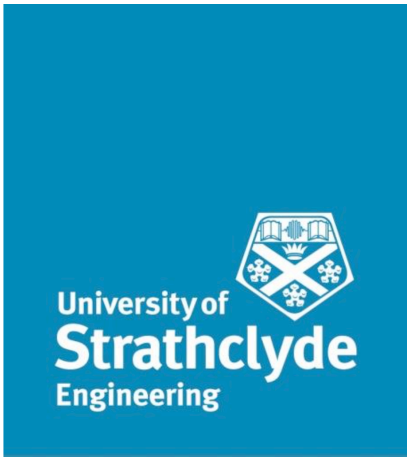


Department of Mechanical and Aerospace Engineering

# Short Term Energy Forecasting Using Neural Networks

Author: Paulo Ban Gurgel Nogueira

Supervisor: Dr. Paul Tuohy

A thesis submitted in partial fulfilment for the requirement of the degree

Master of Science

Sustainable Engineering: Renewable Energy Systems and the Environment

2016

## Copyright Declaration



Signed: Paulo Ban Gurgel Nogueira Date: 31st August 2016

## Abstract

With the increasing need of incorporating intermittent renewable energy sources and other alternative supply/demand management strategies into the energy grid, a need to forecast future demand arises. Knowing, or at least estimating, future consumption, even on a short-term basis can greatly help shift loads or optimize stochastic power sources and stored energy.

This paper focuses on investigating neural network strategies and tests it in three different grid levels, Great Britain's macrogrid, Findhorn's community in Scotland and a single house also in Findhorn. Different input selections along with different database sizes were attempted.

It was concluded that on the macrogrid level, with little effort very good results were generated. The community level can reach a 10% error margin quite reliably with at least one month worth of data. The single house level did not perform well, since single human families behaviour can be too chaotic to produce usable results. Still, on the community level, a not too complex and very adaptable neural network algorithm can be implemented to aid a microgrid better manage its energy needs with a workable level of uncertainty.

## Acknowledgements

I would like to thank my tutor, Paul Tuohy and the University of Strathclyde for leading me into a search for knowledge into the sustainable engineering field. My family for giving me financial, moral and project revision support, which is more important is left to reader's discretion. My friend Paulo Abelha Ferreira, who is doing his PhD in artificial intelligence, for the aid in doing this project.

More importantly, for all of those who came before, will come after and currently share the goal of focusing their energy into create a more sustainable society.

# Table of Contents

## List of figures

## List of tables

## List of tests

## Abbreviations and Acronyms

| | |
|---|---|
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| b | bit |
| B | Byte |
| CPU | Central Processing Unit |
| DSM | Demand Side Management |
| FH | Findhorn |
| G.B. | Gran Britain |
| GB | Gran Britain |
| GPU | Graphics Processing Unit |
| kB | Kilo Byte |
| MAE | Mean Absolute Error |
| MAPE | Mean Absolute Percentage Error |
| MB | Mega Byte |
| MSE | Mean Squared Error |
| NN | Neural Network |
| R | Correlation Coefficient |
| UI | User Interface |
| XOR | Exclusive Or (Logical Operation) |

# 1 Introduction

## 1.1 Motivation

Just staying in the most concrete and direct uses, energy is needed to produce goods, services and provides productive capacity, or jobs, for the population. This allows some experts say that energy consumption is a proxy for economic growth. Countries fiercely aim to improve their status in the world. With that, a lot must be invested in generation capacity. Curiously, ways to diminish the need for said capacity often does not receive the attention it deserves. This often results in energy cost increases. At the same time the capacity market often is fossil fuel based.

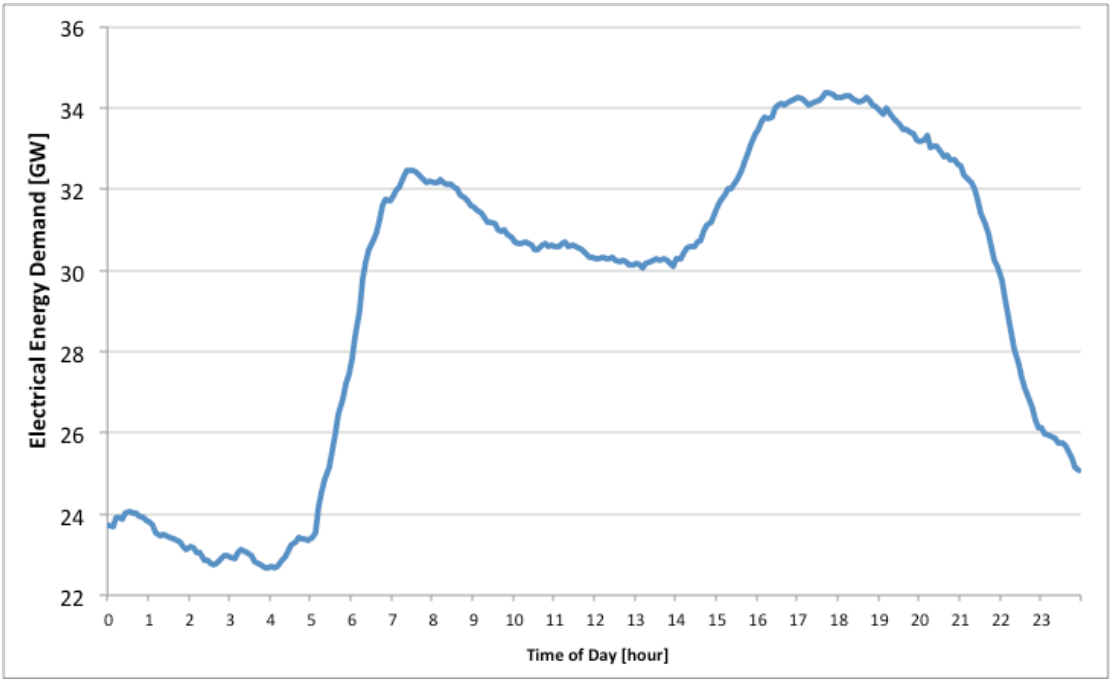


*Figure 1.1: Electrical energy demand for the 6th of June 2016.*

*Data retrieved from G.B. National Grid Status.*

As seen on Figure 1.1, the curve for one random day (6$^{th}$ of June 2016) the overall UK grid demand can vary by nearly 50% in 2 hours (from 5h to 7h). All this

capacity isn't free, there must be power stations apt to supply more than 10 GW for specific times of one single day and do that within the aforementioned 2-hour period.

That single day of June consumes around 705 GWh of energy. Ideally, that would require roughly a steady 30 GW throughout the day. This would lower the necessary installed capacity by more than 4 GW not to mention the costs associated with modulating the power output.

Therefore, instead of solely trying to supply more power, demand-side management must come into focus. Load management is a valuable tool to smoothen the demand curve and ease unnecessary financial impacts of unbalanced energy supply/demand relationships through time.

Since large companies have equally large amounts of operational inertia, the smaller more varied residential consumer appears to be a more varied and possibly adaptable market. Strategies for the residential user are especially interesting if the country incentivises demand behaviour with dynamic energy pricing.

"*Aspirations of grid independence could be achieved by residential power systems connected only to small highly variable loads if overall demand on the network can be accurately anticipated. Absence of the diversity found on networks with larger load cohorts or consistent industrial customers makes such overall load profiles difficult to anticipate on even a short term basis.*" (Stephen, et al. 2015)

In the same vein, "knowing" the future demand is very important for renewable microgeneration integration. Prompting the quote "*knowledge is power*" to become "*knowledge greatly aids power (systems)*".

## 1.2 Project Objectives

Any attempt to create optimized energy systems will pass through the issue of predicting future behaviour. Processing advancements and technical innovations

sprouted and computational intelligence became a more available tool for a wide variety of problems.

The primary aim in this paper is to investigate one aspect of supply and demand matching that is forecasting electrical energy demand. This will be done on different grid levels, and assess ways to optimize it through machine learning using neural networks.

Since most of the knowledge regarding the use of neural networks is mostly empirical in nature this paper will explore different neural network concepts and compare them in order to find the best results with limited data availability.

With reliable forecasting software an energy management control system can potentially incorporate intermittent renewable energy sources and other alternative supply/demand management strategies into the energy grid. Knowing, or at least estimating, future consumption, even on a short-term basis can greatly help shift loads or optimize stochastic power sources and stored energy. Microgrids would be most potentially benefited from this forecasting and control scheme.

## 1.3 Methodology

First, a **research** was conducted to find how much was done within the specified field. Additionally, tools and strategies were collected to inspire possible solutions later to be used. A short review of relevant information is shown on the Literature Review section.

In order to evaluate and optimize a neural network for short-term energy forecast *MATLAB's Neural Network Toolbox* was identified as a good tool for the coming tests. Also, a **structure** was formulated for said tests. This information can be found on the Tools and Test Structure section.

MATLAB was used to perform a series of **tests** with varying parameters and goals. They inform on their efficacy and how to use a machine-learning tool in order to extract the best result possible with limited information.

The tests were performed on three different grid levels:

- **Macrogrid level** with Gran Britain's demand historic data on Application at Macrogrid Level section.
- **Community level** with Findhorn's demand from ORING project data on Application at Community Level section.
- **House level** with two houses from Findhorn's community from ORIGN project data on Application at House Level section.

An array of different error and efficacy measures (description on section 3.2) will be produced and then **evaluated** on the Discussion section.

Lastly, **conclusions** will be made and presented on the Conclusion section.

## 1.4 Scope

The overall goal of this project is to evaluate different neural network strategies for short-term forecasting. Within this context, short-term shall mean a 24-hour lead, or one full day ahead.

Data availability is one of the great issues of any forecasting endeavours. For the writing of this paper limited information was available for this sort of testing. Part of the problem was to deal with the limited amount of data available and use it to create a parallel with real world applications. This issue depends greatly on the grid level being observed, since higher levels (macrogrid) tend to receive more attention than lower ones (communities and housings).

This paper will limit itself on a few parameters in order to make a more concise and adapt to the time scope. Therefore, unless otherwise specified, to following constraints were determined:

- The activation function of the hidden layers will be the tan-sigmoid. Other functions are available, but they will not be evaluated.

- The topology being observed will be a combination of single and two layered neural networks ranging from 5 to 10 neuron on the first layer and 0 to 10 neurons on the second layer.
- The training algorithm will be Levenberg-Marquardt and no exploration was conducted on different ones.

Lastly, to mimic a real application testing will be done with data separate from the training set and always use the latest part of the data set. This means, for example, that if only January, February and March of the same year are available, both January and February will be used for training and March will be left solely for testing.

# 2 Literature Review

## 2.1 Neural Networks

Artificial Neural Networks or just Neural Networks is a machine-learning model inspired on the brain structure of animals. It is a system that is designed to recognize and approximate the pattern of specific problems on its own when presented a sample pool with both inputs and output of previously measured data or historic database in order to serve as a prediction tool. In essence it is a adaptive problem solver. (MacKay 2005)

As seen on the example presented in Figure 2.1 a Neural Network is composed of "neurons" (or nodes) and connections. On that picture there are 3 input (red), 4 hidden (blue) and 2 output nodes (green). The topology used on that case is of one with a single hidden layer. More could be added to add a greater capacity to simulate more complex relationships while also slowing down the learning process of the neural network.

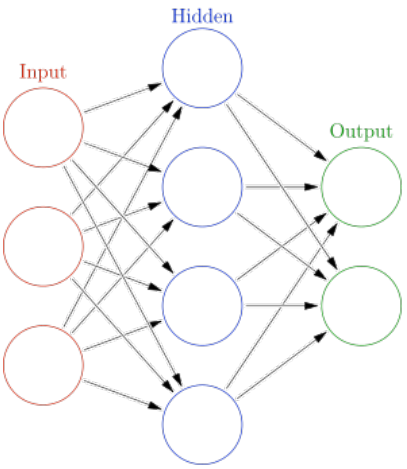


*Figure 2.1: Example Neural Network.*

*(image from Wikipedia)*

During the process of training the network compares the results of the samples give to it and update the weights of the connections between each neuron in order to follow more closely the behaviour presented to it.

Connections are formed between each neuron of adjacent layers. Figure 2.1 depicts this, as each single input is connected to each hidden layer node and each hidden layer node is connected to each output node. The input layer does not communicate directly with the output layer.

As shown on Figure 2.2, each neuron receives each previous layer's values multiplied by their individual weights, adds them all up along with a bias and goes through an activation/transfer function that will be feed to another neuron or the output.

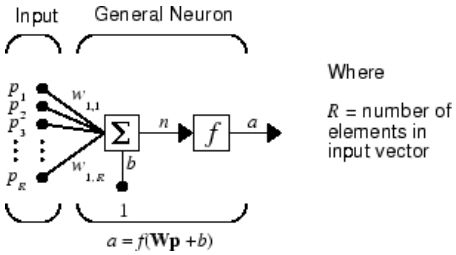


*Figure 2.2: Single neuron internal diagram.*

*(from MathWorks site)*

The bias works the same way as the b coefficient found in a linear function ($y=ax+b$) in the sense that it is responsible to shift the result of the function in order to better fit the data prediction and not be stuck cutting the origin (0,0).

The role of the activation funtion is to take all the inputs and attempt to generalize their combination into a probabilistic function to extract how much they fit into a specific category, in short terms the neuron is a classifier. The neural network does this throughout a couple of layers with a number of neurons in each. They are limited between -1 to 1 or 0 to 1. The sigmoid functions are a very common type of activation function. Although not limited to depending on the problem, they are frequently used within the hidden layers of the neural network.

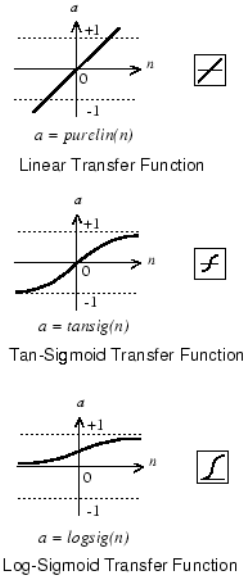


*Figure 2.3: Activation/transfer functions examples.*

Alternatively, problems that require a real output value must eventually go trough a transfer function that is not limited to -1 and 1, therefore a transfer function like the linear one can be used to make a linear combination of all weighted inputs.

The topology and parameters used on a neural network can vary greatly and there are few predefined right and wrong answers. Usually, each problem must go trough a series of tests to determine how to best approximate the desired targets.

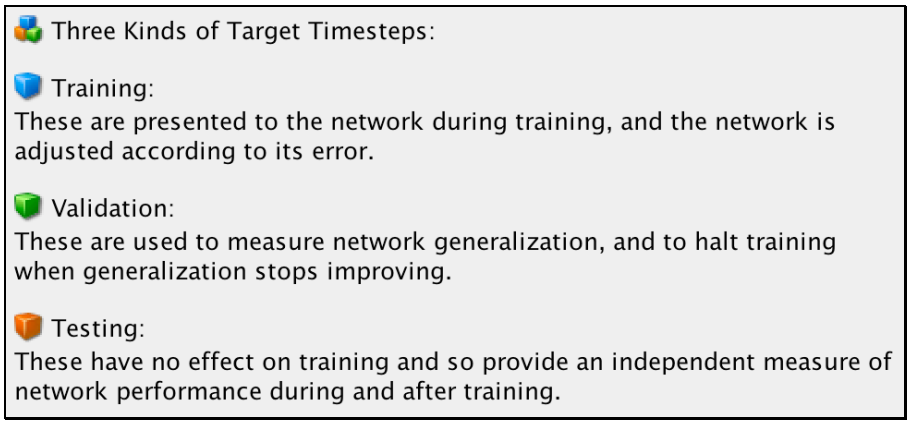


*Figure 2.4: Three kinds of data samples.*

*(from MATLAB's Neural Networks Toolbox app)*

Figure 2.4 shows MATLAB's Neural Networks Toolbox app's explanation of each different type of data samples created from the data pool offered for the training of the NN.

The kind that will take the biggest part of the data pool is the **Training** type. This will be the one used to adjust the weight values of each of the connections. **Validation** will be used as a separate pool to help the model decide when to stop training, or when to stop presenting the NN with the training data samples. Lastly, the **Testing** data pool is used to have a separate unbiased measurement of the NN's performance.

Neural networks depend quite a lot on how the model is designed, especially how the inputs are specified. There are many ways to present different data sets. Usually, experimentation is the usual route as each problem has it's own particularities. Therefore, a lot of work is necessary to prepare the data to better prepare the NN.

The order and how many times each sample is presented to the NN also greatly influences the outcome. For example, if too many training cycles are executed the issue of over fitting can arise. If this happens the network becomes very accurate with the training data in detriment of samples outside the training pool. In essence, this is analogous to using a regression with an unnecessarily high order, the result might be good but the general sensibility of the model is diminished. The validation type of data sample is especially important to avoid this issue.

## 2.2 Demand Prediction Using Neural Networks

Using a case study on a relatively small commercial building in Rome (Italy), the paper *Energy consumption control automation using Artificial Neural Networks and adaptive algorithms Proposal of a new methodology and case study* (Benedetti, et al. 2015) attempts to use neural networks as a demand forecasting tool.

Using feedforward perceptron neural networks three different topologies are attempted. One has a single hidden layer, which the author informs that the optimal number of neurons was identified where the error is minimized. On the second model the first layer has the same number of neurons than the inputs and the second hidden layer with the number of neurons than the inputs plus one (2 n + 1). The last model has four times the number of inputs plus two (4 n + 2) on the first layer and twice the number of inputs plus one (2 n + 1) on the second. These numbers were defined by another paper about perceptron neural networks approximations (Ismailov 2014).

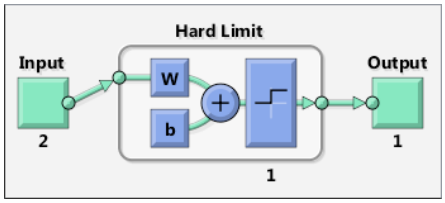


*Figure 2.5: A perceptron neuron (from MATLAB)*

Before selecting the inputs, a correlation (R) analysis is conducted between possible input candidates to identify possible linear relationships and simplify the model. Considering the different limitations on the case being studied the final variables identified as good input sources identified for this specific scenario were:

1. Hour of the day.
2. External temperature (°C).
3. Illuminance (lux).
4. Relative humidity (%).
5. Number of people inside the building.

In particular the building population was estimated from the energy consumption from the personal computers. Naturally every situation will allow for better or worse inputs, this is no different. Additionally it is not explicitly informed how the humidity or illuminance are added to the system, if the values used is the one before the demand is forecast or if the input is forecast first. The building seems to be equipped with sensors that measure these variables. Therefore the question stands.

Other weather conditions considered were:

- Absolute humidity (%).
- Visibility (km).
- Windiness (m/s).

Data is divided into training and testing sets for simulating how the system would actually be used. Performance indexes used were MSE, MAPE and $R^2$. It is unknown how many attempts were made for any of the different formulations, a unsuspecting reader might understand the results as final and not just one iteration of a neural network formulation.

Two different retraining strategies are attempted, one named Mobile and the other Growing training, the first uses the most recent data and second considers all the data collected so far. Both returned similar result, although the tests were not extensive. A minimum $R^2$ value is determined to trigger a new training of the network.

*Uncertainty handling using neural network-based prediction intervals for electrical load forecasting* (Quan, Srinivasan and Khosravi 2014) presents an interesting idea of instead of traying to find one specific result for each time step a upper and lower range is produced instead. This way a band is introduced and therefore the incertainty prevalent within any type of forecasting is more plainly accounted for.

Additionally, a heat map with the results of different network topologies (how many neurons on each layer) is introduced to show what is the optimal structure. But the paper is non-transparent with how those test were run, or how many attempts were made. There are issues of sheer randomness that greatly hinder the certainty of any sort of optimal parameter selection within neural networks.

The results table of the paper sugests that a low amount of five tests, all of which are very similar to each other.

On *Incorporating Practice Theory in Sub-Profile Models for Short Term Aggregated Residential Load Forecasting* (Stephen, et al. 2015) a few forecasting techniques are compared against one another on a community level case study. In this paper the best results are achieved by combining a few different techniques such as: Presisten Forecast, Flat Forecast, ARIMA, Neural Network and Gaussian Load Profile.

Daily profiles were identified and labeled to aid the prediciton power of their algorithms. No other data beyond time and demand were available, therefore enviromental data was absent as an input. The inputs were, for the more elaborate forecasting algorithms, the past demand for every 30 minutes and the day of the week.

The paper also gives some expected reference values for MAPE at different grid levels, 13.8%, at village level, 5.15% at university campus and 1.97% at national level. Additionally, the paper *The Real-Time Optimisation of DNO Owned Storage Devices on the LV Network for Peak Reduction* (Rowe, et al. 2014) sugests that an aggregation of houses can be forecast with an error ranging between 10% to 35%. These number are far from set in stone, but they serve as a comparison parameter.

## 2.3 ORIGIN Project

The Orchestration of Renewable Integrated Generation in Neighbourhoods (or ORIGIN) is a completed project that implemented a renewable energy integration system as shown in Figure 2.6.

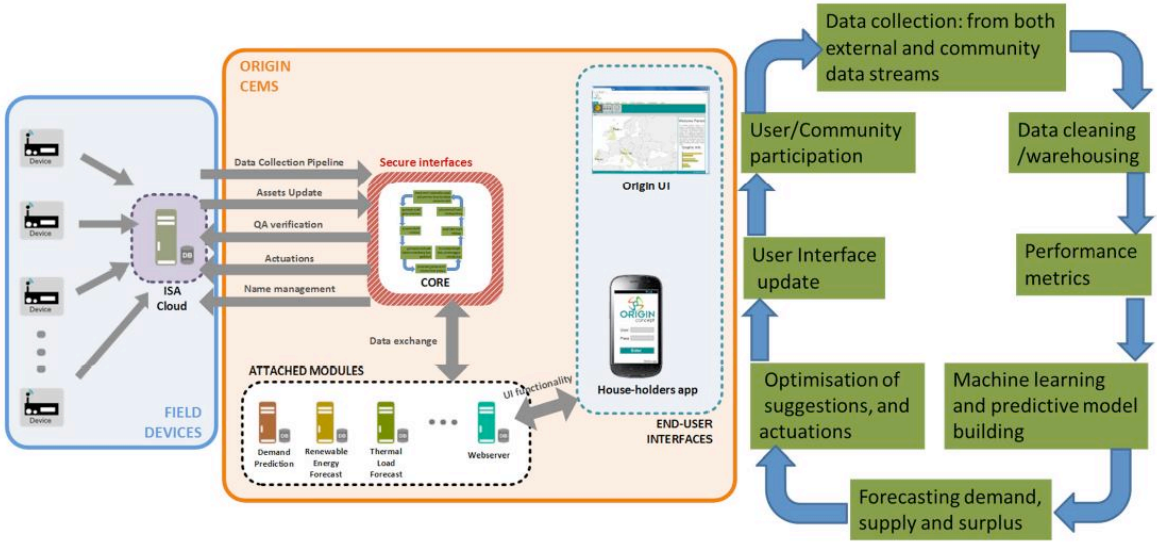


*Figure 2.6: ORIGIN control architecture and Operation.*

*From ORIGIN Final Report*

It includes local energy demand and supply and informs the user through an UI all the data required to better take advantage of the current situation. It helped consumers to shift their demands in accordance with the current/forecast renewable energy availability. Which is the overarching goal of ORIGIN, to maximize the use of renewable energy sources shifting the use of loads to times of day were supply is more abundant. This way less energy is imported, diminishing on and off site emissions along with energy costs.

Although ORIGIN is dependant on technological solutions it also heavily relies on the social side. The users gained the tools to better adjust their behaviour while also being taught how to use it.

The eco-communities benefited by the system were Damanhur in Italy, Findhorn in Scotland and Tamera in Portugal. The dataset generated by Findhorn was used in this paper to explore forecasting solutions with neural networks.

# 3 Tools and Test Structure

This section describes the tools used and how the tests worked.

## 3.1 Software

The software used in this paper is MATLAB with the Neural Network Toolbox. This software and toolbox are very flexible while still keeping a very simple UI, in a way that usual worries and pitfalls with “regular” programming are avoided in favour of focusing on the actual problem at hand.

Additionally, the Parallel Computing Toolbox is also used to employ all cores of the CPU for extra processing power since the sheer amount of training required in the test batteries is quite great. Far larger than what would be required to actually deploy this type of system, therefore being valid for use within the confines of the testing here presented.

### 3.1.1 Neural Network Toolbox

The four types of neural networks defined by the toolbox are for clustering, data fitting, pattern recognition and time series.

**Clustering** is used to classify data into clusters, therefore grouping samples that are alike. In this type no predefined cluster types are defined, only the number of categories that will be available to the NN.

**Data fitting** is good creating a relationship function between numeric inputs and outputs.

**Pattern recognition** will attempt to classify samples with previously known and classified training samples.

**Time series** can be used to forecast dynamic time series. A user-defined number of past inputs are used to predict future behaviour.

