Household Power Consumption Forecasting

CMPT 496: Final Project

April 8, 2023

Max Pyshniak
Department of Computer Science
MacEwan University
Edmonton, Canada
pyshniakm@mymacewan.ca

Joel Lawrence
Department of Computer Science
MacEwan University
Edmonton, Canada
lawrencej43@mymacewan.ca

Dominic Dobosz
Department of Computer Science
MacEwan University
Edmonton, Canada
doboszd@mymacewan.ca

I. Introduction

Households worldwide are looking for ways to be more energy efficient with their power consumption, whether it is to be more sustainable and greener with energy usage or to save money by consuming less power. The idea came from a discussion with a group member's personal contact about one of their start-up companies: Green Gas Tank. Given their specifications, one section of their project required a smart energy storage system capable of adapting to the user's household. Such smart choices could consist of judging when to use stored power vs. city power, or knowing what sub-circuits to cut back on in the event of a power outage.

Having this system in place would benefit regular household owners who are looking to be more efficient with their power consumption, as well as those who are trying to be more self-sufficient. The overall system would be able to handle a wide range of circuits with multiple machines connected to them. It would then be able to forecast expected power consumption on each circuit, ideally with accurate predictions based off of the household's real history and readily accessible features such as weather. These forecasts would be balanced with the available stored power, such that in the event of a power outage the homeowner would be able to prioritize important sub meters.

For what we can accomplish in a capstone and for the sake of time management, we chose to focus primarily on making predictions on power consumption. Here begins the problem that needed to be solved for our project, how do we predict the power consumption of a user's house? To create a forecast for power consumption, a supervised regression learning system was selected to take on this challenge and gain a result that would be the initial steps for such an extensive project. Such predictions would first require a significant amount of historical data collected by the user. This historical data is crucial for the training and evaluation required for any machine learning model. Furthermore, electrical data is known for its extreme variability when recorded at frequent intervals, making it appear fairly unpredictable. We will then need to experiment with various models to determine which would provide the greatest accuracy in forecasting power consumption for a user.

II. Related Works

With the rising global concern for the environment and making smarter energy usage choices, the research in forecasting both residential and commercial power usage has increased greatly over the past couple of decades. This has lead to a great deal of work with statistical and machine learning models on historical power usage data, carried out by various researchers in the field.

One experimenter, Bourhnane [1] searched for a solution to the high demand for energy in Smart Grids, by deploying machine learning models for energy consumption scheduling and prediction. In their testing, they found that artificial neural networks were outperformed by a basic statistical model on their dataset. However, they state that an oversight in their study was using a data set that

was not large enough for training and validating their models. They also did not preprocess their data before training. Therefore, ANN models remain a viable option to investigate with large datasets that undergo proper preprocessing.

Chou [2] examined the performance of many AI methods including single, ensemble, and hybrid models to execute one-day ahead forecasts on a building's power consumption. His results showed that hybrid models had the most accurate predictions against test data with respect to usability and in planning and managing energy. Although less accurate, ensemble and single models also showed evidence of good forecasting techniques, while only requiring a more rudimentary knowledge in machine learning when compared to the hybrid models. He found that users who have a deep knowledge of AI techniques and require highly accurate and effective models should develop hybrid models by integrating machine learning techniques with optimization algorithms. Using 15-minute time intervals, they found the best single model to be an artificial neural network (ANN).

Expanding on the idea of experimenting with different time scales, Bonetto and Rossi [5] performed 2-hour ahead forecasting using the past 30 minutes of residential power data. The results of their machine learning techniques, including Support Vector Machines (SVM), Nonlinear Auto-Regressive Neural Networks (NAR), and Long Short Term Memory Neural Networks (LSTM), showed that each model had their own strengths. LSTM required twenty times more data to train for optimal performance. It had lower accuracy prediction than the others, but also had the smallest variance. Ultimately, NAR and SVM displayed the highest accuracy, opting them to consider a hybrid approach that uses NAR for short time intervals and SVM for longer intervals.

