Appendix

Table of Contents

A Hosting, Licensing, and Maintenance

Our datasets and code are available via the following links:

- Github: `https://github.com/NREL/BuildingsBench`
- Documentation: `https://nrel.github.io/BuildingsBench`
- Datasets and Website: `https://data.openei.org/submissions/5859`
- Tutorials: `https://nrel.github.io/BuildingsBench/tutorials/`
- Dataset DOI: `https://doi.org/10.25984/1986147`

As described in Sec. 3 and Sec. 4, Buildings-900K and the BuildingsBench benchmark datasets are available for download under a CC-4.0 license and our code is available under a BSD 3-Clause license. We ensure the long-term availability and maintenance of the data by hosting it on the Open Energy Data Initiative (OEDI) platform (`https://data.openei.org/about`). The OEDI is a centralized repository for storing energy-related datasets derived from U.S. Department of Energy projects.

B Datasheet for Buildings-900K

Questions from Datasheet for Datasets (v8) [12].

B.1 Motivation

Q: For what purpose was the dataset created?

This dataset was created to research large-scale pretraining of models for short-term load forecasting (STLF). It specifically addresses a lack of appropriately sized and diverse datasets for pretraining STLF models. Buildings-900K, which consists of simulated building energy consumption time series, is derived from the National Renewable Energy Lab (NREL) End-Use Load Profiles (EULP) database. We emphasize that the EULP was *not* originally developed for studying STLF. Rather, it was developed as a general resource to "...help electric utilities, grid operators, manufacturers, government entities, and research organizations make critical decisions about prioritizing research and development, utility resource and distribution system planning, and state and local energy planning and regulation." [45].

Q: Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?

Researchers at NREL created Buildings-900K. Consent from the NREL developers of the EULP project was obtained in writing to extract, process, and redistribute a subset of the EULP. The EULP is a multiyear multi-institutional collaboration lead by NREL and is publicly available for download with a permissive CC-4.0 license.

Q: Who funded the creation of the dataset?

This dataset was developed with funding provided by the Laboratory Directed Research and Development program at NREL.

B.2 Composition

Q: What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)?

Annual hourly energy consumption from EnergyPlus [8] simulations. EnergyPlus is a high fidelity building energy consumption simulator which has been continuously maintained and updated for over 20 years by the U.S. Department of Energy.

Q: How many instances are there in total (of each type, if appropriate)?

There are $\sim$1.8 million annual time series instances (900K buildings $\times$ 2 years).

Q: Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?

Buildings-900K consists of a subset of the EULP database. Specifically, we extract only the `out.site_energy.total.energy_consumption` time series from each `upgrade=0` building in the 2021 version for weather-years AMY2018 and TMY3. The EULP contains additional end-use time series for each building, the EnergyPlus XML model files, the EnergyPlus weather files, and a variety of additional simulations under various electrification scenarios. We did not include these additional data in the initial version of Buildings-900K, although we may consider adding them in the future.

Q: What data does each instance consist of?

Each instance is stored in a Parquet file and has timestamp information and load values (in kWh). We also have spatial metadata that maps each building to a U.S. county.

Q: Is any information missing from individual instances?

No.

Q: Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)?

Each building has a unique identifier and is assigned to a county. That county (or Public Use Microdata Area—PUMA) has its own unique identifier. Each PUMA contains many buildings that are geographically close to each other.

Q: Are there recommended data splits (e.g., training, development/validation, testing)?

- **For test:** We withhold all buildings in 4 counties from both AMY2018 and TMY3 weather-years.
- **For validation:** We withhold dates 2018-12-17 to 2018-12-31 from all buildings in the training set to use as held-out validation data (e.g., for early stopping).
- **For pretraining:** All remaining data is used for pretraining.

Q: Are there any errors, sources of noise, or redundancies in the dataset?

No.

Q: Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?

It is self-contained.

Q: Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals' non-public communications)?

No. All simulated building models are created by sampling from conditional distributions over attributes calibrated to real data.

Q: Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?

No.

Q: Does the dataset relate to people?

No.

B.3 Collection Process

Q: How was the data associated with each instance acquired?

Buildings-900K time series instances are the outputs of extensively calibrated and validated EnergyPlus simulations. These simulations were created and run by the NREL EULP team on high performance computing resources. For completeness, we summarize here the steps taken by the EULP team to create the synthetic building models for these simulations, which are also described in the EULP documentation in more detail [45]. The synthetic buildings used to form large-scale U.S. residential and commercial building stock models were obtained from the ResStock[1] and ComStock[2] analysis tools. An extensive calibration and validation effort based on utility meter data from millions of customers, end-use submetering data, and additional private data sources helped to ensure the accuracy of the synthetic ResStock and ComStock models used in the EULP. A new stochastic occupant behavior model for the residential building stock was developed to help calibrate the simulations. Uncertainty quantification based on a learned surrogate [51] was also used for model calibration, in particular, to characterize sensitivity of simulation outputs (e.g., quantities of interest, or QOIs) with respect to different model parameters.

Q: What mechanisms or procedures were used to collect the data (e.g., hardware apparatus or sensor, manual human curation, software program, software API)?

See above.

Q: If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)?

Buildings-900K consists of a hand-selected subset of the EULP database.

Q: Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)?

N/A

Q: Over what timeframe was the data collected?

A portion of Buildings-900K consists of building load time series simulated with weather from 2018.

Q: Were any ethical review processes conducted (e.g., by an institutional review board)?

No.

B.4 Preprocessing/cleaning/labeling

Q: Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)?

The steps we followed to create Buildings-900K are as follows:

[1] `https://resstock.nrel.gov`
[2] `https://comstock.nrel.gov`

1. Select the `out.site_energy.total.energy_consumption` time series from each AMY2018 and TMY3 building.

2. Aggregate the 15-minute resolution energy consumption to hourly by *summing* the values at intervals of four consecutive timestamps XX:15, XX:30, XX:45, and XX+1:00 (e.g., 12:15, 12:30, 12:45, 13:00). Note that we use a sum aggregation here. When aggregating meter readings (in kWh) for real buildings (e.g., the BuildingsBench evaluation data), we take the *average* of the sub-hourly values.

3. Join the time series for all buildings in the same PUMA into a single Parquet table.

4. Save each PUMA-level Parquet table (compressed with Snappy).

Q: Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)?

The raw data (the EULP) is publicly accessible for download `https://data.openei.org/s3_viewer?bucket=oedi-data-lake&prefix=nrel-pds-building-stock`.

Q: Is the software used to preprocess/clean/label the instances available?

Yes, at `https://github.com/NREL/BuildingsBench`.

B.5 Uses

Q: Has the dataset been used for any tasks already?

In this work, we evaluate pretrained models on zero-shot STLF and transfer learning for STLF (fine-tuning pretrained models on limited data from a target building).

Q: Is there a repository that links to any or all papers or systems that use the dataset?

We will add links to papers or systems that use the dataset to `https://data.openei.org/submissions/5859` and `https://nrel.github.io/BuildingsBench`.

Q: What (other) tasks could the dataset be used for?

Pretrained models can sometimes serve as general-purpose feature extractors. These features can be used for a wide range of tasks, including:

- Predicting energy consumption of a building based on its characteristics.
- Classifying user behavior (e.g., whether a building is currently occupied) based on their energy consumption.
- Classifying building type based on energy consumption.
- Load disaggregation, that is, predicting the energy consumption of individual appliances based on the total energy consumption of a building.

Among others, potentially. We did not explore these tasks in this paper.

Q: Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?

Buildings-900K is (in its current state) representative of buildings in the United States. While we find evidence that models pretrained on Buildings-900K generalize to buildings in other countries in the northwestern hemispheres, such as the UK and Canada, it may not be representative of building energy consumption patterns in other regions of the world.

Q: Are there tasks for which the dataset should not be used?

None that the authors are currently aware of.

B.6 Distribution

Q: Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created?

Yes.

Q: How will the dataset will be distributed (e.g., tarball on website, API, GitHub)?

It will be publicly available for download on the Open Energy Data Initiative (OEDI) website at `https://data.openei.org/submissions/5859`.

Q: ill the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?

CC-4.0.

Q: Have any third parties imposed IP-based or other restrictions on the data associated with the instances?

No.

Q: Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?

No.

B.7 Maintenance

Q: Who is supporting/hosting/maintaining the dataset?

Code is hosted and publicly available on Github. NREL will maintain the code and datasets. The data is hosted on the Open Energy Data Initiative site, which is a centralized repository for energy-related datasets derived from U.S. Department of Energy projects.

