TiCT: A Synthetically Pre-Trained Foundation Model for Time Series Classification

Anonymous authors
Paper under double-blind review

Abstract

The ubiquity of time series data creates a strong demand for general-purpose foundation models, yet developing them for classification remains a significant challenge, largely due to the high cost of labeled data. Foundation models capable of in-context learning (ICL) offer a powerful solution, adapting to new tasks with minimal examples and reducing the need for extensive retraining. However, prior work on large-scale time series models has predominantly focused on forecasting, leaving a critical gap for versatile, fine-tuning-free classification. To address this, we introduce TiCT (Time-series in-Context Transformer), a transformer-based model pre-trained exclusively on synthetic data to perform in-context classification. We make two primary technical contributions: 1) a novel architecture featuring a scalable bit-based label encoding and a special output attention mechanism to handle an arbitrary number of classes; and 2) a synthetic pre-training framework that combines a Mixup-inspired process with data augmentation to foster generalization and noise invariance. Extensive evaluations on the UCR Archive show that TiCT achieves competitive performance against state-of-the-art supervised methods. Crucially, this is accomplished using only in-context examples at inference time, without updating a single model weight. The source code is available at: `https://sites.google.com/view/tsicl`.

Track: Research

1 Introduction

Foundation models with in-context learning (ICL) can adapt to new tasks using only a few labeled examples, eliminating the need for task-specific training (Hollmann et al., 2022; Garg et al., 2022). While large-scale time series models have focused on forecasting (Ansari et al., 2024; Lu et al., 2024; Hoo et al., 2025; Taga et al., 2025), ICL for *classification* (Fig. 1) remains underexplored.

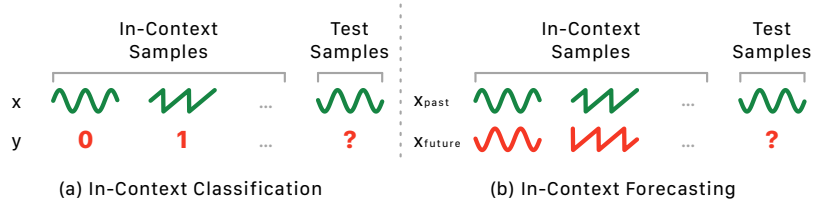

Figure 1: a) ICL for Classification, the focus of this work, predicts the class of a query series by conditioning on a context of labeled examples. b) ICL for Forecasting, the focus of most prior work, uses a historical context to predict future time series values.

We introduce TiCT (Time-series in-Context Transformer), the first foundation model for ICL time series classification trained purely on synthetic data. Pre-training on synthetic data is advantageous: it overcomes data scarcity limitations and has been validated as a successful strategy for ICL models in adjacent domains (Hollmann et al., 2022; Taga et al., 2025). Our approach addresses two key challenges: (1) handling variable class counts (existing ICL classifiers like tabPFN (Hollmann et al., 2022; Thomas et al., 2024; Hollmann et al., 2025) are limited to fixed, small class numbers), and (2) generalizing from synthetic to real-world signals.

Contributions. (1) A scalable architecture with bit-based label encoding and output attention that handles arbitrary class counts while enforcing label equivariance (Garg et al., 2022; Boix-Adsera et al., 2023) (Section A.2). (2) A Mixup-inspired synthetic pre-training framework (Zhang et al., 2017; Wickstrøm et al., 2022) with data augmentations (Wen et al., 2020; Iwana & Uchida, 2021; Yeh et al., 2023) that learns noise invariance in a data-driven manner, replacing hand-crafted functions like Dynamic Time Warping. (3) Evaluation on 128 predominantly real-world UCR Archive datasets (Dau et al., 2019) showing TiCT matches supervised methods (ResNet, Transformer) without any fine-tuning, while being 14-34$\times$ faster by eliminating per-task training.