III. The Data

The original dataset was obtained from the University of California, Irvine (UCI) machine learning repository, and was originally provided by Georges Hebrail and Alice Berard [4]. The dataset contained time series data that included the monitored power measurements of a residential household located in Sceaux, France. The measured features included: active power, reactive power, voltage, intensity, and the active power of three sub meters. These values took the average measure at each minute from December 16, 2006 to November 30, 2010 for a total of nearly 4 years of time series data.

The values of this original dataset were then aggregated to average hourly data, with time steps removed from the beginning and end during the data cleaning process. The removal of beginning and ending time steps ensured that we started at 12AM on the first day and ended at 11PM on the last day, given that the original minutely values started and ended during midday.

Given the idea that power consumption tends to follow a trend with each season, weather data was added as a feature. This data was gathered from Visual Crossing [3] for Paris, France (7 kilometers from Sceaux) and included the features: temperature and "feels like" temperature, cloud coverage, dew, humidity, precipitation and precipitation probability, sea level pressure, visibility, and wind speed, gust, and direction. Not all of these features were expected to be good predictors of a household's power consumption however, the models would be able to extract which of these are important. Therefore, the removal of some seemingly irrelevant variables was considered but ultimately, all remained.

To ensure that the models were able to capture the circular nature of time, they were encoded using the angular distance of months days of the week, and hours. In other words, this allowed Saturday to be as closely related to Sunday, as it was to Friday, rather than being on opposite sides of the week. This also applied to December being as closely related to January as it was to November.

The total dataset with the Sceaux household's power consumption, weather features, and time in angular distance resulted in 17,016 rows (days) and 25 columns (features). The label and predicted value was chosen as the daily sum of active power.

IV. Methods

Considering the similar works that researched the same topic, Neural Network models were determined to be the best solution to the problem of forecasting power consumption, and for the scope of the project.

To perform predictions on future time steps, the collected and cleaned data needed to be transformed

into a supervised task. For this to occur, sections (or windows) of the data had to be taken as input, in order to predict the next time steps. For the chosen models, an input window of size 168 time steps (7 days) was used to forecast the next 24 time steps of active power. Each time step of the input window holds the features, including all the values for each sub meter, the temperature, the active power, etc. at that time. The output prediction of each model could then be compared with the actual active power values of the next 24 hours, which are called the labels. This window continues to slide across the data frame to produce a prediction for active power at each hour throughout the day. Each model's performance can be compared by observing the distance between the labels and the predicted values.

Fig.1 shows how the time series data was transformed into a supervised task for machine learning models.

Fig.2 shows an example forecast window, with the input, labels, and predictions. The blue line is the input of size 168 time steps (hours). The green dots represent the labels–the ground truth of actual observed active power–while the orange crosses represent the predicted time steps–the output of our models, which are to be compared with the labels.

A. Linear

A simple linear model was created to use as a baseline for comparison against the more complex models. This model is simply a linear equation based off of all features at one time-step to predict the next 24 time-steps.

B. Multi-Step Dense

The dense model was created as another simple model to compare with the more complex models. However, unlike the linear model, the full 168 hour window was taken as input to make a predict on the next 24 hours. This would allow the model to learn trends and patterns in the input.

C. CNN

A Convolutional Neural Network (CNN) is a deep learning model that excels at making predictions on data such as image classification, stock prices and time-series analysis. For time-series data, they learn to recognize patterns and trends within historical data, extracting the critical features through convolutional and pooling layers to make forecasts.

D. LSTM

Long Short Term Memory (LSTM) is a type of Recurrent Neural Network (RNN). This model was chosen to be tested as it is a classic model for processing sequential data. A LSTM cell consists of an input gate, output gate, and a forget gate, as well as its long-term memory component, the cell state. Fig.3 shows a complete diagram of a complete LSTM cell.

