Aggregation of Energy Consumption Forecasts Across Spatial Levels

Using CNN-LSTM forecasts of lower spatial levels to forecast on higher spatial levels

Twan Borst[1]

Supervisor(s): Luciano Cavalcante Siebert[1]**, Sietze Kuilman**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

Abstract

Bottom up load forecasting, is a technique where energy consumption forecasts are made on lower spatial levels, after which the resulting forecasts are aggregated to form forecasts of higher spatial levels. With the current move to renewable energy sources and the importance of reducing the strain on an already congested electricity grid, accurately forecasting both the location and time of future energy consumption has become more important than ever. To this end, this paper analyses the impact of applying bottom up load forecasting to different spatial levels on the electricity grid, including appliance, household, community and city spatial levels. Energy consumption data for these spatial levels were gathered from the 15-minute Texas and California datasets from Pecan Street Dataport. The results obtained in this study, suggest that energy consumption volatility at lower spacial levels and forecast difficulty at higher spacial levels, play an important role in the performance of applying the bottom up load forecasting technique to energy consumption forecasting problems.

1 Introduction

Wrong estimates for (local) energy demand or supply, can result in an imbalance in supply and demand of electricity on the power grid. With a total energy consumption (EC) of 118 TWh, households in the Netherlands accounted for 23% of total national EC in 2021 [1]. On the supply side, due to the large scale movement to renewable energy sources, the energy supply has shifted from being a constant and adjustable source, to being a fluctuating source depending on solar and wind conditions [2]. Being able to more accurately forecast both the location and the time of 23% of the total national energy demand could play an important part in managing and reducing the strain on an already congested electricity grid.

Achieving this on a smaller scale, was done in Zheng et al. [3], in which they proposed monitoring EC on an appliance level to help further improve the accuracy of single household EC forecasts by making use of the finer data granularity appliance level data had to offer. Using a Persistence model, Long Short Term Memory (LSTM) model and Kalman filter model, Zheng et al. compared the difference in one day-ahead building EC forecast accuracy between aggregating forecasts of appliance EC and forecasting already aggregated appliance EC. In their research the authors found that both the LSTM model and the Kalman filter model gave more accurate forecasts by aggregating forecasts of appliance EC than forecasting total household EC directly.

More recent papers have also suggested improvements to many aspects of EC forecasting using models involving LSTM. A model combining a convolutional neural network (CNN) with LSTM and auto-encoders (AE) was proposed in Rick & Berton [4] to forecast multiple time series with unequal lengths using a single model. Somu et al. [5] proposed a kCNN-LSTM model for trend characterization, energy related feature identification and the modeling of temporal information in EC with which more efficient accurate energy forecasts could be made. Jin et al. [6] addressed the issues of data noise and high volatility in EC by combining signal spectrum analysis (SSA) with parallel long short term memory (PLSTM) to achieve greater forecasting accuracy on household EC. Jin also compared the proposed model to many other existing models used to forecast household EC among which many models also involving LSTM models.

While past papers have made significant steps in improving the accuracy of forecasting EC of single households, the question remains how these improvements will affect EC forecasts on higher spacial levels, like community or city levels. Predicting the EC of communities and cities by monitoring the EC of every device in every household could help in making more accurate forecasts for these higher spatial level. This paper proposes a CNN-LSTM model that will be combined with simple aggregation to answer the question of: *Can the forecast accuracy of community and city energy consumption data be improved by aggregating the EC forecasts made on lower levels compared to directly forecasting EC on community and city levels?*

The rest of the paper will be organized as follows: Section 2 will explain some of the concepts used in this work. Section 3 will describe the proposed model. Section 4 will be a case study of how the model was applied to a real dataset to gather results. Section 5 will give the achieved results from this case study. Section 6 will have a discussion on the model, results and other problems encountered during the research. Section 7 will reflect on the ethical aspects and reproducibility of this research. Finally, Section 8 will conclude the paper by summarizing the research and suggesting directions for further research.

2 Background

The model proposed in this work makes use of two types of neural network layers, namely, Convolutional Neural Network (CNN) layers and Long Short Term Memory (LSTM) layers. These two types of layers are explained in the sections below.

