university of groningen

faculty of science and engineering

Transfer Learning for Short-Term Load Forecasting: Comparing CNN and LSTM models

Antony Krymski

August 30, 2023

university of groningen / faculty of science and engineering

University of Groningen

Transfer Learning for Short-Term Load Forecasting: Comparing CNN and LSTM Models

Bachelor's Thesis

To fulfill the requirements for the degree of
Bachelor of Science in Computing Science
at University of Groningen under the supervision of Dilek Dustegor, PhD and Andres Tello

Antony Krymski (s4478177)

August 30, 2023

Contents

Acknowledgments

I extend my gratitude to Professor Düstegör for her guidance throughout this thesis. Our weekly meetings provided essential direction and insights. I also thank Andrés Tello for his supervision and dedicated code reviews. His feedback was instrumental in refining my approach. Both have been invaluable mentors, and I appreciate their support in completing this work.

Abstract

Accurate short-term load forecasting is critical in modern energy systems. Machine learning and deep learning models have emerged as effective approaches for achieving this goal. This research delves into the intricacies of Short-Term Load Forecasting (STLF) in educational buildings, emphasizing the potential of transfer learning techniques in enhancing forecasting accuracy. By leveraging a dataset encompassing energy consumption patterns and temperature data, the study investigates the efficacy of various deep learning architectures, including Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs). The results underscore the nuanced impact of temperature data on model performance, with LSTMs exhibiting heightened sensitivity to this external feature. Furthermore, the study reveals the challenges of the transfer learning technique called fine-tuning such as the trade-offs between the history window and forecasting precision. While the research confirms the potential of transfer learning in STLF, it also underscores the requirement for careful feature selection and model customization based on the specific forecasting context. Limitations such as imprecise location data, computational constraints, and limited dataset scope are acknowledged. Looking forward, the paper suggests avenues for future research, including the exploration of meta-learning, attention mechanisms, and the integration of diverse external features to further refine STLF models.

1 Introduction & Motivation

Accurate forecasting of energy demand plays a crucial role in the efficient operation of the electricity grid, the pricing of energy and renewable energy integration [1]. Traditional approaches to energy demand forecasting, such as auto-regressive integrated moving average (ARIMA) and Exponential Smoothing have been shown to be effective [2]. However, these methods may encounter difficulties when attempting to forecast based on volatile, large-scale data typical in current electricity networks [3].

Deep learning (DL) models offer an alternative method for the forecasting of volatile time series due to their ability to model complex, non-linear relationships. Deep Learning models have been effectively employed in various fields, including natural language processing, image recognition and energy load forecasting [4]. Models such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) are especially adept at identifying time-related patterns in datasets [5].

However, for these DL models to produce accurate forecasting they require large quantities of training data. In the context of forecasting energy demand, new buildings lack historical data and as a result, these models are unable to short-term load forecast (STLF) energy demand effectively [6]. A solution to this challenge is transfer learning (TL), where models are trained on a source domain with available data, such as a different building, and then applied to a target domain, such as a new building without sufficient data. An example of a TL technique is fine-tuning, which entails adjusting the weights of a pre-trained model to accommodate a specific target task better [7].

This project will present the development of a transfer learning long short-term memory model (TL-LSTM), a transfer learning one-dimensional convolution neural network model (TL-1DCNN) and a transfer learning two-dimensional convolution neural network model (TL-2DCNN). These approaches will be compared against each other through a set of experiments with the objective of forecasting electricity demand for new/unseen buildings. The best-performing model will be hyperparameter-tuned and then fine-tuned. The project will then analyse and discuss these results. The TL-LSTM, TL-1DCNN, and TL-2DCNN will be implemented based on the architectures that are discussed in Section 4.5. Section 2 gives a state-of-the-art overview and details both transfer learning techniques and current literature.

The subsequent sections of this thesis are organized as follows: Chapter 2 provides an overview of the current state of the art, beginning with a literature review and then outlining the primary research question. Chapter 3 details the methods employed, from data sets used to the variety of models, including baseline and deep learning models, along with the metrics for evaluation. Chapter 4 describes the experimental setup, encompassing tools and technologies, design, data gathering and processing, feature engineering, multi-step forecasting, and optimization techniques such as the Hyperband Algorithm. Chapter 5 presents the results and a comprehensive analysis of the models' performances, the influence of weather data and fine-tuning. Chapter 6 concludes the thesis, summarizing the main contributions, challenges faced during implementation, and potential avenues for future work.

2 State of the Art

2.1 Literature Review

STLF has traditionally been performed using various statistical methodologies. Methods like ARIMA models, exponential smoothing, and regression models have all been employed with various degrees of success [8]. These methods rely on the assumption that the data is stationary, i.e., that the underlying statistical properties of the data do not change over time.

However, electricity demand is influenced by a wide range of factors like weather conditions, socioeconomic trends, and regional activities. To account for these complexities, deep learning methods such as convolutional neural networks (CNNs) and long short-term memory (LSTM) networks can be utilised. These models have the ability to learn complex temporal dependencies and non-linear relationships, which makes them especially suitable for STLF of energy demand [9].

The availability of historical data is crucial for developing effective prediction models that can provide accurate STLF in buildings [4]. However, new buildings often lack adequate historical data, posing challenges in predicting their energy consumption. TL has emerged as a potential solution by allowing one to apply knowledge gained from one domain to a different but still related domain [10].

There are several techniques for implementing TL, each with its own specific advantages and limitations. The most commonly used techniques include:

- *Feature Extraction*: The weights of a pre-trained model are used to extract meaningful features from new data, while the final output layer of the model is trained from scratch. This technique assumes that the lower-level features are general enough to be useful on a new task [11].
- *Fine-Tuning*: Unlike feature extraction, the weights of a pre-trained model are not kept fixed in fine-tuning. Instead, they are 'fine-tuned' or adjusted during training on the new task. Fine-tuning can lead to superior performance as it allows the model to adapt to the specifics of the new task. This approach is highly useful when the source and target tasks are similar but not identical [10].
- *Meta-learning*: a TL technique that seeks to learn a model's optimal parameters or architectures across multiple tasks, enabling rapid adaptation to new tasks with limited data [12]. In STLF, meta-learning can aid in developing models that swiftly adapt to new buildings, even without substantial historical data [13].

This project will be implementing fine-tuning due to the technique's flexibility and comparatively lower requirement for computational resources. By fine-tuning the model on limited data from a new building, learning's from the source building(s) are leveraged.

An example of an implementation of fine-tuning can be seen in a study conducted by Chao

Peng et al [14]. A novel approach was proposed by the name of MTE-LSTM to deal with the concern of insufficient historical data for multi-energy building load forecasting. The method uses a multi-source TL guided ensemble LSTM model in combination with a two-stage source-domain building selection strategy, a basic model ensemble prediction strategy and fine-tuning. Experiments on 17 schools revealed that the MTE-LSTM model, when applied to STLF in educational buildings and benchmarked against traditional LSTM and GRU models, exhibited a 0.61% improvement in the coefficient of determination, a 37.78% reduction in root mean square error, and a 30.69% decrease in mean absolute error. Notably, these results were achieved using just one week of target-building data for fine-tuning.

Lee et al. explored the potential of individualized modelling for short-term electric load forecasting (STLF) using deep neural network techniques and meta-learning [15]. Based on the results, the proposed models outperformed existing statistical methods such as ARIMA, one-for-all models, and traditional individual learning approaches in terms of RMSE and SMAPE. The authors consequently suggested that the latest advances in deep learning have allowed for the possibility to exploit TL and meta-learning for practical advantages in many energy forecasting problems.

Despite various TL approaches being promising, challenges remain. Factors such as differences in energy consumption patterns between industrial and residential buildings or variations in peak and valley patterns may result in significant errors. Additionally, the use of inconsistent data-sets in terms of size, features, and samples can hinder generalisation of results.

2.2 Research Question

Despite the advancements in TL for STLF, there is a notable gap in the literature: the exploration of weather conditions as a feature to enhance the performance of TL, especially when comparing CNN and LSTM models. Therefore, the primary goal is to compare the effectiveness of a LSTM network with that of both 1-dimensional and 2-dimensional CNNs and explore whether one model type demonstrates a significant advantage in accuracy over the other when applied to this specific task.

This leads us to the following primary research question:

Which, out of the TL-LSTM, TL-1DCNN and TL-2DCNN models can most effectively predict the short-term energy consumption of new buildings?

Subdividing this research question leads to the following sub-questions:
Does adding temperature as a feature improve the accuracy of the models?
Does applying fine-tuning on the best-performing model result in an increase in accuracy?

In summary, the tasks associated with this project are:
Task 1: Implementing TL-LSTM, TL-1DCNN and TL-2DCNN models and comparing them.
Task 2: Adding temperature as a feature and evaluating the models.
Task 3: Hyper-tuning the best model and then fine-tuning it on the test data.