An LSTM cell processes each time-step sequentially to build up its state. For example, features from the first time-step will enter the LSTM cell through x_t. These features will be concatenated with the hidden state from the previous time step (h_{t-1}). As we are processing the first time step, the hidden state of the previous cell is zeros. This new vector ($z = [h_{t-1}, x_t]$) will pass through a few gates, both to update the hidden state, and the cell state (C_t). The forget gate uses a sigmoid function to output a value between 0 and 1 for each value of z, these values then update the cell state via point-wise multiplication - essentially causing the cell state to "forget" some information. The input gate computes the point-wise multiplication of both the sigmoid, and tanh functions of z. The sigmoid function again essentially determines which values will be updated, and the tanh function will produce candidate values. The result of this point-wise multiplication then adds the new information to the cell state via point-wise addition. This final cell state is passed along to be updated in the next time step, as well as used in the output gate to update the hidden state. The output gate computes the tanh of the cell state to produce a range of values between -1 and 1, which is then point-wise multiplied by the sigmoid of z which determines the magnitude of these values to pass through to the output hidden state. This hidden state is then passed through to be concatenated with the features of the next time step, and the process is repeated.

This architecture allows LSTM models to both keep-track of long term, and short-term trends. Given the erratic nature of the power consumption

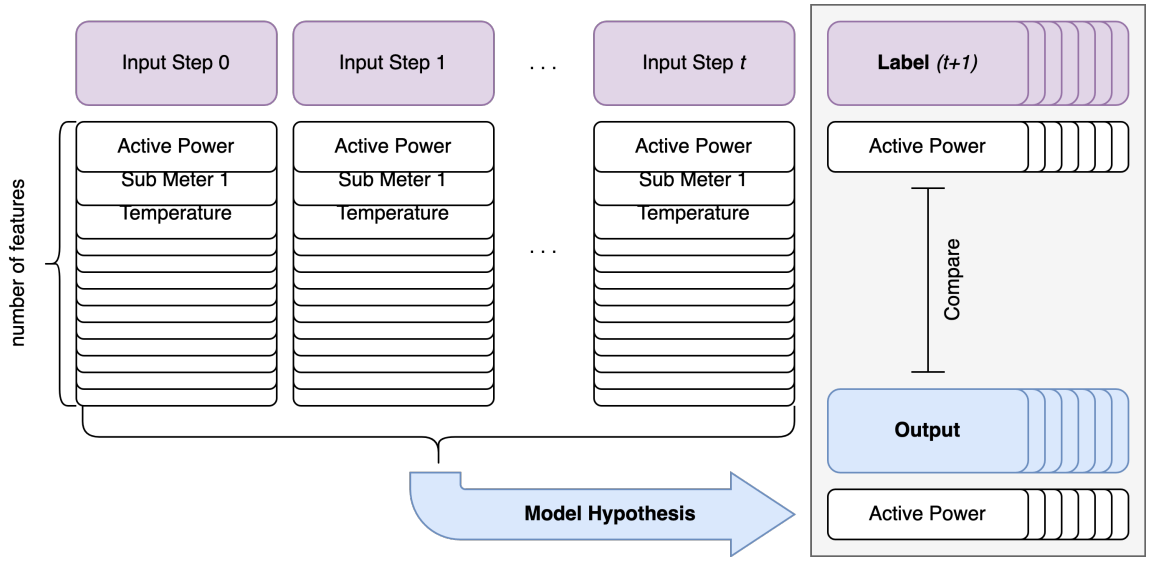

Fig. 1: Time series data to supervised data

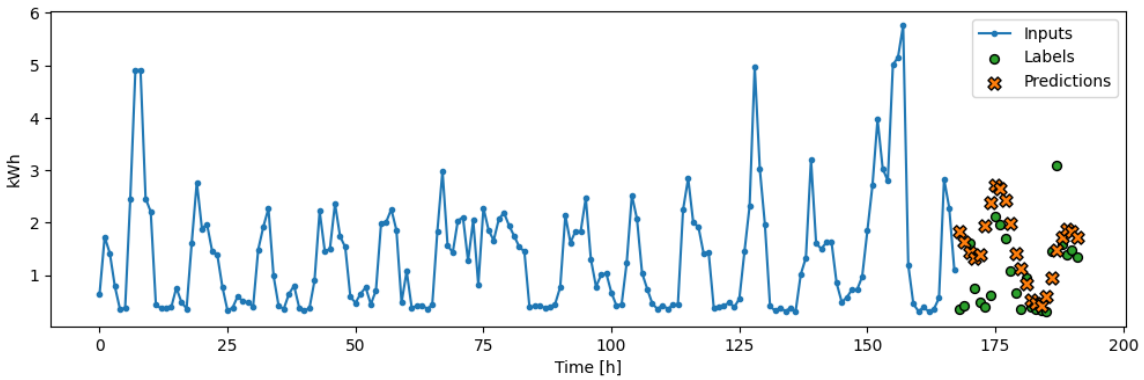

Fig. 2: Example forecast window

data, we hoped that this memory component would be able to capture and predict these spikes and downturns.

V. Experiments