Q: How can the owner/curator/manager of the dataset be contacted (e.g., email address)?

The lead maintainer at NREL is Patrick Emami (`Patrick.Emami@nrel.gov`).

Q: Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?

Yes. A version naming system will be used to indicate updated versions. We are also maintaining a GitHub repository for the associated benchmark (BuildingsBench) where notifications of updates will be posted.

Q: Will older versions of the dataset continue to be supported/hosted/maintained?

Yes, older versions will remain available on the OEDI website.

Q: If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?

Not officially, but our benchmark code is open source and pull requests are welcome.

C BuildingsBench Datasets

C.1 ElectricityLoadDiagrams20112014

This dataset is available from the UCI Machine Learning Repository[3] under a CC-4.0 license. It contains 15-minute consumption time series (in kW) for 370 clients in Portugal from 2011 to 2014. The magnitudes of the loads are much larger than residential homes, thus, we consider these time series as commercial buildings. After filtering for missing data, we kept 359 buildings. We included this dataset in BuildingsBench because of its permissive license, popular use in machine-learning-based time series forecasting studies [49], and size.

C.2 The Building Data Genome Project 2

This dataset is available[4] under an MIT License. It consists of measurements from 3,053 meters from 1,636 commercial buildings over two years (2016 and 2017). One or more meters per building measured the total electrical, heating and cooling water, steam, solar energy, water, and irrigation

[3] `https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014`

[4] `https://github.com/buds-lab/building-data-genome-project-2/`

usage. This dataset was curated by the *Building Data Genome Project*, a consortium of academics and practitioners. We use the whole building electricity meter measurements from Bear, Fox, Panther and Rat sites, totalling 611 buildings (from the CSV file `electricity_cleaned.csv`). This dataset was included in BuildingsBench because of its permissive license, size, and diversity.

C.3 Low Carbon London

This dataset is available[5] under a CC-4.0 license. It consists of consumption measurements from 5,567 London households that participated in a project led by UK Power Networks between November 2011 and February 2014. The dataset consists of half-hourly consumption (in kWh), a unique household identifier, and timestamps. Due to the large number of households included in the dataset, we randomly subsampled 713 of them (so as to keep the overall number of residential and commercial buildings in the benchmark approximately similar). We included this dataset because of its permissive license and size.

C.4 SMART

This dataset is available online unaccompanied by a specific license[6]. It contains meter measurements of aggregate electricity data for 7 residential homes (which are called Home A,B,C,D,F,G,H) collected by researchers at UMass Amherst. After filtering out 2 homes due to missing data, we were left with Home B (years 2014–2016), Home C (2014–2016), Home D (2016), Home F (2014–2016), and Home G (2016). We included this dataset in BuildingsBench due to its accessibility and because the homes are located in the U.S. (Western Massachusetts), are are the simulated homes in Buildings-900K.

C.5 IDEAL

This dataset is available[7] under a CC-4.0 license. It has meter readings for electric and gas usage, as well as room temperature, humidity, and boiler readings from 255 UK homes. We use the total home electricity measurements. The measurements were collected by researchers at the University of Edinburgh. After our filtering process, we kept 219 homes. This dataset was included in BuildingsBench due to its permissive license and size.

C.6 Individual household electric power consumption (Sceaux)

This dataset is available via Kaggle[8] under a DbCL-1.0 license. It contains consumption measurements from a home in Sceaux, France between December 2006 and November 2010. It provides 7 types of meter readings at one minute resolution with timestamps. We use the global active power readings. This dataset was included in BuildingsBench due to its permissive license and due to its popularity both in the literature [28] and on Kaggle.

C.7 Borealis

This dataset is available[9] under a CC0-1.0 license. It is 6-second resolution measurements of total real power consumption for 25 households in Waterloo, ON for at least three months between 2011 and 2012. After filtering out homes with excessive missing data, we kept 15 of them. The data was collected by researchers at the University of Waterloo. We included this dataset in BuildingsBench due to its permissive license and size.

C.8 Outlier Removal

Here we provide additional details on outlier removal for noisy meter data. The algorithm we use compares each load value to all others in a sliding window of 24 hours and computes the absolute difference with respect to the 1-nearest neighbor (1-NN). To select an outlier threshold, we compute

[5] `https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households`

[6] `https://traces.cs.umass.edu/index.php/smart/smart`

[7] `https://datashare.ed.ac.uk/handle/10283/3647`

[8] `https://www.kaggle.com/datasets/uciml/electric-power-consumption-data-set`

[9] `https://borealisdata.ca/dataset.xhtml?persistentId=doi:10.5683/SP2/R4SVBF`

the average daily peak load and average daily base load, and compute the difference. If the 1-NN distance is larger than this difference, we classify the spike as an outlier and replace the spike with the median of the sliding window.

D Evaluation Metrics

D.1 Additional Accuracy Metrics: NMAE, NMBE

NMAE: Similar to the NRMSE, the normalized mean absolute error (NMAE) measures the ability of a model to predict the correct load shape. However, it is less sensitive to large errors than the NRMSE. For a building with M days of load time series,

$$NMAE := 100 \times \frac{1}{\bar{y}} \left[\frac{1}{24M} \sum_{j=1,i=1}^{M,24} |y_{i,j} - \hat{y}_{i,j}| \right]. \tag{5}$$

NMBE: The normalized mean bias error (NMBE) is informative about a model's tendency to over or under-estimate building loads, which is practically useful to measure. However, a model that tends to over-estimate or under-estimate the load by an equal amount can achieve an NMBE close to zero (i.e., positive and negative errors cancel). Therefore, it is not a useful indicator of *accuracy*. For a building with M days of load time series,

$$NMBE := 100 \times \frac{1}{\bar{y}} \left[\frac{1}{24M} \sum_{j=1,i=1}^{M,24} (y_{i,j} - \hat{y}_{i,j}) \right]. \tag{6}$$

We visualize performance profiles and dataset-specific results for all metrics in App. H and App. I, respectively.

D.2 Categorical Ranked Probability Score

Here, we define the ranked probability score (RPS) for categorical distributions. This is used to compute the RPS for our time series transformer variant that predicts a categorical distribution over a vocabulary of $|\mathcal{V}|$ discretized load tokens. Each load token is the centroid of a cluster, where clusters are estimated via KMeans. See App. E for more details about the tokenizer.

In what follows, we assume that the vocabulary of load tokens is sorted in increasing order by value, that is, $0 \leq v_0 < v_1 < \cdots < v_{|\mathcal{V}|-1}$, $v_i \in \mathcal{V}$.

Let $p(\hat{Y}_{i,j})$ be the predicted categorical distribution over $\mathcal{V}$ for hour i on day j. We assume this is a normalized distribution, which is accomplished in practice by computing the softmax of the model's predicted logits. The corresponding CDF for $p(\hat{Y}_{i,j})$ is $F(\hat{Y}_{i,j})$, which is given by the cumulative sum of the $|\mathcal{V}|$ sorted entries of $p(\hat{Y}_{i,j})$. Then, let $F(Y_{i,j})$ be the CDF of the ground truth discretized load. If the ground truth category is $k \in \{0, \ldots, |\mathcal{V}| - 1\}$, the CDF $F(Y_{i,j})$ is 0 up to index $k-1$ and 1 thereafter (resembling a step function that goes from 0 to 1 at ground truth load index k). Finally, define $\bar{v}[k]$ as the difference between the maximum load value and the minimum load value assigned to load token $v[k]$ by KMeans (recall, load tokens are centroids of 1D KMeans clusters).