3 Methods

3.1 Baseline Models

3.1.1 Persistence Model

The Persistence Model is often used as a baseline model for time series forecasting in the literature. An example of this can be seen in [16]. At its core, this model predicts that the next value in a sequence will be the same as the most recent observed value. Mathematically, this can be represented as:

$$\hat{y}_{t+1} = y_t \tag{1}$$

where $\hat{y}_{t+1}$ is the forecasted value for the next time step and y_t is the observed value at the current time step.

In the context of STLF, the model is often used as a baseline when compared to more complex models such as LSTM and CNN models [17]. If sophisticated models cannot outperform this naive approach, it may indicate challenges in the data or in the modelling approach.

Furthermore, the performance of the persistence model can provide insights into the nature of the time series data. For instance, if the persistence model performs exceptionally well, it might suggest that the series has strong auto-regressive properties and that the most recent values are highly indicative of future values.

3.1.2 ARIMA Model

The ARIMA model, which stands for Auto Regressive Integrated Moving Average, is a widely used approach for time series forecasting [2]. It combines autoregressive (AR) and moving average (MA) models, as well as differencing of the data to make it stationary (Integrated).

Before implementing ARIMA, it is first necessary to know whether the data is stationary or not. This can be done using the Augmented Dickey-Fuller. This implementation is discussed in Section 4.7.2.

Mathematically, an ARIMA model is often represented as $ARIMA(p, d, q)$, where:

- p is the number of autoregressive terms,
- d is the number of nonseasonal differences, and
- q is the number of lagged forecast errors in the prediction equation.

Due to ARIMA's versatility of being able to model both trend and seasonality for various time series data, it has become prominent in time series forecasting and often serves as a benchmark [2]. Therefore, before developing deep learning models like LSTM and CNN, establishing a performance baseline with ARIMA can be insightful.

Similarly to the persistence model, an ARIMA model outperforming pre-trained LSTM or CNN models indicates that the data may be autoregressive or the architecture of the models needs to be adapted.

3.2 Deep Learning Models

3.2.1 LSTM

The LSTM model, first introduced by Hochreiter and Schmidhuber in 1997 [18], is a variety of recurrent neural networks (RNN), a class of networks that have an infinite impulse response where the output depends not only on the current and previous input values but also on previous output values. In the context of STLF, LSTMs are particularly relevant due to their ability to capture long-term dependencies in time series data. Traditional RNNs struggle to do so and as a result, encounter the vanishing gradient problem where the weights of each layer of the network are not updated reducing accuracy in the final prediction.

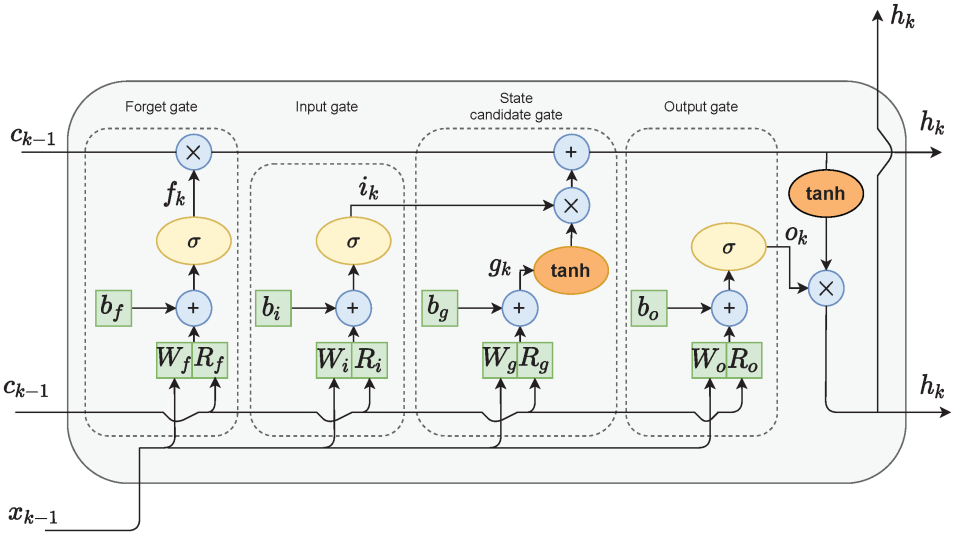

Figure 1: A diagram depicting a typical LSTM architecture [19].

In Figure 1, a standard LSTM unit is delineated, comprising a forget gate (f), an input gate (i), a cell state candidate gate (g), and an output gate (o).

The forget gate decides whether to retain or discard information from the prior state by evaluating the previous state against the current input and assigning a value between 0 and 1. A value close to 1 retains the information, whereas a value near 0 discards it. The input gate selects values from the preceding hidden state and the present input, passing them through the sigmoid function, and then multiplies the result by the previous cell state. The cell state candidate gate controls information flow by applying the tanh function to the previous hidden state and current input, multiplying the result by the input gate output, and then adding it to the previous cell state. The output gate determines what information from the current state should be conveyed, assigning values from 0 to 1 based on both prior and current states.

As shown in Figure 1, the cell state undergoes only a few linear operations, so the gradient flow during back-propagation training is mostly uninterrupted, mitigating the vanishing gradient problem. Overall, the LSTM's architecture enhances its ability to model complex time dependencies in load data by selectively retaining or forgetting patterns.

The individual components in the LSTM unit described mathematically in [18] can be seen below:

$$i(k) = \sigma(W_i x(k) + R_i h(k-1) + b_i) \tag{2}$$
$$f(k) = \sigma(W_f x(k) + R_f h(k-1) + b_f) \tag{3}$$
$$g(k) = \tanh(W_g x(k) + R_g h(k-1) + b_g) \tag{4}$$
$$o(k) = \sigma(W_o x(k) + R_o h(k-1) + b_o) \tag{5}$$

The new cell state at the time instant value k is then defined as:

$$c(k) = f(k) \circ c(k-1) + i(k) \circ g(k) \tag{6}$$

The hidden state at the time instant value k can be defined as:

$$h(k) = o(k) \circ \tanh(c(k)) \tag{7}$$

3.2.2 CNN

In contrast to RNN, CNN refers to a class of networks that have finite impulse responses [20]. Unlike LSTMs, which are designed to capture sequential dependencies, CNNs excel at identifying spatial hierarchies or patterns in data [21].

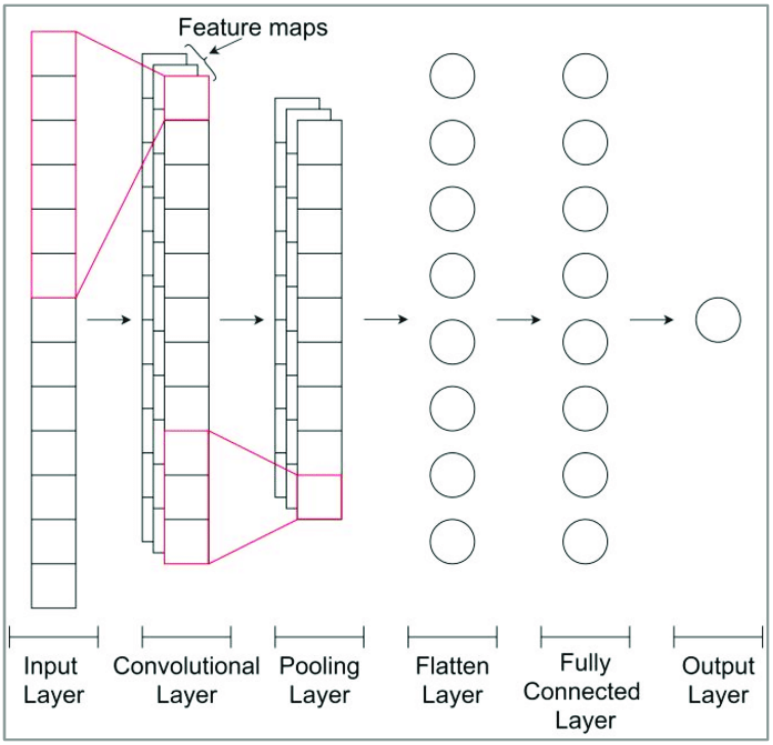

Figure 2: 1D-CNN architecture [22]

As depicted in Figure 2, a typical CNN architecture comprises of three primary types of layers: convolutional layers, pooling layers, and fully connected layers.

- **Convolutional Layer**: This layer applies a set of filters to the input data. Each filter, also known as a kernel, convolves around the input to produce a feature map. In the context of time-series data:
  - **1D CNNs** slide the kernels along the temporal dimension, enabling the model to learn important temporal features, such as trends or patterns within each window of data.
  - **2D CNNs** slide the kernels in both temporal and feature dimensions, capturing patterns across multiple features over time.
- **Pooling Layer**: Following the convolutional layer, the pooling layer reduces the size of the feature maps, decreasing computational demands and mitigating overfitting. Pooling operates independently on each feature map. For time-series data:
  - **1D Pooling** is applied across the temporal dimension of each feature map.
  - **2D Pooling** is applied across both the temporal and feature dimensions.
- **Fully Connected Layer**: This layer connects every neuron in one layer to every neuron in the subsequent layer. It often performs the final prediction based on the high-level features learned by the preceding layers.