A consecutive training, validation, and test split was used across the 47 months of data. This accounted for 1 full year of test data, 1 full year of validation data and the remaining data for training. The justification of this split comes from wanting to capture seasonality throughout a year in both our validation and test splits, as well as ensuring that models were trained on historical data instead of future data. Thus, roughly the first 23 months were reserved for training data, the next 12 months for validation data, and remaining 12 months for test data.

With the time-split of hourly total values, the number of training samples is not insignificant. Given our split, we have 16,825 samples for training, 8,569 samples for validation, and 8,569 samples for testing. Given that each of these samples predicts 24 time steps, that is 403,800 individual predictions for training, and 205,656 predictions for

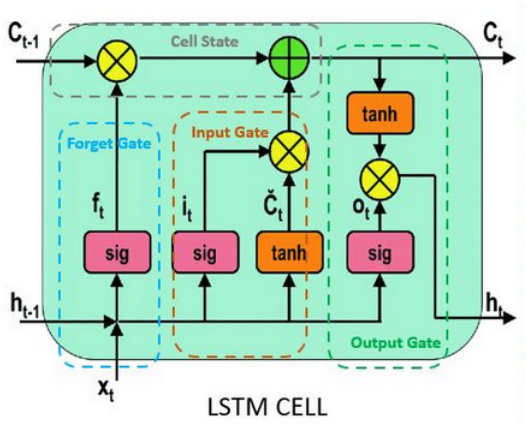

Fig. 3: LSTM Cell Diagram [6]

validation, as well as testing. Given the relatively large data sizes–for limited hardware performance– a few simple configurations of LSTM and CNN networks were tested with varying batch sizes, and a batch size of 64 was determined to have the best trade-off between model performance and speed.

Model tuning was largely completed using TensorFlow's Keras Tuner API, specifically using the Hyperband optimizer. Two major configurations were tested in the tuner for each of the models. First, custom metrics were created such that the sum of the 24 hours of predictions and labels would be optimized by the coefficient of determination. Secondly, the tuner was set to optimize for mean squared error between each individual hours prediction. For future reference, the former will be referred to as the summed models, and the latter will be referred to as the hourly models.

Appendix A. shows the hyperparameters tuned via the Keras Tuner. Note that the ranges used were determined based off of previous, more expansive tests, which took much too long to repeat.

A. Linear

Using TensorFlow, the linear model was created using a dense layer with a single node, connected to a dense layer with 24 nodes to act as the output layer. Predictions were made using the single last time step, to forecast the next 24 time steps. Note that Keras Tuner was not used on this model as it is to serve as a baseline, and that the linear summed results model will be the same as the hourly results model.

