Random Initialization Can't Catch Up: The Advantage of Language Model Transfer for Time Series Forecasting

Roland Riachi [*1 2] **Kashif Rasul** [3] **Arjun Ashok** [1 2] **Prateek Humane** [1 2] **Alexis Roger** [1 4]
Andrew R. Williams [*1 2] **Yuriy Nevmyvaka** [3] **Irina Rish** [1 2]

Abstract

Recent works have demonstrated the effectiveness of adapting pre-trained language models (LMs) for forecasting time series in the low-data regime. We build upon these findings by analyzing the effective transfer from language models to time series forecasting under various design choices including upstream post-training, time series tokenizer and language backbone size. In the low-data regime, these design choices have a significant impact on the validation loss, with clear-cut choices that outperform others. Contrary to (Hernandez et al., 2021), we observe that the validation loss of the LMs continues to smoothly decrease long after the validation loss of the randomly initialized models has converged, leading to a *non-vanishing transfer gap* that holds across design choices. These findings not only help shed light on the effective use of compute-efficient training for time series, but also open the way for the study of *modality-agnostic* properties of data distributions leveraged by these models[1].

1. Introduction

Transfer learning enables the adaptation of models trained in one domain to downstream tasks in another. With the rise of large foundation models trained on massive text corpora, many specialized models have been created by fine-tuning these models for specific downstream applications (Bommasani et al., 2022). Notably, it has been proven possible to achieve state-of-the-art results on downstream tasks significantly different from the original setting (Raffel et al., 2019).

[*]Equal contribution [1]Mila - Québec AI Institute [2]University of Montreal [3]Morgan Stanley [4]McGill University. Correspondence to: Roland Riachi <roland@riachi.com>, Andrew R. Williams <andrew.williams@umontreal.ca>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

[1]Trained checkpoints and code will be released upon acceptance.

(Lu et al., 2021) explore this phenomenon through various synthetic and real world tasks. In this paper, we extend this line of work to univariate time series forecasting. Similarly to text generation, time series forecasting is a sequence prediction problem, albeit with numerical values. This transition from discrete text tokens to continuous numerical values introduces new challenges in tokenization, embedding, and output representation.

Several recent studies have attempted to apply LLMs to time series forecasting using different techniques. Reprogramming-based methods, such as that of Jin et al. (2023), train an encoder to align input summary statistics with text-like representations. Other works explore zero-shot forecasting by directly feeding digitized time series data into LLMs as natural language inputs (Gruver et al., 2024; Requeima et al., 2024; Williams et al., 2024). Still others investigate parameter-efficient fine-tuning strategies. For instance, Zhou et al. (2023) adapt normalization layers of GPT2, while Ansari et al. (2024) train T5 models using bin-based tokenization. Their results show limited gains from pretrain initializing from pre-trained language models, in contrast to Bayazi et al. (2024), who find positive transfer from T5 to EEG signals, which suggests that the properties of the time series data may have an impact on whether transfer occurs.

In this paper, we seek to contribute to this discussion by examining the transferability of language models to univariate time series forecasting. Specifically, we ask:

1. What is the most effective way to encode time series data into a latent space?
2. How does the upstream model's performance influence downstream forecasting accuracy?

To this end, we present the following contributions:

1. We demonstrate that transferring from pre-trained weights is more suitable in almost every scenario we test. Regardless of tokenization/embedding techniques or model size, models initialized with the weights of pre-trained language models outperform their randomly initialized counterparts.

2. We quantify the amount of additional data samples necessary for a model trained from scratch to achieve the same loss on time series as a model initialized with pre-trained LLM weights. In particular, our results differ from those of (Hernandez et al., 2021), in that there exists a "transfer gap" such that no amount of additional tokens would enable a randomly initialized model to perform as well as the pretrained language initialization.
3. We find that upstream model performance is not directly correlated with transfer to time series forecasting. Namely, while scaling parameter count increases effective transfer, upstream instruction tuning on language tasks results in worse downstream validation loss.

2. Experiments/Methodology

To study the effective transfer of language to time series, we experiment with the T5[2] encoder-decoder transformer as the LLM backbone that we adapt to the time series modality. Our choice of T5 LLM is inspired by the initial success of (Bayazi et al., 2024) with this model on EEG time series. The choice of LLM backbone is quite important as it may significantly affect the transfer, as we observed in our earlier experiments trying different language models; however, a systematic ablation study with respect to the LLM backbone choice is out of scope of this paper.