The design of CNNs ensures the preservation of temporal relationships between data points by learning features using small windows of input data. This characteristic makes CNNs suitable for capturing temporal dependencies in data that allows them to effectively forecast time-series.

3.3 Evaluation Metrics

Predicting STLF in school buildings is approached as a regression problem, necessitating quantitative metrics to assess model accuracy. This study employs three standard regression metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE) and an approximate calculation of Mean Absolute Percentage Error (MAPE).

3.3.1 MAE & MSE

MAE is defined as the average of the absolute differences between the predicted and actual values. It provides an error measure in the same units as the dependent variable. It can be described by the following equation:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8}$$

MSE calculates the average of the squared differences between the predicted and actual values, emphasizing larger errors over smaller ones due to its squared term. It can be described by the following equation:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{9}$$

MAE offers a direct interpretation by measuring the average absolute differences between predicted and actual values, treating all errors uniformly. In contrast, MSE emphasizes larger deviations due to its quadratic nature, making it sensitive to outliers and especially useful when larger forecasting errors are less desirable.

3.3.2 MAPE

MAPE, is defined as the average of the absolute percentage differences between the predicted and actual values. It provides an error measure in percentage terms, giving a relative understanding of the prediction error. The MAPE can be described by the following equation:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{10}$$

However, a challenge arises when using the MAPE with datasets that contain zero values. The original equation for MAPE involves dividing by the actual value, y_i. When this value is zero, the division results in undefined behavior, making the MAPE calculation impossible. One might consider adding a small value to the zeros to avoid this issue, but this approach is not ideal as adding an arbitrary value can distort the actual MAPE, leading to inaccurate results.

To address this, a modification can be used. Instead of relying on the original MAPE formula, the MAE is divided by the mean of the data. This approach provides a relative error measure without the complications arising from zero values in the dataset. The Modified MAPE (mMAPE) can be described as:

$$\text{mMAPE} = \frac{\text{MAE}}{\bar{y}} \tag{11}$$

where $\bar{y}$ is the mean of the actual values in the dataset.

4 Experimental Setup

4.1 Data Set

To train the models, the EnerNOC dataset comprising 100 anonymous buildings' energy consumption recorded at 5-minute intervals for the year 2012 will be utilised [23]. Additional metadata for each building such as the square footage, geographical coordinates and the industry sector the building operates within are also supplied. For the purpose of this thesis, only school buildings were trained, validated and tested on. Selecting how the data is processed and selected will be elaborated upon in subsection 4.2.

To test whether or not features based on the weather can improve the performance of the TL-LSTM and TL-CNN models, temperature data needs to be found and integrated into the problem input. Temperature data for the year 2012 will be sourced from an API called: Open Meteo [24] that requests on input the longitude and latitude and in return provides the temperature in celisus per hour in that location.

4.2 Tools and Technologies

Several tools and technologies were employed to facilitate data processing and model development. Python was utilized as the primary programming language due to its extensive libraries and capabilities. For data manipulation and exploration, the Pandas library was used with the modeling process being conducted using TensorFlow and Keras. TensorFlow, an open-source machine learning framework, provided the necessary infrastructure to design and train neural network architectures. Concurrently, Keras, a high-level neural networks API, simplified the development process with its modular approach. These tools collectively supported the research and experimentation conducted in this thesis.

4.3 Experiment Design

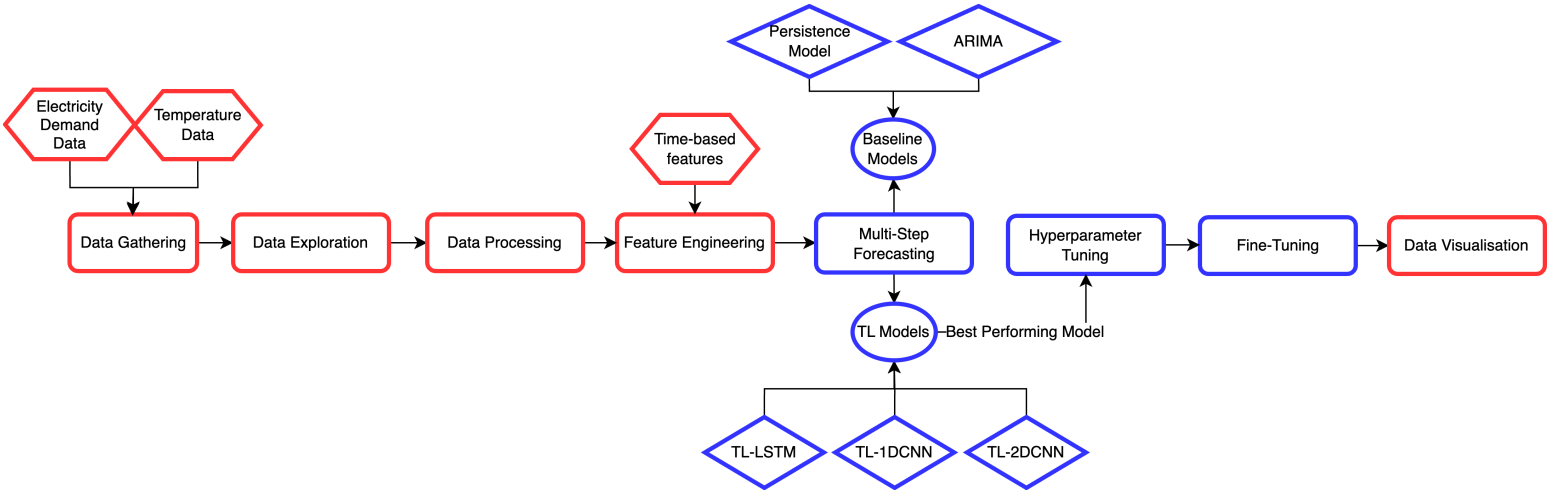

Figure 3: The experiment setup and procedure.

Figure 3 denotes the step by step process by which this project was implemented. Steps in blue indicate segments of the project where baseline or DL models were implemented and evaluated. Steps in red indicate segments of the project where data was transformed and or analysed. The specifics of each step are described below.

4.4 Data Related

4.4.1 Data Gathering

For the data gathering stage, temperature values were sourced from the weather dataset from OpenMeteo [24] and records of energy consumption for buildings were extracted from the EnerNoc [23] dataset. A total of 25 schools were sourced from three different cities, namely: New York, Chicago and Denver.

4.4.2 Data Exploration

During data exploration, temperature data of the three different locations of schools is analysed. Trends are found across all schools, such as a decrease in energy demand during the weekends and summer holidays. Additionally, the electricity demand data for schools was checked for missing values. Each school exhibited less than 1% of missing values as a ratio of the total values for that school. With the longest consecutive period of missing values across all schools being smaller than 0.5%, all schools were kept as part of the data set.

4.4.3 Data Processing

Any missing values in the dataset were filled using backwards and forward filling. This was done prior to re-sampling the data into watts per hour per square foot of the school from the originally recorded kilowatts per 5-minute electricity demand. This was done to provide a more standardized and interpretable measure.

4.5 Feature Engineering

There are various ways of splitting the training, validation and testing sets as noted in the literature [25]. For the purpose of this thesis, the training, validation and testing set were split $60\%, 20\%, 20\%$ respectively.

4.5.1 Windowing

The sliding window approach was implemented for inputting data into the model with a window size of 168 time steps, representing one week. The objective was to forecast 24 time steps ahead.

For multi-school data, individual windows were extracted for each school shifted by one hour. During training, these windows were presented to the model in a randomized order. This strategy is an attempt to prevent the DL models from overfitting to patterns specific to a single school.

4.5.2 Feature Experimentation: Temperature

Based on the literature, the temperature has been shown to be the most important for STLF when compared against other weather features [26]. Adding additional weather features may add more noise to the data and can introduce multi-collinearity, where features are correlated with each other, making it harder to discern the individual impact of each feature.

To assess the impact of external temperature on electricity demand, an experiment is conducted where the models are trained and evaluated both with and without the inclusion of the temperature feature. This allows for a comparative analysis to understand the significance of temperature in predicting electricity demand.

4.5.3 Train Set

The training set is composed of electricity demand data from 15 schools for the entire year of 2012. These schools span across three locations in the US, the cities of New York, Chicago and Denver. In addition, the training set includes other features such as the one-hot encoded variables for the day of the week, month, and school holidays. These features are one-hot encoded to avoid unintended ordinal interpretations and ensure each category is treated distinctly by the model. Additionally, the hour of the day is circularly encoded to capture the cyclical nature of time. In total, the training set comprises of 23 features.

The features can be seen in the table below:

Features	**Continuous**	**Cyclical**	**One-Hot Encoded**
Electricity demand, temperature	✓		
hour in sin, hour in cos		✓	
day of the week {0,1,...,6}			✓
month {0,1,...,12}			✓
school holiday			✓

Table 1: Feature categorization based on encoding type.

4.5.4 Test Set & Validation Set

Both the test and validation sets include electricity demand for the year 2012 of 5 different schools, carefully chosen so that all three locations are present. These sets are used to evaluate the model's performance and to ensure that it generalizes well to new, unseen data.