B. Multi-Step Dense

The TensorFlow implementation of the multi-step dense was created using a dense layer with a varying number of nodes, connected to a dense layer with 24 nodes to act as the output layer. Predictions were made using all 168 time steps for all features, flattened, to be the input.

C. CNN

The implementation of our CNN network consists of a TensorFlow Conv1D layer, a MaxPooling1D layer, and a dense layer with 24 nodes to act as the output layer.

D. LSTM

The implementation of our LSTM network consists of a single TensorFlow LSTM layer, connected to a dense layer with a 24 nodes to act as the output layer.

VI. Results

Three evaluation metrics were chosen to compare the models: the coefficient of determination (R^2), the mean absolute percentage error (MAPE), and the mean squared logarithmic error (MSLE). The R^2 score was chosen as it is a well-known measure of goodness of fit, with 1 representing a perfectly fitting model, and 0 (or negative) being a very poor fit. MAPE was chosen as it is an easily interpretable variable. The MSLE, on the other hand, is very hard to intuitively interpret. However, this metric was chosen as it penalizes underpredicting more than overpredicting, and given the use-case of estimating time left on stored battery power, this acts as a safety factor.

For each of these evaluation metrics, the summed offset and the hourly offsets of every model were observed to capture the behaviour of the models over different intervals. Each model forecasts each individual hour for every 24 hour period, starting from the first hour of the dataset to the last hour. Under the summed models, the optimization was such that the total sum of the 24 hour prediction for each sample was as close as possible to the total sum of the 24 hours of labels for that sample. Under the hourly models, the optimization was such that

each hour of the 24 hour prediction was as close as possible to each hour of their respective labels.

Summed Models

For the summed offset models, performance was evaluated at each starting index. For example, predictions were made for the total amount of power consumed from midnight to midnight, 1AM to 1AM, 2AM to 2AM, etc. for all of the 24 hour intervals in the dataset. Fig. 4 shows the results of the MSLE for all predictions of each model, starting at offsets as previously described.

Fig. 4: Summed Offset MSLE

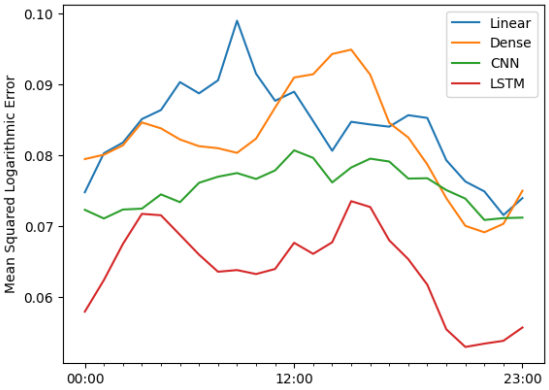

TABLE I: Model Performance Metrics (Summed)

Model	Measure	Validation	Test
Linear	R^2	0.38	0.30
	MSLE	0.078	0.084
	MAPE	0.250	0.261
Dense	R^2	0.43	0.33
	MSLE	0.075	0.082
	MAPE	0.228	0.240
CNN	R^2	0.47	**0.36***
	MSLE	0.063	0.075
	MAPE	0.227	**0.228**
LSTM	R^2	0.42	**0.36***
	MSLE	0.065	**0.064**
	MAPE	0.230	0.229

Bold value denotes the best performance.
* Denotes competing values with same result

Table I shows the performance results under validation and test data for each of the summed models.

Hourly Models

For the hourly offset models, performance was evaluated in two ways. First, for each individual predicted hour vs its label. And secondly, based off of the accuracy from x time-steps out from the last available feature time step.