2.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) have been successfully applied to various problems, among which 2D image recognition, load forecasting, and Non Intrusive Load Monitoring [7, 8, 9]. A complete CNN consists of a combination of convolutional and pooling layers, and often terminated with layers of fully connected neural networks.

The convolutional layers inside CNNs are used to detect patterns with the use of filters, also called kernels. Filters are groups of trainable weights with a particular size, that slide over the data. The size of a filter is usually much smaller than the size of the data which it is applied to. Each time the filter moves, it is applied to a portion of the data. The weights of the filter are multiplied with values in the data and the aggregate is passed to the next layer as the feature value of that part of the data. By how much the filter slides after each calculation, is determined by its stride.

After the convolutional layer is applied, pooling layers are used to reduce the dimensionality of its output. An example of a commonly used pooling technique is Max pooling. With max pooling, windows of a particular size, also called its pooling size, slide over calculated features, similar to how filters slide over data in the convolutional layer. The output of the window, sliding over the data, is the value of the largest feature within the window.

2.2 Long Short Term Memory

Long Short Term Memory (LSTM) is an improvement over the Recurrent Neural Network (RNN) architecture and was first introduced in Hochreiter et al. [10]. It can retain long term patterns from sequential data and solves the vanishing gradient descent problem of RNNs. This is achieved using multiple memory cells, also called LSTM units, each containing an Input, Forget and Output gate in the form of sigmoid functions [11]. The input gate is responsible for determining whether new data will be stored in the current cell state. The Forget gate is used to gauge whether data from previous LSTM unit's cell states should be used in calculating the current cell state. At last, the output gate decides whether to pass the current cell state to the output of the LSTM unit.

3 Methodology

In this work, two variations of CNN-LSTM models were used in combination with the bottom up forecasting technique, to make forecasts for individual appliances, households, communities or cities. A general overview of both models is given in Section 3.1. The layers of both the CNN-LSTM model with and without hyperparameters are described in Sections 3.2 and 3.3 respectively.

3.1 Model Overview

Just as in the *k*CNN-LSTM model proposed in [5], the input of both models consists of historical electricity consumption and 7 extracted timestamp features: day of the year, season, month, day of the week, hour of the day, minute of the hour, work or weekend day. The output forecasts from the models are aggregated using a bottom up approach, which will be further explained in Section 4.2. The model itself was created using the TensorFlow python library[1]. The code used to create the model can be reviewed on the authors' public GitHub repository[2].

Using the electrical consumption and 7 timestamp features, the model creates a 12 hour forecast with a 15 minute temporal granularity based on the previous 3 days of historical electricity consumption. Temporal granularities used in related research were also considered, among which 1 minute and 60 minutes, which where already used in Zheng et al.[3] and Somu et al.[5] respectively. In the end the 15 minute granularity was chosen over smaller temporal granularities due to resulting in a dataset size and training time, fitting to the computational resources available.

[1]https://www.tensorflow.org/

[2]https://github.com/TwanBorst/aggregation-of-ec-forecasts-accross-spatial-levels

Although, hyperparameters are usually preferred, due to the available training time and computational resources, a CNN-LSTM model without hyperparameters was used for the largest experiments. See Section 5.1, for the results of applying this model to the complete dataset. To comprehend the impact the lack of hyperparameters has on the end results, a CNN-LSTM model with hyperparameters and dropout layers was also used on a smaller portion of the dataset. The results of this experiment can be found in Section 5.2.

3.2 Base Model Layers

Due to not making use of hyperparameters, the layout of the base CNN-LSTM model, visualized in Figure 1, is fixed. As shown as in Table 1, the model will consist of an input layer that takes in data with a shape of (64, 288, 8), where the dimensions correspond to the batch size, amount of 15 minute data points, and the number of features. It is then followed by a CNN layer, an LSTM layer, and two dense layers, after which it will return a result with the shape of (64, 48), corresponding to the batch size and the number of 15 minute forecasted data points.