The selection of schools for each set can be seen in the table below:

Set	SITE_ID	TIME_ZONE
Train	197	New York
	103	New York
	218	Chicago
	92	New York
	100	New York
	99	Chicago
	109	Chicago
	137	New York
	186	New York
	224	Chicago
	270	New York
	153	New York
	228	Denver
	101	Chicago
	88	New York
Test	213	Denver
	275	New York
	116	Chicago
	111	New York
	259	New York
Validation	144	New York
	136	New York
	214	Chicago
	236	Denver
	217	New York

Table 2: School ID and their corresponding TIME_ZONE for Train, Test, and Validation sets.

After employing this partitioning strategy, the training set underwent a min-max scaling transformation. The same minimum and maximum values derived from the training set were then used to scale the test and validation sets, ensuring that no data leakages occur.

4.6 Multi-Step Forecasting

4.6.1 Persistence Model

For each entry in the test dataset, the persistence model extracts the last 24 hours of energy consumption data and uses these as the prediction for the next 24 hours. This assumes that energy consumption remains constant and that the last observed value continues for the specified prediction horizon.

After obtaining predictions for all the test data points, the model's performance is assessed using the MAE, MSE and mMAPE for the 24-hour forecasts of each school in the test set. The final results present the average MAE, MSE and mMAPE values, derived from the individual errors of all schools.

4.6.2 ARIMA

The ARIMA model is employed to predict short-term energy consumption patterns across various schools. Each school is analyzed as an individual time series, with a distinct ARIMA model trained and assessed for its data.

For each school, an initial period of 168 hours, equivalent to seven days, is selected as the training set. This week-long span is chosen to capture latent regularities and inherent seasonal fluctuations in the energy consumption metrics. After this training phase, the ARIMA model predicts energy usage for the subsequent 24 hours, defining this as the test dataset.

Before implementing the ARIMA model, it is first necessary to know whether the data is stationary or not. This can be done using the Augmented Dickey-Fuller test on the electricity demand values of the dataset. All the schools had a p-value below the significance level of 0.05 suggesting that the data is stationary. As a result, the ARIMA model is configured with specific hyperparameters considering autoregressive components up to a fifth-order lag, with the differencing term is set to zero, indicating an assumption of the series being stationary. A stepwise search approach is utilized to efficiently select the best hyperparameters.

Predictions are made for 24 timesteps into the future. These forecasts, along with the actual energy consumption values for the same period, are recorded. This method is applied consistently for each school in the test set, leading to a series of forecasts and actual consumption values for each school.

The model's performance is assessed using the MAE, MSE and mMAPE for the 24-hour forecasts of each school. The final results present the average MAE, MSE and mMAPE values, derived from the individual errors of all schools.

4.6.3 TL-LSTM

The architecture of the LSTM network utilized in this study is derived from the implementation of a transfer learning LSTM Model for Building Energy Demand Forecasting [27]. The architecture was adapted with the last dense layer having 24 neurons to fit the multi-step forecasting objective of the task.

Layer(neurons)	Output Shape	Param #
LSTM(8)	(None, 168, 8)	1056
LSTM(16)	(None, 168, 16)	1600
LSTM(32)	(None, 168, 32)	6272
LSTM(32)	(None, 168, 32)	8320
LSTM(64)	(None, 64)	24832
Dropout(dropout = 0.5)	(None, 64)	0
Dense(24)	(None, 24)	1560
Total params: 43,640		
Trainable params: 43,640		
Non-trainable params: 0		

Table 3: LSTM Architecture

4.6.4 TL-1DCNN

The architecture of the CNN network utilized in this study is derived from a research paper on Time Series Classifications and the use of a DL models [28]. The architecture was adapted with the last dense layer having 24 neurons to fit the multi-step forecasting objective of the task.

Layer(filter size, kernel size)	Output Shape	Param #
Conv1D(128, 8)	(None, 168, 128)	24,704
Conv1D(256, 5)	(None, 168, 256)	164,096
Conv1D(128, 3)	(None, 168, 128)	98,432
GlobalAveragePooling1D	(None, 128)	0
Dense(units = 24)	(None, 24)	3,096
Total params: 290,328		
Trainable params: 290,328		
Non-trainable params: 0		

Table 4: 1D CNN Model Architecture

4.6.5 TL-2DCNN

The architecture of the CNN network utilized in this study is adapted from the known architecture called VGGNET [29] that is typically used for image classification. The architecture was adapted with the last dense layer having 24 neurons to fit the multi-step forecasting objective of the task.

Layer(filter, kernel size, dilation rate)	**Output Shape**	**Param #**
Conv2D(64, (3,3), 1)	(None, 168, 24, 64)	640
Conv2D(64, (3,3), 1)	(None, 168, 24, 64)	36,928
MaxPooling2D(poolsize = (2,2))	(None, 84, 12, 64)	0
Conv2D(128, (3,3), 1)	(None, 84, 12, 128)	73,856
Conv2D(128, (3,3), 1)	(None, 84, 12, 128)	147,584
MaxPooling2D(poolsize = (2,2))	(None, 42, 6, 128)	0
Flatten	(None, 32256)	0
Dense(units = 256)	(None, 256)	8,257,792
Dropout(dropout = 0.5)	(None, 256)	0
Dense(units = 24)	(None, 24)	6,168
Total params: 8,522,968 Trainable params: 8,522,968 Non-trainable params: 0		

Table 5: VGGNET adapted architecture

4.6.6 Hyper-parameter Tuning of TL-2DCNN

Hyperparameter tuning is the process of systematically searching for the best combination of hyperparameters that optimize a model's performance for a given task [30]. Unlike model parameters, which are learned during training, hyperparameters are set prior to the training process and remain constant during it. Examples include learning rate, batch size, and the architecture of the model itself.

The algorithm used for hyperparameter tuning in this study is the Hyperband algorithm [31]. Hyperband is an adaptive resource allocation and early-stopping strategy to accelerate random search. It dynamically allocates resources to configurations based on their performance, allowing the algorithm to focus on more promising configurations. This results in a more efficient search over the hyperparameter space.

The specific architecture and Hyper-parameter tuning is described below:

1. **Model Architecture**:
    - The model is built using the Sequential API and is named "VGG-Custom."
    - The input shape is determined by the `timesteps` and `features`, which in our case are 168 and 24 respectively.
    - Multiple VGG-like blocks are stacked, with the depth controlled by the `depth` hyperparameter.
    - The model concludes with a Flatten layer, followed by a Dense layer with 256 units and a Dropout layer with a rate of 0.5. Finally, a Dense layer with 24 units and a sigmoid activation function is added.

2. **VGG-like Block**:
    - The function `vgg_block` defines a VGG-like block consisting of a series of convolutional layers followed by a max-pooling layer.
    - The number of convolutional layers is set to 2, and the number of filters, kernel size, and dilation rate are treated as hyperparameters.
3. **Hyperparameter Search Strategy**:
    - The Hyperband optimization algorithm is utilized for hyperparameter tuning.
    - The objective of the tuning is to minimize the validation loss (`val_loss`).
    - A maximum of 15 epochs is set for each trial, and the search factor is set to 3.
4. **Search Execution**:
    - An early stopping callback is used, which monitors the validation loss. The training will halt if the validation loss does not improve for 5 consecutive epochs.
    - The search is executed using a training data generator (`train_gen`) and is validated against a separate validation dataset (`X_val`, `y_val`).
5. **Model Compilation**:
    - The model is compiled using the MSE as the loss function and MAE as an additional metric.
    - The Adam optimizer is used with a learning rate of 1×10^{-3}.

4.6.7 Fine-Tuning

Fine-tuning is implemented on the hypertuned TL-2DCNN model to adapt the model further to the electrictiy demand of the unseen schools. To test the technique on individual schools, schools A and B with two different locations, Denver and New York respectively, are taken from the test set. This process of fine-tuning is described below.

1. **School Selection**:
    - Schools A and B, situated in distinct climatic zones of the US (Denver and New York), are chosen from the test set to ensure model generalization across varying environmental conditions.
2. **Train and Test Set Formation**:
    - A parameter N is defined, which can take values of 2, 4, 8, or 24 weeks.
    - The training datasets X_{train} and y_{train} are formed using data from the duration specified by N from the existing test set.
    - After the duration N, the remaining data of the specific School is used to create new test datasets X_{test} and y_{test}.
    - This procedure is first executed for School A and then for School B.

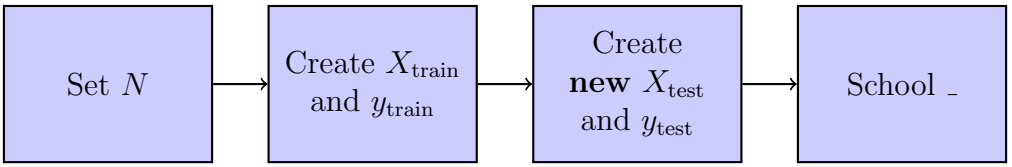

Figure 4: Flow diagram of the train and test set formation for fine-tuning for a School.