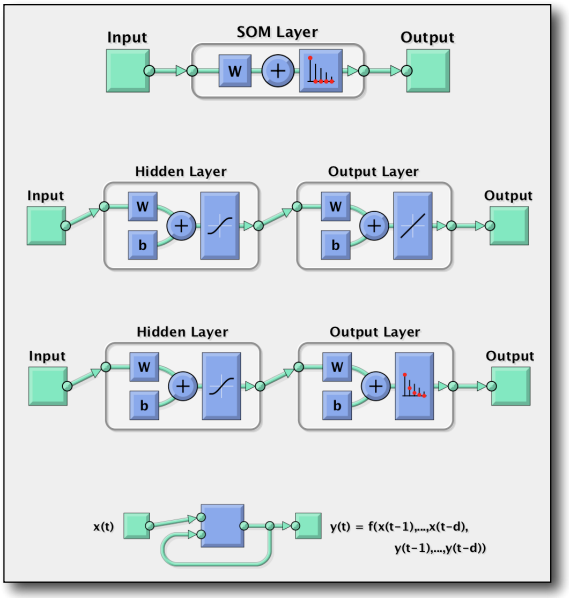


*Figure 3.1: MATLAB's NN example architectures*

*From top to bottom: clustering, data fitting, pattern recognition and time series*

Clustering could be used to find demand patterns on each day. Therefore having different profiles that repeat throughout the year. Some pre-testing was done with the strategy, the results were not outstanding and there is a great risk of creating two layers of errors within different neural networks.

The time series type is quite promising in using past values of the network to predict future ones. Pre-tests were conducted with interesting results, but not many feedback delays were required to significantly slowdown the process. Therefore, data fitting was chosen instead.

In order to give a greater focus on the problem itself both clustering and time series network types within MATLAB were dismissed in favour of data fitting was used exclusively.

### 3.1.2 Test Structure and Approach

Unless otherwise specified, the standard type used was Neural Net Fitting with a varying amount of inputs, one output and network topology. A 6 input, 10 neuron single hidden layer and single output architecture is exemplified in Figure 3.2.

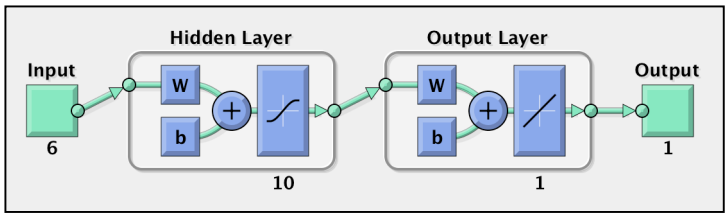


*Figure 3.2: Neural network topology example with 10 cells on a single hidden layer.*

*(taken from MATLAB NN Toolbox)*

The tests used random data division (between **Training** and **Validation** sample kinds) and Levenberg-Marquardt training algorithm.

In order to mimic as best as possible an actual implementation of a demand forecasting neural network the data will be separated and used according to the guidelines below.

**Training data:**

- Most of the data will be used to train the network.
- Whenever possible the train data will contain preferably at least one year.
- Within the NN this set will be split between the **Training**, **Validation** and **Testing** data types in a 0.75, 0.25 and 0.00 ratio. The absence of the test type is explained by the external testing performed, being therefore unrequired for the actual training. This is done in the mind-set of getting a more realistic result than testing data points spread around training ones.

**Testing data:**

- The data used to test the network accuracy will be the most recent one.
- When the training data is large the test data will contain preferably at least one month. During more limited tests it will span a single week.
- Test data, when not specified means the data used to test
- Test data does not mean section 2.1's Figure 2.4 data unless otherwise specified. That section talks about a data type within the neural network that will separate some samples to independently evaluate the network with no

direct effect on the training. This data is usually random. Since the goal here is to predict future behaviour with past values this form of testing does not serve, therefore a separate testing mechanism was employed.

Since redoing a neural network with the same parameter will very likely yield different results it is necessary to make many repetitions of the same test and evaluate the overall tendency. This will be done through the averaging of the results of each architecture and input set.

Unless otherwise specified, each of these tests will be run 50 times for different network topologies ranging from 5 to 10 neurons in the first hidden layer and from 0 to 10 neurons in the second hidden layer

The following page show the test results form to be used for the input strategies sections.

**Test Number: Title**

**Inputs:** Input1 (lower_bound1~upper_bound1), Input2 (lower_bound2~upper_bound2), Input3 (lower_bound3~upper_bound3), … , InputN (lower_boundN~upper_boundN)

**Time step:** number of minutes

**Training Duration:** number of days or months

**Testing Duration:** number of days or months

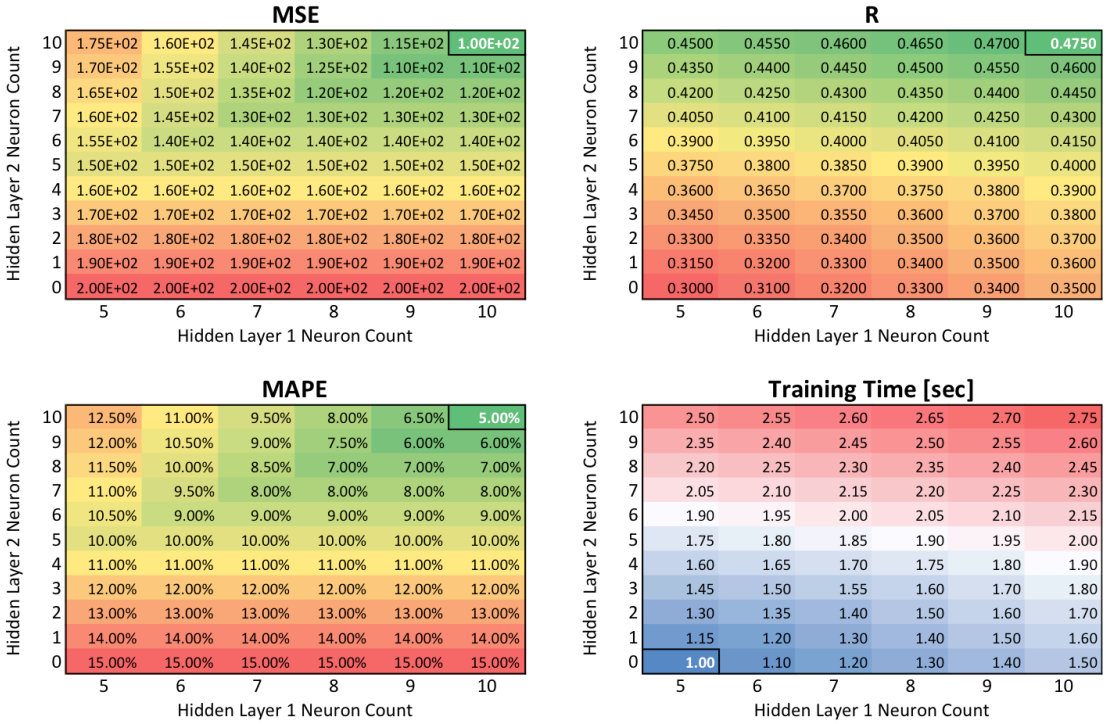


*Figure 3.3: Network topology example testing result average*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |

*Table 3.1: Template top 3 results table*

Plot of the best run including Measured and Predicted.

**Comments:** Short commentary of this test battery.

These forms include:

- Test number (sequential) and title.
- Input names with lower and upper bounds in parenthesis.
- Time step, or how much is the time difference between each measurement/sample.
- The number of days or months of the training dataset.
- The number of days or months of the testing dataset.
- A table with each run's evaluation parameters (MSE, R, MAE, MAPE) that are explained in section 3.2 and the test battery average.
- Time is the training duration taken by the test computer to train the network.
- Plot of the best network's Measured and Predicted curves.
- Short commentary of the findings. It is optional since the Discussion section is supposed to house the core concepts extracted from the tests.

The criterion used to define the 3 best networks in a test battery was chosen based on MSE. The other measurements are more easily understood, but MSE is the variable being optimized by the neural network using the Levenberg-Marquardt algorithm, resulting in it being chosen as the primary efficacy measurement.

The top 3 best networks table will have a slight variation when the test is aimed at evaluating the necessary training duration to achieve good results. On that part of the testing a "Start Time" was added to inform when the measurement started to take place to train the network.

Unless otherwise specified, each pair of neuron count (on the first and second hidden layer) will be tested for a total of 50 times and the results will be averaged in the result matrixes. This number was reached after a couple of attempts with higher and lower values in order to achieve a good result while not consuming too much time. During the pre-test phase the test count was attempted at 100 times and the whole process took over 10 hours, this is not only long but the computer is more exposed to some type of fault that may render the whole process useless.

### 3.1.3 Test Hardware

The test computer specifications were: MacBook Pro running OSX Yosemite v. 10.10.5 with 2.5 GHz Intel Core i7 processor and 16 GB 1600 MHz DDR3 RAM memory.

Since the sheer amount of neural networks being calculated for all the tests were so high, the need for additional computing power was obvious. MATLAB's Parallel Computing toolbox became a necessity. All four core of the processor were employed by activating the 'useParallel' parameter. Although, there is no certainty in how well the process is spread between each core.

Unfortunately, the GPU could not be called upon with the 'useGPU' parameter since the graphics card was an AMD Radeon R9 M370X 2048 MB and MATLAB apparently only supports CUDA-enabled NVIDIA GPUs. (MathWorks n.d.)

## 3.2 Error/Efficacy Measurements

In order to compare how effective each formulation of the neural networks outcomes it is necessary to have a quantitative error or efficacy measurements. The ones used in this paper are MSE, R, MAE and MAPE. Each of them are described in this section and will be utilized on the tests performed on the "Application" sections.

MSE is the objective function being minimized by the Levenberg-Marquardt algorithm on the neural network. But, it is wise to have additional measurements that explain the performance of the system from different points of view. Although tending to increase or decrease together, this is not a rule and the behaviour of a system can be better understood with the help of multiple quantitative evaluation tools.

| **Acronym** | **Definition** | **Short Synthesis** |
|---|---|---|
| MSE | Mean Squared Error | Average squared difference between measured and predicted variables. The squared characteristic gives greater emphasis on large errors. |
| R | Correlation Coefficient | Linear correlation coefficient, or how much does the measured and predicted values have a good linear relationship. Since it is bounded from -1 to 1 it is easier to compare different systems. |
| MAE | Mean Absolute Error | Average difference between the measured and predicted variables. |
| MAPE | Mean Absolute Percentage Error | Average perceptual difference between the measured and predicted variables. It is a more intuitive error measurement. Does not have an overt bias such as MSE. |

*Table 3.2: Error/Efficacy Measurements short synthesis*

Each individual error/efficacy measurement is discussed and shown in more detail below.

### 3.2.1 MSE – Mean Squared Error

As the name suggests, MSE is an average measurement of all squared values of individual errors, or the difference between predicted and measured variable of

interest. The MSE should be as low as possible for an effective NN. The formula used is as shown bellow.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - Y_i\right)^2$$

where:

- **n** is the number of samples.
- $\mathbf{X_i}$ is the *i* sample's predicted value.
- $\mathbf{Y_i}$ is the *i* sample's measured value.

Since the error value is squared, bigger error values receive more emphasis than smaller ones. Therefore, outliers have a greater impact on MSE. Therefore it is a good indicator of the present of large errors and not the average error of the system.

MSE cannot be used to compare systems that have different ranges if they are not normalized first. E.g.: a system in which the output goes from 0 to 300 kW cannot be compared using MSE on a system that goes from 0 to 1,000 kW, since it will vary a lot in function with the squared error measurements.

MSE is one of the more frequent objective functions used by neural networks. Minimizing it is the whole point of the algorithm.

### 3.2.2 R – Correlation Coefficient

The correlation coefficient R is a coefficient that informs how much two arrays correlate to one another linearly. The equation below demonstrates how to calculate R.

$$R = \sum_{i=1}^{n}\frac{\left(\left(X_i - M_y\right)\left(Y_i - M_x\right)\right)}{\sqrt{\left(\left(SS_x\right)\left(SS_y\right)\right)}}$$

$$SS_x = \sum_{i=1}^{n}\left(X_i - M_x\right)^2 \qquad SS_y = \sum_{i=1}^{n}\left(X_i - M_y\right)^2$$

where:

- **n** is the number of samples.
- $\mathbf{X_i}$ is the *i* sample's predicted value.
- $\mathbf{Y_i}$ is the *i* sample's measured value.
- $\mathbf{M_x}$ is the mean of $X_i$ values

- $\mathbf{M_y}$ is the mean of $Y_i$ values

The value of R ranges from -1 to 1. A value of -1 indicates total negative linear correlation, while a 1 indicates a total positive linear correlation and a 0 represents no correlation at all.

If the goal is to compare two arrays that are supposed to be as similar as possible a near positive one value is to be expected.

### 3.2.3 MAE – Mean Absolute Error

This error metric is the average of the module of all error measurements. So, it shows the positive error average of a sampling pool. MAE should be as low as possible for an effective NN.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|X_i - Y_i|$$

where:

- **n** is the number of samples.
- $\mathbf{X_i}$ is the *i* sample's predicted value.
- $\mathbf{Y_i}$ is the *i* sample's measured value.

This tool is useful to give a feeling of how much error each sample has produced in average. Again, as the MSE it does not translate well between two different systems with different output ranges. Therefore, there are better tools than MAE, such as MAPE.

### 3.2.4 MAPE – Mean Absolute Percentage Error

MAPE, is the average of the module of the perceptual error. MAPE should be as low as possible for an effective NN.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - X_i}{Y_i}\right|$$

where:

- **n** is the number of samples.
- $\mathbf{X_i}$ is the *i* sample's predicted value.
- $\mathbf{Y_i}$ is the *i* sample's measured value.

MAPE in essence represents how much in average a model is distant from the measured reality. This measurement is very interesting in the sense that the result is in percentage form, therefore a value that humans are more used to encounter and more easily understood. Differently from MSE and MAE different systems can be compared even when they gave different output ranges.

One weakness of the MAPE is that null values of the measured variable take the result to infinity. Since during the tests performed in the Application sections some of these where encountered a safeguard was put in place that if the measured sample was zero the error value for that data point would be set to 1, or 100%.

MAPE can be shown as a percentile value. This paper will adopt this format since it is easier to be understood that way.

# 4 Application at Macrogrid Level

This section encompasses “hard” results of the testing done with Macrogrid level datasets.

The objective of the tests was to find if with only time and demand data information a good prediction could be made on what is considered the easiest level to forecast. Additionally, what form or treatment the inputs required to reach the desired results. Since, data on this level is abundant there was no exploration on dataset size.

## 4.1 Introduction

G.B. Grid Watch is an online 5-minute step database of the behaviour of Gran Britain’s grid behaviours. There are plenty of data available for past years to be found of not only demand, but also supply (from different sources) and frequency.

Using the G.B. Grid Watch database an attempt was made to create a NN that would predict future demand behaviour. This section will explore how the tests evolved to reach good forecasting results.

The following attempts trained the network with data from the calendar year of 2015 (January to December) to train the network and data from 2016 (January to July) to test the forecasting proficiency of the network.

## 4.2 Test Results

### 4.2.1 Input Strategies

This sub section is focused on different input sets and how they affect the end result. Firstly, the histogram for each both the training and testing datasets will be analysed.

**Training dataset histogram:**

Figure 4.1 shows the histogram for the day of year samples within the training data set, or the data pool used by the network to train its weights in order to reach a good generalization of the rules driving the demand. The values range from 1 to 365, or from January to December of 2015. There are entries for each of the demand samples.

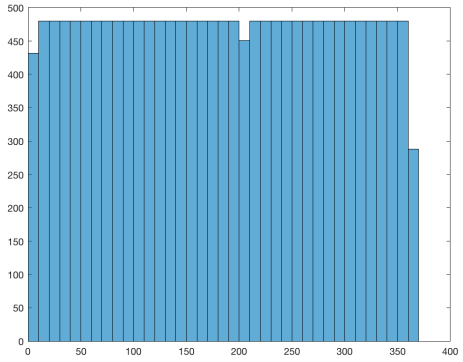


*Figure 4.1: GB training dataset day of year histogram*

Just like the test set there are a few missing points missing from the data pool. Overall this dataset represents the whole year of 2015 for the G.B. Grid Watch.

Figure 4.2 shows a close-up of the demand histogram for the training data set and its unit is mega-watt (MW). The curve is nicely distributed. Therefore, there is not much of a bias towards any particular value, which avoids making the network less of a generalist.

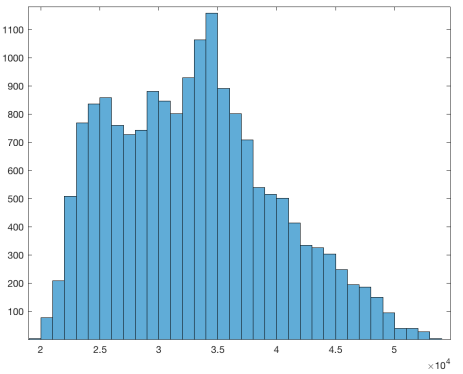


*Figure 4.2: GB training dataset demand histogram close-up*

The actual histogram is represented in Figure 4.3. There are 3 very big outliers that simply spike on

three different dates. Since they are so few it is not visible on the graph. Due to them being so high and lasting a mere 30 minutes time step it is very likely that they are reading errors, but will be kept regardless.

The G.B. Watch training pool includes a total of 17,491 entries.

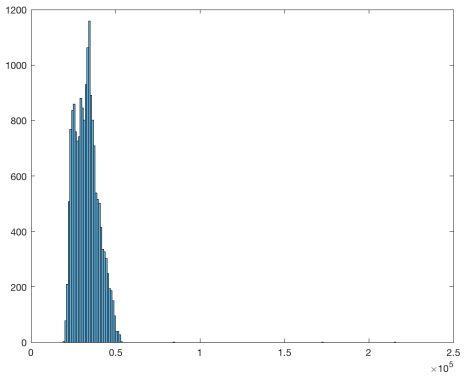


*Figure 4.3: GB training dataset demand full histogram*

**Testing dataset histogram:**

Figure 4.4 shows Grand Britain's Grid Watch histogram for the day of year samples within the test data set, or the data pool used after the network is trained to evaluate the forecasting power of the network. The values range from 1 to 190, or from January to mid-July of 2016. There are entries for each of the demand samples.

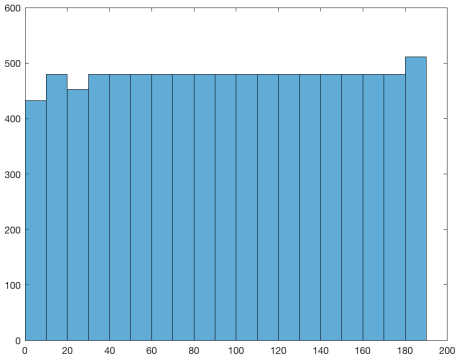


*Figure 4.4: GB test dataset day of year histogram*

Although G.B. Grid Watch has a 5-minute interval between each sampling a time step of 30 minutes was used since higher levels of detail is unneeded. Each day that was fully sampled will have 48 entries on the histogram. It can be noted that there are missing data points on the [0,10[ and [20,30[ columns. The last column is slightly larger because its range is [180,190], therefore including one additional day beyond its peers.

Figure 4.5 shows Great Britain's Grid Watch demand histogram for the test data set, its unit is mega-watt (MW). Overall it is a good curve to work with since it has a "normal like" aesthetic to it.

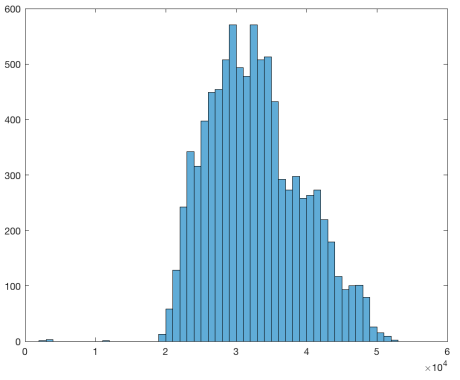


*Figure 4.5: GB test dataset demand histogram*

There are some outlier data points where the demand can reach low values. This can be due to blackouts or straight reading errors. Just 5 entries were lower than 19,000 MW and they were all within the same week, which could indicate plausible reasons for happening. Regardless, it was kept and the networks will have to deal with this sort of "trash data".

The G.B. Watch testing pool includes a total of 9,075 entries.

### *Test 4.1 – GB, Date/time inputs only*

**Inputs:** Year (2015~2016), Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7)

**Time step:** 30 minutes

**Training Duration:** 12 months

**Testing Duration:** 6.5 months

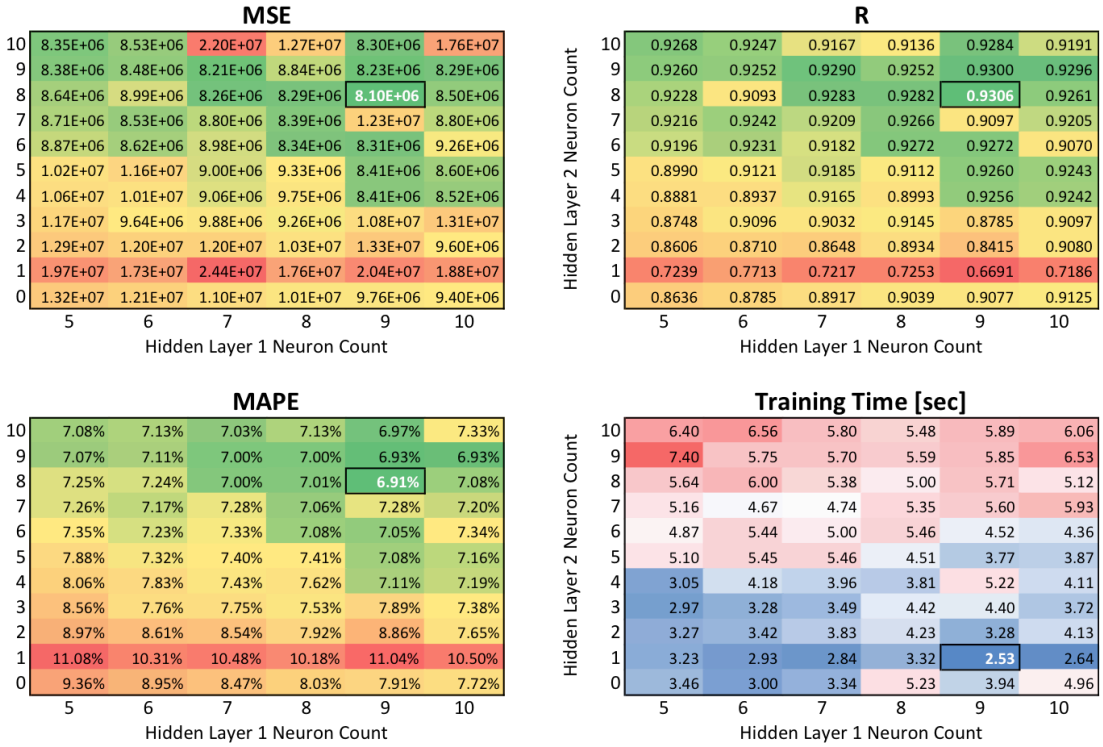


*Figure 4.6: Test 4.1 – GB, Date/time inputs only average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 7,141,251.7 | 0.9390 | 1,923.3 | 6.51 | 3.14 | 9 | 10 |
| 7,210,620.2 | 0.9365 | 1,924.8 | 6.50 | 4.52 | 9 | 4 |
| 7,218,014.3 | 0.9383 | 1,958.3 | 6.60 | 5.75 | 10 | 4 |

*Table 4.1: Test 4.1 – GB, Date/time inputs only top 3 results*

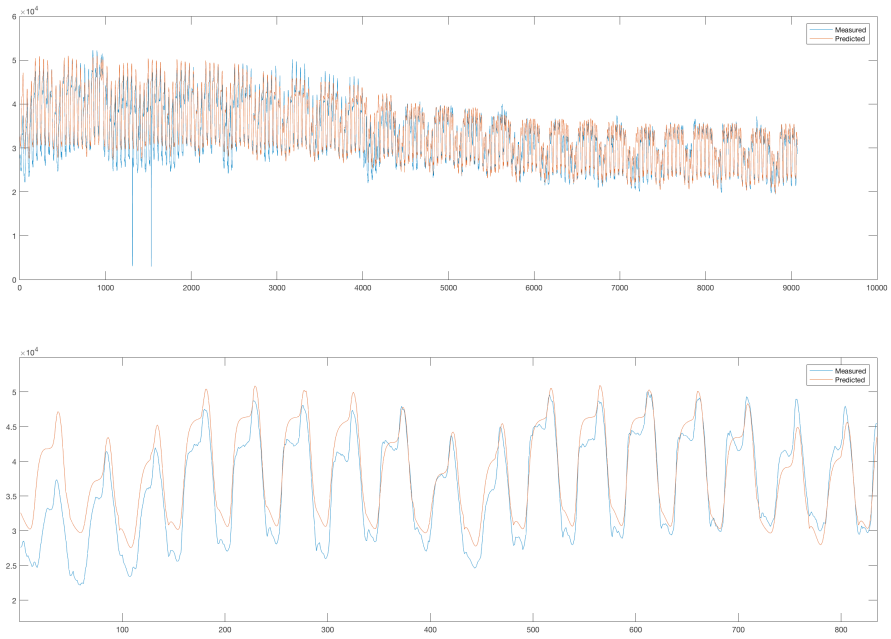


*Figure 4.7: Test 4.1 – GB, Date/time inputs, best general and close-up graph*

*X-axis = Time Step (30 min), Y-axis = Demand (MW)*

**Comments:** Interestingly, Gran Britain macrogrid demand is so predictable that a network with only the most basic date inputs and trained with only 2015 data is enough to yield quite accurate results. But, better results are likely with better input strategies.