Then, the categorical RPS at hour i for a building with M days of load time series is

$$CatRPS_i := \frac{1}{M} \sum_{j=1}^{M} \sum_{k=0}^{|\mathcal{V}|-1} (F(\hat{Y}_{i,j})[k] - F(Y_{i,j})[k])^2 \bar{v}[k]. \tag{7}$$

D.3 Gaussian Ranked Probability Score

The *continuous* ranked probability score for Gaussian distributions can be evaluated in closed form. In particular, let $\hat{\mu}_{i,j}$ and $\hat{\sigma}_{i,j}$ be a predicted mean and standard deviation, and $y_{i,j}$ be the ground truth value. Define $z_{i,j} = \frac{y_{i,j} - \hat{\mu}_{i,j}}{\hat{\sigma}_{i,j}}$. Then, the Gaussian (C)RPS at hour i for a building with M days

of load time series is

$$GaussCRPS_i := \frac{1}{M}\sum_{j=1}^{M}\hat{\sigma}_{i,j}\left[z_{i,j}\left(2\underbrace{\left(\frac{1+\text{erf}(z_{i,j})}{2\sqrt{2}}\right)}_{\text{Gaussian CDF}}-1\right)+2\underbrace{\left(\frac{\exp\frac{-z_{i,j}^2}{2}}{\sqrt{2\pi}}\right)}_{\text{Gaussian PDF}}-\frac{1}{\sqrt{\pi}}\right]. \quad (8)$$

Gaussian approximation of the inverse Box-Cox transform: Our time series transformer variant that predicts a Gaussian distribution for each hour i in the day-ahead forecast uses a Box-Cox power transformation to normalize the load values. In App. K, we show that standard scaling was insufficient to remove training instabilities caused by the non-Gaussian load time series. Since the Gaussian distribution is learned in the *Box-Cox transformed space*, we cannot directly use the GaussCRPS metric to evaluate uncertainty quantification *in the original un-scaled space* (i.e., with units of kWh). Our current solution, introduced here, is to compute a Gaussian distribution in the original un-scaled space that closely approximates the distribution given by the inverse Box-Cox transformation (for a reasonable range of standard deviations). We leave exploring advanced non-approximate solutions for future work.

First, we define the Box-Cox transformation of a random variable X

$$Y = f_\lambda(X) := \begin{cases}\frac{X^\lambda-1}{\lambda} & \text{if } \lambda \neq 0\\ \ln(X) & \text{if } \lambda = 0,\end{cases} \quad (9)$$

and its inverse

$$X = f_\lambda^{-1}(Y) := \begin{cases}(Y\lambda+1)^{\frac{1}{\lambda}} & \text{if } \lambda \neq 0\\ \exp(Y) & \text{if } \lambda = 0.\end{cases} \quad (10)$$

We estimate a Gaussian distribution in the un-scaled space that *approximates* the (power-normal) distribution of the samples inverted with f_λ^{-1}. Recall that $\hat{\mu}_{i,j}$ and $\hat{\sigma}_{i,j}$ are the predicted mean and standard deviation in the Box-Cox transformed space. We take the inverse of the predicted mean to be $f_\lambda^{-1}(\hat{\mu}_{i,j})$ (Eq. 10). Then, the standard deviation is estimated with:

$$\hat{\sigma}_{i,j}^+ = f_\lambda^{-1}(\hat{\mu}_{i,j}+\hat{\sigma}_{i,j}) - f_\lambda^{-1}(\hat{\mu}_{i,j}) \quad (11)$$

$$\hat{\sigma}_{i,j}^- = f_\lambda^{-1}(\hat{\mu}_{i,j}) - f_\lambda^{-1}(\hat{\mu}_{i,j}-\hat{\sigma}_{i,j}) \quad (12)$$

$$\tilde{\sigma}_{i,j} \approx \frac{\hat{\sigma}_{i,j}^+ + \hat{\sigma}_{i,j}^-}{2}. \quad (13)$$

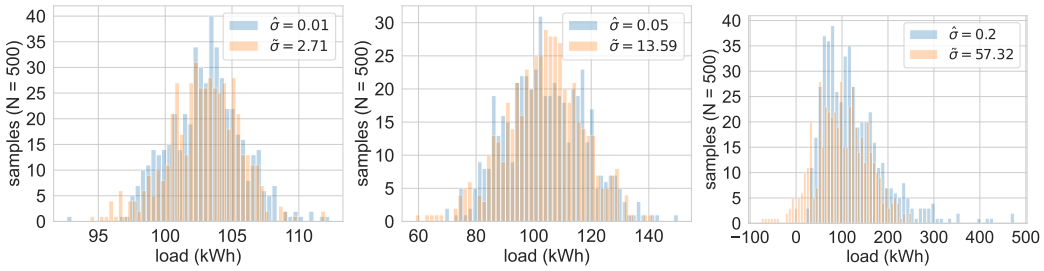
The Gaussian distribution in the un-scaled space is thus $\mathcal{N}(f_\lambda^{-1}(\hat{\mu}_{i,j}), \tilde{\sigma}_{i,j}^2)$. We find that this Gaussian is a good approximation for the distribution of the samples obtained by applying Eq. 10 to samples drawn from $\mathcal{N}(\hat{\mu}_{i,j}, \hat{\sigma}_{i,j}^2)$ for reasonably small values of $\tilde{\sigma}_{i,j}$. We compare these two distributions in Fig. 5. When the model is uncertain about the forecast and the standard deviation in the un-scaled space $\tilde{\sigma}_{i,j}$ is large, the true distribution in the un-scaled space is heavily skewed to the right, making a Gaussian a poor approximation (Fig. 5c).

E Time Series Transformer Details

Table 5: Transformer architecture hyperparameters.

	Predicted Distribution	# Layers	# Heads	Embedding Dim	MLP Dim	# Params
Transformer-S	Categorical	2	4	256	512	3.3M
Transformer-M	Categorical	3	8	512	1024	17.2M
Transformer-L	Categorical	12	12	768	2048	160.7M
Transformer-S	Gaussian	2	4	256	512	2.6M
Transformer-M	Gaussian	3	8	512	1024	15.8M
Transformer-L	Gaussian	12	12	768	2048	160.7M

In this section, we provide additional details about the time series transformer architecture used in this work. We use the original encoder-decoder model from Vaswani et al. [43] as implemented in the



(a) (b) (c)

Figure 5: **Gaussian approximation of the inverse Box-Cox**. Visualization of the Gaussian distribution $\mathcal{N}(f^{-1}(\hat{\mu}_{i,j}), \tilde{\sigma}_{i,j})$ (orange) in the un-scaled space of loads (in kWh) for computing the RPS when using Box-Cox scaling. For reasonably small standard deviations $\tilde{\sigma}$ in the un-scaled space, this Gaussian is a reasonable approximation of the power-normal distribution given by the inverse Box-Cox (blue). c) When the model is highly uncertain so that $\tilde{\sigma}$ is large, the power-normal is extremely right-skewed. In this case, the Gaussian approximation is inaccurate.

PyTorch `nn.Transformer` module. This autoregressive model is trained to predict the load at each time step with teacher forcing. Following Wu et al. [47], we use masking in the decoder to prevent the attention from peeking at the target value at each time step. The first value in the input sequence passed to the decoder is the last value of the input sequence passed to the encoder. The Gaussian time series transformer uses a linear layer (with parameters shared across time steps) to transform the high-dimensional output of the decoder into a mean and standard deviation at each time step. The time series transformer trained on tokenized load values uses a linear layer to transform the decoder outputs into $|\mathcal{V}|$ logits for the cross-entropy loss, also shared across time steps. Our tokenizer is based on KMeans (a stochastic algorithm) followed by a token merging step. Depending on the random seed, multiple runs of the tokenizer will produce token vocabularies of different sizes. The tokenizer is described in detail in App. E.1.

The architecture hyperparameters for the L, M, and S models are shown in Table 5. The largest model has 12 layers, 12 heads, and an internal dimension of 768 selected to match Radford et al. [35], but uses a smaller feedforward dimension of 2048 to keep the number of parameters at approximately 10^8. We downsize these architecture hyperparameters to reduce the number of parameters by roughly an order of magnitude for the S and M models. Similar to Radford et al. [35], all models use GELU [16] activations and $\mathcal{N}(0, 0.02)$ weight initialization.

Model input encoding: Let E be the embedding dimension and s be a scaling factor given by $\mathtt{E}/256$ (e.g., $s = 3$ for Transformer-L). The model inputs are encoded, concatenated, and passed to the transformer encoder as follows:

- **day of year**, **day of week**, **hour of day**: These are encoded into a 2-dim space with $\sin(\pi x)$ and $\cos(\pi x)$ then projected with a linear layer to $32s$-dim.
- **latitude, longitude**: These scalars are each projected to $32s$-dim space with linear layers.
- **building type**: A linear embedding layer is used to learn a projection for each of residential and commercial building types to $32s$-dim space.
- **continuous loads**: When the load is a continuous scalar, it is projected to $64s$-dim space with a linear layer.
- **discrete load tokens**: When the load is a discrete token, a linear embedding layer is used to learn a projection for each of the $|\mathcal{V}|$ tokens to $64s$-dim space.