3. **Model Configuration**:
   - All layers of the TL-2DCNN model, barring the final one, are rendered non-trainable.
   - A learning rate of 1×10^{-5} is set to prevent possible overfitting.
   - The model undergoes training for 10 epochs on X_{train} and y_{train} and then evaluated against the y_{test} set. The optimal model configuration is retained with the other configurations in Appendix _.
4. **Evaluation Procedure**:
   - Post fine-tuning, the model is assessed on the y_{test} set of the respective school, excluding the training set values.
   - The overall MAE, MSE and mMAPE is found.

5 Results and Analysis

The objective of this research was to discern the impact of incorporating temperature data on the performance of different neural network architectures for TL STLF. A clear pattern was evident: introducing temperature data as an input feature consistently improved the forecasting accuracy of all models.

For the sake of clarity, models ending with _T include temperature as a feature and models starting with HT_ have been hyperparameter tuned.

5.1 Model Performance Assessment Prior to Fine Tuning

Each model's performance was gauged using the MAE, MSE and mMAPE metrics. The table and graph below consolidate the performance metrics for all models being evaluated on the test set of unseen schools before hyperparameter tuning and fine-tuning:

Model	**MAE**	**MSE**	**Approx. MAPE(%)**	**Percentage vs. ARIMA MAE (%)**	**Percentage vs. ARIMA MSE (%)**
Persistence	16.407	1086.617	21.29	-	-
ARIMA	14.444	443.750	18.74	0.00%	0.00%
TL-1DCNN	15.659	579.473	20.32	8.41%	30.59%
TL-1DCNN-T	15.068	503.665	19.55	4.32%	13.50%
TL-LSTM	13.740	467.073	17.83	-4.87%	5.26%
TL-LSTM-T	13.128	419.350	17.03	-9.11%	-5.50%
TL-2DCNN	12.714	385.305	16.50	-11.98%	-13.17%
TL-2DCNN-T	12.693	380.108	16.47	-12.12%	-14.34%

Table 6: Results of all models with and without temperature data as a feature

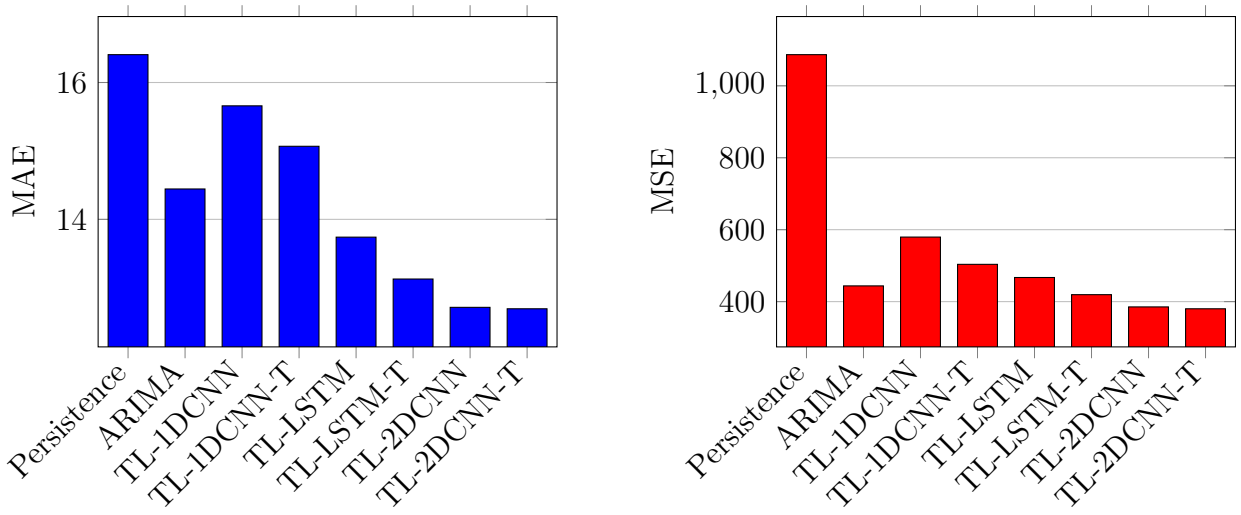

Figure 5: Performance metrics of the models with and without temperature data

Additionally, the training and validation losses can be seen in **Appendix A** for all the DL models in the form of a graph.

5.2 Impact of Temperature as a Feature

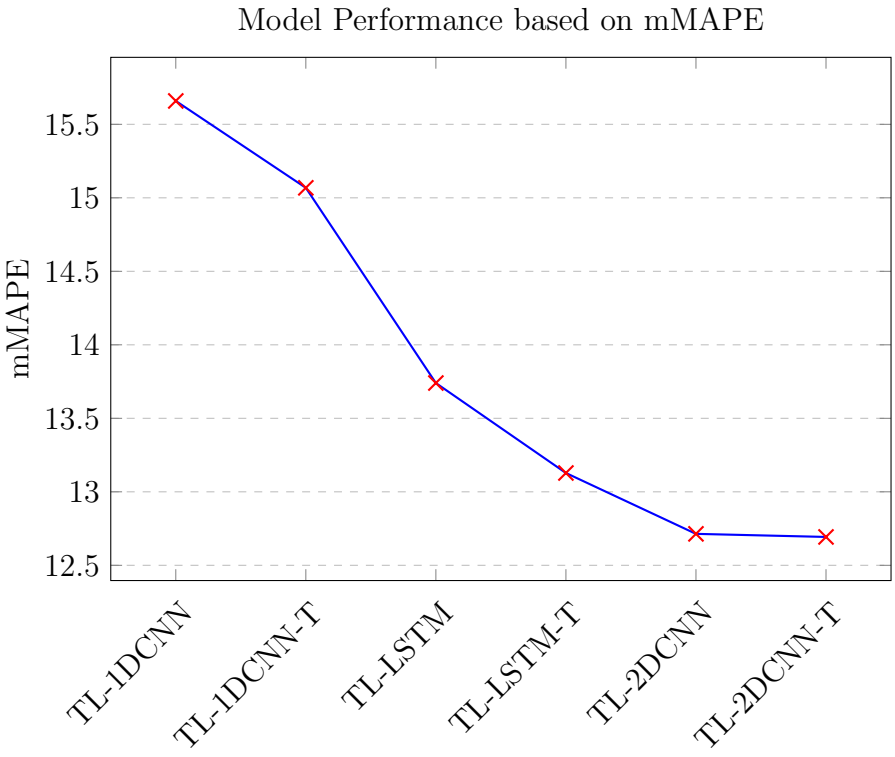

Figure 6: A graph of the mMAPE values for all DL models.

Model Complexity vs. Temperature Data Impact: As model complexity increases, the relative benefit of incorporating temperature data diminishes as can be seen in Figure 5.

For example, while the TL-1DCNN model exhibited a 0.77% reduction in MAPE with temperature, the more complex TL-2DCNN model registered only a 0.03% reduction. This trend can be attributed to the capacity of more complex models being capable of extracting intricate patterns and relationships from the existing features. Therefore, the incremental value added by an additional feature like temperature becomes marginal.

Metric Consistency: Some models, such as TL-LSTM, demonstrated a decrease in the Percentage vs. ARIMA MAE metric but an increase in the Percentage vs. ARIMA MSE metric when the temperature feature was not included as seen in Table 4. This discrepancy suggests that LSTM models may be more sensitive to outliers and that the addition of the temperature feature has improved the LSTM model more significantly than the other models. This hypothesis is also supported by the inherent characteristic of LSTM models having the ability to learn long-term dependencies in sequence data. An outlier, being a data point that deviates significantly from other observations, can introduce a strong temporal dependency that the LSTM might focus on. Consequently, while the model may generally improve in terms of MAE, the MSE can increase due to the amplified impact of these outliers.

LSTM vs. CNN Performance: LSTMs, tailored for sequence data, displayed a more pronounced advantage from temperature data compared to CNNs. Specifically, the TL-LSTM model achieved a 4.87% decrease in MAE with temperature, surpassing the 4.32% decrease of the TL-1DCNN model. This can be attributed to the inherent design of LSTMs which are adept at capturing long-term dependencies in time-series data. Temperature, being a temporal variable with potential lagged effects on energy consumption, is more effectively integrated and leveraged by LSTMs. In contrast, CNNs, primarily designed for spatial hierarchies in data like images, might not be as efficient in extracting the sequential patterns introduced by temperature variations.

Performance Limitations: The models, regardless of architecture or temperature data inclusion, converged towards similar performance metrics, hinting at a potential predictive ceiling with the current dataset.

5.3 Hyper-parameter Tuning Results

With the TL-2DCNN-T model performing better than the other deep-learning and baseline models it was selected to be hyper-tuned and then fine-tuned on the test data.

After hyper-parameter tuning the TL-2DCNN architecture described in Section 4.5.5, the resulting HT-TL-2DCNN-T architecture can be seen below.