The CNN layer consists of a 1 dimensional convolutional layer followed by a 1 dimensional max pooling layer with a pooling size of 2. The convolutional layer is made up of 64 filters, a kernel size of 2, a stride of 1, 0-padding that corresponds to the TensorFlow option ”same”, and a Relu activation function. These parameters performed well in the very similar *k*CNN-LSTM model from [5], which is why they were also chosen for this model. Other than in [5], where the amount of convolutional networks varied between one and three, our model was restricted to having one convolutional network in order to reduce training times.

Followed by the convolutional network, is an LSTM layer. The LSTM layer is built using 64 units and uses a tanh activation function. This is similar to the model proposed in Somu et al. [5], with the only difference that the authors in this paper made the amount of LSTM layers variable between 1 and 3 using a hyperparameter. Just like with the amount of convolutional networks, in order to reduce training times, the amount of LSTM layers used for the CNN-LSTM model was kept at one.

Lastly, the result from the LSTM layer is passed to the two dense layers of 32 and 48 neurons respectively. This is different from the one dense output layer approach used in [3, 8, 6], but similar to the two dense layers used in Somu et al. [5]. The output of the last dense layer is the output of the model,

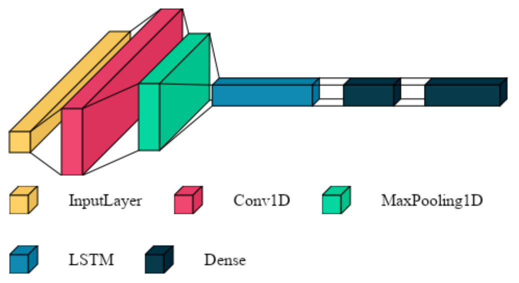

Figure 1: Base CNN-LSTM model

Layer	Parameter	Values
InputLayer	shape	(288, 8)
Conv1d & MaxPooling1d	layers	1
	filters	64
	kernel	2
	stride	1
	pool size	2
LSTM	layers	1
	units	64
Dense 1	units	32
Dense 2	units	48
Optimizer		Adam

Table 1: Model parameters for base CNN-LSTM model

which represents 12 hours of energy consumption data with a 15 minute temporal granularity.

3.3 Hyperparameter Model Layers

The CNN-LSTM model with hyperparameters' input and output is identical to that of the base CNN-LSTM model. The main differences between the models lies in the inclusion of hyperparameters, which allows this model to be more flexible with the composition of its layers. As visualized in Figure 2, another difference compared to the base CNN-LSTM model, is that this model contains a dropout layer after both the CNN and LSTM networks.

Table 2 lists the possible hyperparameter values available to the model. The optimization of these hyperparameters is done individually for each appliance, household, community or city using the Hyperband algorithm [12]. The Conv1d filters parameter is decides the amount of filters in the first Conv1d layer, consecutive Conv1d layers have half the amount of filters as the one before it.

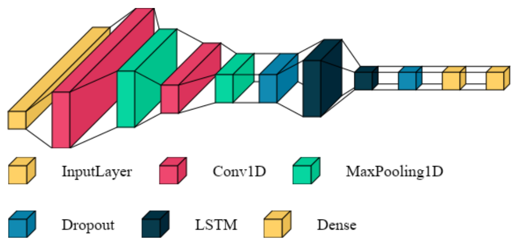

Figure 2: CNN-LSTM model with hyperparameters and dropout layers

Layer	Parameter	Values
InputLayer	shape	(288, 8)
Conv1d & MaxPooling1d	layers	interval=[1, 3], step=1
	filters	interval=[16, 128], step=2, sampling='log'
	kernel	2
	stride	1
	pool size	2
Dropout 1	rate	0.25
LSTM	layers	interval=[1, 3], step=1
	units	interval=[16, 128], step=2, sampling='log'
Dropout 2	rate	0.25
Dense 1	units	interval=[16, 64], step=2, sampling='log'
Dense 2	units	48
Optimizer		Adam

Table 2: Model parameters for extended CNN-LSTM model

4 Experimental Setup

4.1 Dataset

The two datasets that were used in this work came from the 15-minute residential energy data from Pecan Street Dataport[3]. They both contain 15-minute timestamped appliance level energy data, total grid energy data, and other non energy related information. The first dataset contains 1 year of data for 25 household located in the state of Texas starting from the 1st of January 2018. The second dataset also contains 1 year of data for 23 households located in the state of California with starting dates varying between 2014 and 2018. Before the two datasets can be merged and applied to the model, some preprocessing is required. To limit the training time, only data from June, July and August was used, irrespective of the year in which it was recorded. Households that did not have data within this period where excluded from the final dataset.