Code, pre-trained models, and experimental results are available at `https://sites.google.com/view/tsicl`.

2 Methodology

2.1 Problem Setup

An ICL classifier predicts class labels y_{test} for test samples x_{test} by conditioning on labeled context examples $\mathcal{D}_{\text{context}} = \{(x_i, y_i)\}_i^k$ (Hollmann et al., 2022). The model must generalize to arbitrary class counts and treat labels as abstract symbols (label equivariance).

2.2 Architecture

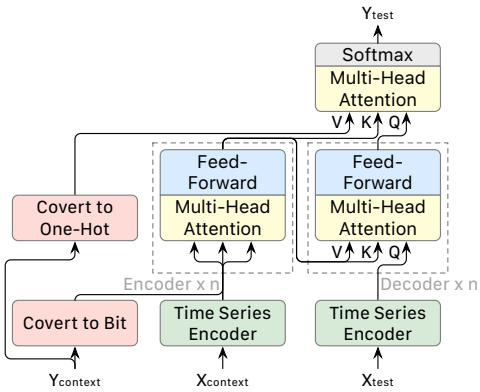

Figure 2: An overview of the TiCT architecture. The model processes in-context samples and labels through an encoder and uses a decoder to relate them to the test sample. A final output attention mechanism computes class probabilities by attending to the in-context labels.

TiCT (Fig. 2) consists of: (1) an input time series encoder (ResNet (Wang et al., 2017; Ismail Fawaz et al., 2019; Yeh et al., 2023) or Transformer (Vaswani et al., 2017)), (2) a Transformer encoder-decoder, and (3) output attention for classification.

Input Representation and Encoding. All input time series (X_{context} and X_{test}) are transformed into fixed-dimensional embeddings using a shared encoder. The in-context labels, Y_{context}, are encoded in two distinct formats. First, a compact *bit representation* of each label is concatenated with its corresponding time series embedding for the Transformer encoder input. Second, a *one-hot representation* of the labels, $Y_{\text{one-hot}}$, is prepared for the output attention layer.

For a model to generalize across ICL tasks, it must treat labels as abstract, interchangeable symbols rather than learning intrinsic meanings (Garg et al., 2022; Boix-Adsera et al., 2023). For encoding discrete labels, three natural strategies exist spanning the design space: numerical encoding (single integer, most compact), one-hot encoding (sparse C-dimensional vector, standard in classification), and bit representation (dense $\lceil \log_2 C \rceil$-dimensional binary vector, our proposed middle ground). We chose bit representation based on three criteria: (1) *Discrete symbolic nature*: Unlike numerical encoding (Hollmann et al., 2022; 2025), bit representation enforces discrete semantics. (2) *Compactness*: Bit vectors ($\lceil \log_2 C \rceil$ dimensions) are exponentially smaller than one-hot vectors (C

dimensions). (3) *Learning efficiency*: Each bit has 0.5 activation probability vs. $\frac{1}{C}$ for one-hot, providing denser gradients. Theoretical comparison in Section A.1; empirical evaluation in Section 3.1 confirms bit encoding converges dramatically faster.

Transformer Encoder-Decoder Processing. The context embeddings combined with label bits are processed by the Transformer encoder to produce H_{context}. The test embedding and H_{context} are fed into the Transformer decoder to produce H_{test}. To ensure that each test sample is predicted independently, we remove the self-attention mechanism from the decoder.

Output Attention and Label Equivariance. The final classification uses output attention with H_{test} as query (Q), H_{context} as keys (K), and $Y_{\text{one-hot}}$ as values (V). The output logits are $l = \sum_{i=1}^{k} \alpha_i Y_{\text{one-hot},i}$ where $\alpha = \text{Softmax}(QK^T/\sqrt{d_k})$. For any class c, its logit $l[c]$ equals the sum of attention weights for all samples with that label, ensuring label equivariance (proof in Section A.2).