Layer(filter, kernel size, dilation rate)	**Output Shape**	**Param #**
Conv2D(32, (3,3), 1)	(None, 168, 24, 32)	320
Conv2D(32, (3,3), 1)	(None, 168, 24, 32)	9,248
MaxPooling2D(poolsize=(2,2))	(None, 84, 12, 32)	0
Conv2D(64, (1,3), 1)	(None, 84, 12, 64)	6,208
Conv2D(64, (1,3), 1)	(None, 84, 12, 64)	12,352
MaxPooling2D(poolsize=(2,2))	(None, 42, 6, 64)	0
Flatten	(None, 16128)	0
Dense(units = 256)	(None, 256)	4,129,024
Dropout(dropout=0.5)	(None, 256)	0
Dense(units=24)	(None, 24)	6,168
Total params: 4,163,320		
Trainable params: 4,163,320		
Non-trainable params: 0		

Table 7: VGG-Tuned architecture

The results of testing the HT-TL-2DCNN-T model on the test data of unseen schools can be seen below:

Model	**MAE**	**MSE**	**mMAPE(%)**	**vs. TL-2DCNN-T MAE (%)**	**vs. TL-2DCNN-T MSE (%)**	**vs. TL-2DCNN-T mMAPE (%)**
HT-TL-2DCNN-T	12.530	367.288	16.26	-1.29%	-3.37%	-0.21%

Table 8: Results of the HT-TL-2DCNN-T

Additionally, the training and validation loss can be seen in **Appendix B** for the HT-TL-2DCNN-T models in the form of a graph.

Resource Efficiency: Despite the computational overhead of hyper-tuning, the improvements in MAE, MSE, and mMAPE are marginal, suggesting a diminishing return on computational investment.

Parameter Reduction: The term parameters refers to the weights and biases in the neural network model that are learned during the training process. The HT-TL-2DCNN-T has nearly 50% fewer parameters than the non-hyperparameter tuned architecture but achieves slightly better performance. This reduction leads to faster computing times and is advantageous for deployment in resource-constrained settings.

5.3.1 Fine-Tuning

Fine-tuning was implemented on the HT-TL-2DCNN-T to assess the impact of incrementally increasing the number of weeks used to train from a singular school's test set. The methodology adhered to is described in Section 4.5.7.

Schools	**No. of weeks used to fine-tune**	**MAPE(%)**	**vs. HT-TL-2DCNN-W mMAPE (%)**
School ID: 101	2 weeks	15.22	-1.04
	4 weeks	15.01	-1.25
	8 weeks	14.57	-1.69
	24 weeks	14.07	-2.19
	Overall	14.72	-1.54
School ID: 111	2 weeks	15.59	-0.66
	4 weeks	16.25	0.00
	8 weeks	16.48	0.22
	24 weeks	17.17	0.91
	Overall	16.37	0.12

Table 9: Fine-tuning results for Schools ID: 101 and 111

Fine-tuning Duration Impact: For School ID: 101, there's a clear trend that as the duration of fine-tuning increases, the mMAPE decreases. This suggests that longer fine-tuning periods lead to better model performance for this dataset. However, this trend is not observed for School ID: 111, where the mMAPE increases as the fine-tuning duration extends, especially noticeable with the jump from 8 weeks to 24 weeks. This indicates that the effectiveness of fine-tuning can vary significantly between schools and that a "one-size-fits-all" approach may not be optimal.

Optimal Fine-tuning Duration: For School ID: 101, 24 weeks appears to be the optimal fine-tuning duration, yielding the lowest mMAPE. This could be attributed to the model having a more extended period to adapt to the specific characteristics and patterns of the data from School ID: 101, allowing it to make more accurate predictions. Conversely, for School ID: 111, the optimal duration is less clear, with 2 weeks of fine-tuning providing the best results. This might suggest that prolonged fine-tuning for School ID: 111 could lead to overfitting or that the model quickly captures the essential patterns in the data within the initial weeks. Notably, the 2-week fine-tuned model is only marginally better than the non-fine-tuned model.

Comparison with Other Models: When compared with results from other models like TL-2DCNN-T, the fine-tuned models, especially for School ID: 101, show competitive or even superior performance. This underscores the potential benefits of school-specific fine-tuning, at least for certain datasets.

While artificially extending the data for new buildings proves effective for model testing, it poses challenges in real-world applications. Expecting 24 weeks of data for a recently constructed building is unrealistic. In real applications, new buildings might only have limited data available, making it imperative to adjust the model to train efficiently on specific data lengths tuned to the task at hand. Achieving a universal model that delivers consistent results across varying data lengths remains a difficult task to accomplish.

6 Conclusion

6.1 Summary of Main Contributions

This projects primary objective was to research and experiment the comparison of the efficacy of CNNs and LSTM models for STLF of new buildings. Additionally, this thesis investigated the impact of temperature as a feature and whether or not improved the performance of the DL models. The study has successfully implemented fine-tuning, to address challenges associated with forecasting in new or unseen buildings that lack substantial historical data. The models developed, including TL-LSTM, TL-1DCNN, and TL-2DCNN, have been tested and analyzed, with results shedding light on the potential and limitations of each approach.

6.2 Addressing the Research Questions

The primary research question posed was: "Which, out of the TL-LSTM, TL-1DCNN, and TL-2DCNN models can most effectively predict the short-term energy consumption of new buildings?" Based on the findings, each of the models demonstrated varying degrees of effectiveness, with certain architectures benefiting more from the inclusion of temperature data. However, although the 2DCNN model performed the best across the evaluation metrics, no single model emerged as the definitive best performer across all scenarios, indicating that the choice of model may need to be tailored based on specific building characteristics and available data.

The first sub-question inquired about the impact of adding temperature as a feature on model accuracy. The results indicated that temperature data generally enhanced model performance, but its influence varied across different architectures. For instance, LSTM models, which are inherently designed for sequence data, showed a more pronounced improvement with temperature data compared to CNNs.

The second sub-question focused on the potential accuracy gains from fine-tuning the best-performing model. The fine-tuning process, especially when applied over varying durations, yielded mixed results. For some buildings, like School ID: 101, extended fine-tuning durations led to improved accuracy, while for others, like School ID: 111, the benefits were not evident.

In summary, while the research provided valuable insights into the potential of TL-LSTM, TL-1DCNN, and TL-2DCNN models for STLF in new buildings, it also highlighted the complexities involved. The effectiveness of a model can be influenced by multiple factors, including the nature of the data, the inclusion of external features like temperature, and the fine-tuning process. As such, while certain general trends and recommendations can be derived, a one-size-fits-all solution remains elusive.

6.3 Limitations

Deep learning models, despite their prowess in handling complex temporal dependencies, are heavily reliant on a large number of data for accurate forecasting. As the electricity demand dataset only gave access to a year's worth of electricity demand, the models may have not been

able to capture all seasonal and annual variations in energy consumption. This limitation could have affected the models' ability to generalize across different time periods, leading to potential inaccuracies in forecasts that span multiple seasons or years.

Furthermore, the provided location of schools was not exact, which potentially led to misalignment with the temperature data. This misalignment could have introduced noise into the models, especially those that heavily relied on temperature as a feature, affecting the accuracy of forecasts.

Computational constraints limited the extent of model fine-tuning. This meant that potentially better-performing model configurations remained unexplored. The impact of this limitation was most evident when comparing the performance of the hypertuned models against their non-hypertuned counterparts.

6.4 Future Work

Future endeavors in this domain could delve deeper into other transfer learning techniques, such as meta-learning, which offers rapid adaptation to new tasks with limited data. There's also potential in exploring individualized modeling for STLF using deep neural network techniques. Techniques such as attention mechanisms, which have shown promise in other time series forecasting tasks, could be adapted to the STLF domain to capture intricate temporal dependencies in energy consumption data.

Additionally, expanding the dataset to include more diverse buildings and energy consumption patterns would provide a more comprehensive understanding of the models' capabilities. Incorporating data from buildings with different functionalities, sizes, and geographical locations can offer insights into how these factors influence short-term load forecasting.

Lastly, integrating other external features beyond temperature, such as occupancy data, building operational schedules, or local events, could enhance the models' predictive accuracy. Such features can capture the dynamic nature of energy consumption in buildings, which often fluctuates based on human activities and external events.

Bibliography

[1] Y.-K. Juan, P. Gao, and J. Wang, "A hybrid decision support system for sustainable office building renovation and energy performance improvement," *Energy and Buildings*, vol. 42, no. 3, pp. 290–297, 2010.

[2] I. Ghalehkhondabi, E. Ardjmand, G. Weckman, *et al.*, "An overview of energy demand forecasting methods published in 2005–2015," *Energy Syst*, vol. 8, pp. 411–447, 2017.

[3] H. K. Alfares and M. Nazeeruddin, "Electric load forecasting: Literature survey and classification of methods," *International Journal of Systems Science*, vol. 33, no. 1, pp. 23–34, 2002.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[5] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv:1703.04691 [stat.ML]*, 2017.

[6] Y.-K. Juan, P. Gao, and J. Wang, "A hybrid decision support system for sustainable office building renovation and energy performance improvement," *Energy and Buildings*, vol. 42, no. 3, pp. 290–297, 2010.