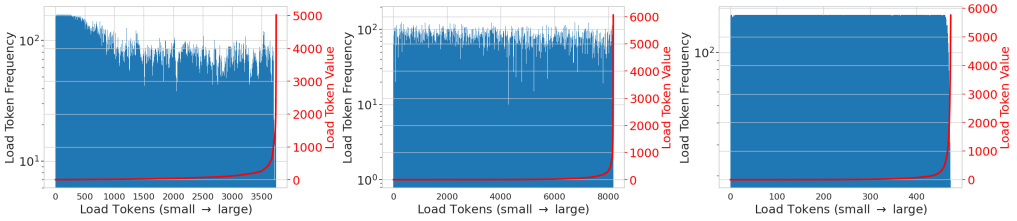
These various embeddings are concatenated into a feature vector of size E at each time step.

E.1 Load Tokenizer

We explore quantizing load values into a vocabulary of discrete tokens to closely mimic the application of transformers to natural language. The design of the tokenizer is detailed here. We use simple KMeans clustering with a basic merging strategy inspired by Byte Pair Encoding [11] which merges

Algorithm 1 Load tokenizer

```
// Inputs: loads ((n,1) array), K initial centroids, τ = 0.01 threshold (kWh)
kmeans = faiss.Kmeans(K)
kmeans.train(loads)
// Merge step. Initialize state.
sorted_centroids = sort(kmeans.centroids)
current_centroid = sorted_centroids[0]
merged_centroids = []
temp_centroids = [current_centroid]
// Iterate over sorted centroids in increasing order
for i = 1 to K do
  // The next centroid is less than τ away, merge it
  if sorted_centroids[i] - current_centroid < τ then
    temp_centroids.append(sorted_centroids[i])
  else
    merged_centroids.append(mean(temp_centroids))
    temp_centroids = [sorted_centroids[i]]
    current_centroid = sorted_centroids[i]
// Add the last centroid to the new list of merged centroids
merged_centroids.append(mean(temp_centroids))
return kmeans, merged_centroids
```



(a) $|\mathcal{V}|$ = 3,747 (K = 8,192) (b) $|\mathcal{V}|$ = 8,192 (No merging) (c) $|\mathcal{V}|$ = 474 (K = 512)

Figure 6: **Load token frequency**. We visualize histograms of token frequencies from a subsample of Buildings-900K. a) After merging K = 8,192 into 3,747, there are some tokens that appear often near load values of zero while most other tokens have nearly equal usage. The mean absolute quantization error for commercial buildings is approximately 0.2 kWh whereas for residential buildings it is 0.003 kWh. b) Without merging, most tokens correspond to small load values and all appear to have roughly equal usage. c) When $|\mathcal{V}|$ is small (474), there are few tokens for large load values.

clusters that are overly close to each other. We leave exploring other simple and sophisticated tokenization strategies for future work.

The tokenizer uses KMeans clustering on a random subsample of load values from the Buildings-900K training set to fit K clusters (the size of the subsample is chosen internally by `faiss-gpu`). The merge step is as follows. First, we sort the K cluster centroids in increasing order and then iterate over them. For any cluster centroid i, if there are successive centroids that are less than $\tau = 0.01$ kWh greater than i, we replace all of these centroids with a new centroid equal to their average. We provide pseudocode in Alg. 1. This algorithm is stochastic, such that running the tokenizer multiple times will produce vocabularies with different (but similar) sizes $|\mathcal{V}|$. To tokenize a continuous load value, we first look up the index of the original centroid it belongs to, then use this index to look up the corresponding index in the list of merged centroids. See Fig. 6 for a visual comparison of three different vocabularies produced by the tokenizer, and App. K.1 for ablation study results on the choice of K and the merging step.

F Training Details

In this section, we describe how training hyperparameters were selected for the transformer models. We use the loss on the Buildings-900K validation dataset for model selection. However, we observed

that a better Buildings-900K validation loss does not necessarily indicate better downstream performance on the real (out-of-distribution) evaluation data. We encourage exploring sophisticated model selection strategies that account for distribution shifts in future work.

Pretraining hyperparameters: The learning rate for all models uses a linear warmup of 10K steps followed by cosine decay to zero. Models are trained on 1 billion load hours, as we observed in early runs that the validation loss would no longer improve after this point. We perform a grid search over the max learning rate {6e-4, 6e-5, 6e-6} and the batch size {64, 128, 256} for the two Transformer-S models. For the Transformer-S (Gaussian) model, the highest learning rate and smallest batch size had the best validation loss. However, when applying these hyperparameters to the larger models, the high learning rate of 6e-4 caused training instability. Thus, we used the lower 6e-5 learning rate, which performed best with the smallest batch size of 64 (650K total gradient updates). Similarly for the Transformer-S (Tokens) model, the highest learning rate achieved the best validation loss, but for the larger models we used the lower 6e-5 learning rate with batch size of 64. We use the AdamW [26] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon =$ 1e-9, and weight decay of 0.01 following Radford et al. [35].

Transfer learning hyperparameters: We fine-tune all layers of the transformers (as opposed to only the last logits layer—see App. K.4 for an ablation study on this) with a lower learning rate of 1e-6. We allow a maximum of 25 fine-tuning epochs with an early stopping patience of 2. We tune the learning rate for the Linear, DLinear, and RNN baselines by sweeping over 5 values: {1e-2, 1e-3, 1e-4, 1e-5, 1e-6}. All non-pretrained models are trained for a maximum of 100 epochs.

G Compute Details

- **Buildings-900K processing:** Creating the Buildings-900K dataset by processing a subset of the EULP was accomplished with a 96-core AWS EC2 instance in 2-3 days.
- **Model pretraining:** The largest model variant, the Transformer-L, was trained on 1 billion hours from Buildings-900K with one NVIDIA 40GB A100 GPU in 24 hours. Smaller model variants train faster, in about 18 hours. We use PyTorch automatic mixed precision training with gradient scaling, which is crucial for achieving fast training times.
- **Zero-shot STLF:** The amount of time to run each benchmark task varies depending on model inference speed. This task uses all available years for every building in the benchmark. For the largest model variant, it takes about 2-3 hours on one NVIDIA 40GB A100. By contrast, the persistence baselines run in well under one hour.
- **Transfer learning:** Repeating the fine-tuning process for every building in the benchmark can be computationally demanding. For example, we sub-sample 200 total buildings (100 residential and 100 commercial) to use for this task, and this takes the Transformer-L models 1.5-3.5 hours to complete on one NVIDIA 40GB A100. We estimate it would take over 15 hours to include all buildings in this task. Models that are fine-tuned from frozen pretrained weights finish this task faster, as we stop the fine-tuning process with early stopping using a patience of 2 epochs.

H Performance Profiles

Performance profiles plot the fraction of all buildings with performance greater than a threshold τ for a range of threshold values. These can be used to understand the tails of the error distributions across all buildings in the benchmark. For example, these plots show the fraction of buildings with extremely high forecasting errors.

Here we show performance profiles for commercial (Fig. 12, Fig. 13) and residential (Fig. 14, Fig. 15) BuildingsBench buildings for the NRMSE, NMAE, NMBE, and RPS metrics. For reference, we also plot the performance profile of the Persistence Ensemble baseline. The methods that estimate the forecast with a Gaussian distribution (the Persistence Ensemble and Transformer (Gaussian)) achieve noticeably better NMBE performance than Transformer (Tokens).

I Benchmark Dataset Results

Per-dataset zero-shot accuracy (NRMSE, NMAE, NMBE) are visualized in Fig. 16.

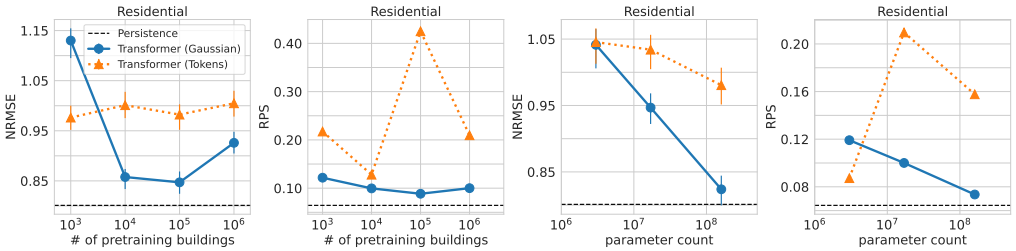


Figure 7: Empirical scaling laws for zero-shot generalization on residential buildings. Intervals are 95% stratified bootstrap CIs of the median. a-b) Dataset size vs. zero-shot performance. c-d) Model size vs. zero-shot performance.