2.3 Synthetic Pre-training

The pre-training framework teaches in-context classification through diverse, synthetically generated tasks. Our core algorithm (detailed in Section B) leverages specialized time series augmentations (Wen et al., 2020; Iwana & Uchida, 2021; Yeh et al., 2023) within a Mixup-based framework (Zhang et al., 2017; Wickstrøm et al., 2022) to capture invariant temporal features.

Synthetic Task Generation. For each task, we generate two time series shape templates using KernelSynth (Ansari et al., 2024), creating a binary classification problem. We intentionally use only two templates to construct a single, clear decision boundary, hypothesizing that if a model can discriminate between two classes, this capability generalizes to multi-class problems, analogous to SVMs. The one-hot labels are fixed to $[1, 0]$ and $[0, 1]$, but bit-based labels are randomized binary vectors. Training on randomized bit vectors forces the model to treat labels as abstract symbols, enabling generalization to $2^{n_{\text{bit}}}$ classes.

Promoting Generalization. To enhance generalization, we introduce two sources of randomness. First, a class imbalance threshold $t \sim \mathcal{U}(0.1, 0.9)$ simulates real-world skewed distributions. Second, a pipeline of 0 to n_{max} augmentations (jittering, smoothing, slope, spike, step, cropping, masking, shifting, time warping) is randomly selected. Each sample is created by mixing the two templates with ratio $r \sim \mathcal{U}(0, 1)$, applying augmentations, and assigning a label based on whether r exceeds t. Each generated dataset is split in half: the first half as context examples $(X_{\text{context}}, Y_{\text{context}})$, the second half as test queries $(X_{\text{test}}, Y_{\text{test}})$. The model is trained end-to-end with cross-entropy loss.

3 Experiments

3.1 Label Representation Study

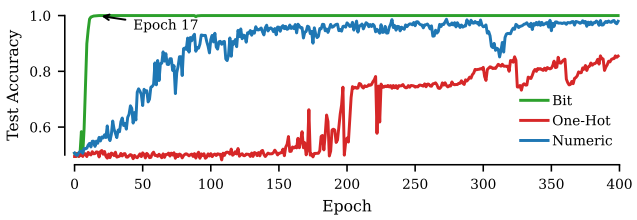

Figure 3: Test accuracy over 400 training epochs for three label representation strategies. The bit representation demonstrates significantly faster convergence and superior final performance, reaching perfect accuracy by the 17th epoch.

We validate our bit-based label encoding on a synthetic binary matching task where models must identify context items matching a query label from a universe of $2^8 = 256$ classes (Fig. 3). Bit encoding converges dramatically faster than numerical and one-hot alternatives, reaching near-perfect test accuracy by the 17th epoch (vs. >100 epochs for others). This validates our hypothesis that compact, dense representations (0.5 activation probability per dimension) simplify learning sym-

bolic label relationships compared to sparse one-hot vectors ($1/C$ activation probability). Full experimental details in Section C.1.

3.2 UCR Archive Evaluation

We evaluate on all 128 UCR Archive datasets (Dau et al., 2019), consisting predominantly of real-world time series from healthcare, sensors, and industrial monitoring. We use official train/test splits with $k = 64$ context examples retrieved via z-normalized Euclidean distance. No hyperparameter tuning is performed. Evaluation protocol details in Section C.2. Full per-dataset results are available at the project website.

Table 1: Performance and efficiency comparison on 128 UCR Archive datasets. TiCT is a single foundation model, while ResNet and Transformer are fully supervised (128 trained models each).

	Classic		Supervised Deep Learning		ICL
	1NN-ED	1NN-DTW	ResNet	Transformer	TiCT
Avg. Rank	3.55	3.15	2.86	2.58	2.86
Total Time (s)	2.7	363,198	3,387	8,170	242.5

Table 2: Ablation study of TiCT on the UCR Archive. We report the average rank (lower is better).