It recommends ideally above 8 neurons in the first hidden layer.

### *Test 4.2 – GB, Year removed*

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7)

**Time step:** 30 minutes

**Training Duration:** 12 months

**Testing Duration:** 6.5 months


*Figure 4.8: Test 4.2 – GB, Year removed average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 7,089,826.7 | 0.9391 | 1,920.5 | 6.53% | 3.36 | 9 | 5 |
| 7,166,602.8 | 0.9333 | 1,969.5 | 6.67% | 10.94 | 7 | 6 |
| 7,176,580.4 | 0.9370 | 1,914.7 | 6.54% | 5.12 | 10 | 7 |

*Table 4.2: Test 4.2 – GB, Year removed top 3 results*

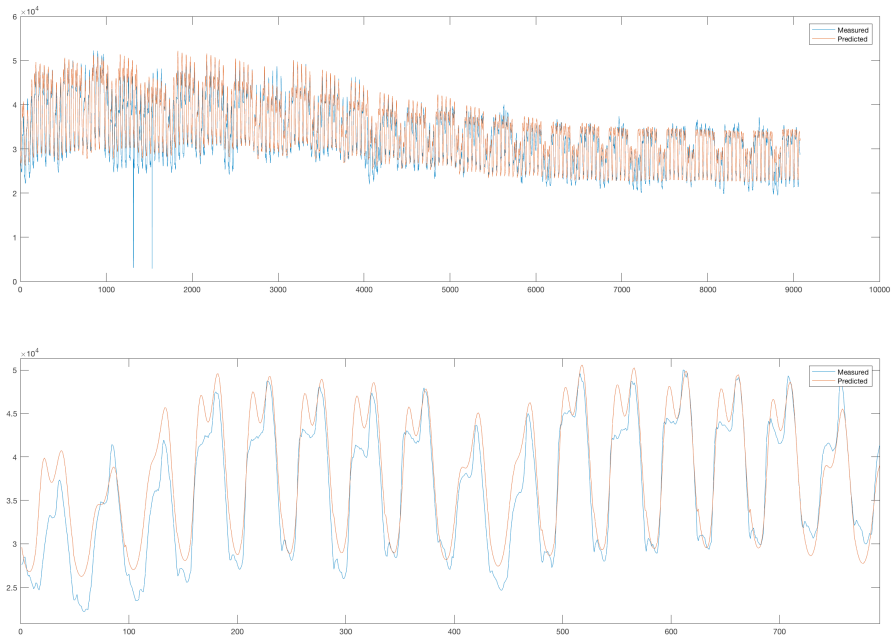


*Figure 4.9: Test 4.2 – GB, Year removed, best general and close-up graph*

*X-axis = Time Step (30 min), Y-axis = Demand (MW)*

**Comments:** Figure 4.10 compares the results with and without the year. In it, the values for the testing done for each battery are subtracted from one another and the resulting number signal is analysed. Red values lean towards the yearless networks while blue towards the one that contains the year. The removal of outliers, e.g. on MSE [9,6] and [5,10], further this notion.

Since the training data only contains one year (2015) the network is not supposed to actually learn anything from it. Therefore it is not seen as useful to keep it as an input.

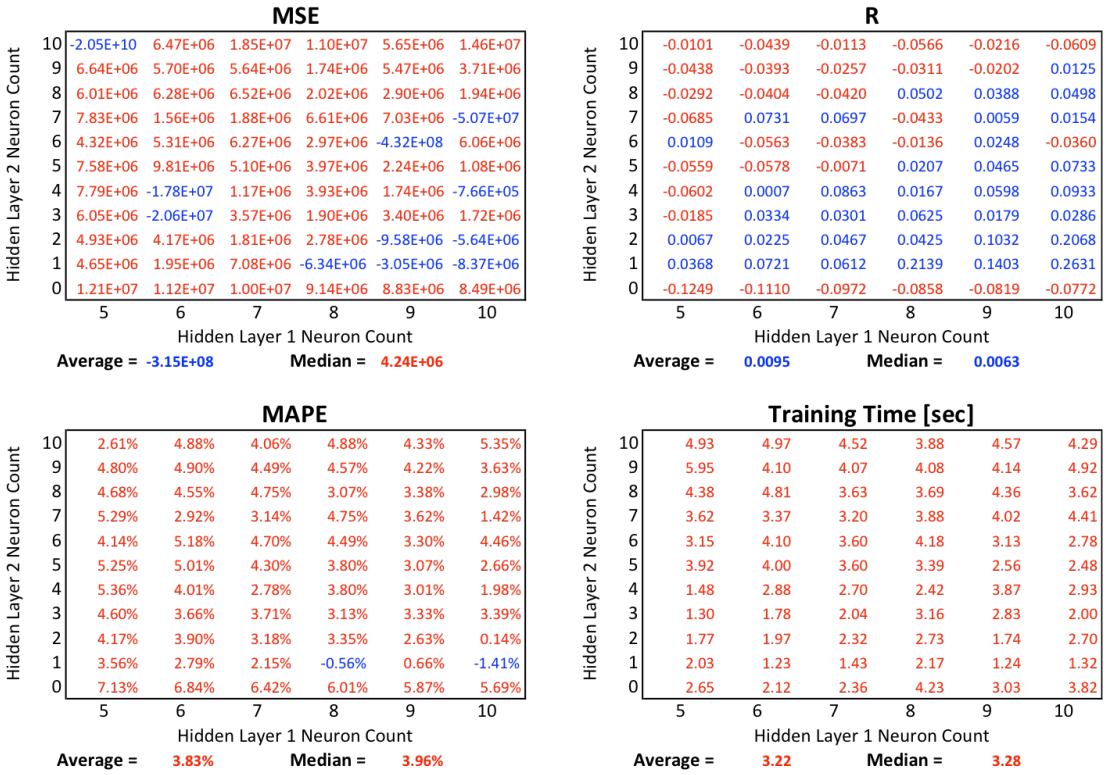


*Figure 4.10: Test 4.1 and Test 4.2 comparison*

Red values represents a better result on Test 4.2 (without year input) and blue indicates a better result on Test 4.1 (with year input).

### *Test 4.3 – GB, Previous hour demand*

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous hour demand (0~220,00 MW)

**Time step:** 30 minutes

**Training Duration:** 12 months

**Testing Duration:** 6.5 months

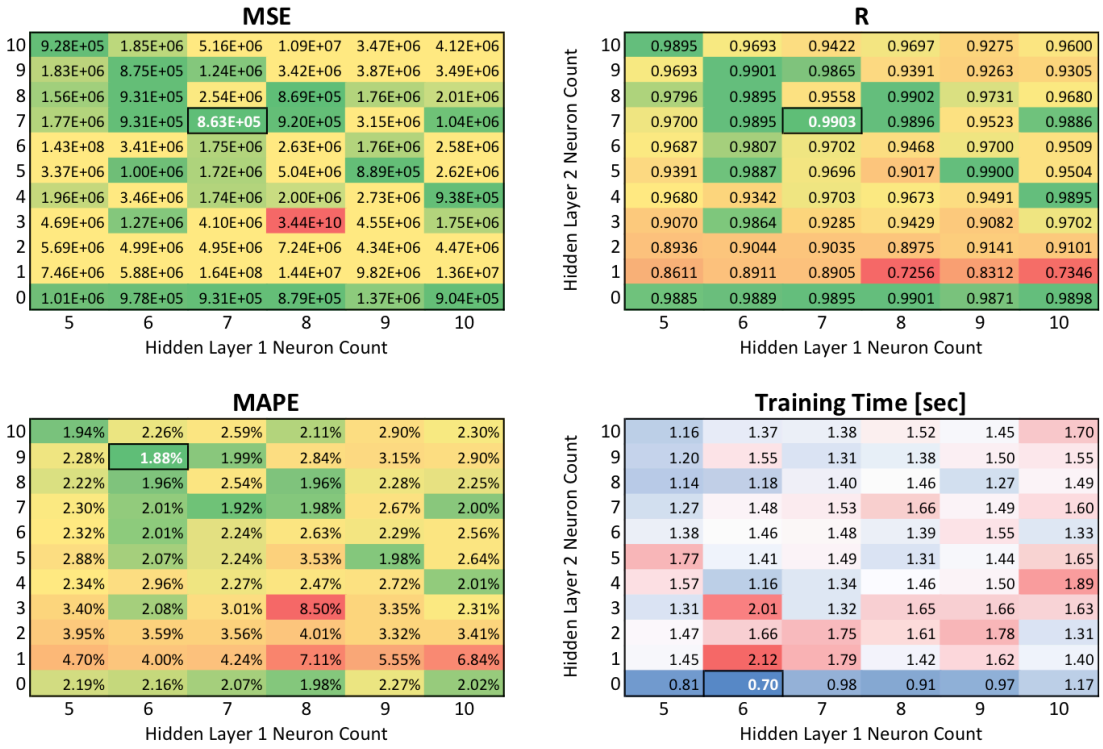


*Figure 4.11: Test 4.3 – GB, Previous hour demand average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 552,772.9 | 0.9937 | 399.0 | 1.49% | 1.46 | 10 | 9 |
| 571,266.1 | 0.9935 | 393.0 | 1.49% | 2.71 | 10 | 7 |
| 579,174.7 | 0.9935 | 397.4 | 1.47% | 1.66 | 10 | 10 |

*Table 4.3: Test 4.3 – GB, Previous hour demand top 3 results*

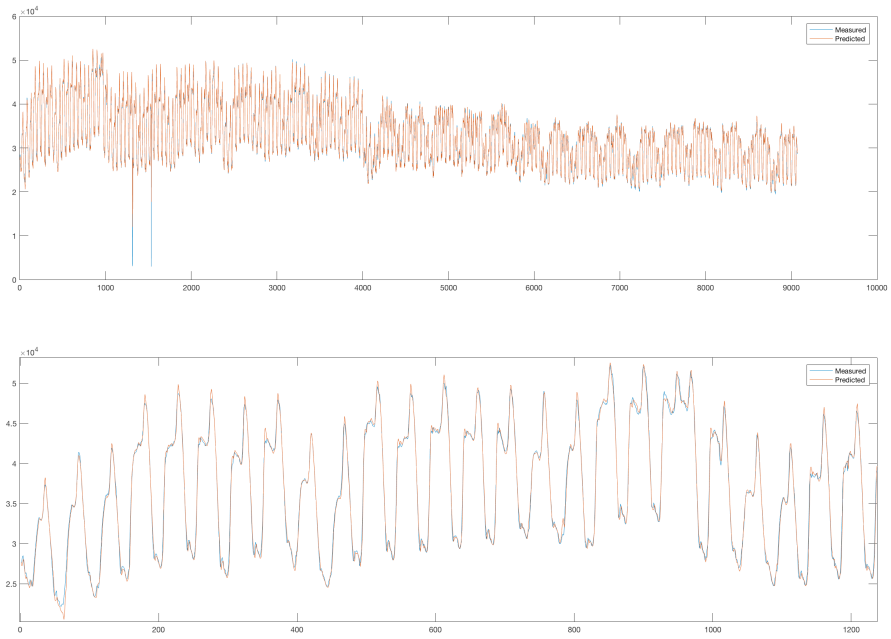


*Figure 4.12: Test 4.3 – GB, Previous hour demand best general and close-up graph*

*X-axis = Time Step (30 min), Y-axis = Demand (MW)*

**Comments:** The previous hour demand of each point was included to give the NN a better idea of the demand output to be expected. Therefore, this network forecasts with one hour in advance. This result is very good, but it is a shorter term forecast then what this report is aiming for.

### *Test 4.4 – GB, Previous hour demand and day of year*

**Inputs:** Day of year (1~365), Hour (0~23.5), Day of the week (1~7), Previous hour demand (0~220,00 MW)

**Time step:** 30 minutes

**Training Duration:** 12 months

**Testing Duration:** 6.5 months

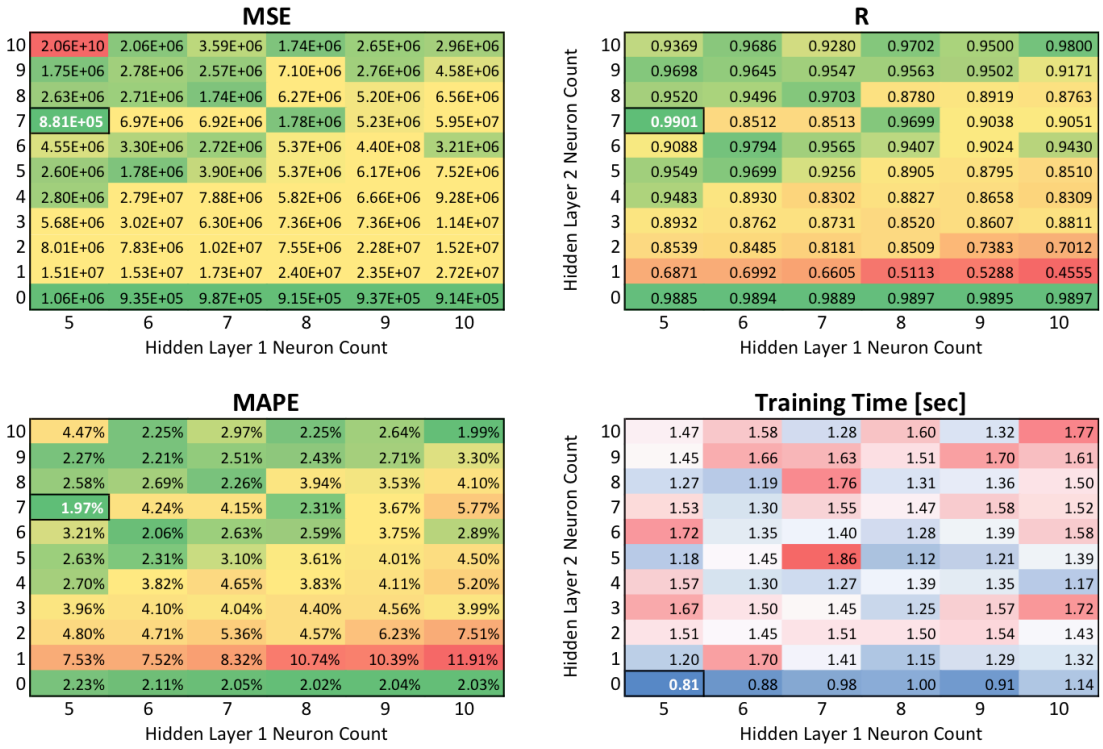


Figure 4.13: Test 4.4 – GB, Previous hour demand and day of year average test results

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 559,966.4 | 0.9937 | 384.7 | 1.41% | 1.25 | 9 | 8 |
| 583,143.2 | 0.9934 | 369.6 | 1.34% | 1.64 | 8 | 10 |
| 584,951.8 | 0.9934 | 392.9 | 1.47% | 3.23 | 10 | 10 |

*Table 4.4: Test 4.4 – GB, Previous hour demand and day of year top 3 results*

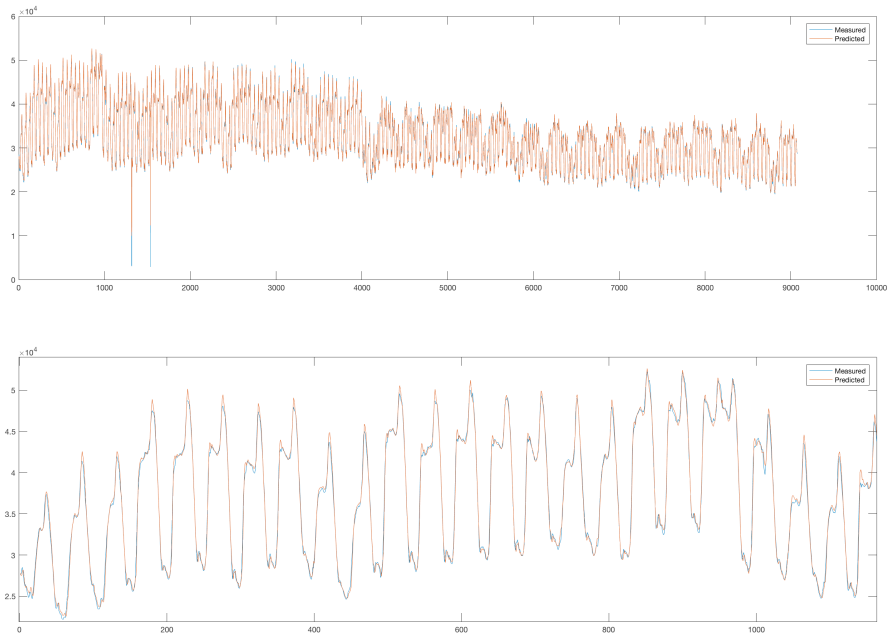


*Figure 4.14: Test 4.4 – GB, Previous hour demand and day of year, best general and close-up graphs*

*X-axis = Time Step (30 min), Y-axis = Demand (MW)*

**Comments:** The substitution of the month and day for the "day of year" produced similar results.

Figure 4.15 compares the results from the "day of year" and the month/day combination. Red values lean towards the month/day inputs while blue towards the "day of year" scheme. There is a slight predominance of the regular date scheme. The removal of outliers, e.g. on MSE [8,3], aid this scenario.

This result is plausible from the standpoint that with a full year's data set the most important aspect is not on the day itself and more on the season, in this case better represented by the months. This might change with different tests and data set, specially with incomplete ones.

Further testing is required. Regardless, the month/day scheme will be kept.

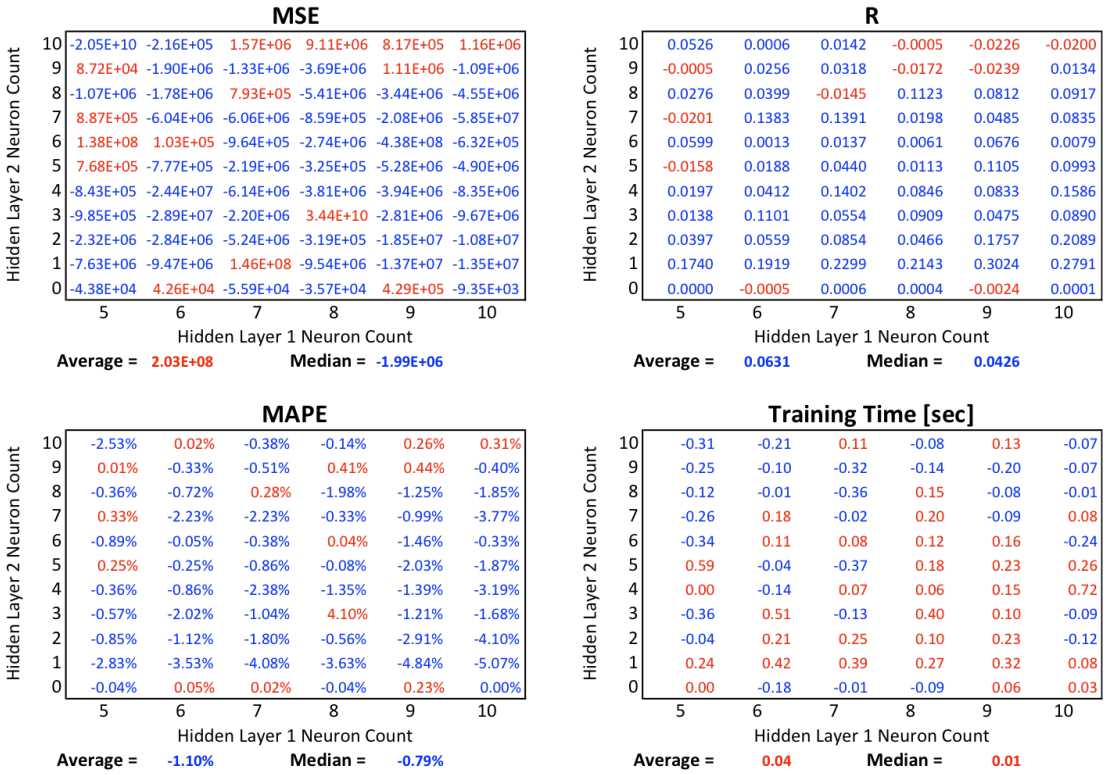


*Figure 4.15: Test 4.3 and Test 4.4 comparison*

Red values represents a better result on Test 4.4 (with day of the year) and blue indicates a better result on Test 4.3 (with month and day inputs).

### *Test 4.5 – GB, Previous day's average demand*

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous day's average demand (0~55,00 MW)

**Time step:** 30 minutes

**Training Duration:** 12 months

**Testing Duration:** 6.5 months

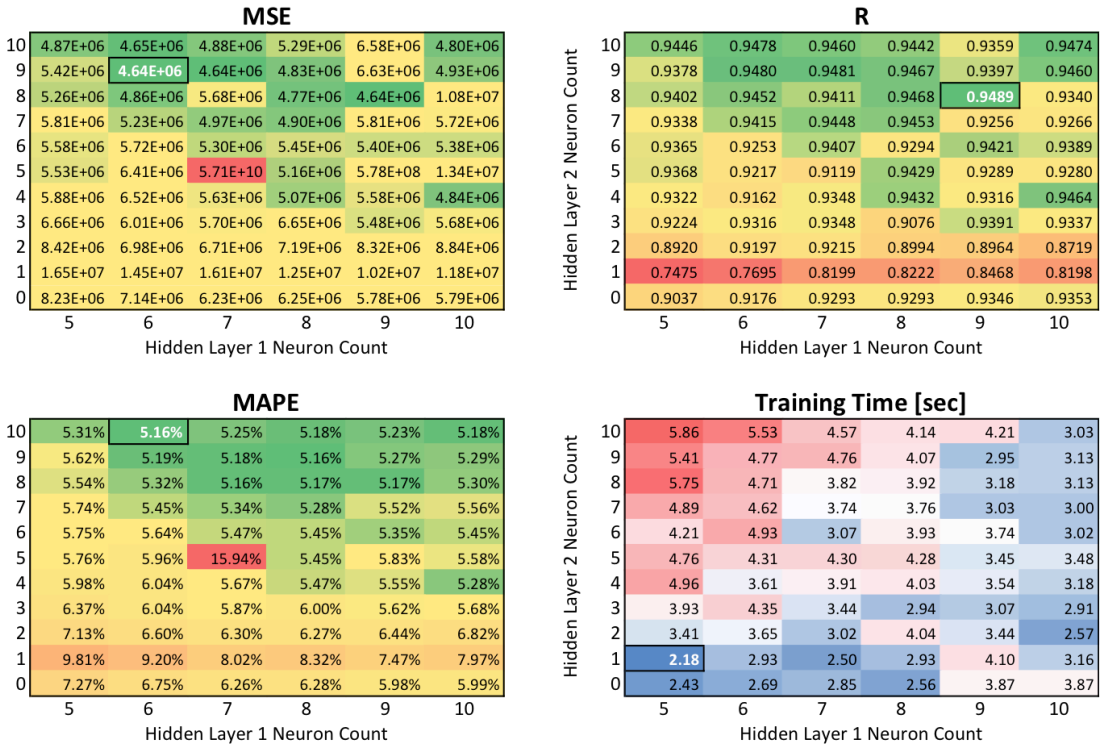


Figure 4.16: Test 4.5 – GB, Previous day's average demand average test results

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 3,800,879.9 | 0.9569 | 1,380.5 | 4.70% | 2.81 | 9 | 7 |
| 3,823,317.6 | 0.9563 | 1,383.6 | 4.71% | 2.65 | 9 | 10 |
| 3,853,457.7 | 0.9575 | 1,375.7 | 4.72% | 6.16 | 7 | 9 |

*Table 4.5: Test 4.5 – GB, Previous day's average demand top 3 results*

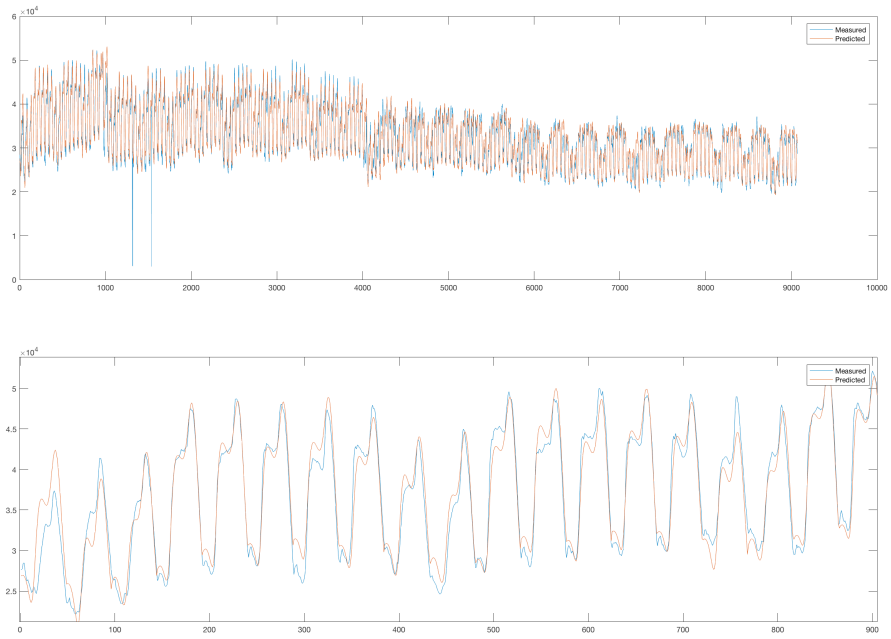


*Figure 4.17: Test 4.5 – GB, Previous day's average demand, best general and close-up graphs*

*X-axis = Time Step (30 min), Y-axis = Demand (MW)*

**Comments:** The network learned a standard demand pattern and scales it according to the last day's average demand while taking into consideration the date variables. With this it is possible to have a day lead forecast.