The adaptation of LLM backbones to time series requires several architectural modifications, which we investigate. First, we compare the performance of three different **weight initialization** strategies for the backbone. Second, we replace the **input tokenization** and embedding layers to handle continuous-valued inputs. Third, we vary the **backbone size**.

All models use the same output distribution as in (Ansari et al., 2024) and are trained using the LOTSA dataset (see Appendix A of (Woo et al., 2024)) across 3 random seeds. For details regarding tokenizer design as well as initializations, see Appendix C.

2.1. Effect of Backbone Initialization

To assess the effective transfer from language weights to time series, we initialize the T5 architecture with three different sets of weights before fine-tuning with time series data:

1. **Random initialization**: The baseline against which we compare transfer from language models is a random initialization according to Hugging Face defaults.
2. **Purely pre-trained initialization**: We measure transfer from language by initializing the model with the weights of T5-Efficient (Tay et al., 2022).
3. **Instruction-tuned initialization**: We use the weights of Flan-T5 (Chung et al., 2022), an instruction-tuned version of T5.

[2]We use the efficient variants: https://huggingface.co/google/t5-efficient-base.

2.2. Effect of Tokenizer

A critical aspect of adapting transformer-based models for time series is the transformation of continuous time observations into tokens. Given a time series sequence $\{x_t\}_{t=1}^{T}$ of length T, we aim to produce a sequence of tokens $\{s_t\}_{t=1}^{T'} \in \mathbb{R}^{d_{token} \times T'}$, where each token is of dimension d_{token} and $T' \leq T$ denotes the length of the token sequence[3]. We test three tokenization methods:

1. **Naive tokenization**: The most straightforward approach; each time series observation is directly treated as a token, i.e., $s_t = x_t$.
2. **Lag tokenization**: To incorporate seasonal dependencies, Rasul et al. (2024) constructs token vectors using current and past time series values based on a set of lag indices $\mathcal{L} = \{\ell_1, \ldots, \ell_p\}$. Each token vector $s_t \in \mathbb{R}^{d_{token}}$ is given by $s_t = [x_t \; x_{t-\ell_1} \; \ldots \; x_{t-\ell_p}]^T$.
3. **Bin tokenization**: To better align time series data with natural language tokens, Ansari et al. (2024) bins time series values into B linearly-spaced bins within the range $[-a, a]$. Each token is assigned an index based on its bin.

2.3. Effect of Backbone Size

Lu et al. (2021) find that performance scales reliably with model size for frozen pretrained language transformers adapted to CIFAR-10. To investigate the effect of model size on transfer to time series, we test three sizes: **small** (60M), **base** (220M), and **large** (770M). Intuitively, we hypothesize that using larger models leads to lower loss and more effective transfer in the low-data regime since larger models require more training to converge (Kaplan et al., 2020); as a result, we expect effective transfer in the low-data regime to increase as model size increases.

3. Results

Do pre-trained weights lead to better performance than randomly initialized weights?

Figure 1 demonstrates that language weights, both pre-training only (pink) and instruction-tuned (yellow), exhibit

[3]Prior to tokenization, time series entries are normalized using the standard deviation across the input window to ensure consistent scales across different inputs. We consistently employ the same pre-processing steps to compare tokenization methodologies in isolation.

positive transfer against the random initialization (green). In particular, at the end of training, both the pre-trained and instruction-tuned models exhibit a **non-vanishing transfer gap** that persists even as training concludes. While the performance of randomly initialized models plateaus, the performance of the language and instruction-tuned models continues to improve. This behavior contrasts with the findings of Hernandez et al. (2021), where the transfer gap vanishes as the number of relevant training tokens increases.

However, in the low-data regime, the pre-trained weights and the instruction-tuned weights exhibit very different behaviours. Whereas the pretrained model immediately and consistently outperforms the random initialization, the Flan-T5 initialization performs worse than the random initialization for the first 6M tokens. As training progresses, the validation loss of the Flan-T5 initialization gradually converges towards that of the T5 model, although the curve qualitatively appears to be much noisier and spikier.

Simply choosing the best language model does not immediately return the best downstream performance for time series forecasting. Instead, these results suggest that post-training may have a complex impact on the representation space induced by the model's parameters, which improves language performance at the expense of a more jagged representation landscape that is less accommodating to transfer (Mosbach et al., 2021).