Model Variant	**Encoder**	**Mixup**	**Augment.**	**Imbalance**	**Avg. Rank**
TiCT - Augmentation	ResNet	Yes	No	Yes	5.01
TiCT - Mixup	ResNet	No	Yes	Yes	4.25
TiCT - Imbalance	ResNet	Yes	Yes	No	3.39
TiCT (Transformer)	Transformer	Yes	Yes	Yes	3.17
TiCT (ResNet, 2M)	ResNet	Yes	Yes	Yes	3.13
TiCT-Large (ResNet, 47M)	ResNet	Yes	Yes	Yes	**2.04**

Main Results. Table 1 shows TiCT-Large (single foundation model) achieves Rank 2.86, matching fully-supervised ResNet (2.86) and competitive with Transformer (2.58), while being 14-34× faster (total runtime 242.5s vs. 3,387-8,170s) and 1,500× faster than DTW. Wilcoxon tests confirm no significant difference in accuracy, validating that ICL without fine-tuning matches task-specific supervised learning. This demonstrates practical efficiency advantages: as tasks scale beyond 128, TiCT's fixed pre-training cost amortizes while supervised training scales linearly.

Ablation Study. Table 2 validates the necessity of each component; forgoing augmentations severely degrades performance (from 3.13 to 5.01), underscoring their role in facilitating synthetic-to-real transfer. We find that Binary Mixup outperforms multi-class variants (3.13 vs. 4.25) and that class imbalance simulation remains vital (3.13 vs. 3.39). Finally, scaling to 47M parameters yields substantial gains (improving to 2.04), suggesting that ICL effectiveness scales with model capacity. We provide an extended analysis in Section C.3.

Synthetic-to-Real Generalization. The fact that TiCT, trained purely on synthetic data, matches ResNet trained on real task-specific data validates our design: diverse augmentations teach noise invariance, Mixup creates a continuum of patterns, and class imbalance prepares for realistic distributions. This demonstrates learned representations effectively replace hand-crafted functions (DTW) while generalizing across diverse real-world domains.

4 Conclusion

We introduced TiCT, a foundation model for in-context time series classification that handles arbitrary class counts through bit-based label encoding and output attention. Trained purely on synthetic data with Mixup and augmentations, TiCT matches supervised methods on 128 real-world datasets without fine-tuning, while being 14-34× faster by eliminating per-task training. This demonstrates that learned representations can effectively replace hand-crafted functions for time series classification. Limitations and future directions are discussed in Section D.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Enric Boix-Adsera, Omid Saremi, Emmanuel Abbe, Samy Bengio, Etai Littwin, and Joshua Susskind. When can transformers reason with abstract symbols? *arXiv preprint arXiv:2310.09753*, 2023.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: How tabpfn-v2 outperforms specialized time series forecasting models. *arXiv preprint arXiv:2501.02945*, 2025.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021.

Jiecheng Lu, Yan Sun, and Shihao Yang. In-context time series predictor. *arXiv preprint arXiv:2405.14982*, 2024.

Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Timepfn: Effective multivariate time series forecasting with synthetic data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20761–20769, 2025.

Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maks Volkovs, and Anthony L Caterini. Retrieval & fine-tuning for in-context tabular models. *Advances in Neural Information Processing Systems*, 37:108439–108467, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585. IEEE, 2017.

Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*, 2020.

Kristoffer Wickstrøm, Michael Kampffmeyer, Karl Øyvind Mikalsen, and Robert Jenssen. Mixing up contrastive learning: Self-supervised representation learning for time series. *Pattern Recognition Letters*, 155:54–61, 2022.

