise the RFT (M_2) by $\frac{1}{N}$ but not the iRFT (M_1) or whether we normalise them both equally, leads the same update.

D. Further Proofs

D.1. DLinear

In Lemma 3.1 we write the padded moving average, utilised in DLinear to obtain the trend of $\vec{x}$, as a matrix multiplication $D\vec{x}$. In this part we explain the structure of D. We do this by means of an example: Consider the simple case where we have a context vector $\vec{x}$ of length 6 and we take a moving average with a kernel size of 3. In order to preserve dimension of $\vec{x}$ on must pad either side of $\vec{x}$. We do this by repeating the first and last values twice before applying the moving average. That is:

$$(x_1, x_2, x_3, x_4, x_5, x_6) \mapsto (x_1, x_1, x_2, x_3, x_4, x_5, x_6, x_6)$$

In general if we have a kernel size of K where K is odd then we must pad each side with $\frac{K-1}{2}$ repeated entries. (In DLinear they use a kernel size of 25 (Zeng et al., 2023)).

This padding operation can be expressed in matrix form. For this example

$$\begin{bmatrix} x_1 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

The moving average of the expanded $\vec{x}$ is calculated by taking the arithmetic mean of each successive run of $K = 3$ values. This may be written as a matrix multiplication:

$$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_6 \end{bmatrix} \tag{13}$$

We can combine these two operations into a single matrix multiplication to obtain

$$D\vec{x} := \frac{1}{3} \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

D.2. Closed Form Solution to Linear Regression

A well-known property of least-squares linear regression is that it admits a closed-form solution (Hastie et al., 2009). There are a number of ways in which one may compute this solution numerically. Below we define one of the more common approaches:

Definition D.1 (Closed-Form Solution)**.** Let X denote the $N \times L$ design matrix containing our training data, and let Y denote the $N \times T$ matrix of training targets. The $L \times T$ weight matrix W that minimises the training loss $\|XW - Y\|_2^2$ is given in closed form as follows:

$$W^\star = (X^T X)^{-1} X^T Y \tag{14}$$

If the rank of X is less than L, indicating that X is rank-deficient, a unique solution may not exist. In such cases, a solution can be obtained using the Moore-Penrose pseudo-inverse, denoted as $(X^T X)^+$, instead of the regular inverse.

In practice the solution given in Def. D.1 may be numerically unstable if $X^T X$ is ill-conditioned. In Section 5 we use the more stable but more expensive SVD approach. Details of this may be found in Silva (2024).

D.2.1. Closed Form Solutions for Linear Regression plus Data Normalisation

Standard least-squares linear regression admits a closed-form solution. We claimed in Section 4 that the other model families in Table 1, corresponding to RLinear and NLinear also admit a closed form solution under a least-squares loss. The reason for this is that one can formulate each of these models as linear regression on a suitable feature set of $\vec{x}$.

In the following we will let X, Y be $N \times L$ and $N \times T$ denote matrices containing N training samples and their targets respectively.

NLinear: Suppose that we wish to find the matrix $\tilde{A} \in \mathbb{R}^{T \times L}$ and bias $\vec{b} \in \mathbb{R}^{T}$ which minimises $||AX^T + \vec{b} - Y^T||_2$ subject to the condition that the rows of A must sum to one. Augment X and Y by computing the row-mean of X and subtracting this off each of the rows of both X and Y. We denote the augmented matrices as $\tilde{X}$ and $\tilde{Y}$. Specifically, for row i; $\tilde{X}_i := X_i - \mu(X_i)$ and $\tilde{Y}_i := Y_i - \mu(X_i)$. We now solve the unconstrained least-squares regression on these augmented matrices. This yields a matrix A^* and a bias $\vec{b}$ which minimises $||A\tilde{X}^T + \vec{b} - \tilde{Y}^T||_2$. This matrix is not uniquely defined since one may add any multiple of the vector $(1, 1, 1, \ldots, 1)$ to each row and obtain the same train loss since $\vec{1}\tilde{X} = \vec{0}$. Thus, we can choose to project A so that each of it's rows do sum to 1. We claim that this matrix minimises our original constrained objective $||AX^T + \vec{b} - Y^T||_2$.