Electricity consumption data from different years can have different usage patterns, due to among others, differences in climate, available technology or energy prices. On a household scale the difference in years plays no role, since data aggregated to this level will all have happened at the same time. On a community or city scale on the other hand, aggregating data coming from different years is no longer guaranteed to be representative to the real world.

In the second preprocessing step, households where assigned to communities of 5 houses each. For every household, both a city and a state is given. In order to still be able to use the dataset, communities where added to the dataset. This was done by grouping the households by city and sorting them based on their unique *dataid*. Next households were grouped in groups of five, going from the lowest *dataid* to highest. Households that where left over and did not have a community assigned, where dropped from the dataset. A community size of 5 was chosen to have a similar amount of houses inside each community as communities within each city.

[3]https://dataport.pecanstreet.org

	Appliance	Household	Community	City
Mean EC	0.080	1.017	5.084	22.879
Mean EC SD	0.175	1.270	4.309	16.168
Entities in level	619	45	9	2

Table 3: Mean Energy Consumption and mean standard deviation of Energy Consumption per spatial level from the combined dataset of Pecan Street Dataport

The third preprocessing step entailed removing outliers and dealing with gaps in the data. Removing outliers was done using the ”Three sigma rule of thumb”, where any value that is more than 3 standard deviations away from the mean is removed. Filling in the remaining gaps is done by repeating the last seen value for at most 5 times. After this all values that remain empty are set to 0.

The fourth preprocessing step, involved aggregating the EC data from appliances to their corresponding households, communities and cities as visualized by Figure 3, after which they would all be stored separately from each other. Aggregating the data had to be done, since total EC data for communities or cities was not present in the original dataset. The total household grid EC data was available for some of the households, which differed from the aggregated appliances by at most 1kWh. In order to stay consistent however, aggregated appliance EC data was used as the ground truth for the household spatial level as well. Statistics for mean EC and mean standard deviation of EC is shown in Table 3 for each spatial level.

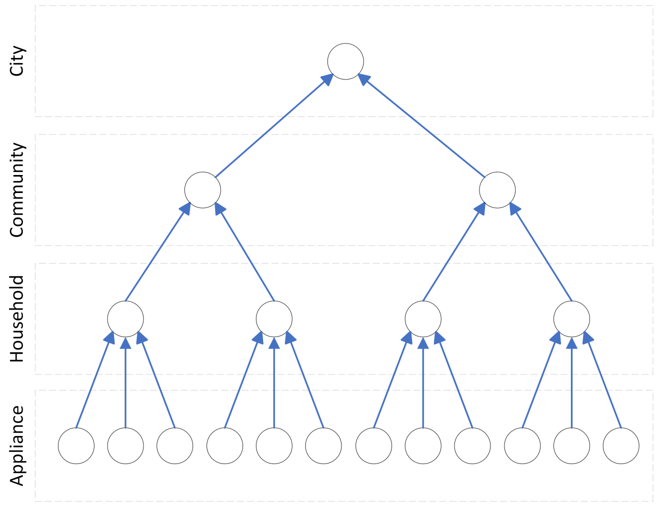

Figure 3: Visualization of aggregating and forecasting across spatial levels

Finally, the 7 timestamp features required by the model, as described in Section 3, where extracted from the original data, after which a copy of the data was normalized. This meant parsing the *timestamp* field for every EC data point of every appliance and replacing it with the equivalent 7 timestamp features. Once every data point contained the 8 features it required, a normalized copy of the data was made, where every feature column was scaled between 0 and 1 using min-max-normalization, which is described by Equation 1.

$$f(x_i) = \frac{x_i - min(x)}{max(x) - min(x)} \quad (1)$$

4.2 The aggregation network

This work proposes to make short term load forecast of the energy consumption of higher spatial levels, like communities or cities, using a bottom up approach, similar to what has been done in [3] on a household level and in [13] on a substation level. With this bottom up approach, energy forecasts are made on one of the lower spatial levels using a CNN-LSTM model, after which they are aggregated to a higher spatial level. This is similar to how the EC data was aggregated in the previous section and was visualized by Figure 3.