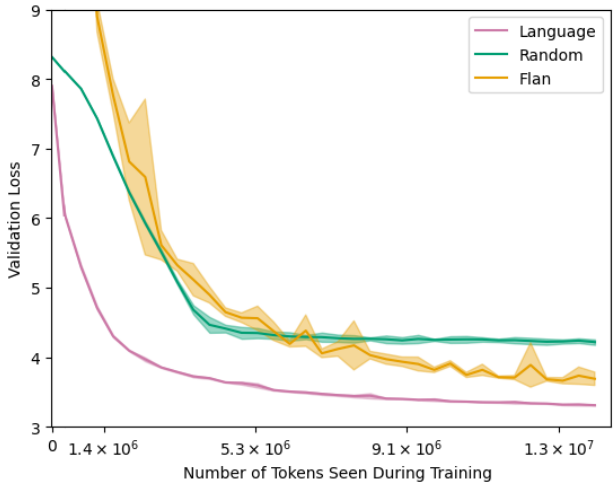

Figure 1. **Validation losses for different weight initializations:** pre-training only, or "Language" (pink); pre-training + instruction-tuning, or "Flan" (yellow); and random initialization (green). Both pre-training and instruction tuned weights quickly outperform the random initialization. Furthermore, even though the Flan-T5 initialization outperforms solely pre-trained models on language tasks (see Table 5 of (Chung et al., 2022)), Flan-T5 performs worse than the random initialization for the first 6M time series training tokens, and performs worse than the solely pre-trained initialization across the entire training duration.

How do tokenization schemes affect effective-transfer?

Figure 2 (a) shows that transfer is beneficial regardless of tokenization/embedding scheme, but that the chosen tokenizer affects the validation loss and the convergence speed.

The bin tokenizer with pre-trained language weights (solid green line) achieves the smallest validation loss while converging smoothly and quickly. We conjecture that this is because bin tokenization effectively shrinks the vocabulary and recasts forecasting as a classification task, which natu-

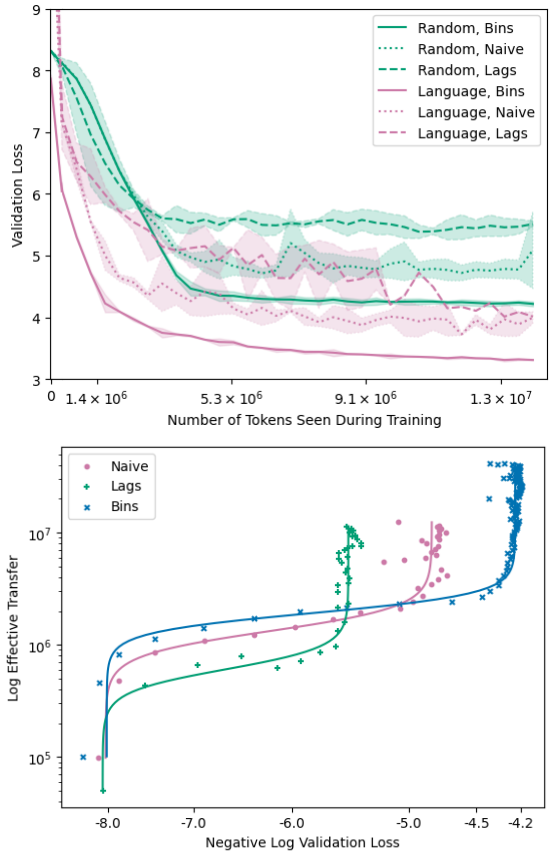

Figure 2. (a) **Validation losses across tokenizers**. Naive and lag tokenizations yield significantly worse zero-shot, i.e. initial, validation loss with pre-trained weights against random initializations. Nevertheless, after 1M time series training tokens, pre-trained models all have lower loses than their randomly initialized counterparts. Notably, each pre-trained model eventually achieves similar or lower validation loss compared to every randomly initialized model. (b) **Effective transfer across tokenizers for T5 220M**. Defined as the difference in training tokens required to obtain a given validation loss (see Appendix D), effective transfer quantifies the additional number of training tokens necessary for the randomly initialized model to match the validation loss of the model initialized from pre-trained weights. Vertical asymptotes occur when the models with the randomly initialized weights converge before their pre-trained counterparts. We observe that log effective transfer increases with lower log validation loss, meaning that the models initialized from pre-trained weights converge model quickly than randomly initialized models.

rally aligns with the expectations of language models. This is supported by Chronos, which attributes its strong performance to the natural compatibility of quantization-based tokenization with language model architectures (Ansari et al., 2024). This may also explain why the loss curves for the bin tokenizer are smoother than for the lag and linear tokenizers.