Fig. 5: Hourly Offset MSLE

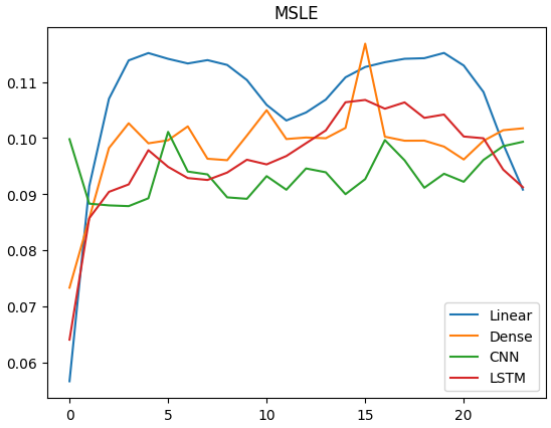

TABLE II: Model Performance Metrics (Hourly)

Model	Measure	Validation	Test
Linear	R^2	0.185	0.176
	MSLE	0.115	0.106
	MAPE	0.939	0.856
Dense	R^2	0.334	0.245
	MSLE	0.093	0.099
	MAPE	0.742	**0.739**
CNN	R^2	0.363	**0.266**
	MSLE	0.088	**0.093**
	MAPE	0.747	0.758
LSTM	R^2	0.318	0.252
	MSLE	0.095	0.096
	MAPE	0.790	0.769

Bold value denotes the best performance.

Table II shows the results of the hourly linear, dense, CNN, and LSTM models.

VII. Discussion

In this work, we presented the performance comparison of four neural network models and their ability to forecast power consumption framed in two

different ways: the upcoming summed 24 hour consumption, and the upcoming 24 hour consumption per hour. Three major metrics were inspected for each of the models and frames, the coefficient of determination, mean absolute percent error, and the mean squared logarithmic error.

Summed Models

Based on the results in Fig. 4 and Table I we notice that although none of the metrics showed drastic difference between models, there was a consistent preference towards the LSTM model. Most notable of the differences was the MSLE score. Given the behaviour of MSLE, the fact that the LSTM model showed the most conservative results, with respect to a safety factor for forecasting consumption, leads us to support the usage of LSTM over the simpler models, merely for safety.

Hourly Models

The results in Fig. 5 shows us that the CNN model performed the most consistently from hour-to-hour when compared to the other models. Table II shows that the CNN model also has the best overall performance metrics, aside from MAPE.

VIII. Conclusion

Given the intense variability of the active power data (having many upward spikes), the nature of energy consumption appears to be fairly unpredictable. Therefore, the slightest increase in metric performance is still somewhat relevant.

One big takeaway from this report was that it may be best not to over-complicate the problem with machine learning, and perhaps opt for a simpler approach. This is evident with how well the linear model performed against the more complex machine learning approaches, with the difference between each model's test metric.

The LSTM model had surprisingly subpar results, given its well-known usage for time-series forecasting problems. We had initially hoped that the "memory" of LSTM networks would better be able to catch and predict drastic upturns and downturns. However, the poor forecasting of the LSTM model may also be due to the framing of our problem. Perhaps, given a more granular breakdown (i.e. hourly data), the LSTM model may better capture this variability.

The network architectures tested in this report were very simple, i.e. a single LSTM (or CNN) layer, connected to a single dense node as the output layer. Given the clear complexity of the problem, it is likely worthwhile to still explore and test some more complex layer architectures.

The results of our hourly and summed models become especially interesting considering the best model differs based on the time interval chosen. On this basis, users may be inclined to use a hybrid approach where a CNN-LSTM Network is introduced. This would involve the components of both models, where a convolutional layer, pooling layer, and LSTM cell are integrated in one model.

Future Work

Given the nature of the "big-picture" project, the somewhat subpar results found in this report are not a great hindrance. The final working system will be more concerned with power usage throughout multiple sub-circuits within the house. For example, a household may have separate sub-circuits for heating and cooling, lighting, communications, entertainment, etc. and monitor each of these individually. With these various sub-circuits, the household system may be better able to forecast consumption on specific circuits. For example, heating and cooling are highly predictable by weather, and communications may be a very low and consistent power draw.