Let $\vec{x}, \vec{y}$ be an arbitrary context-target pair. Let $\mu(\vec{x})_k$ be notation for a k-dimensional vector formed by taking the mean of $\vec{x}$ and repeating this k-times. Since the rows of of A sum to one then $A\mu(X)_L = \mu(X)_T$. Therefore:

$$\begin{aligned} ||(A\vec{x} + \vec{b} - \vec{y})||_2 &= ||(A\vec{x} + \vec{b} - \vec{y}) + \mu(\vec{x})_T^T - \mu(\vec{x})_T^T||_2 \\ &= ||(A\vec{x} + \vec{b} - \vec{y}) + \mu(\vec{x})_T^T - (A\mu(\vec{x})_L)^T||_2 \\ &= ||(A(\vec{x} - \mu(\vec{x})_L) + \vec{b} - (\vec{y} - \mu(\vec{x})_T)||_2 \end{aligned}$$

Therefore, any matrix A will get the same MSE on any pair $\vec{x}, \vec{y}$ as it will on the augmented versions $\vec{x} - \mu(\vec{x})_L$, $\vec{y} - \mu(\vec{x})_T)$. It follows that A^* obtains the same MSE on $\tilde{X}, \tilde{Y}$ as X, Y and vice versa. In particular if A^* is also optimal for $\tilde{X}, \tilde{Y}$ then it is too for X, Y. Thus, the matrix which we obtained by closed-form OLS on $\tilde{X}, \tilde{Y}$ satisfies the properties claimed.

RLinear: We showed that one can find a global optima for the NLinear models class in closed form when using a mean-squares loss function. The same is true for RLinear and the constuction is much the same. We wish to find a matrix A and bias $\vec{b}$ which minimise $||AX^T + \vec{b}\sigma(X) - Y^T||_2$ subject to the condition that the rows of A must sum to one. Here $\sigma(X)$ denotes an N-dimensional vector formed of the standard deviation of the rows of X. $AX^T + \vec{b}\sigma(X)$ is equivalent to augmenting X by appending $\sigma(X)$ to X as an additional column and then fitting a $(T+1) \times L$ matrix and no bias. Having appended this columns we then proceed along the same lines as before; subtracting the row means from X and Y and solving the resulting regression problem in closed form. One should be careful not to change the final column in this process and not include $\sigma(x)$ in the computation of the mean.

E. Experiment Details

Datasets: For our experiments in Section 5.2 we use 8 standard time series benchmarking datasets: ETTh1 and ETTh2: 7-channel hourly datasets (Train-Val-Test Splits [8545,2881,2881]). Their per-minute equivalents; ETTm1, ETTm2 (also 7-channel) (Train-Val-Test Splits [34465,11521,11521]). ECL, an hourly 321-channel Electricity dataset (Train-Val-Test Splits [18317,2633,5261]), Weather, a per-10-minute resolution 21-channel weather dataset (Train-Val-Test Splits [36792,5271,10540]), Traffic; an 862-channel traffic dataset (Train-Val-Test Splits [12185,1757,3509]) and Exchange: a small 8-channel finance dataset (Train-Val-Test Splits [5120,665,1422]).

In each case we use the well-established dataset divisions and normalisation protocols. We refer the reader to (Wu et al., 2021) for further details.

Models: The models we compare are DLinear, NLinear, RLinear, FITS and Linear (a single linear layer neural network). We also run FITS+IN and DLinear+IN. FITS+IN corresponds to the implementation of FITS used in Xu et al. (2023). Alongside these we run the closed-form solutions (OLS and OLS+IN). The mathematics behind these solutions are explained in Sec D.2. These are implemented using the LinearRegression model from scikit-learn using an SVD solver.