### *Test 4.6 – GB, Previous 24 hour demand*

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~220,00 MW)

**Time step:** 30 minutes

**Training Duration:** 12 months

**Testing Duration:** 6.5 months


Figure 4.18: Test 4.6 – GB, Previous 24 hour demand average test results

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 4,223,905.6 | 0.9531 | 1,385.3 | 4.78% | 2.84 | 6 | 7 |
| 4,261,385.1 | 0.9535 | 1,391.9 | 4.83% | 2.06 | 10 | 7 |
| 4,281,058.9 | 0.9532 | 1,395.7 | 4.81% | 2.52 | 6 | 9 |

*Table 4.6: Test 4.6 – GB, Previous 24 hour demand top 3 results*

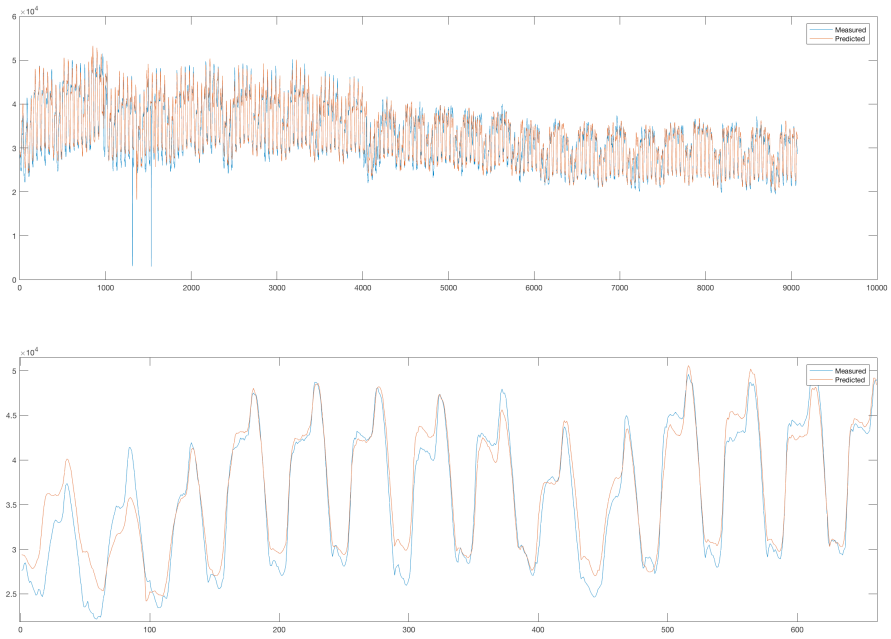


*Figure 4.19: Test 4.6 – GB, Previous 24 hour demand, run 1 general and close-up graphs*

*X-axis = Time Step (30 min), Y-axis = Demand (MW)*

**Comments:** In order to expand the forecasting capabilities an attempt that uses the demand 24-hour in the past (the same time of day on the day before) as an alternative from Test 4.5. The results are very good and show a small absolute error of a little above 5%. Differently from Test 4.5 the shape of the demand changes depending on the previous 24 hour demand and the graph is not as smooth.

The comparison made in Figure 4.20 indicates that the previous day' average demand is a better forecasting strategy, especially if the outlier [7,5] in the MSE is removed.

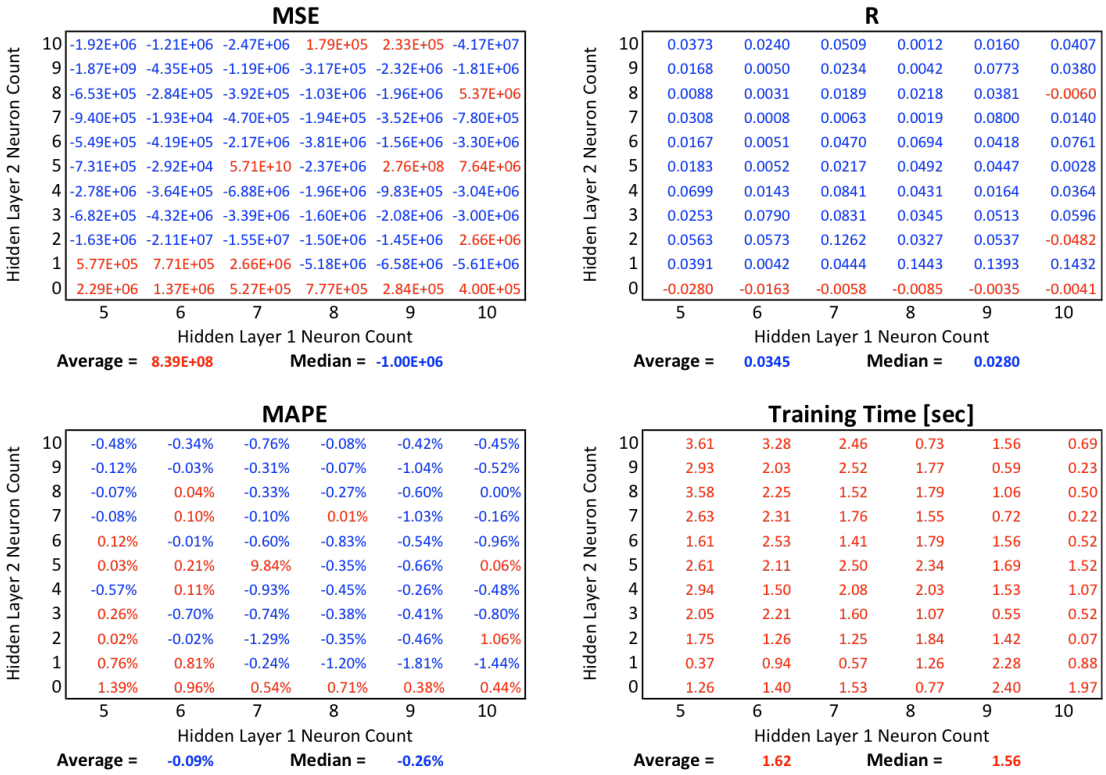


*Figure 4.20: Test 4.5 and Test 4.6 comparison*

Red values represents a better result on Test 4.6 (previous 24-hour) and blue indicates a better result on Test 4.5 (previous day's average demand).

### *Test 4.7 – GB, No date and previous day's average demand*

**Inputs:** Hour (0~23.5), Day of the week (1~7), Previous day's average demand (0~55,00 MW)

**Time step:** 30 minutes

**Training Duration:** 12 months

**Testing Duration:** 6.5 months

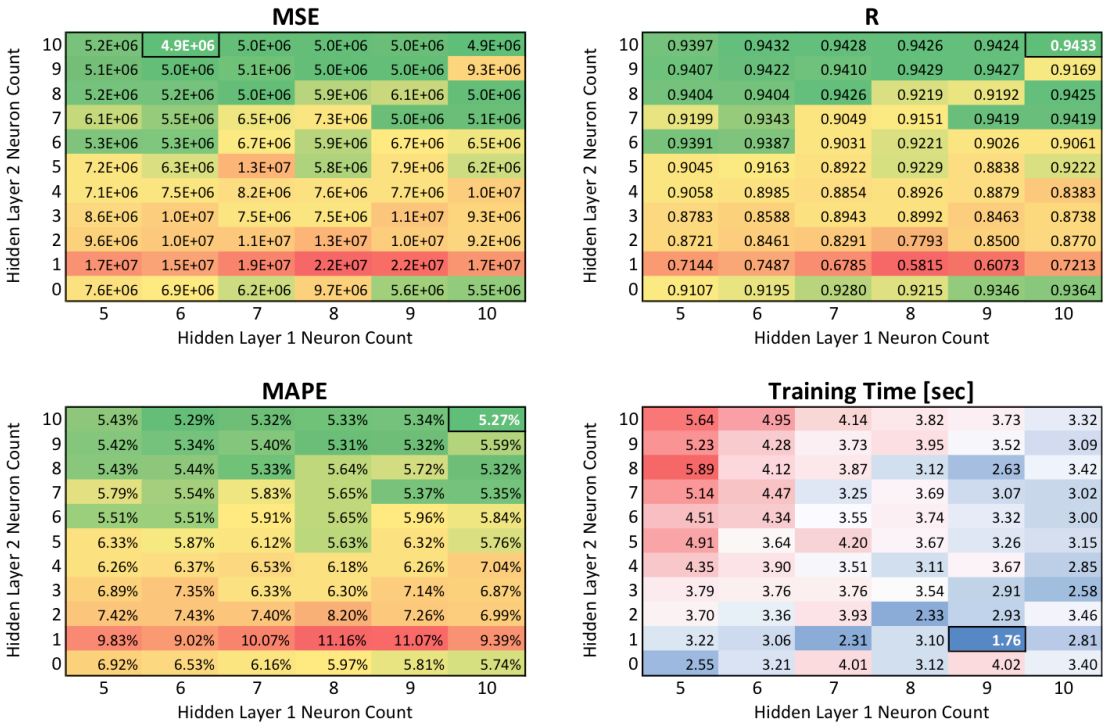


Figure 4.21: Test 4.7 – GB, No date and previous day's average demand average test results

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 4,455,212.3 | 0.9490 | 1,507.7 | 5.04% | 5.30 | 8 | 7 |
| 4,539,204.3 | 0.9476 | 1,523.7 | 5.09% | 2.42 | 10 | 7 |
| 4,539,496.2 | 0.9481 | 1,529.2 | 5.11% | 4.50 | 9 | 7 |

*Table 4.7: Test 4.7 – GB, No date and previous day's average demand top 3 results*

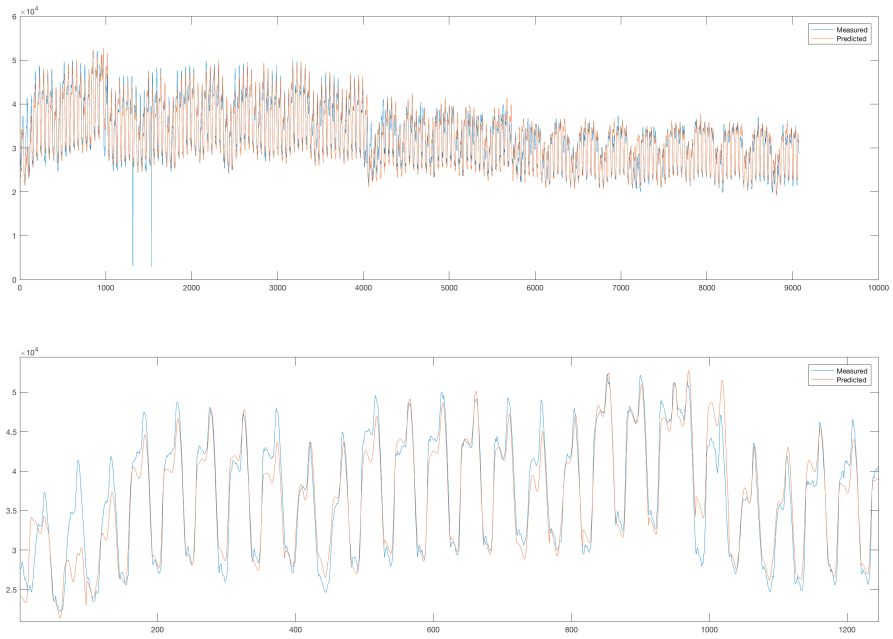


*Figure 4.22: Test 4.7 – GB, No date and previous day's average demand, best general and close-up graphs*

*X-axis = Time Step (30 min), Y-axis = Demand (MW)*

**Comments:** Figure 4.23 shows that this configuration is only marginally better in this scenario. Therefore, for the macrogrid level where information is supposedly abundant it is advisable to keep date variables in the model.

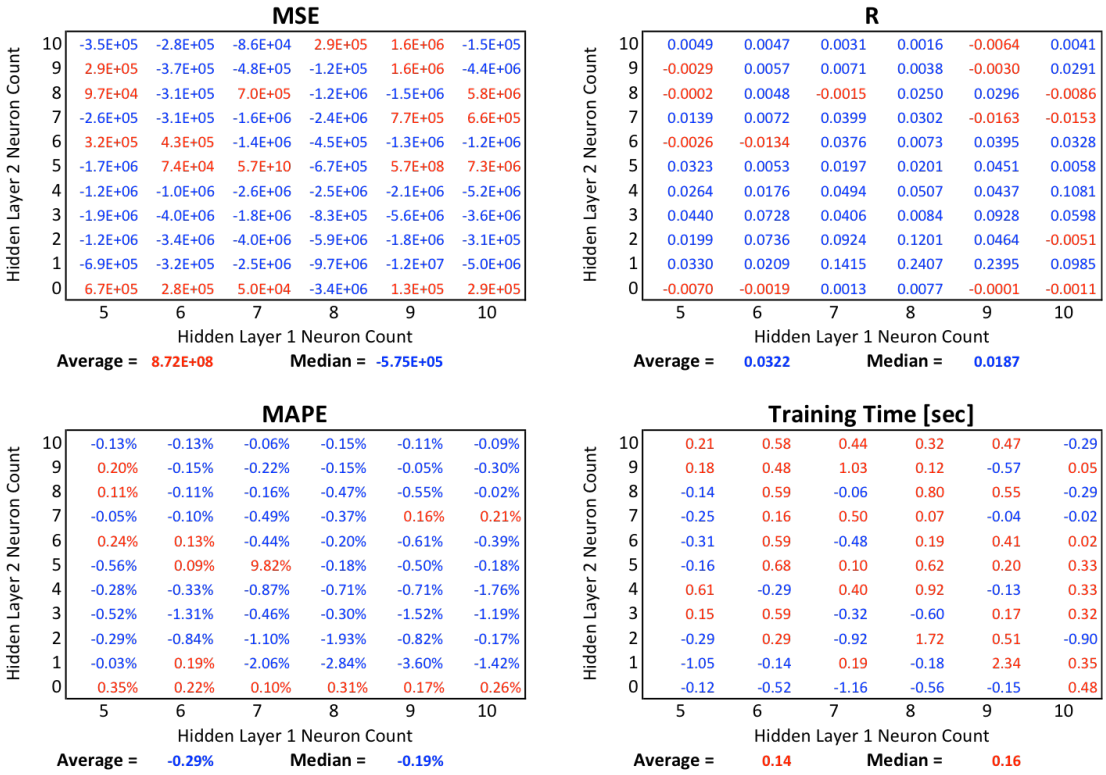


*Figure 4.23: Test 4.5 and Test 4.7 comparison*

Red values represents a better result on Test 4.7 (without date inputs) and blue indicates a better result on Test 4.5 (with date inputs).

## 4.3 Overview

The following table contains the best run for the test batteries with one-hour ahead forecast capability.

| Test # / Title | MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|---|
| Test 4.3 – GB, Previous hour demand | 552,772.9 | 0.9937 | 399.0 | 1.49% | 1.46 | 10 | 9 |
| Test 4.4 – GB, Previous hour demand and day of year | 559,966.4 | 0.9937 | 384.7 | 1.41% | 1.25 | 9 | 8 |

*Table 4.8: GB Grid Watch, **one hour lead** forecast test best results overview*

The following table contains the best run for the test batteries with one-day ahead forecast capability.

| Test # / Title | MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|---|
| Test 4.1 – GB, Date/time inputs | 7,141,251.7 | 0.9390 | 1,923.3 | 6.51 | 3.14 | 9 | 10 |
| Test 4.2 – GB, Year removed | 7,089,826.7 | 0.9391 | 1,920.5 | 6.53% | 3.36 | 9 | 5 |
| Test 4.5 – GB, Previous day's average demand | 3,800,879.9 | 0.9569 | 1,380.5 | 4.70% | 2.81 | 9 | 7 |
| Test 4.6 – GB, Previous 24 hour demand | 4,223,905.6 | 0.9531 | 1,385.3 | 4.78% | 2.84 | 6 | 7 |
| Test 4.7 – GB, No date and previous day's average demand | 4,455,212.3 | 0.9490 | 1,507.7 | 5.04% | 5.30 | 8 | 7 |

*Table 4.9: GB Grid Watch, **one day lead** forecast test best results overview*

Test 4.5 had the best run that with a day ahead forecasting capability. These tests show that for a macro grid level with minimal data (time/demand) a very decent forecast can be made with relative ease. Even with only date/time inputs the results were quite good. Additionally, it is worth repeating that the data sets had untreated gaps derived from the G.B. Grid Watch.

More tests could be made to further optimize this network: like checking how much data is needed to train it properly and evaluating retraining requirements. But, on the macrocrid level the demand is shared between many consumers and produces a more consistent load profile. Therefore, more effort should be allocated elsewhere, on a lower network levels.

Since data availability is not an issue with the macrogrid there is no shortage of information. A new database is not necessary to be constructed before any type of forecasting can be deployed. Therefore, it is not necessary to test how much data is required to properly train a neural network.

At this level even a very poor set of inputs yielded good results, therefore testing further in this level is unrequired.

# 5 Application at Community Level

This section encompasses “hard” results of the testing done with Community level datasets.

The objective here is to explore input selections, the impact of the database size on the performance and analyse how different is the impact of the prediction depending on the time of day. These will be explored in Input Strategies, Dataset Build-up Time and Performance Distribution sub-sections respectively.

## 5.1 Introduction

Findhorn is a village in Moray, Scotland. The ORIGIN project developed there created a year worth of demand readings that will be used in this section to test at the community level forecasting using neural networks.

With a community level grid the foreseeability of the demand is expected to be lesser than that of the macrogrid. With a larger set of consumers the macrogrid averages out their uncertainties. A benefit that is greatly reduced on communities.

The database available for Findhorn community’s electrical energy demand ranged from September 2014 to September 2015.

## 5.2 Test Results

### 5.2.1 Input Strategies

This sub section is focused on different input sets and how they affect the end result. Firstly, the histogram for each both the training and testing datasets will be analysed.

**Training dataset histogram:**

Figure 5.1 shows Findhorn's training data set, or the data pool used by the network to train its weights in order to reach a good generalization of the rules driving the demand. It can range from 1 to 365, and covers from August 2014 to July 2015. There are entries for each of the demand samples.

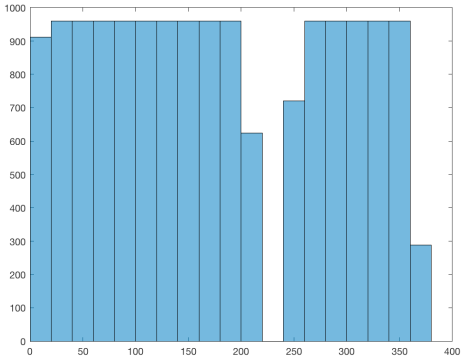


*Figure 5.1: FH training dataset day of year histogram*

The missing data is mostly from the month separated for the test dataset.

Figure 5.2 shows Findhorn's demand histogram for the training data set, its unit is kilo-watt (kW). The values encountered ranged from 0 to 300 kW. The curve is nicely distributed. Therefore, there is not much of a bias towards any particular value, which avoids making the network less of a generalist.

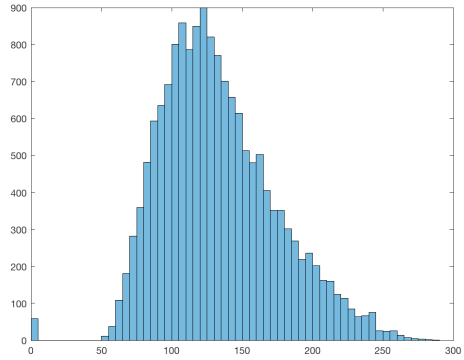


*Figure 5.2: FH training dataset demand histogram*

There are 58 outlier data points where entries were lower than 1 KW. Some of them occur continuously, quite likely due to planned maintenance, a few others are more spurious and are probably reading errors. They were kept in the database.

Findhorn's training pool includes a total of 15,984 entries.

**Testing dataset histogram:**

Figure 5.3 shows Findhorn's test data set, or the data pool used after the network is trained to evaluate the forecasting power of the network. The values range from 213 to 244, August early September of 2015. There are entries for each of the demand samples.

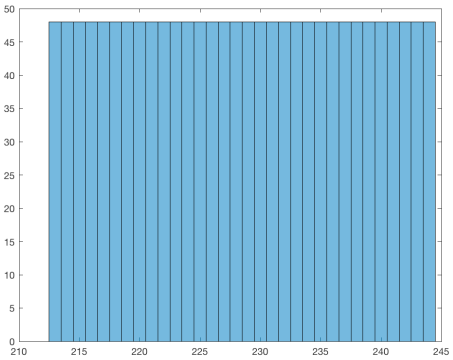


*Figure 5.3: FH test dataset day of year histogram*

As the histogram depicts there are no missing data points, which is a good sign of clean data.

Figure 5.4 shows the demand histogram for Findhorn's test data set, its unit is kilo-watt (kW). The values encountered ranged from 0 to 180 kW. No biases are expected from the form.

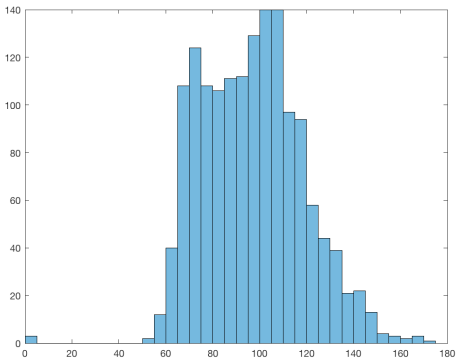


*Figure 5.4: FH test dataset demand histogram*

There are 3 outlier data points where entries were 0 KW. They occur in the last 1.5 hour of the 1st September 2015. Since they are continuous it might be due to technical faults in the network. They were kept in the database.

Findhorn's testing pool includes a total of 1,536 entries.

### *Test 5.1 – FH, Previous day average demand*

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous day average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 11 months

**Testing Duration:** 1 month

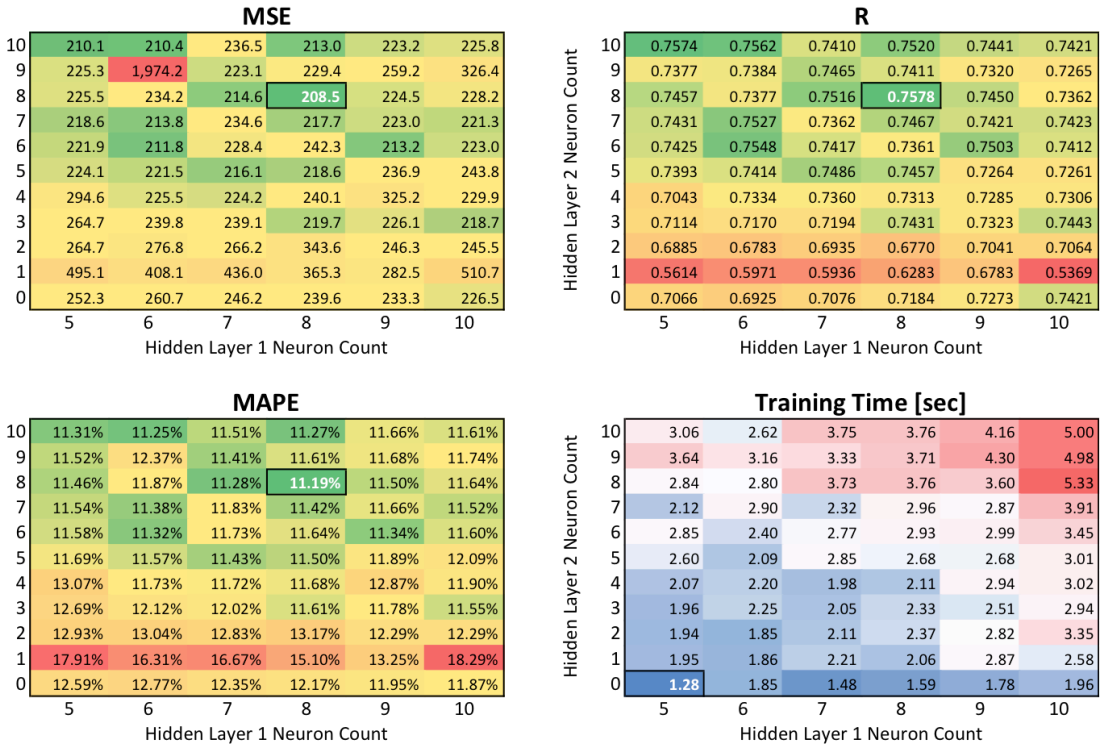


Figure 5.5: Test 5.1 – FH, Previous day average demand average test results

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 175.5 | 0.7951 | 9.8 | 10.11% | 2.79 | 10 | 5 |
| 175.9 | 0.7972 | 9.8 | 10.13% | 2.01 | 9 | 6 |
| 176.7 | 0.7910 | 9.8 | 10.05% | 13.93 | 5 | 10 |

*Table 5.1: Test 5.1 – FH, Previous day average demand results*

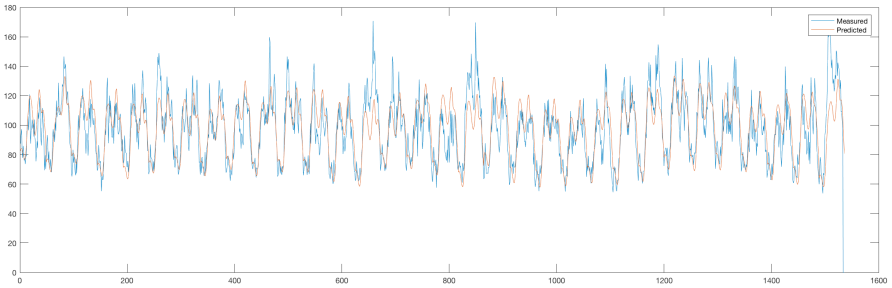


*Figure 5.6: Test 5.1 – FH, Previous day average demand, best graph*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** The network shows a decent performance. But it is clear that the unpredictability of this gird level induces some hard errors due to unforeseen spikes in demand.