With the series of sub-circuits in place, as well as the existence of various battery stores, solar panels, and inverters for power distribution, the system will require both a load monitoring and database system. The database system will require the ability to be flexible to account for potential new hardware components. The load monitoring which goes into logging recordings, will also have a focus on monitoring the health of components and being able to report the health metrics to the homeowners. Determining these health metrics is a whole task in itself, for example determining the most efficient way to determine clipping in inverters, or to measure degradation of solar panel performance while accounting for easily repairable issues (such as simple dirty solar panels versus actually degraded panels.)

With this household monitoring database in place, different users with the same system will each have various different suppliers for their hardware, as well as live in different locations and climates. A major long-term goal of the Green Gas Tank is the ability to crowd-source performance metrics. For example, a manufacturer may report certain qualities and performance-over-time expectations; however, manufacturers can not always be trusted.

More in-line with the power-consumption forecasting of this report, the forecasting of solar power generation must also be investigated. There are many well designed physics models which do a fairly decent job of determining expected solar generation over time, however these models may not hold up to the very location-specific instances of cloud-coverage, or the constantly changing light-blockage of surrounding trees. To be able to accurately estimate the amount of power generated and stored, machine learning models should be inspected with real data.

Acknowledgement

The authors thank Dr. Dana Cobzas and Richard Meunier for their support, guidance, and patience throughout our capstone project.

REFERENCES

[1] Safae Bourhnane. Machine learning for energy consumption prediction and scheduling in smart buildings. *SN Applied Sciences*, 2(297), 2020.

[2] Jui-Sheng Chou. Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders. *Energy*, 165(B), 2018.

[3] Visual Crossing Corporation. Visual crossing weather. https://www.visualcrossing.com/.

[4] Alice Berard Georges Hebrail. Individual household electric power consumption data set, 2012. https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption#.

[5] Michele Rossi Riccardo Bonetto. Machine learning approaches to energy consumption forecasting in households. *arXiv e-prints*, 2017.

[6] Gaurav Singhai. Introduction to lstm units in rnn. https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn.

Appendix A
MODEL HYPERPARAMETERS

Model	Hyperparameter	Range	Optimal: summed	Optimal: hourly
Dense	Units	Start: 4, Stop: 128, Step: 4	72	36
	Kernel Regularizer	None, L1, L2, L1L2	L1	None
		Log sampling: 0.0001 to 0.1	0.0005	-
	Bias Regularizer	None, L1, L2, L1L2	L1	None
		Log sampling: 0.0001 to 0.1	0.0005	-
	Activity Regularizer	None, L1, L2, L1L2	None	None
		Log sampling: 0.0001 to 0.1	-	-
CNN	Filters	Start: 2, Stop: 12, Step: 1	4	10
	Kernel Size	Start: 2, Stop: 12, Step: 1	6	8
	Pool Size	Start: 2, Stop: 10, Step: 1	4	4
	Kernel Regularizer	None, L1, L2, L1L2	L1	None
		Log sampling: 0.0001 to 0.1	0.01	-
	Bias Regularizer	None, L1, L2, L1L2	L1	L2
		Log sampling: 0.0001 to 0.1	0.01	0.013
	Activity Regularizer	None, L1, L2, L1L2	None	None
		Log sampling: 0.0001 to 0.1	-	-
LSTM	Units	Start: 4, Stop: 64, Step: 4	30	28
	Recurrent Dropout	Start: 0.00, Stop: 0.50, Step: 0.05	0.3	0.1
	Recurrent Regularizer	None, L1, L2, L1L2	L1L2	L2
		Log sampling: 0.0001 to 0.1	0.0027	0.007
	Kernel Regularizer	None, L1, L2, L1L2	L2	None
		Log sampling: 0.0001 to 0.1	0.0027	-
	Bias Regularizer	None, L1, L2, L1L2	None	None
		Log sampling: 0.0001 to 0.1	-	-
	Activity Regularizer	None, L1, L2, L1L2	L1	None
		Log sampling: 0.0001 to 0.1	0.0027	-