Hyperparameters: For each model, dataset, and horizon combination we train for 50 epochs using a learning rate of 0.0005 and the Adam optimizer with the default hyperparameter settings. We use a batch size of 128 in all experiments. We track the validation loss during training. At test time we load the model with minimal validation loss to evaluate on the training set, which is equivalent to early stopping. Each experiment is run (at least) 3 times using different random seeds and the standard deviation of the MSEs is computed and given in Table 2. We test on prediction horizons of 96, 192, 336 and 720 which are the standard in the literature (Nie et al., 2022). In all cases we use a context length of 720, as per the setting used by Xu et al. (2023). Our implementation of the DLinear model is taken from Zeng (2023). Our implementation of FITS is taken from Zhijian (2023). We re-implemented the RLinear and NLinear models using the detailed descriptions of these models in their respective papers (Zeng et al., 2023; Li et al., 2023).

Weight Comparison Experiments: In Section 5 we compare the weight matrices, biases and forecasts of the different models. The hyperparameter settings are largely identical to those used to populate Table 2. One difference is that we compare our weights/biases/forecasts at the end of 50 epochs of training rather than using early stopping. A second difference is that the Figure 2 shows the cosine similarity over 350 training epochs and uses a learning rate of 0.0002 rather than 0.0005. The purpose of this change was to demonstrate clearly the convergence behaviour of these models, which inevitably requires a longer training run. All figures are obtained after training on the ETTh1 dataset. For the weight, forecast and cosine similarity figures (Figures 1, 3, 2) we use a prediction horizon of 336, The bias figure (Figure 4) uses a horizon of 720.

F. Further Experiments

In Table 3, we analyse the performance of several recent state-of-the-art (SoTA) deep learning models on the datasets referenced in Table 2, comparing them to the OLS+IN column in Table 2 and an additional setting that uses L2 regularisation: OLS+IN+Reg. OLS+IN+Reg stands for Ordinary Least Squares with instance normalisation and L2 regularization (commonly known as Ridge Regression). This method is implemented using the Ridge Regression function from the SciKit-Learn library. The regularisation coefficient λ was set to 25000 for ETTh datasets and 500 otherwise (owing to the small size of ETTh, more regularisation may be required). Future work should entail performing dataset-specific hyper parameter searches with the validation splits.

Comparative analysis in Table 2 reveals that **OLS+IN+Reg** generally outperforms the standard OLS approach, particularly in smaller datasets where overfitting is a concern for simple linear models. In contrast, for larger datasets, the performance difference between Ridge Regression and standard OLS becomes negligible.

Table 3 further indicates that the performance of the OLS model is competitive with state-of-the-art models across various dataset and horizon configurations. This suggests that despite significant advances in machine learning over the past decade, the Linear Regression method implemented in SciKit-Learn remains highly effective. This observation supports the findings of (Zeng et al., 2023), highlighting the enduring relevance of traditional regression techniques in contemporary predictive modeling.

G. Further Discussion

G.1. Extracting the Weight Matrices

Suppose that we have a trained model of the form $f(\vec{x}) = A\vec{x} + \vec{b}\sigma(x)$ and we wish to determine A and $\vec{b}$. The vector of all ones has standard deviation equal to zero. Therefore passing in this vector we obtain $f(\vec{1}) = A\vec{1} = \sum_{i=1}^{L} A_{ji}$, i.e the sum of the columns of A. Let $\frac{L}{\sqrt{L-1}}\vec{e_i}$ be a multiple of the i^{th} coordinate vector $\vec{e_i}$, where the multiple is chosen so that the vector has standard deviation equal to one. Passing in this vector for f we get:

$$f(\vec{e_i}) = \frac{L}{\sqrt{L-1}}A\vec{e_i} + \vec{b}\sigma\left(\frac{L}{\sqrt{L-1}}\vec{e_i}\right) = \frac{L}{\sqrt{L-1}}A\vec{e_i} + \vec{b} = \frac{L}{\sqrt{L-1}}A_{\cdot,i} + \vec{b} \tag{15}$$