### *Test 5.2 – FH, Previous 24 hour and day average demand*

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous day average demand (0~300 kW), Hour (0~23.5), Previous 24 hour demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 11 months

**Testing Duration:** 1 month


*Figure 5.7: Test 5.2 – FH, Previous 24 hour and day average demand average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 164.4 | 0.8105 | 9.6 | 9.83% | 1.21 | 9 | 6 |
| 166.8 | 0.8057 | 9.5 | 9.59% | 5.16 | 10 | 4 |
| 168.2 | 0.8013 | 9.7 | 9.86% | 1.37 | 7 | 6 |

*Table 5.2: Test 5.2 – FH, Previous 24 hour and day average demand results*

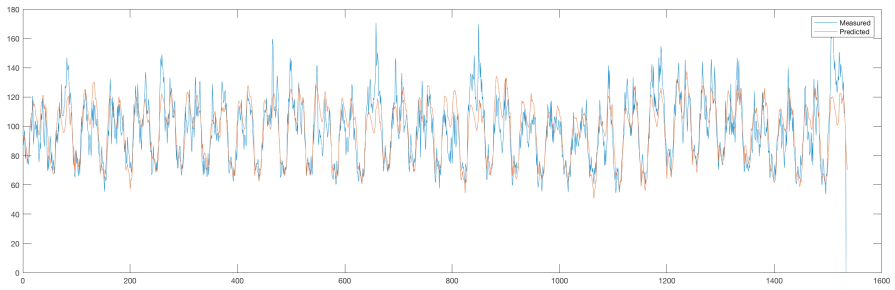


*Figure 5.8: Test 5.2 – FH, Previous 24 hour and day average demand, best plot*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** As *Figure 5.9* shows, this input selection outperforms one with just the previous day's average demand.

**MSE**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 1.63E+00 | 4.34E+00 | 6.22E+00 | -7.14E+00 | 7.74E+00 | 1.58E+00 |
| 9 | 2.05E+01 | 1.77E+03 | 1.30E+01 | -2.13E+01 | 4.54E+01 | 1.20E+02 |
| 8 | 1.26E+01 | 2.72E+01 | 2.45E+00 | -6.06E+00 | 1.26E+01 | -3.42E+01 |
| 7 | -4.15E+01 | 2.05E-01 | 3.22E+01 | 1.03E+01 | 1.50E+01 | 7.69E+00 |
| 6 | 1.13E+01 | 1.24E+00 | 1.51E+01 | 2.45E+01 | -7.05E+00 | 9.05E+00 |
| 5 | 5.28E+00 | -2.60E+01 | 2.59E+00 | 3.43E+00 | 5.38E+00 | 2.67E+01 |
| 4 | 6.60E+01 | -6.33E+00 | 5.44E+00 | 1.81E+01 | 1.11E+02 | 1.92E+01 |
| 3 | 3.87E+01 | 1.25E+01 | 2.39E+01 | -2.40E+00 | 1.16E+01 | -1.91E+01 |
| 2 | 2.71E+01 | 4.91E+01 | 1.89E+01 | 1.19E+02 | 2.36E+01 | 1.89E+01 |
| 1 | 1.77E+02 | 7.93E+01 | 1.44E+02 | 5.30E+00 | 1.22E+01 | 2.00E+02 |
| 0 | 1.92E+01 | 3.23E+01 | 1.79E+01 | 1.77E+01 | 1.59E+01 | 6.80E+00 |

Hidden Layer 1 Neuron Count

**Average =** 5.05E+01 **Median =** 1.26E+01

**R**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | -0.0025 | -0.0010 | -0.0097 | 0.0051 | -0.0073 | -0.0007 |
| 9 | -0.0230 | -0.0227 | -0.0105 | -0.0001 | -0.0187 | -0.0318 |
| 8 | -0.0038 | -0.0214 | 0.0001 | 0.0111 | -0.0066 | -0.0021 |
| 7 | 0.0006 | -0.0010 | -0.0283 | -0.0112 | -0.0137 | -0.0085 |
| 6 | -0.0109 | 0.0003 | -0.0077 | -0.0105 | 0.0072 | -0.0085 |
| 5 | -0.0009 | 0.0137 | 0.0011 | -0.0038 | 0.0004 | -0.0181 |
| 4 | -0.0254 | 0.0049 | -0.0083 | -0.0107 | -0.0182 | -0.0213 |
| 3 | -0.0188 | -0.0137 | -0.0257 | 0.0053 | -0.0139 | 0.0082 |
| 2 | -0.0305 | -0.0519 | -0.0192 | -0.0559 | -0.0317 | -0.0316 |
| 1 | -0.1099 | -0.0592 | -0.0926 | -0.0147 | -0.0024 | -0.1351 |
| 0 | -0.0158 | -0.0359 | -0.0220 | -0.0191 | -0.0165 | 0.0013 |

Hidden Layer 1 Neuron Count

**Average =** -0.0171 **Median =** -0.0108

**MAPE**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 0.50% | 0.39% | 0.66% | 0.20% | 0.56% | 0.31% |
| 9 | 0.68% | 1.56% | 0.50% | 0.42% | 0.69% | 0.88% |
| 8 | 0.40% | 1.07% | 0.28% | 0.24% | 0.49% | 0.39% |
| 7 | 0.30% | 0.39% | 1.10% | 0.52% | 0.78% | 0.47% |
| 6 | 0.54% | 0.37% | 0.69% | 0.59% | 0.19% | 0.55% |
| 5 | 0.40% | -0.24% | 0.41% | 0.40% | 0.35% | 0.90% |
| 4 | 1.73% | 0.23% | 0.49% | 0.50% | 1.82% | 0.95% |
| 3 | 1.25% | 0.60% | 0.95% | 0.28% | 0.69% | 0.20% |
| 2 | 0.99% | 1.46% | 0.81% | 1.79% | 0.90% | 0.98% |
| 1 | 4.25% | 2.08% | 3.42% | 0.71% | 0.30% | 4.71% |
| 0 | 1.03% | 1.35% | 0.88% | 0.96% | 0.90% | 0.69% |

Hidden Layer 1 Neuron Count

**Average =** 0.86% **Median =** 0.63%

**Training Time [sec]**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | -0.22 | -0.25 | 0.64 | -0.06 | -0.35 | 0.35 |
| 9 | 0.93 | 0.12 | -0.18 | 0.18 | 0.08 | 0.61 |
| 8 | 0.61 | 0.03 | 0.94 | 0.91 | -0.35 | 0.99 |
| 7 | -0.06 | 0.06 | -0.75 | 0.09 | 0.24 | -0.08 |
| 6 | 0.77 | 0.00 | 0.26 | 0.13 | -0.28 | -0.18 |
| 5 | 0.87 | -0.17 | 0.46 | -0.57 | -0.34 | -0.01 |
| 4 | 0.45 | 0.28 | -0.31 | -0.31 | 0.47 | -0.46 |
| 3 | 0.07 | 0.23 | -0.08 | 0.09 | 0.16 | -0.21 |
| 2 | 0.34 | 0.29 | 0.35 | -0.05 | 0.68 | 1.02 |
| 1 | -0.09 | -0.14 | 0.14 | 0.34 | 0.94 | 0.03 |
| 0 | -0.01 | 0.51 | 0.03 | -0.08 | -0.17 | -0.01 |

Hidden Layer 1 Neuron Count

**Average =** 0.15 **Median =** 0.08

*Figure 5.9:* Test 5.1 *and* Test 5.2 *comparison*

Red values represents a better result on Test 5.2 (with both previous day's average demand and previous 24-hour demand) and blue indicates a better result on Test 5.1 (with only previous day's average demand).

***Test 5.3 – FH, Previous 24 hour, 168 hour and day average demand***

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~300 kW), Previous 168 hour demand (0~300 kW), Previous day's average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 11 months

**Testing Duration:** 1 month

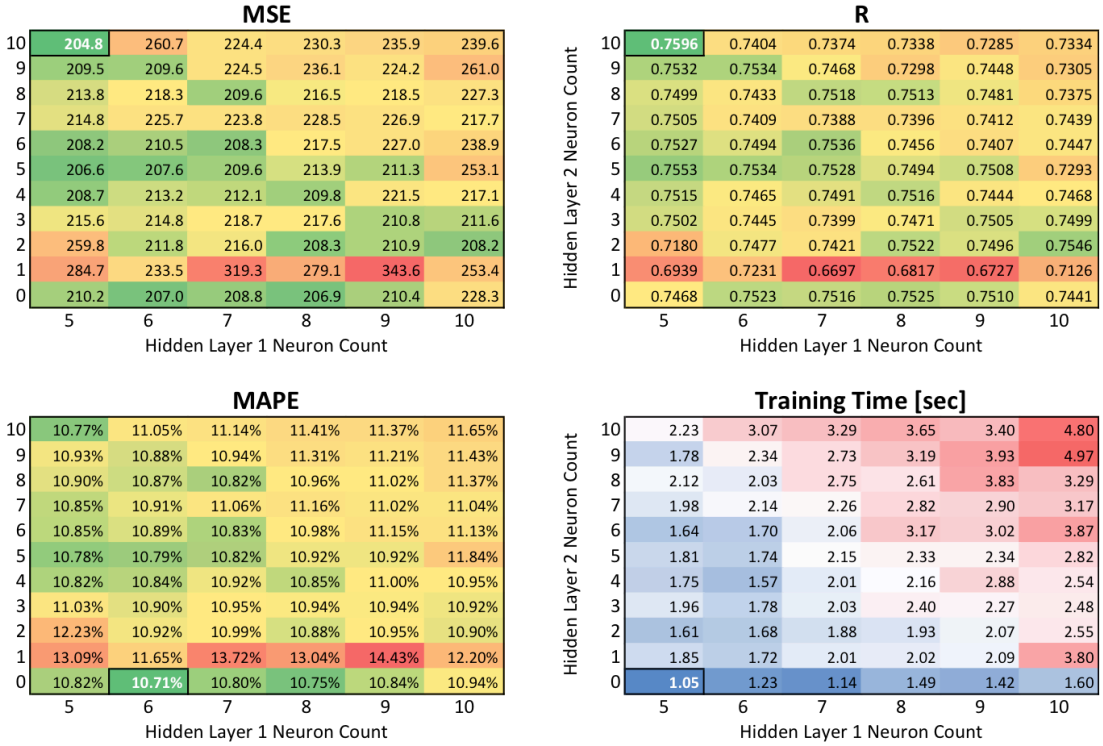


*Figure 5.10: Test 5.3 – FH, Previous 24 hour, 168 hour and day average demand average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 162.9 | 0.8082 | 9.3 | 9.42% | 5.43 | 10 | 6 |
| 165.4 | 0.8087 | 9.5 | 9.73% | 2.94 | 9 | 8 |
| 166.4 | 0.8093 | 9.5 | 9.66% | 2.83 | 10 | 5 |

*Table 5.3: Test 5.3 – FH, Previous 24 hour, 168 hour and day average demand results*

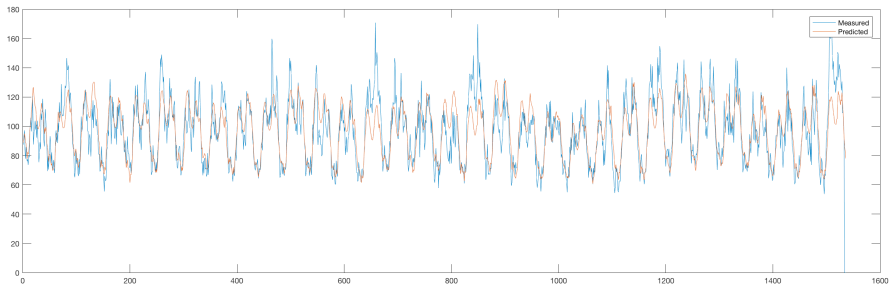


*Figure 5.11: Test 5.3 – FH, Previous 24 hour, 168 hour and day average demand, best plot*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** According to Figure 5.12 including the previous 168 (week) demand appears marginally better in average. Not enough to say to use this configuration instead of the one without. But, beyond that, it requires the user to discard the first week of the database to be used since it will not include the previous week's values to be used as inputs. Therefore, it should be less prioritized based on that fact alone.

**MSE**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 3.7 | -54.6 | 5.8 | -10.3 | -20.4 | -15.4 |
| 9 | -4.7 | -4.9 | -14.3 | 14.5 | -10.4 | -54.3 |
| 8 | -0.8 | -11.3 | 2.5 | -1.9 | -6.5 | 35.1 |
| 7 | 45.2 | -12.1 | -21.4 | -21.2 | -18.9 | -4.0 |
| 6 | 2.4 | 0.1 | 5.0 | 0.3 | -6.6 | -25.0 |
| 5 | 12.2 | 39.9 | 3.9 | 1.3 | 20.3 | -36.0 |
| 4 | 19.9 | 18.7 | 6.6 | 12.2 | -7.0 | -6.4 |
| 3 | 10.4 | 12.5 | -3.4 | 4.5 | 3.7 | 26.2 |
| 2 | -22.1 | 15.9 | 31.3 | 16.3 | 11.9 | 18.4 |
| 1 | 33.3 | 95.3 | -27.3 | 80.9 | -73.3 | 57.5 |
| 0 | 22.9 | 21.4 | 19.4 | 15.0 | 7.0 | -8.6 |

Hidden Layer 1 Neuron Count

Average = 3.79E+00 Median = 3.09E+00

**R**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 0.0003 | 0.0169 | 0.0133 | 0.0131 | 0.0229 | 0.0094 |
| 9 | 0.0074 | 0.0076 | 0.0102 | 0.0114 | 0.0059 | 0.0278 |
| 8 | -0.0003 | 0.0158 | -0.0003 | -0.0046 | 0.0035 | 0.0009 |
| 7 | -0.0080 | 0.0127 | 0.0258 | 0.0183 | 0.0147 | 0.0069 |
| 6 | 0.0007 | 0.0051 | -0.0042 | 0.0011 | 0.0024 | 0.0051 |
| 5 | -0.0150 | -0.0257 | -0.0053 | 0.0001 | -0.0248 | 0.0149 |
| 4 | -0.0219 | -0.0180 | -0.0048 | -0.0096 | 0.0022 | 0.0051 |
| 3 | -0.0200 | -0.0138 | 0.0053 | -0.0093 | -0.0043 | -0.0138 |
| 2 | 0.0010 | -0.0175 | -0.0294 | -0.0193 | -0.0138 | -0.0165 |
| 1 | -0.0225 | -0.0668 | 0.0165 | -0.0386 | 0.0079 | -0.0406 |
| 0 | -0.0244 | -0.0239 | -0.0219 | -0.0150 | -0.0072 | -0.0032 |

Hidden Layer 1 Neuron Count

Average = -0.0038 Median = -0.0001

**MAPE**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 0.04% | -0.20% | -0.29% | -0.35% | -0.27% | -0.35% |
| 9 | -0.09% | -0.08% | -0.04% | -0.12% | -0.22% | -0.57% |
| 8 | 0.16% | -0.06% | 0.18% | -0.01% | -0.01% | -0.13% |
| 7 | 0.39% | 0.08% | -0.32% | -0.26% | -0.15% | 0.01% |
| 6 | 0.19% | 0.06% | 0.21% | 0.07% | 0.00% | -0.08% |
| 5 | 0.51% | 1.02% | 0.20% | 0.18% | 0.62% | -0.65% |
| 4 | 0.52% | 0.66% | 0.31% | 0.33% | 0.06% | -0.01% |
| 3 | 0.41% | 0.62% | 0.11% | 0.39% | 0.15% | 0.43% |
| 2 | -0.29% | 0.65% | 1.03% | 0.50% | 0.45% | 0.41% |
| 1 | 0.56% | 2.58% | -0.47% | 1.36% | -1.48% | 1.38% |
| 0 | 0.74% | 0.72% | 0.67% | 0.46% | 0.21% | 0.25% |

Hidden Layer 1 Neuron Count

Average = 0.20% Median = 0.15%

**Training Time [sec]**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 1.05 | -0.20 | -0.18 | 0.16 | 1.11 | -0.15 |
| 9 | 0.94 | 0.70 | 0.78 | 0.33 | 0.29 | -0.60 |
| 8 | 0.12 | 0.75 | 0.04 | 0.24 | 0.13 | 1.05 |
| 7 | 0.21 | 0.70 | 0.81 | 0.04 | -0.27 | 0.82 |
| 6 | 0.45 | 0.70 | 0.45 | -0.37 | 0.25 | -0.24 |
| 5 | -0.07 | 0.52 | 0.24 | 0.92 | 0.67 | 0.21 |
| 4 | -0.14 | 0.36 | 0.29 | 0.26 | -0.41 | 0.95 |
| 3 | -0.07 | 0.24 | 0.10 | -0.16 | 0.08 | 0.67 |
| 2 | 0.00 | -0.12 | -0.12 | 0.48 | 0.07 | -0.23 |
| 1 | 0.20 | 0.28 | 0.06 | -0.31 | -0.15 | -1.24 |
| 0 | 0.24 | 0.11 | 0.31 | 0.18 | 0.53 | 0.36 |

Hidden Layer 1 Neuron Count

Average = 0.23 Median = 0.22

*Figure 5.12: Test 5.2 and Test 5.3 comparison*

Red values represents a better result on *Test 5.3* (with previous day's average demand, previous 24-hour and 168-hour demand) and blue indicates a better result on Test 5.2 (with both previous day's average demand and previous 24-hour demand).

### *Test 5.4 – FH, No date inputs*

**Inputs:** Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~300 kW), Previous day average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 11 months

**Testing Duration:** 1 month

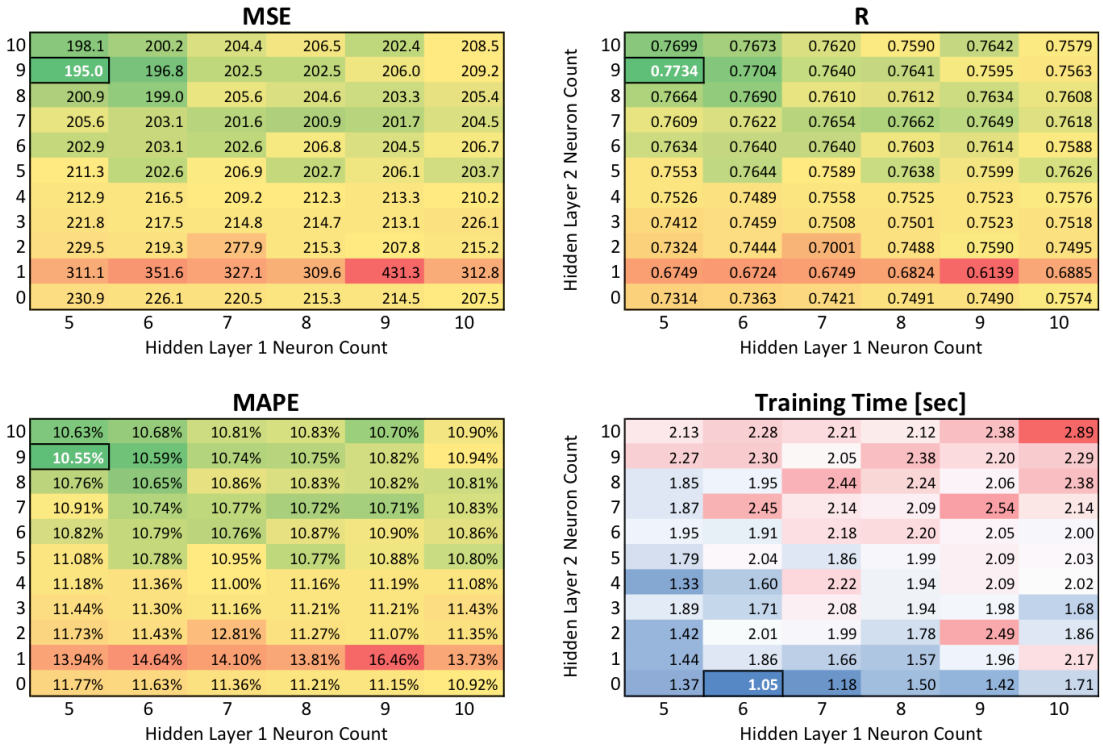


Figure 5.13: Test 5.4 – FH, No date inputs average test results

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 178.0 | 0.7942 | 9.8 | 9.99% | 1.40 | 7 | 9 |
| 178.4 | 0.7960 | 10.0 | 10.09% | 1.03 | 7 | 9 |
| 178.5 | 0.7935 | 9.8 | 9.97% | 5.27 | 7 | 10 |

*Table 5.4: Test 5.4 – FH, No date inputs results*

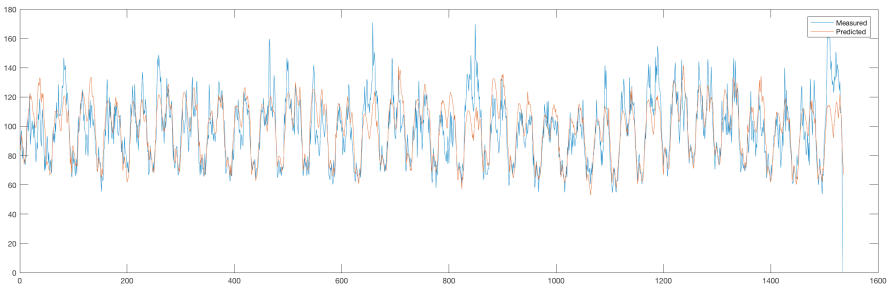


*Figure 5.14: Test 5.4 – FH, No date inputs best graphs*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** Regardless of the result, this format becomes necessary when the data pool is small, therefore not having enough different samples to properly train the date variables.

### *Test 5.5 – FH, Binary day of the week*

**Inputs:** Hour (0~23.5), Sunday (0 or 1), Monday (0 or 1), Tuesday (0 or 1), Wednesday (0 or 1), Thursday (0 or 1), Friday (0 or 1), Saturday, Previous 24 hour demand (0~300 kW), Previous day's average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 11 months

**Testing Duration:** 1 month


Figure 5.15: *Test 5.5 – FH, Binary day of the week* average test results

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 131.7 | 0.8522 | 8.5 | 8.82% | 6.85 | 7 | 8 |
| 133.3 | 0.8525 | 8.6 | 9.07% | 7.60 | 10 | 2 |
| 133.8 | 0.8508 | 8.5 | 8.87% | 3.49 | 5 | 9 |

*Table 5.5: Test 5.5 – FH, Binary day of the week results*

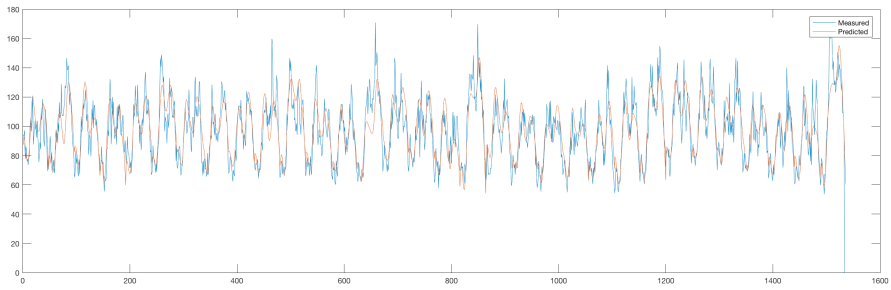


*Figure 5.16: Test 5.5 – FH, Binary day of the week, best general and close-up graphs*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** As shown on Table 5.5 when compared to Table 5.2, the binary format helps the network achieve more optimal performances. Figure 5.17 shows that it indeed performs better when the topology is more complex (more neurons on the hidden layers). This can be specially seen on the top right corner of the comparison where the topology is 10 neurons on both layers.