The lag tokenizer has the highest validation loss and is slow to converge, despite having access to the additional information provided by the lags. This suggests that the model may be struggling to make use of this extra information, or that the information is not useful.

We also note that, across all three tokenizers, the pretrained initializations outperform their respective counterparts with random initializations. Figure 2 (a) clarifies that initializing the model from pre-trained language weights leads to meaningfully lower validation losses when compared with random weight initializations for all three tokenization methods, while figure (b) shows that this transfer is present in the low-data regime as well as at the end of training.

How does model size affect effective transfer?

Consistent with Hernandez et al. (2021), we observe that increasing the number of model parameters influences the performance disparity between weight initializations. Figure 3 demonstrates that scaling model size with pre-trained initializations improves downstream performance on time series data in the low-data regime: the differences in validation loss between model sizes are most pronounced at the start of training, which aligns with the conjecture that larger models exhibit higher effective transfer in the low-data regime (Hernandez et al., 2021). In particular, models initialized with random weights exhibit an inverse relationship between scale and validation loss, highlighting the greater training needs of larger models starting from scratch.

We also note that the validation loss of the models with language initializations smoothly continues to improve with training, in contrast to the randomly initialized models which converge at a higher loss value and earlier. This suggests that the language initialization meaningfully affects the loss landscape and training dynamics, resulting in asymptotically lower, predictably decreasing loss values on the downstream task of time series prediction.

4. Conclusion and Future Work

We find that training the T5 architecture on time series data performs better when initializing from pretrained language weights than from random weights. These results hold across different experimental setups, including whether the language weights were post-trained or not, different choices of tokenizers and various model sizes.

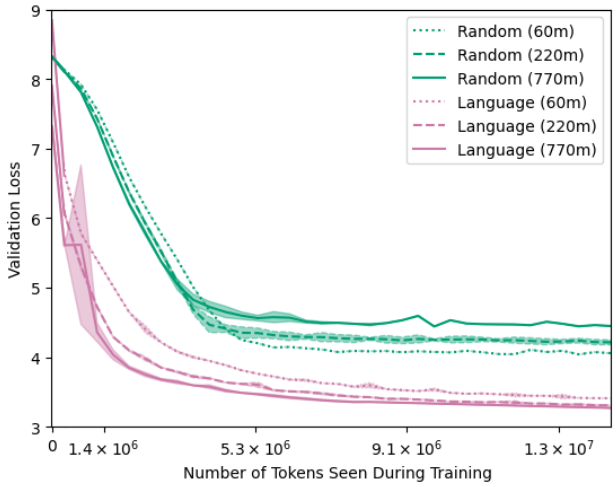

Figure 3. **Validation losses across backbones sizes.** Validation losses of models initialized with language weights decrease as model size increases, whereas those of randomly initialized do not. Moreover, across all model scales, the randomly initialized models converge early in training while the models with pre-trained weights did not converge within the length of training tested. In the low-data regime, pretrained larger models outperform pretrained smaller models, but this difference tapers out as training continues.

We hypothesize that transfer is effective because pretraining biases the model toward a region of parameter space that is already well-suited for learning temporal patterns. While our experiments are limited to the T5 model family, we aim to test transfer across different language model backbones in future work to control for the impact of the architecture.

Another possibility is that time series data is distributionally similar to language data, and the benefit arises simply due to increased training data. To investigate this, we plan on controlling for total training data size in future work.

Interestingly, the Flan-T5 initialization underperforms against the T5 initialization. Therefore, further upstream tuning of language weights to improve language performance (Wei et al., 2022) does not translate to improvements in effective transfer to time series. We plan to study this by ablating upstream datasets and fine-tuning approaches.

Finally, it is possible that the model learns to map time series data into a representation space similar to that of language. To investigate this, we plan to use statistical analysis tools, such as PQMass (Lemos et al., 2024) to compare the distributions of time series and language representations before and after encoding.