One may solve this system of equations to derive A and $\vec{b}$. In particular, $\sum_{i=1}^{L} f(\vec{e_i}) = L\vec{b} + \frac{L}{\sqrt{L-1}}\sum_{i=1}^{L} A_{\cdot,i}$. So:

$$\left(\frac{\sqrt{L-1}}{L}\sum_{i=1}^{L} f(\vec{e_i})\right) - f(\vec{1}) = (\sqrt{L-1})\vec{b}$$

Having obtained $\vec{b}$ one can then use Eqn. 15 to derive the columns of A.

Table 3. Long-term multivariate forecasting results, showing MSE values for 3 SoTA deep models.

	T	**OLS+IN**	**OLS+IN+reg**	**PatchTST**	**iTransformer**	**DAM**
ETTm1	96	0.307	0.307	0.303	0.342	0.308
	192	0.336	0.336	0.334	0.381	0.343
	336	0.365	0.365	0.364	0.418	0.351
	720	0.415	0.415	0.416	0.489	0.407
ETTm2	96	0.162	0.161	0.166	0.183	0.170
	192	0.216	0.216	0.222	0.253	0.220
	336	0.268	0.268	0.274	0.315	0.232
	720	0.349	0.349	0.361	0.412	0.325
ETTh1	96	0.375	0.366	0.372	0.393	0.367
	192	0.413	0.401	0.416	0.448	0.391
	336	0.445	0.428	0.432	0.491	0.396
	720	0.460	0.436	0.458	0.523	0.421
ETTh2	96	0.270	0.268	0.276	0.300	0.280
	192	0.331	0.329	0.339	0.381	0.338
	336	0.353	0.351	0.364	0.424	0.346
	720	0.380	0.378	0.391	0.431	0.392
ECL	96	0.133	0.133	0.129	0.149	0.154
	192	0.148	0.148	0.148	0.165	0.171
	336	0.164	0.164	0.164	0.178	0.176
	720	0.203	0.203	0.200	0.215	0.237
Traffic	96	0.385	0.385	0.360	0.393	0.460
	192	0.397	0.397	0.380	0.413	0.474
	336	0.410	0.410	0.392	0.425	0.479
	720	0.448	0.448	0.447	0.458	0.538
Weather	96	0.141	0.142	0.148	0.176	0.154
	192	0.184	0.185	0.193	0.225	0.191
	336	0.234	0.235	0.244	0.282	0.203
	720	0.307	0.307	0.315	0.361	0.280
Exchange	96	0.086	0.085	0.093	0.087	0.090
	192	0.180	0.180	0.231	0.180	0.178
	336	0.343	0.343	0.352	0.335	0.208
	720	0.992	0.968	0.992	0.854	0.893

G.2. Performance Differences Between Approaches in Table 2

Difference in Performance: Referencing Table 2, models within the same class (i.e. with or without instance norm) generally exhibit comparable performance across various datasets and forecast horizons. This is consistent with our expectations since they define identical convex optimisation problems. For example, FITS, DLinear and Linear obtain $0.165, 0.164, 0.163$ respectively on ETTm2 when $T = 96$. Typically, the DLinear and Linear models often differ only in the third decimal. FITS occasionally diverges slightly from the other models (e.g. ETTh2 for $T = 192$). This is because, as mentioned in Section 4.1 and Section C, its parameterization makes it act like a bias-free linear model. Variability in performance is more pronounced on the Exchange dataset, which is noted for its randomness and lack of predictable signals, thus providing a context where early stopping can be advantageous.

Constrained vs. Unconstrained: As shown in Table 2, OLS+IN generally outperforms the standard OLS solution. Performance is better on the ETTh1, ETTh2, ETTm2, Weather and Exchange datasets and equivalent on the remainder. Roughly speaking we observe that OLS+IN grants a modest gain on the smaller datasets [9]. This is to be expected. The weight matrix for OLS+IN is constrained whereas OLS is entirely unconstrained. This constraint can be a benefit on the smaller datasets where one risks overfitting.