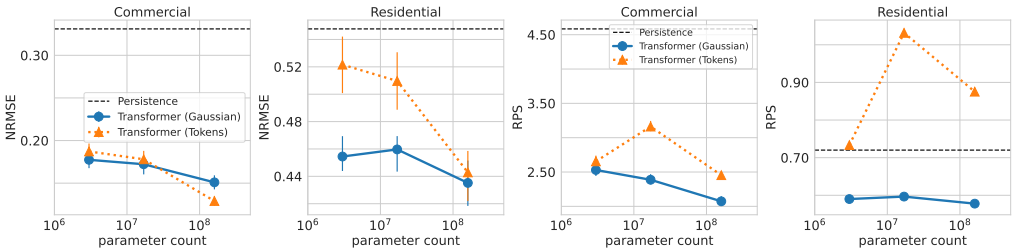


Figure 8: Model size vs. Buildings-900K-Test zero-shot STLF performance. Intervals are 95% stratified bootstrap CIs of the median. The trends in (a)-(c) suggest superior generalization to the test data is achievable by larger models, which might be unsurprising due to the *small* distribution shift between the simulated pretraining and simulated test data (relative to the *large* distribution shift between the simulated pretraining and real test data).

J Additional Empirical Scaling Law Results

The residential building results for the experiments where the Transformer-M models are trained under increasing dataset scale (size and number of unique pretraining buildings) and where model scales are compared are shown in Fig. 7. In most cases, the accuracy (NRMSE) and uncertainty quantification (RPS) are independent of the dataset and model size. Exceptions are the Transformer-M (Gaussian) performance on the smallest pretraining dataset size and the effect of model size on the Transformer (Gaussian), which appears to show a power-law scaling. By contrast, Fig. 3 shows that the smallest Transformer-S (Gaussian) model achieves comparable performance to the larger variants on commercial buildings. A possible simple explanation is that the small model struggles to learn to jointly forecast for both residential *and* commercial buildings due to limited capacity. A key benefit of the larger model, then, is unlocking the ability to jointly forecast for both types of buildings. As discussed in Sec. 6.2, alternative modeling approaches that consider auxiliary inputs might be more successful for large-scale pretraining of residential building STLF. We believe this is an interesting future research direction. We also examine empirically how performance on the simulated Buildings-900K-Test varies with model scale (Fig. 8), and observe trends that clearly indicate superior generalization is achievable with larger models in this setting.

K Ablation Studies

We explore the following aspects of the transformer models in this section:

- App. K.1) The size of the vocabulary and the impact of the merging step in the load tokenizer.
- App. K.2) Using standard scaling vs. Box-Cox scaling for training the Transformer (Gaussian) model.
- App. K.3) Impact of the lat, lon covariate on generalization.

Table 6: **Ablation study results on the zero-shot STLF task.** Median accuracy (NRMSE) and ranked probability score (RPS). Results are for Transformer-L (Tokens). We ablate the use of the lat,lon covariate and the size of the token vocabulary $|\mathcal{V}|$. The vocabulary with 8,192 tokens does not use merging (note the high RPS for residential buildings).

		Buildings-900K-Test (*simulated*)				**BuildingsBench (*real*)**			
		Commercial		Residential		Commercial		Residential	
$\|\mathcal{V}\|$	Lat,Lon	NRMSE (%)	RPS	NRMSE (%)	RPS	NRMSE (%)	RPS	NRMSE (%)	RPS
3,747	✗	15.22	2.79	45.85	0.913	14.40	5.73	97.84	0.199
474	✓	12.70	1.62	42.94	0.544	14.71	5.16	94.75	0.113
8,192	✓	15.06	2.85	45.50	7.05	14.62	5.90	99.88	14.37
3,747	✓	12.91	2.45	44.29	0.875	14.46	5.62	95.34	0.152

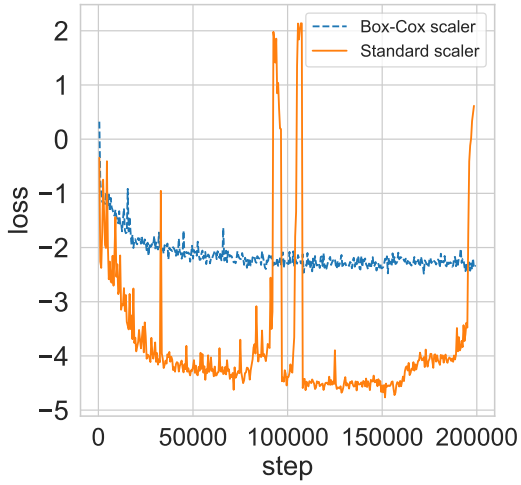


Figure 9: **Box-Cox scaling vs. standard scaling training stability**. Standard normalizing the load values did not prevent the Transformer (Gaussian) model from suffering from training instability. Box-Cox scaling was successful at achieving stable training.

- App. K.4) Fine-tuning all layers vs. only the last layer for transfer learning.

K.1 Load Tokenizer

Our experiments are conducted with a merged vocabulary of size $|\mathcal{V}| = 3{,}747$ ($K = 8{,}192$). To investigate the impact of using a smaller vocabulary, we trained the Transformer-L model with a merged vocabulary of size $|\mathcal{V}| = 474$ ($K = 512$) (Fig. 6c). We also trained a model *without merging*, thus, $|\mathcal{V}| = 8{,}192$ ($K = 8{,}192$) (Fig. 6b). The results in Table 6 indicate that when the vocabulary is too small ($|\mathcal{V}| = 474$), the accuracy on real commercial buildings somewhat deteriorates. Without merging ($|\mathcal{V}| = 8{,}192$), the RPS on real residential buildings is significantly worse. Nearly all tokens correspond to small load values, which causes the model to allocate probability to a large number of incorrect tokens whose load values are very close to the target value.

K.2 Standard Scaling Data Normalization

As can be seen in Fig. 9, global standard normalization was not sufficient to prevent training instability (large spikes in the loss) due to the highly non-Gaussian load values. By contrast, Box-Cox scaling was successful at stabilizing training.

K.3 Latitude, Longitude Covariate

In our experiments, the transformer models have a latitude, longitude covariate that roughly localizes where a building is in the world. The result of training Transformer-L (Tokens) without this covariate

Table 7: **Ablation study results on transfer learning task**. Median accuracy (NRMSE) and ranked probability score (RPS). For statistical robustness, we compute average probability of improving NRMSE due to fine-tuning: $P(X < Y) := \frac{100}{N} \sum_{i=1}^{N} \mathbf{1}_{[X_i < Y_i]}$ where X_i is the pretrained + fine-tuned NRMSE for building i and Y_i is the pretrained NRMSE. We compare fine-tuning only the last layer and fine-tuning all layers.

	Commercial buildings			Residential buildings		
	NRMSE (%)	RPS	$P(X < Y)$	NRMSE (%)	RPS	$P(X < Y)$
Fine-tune last layer						
Transformer-L (Tokens)	14.18	5.07	53.5	93.51	0.136	47
Transformer-L (Gaussian)	12.93	4.50	51	79.57	0.063	36
Fine-tune all layers						
Transformer-L (Tokens)	14.07	4.99	61.5	94.53	0.137	39
Transformer-L (Gaussian)	12.96	4.37	71	77.20	0.057	68

are in Table 6. The buildings in the Buildings-900K-Test set are from held-out PUMAs (i.e., counties) with never-before-seen latitude, longitude coordinates. Without latitude and longitude covariates, we observe a negligible drop in zero-shot accuracy on real buildings, but a significant loss in accuracy on the *simulated* Buildings-900K-Test split. As the majority of real buildings are located in non-U.S. out-of-distribution locations, the latitude and longitude covariate appears to not be particularly helpful for generalizing to these buildings. This is despite mapping the non-U.S. latitude longitude coordinates to U.S. locations with similar climates, which we do to help avoid introducing errors due to extrapolation. We believe further study on the geospatial aspects of generalizable building STLF is a promising use-case of our benchmark.

K.4 What Layers Get Fine-tuned?

For our main transfer learning benchmark results, we fine-tune all layers of the transformers. Here, we compare the performance of these models with fine-tuning only the last layer. Table 7 shows that when fine-tuning only the last linear layer (that output logits or Gaussian parameters), the performance suffers, which can be best seen by comparing the respective probabilities of improvement. Our intuition is that since the pretraining data is synthetic, fine-tuning on real data may require updating all layers. We believe a deeper investigation into transfer learning paradigms for time series is an important topic of study.