In order to determine whether forecasting from a particular spatial level is better or worse compared to forecasting from a higher spatial level, models were not only trained for appliances, but also for households, communities and cities. Forecasts from the models of each layer were then aggregated to all of the spatial levels above the already forecasted layer. As an example, forecasts from models made for appliances would be aggregated to their respective households, communities, and cities, while forecasts from household models where aggregated to communities and cities.

All aggregation steps in this work were done using simple addition, therefore no weights where used. Adding weights to the aggregation network could have resulted in more accurate results, however this would have made the implementation of the network a lot more complex, while also increasing the the time required to train and test the network including the models in it.

4.3 Training & Testing

With the model and the aggregation network complete, the last step is running the experiments. In all experiments, models are trained on the training set and evaluated on the test set. Evaluation on the test set happens both individually, as well as combined with the other models of its layer within the aggregation network, using the Mean Absolute Error (MAE) as the loss function. The used data is always split into a train and a test set, and depending on the model being used, an additional validation set is cut from the train set. The complete training data, including a possible validation set, has a train test data point ratio of 4:1 and no overlap between the windows of the train and test sets. Generating the model input and output was done using two sliding windows, both having a step size of 1. The windows used for model input contained 8 features for each of the 288 normalized input data points. The output windows where made out of 48 non normalized EC values. In total three experiments were done.

The first experiment made use of the base model. In this experiment models were trained and evaluated for every appliance, household, community, and city. All the models were trained for 25 epochs, with a batch size of 64. The batch size and number of epochs were chosen in such a way that the training and testing of appliance models was feasible within the timeframe of this research.

A second experiment, used the same base model as used in the first experiment, but this time trained for 100 epochs on all households, communities, and cities. The purpose of this experiment was to determine the impact of the amount of epochs used for training on the final results. Similar to the first experiment a batch size of 64 was used.

Finally, the third experiment compared the performance of the base CNN-LSTM model, with the performance of the hyperparameter version , on 3 households and their appliances. For this experiment the same amount of epochs was chosen as in the first experiment. The small sample size was chosen out of necessity, due to the amount of time required for hyperparameter tuning.

In total 779 models where trained and tested over all experiments. The computational environment used for training and testing consisted of three systems, of which the RAM and CPU specifications can be found in Appendix A Table 8.

5 Results

After creating, training, and testing the models, this section describes the achieved results. The tables in this section show the mean, as well as the standard deviation, for the MAE metrics recorded during the evaluation of the trained models on the combined dataset. Next to being evaluated individually, the trained models were also evaluated after every aggregation step to a higher spatial level in the aggregation network. The metrics on the diagonal of every table in this section originated from the individual evaluations of the models and serve as the baseline performance for the metrics above the diagonal, coming from the aggregation step evaluations. Both the a base model, without hyperparameters, and the upgraded model, with hyper parameters and dropout layers, were evaluated.

5.1 Base Model Results

As stated in Section 4.3, the base model was first trained for 25 epochs on every appliance, household, community, and city in their respective spatial levels. Lower spatial levels were then aggregated to the city spatial level and test results were gathered after each aggregation step. Table 4 shows the mean and standard deviation of the MAE obtained by the models.