[9]For the Weather dataset, unlike the others, we train a separate model for each channel, aligning with the approach suggested by (Zeng et al., 2023). This methodology effectively reduces the dataset size per model parameter, making it one of the smaller datasets in terms of sample availability per parameter.

G.3. Closed-Form versus SGD

OLS vs. SGD: Despite all models within a given class (e.g. without IN: DLinear, Linear, and FITS) theoretically converging to the same optimum under an MSE loss, practical differences can emerge based on the training approach. This subsection discusses SGD versus the closed-form OLS solution along four dimensions: early-stopping benefits, memory constraints, stability, and computational costs.

Early-Stopping: Using SGD for training provides an opportunity to perform early stopping, since one is able to monitor validation error during training, and, using this information, prevent a model from overfitting to the training data. However, our analysis indicates negligible benefits from this strategy within the examined datasets. For instance, models employing instance normalization such as OLS+IN, achieved comparable or better results than their SGD counterparts, underscoring the limited advantage of early stopping under these conditions.

Stability, Memory, and Cost: The conventional closed-form OLS solution, reliant on the inversion of $X^T X$, often faces numerical stability challenges in cases of near-singularity or poor conditioning of the matrix. We address this by incorporating a small L2 regularization term ($\lambda = 0.00001$) and opting for an SVD-based solver. Given that our datasets typically have more observations than features, the SVD solver incurs a memory cost of $\mathcal{O}(N \times L)$ and a computational cost of $\mathcal{O}(N \times L^2)$. Conversely an SGD approach will typically have the same computational complexity but with a reduced memory complexity due to the need to store only a single batch in memory at each iteration. The $\mathcal{O}(N \times L^2)$ memory cost of the SVD approach can be substantial for extensive datasets, such as those spanning several years on a per-second basis. That said, one can easily sub-sample the training data (e.g., by randomly selecting $1M$ data points) to mitigate against this: we ran experiments on the traffic and electricity datasets to validate this and observed only a negligible change in performance. Furthermore, for very large datasets, online SVD techniques may also mitigate these costs.

Speed Comparison Empirical tests reveal that the OLS solution is generally faster than training comparable models via SGD, though the exact speed differential will depend on specific hardware setups. Detailed timings for the OLS+IN solution across various datasets and horizon lengths are presented in Table G.3 compared against DLinear and FITS trained with early-stopping over 50 epochs with a batch-size of 128[10]. Results are reported as the mean and (unbiased) standard deviation across four runs, each with a different seed, on an NVIDIA GeForce RTX 2080 Ti GPU. Notably, while the OLS solution is executed on a CPU, the DLinear/FITS models are trained on a GPU. Despite this hardware disadvantage, the OLS solution consistently proves to be an order of magnitude faster. *We reiterate though that the differential will be highly contingent on the hardware*.

We used a batch size of 128 when training DLinear/FITS; for smaller batch sizes (as used in (Zeng et al., 2023)) the training time will typically be longer. In order to do early stopping one needs to evaluate the model on the validation set after each training epoch. We opt not to include the time taken to compute validation error in our computation of training time for DLinear and FITS.

Conclusion In most scenarios, the OLS solution is obtained more quickly and either matches or surpasses the performance of equivalent models (e.g., FITS) trained via SGD. The benefits of early stopping are generally minimal to nonexistent, and the higher memory demands of the SVD solver are not prohibitive using the datasets on which we have tested.

In cases involving very large datasets where memory constraints are significant, training a linear model using SGD could potentially offer advantages over a closed-form approach. Nonetheless, in such settings, we recommend employing a single linear layer network over complex alternatives, as it converges to the same optima.