Ultimately, we hope to clarify whether transfer is effective because the models are trained with more data or because language and time series tasks share underlying structural similarity.

References

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Wilson, A. G., Bohlke-Schneider, M., and Wang, Y. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

Bayazi, M. J. D., Ghonia, H., Riachi, R., Aristimunha, B., Khorasani, A., Arefin, M. R., Darabi, A., Dumas, G., and Rish, I. General-purpose brain foundation models for time-series neuroimaging data. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P. On the opportunities and risks of foundation models, 2022. URL https://arxiv.org/abs/2108.07258.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large language models are zero-shot time series forecasters, 2024. URL https://arxiv.org/abs/2310.07820.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. Scaling laws for transfer, 2021.

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., and Wen, Q. Time-llm: Time series forecasting by reprogramming large language models, 2023.

Kaplan, J., McCandlish, S., Henighan, T. J., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020. URL https://www.arxiv.org/abs/2001.08361.

Lemos, P., Sharief, S., Malkin, N., Salhi, S., Stone, C., Perreault-Levasseur, L., and Hezaveh, Y. Pqmass: Probabilistic assessment of the quality of generative models using probability mass estimation. *arXiv preprint arXiv:2402.04355*, 2024.

Lu, K., Grover, A., Abbeel, P., and Mordatch, I. Pretrained transformers as universal computation engines, 2021. URL https://arxiv.org/abs/2103.05247.

Mosbach, M., Andriushchenko, M., and Klakow, D. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines, 2021. URL https://arxiv.org/abs/2006.04884.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Rasul, K., Ashok, A., Williams, A. R., Ghonia, H., Bhagwatkar, R., Khorasani, A., Bayazi, M. J. D., Adamopoulos, G., Riachi, R., Hassen, N., Biloš, M., Garg, S., Schneider, A., Chapados, N., Drouin, A., Zantedeschi, V., Nevmyvaka, Y., and Rish, I. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2024. URL https://arxiv.org/abs/2310.08278.

Requeima, J., Bronskill, J., Choi, D., Turner, R., and Duvenaud, D. K. Llm processes: Numerical predictive distributions conditioned on natural language. *Advances in Neural Information Processing Systems*, 37:109609–109671, 2024.

Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., and Metzler, D. Scale efficiently: Insights from pre-training and fine-tuning transformers, 2022. URL https://arxiv.org/abs/2109.10686.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners, 2022. URL https://arxiv.org/abs/2109.01652.

Williams, A. R., Ashok, A., Marcotte, É., Zantedeschi, V., Subramanian, J., Riachi, R., Requeima, J., Lacoste, A., Rish, I., Chapados, N., et al. Context is key: A benchmark for forecasting with essential textual information. *arXiv preprint arXiv:2410.18959*, 2024.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. Unified training of universal time series forecasting transformers, 2024. URL https://arxiv.org/abs/2402.02592.

Zhou, T., Niu, P., Wang, X., Sun, L., and Jin, R. One fits all:power general time series analysis by pretrained lm, 2023.

A. Additional Results and Visualizations

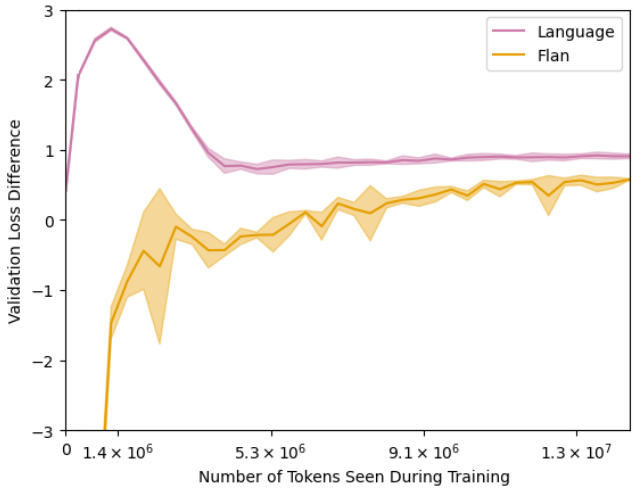

Figure 4. Difference in validation loss (pre-trained minus language) across weight initializations for the 220M parameter T5 backbone.