Forecasted On	Aggregated To			
	Appliances	Households	Communities	Cities
Appliances	0.067	0.697	2.332	8.837
Households	-	0.692	2.090	6.917
Communities	-	-	2.009	$\boxed{5.734}$
Cities	-	-	-	6.411

(a) Mean of MAE

Forecasted On	Aggregated To			
	Appliances	Households	Communities	Cities
Appliances	0.038	0.284	0.769	2.740
Households	-	0.268	0.633	2.181
Communities	-	-	0.595	$\boxed{1.774}$
Cities	-	-	-	1.897

(b) Standard deviation of MAE

Table 4: Evaluation of base model, trained for 25 epochs on the appliance, household, community and city spatial levels

What can be observed from the tables above is that, overall, aggregating to one spatial level higher than the spatial level of the original forecasts, obtains a similar, all be it slightly worse, mean and standard deviation as forecasting on that higher spacial level directly. The only exception to this observation is aggregating community level forecasts to a city level, for which the metrics are highlighted with a box. Here an 11% reduction in mean MAE is observed compared to the baseline, while maintaining a similar standard deviation.

Going from one to two or more spacial levels above the original forecasts, model performance is worse in all cases with respect to the mean MAE compared to the baseline forecasts. Aggregating from an appliance to a city spacial level performs the worst, with a 38% increase in mean MAE compared to forecasting on a city level directly. Standard deviation also remains larger compared to directly forecasting EC.

As an additional verification, Table 5 shows the mean and standard deviation of the MAE metrics gathered while evaluating the same model trained for 100 epochs instead of 25. Due to the quadrupled training time, the model was only evaluated on the household, community and city spacial levels.

Forecasted On	Aggregated To		
	Households	Communities	Cities
Households	0.697	1.985	5.641
Communities	-	1.967	$\boxed{4.968}$
Cities	-	-	5.347

(a) Mean of MAE

Forecasted On	Aggregated To		
	Households	Communities	Cities
Households	0.276	0.582	1.555
Communities	-	0.570	$\boxed{1.465}$
Cities	-	-	1.573

(b) Standard deviation of MAE

Table 5: Evaluation of base model, trained for 100 epochs on the household, community and city spatial levels

The most notable difference between training the base model for both 25 as well 100 epochs can be seen in the last column of Table 5. Comparing the baseline performance between 25 and 100 epochs, the models trained on the household spacial level see no improvements regarding the mean or standard deviation of MAE. Community spacial level models also perform similar in mean MAE, but have a slight decrease in standard deviation. On a city spacial level however, both the directly forecasted baseline as well as the aggregated forecasts perform better. Aggregating the community level forecasts to a city level, using models trained for 100 epochs, both the mean as well as the standard deviation of the MAE metric were lower than the baseline performance metrics. Going from household to city spacial levels, the mean MAE is still slightly worse than the baseline performance, but standard deviation has become better, all be it by a very small margin.

5.2 Model Comparison Results

To verify whether the lack of hyperparameters played an important role in the results of Table 4, a new CNN-LSTM model including hyperparameters and dropout layers was trained for 25 epochs on 3 households and their appliances. Table 6 contains the evaluation results of the models trained with hyperparameters, while Table 7 contains the evaluation results of the same three houses, trained for the same amount of epochs, but without hyperparameters or dropout layers.

Forecasted On	Aggregated To	
	Appliances	Households
Appliances	0.094	1.106
Households	-	1.067

(a) Mean of MAE

Forecasted On	Aggregated To	
	Appliances	Households
Appliances	0.053	0.345
Households	-	0.340

(b) Standard deviation of MAE

Table 6: Evaluation of model with hyperparameters and dropout layers, trained for 25 epochs on households 661, 1642, and 2335, including their appliances

Forecasted On	Aggregated To	
	Appliances	Households
Appliances	0.094	1.106
Households	-	1.067

(a) Mean of MAE

Forecasted On	Aggregated To	
	Appliances	Households
Appliances	0.053	0.345
Households	-	0.340

(b) Standard deviation of MAE

Table 7: Evaluation of base model, trained for 25 epochs on households 661, 1642, and 2335, including their appliances

As can be seen from the tables above, the achieved results are identical to each other. Although these results are based on a small sample size, the results give us an indication that the addition of different hyperparameters does not improve the performance of the model when the bottom up strategy is applied to the appliance spatial level. Whether this is the same for higher spatial levels and whether the same results are found on a larger sample size, requires further research.