[7] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196197–196211, 2020.

[8] H. Hippert, C. Pedreira, and R. Souza, "Neural networks for short-term load forecasting: a review and evaluation," *IEEE Transactions on Power Systems*, vol. 16, no. 1, pp. 44–55, 2001.

[9] S. Yi, H. Liu, T. Chen, J. Zhang, and Y. Fan, "A deep lstm-cnn based on self-attention mechanism with input data reduction for short-term load forecasting," *IET Generation, Transmission & Distribution*, vol. 17, no. 7, pp. 1538–1552, 2023.

[10] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[11] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Department of computer science and operations research, U. Montreal*, 2011.

[12] M. Jiang, F. Li, and L. Liu, "Continual meta-learning algorithm," *Applied Intelligence*, vol. 52, pp. 4527–4542, 2022.

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, 11 2014.

[14] C. Peng, Y. Tao, Z. Chen, Y. Zhang, and X. Sun, "Multi-source transfer learning guided ensemble lstm for building multi-load forecasting," *Expert Systems with Applications*, vol. 202, p. 117194, 2022.

[15] E. Lee and W. Rhee, "Individualized short-term electric load forecasting with deep neural network based transfer learning and meta learning," *IEEE Access*, vol. 9, pp. 15413–15425, 2021.

[16] P. J. Axaopoulos and G. T. Tzanes, "2.04 - wind energy potential (measurements, evaluation, forecasting)," in *Comprehensive Renewable Energy (Second Edition)* (T. M. Letcher, ed.), pp. 79–103, Oxford: Elsevier, second edition ed., 2022.

[17] F. Dama and C. Sinoquet, "Time series analysis and modeling to forecast: a survey," 2021.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[19] K. Zarzycki and M. Ławryńczuk, "Lstm and gru neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors," *Sensors*, vol. 21, no. 16, p. 5625, 2021.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," 1989.

[21] A. L'Heureux, K. Grolinger, and M. A. M. Capretz, "Title of the article (replace with the actual title)," *Energies*, vol. 15, no. 14, p. 4993, 2022.

[22] E. Chaerun Nisa and Y.-D. Kuan, "Comparative assessment to predict and forecast water-cooled chiller power consumption using machine learning and deep learning algorithms," *Sustainability*, vol. 13, p. 744, 01 2021.

[23] "Energy solutions for people, governments and businesses," *Enel X Corporate*. 22/07/2023.

[24] Open-Meteo, "Open-meteo," 2023.

[25] J. Tan, J. Yang, S. Wu, G. Chen, and J. Zhao, "A critical look at the current train/test split in machine learning," *CoRR*, vol. abs/2106.04525, 2021.

[26] P. Shiel and R. West, "Effects of building energy optimisation on the predictive accuracy of external temperature in forecasting models," *Journal of Building Engineering*, vol. 7, pp. 281–291, 2016.

[27] D. Kim, Y. Lee, K. Chin, P. J. Mago, H. Cho, and J. Zhang, "Implementation of a long short-term memory transfer learning (lstm-tl)-based data-driven model for building energy demand forecasting," *Sustainability*, vol. 15, p. 2340, Jan 2023.

[28] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *CoRR*, vol. abs/1611.06455, 2016.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations (ICLR 2015)*, pp. 1–14, Computational and Biological Learning Society, 2015.

[30] M. Feurer and F. Hutter, *Hyperparameter Optimization*, pp. 3–33. 05 2019.

[31] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

Appendices

Appendix: A

Graphs of Validation Loss and Training Loss vs. Number of Epochs of all DL models before Fine-tuning

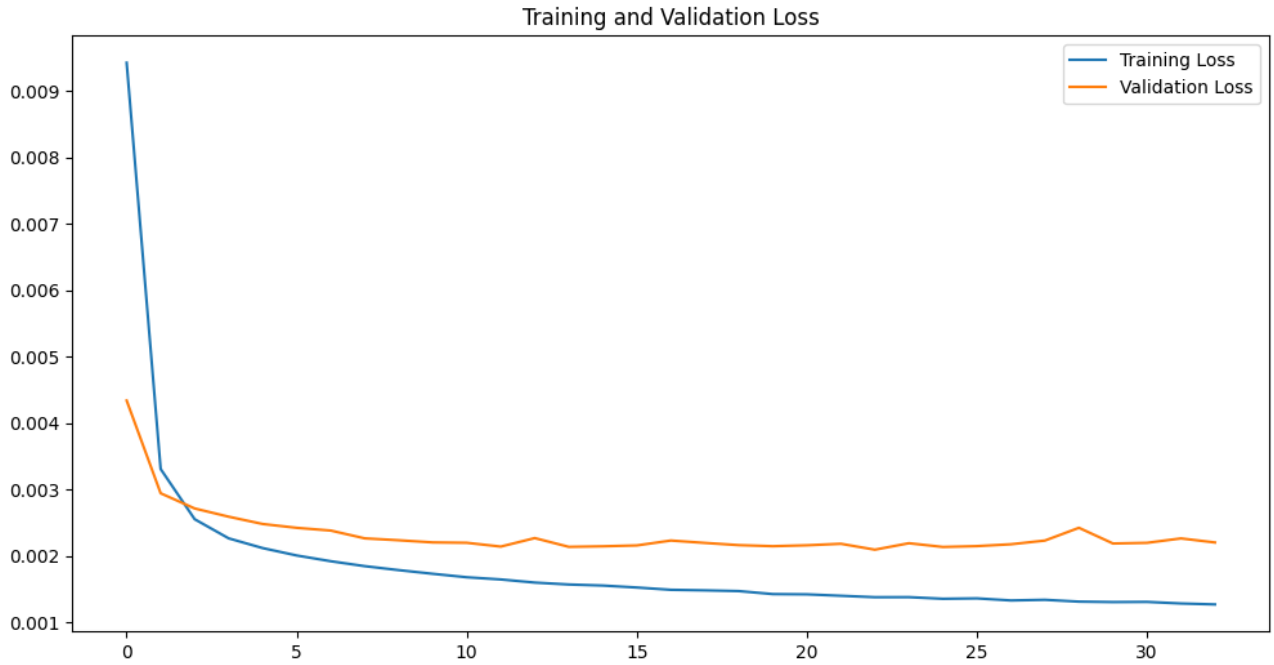

Figure 7: Plot of validation loss and training loss for TL-LSTM vs. number of epochs without weather added as a feature.

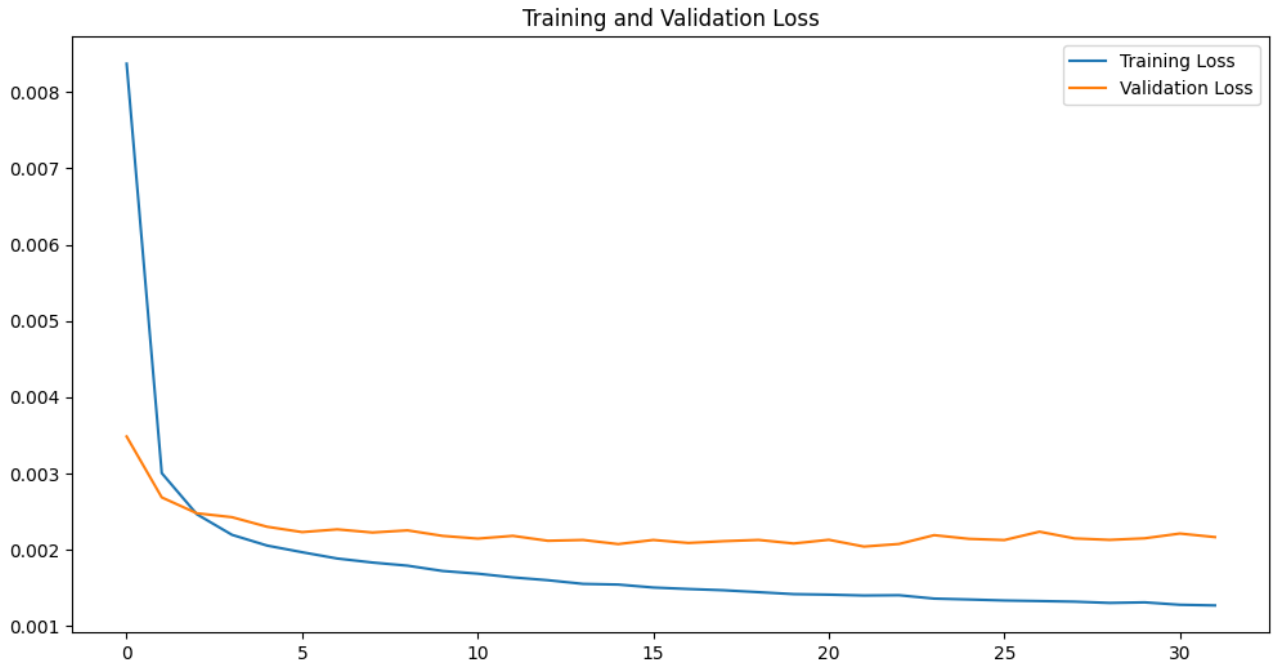

Figure 8: Plot of validation loss and training loss for TL-LSTM vs. number of epochs with weather added as a feature.