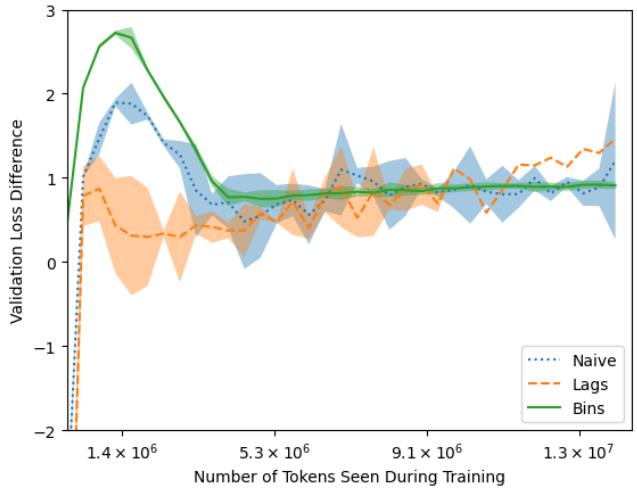

Figure 5. Difference in validation loss (random minus language) across tokenizers for the 220M parameter T5 backbone.

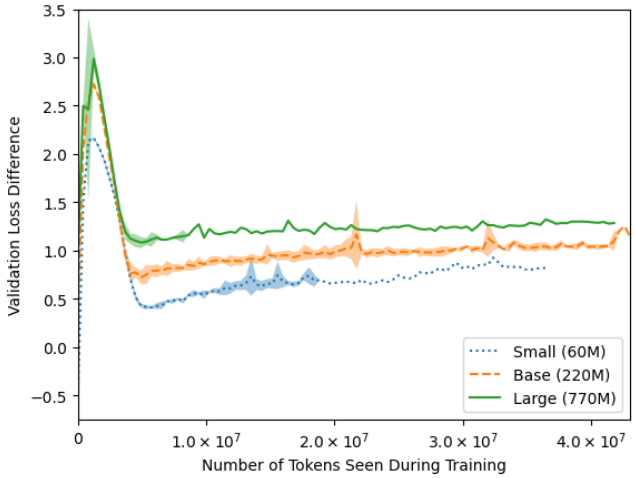

Figure 6. Difference in validation loss (random minus language) across model backbone scales.

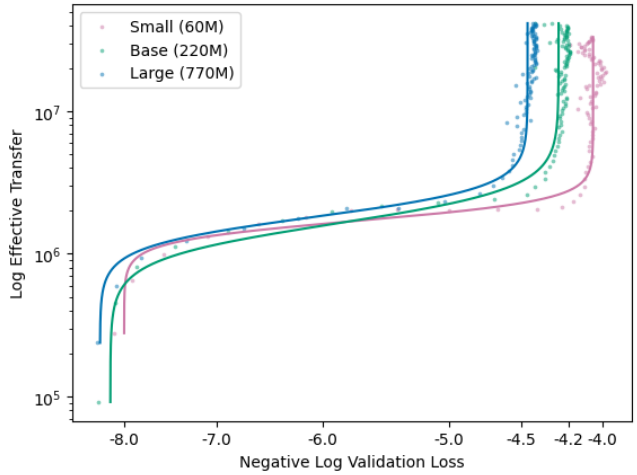

Figure 7. Effective transfer across model backbone scales.

B. Hyperparameter Search

When using a pre-trained model for downstream tasks, choices of hyperparameters are different compared to when training a model from scratch. To avoid unfair comparisons caused by a given choice of hyperparameters, we perform a sweep across common settings for the 220M parameter T5 backbone with the bin tokenizer. Figure 8 shows the mean validation loss curve with standardard deviations across all such configurations. Moreover, Figure 9 shows the individual validation loss curves (transparent) across for all these configurations along with the mean validation loss curve for both language and random weights (opaque). As can be seen from both figures, initializing the model with pre-trained weights consistently yields lower validation losses than random initializations, across most choices of hyperparameters.