6 Discussion

6.1 Volatility of the data

The results, shown in Section 5, where not the same positive results as initially expected. The initial hypothesis was that applying the CNN-LSTM model with a bottom up approach, would lead to a more accurate forecast compared to directly forecasting the aggregated EC. A likely cause for the similar, though still slightly worse performance of aggregating forecasts from appliance and household levels to higher spatial levels, is the volatility of EC on lower spatial levels. Similar results were also encountered in Zheng et al. [3], where applying the same technique to the UK-dale dataset resulted in a 16% increase in forecast error.

Aggregating community EC forecasts to the city spatial level however, did result in more accurate forecasts. While the community level EC is more volatile than the city level EC, the city level EC is comprised of considerably more distinct EC patterns compared to community level EC. The authors hypothesis is that, the reduction in both the volatility on one side and the amount of distinct EC patterns on the other, led to community EC being a less complicated pattern to predict than city EC, thereby reducing the forecast error.

Reducing the volatility of the EC for the current dataset could be achieved by utilizing interpolation to fill in missing data. As discussed in Section 4.1, this work prolongs the last seen EC for a short time, after which the EC is set to zero. This approach leads to sharp, harder to predict, drops in EC, which would be solved by applying interpolation instead.

To be able to draw conclusions for whether or not to use the bottom up approach in order to improve EC forecasts on different spatial levels, further research is required. Both the use of a larger variety of datasets and the application of different models, more suitable for forecasting volatile data, could help in getting a better understanding of the possible benefits and drawback of the bottom up approach.

6.2 Real world application

Although, aggregating community level forecasts to a city spatial level, showed improvements for both mean and standard deviation of MAE, these improvements have not yet been seen on other spatial levels. In the current state, the proposed model combined with the aggregation network is only useful as a proof of concept. Thus far, the model has shown sub-optimal performance on a limited number of datasets. This, together with the increased amount time and computational power associated with monitoring and modeling each appliance individually, makes it hard to recommend this approach for real electricity grid application.

Furthermore, the assumptions that every electrical appliance or system is monitored and that the total EC is the same on every spatial level, do not carry over well to real world use cases. This simplification made it possible to design a less complex aggregation network, but is not an accurate reflection of the real world. Applied to the actual electricity grid, the aggregation network has to be able to deal with transmission losses and electricity consumers/sources that are not individually monitored, which is not possible with the current model and aggregation network.

Lastly, the performance of the aggregation network on different community sizes and more complex grid layouts is not yet known. For this work, a community size was chosen that provided a good middle ground between the amount of households inside a community and the amount of communities in the dataset. In the real world however, communities don't necessarily consist of the same number of households or grid layouts. How different grid layouts and community sizes will affect the proposed bottom up approach, requires further research.

7 Responsible Research

The bottom up approach, proposed in this work, can makes use of the energy consumption data of appliances inside a household, which is highly sensitive personal data to the members of that household. Following the responsibility principle as outlined in the Netherlands Code of Conduct[4], the privacy impact of this research should be minimized wherever possible. To that end, appliance models should be trained locally and neither the appliance model, the data used for training, nor non-aggregated forecasts generated by these models, should ever be able to leave the household itself. Additionally, to preserve the anonymity of participating households, household forecasts should be aggregated with data from other households as early as possible, after which the original forecasts should be forgotten.

Another principal from the Netherlands Code of Conduct is honesty. Although the found results have not been as positive as initially expected, with responsible research practices in mind, the authors still found it import to publish the achieved results as is. No test results or spatial levels were adjusted or left out due to resulting in negative results.

Finally, the authors of this work tried to work as transparent as possible. In an effort for this research to be easily reproducible all experiments were carried out as described in Section 4. Additionally, the code used for the experiments was made open source and published to GitHub[5].

8 Conclusions and Future Work

In this work, a comparison of the energy consumption forecasting performance of applying the bottom up strategy to different spatial levels was made. Considered in this work were appliance, household, community, and city spatial levels. A CNN-LSTM model was used to generate energy consumption forecasts of lower spatial levels, which where then aggregated to higher spatial levels by a forecast aggregation network. A combination of both the California and Texas 15-minute temporal granularity datasets from Pecan Street Dataport was used for this.