**MSE**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 18.6 | -6.9 | 2.3 | 22.0 | 15.5 | 33.0 |
| 9 | 3.6 | -13.7 | -4.1 | -5.5 | -6.1 | 2.1 |
| 8 | -24.3 | -11.6 | 20.1 | 25.6 | -8.6 | 21.2 |
| 7 | -10.9 | -16.9 | 9.2 | -5.9 | 3.8 | -26.5 |
| 6 | -40.5 | -33.6 | -62.0 | -73.6 | -2.8 | -5.6 |
| 5 | -8.5 | -117.4 | -25.7 | -17.6 | -19.5 | -30.7 |
| 4 | -99.0 | -67.6 | -122.0 | -39.9 | -85.8 | -32.9 |
| 3 | -95.4 | -143.5 | -37.0 | -106.3 | -47.2 | -57.6 |
| 2 | -166.4 | -152.4 | -35.2 | -39.2 | -181.9 | -79.4 |
| 1 | -388.6 | -409.7 | -384.1 | -303.1 | -341.7 | -281.4 |
| 0 | -12.2 | 10.2 | 13.3 | 24.9 | 21.4 | 29.8 |

Hidden Layer 1 Neuron Count

**Average =** -6.08E+01 **Median =** -2.19E+01

**R**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | -0.0241 | -0.0131 | -0.0146 | -0.0313 | -0.0245 | -0.0426 |
| 9 | -0.0080 | -0.0013 | -0.0131 | -0.0022 | -0.0113 | -0.0148 |
| 8 | 0.0105 | 0.0000 | -0.0292 | -0.0333 | -0.0073 | -0.0289 |
| 7 | 0.0052 | 0.0061 | -0.0209 | -0.0056 | -0.0179 | 0.0141 |
| 6 | 0.0259 | 0.0242 | 0.0326 | 0.0277 | -0.0106 | -0.0055 |
| 5 | 0.0070 | 0.0786 | 0.0223 | 0.0050 | 0.0096 | 0.0224 |
| 4 | 0.0882 | 0.0640 | 0.0806 | 0.0394 | 0.0433 | 0.0209 |
| 3 | 0.0843 | 0.0680 | 0.0331 | 0.0753 | 0.0310 | 0.0441 |
| 2 | 0.1212 | 0.1009 | 0.0199 | 0.0314 | 0.1046 | 0.0418 |
| 1 | 0.2031 | 0.2222 | 0.1883 | 0.1751 | 0.1802 | 0.1765 |
| 0 | 0.0119 | -0.0165 | -0.0215 | -0.0336 | -0.0331 | -0.0392 |

Hidden Layer 1 Neuron Count

**Average =** 0.0309 **Median =** 0.0112

**MAPE**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 0.41% | -0.10% | 0.08% | 0.58% | 0.35% | 0.93% |
| 9 | -0.01% | -0.42% | -0.04% | -0.16% | 0.07% | 0.11% |
| 8 | -0.58% | -0.27% | 0.55% | 0.61% | -0.09% | 0.49% |
| 7 | -0.39% | -0.54% | 0.17% | -0.17% | 0.09% | -0.74% |
| 6 | -1.03% | -0.90% | -1.51% | -1.60% | 0.02% | -0.20% |
| 5 | -0.38% | -2.92% | -0.76% | -0.62% | -0.58% | -0.83% |
| 4 | -2.47% | -1.81% | -3.00% | -1.17% | -1.88% | -0.70% |
| 3 | -2.29% | -3.17% | -1.05% | -2.54% | -1.07% | -1.39% |
| 2 | -3.70% | -3.41% | -0.76% | -1.29% | -4.04% | -1.73% |
| 1 | -8.05% | -8.81% | -8.49% | -6.39% | -7.01% | -5.86% |
| 0 | -0.43% | 0.09% | 0.18% | 0.50% | 0.52% | 0.73% |

Hidden Layer 1 Neuron Count

**Average =** -1.38% **Median =** -0.60%

**Training Time [sec]**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | -1.80 | -2.12 | -1.99 | -2.13 | -2.00 | -2.69 |
| 9 | -1.74 | -1.67 | -2.26 | -1.69 | -2.32 | -3.39 |
| 8 | -1.34 | -1.70 | -1.97 | -1.22 | -1.68 | -2.82 |
| 7 | -1.76 | -1.10 | -1.04 | -2.39 | -1.32 | -2.20 |
| 6 | -0.52 | -1.89 | -0.73 | -0.88 | -1.81 | -2.01 |
| 5 | -0.83 | -0.77 | -2.12 | -0.91 | -1.89 | -1.67 |
| 4 | -1.73 | -0.84 | -0.39 | -0.69 | -1.47 | -1.65 |
| 3 | -0.40 | -0.98 | -0.97 | -0.63 | -1.30 | -1.67 |
| 2 | -0.66 | -0.42 | -0.49 | -0.62 | 0.09 | -1.38 |
| 1 | -0.61 | 0.16 | -0.27 | -0.41 | -0.24 | -0.72 |
| 0 | 0.37 | -0.03 | -0.63 | -0.03 | -0.38 | -0.86 |

Hidden Layer 1 Neuron Count

**Average =** -1.25 **Median =** -1.26

*Figure 5.17: Test 5.4 and Test 5.5 comparison*

Red values represents a better result on *Test 5.5* (with binary day of the week) and blue indicates a better result on Test 5.4 (integer day of the week).

***Test 5.6 – FH, Whole day network demand***

**Inputs:** Sunday (0 or 1), Monday (0 or 1), Tuesday (0 or 1), Wednesday (0 or 1), Thursday (0 or 1), Friday (0 or 1), 48 demand inputs (0~300 kW) one for each time step of the previous day

**Output:** 48 outputs (0~300 kW) one for each time step of the day being forecast

**Time step:** 30 minutes

**Training Duration:** 11 months

**Testing Duration:** 1 month

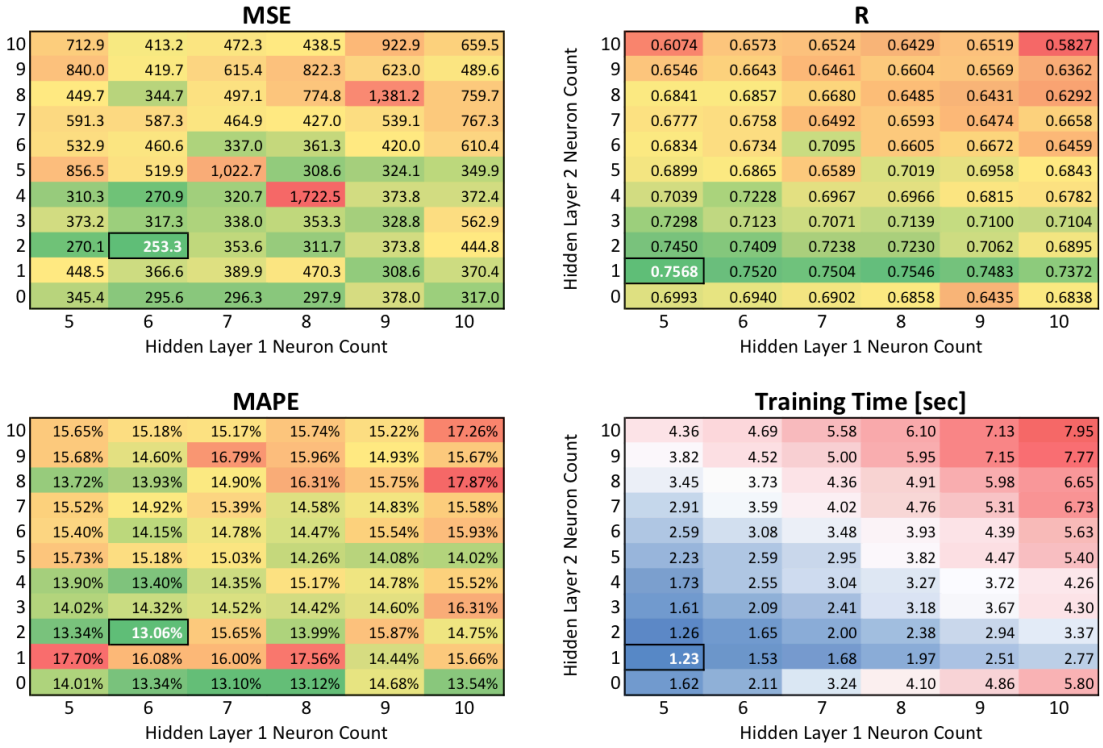


Figure 5.18: *Test 5.6 – FH, Whole day network demand* average test results

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 169.3 | 0.8026 | 9.5 | 9.77% | 1.72 | 7 | 1 |
| 169.7 | 0.8011 | 9.5 | 9.61% | 1.51 | 8 | 1 |
| 170.2 | 0.8056 | 9.9 | 10.20% | 2.18 | 8 | 2 |

*Table 5.6: Test 5.6 – FH, Whole day network demand results*

**Comments:** This configuration differs in the sense that it has multiple outputs (one for each time step of the day being forecast). It is also a very common way adopted to use neural networks for demand forecasting. Interestingly it performs better with simpler networks (small amount of neurons).

From what is seen on Figure 5.19, at least with the neuron count constraints adopted here, this configuration underperforms the single output network.

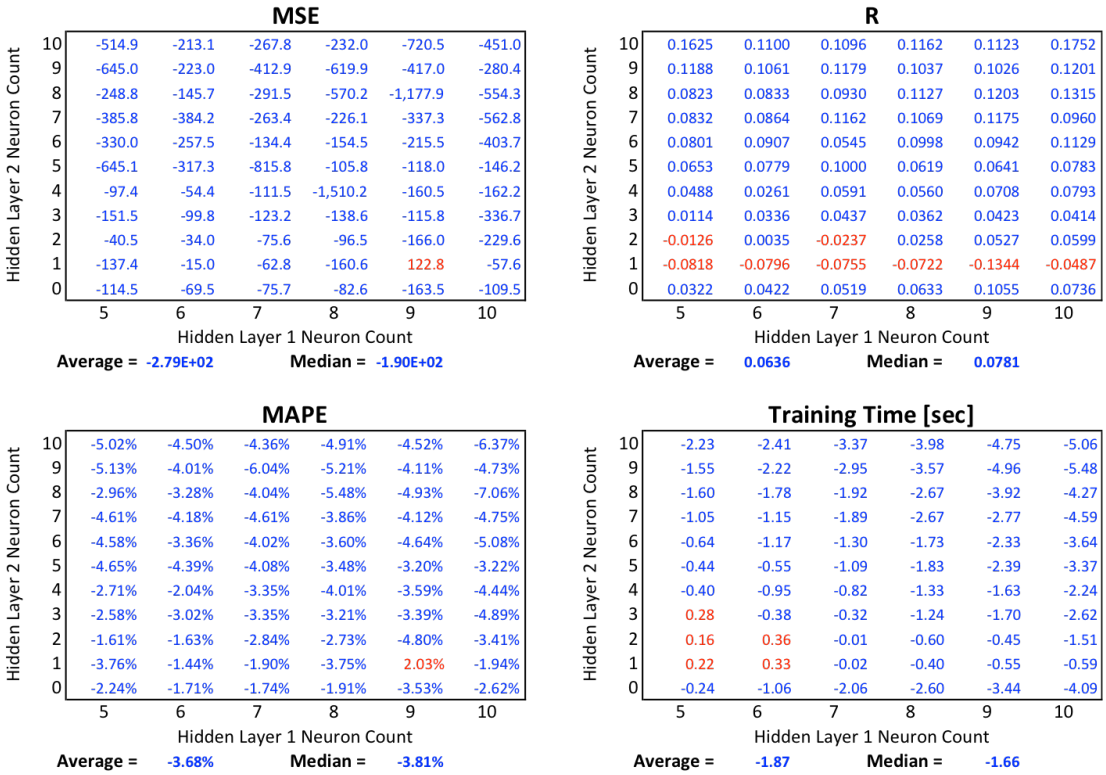


*Figure 5.19: Test 5.4 and Test 5.6 comparison*

Red values represents a better result on *Test 5.6* (whole day network) and blue indicates a better result on Test 5.4 (previous day's average demand and pervious 24-hou demand).

### 5.2.2 Dataset Build-up Time

On this section, differently from what was done in the previously, will focus on how much the time used for training will affect the results. Also, the tests performed here swept the whole data available. This means that training was performed for example on January then tested on the first week of February, then another run trained with February and was tested with the first week of March and so on and so forth.

Test weeks that contained near zero values had could not be used and were skipped. This had to be done because near zero values skyrocket the error measurements and the reading become unreasonable. This issue will be discussed on later section and can be visualized on Figure 5.32.

***Test 5.7 – FH, Previous 24 hour and day average demand, 7 days training***

**Inputs:** Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~300 kW), Previous day average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 7 days

**Testing Duration:** 7 days

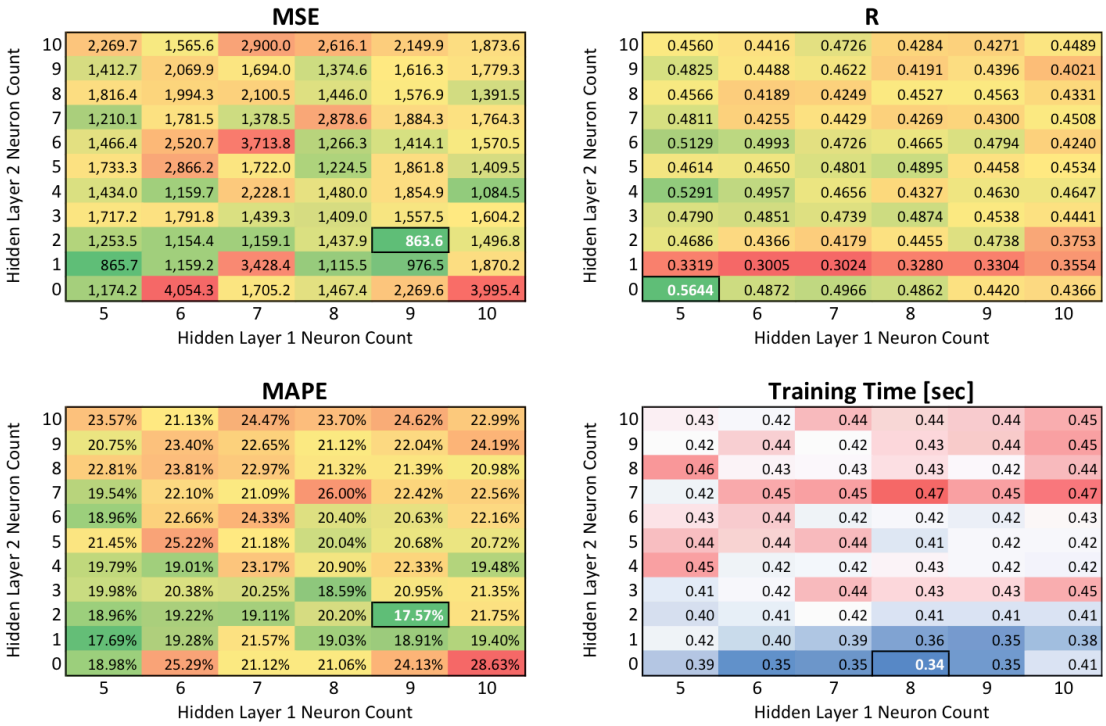


*Figure 5.20: Test 5.7 – FH, Previous 24 hour and day average demand, 7 days training average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 | Start Date |
|---|---|---|---|---|---|---|---|
| 168.6 | 0.7785 | 10.0 | 0.1023 | 0.36 | 8 | 4 | 27/07/15 |
| 177.5 | 0.7424 | 10.4 | 0.1087 | 0.32 | 7 | 2 | 13/07/15 |
| 185.1 | 0.7223 | 10.4 | 0.1052 | 0.27 | 7 | 0 | 13/07/15 |

*Table 5.7: Test 5.7 – FH, Previous 24 hour and day average demand, 7 days training best results*

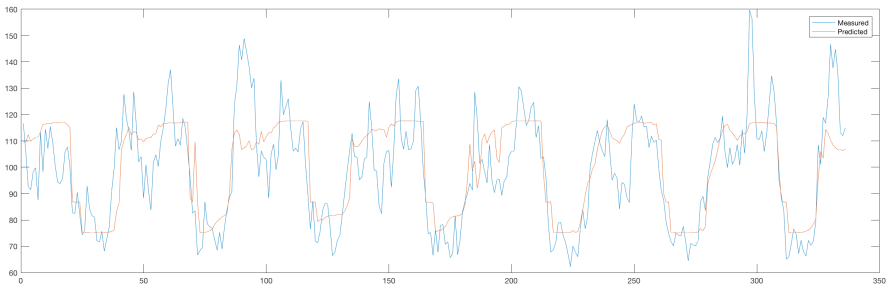


*Figure 5.21: Test 5.7 – FH, Previous 24 hour and day average demand, 7 days training best graph*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** With only 7 days worth of training the results is quite bad as expected. It is not a complete waste, but the error values are indeed high.

***Test 5.8 – FH, Previous 24 hour and day average demand, 30 days training***

**Inputs:** Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~300 kW), Previous day average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 30 days

**Testing Duration:** 7 days

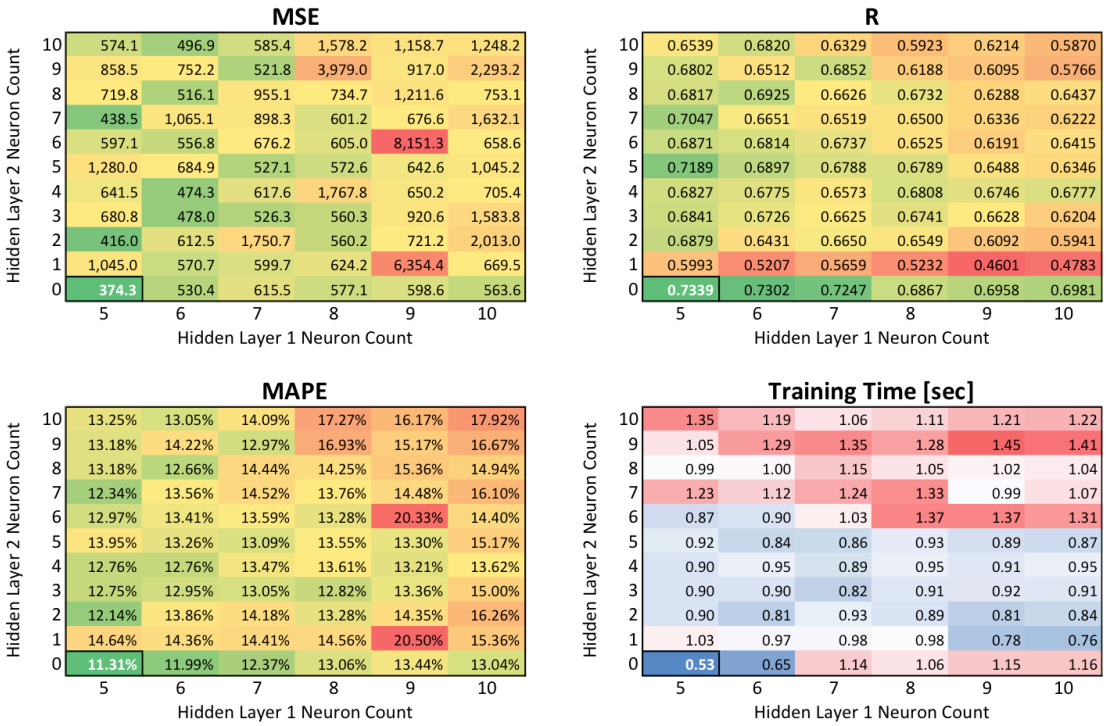


*Figure 5.22: Test 5.8 – FH, Previous 24 hour and day average demand, 30 days training average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 | Start Date |
|---|---|---|---|---|---|---|---|
| 121.3 | 0.8451 | 8.4 | 0.0839 | 1.02 | 7 | 10 | 05/07/15 |
| 124.7 | 0.8192 | 9.0 | 0.0923 | 1.15 | 10 | 0 | 29/06/15 |
| 126.9 | 0.8159 | 8.9 | 0.0910 | 1.32 | 6 | 3 | 29/06/15 |

*Table 5.8: Test 5.8 – FH, Previous 24 hour and day average demand, 30 days training best results*

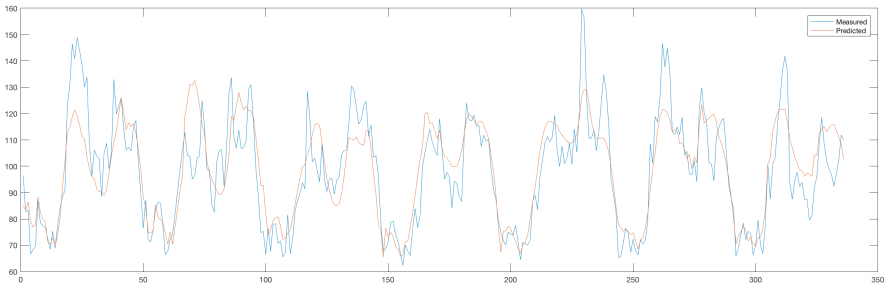


*Figure 5.23: Test 5.8 – FH, Previous 24 hour and day average demand, 30 days training best graph*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** Interestingly, the binary days of the week configuration performed worse with less data available, therefore it might be interesting to change the strategy depending on how much data is available.

***Test 5.9 – FH, Previous 24 hour and day average demand, 90 days training***

**Inputs:** Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~300 kW), Previous day average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 90 days

**Testing Duration:** 7 days

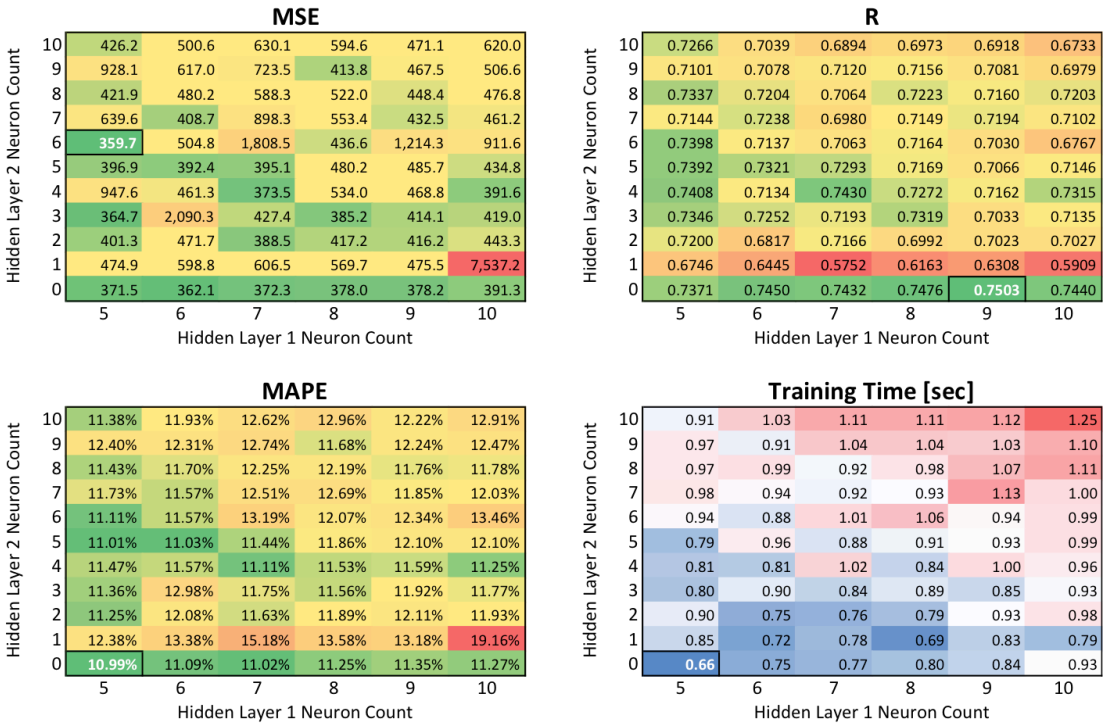


*Figure 5.24: Test 5.9 – FH, Previous 24 hour and day average demand, 90 days training average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 | Start Date |
|---|---|---|---|---|---|---|---|
| 105.1 | 0.8551 | 8.1 | 0.0843 | 0.67 | 6 | 5 | 13/04/15 |
| 105.8 | 0.8813 | 7.9 | 0.0805 | 0.60 | 5 | 8 | 21/05/15 |
| 107.6 | 0.8545 | 8.1 | 0.0831 | 0.53 | 8 | 3 | 05/05/15 |

*Table 5.9: Test 5.9 – FH, Previous 24 hour and day average demand, 90 days training best results*

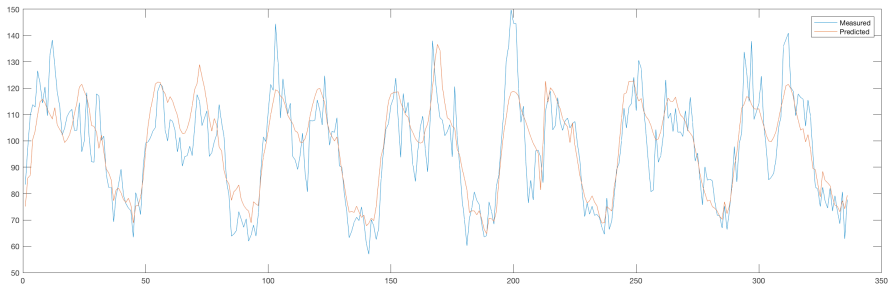


*Figure 5.25: Test 5.9 – FH, Previous 24 hour and day average demand, 90 days training, best graph*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** As expected more data means more accuracy as the comparison on Figure 5.26.