Chin-Chia Michael Yeh, Xin Dai, Huiyuan Chen, Yan Zheng, Yujie Fan, Audrey Der, Vivian Lai, Zhongfang Zhuang, Junpeng Wang, Liang Wang, et al. Toward a foundation model for time series data. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 4400–4404, 2023.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

A Detailed Architecture and Theoretical Analysis

A.1 Label Representation: Theoretical Comparison

We compare three candidate label encoding strategies for in-context learning. *Numerical encoding* represents labels as single integers (e.g., 0, 1, 2, ...), as used in tabPFN-based models (Hollmann et al., 2022; 2025; Thomas et al., 2024). The main drawback is that it may encourage the model to interpret labels as having continuous numerical meaning rather than as discrete symbolic categories. *One-hot encoding* represents each label as a sparse vector of size C where C is the number of classes. Only one dimension is active (value 1), while all others are 0. For tasks with many classes, this becomes high-dimensional and sparse. The probability of any dimension being active is $\frac{1}{C}$, making learning more difficult as C increases. *Bit representation* (ours) represents each label as a binary vector of size $\lceil \log_2 C \rceil$. This is significantly more compact than one-hot encoding while maintaining a discrete symbolic nature. Each bit has a 0.5 probability of being active (assuming uniform class distribution), providing a dense representation that simplifies learning. Empirical validation in Section 3.1 and detailed experimental results in Section C.1 confirm our theoretical prediction.

A.2 Complete Label Equivariance Proof

We provide a complete formal proof that our output attention mechanism ensures label equivariance. Let $\pi : \{1, \ldots, C\} \to \{1, \ldots, C\}$ be a permutation function on the class indices. Assuming the Transformer encoder has been successfully trained to treat labels as abstract symbols, the query vector Q and key vectors K_i that it produces are equivariant. Consequently, the attention weights α_i, which measure the semantic similarity between the query and keys, remain invariant under this label permutation. The one-hot value vectors V_i, however, are permuted directly. Let V_i be the original one-hot vector for a sample with label c, and let V_i' be the new one-hot vector for the same sample, now with label $\pi(c)$. The new output logits, denoted l', are computed using the invariant weights α_i and the permuted values V_i': $l' = \sum_{i=1}^{k} \alpha_i V_i'$. The logit for a new class index j is given by the sum of weights for all samples now labeled j, which means $y_i' = j$. Since $y_i' = \pi(y_i)$, the condition $y_i' = j$ is equivalent to $y_i = \pi^{-1}(j)$. Thus, the new logit for class j is: $l'[j] = \sum_{i \text{ s.t. } y_i = \pi^{-1}(j)} \alpha_i$. This is precisely the original logit for class $\pi^{-1}(j)$. Therefore, we have $l'[j] = l[\pi^{-1}(j)]$. This result shows that the new logits vector is a permutation of the original logits vector, determined by π. This demonstrates that our output attention mechanism is formally label-equivariant, ensuring that the model's predictions remain consistent regardless of the arbitrary assignment or shuffling of class indices.

B Extended Pre-training Details

The synthetic data generation algorithm (Algorithm 1) creates diverse in-context learning tasks through several key mechanisms:

KernelSynth Template Generation. We use KernelSynth (Ansari et al., 2024), a method that generates realistic time series by sampling from Gaussian processes with various kernel combinations (periodic, linear, RBF, and rational quadratic) and randomly selected parameters. This produces two distinct template shapes for each synthetic task, ensuring clear separability while maintaining realistic temporal characteristics.

Randomized Bit Labels. While one-hot labels are fixed to $[1, 0]$ and $[0, 1]$, bit-based labels are randomly generated unique binary vectors (Lines 5-6). This randomization is crucial: it prevents

Algorithm 1 Synthetic In-Context Dataset Generation

Input: dataset size n, label bits n_{bit}, max augmentations n_{max}
Output: samples X, one-hot labels $Y_{\text{one-hot}}$, bit labels Y_{bit}