We ablate across the following choices of hyperparameters:

- learning rate: 0.0001, 0.0005, 0.001
- batch size: 64, 128
- weight decay: 0, 0.01, 0.1
- warm-up duration (as a percentage of maximum training duration): 0, 0.5%, 1%, 2%

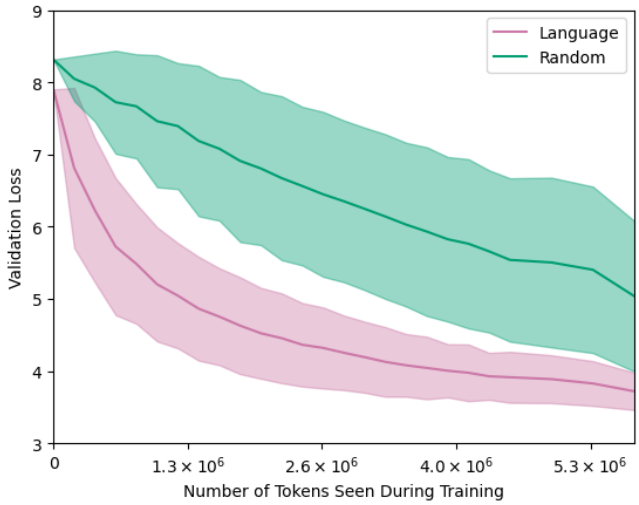

Figure 8. Mean validation loss across hyperparameter configurations.

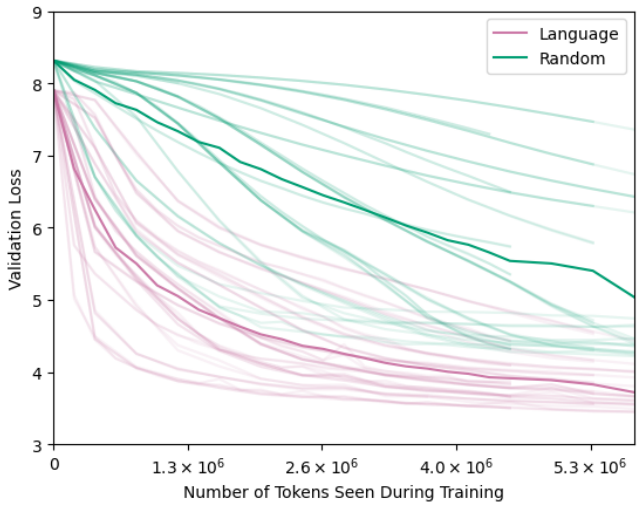

Figure 9. Validation losses across hyperparameter configurations.

C. Embeddings

As mentioned above, language data is discretely valued whereas time series data is continuously valued. This distinction affects how we usually embed tokens for LLMs, which explicitly assign learnable embedding vectors e_t to each token s_t.

C.1. Embeddings Architectures

We employ different embedding methods for discrete and continuous time series tokenization methods:

1. Discrete mappings (bins): We apply a standard embedding layer, where the embedding e_t for token $s_t = i$ is a lookup from an embedding matrix based on the bin index i.
2. Continuous mappings (naive, lags): We apply a linear transformation to the token vectors:

$$e_t = W_e s_t + b_e,$$

where $W_e \in \mathbb{R}^{d_{model} \times d_{token}}$ and $b_e \in \mathbb{R}^{d_{model}}$.

C.2. Embeddings Pre-Trained Initializations

For each tokenization approach, the embedding layers of the pre-trained model cannot be directly re-used, and instead require an intermediate processing step:

1. Discrete mapping: Initialize the embedding matrix to the first $B = 4096$ vocabulary vectors of the pre-trained backbone.
2. Continuous mapping: Each row of the matrix W_e is initialized to the mean vocabulary vector of the pre-trained model, while the bias b_e is initialized as the zero vector.

D. Effective Transfer

Let $\mathcal{L}_R(d)$ and $\mathcal{L}_P(d)$ denote the validation losses achieved after training for d time series tokens using random and pre-trained initializations, respectively. For a given validation loss ℓ, $\mathcal{L}_\cdot^{-1}(\ell)$ therefore denotes the amount of data required to train a model to reach the loss level ℓ begining from a given initialization. We define transfer as "effective" or "positive" when starting the timeseries model training with the language model's weights allows us to achieve the same validation loss with less data than a model initialised randomly. This data quantity difference is what we refer to as the amount of data we "saved". Concretely, the *effective data transferred* $D_T(\ell)$ for a target loss ℓ is defined as the difference of these data amounts:

$$D_T(\ell) := \mathcal{L}_R^{-1}(\ell) - L_P^{-1}(\ell).$$

A positive $D_T(\ell)$ indicates that initializing the model from pre-trained language weights requires fewer training examples to reach the loss ℓ compared to random initialization, signifying an effective transfer from the upstream task to time series forecasting.