L Author Responsibility Statement

The authors confirm that they bear all responsibility in case of any violation of rights during the collection of the data or other work, and will take appropriate action when needed, e.g. to remove data with such issues. The authors also confirm the licenses provided with the data and code associated with this work.

M Additional Plots

We provide additional qualitative results here (Fig. 10 and Fig. 11).

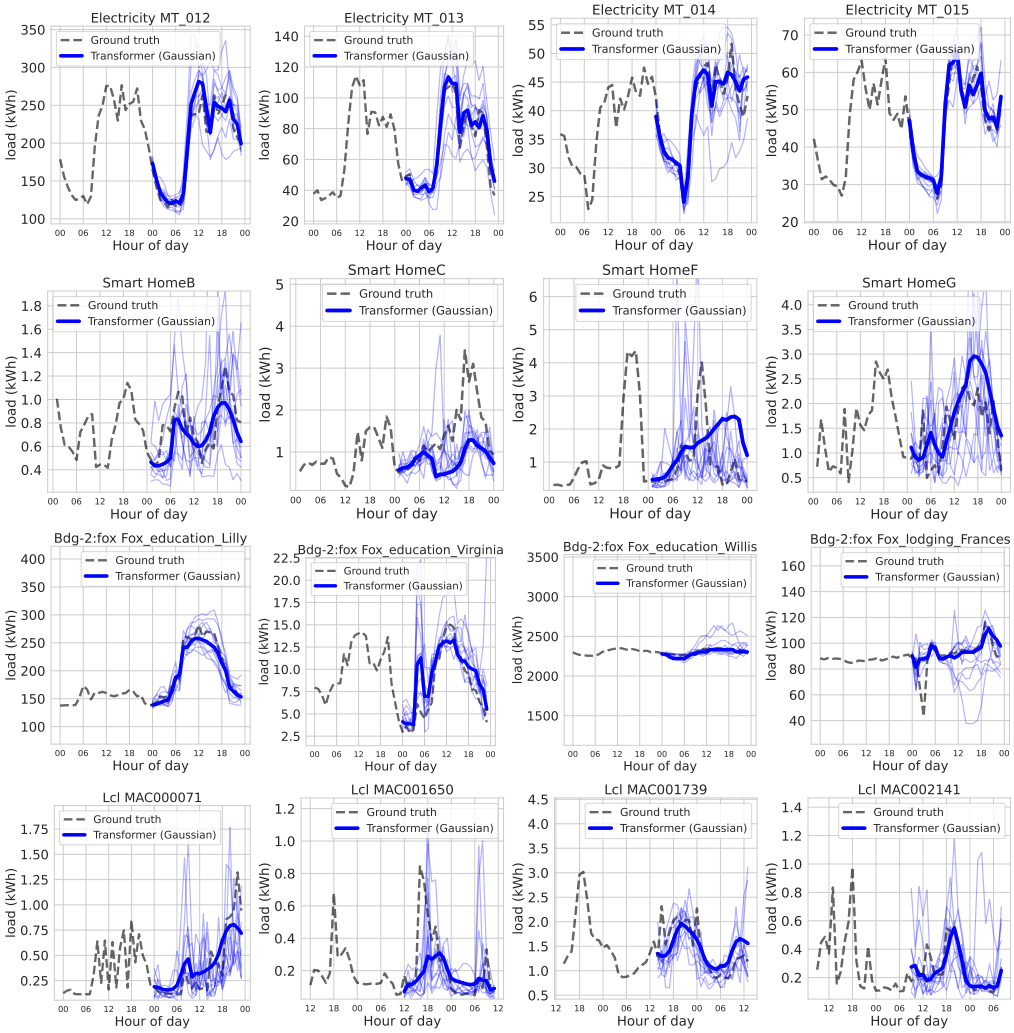


Figure 10: **Qualitative results for Transformer-L (Gaussian)**. Ground truth time series are truncated to previous 24 hours for visibility. Light blue lines are 10 samples from the predicted distribution.

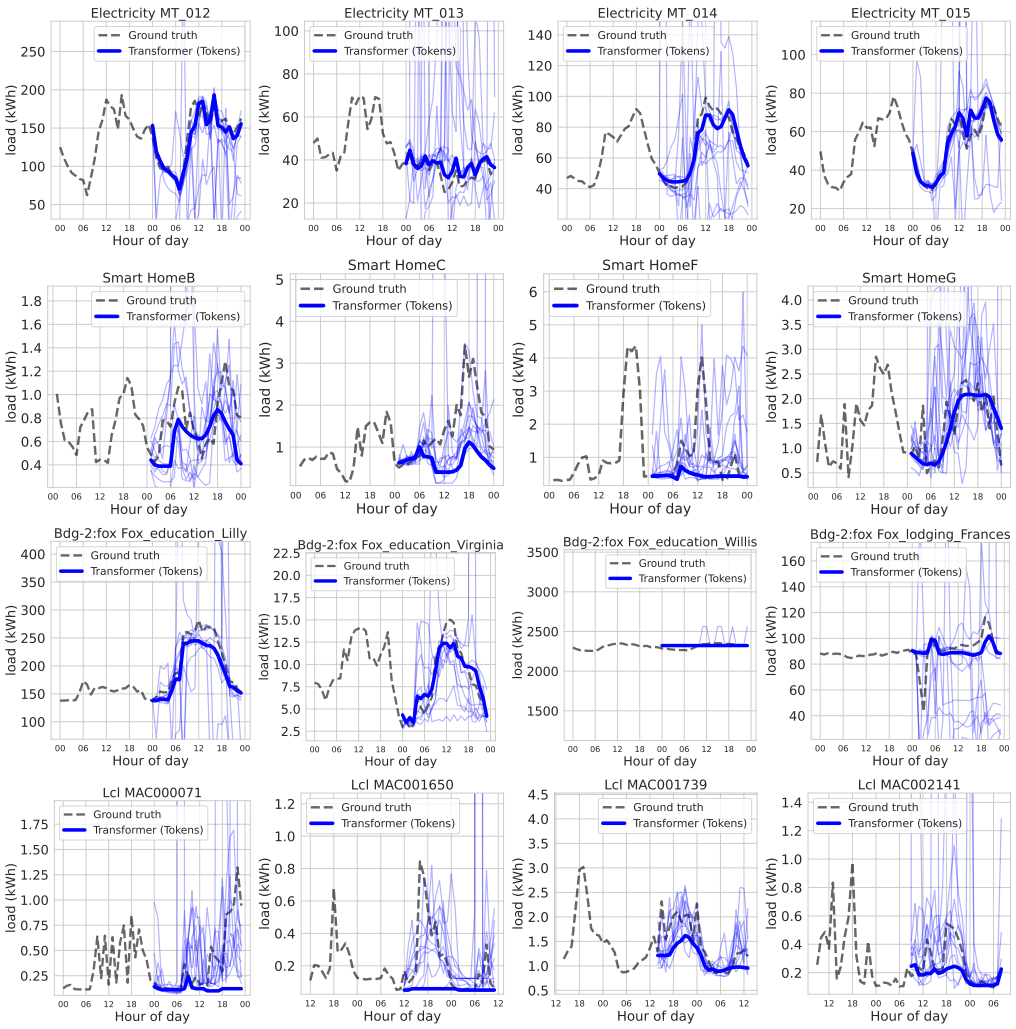
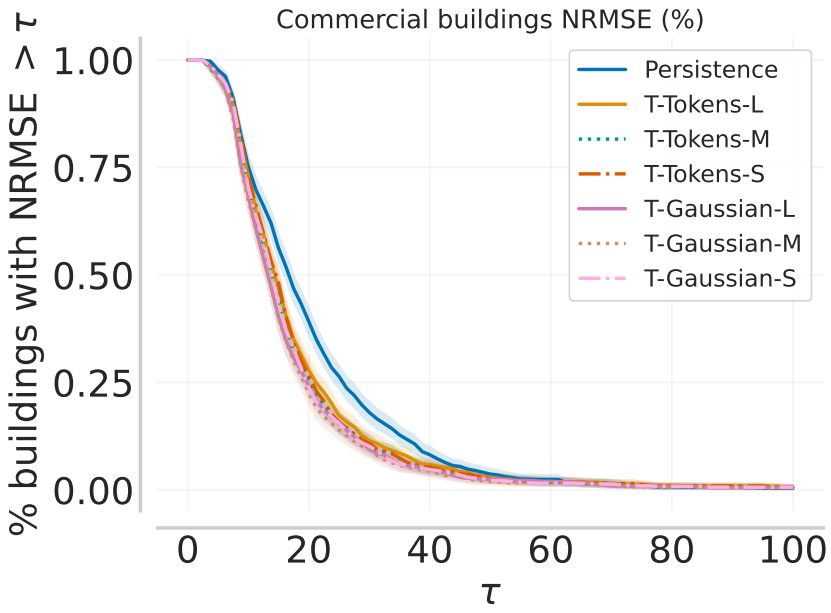
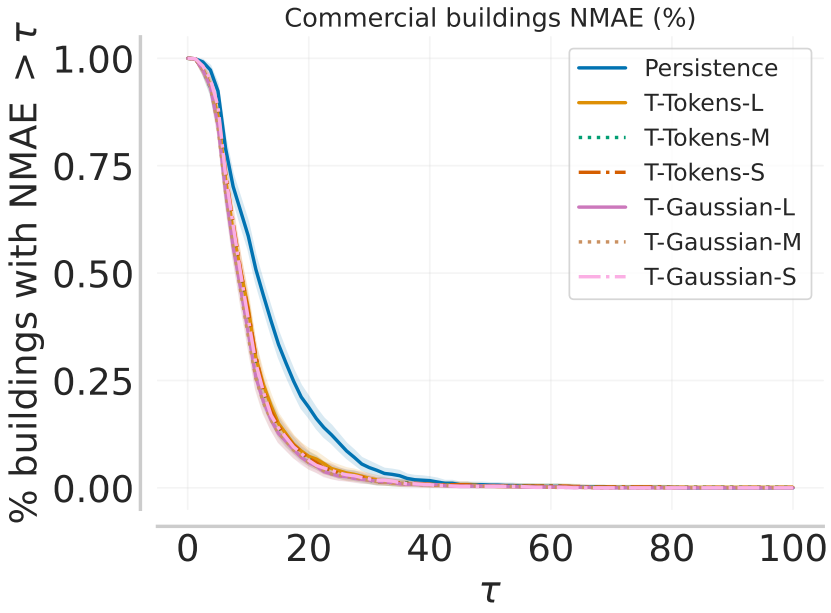