**MSE**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 147.9 | -3.7 | -44.7 | 983.6 | 687.6 | 628.2 |
| 9 | -69.6 | 135.1 | -201.6 | 3,565.2 | 449.4 | 1,786.6 |
| 8 | 297.9 | 35.9 | 366.8 | 212.6 | 763.2 | 276.3 |
| 7 | -201.1 | 656.4 | 0.0 | 47.8 | 244.1 | 1,170.9 |
| 6 | 237.3 | 52.0 | -1,132.3 | 168.4 | 6,937.0 | -253.0 |
| 5 | 883.1 | 292.5 | 132.0 | 92.4 | 156.9 | 610.4 |
| 4 | -306.1 | 13.0 | 244.1 | 1,233.9 | 181.4 | 313.8 |
| 3 | 316.0 | -1,612.3 | 98.9 | 175.1 | 506.4 | 1,164.8 |
| 2 | 14.6 | 140.8 | 1,362.2 | 143.0 | 305.0 | 1,569.7 |
| 1 | 570.2 | -28.1 | -6.8 | 54.5 | 5,879.0 | -6,867.7 |
| 0 | 2.8 | 168.4 | 243.2 | 199.1 | 220.5 | 172.4 |

Hidden Layer 1 Neuron Count

**Average =** 4.03E+02 **Median =** 1.90E+02

**R**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | -0.0727 | -0.0219 | -0.0564 | -0.1050 | -0.0704 | -0.0863 |
| 9 | -0.0300 | -0.0566 | -0.0268 | -0.0968 | -0.0987 | -0.1213 |
| 8 | -0.0521 | -0.0280 | -0.0438 | -0.0491 | -0.0871 | -0.0766 |
| 7 | -0.0097 | -0.0587 | -0.0461 | -0.0649 | -0.0858 | -0.0880 |
| 6 | -0.0527 | -0.0324 | -0.0327 | -0.0639 | -0.0839 | -0.0352 |
| 5 | -0.0202 | -0.0424 | -0.0505 | -0.0379 | -0.0578 | -0.0800 |
| 4 | -0.0581 | -0.0359 | -0.0858 | -0.0464 | -0.0416 | -0.0538 |
| 3 | -0.0505 | -0.0526 | -0.0569 | -0.0578 | -0.0405 | -0.0930 |
| 2 | -0.0321 | -0.0386 | -0.0516 | -0.0443 | -0.0931 | -0.1086 |
| 1 | -0.0752 | -0.1238 | -0.0093 | -0.0931 | -0.1706 | -0.1127 |
| 0 | -0.0033 | -0.0148 | -0.0185 | -0.0609 | -0.0545 | -0.0459 |

Hidden Layer 1 Neuron Count

**Average =** -0.0598 **Median =** -0.0541

**MAPE**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 1.87% | 1.12% | 1.47% | 4.31% | 3.95% | 5.01% |
| 9 | 0.78% | 1.91% | 0.24% | 5.25% | 2.93% | 4.20% |
| 8 | 1.75% | 0.97% | 2.20% | 2.06% | 3.60% | 3.16% |
| 7 | 0.60% | 1.99% | 2.01% | 1.07% | 2.63% | 4.08% |
| 6 | 1.86% | 1.84% | 0.40% | 1.21% | 7.99% | 0.93% |
| 5 | 2.94% | 2.23% | 1.65% | 1.69% | 1.20% | 3.08% |
| 4 | 1.29% | 1.19% | 2.37% | 2.08% | 1.62% | 2.37% |
| 3 | 1.39% | -0.02% | 1.30% | 1.26% | 1.44% | 3.24% |
| 2 | 0.90% | 1.79% | 2.55% | 1.39% | 2.24% | 4.33% |
| 1 | 2.26% | 0.97% | -0.77% | 0.99% | 7.33% | -3.80% |
| 0 | 0.32% | 0.90% | 1.35% | 1.81% | 2.09% | 1.78% |

Hidden Layer 1 Neuron Count

**Average =** 2.03% **Median =** 1.80%

**Training Time [sec]**

| Hidden Layer 2 Neuron Count | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 10 | 0.43 | 0.15 | -0.04 | 0.00 | 0.09 | -0.03 |
| 9 | 0.09 | 0.38 | 0.31 | 0.24 | 0.42 | 0.31 |
| 8 | 0.02 | 0.00 | 0.23 | 0.07 | -0.05 | -0.07 |
| 7 | 0.26 | 0.19 | 0.33 | 0.40 | -0.14 | 0.07 |
| 6 | -0.07 | 0.02 | 0.02 | 0.31 | 0.42 | 0.32 |
| 5 | 0.13 | -0.12 | -0.02 | 0.02 | -0.05 | -0.12 |
| 4 | 0.09 | 0.13 | -0.13 | 0.11 | -0.09 | -0.01 |
| 3 | 0.10 | 0.00 | -0.02 | 0.02 | 0.06 | -0.02 |
| 2 | 0.00 | 0.06 | 0.17 | 0.10 | -0.12 | -0.14 |
| 1 | 0.17 | 0.25 | 0.20 | 0.30 | -0.05 | -0.03 |
| 0 | -0.13 | -0.11 | 0.38 | 0.26 | 0.31 | 0.23 |

Hidden Layer 1 Neuron Count

**Average =** 0.10 **Median =** 0.07

*Figure 5.26: Test 5.8 and Test 5.9 comparison*

Red values represents a better result on Test 5.9 (60 days of training) and blue indicates a better result on Test 5.8 (30 days of training).

***Test 5.10 – FH, Previous 24 hour and day average demand, 180 days training***

**Inputs:** Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~300 kW), Previous day average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 180 days

**Testing Duration:** 7 days

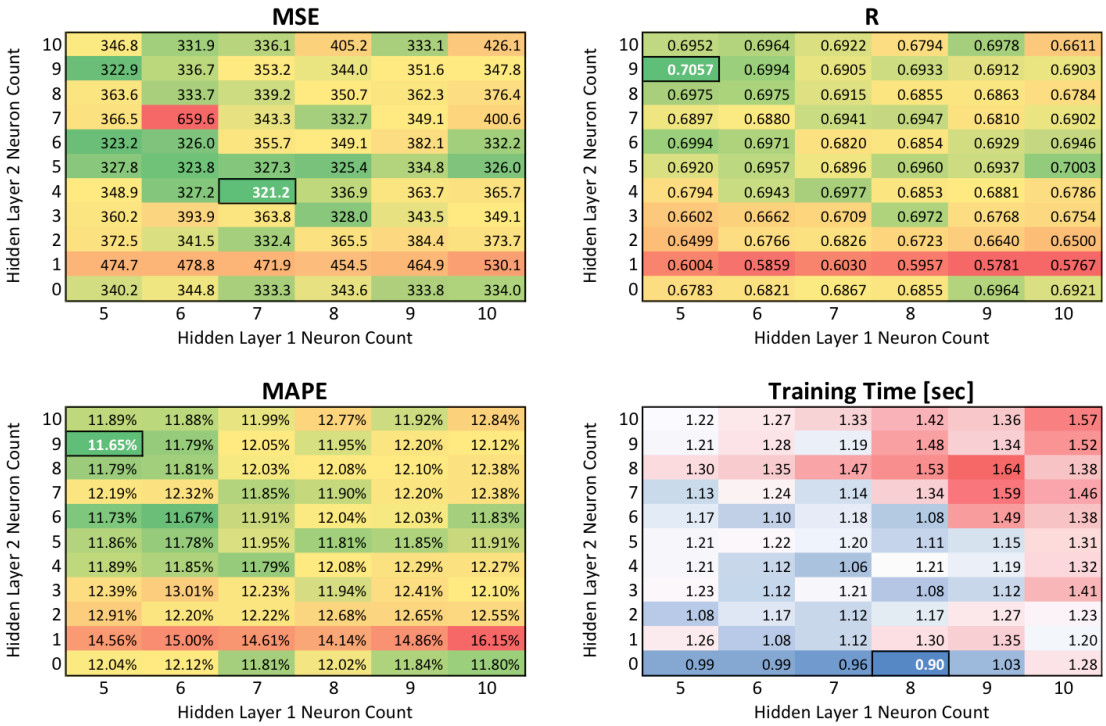


*Figure 5.27: Test 5.10 – FH, Previous 24 hour and day average demand, 180 days training average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 | Start Date |
|---|---|---|---|---|---|---|---|
| 102.4 | 0.8610 | 7.8 | 0.0817 | 1.79 | 8 | 8 | 13/01/15 |
| 104.8 | 0.8518 | 7.8 | 0.0809 | 0.78 | 7 | 2 | 13/01/15 |
| 105.6 | 0.8561 | 8.0 | 0.0827 | 1.17 | 10 | 0 | 13/01/15 |

*Table 5.10: Test 5.10 – FH, Previous 24 hour and day average demand, 180 days training best results*

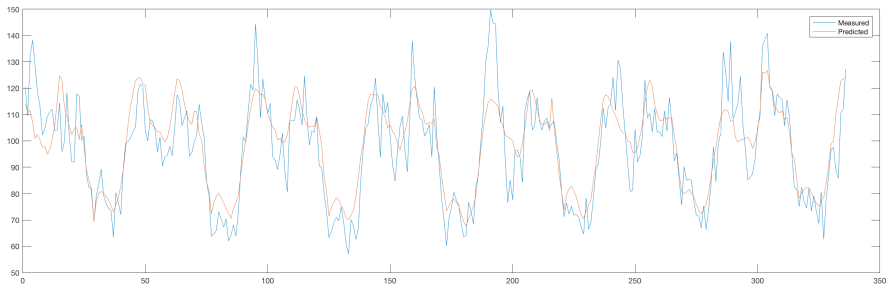


*Figure 5.28: Test 5.10 – FH, Previous 24 hour and day average demand, 180 days training, best graph*

*X-axis = Time Step (30 min)(kW), Y-axis = Demand (kW)*

**Comments:** As shown on Figure 5.29 a larger training set reduced MSE, which is the target variable to be minimized by the neural network. But the results from MAPE and R were not so cut and dry. They both slightly weight towards less days, on the case of MAPE this holds true for simples networks (lower left side). It is likely that since the database had to be cut differently between both training sizes that best results could be more easily found on the early part of the dataset, which the 180 days could not use for testing purposes.

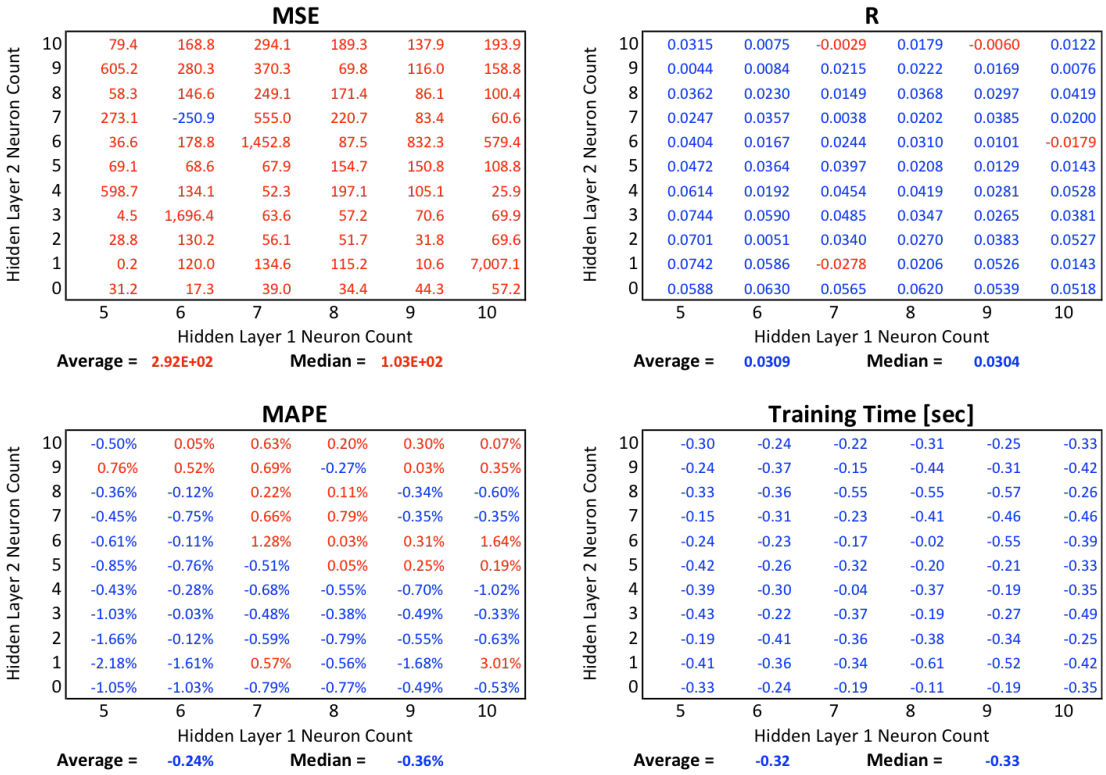


*Figure 5.29: Test 5.9 and Test 5.10 comparison*

Red values represents a better result on Test 5.10 (180 days of training) and blue indicates a better result on Test 5.9 (60 days of training).

### 5.2.3 Performance Distribution

From the information found on the Input Strategies and Dataset Build-up Time sections a new exploration took place to investigate the performance of the network trought time. This has the goal of finding where the errors are most concentrated at on a day-by-day and year wide basis.

***Test 5.11 – FH, Previous 24 hour and day average demand, 30 days training***

**Inputs:** Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~300 kW), Previous day average demand (0~300 kW)

**Time step:** 30 minutes

**Training Duration:** 30 days

**Testing Duration:** 7 days

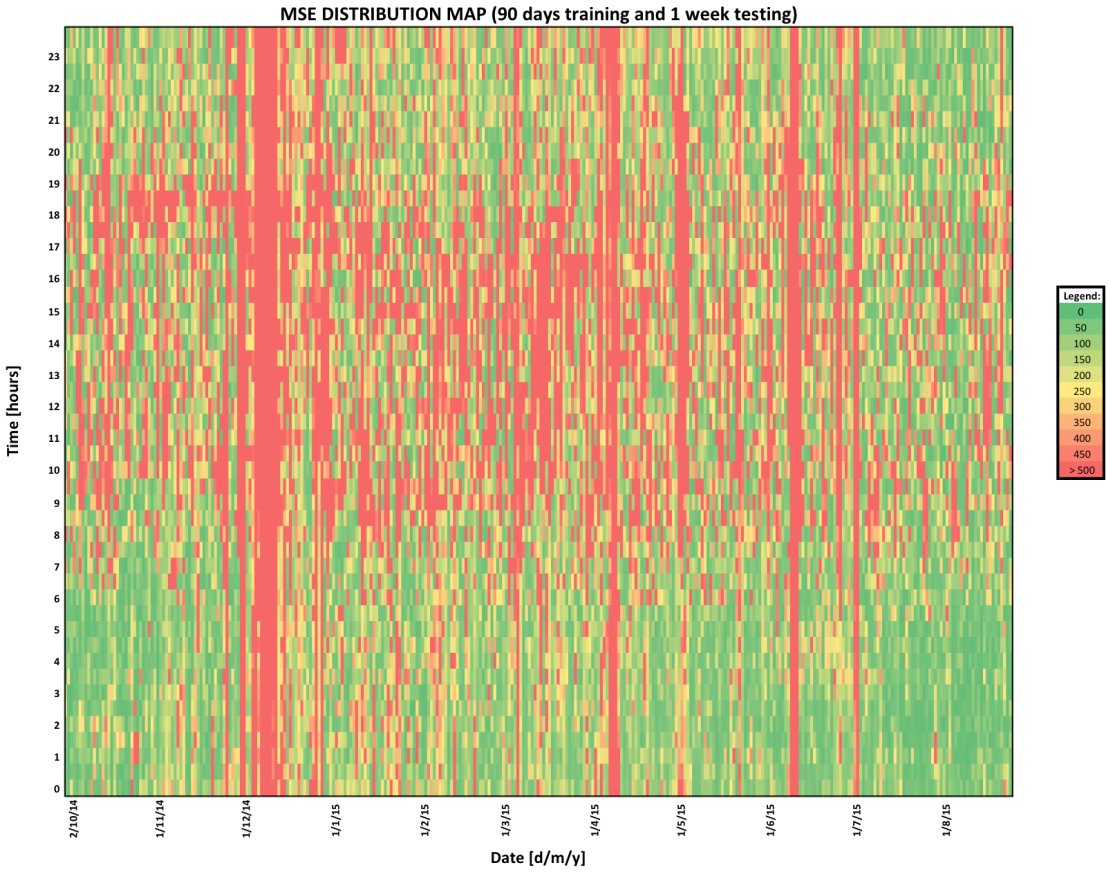


*Figure 5.30: Test 5.11 – FH, Previous 24 hour and day average demand, 30 days training, MSE Distribution Map*

*X-axis=Date of year (d/m/y), Y-axis = Time of day (hour)*

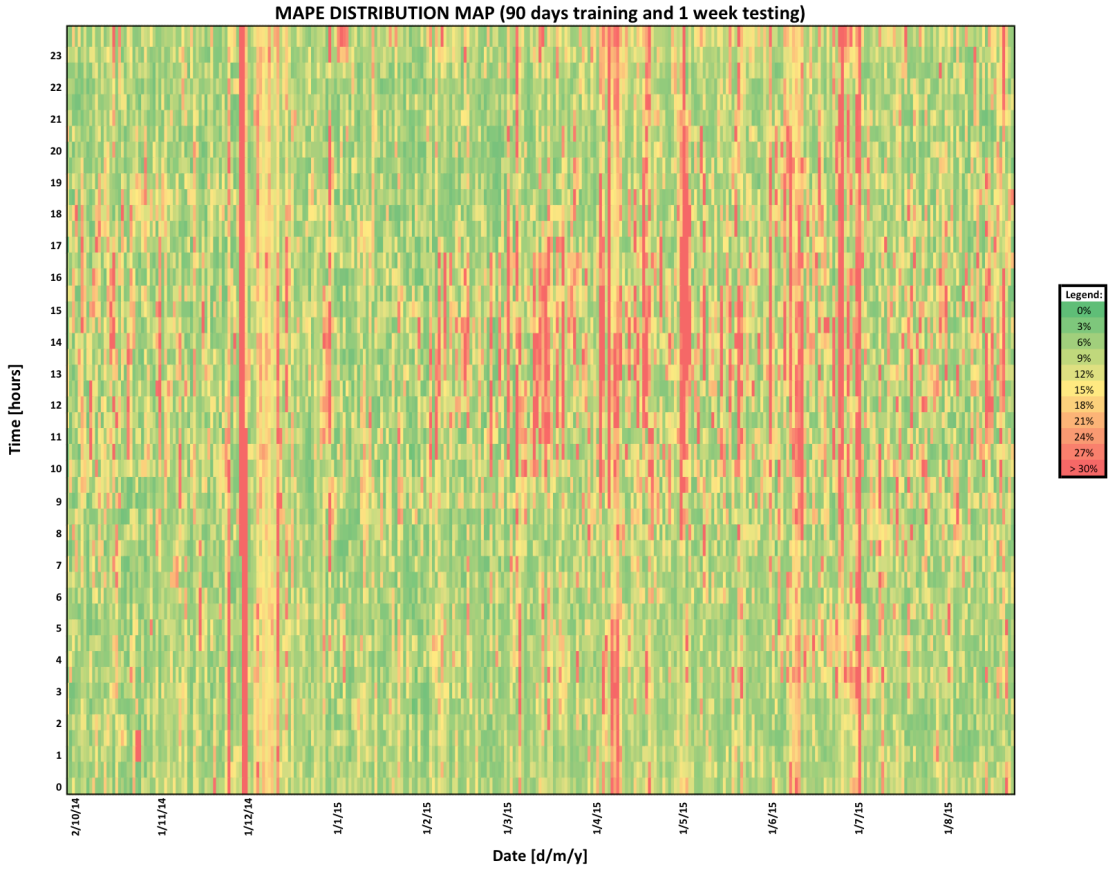


*Figure 5.31: Test 5.11 – FH, Previous 24 hour and day average demand, 30 days training, MAPE Distribution Map*

*X-axis=Date of year (d/m/y), Y-axis = Time of day (hour)*

**Comments:** Both Figure 5.30 and Figure 5.31 appear to have a generally similar appearance. Errors are more aggravated a little before noon and the earlier part of the afternoon. These errors are very likely due to demand spikes on that time period. Some individual days perform very poorly, especially ones with near zero test values that will be better discussed on the Overview section.

## 5.3 Overview

The following table contains the best run of each of the test batteries of the neural networks during the Input Strategies section.

| Test # / Title | MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|---|
| Test 5.1 – FH, Previous day average demand | 175.5 | 0.7951 | 9.8 | 10.11% | 2.79 | 10 | 5 |
| Test 5.2 – FH, Previous 24 hour and day average demand | 164.4 | 0.8105 | 9.6 | 9.83% | 1.21 | 9 | 6 |
| *Test 5.3 – FH, Previous 24 hour, 168 hour and day average demand* | 162.9 | 0.8082 | 9.3 | 9.42% | 5.43 | 10 | 6 |
| Test 5.4 – FH, No date inputs | 178.0 | 0.7942 | 9.8 | 9.99% | 1.40 | 7 | 9 |
| *Test 5.5 – FH, Binary day of the week* | 131.7 | 0.8522 | 8.5 | 8.82% | 6.85 | 7 | 8 |
| *Test 5.6 – FH, Whole day network demand* | 169.3 | 0.8026 | 9.5 | 9.77% | 1.72 | 7 | 1 |

*Table 5.11: Findhorn, inputs strategies best test results overview*

Using a demand reading from the previous week entails having to discard the whole previous week from the database since it will not include that input and only serve to feed future samples' inputs. Therefore, it delays, or weakens, the database slightly. Not using it is preferable when the results are similar.

The following table contains the best run of each of the test batteries of the neural networks during the Dataset Build-up Time.

| Test # / Title | MSE | R | MAE | MAPE | Time [sec] | L 1 | L 2 | Start Date |
|---|---|---|---|---|---|---|---|---|
| Test 5.7 – FH, Previous 24 hour and day average demand, 7 days training | 168.6 | 0.7785 | 10.0 | 0.1023 | 0.36 | 8 | 4 | 27/07/15 |
| Test 5.8 – FH, Previous 24 hour and day average demand, 30 days training | 121.3 | 0.8451 | 8.4 | 0.0839 | 1.02 | 7 | 10 | 05/07/15 |
| Test 5.9 – FH, Previous 24 hour and day average demand, 90 days training | 105.1 | 0.8551 | 8.1 | 0.0843 | 0.67 | 6 | 5 | 13/04/15 |
| Test 5.10 – FH, Previous 24 hour and day average demand, 180 days training | 102.4 | 0.8610 | 7.8 | 0.0817 | 1.79 | 8 | 8 | 13/01/15 |

*Table 5.12: Findhorn, dataset build-up best test results overview*

The same strategy used on *Test 5.5*, which was the binary day of the week, underperformed compared to using an integer to denote it. Therefore, these tests all kept it as an integer. Keeping in mind that distinct sample sizes behave differently depending on the input selection.

Since the database available was limited to one year worth of samples the results are expected to have some form of bias. Some times of the year are easier to predict than others. This becomes evident on Table 5.7, Table 5.8, Table 5.9 and Table 5.10. The best results for each of the different training sizes focus their start date on small portions of the year. This induces that the best weeks to test the algorithms are near one another. The database limitation forces a different range of time ranges for each test setup, therefore introducing the bias. But still, the results

improved quite consistently. Even with reservation are to be had on the comparison between 90 days and 180 days of training.

One big issue that arose during this phase of testing, which took snippets of the whole dataset was with using samples with near zero values on the test data pool. Those cases had to be removed since a seemingly random near zero value skyrocketed the error measurements, creating a huge distortion. Figure 5.32 greatly exemplifies this, on the final time steps of the test week the measured demand drops to near zero values while the predicted value stays "normal". The division between those numbers creates high error and low performance results.

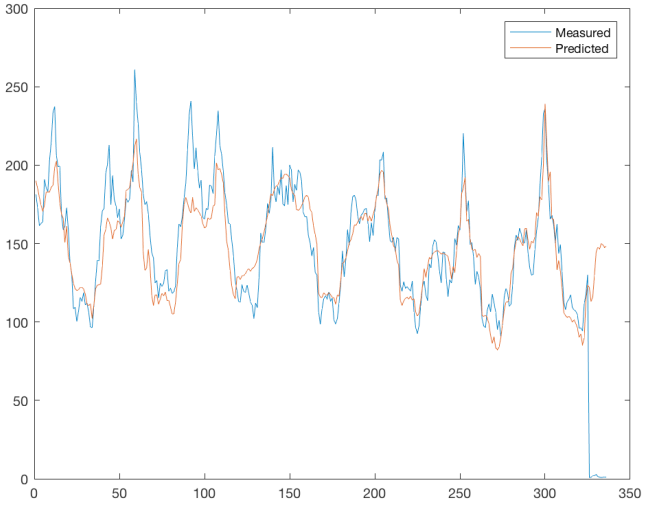


*Figure 5.32: Bad data sample returning skewed results example.*

Table 5.12 show the single best runs of each test battery. It shows that potentially even a low 7-day training can be used. But, care should be given to their reliability since a total of 3,300 different networks were tested in each individual test.

# 6 Application at House Level

This section encompasses “hard” results of the testing done with House level datasets.

The objective here is to explore input selections on a neural network to forecast a single house’s demand. This level is usually considered very hard to predict, but number are not often shown. Therefore, having the actual error measurements aids in assessing the potential use of forecasting on the single residence level.