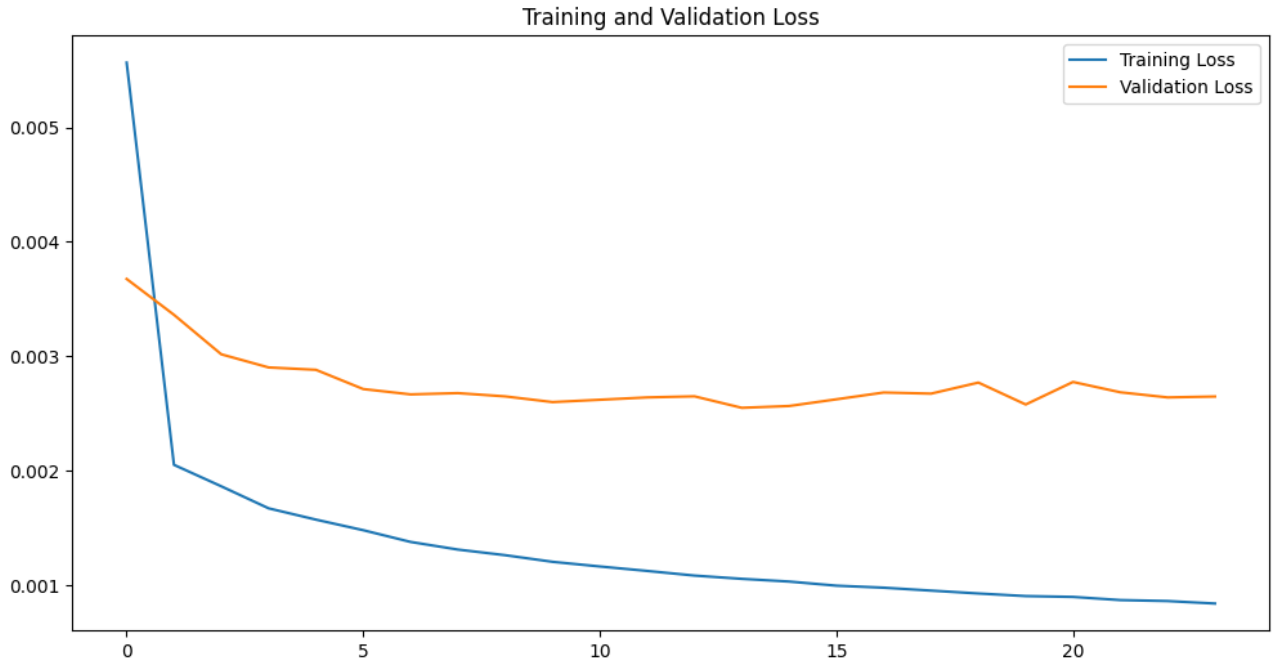

Figure 9: Plot of validation loss and training loss for TL-1DCNN vs. number of epochs without weather added as a feature.

Figure 10: Plot of validation loss and training loss for TL-1DCNN vs. number of epochs with weather added as a feature.

Figure 11: Plot of validation loss and training loss for TL-2DCNN vs. number of epochs without weather added as a feature.

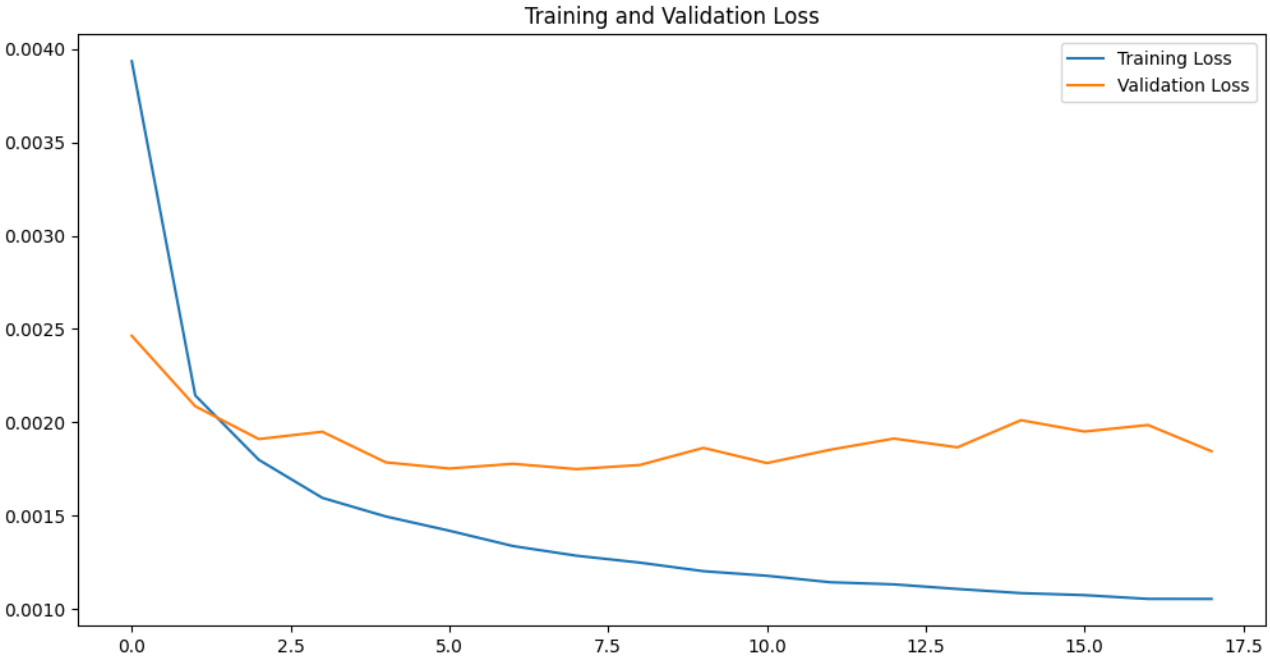

Figure 12: Plot of validation loss and training loss for TL-2DCNN vs. number of epochs with weather added as a feature.

Appendix: B

Graphs of Validation Loss and Training Loss vs. Number of Epochs for HT-TL-2DCNN-T

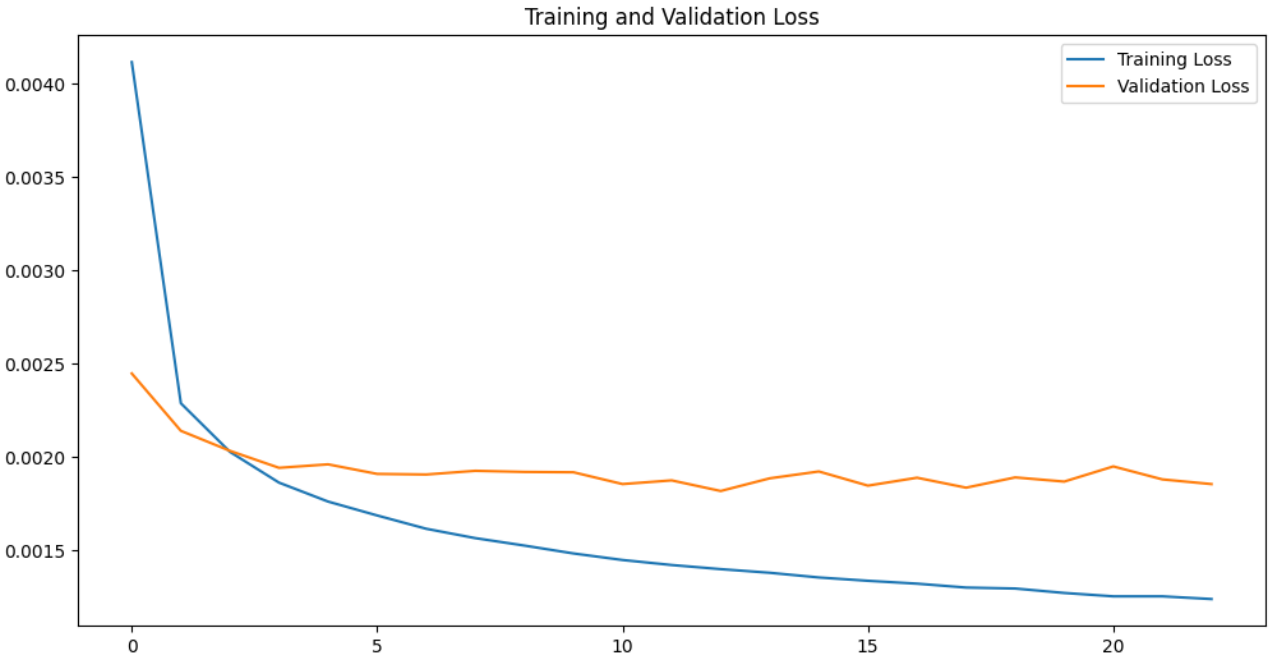

Figure 13: Plot of Validation Loss and Training Loss vs. Number of Epochs HT-TL-2DCNN-T