1 **function** GENERATEDATASET($n, n_{\text{bit}}, n_{\text{max}}$)
2 $\quad$ $\text{template}_0, \text{template}_1 \leftarrow$ KERNELSYNTH(), KERNELSYNTH()
3 $\quad$ $(\text{one_hot}_0, \text{bit}_0) \leftarrow ([1, 0], \text{RANDBINARY}(n_{\text{bit}}))$
4 $\quad$ $(\text{one_hot}_1, \text{bit}_1) \leftarrow ([0, 1], \text{RANDBINARY}(n_{\text{bit}}))$ s.t. $\text{bit}_1 \neq \text{bit}_0$
5 $\quad$ $t \leftarrow \mathcal{U}(0.1, 0.9)$
6 $\quad$ AUGF $\leftarrow$ SELECTRANDAUGS(n_{max})
7 $\quad$ **for** $i = 1$ **to** n **do**
8 $\quad\quad$ $r \leftarrow \mathcal{U}(0, 1)$
9 $\quad\quad$ $x \leftarrow r \cdot \text{template}_0 + (1 - r) \cdot \text{template}_1$
10 $\quad\quad$ $x \leftarrow \text{AUGF}(x)$
11 $\quad\quad$ $(y_{\text{one-hot}}, y_{\text{bit}}) \leftarrow \begin{cases} (\text{one_hot}_0, \text{bit}_0) & \text{if } r > t \\ (\text{one_hot}_1, \text{bit}_1) & \text{otherwise} \end{cases}$
12 $\quad\quad$ Append $(x, y_{\text{one-hot}}, y_{\text{bit}})$ to outputs
13 $\quad$ **return** $X, Y_{\text{one-hot}}, Y_{\text{bit}}$

the model from learning any fixed mapping between bit patterns and semantic meaning, forcing it to treat labels as abstract interchangeable symbols. During training, the model encounters thousands of different bit-label assignments, learning to solve the general ICL problem rather than memorizing specific label meanings.

Class Imbalance Simulation. The threshold $t \sim \mathcal{U}(0.1, 0.9)$ (Line 7) determines the mixing ratio boundary between classes. When $t = 0.5$, classes are perfectly balanced. When $t = 0.2$, one class appears $4\times$ more frequently than the other. This variation teaches the model to handle realistic imbalanced distributions commonly found in real-world datasets.

Augmentation Pipeline. We randomly select between 0 and n_{max} augmentations from the following set: jittering (adding Gaussian noise), smoothing (moving average), slope trend addition, spike insertion, step function addition, random cropping, random masking, temporal shifting, and time warping (Yeh et al., 2023). Each augmentation has stochastic parameters (e.g., spike location, noise magnitude), ensuring diversity even within a single synthetic dataset.

C COMPLETE EXPERIMENTAL RESULTS

C.1 LABEL REPRESENTATION STUDY: FULL DETAILS

Task Definition. The binary matching problem is defined formally as follows. Given a context set $\mathcal{D}_{\text{context}} = \{(x_i, y_i)\}_{i=1}^{n}$ where $y_i \in \{1, 2, \ldots, 2^{n_{\text{bit}}}\}$ and a query label q, the model must output $o_i \in \{0, 1\}$ for each context item where $o_i = 1$ if $y_i = q$ and $o_i = 0$ otherwise.

For each generated dataset, we randomly select q and another label $p \neq q$. Each context item's label y_i is then assigned to either q or p with probability 0.5. This forces the model to learn label comparison rather than memorizing specific class meanings.

Architectural Details. All three models use identical Transformer architectures with 8 layers, embedding dimension 128, 4 attention heads, feedforward dimension 512, context size $n = 15$, and a label universe of $2^{n_{\text{bit}}} = 256$ possible classes. The only difference is the input projection dimension: 1 for numerical, 256 for one-hot, and 8 for bit (ours), all projecting to dimension 128.

Convergence Analysis. As shown in Fig. 3, the bit representation converges dramatically faster than alternatives, reaching near-perfect test accuracy by the 17th epoch, while numerical and one-hot representations require significantly more training. This empirically validates our theoretical prediction that compact, dense representations with higher activation probabilities simplify learning symbolic label relationships.