G.4. Limitations and Future Work

In Section 3 we show how Linear+IN and Linear+RevIN have the same model classes. While Linear+RevIN and Linear+IN are identical in the single channel setting, they can differ subtly in the multi-channel setting. Specifically, in the setting where one shares weights of the linear layer across channels, but allows RevIN separate affine parameters per channel, then RevIN can have marginally different biases for each channel.

We wish to reiterate that our findings for FITS hold when the low-pass filter is not applied. As we have said, we are motivated to understand each model under their optimal settings, as such we have ignored the LPF which typically hinders

[10]Other hyperparameters are sourced from (Zeng et al., 2023)

Table 4. Mean Fitting Times with Unbiased Standard Deviations for OLS (SVD solver), DLinear, and FITS taken over four runs. Across all datasets OLS-SVD is substantially faster than DLinear and FITS. The high variance in training time for DLinear and FITS is a consequence of early stopping.

Model	**Horizon**	**Mean Speed (s)±Std**		
		OLS-SVD	**DLinear**	**FITS**
ETTh1	96	$3.98_{\pm 0.67}$	$18.86_{\pm 5.75}$	$149.01_{\pm 23.90}$
	192	$3.89_{\pm 0.63}$	$19.56_{\pm 5.05}$	$116.24_{\pm 88.80}$
	336	$3.68_{\pm 0.21}$	$18.20_{\pm 4.92}$	$14.23_{\pm 2.04}$
	720	$4.31_{\pm 0.53}$	$27.64_{\pm 6.36}$	$34.82_{\pm 9.77}$
ETTh2	96	$3.70_{\pm 0.61}$	$32.22_{\pm 6.59}$	$150.46_{\pm 30.94}$
	192	$3.66_{\pm 0.41}$	$21.76_{\pm 12.45}$	$155.97_{\pm 36.64}$
	336	$3.81_{\pm 0.48}$	$27.43_{\pm 14.72}$	$179.89_{\pm 11.63}$
	720	$4.01_{\pm 0.60}$	$23.57_{\pm 9.03}$	$202.73_{\pm 11.70}$
ETTm1	96	$14.29_{\pm 1.02}$	$144.78_{\pm 17.45}$	$427.42_{\pm 213.52}$
	192	$15.05_{\pm 1.92}$	$120.14_{\pm 32.60}$	$315.82_{\pm 186.93}$
	336	$15.24_{\pm 1.46}$	$47.69_{\pm 15.11}$	$425.83_{\pm 137.68}$
	720	$16.66_{\pm 0.95}$	$90.54_{\pm 24.68}$	$13.69_{\pm 7.59}$
ETTm2	96	$13.98_{\pm 0.74}$	$131.11_{\pm 22.51}$	$147.45_{\pm 45.43}$
	192	$14.10_{\pm 0.36}$	$127.11_{\pm 22.87}$	$196.34_{\pm 48.95}$
	336	$14.78_{\pm 0.44}$	$141.68_{\pm 17.00}$	$235.09_{\pm 12.54}$
	720	$16.49_{\pm 1.19}$	$94.63_{\pm 15.19}$	$209.55_{\pm 25.68}$
Weather	96	$48.60_{\pm 4.44}$	$1081.91_{\pm 786.08}$	$579.84_{\pm 4.75}$
	192	$49.12_{\pm 3.72}$	$1563.82_{\pm 1120.49}$	$550.85_{\pm 26.82}$
	336	$51.18_{\pm 4.16}$	$1120.99_{\pm 880.48}$	$639.76_{\pm 55.51}$
	720	$59.67_{\pm 6.77}$	$1574.74_{\pm 870.11}$	$708.11_{\pm 52.76}$

performance. When one uses an LPF there will typically be restrictions on the model class meaning it is not equivalent to unconstrained linear regression. A further analysis of this is required in future work.

One of the key contributions of FITS (Xu et al., 2023) is that it allows one to compress models by disregarding higher frequencies during training. Having established how to map between FITS models and their underlying affine representations, this opens the possibility of using the FITS technique to compress OLS solutions post hoc. This is something which should be looked at in future work.