Figure 11: **Qualitative results for Transformer-L (Tokens)**. Ground truth time series are truncated to previous 24 hours for visibility. Light blue lines are 10 samples from the predicted distribution.

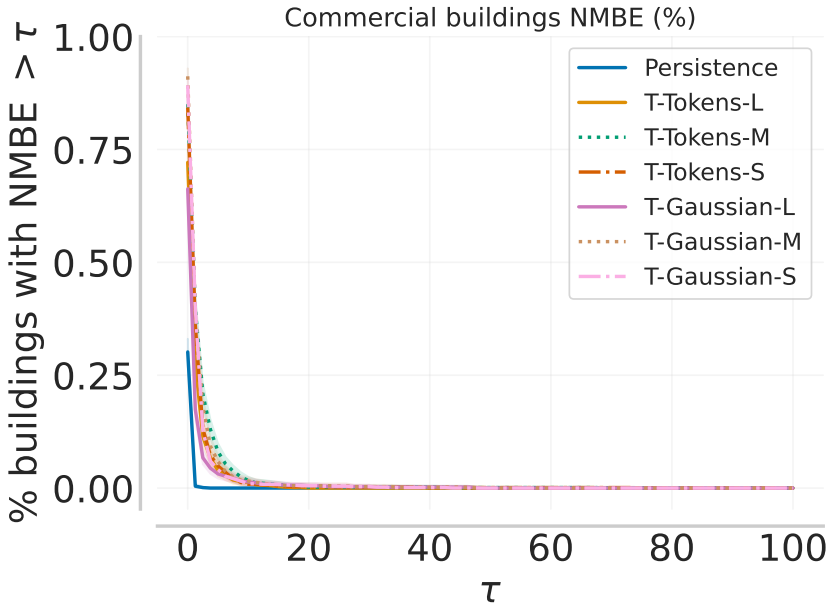


(a)

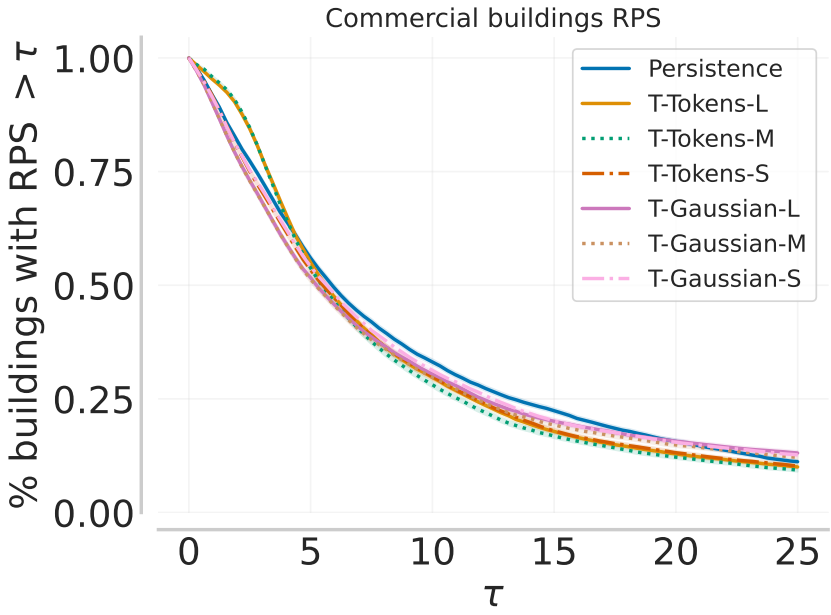


(b)

Figure 12: Real commercial building zero-shot NRMSE and NMAE performance profiles with 95% bootstrap CIs. Curves closer to the bottom left are better.

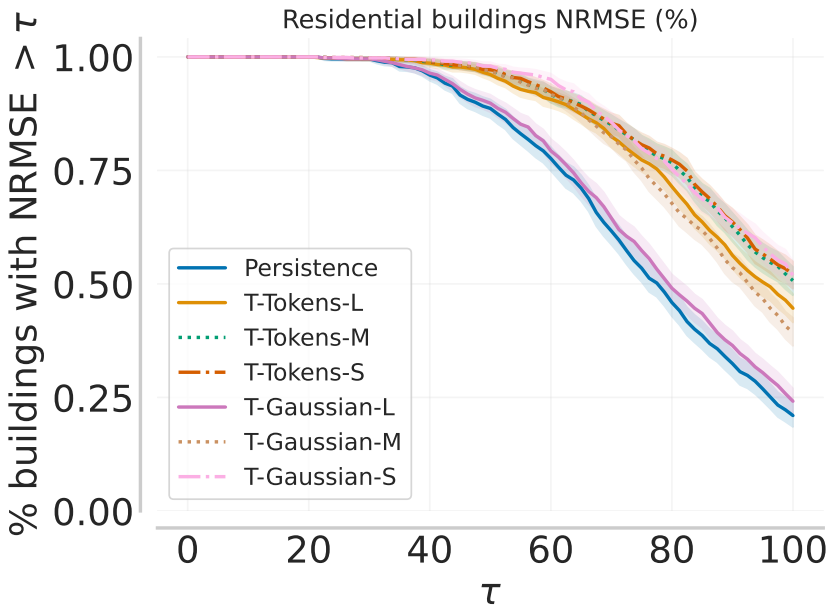


(a)

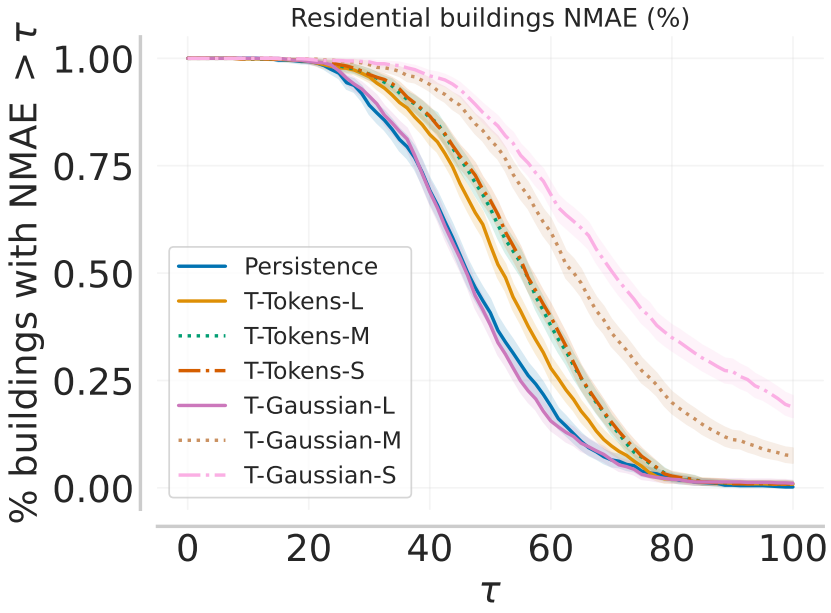


(b)

Figure 13: Real commercial building zero-shot NMBE and RPS performance profiles with 95% bootstrap CIs. Curves closer to the bottom left are better.