In total three experiments were done. In the first, a basic CNN-LSTM model without hyperparameters was applied to all spacial levels and trained for 25 epochs. The second experiment used the same model, trained for 100 epochs, but applied it only to the three highest spatial levels. Lastly, the third experiment applied a CNN-LSTM model with a large amount of hyperparameters and dropout layers, trained for 25 epochs, to 3 households and their appliances.

Results from these experiments showed that aggregating appliance and household energy consumption forecast led to a worse forecast accuracy compared to directly predicting the energy consumption of higher spatial levels. Aggregating community level forecasts to a city level did result in more accurate forecasts compared to directly forecasting the energy consumption of cities. The authors hypothesised that this re-

[4]https://www.nwo.nl/en/netherlands-code-conduct-research-integrity

[5]https://github.com/TwanBorst/aggregation-of-ec-forecasts-accross-spatial-levels

sult was caused by the reduced volatility of energy consumption at the community level, compared to lower levels, and the reduction of the amount of distinct energy consumption patterns in the signal, compared to higher spatial levels.

Further research is needed to determine whether this hypothesis also holds for other datasets, particularly datasets showing different degrees of volatility to the volatility displayed by the datasets used in this work. Alternatively, further research could focus on the effects of applying different models to the bottom up approach, that are better or worse than the CNN-LSTM model in forecasting volatile data.

References

[1] Eurostat. Final energy consumption by sector. https://ec.europa.eu/eurostat/databrowser/view/TEN00124/default/table?lang=en, 4 2023. Accessed: 2023-05-11.

[2] Katrin Schmietendorf, Joachim Peinke, and Oliver Kamps. The impact of turbulent renewable energy production on power grid stability and quality. *The European Physical Journal B*, 90:222, 11 2017.

[3] Zhuang Zheng, Hainan Chen, and Xiaowei Luo. A kalman filter-based bottom-up approach for household short-term load forecast. *Applied Energy*, 250:882–894, 9 2019.

[4] Rodney Rick and Lilian Berton. Energy forecasting model based on cnn-lstm-ae for many time series with unequal lengths. *Engineering Applications of Artificial Intelligence*, 113:104998, 8 2022.

[5] Nivethitha Somu, Gauthama Raman M R, and Krithi Ramamritham. A deep learning framework for building energy consumption forecast. *Renewable and Sustainable Energy Reviews*, 137:110591, 3 2021.

[6] Ning Jin, Fan Yang, Yuchang Mo, Yongkang Zeng, Xiaokang Zhou, Ke Yan, and Xiang Ma. Highly accurate energy consumption forecasting model based on parallel lstm neural networks. *Advanced Engineering Informatics*, 51:101442, 1 2022.

[7] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition, 2017.

[8] Musaed Alhussein, Khursheed Aurangzeb, and Syed Irtaza Haider. Hybrid cnn-lstm model for short-term individual household load forecasting. *IEEE Access*, 8:180544–180557, 2020.

[9] Inoussa Habou Laouali, Hamid Qassemi, Manal Marzouq, Antonio Ruano, Saad Dosse Bennani, and Hakim El Fadili. A survey on computational intelligence techniques for non intrusive load monitoring. *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, 12 2020.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997.

[11] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly Media, Inc., 1st edition, 2017.

[12] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18:1–52, 3 2016.

[13] Weicong Kong, Zhao Yang Dong, Youwei Jia, David J. Hill, Yan Xu, and Yuan Zhang. Short-term residential load forecasting based on lstm recurrent neural network. *IEEE Transactions on Smart Grid*, 10:841–851, 1 2019.

A Computational Environment

	Type	CPU	RAM
System 1	Desktop	AMD Ryzen 5 5600X	32GB
System 2	Laptop	Intel i7 12700H	32GB
System 3	Laptop	Intel i9 8950HK	32GB

Table 8: Hardware specifications for systems in computational environment