C.2 UCR ARCHIVE: COMPLETE RESULTS

Evaluation Protocol Details. For each of the 128 UCR datasets, we use official train/test splits without modification. Context examples are retrieved using z-normalized Euclidean distance with context size $k = 64$ samples per test query (for datasets with < 64 training samples, we use all available samples). No hyperparameter tuning is performed; the same model and retrieval strategy is used for all 128 datasets.

Statistical Significance Testing. A Wilcoxon signed-rank test confirms that the performance differences between TiCT-Large (Rank 2.86), ResNet (Rank 2.86), and Transformer (Rank 2.58) are not statistically significant, validating that TiCT performs on par with supervised deep learning methods.

Complete Per-Dataset Results. The full experimental results for all 128 UCR Archive datasets, including per-dataset accuracy and rank metrics, are available at the project website: `https://sites.google.com/view/tsicl`.

C.3 ABLATION STUDY: EXTENDED ANALYSIS

Effect of Augmentation. Removing data augmentation (-Augmentation, Rank 5.01) causes severe performance degradation, confirming its critical role in synthetic-to-real transfer. Without augmentation, the model overfits to the smooth synthetic templates and fails to generalize to noisy real-world signals. The augmentation suite (jittering, smoothing, warping, etc.) effectively teaches noise invariance.

Binary vs Multi-Class Mixup. Our binary Mixup approach (Rank 3.13) outperforms a multi-class variant where we generate $C > 2$ templates per task (-Mixup, Rank 4.25). We hypothesize that binary Mixup creates a richer decision boundary by generating a continuum of interpolated examples between two poles, while multi-class templates remain discrete. This mirrors the success of binary SVMs extended to multi-class problems.

Effect of Class Imbalance. Disabling class imbalance simulation (-Imbalance, Rank 3.39) worsens performance, showing that training on balanced data alone is insufficient. Real-world UCR datasets often have imbalanced class distributions, and our synthetic pre-training must expose the model to this variability.

Encoder Comparison. ResNet (Rank 3.13) slightly outperforms Transformer (Rank 3.17) for the base model, confirming ResNet as the superior encoder. However, both are competitive, suggesting the ICL framework is encoder-agnostic.

Scaling Analysis. Scaling to the large model (TiCT-Large, Rank 2.04) provides substantial performance gains, demonstrating that ICL performance benefits significantly from model capacity.

C.4 EFFICIENCY ANALYSIS: EXTENDED DISCUSSION

Speedup over DTW. TiCT (242.5s) is approximately 1,500× faster than 1NN-DTW (363,198s).

Speedup over Supervised Methods. TiCT is 14× faster than ResNet and 34× faster than Transformer, which must train separate models for each of the 128 tasks.

Amortization Advantage. As the number of tasks grows beyond 128, TiCT's advantage increases since pre-training cost is fixed while supervised training scales linearly with task count.

D EXTENDED LIMITATIONS AND FUTURE DIRECTIONS

Multivariate Time Series. The current architecture processes univariate time series. Extending to multivariate data (e.g., UEA Multivariate Archive) would require:

- Modifying the encoder to handle multiple channels and unknown amounts of input channels
- Ensuring label equivariance is preserved with channel-wise processing
- Ensuring channel permutation invariance, as channel ordering should not affect predictions
- Updating KernelSynth to generate correlated multi-channel templates

Theoretical Understanding. While our empirical results demonstrate successful synthetic-to-real transfer, a rigorous theoretical characterization of what properties enable this generalization remains an open question. Understanding the implicit regularization induced by Mixup and augmentation could inform future design choices.

Alternative Retrieval Strategies. We use simple Euclidean distance for context retrieval. Exploring learned retrieval metrics or more sophisticated selection strategies (e.g., diversity-based sampling) could improve performance.