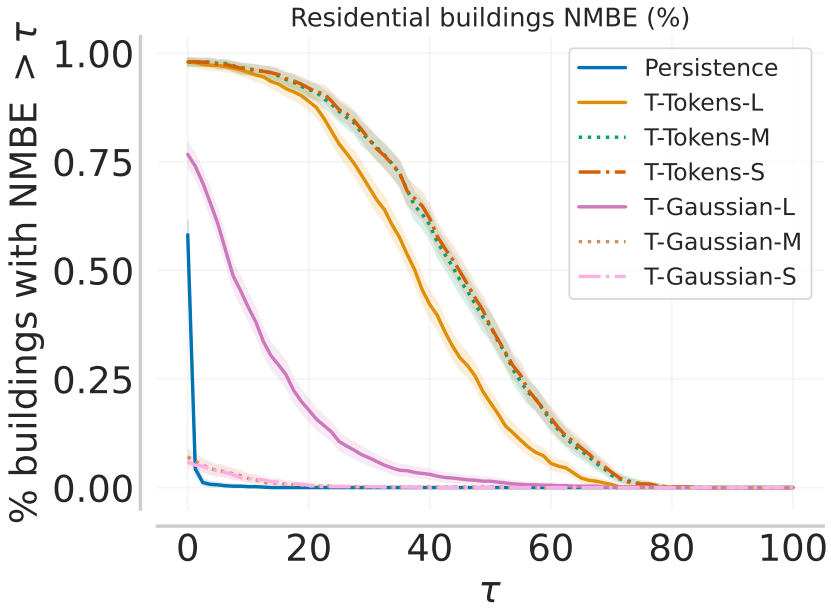


(a)

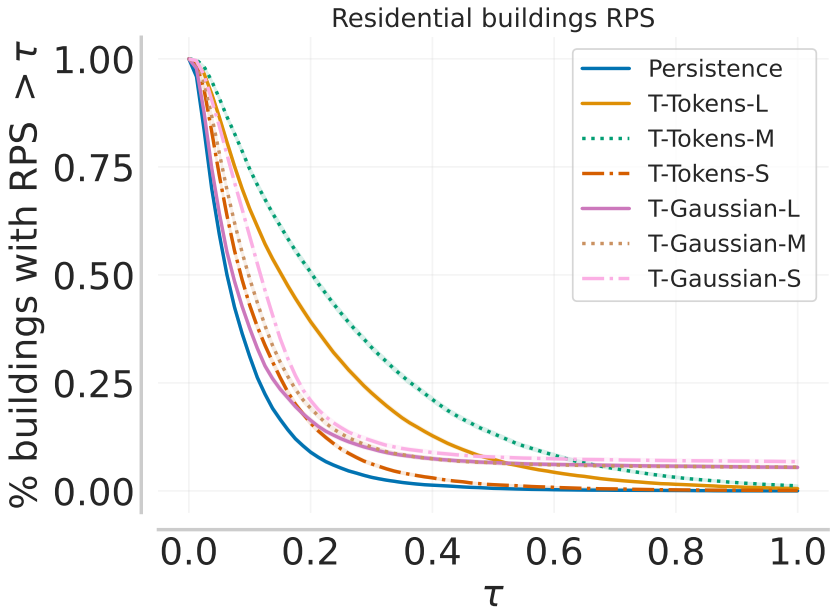


(b)

Figure 14: Real residential building zero-shot NRMSE and NMAE performance profiles with 95% bootstrap CIs. Curves closer to the bottom left are better.



(a)



(b)

Figure 15: Real residential building zero-shot NMBE and RPS performance profiles with 95% bootstrap CIs. Curves closer to the bottom left are better.

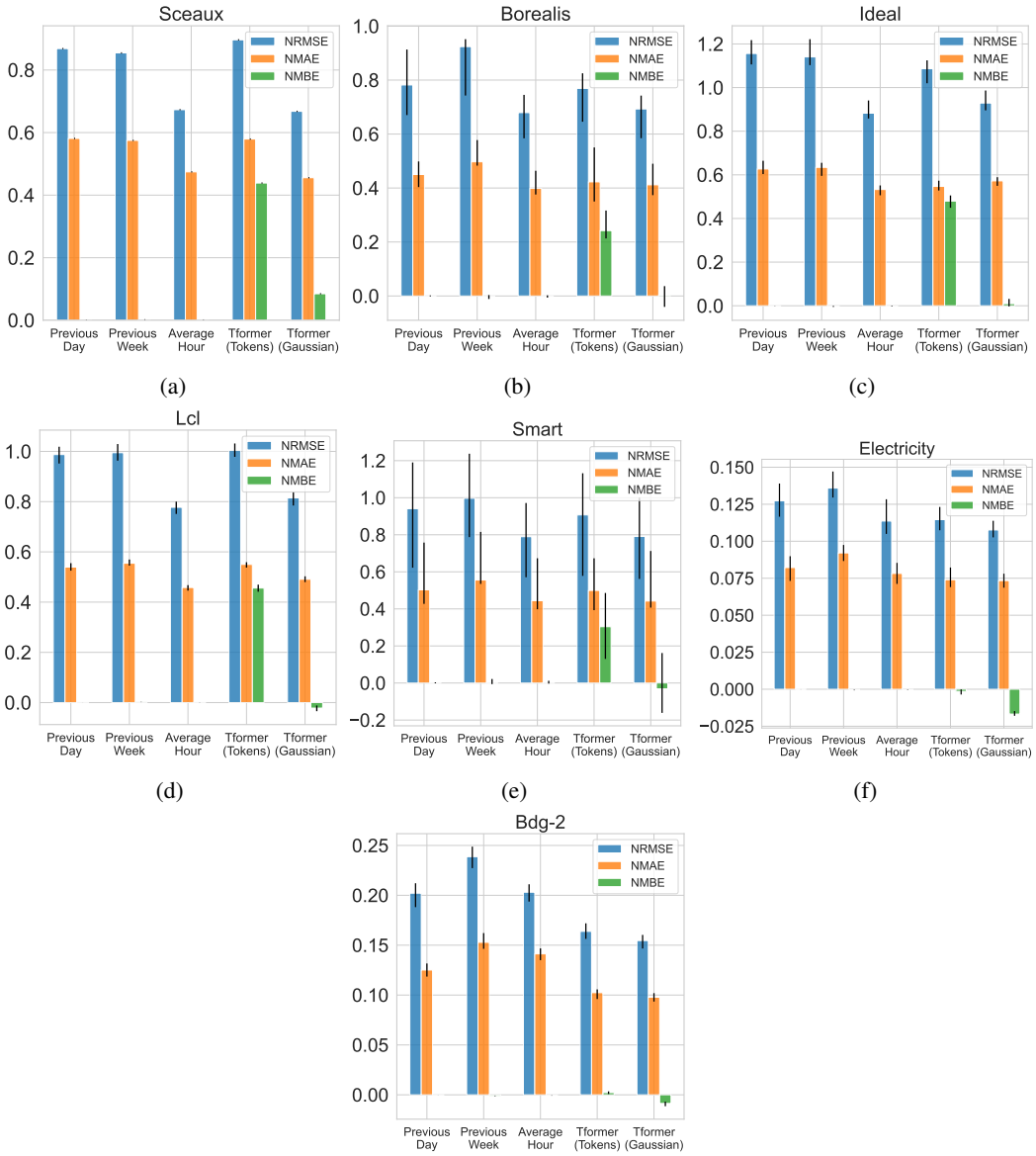


Figure 16: Per-dataset median zero-shot accuracy (NRMSE, NMAE, NMBE) with 95% bootstrap CIs. Lower is better.