## 6.1 Introduction

Findhorn the ORIGIN project also produced demand datasets for single residences. One of those will be used in this section to test at the house level forecasting using neural networks.

This level of forecasting is often deemed, if not impossible, very hard to predict. This is due to the unpredictable nature of averaged human behaviour. With a larger number of consumers this problem is greatly alleviated, but looking at one single person or family problems may arise. Therefore, bad results are expected.

The database available for this single Findhorn house demand ranged from January 2015 to October 2015.

## 6.2 Test Results

### 6.2.1 Input Strategies

This sub section is focused on different input sets and how they affect the end result. Firstly, the histogram for each both the training and testing datasets will be analysed.

The original database spanned from the 1st of January 2015 to 5th of October 2015. But, due to null values in the database, signifying that the equipment was turned off or some kind of failure occurred. Regardless, zero values must be removed if a large amount of them is present, otherwise they might disrupt the calculations. Therefore, the histograms shown bellow have the zero demand value removed from both the demand, the day of the year and outside temperature pools. The same had to be done on the outside temperature data set.

**Training dataset histogram:**

Figure 6.1 shows the case study's house day of year histogram for the training data set, or the data pool used by the network to train its weights in order to reach a good generalization of the rules driving the demand. The values range from 1 to 253, or from January 2015 to September 2015 with many missing samples along the way. Again, this is due to times where the wattmeter was turned off, communication errors or the power to the house was shut down. These points were detected for having a zero/null value and had to be removed due to the large amount of them being present.

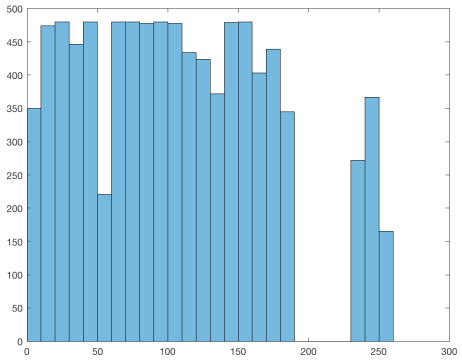


*Figure 6.1: House training dataset day of year histogram*

Figure 6.2 shows the case study's house demand histogram for training data set, its unit is kilo-watt (kW). The values encountered ranged from 0 to 8 kW.

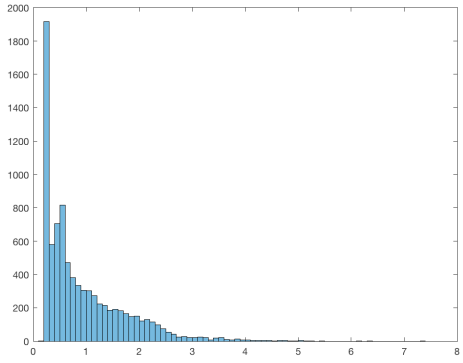


*Figure 6.2: House training dataset demand histogram*

As previously stated all entries with a null value had to be removed. But, low, non-zero values were not only accepted but also found in large amounts. Those are low consumption periods that can very plausibly occur in a single-family house, unlike what is expected from the macrogrid and community levels. Although, this excess of low values (around 1900 samples just on the first bucket of the histogram) can be a problem since the neural network can attach itself too much on them and develop a bias.

Training demand samples with non-zero values have a total of 8,439 entries.

Lastly, Figure 6.3 shows the case study's house outside temperature histogram for training data set, its unit is degrees Celsius (°C). The values encountered ranged form -5 to +30 °C. The curve looks quite good, within what can be expected as a balanced sample space.

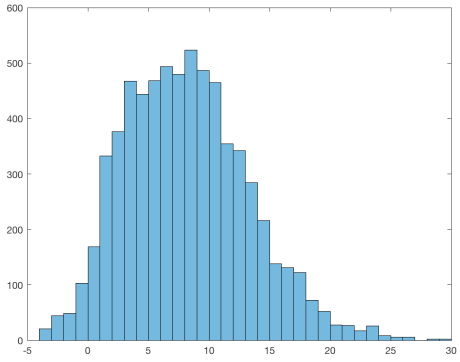


*Figure 6.3: House training dataset outside temperature histogram*

The outside temperature data pool

also contained many null values, which were considered as invalid samples. Therefore, when using both demand and outside temperature the resulting dataset being much smaller.

Training outside temperature samples with non-zero values that also have non-zero demands total 6,755 entries.

**Testing dataset histogram:**

Figure 6.4 shows the case study's house day of year histogram for the test data set, or the data pool used after the network is trained to evaluate the forecasting power of the network. The values range from 254 to 278, or from September 2015 to October 2015. Differently from the training dataset, this pool was selected in a ways to take almost a whole month with full consecutive samples, therefore there are no holes in it.

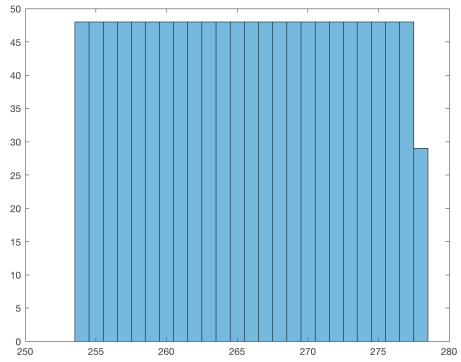


*Figure 6.4: House test dataset day of year histogram*

Figure 6.5 shows the case study's house demand histogram for test data set, its unit is kilo-watt (kW). The values encountered ranged from 0 to 3.5 kW. No null values were present.

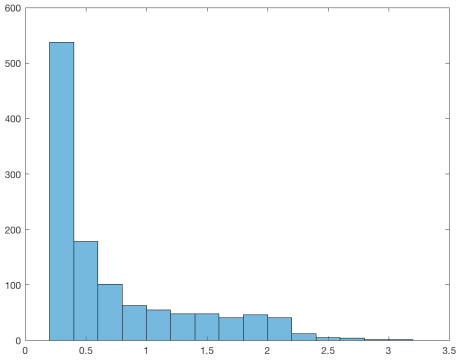


*Figure 6.5: House test dataset demand histogram*

It suffers the same problem with the training dataset, which is the over abundance of low, non-zero values, adding up to nearly 550 on the first bucket.

Test demand samples with a non-zero value have a total of 1,181 entries.

Finally, Figure 6.6 shows the case study's house outside temperature histogram for the test data set, its unit is degrees Celsius (°C). The values encountered ranged form +2 to +22 °C. The curve looks decent, fairly balanced.

Again, the outside temperature test data pool was specifically selected to contain no null values, which could greatly distort test results.

Test outside temperature samples with non-zero values that also have non-zero demands total 1,181 entries.

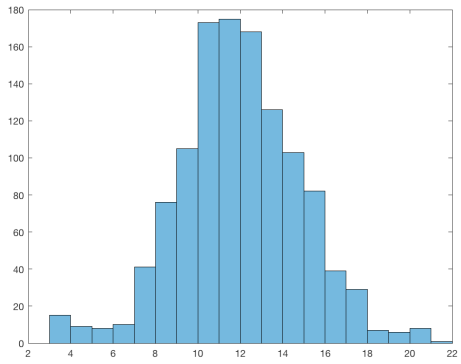


*Figure 6.6: House test dataset outside temperature histogram*

***Test 6.1 – House, Previous 24 hour demand***

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~8 kW)

**Time step:** 30 minutes

**Training Duration:** equivalent of 176 days (non-consecutive)

**Testing Duration:** 24.5 days

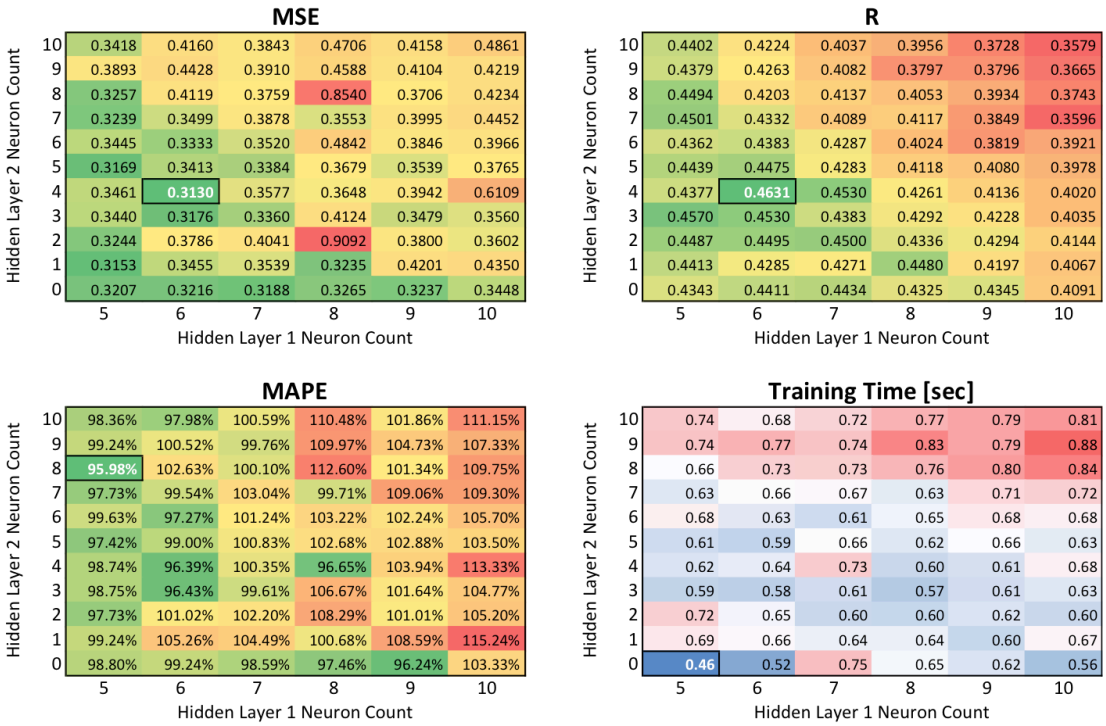


*Figure 6.7: Test 6.1 – House, Previous 24 hour demand average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 0.2688 | 0.5454 | 0.4289 | 92.91% | 1.03 | 9 | 2 |
| 0.2689 | 0.5317 | 0.4168 | 84.64% | 0.71 | 6 | 4 |
| 0.2692 | 0.5241 | 0.4130 | 82.73% | 0.60 | 8 | 8 |

*Table 6.1: Test 6.1 – House, Previous 24 hour demand best results*

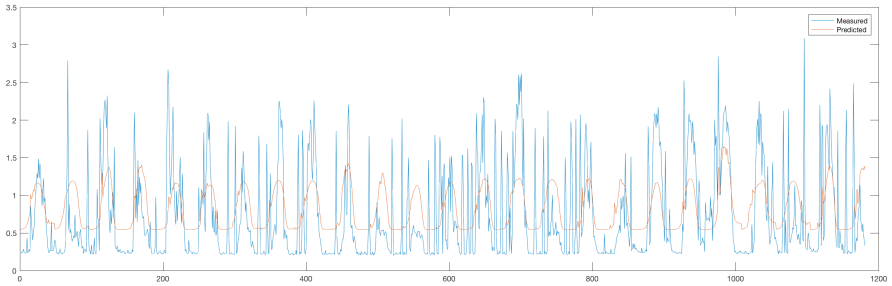


*Figure 6.8: Test 6.1 – House, Previous 24 hour demand best plot*

**Comments:** Very poor results. No capability of dealing with neither low demand nor demand spikes.

***Test 6.2 – House, Previous 24 hour and previous day's average demand***

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~8 kW), Previous day's average demand (0~8 kW)

**Time step:** 30 minutes

**Training Duration:** equivalent of 176 days (non-consecutive)

**Testing Duration:** 24.5 days

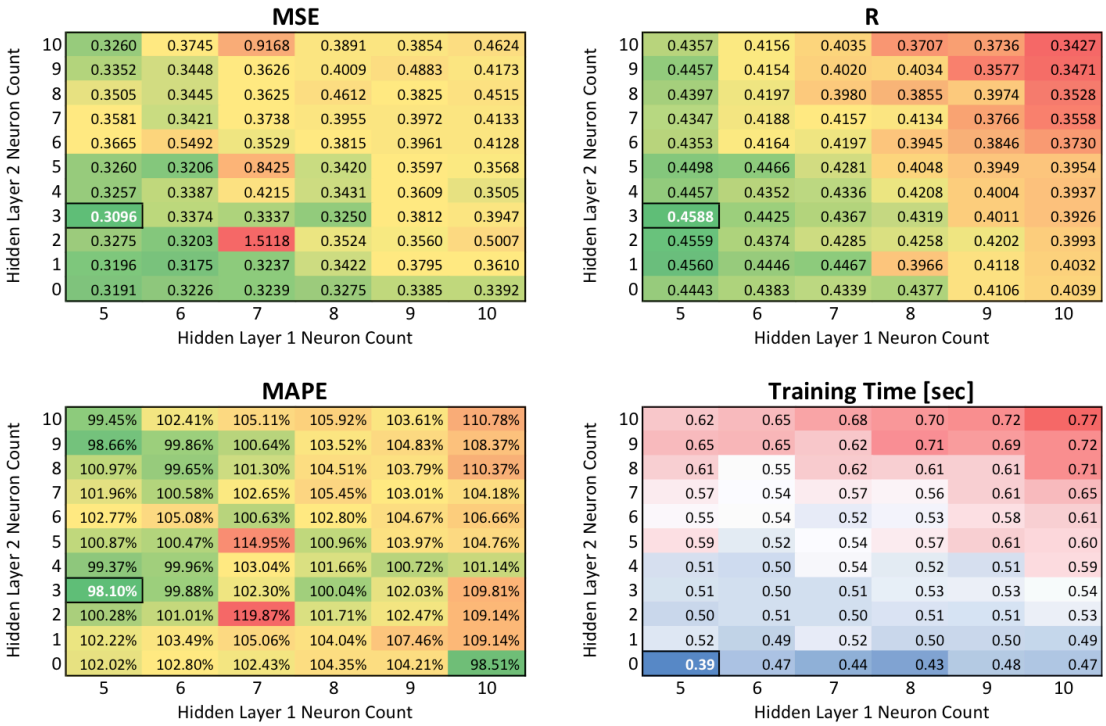


*Figure 6.9: Test 6.2 – House, Previous 24 hour and previous day's average demand average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 0.2710 | 0.5255 | 0.4159 | 87.75% | 0.45 | 8 | 7 |
| 0.2724 | 0.5241 | 0.4247 | 89.52% | 0.69 | 9 | 2 |
| 0.2738 | 0.5147 | 0.4217 | 87.76% | 0.40 | 5 | 9 |

*Table 6.2: Test 6.2 – House, Previous 24 hour and previous day's average demand best results*

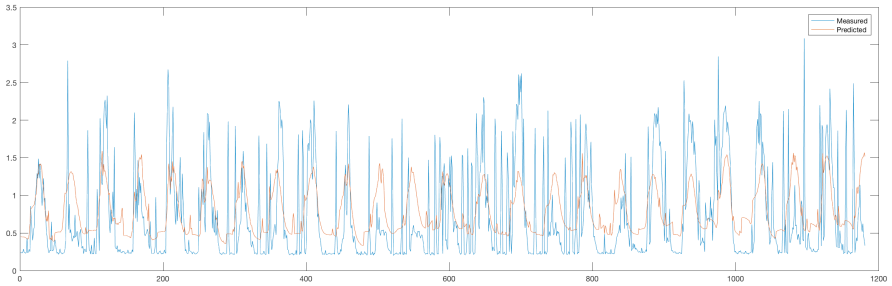


*Figure 6.10: Test 6.2 – House, Previous 24 hour and previous day's average demand best plot*

**Comments:** Results very similar to Test 6.1, still unusable.

***Test 6.3 – House, Previous 24 hour demand, previous day's average demand and outside temperature***

**Inputs:** Month (1~12), Day (1~31), Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~8 kW), Previous day's average demand (0~8 kW), Outside temperature (-4~30 °C)

**Time step:** 30 minutes

**Training Duration:** equivalent of 130 days (non-consecutive)

**Testing Duration:** 24.5 days

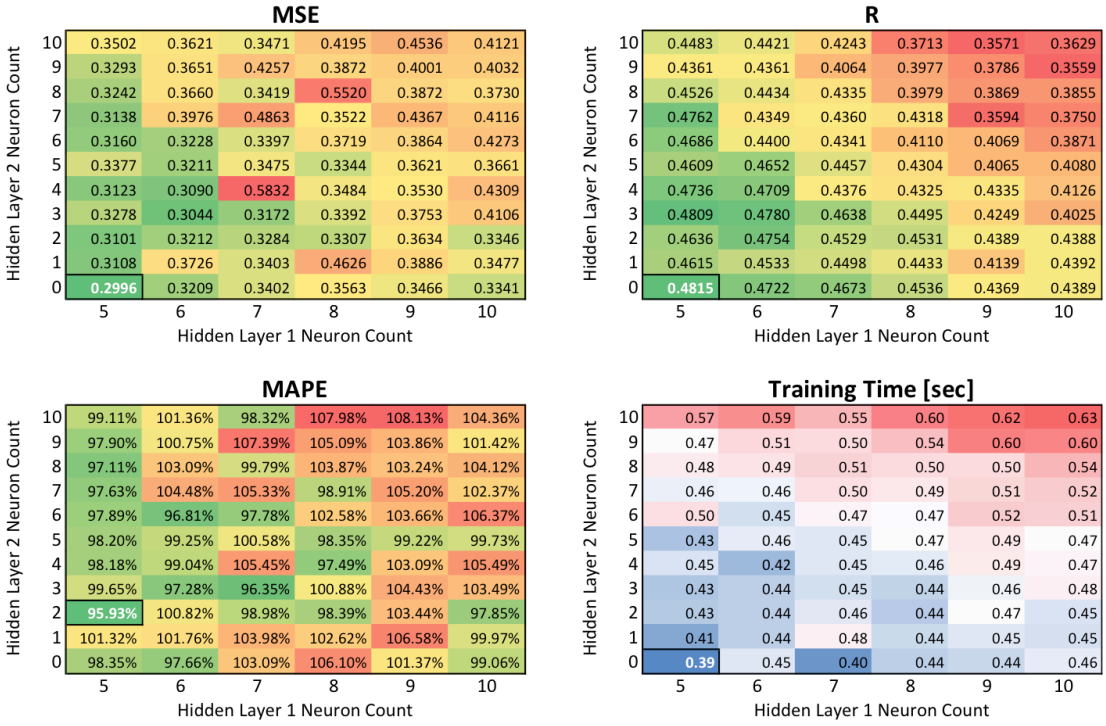


*Figure 6.11: Test 6.3 – House, Previous 24 hour demand, previous day's average demand and outside temperature average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 0.2631 | 0.5375 | 0.3982 | 78.02% | 0.54 | 6 | 3 |
| 0.2640 | 0.5354 | 0.4020 | 80.60% | 0.42 | 5 | 9 |
| 0.2648 | 0.5352 | 0.4027 | 80.14% | 0.34 | 7 | 1 |

*Table 6.3: Test 6.3 – House, Previous 24 hour demand, previous day's average demand and outside temperature best results*

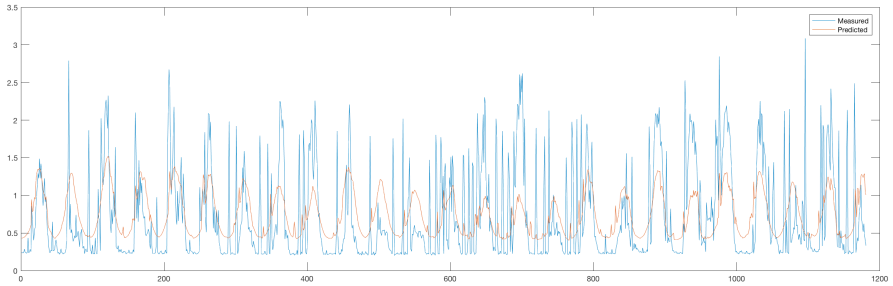


*Figure 6.12: Test 6.3 – House, Previous 24 hour demand, previous day's average demand and outside temperature best plot*

**Comments:** It is important to note that this network test battery used as n input the precise knowledge of future temperature. Even then outside temperature did not help much. Although the best results improved, they still fall on the not usable category.

### *Test 6.4 – House, No date inputs*

**Inputs:** Hour (0~23.5), Day of the week (1~7), Previous 24 hour demand (0~8 kW), Previous day's average demand (0~8 kW)

**Time step:** 30 minutes

**Training Duration:** equivalent of 176 days (non-consecutive)

**Testing Duration:** 24.5 days

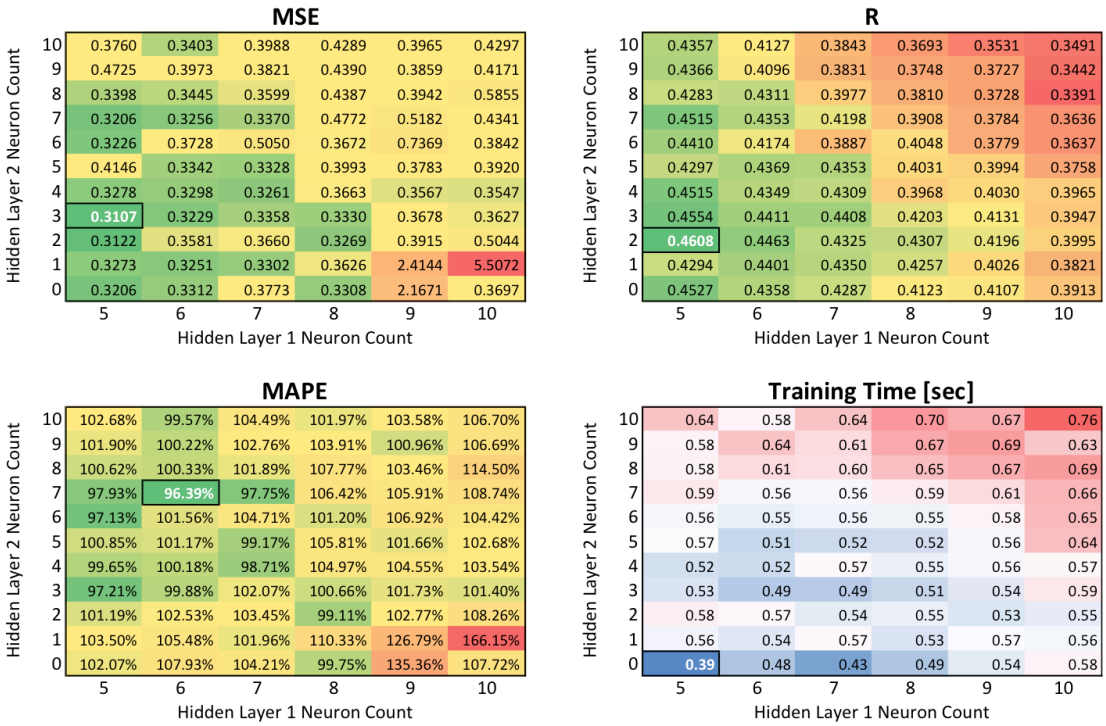


*Figure 6.13: Test 6.4 – House, No date inputs average test results*

| MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|
| 0.2677 | 0.5352 | 0.4126 | 84.67% | 0.50 | 6 | 7 |
| 0.2698 | 0.5297 | 0.4201 | 87.36% | 0.45 | 6 | 8 |
| 0.2701 | 0.5324 | 0.4243 | 89.09% | 0.47 | 7 | 1 |

*Table 6.4: Test 6.4 – House, No date inputs best results*

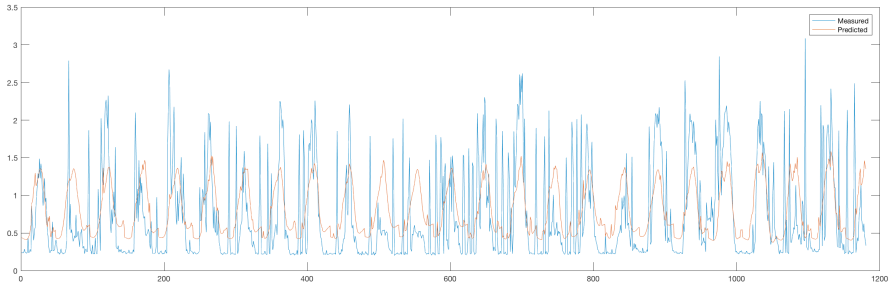


*Figure 6.14: Test 6.4 – House, No date inputs best plot*

**Comments:** The best result is marginally better than the one with date inputs. That is to be expected since the data available is so small and the algorithm has a hard time understanding how to use dates properly. But again, the results are unusable.

## 6.3 Overview

The following table contains the best run for the test batteries.

| Test # / Title | MSE | R | MAE | MAPE | Time [sec] | L1 | L2 |
|---|---|---|---|---|---|---|---|
| Test 6.1 – House, Previous 24 hour demand | 0.2688 | 0.5454 | 0.4289 | 92.91% | 1.03 | 9 | 2 |
| Test 6.2 – House, Previous 24 hour and previous day's average demand | 0.2710 | 0.5255 | 0.4159 | 87.75% | 0.45 | 8 | 7 |
| Test 6.3 – House, Previous 24 hour demand, previous day's average demand and outside temperature | 0.2631 | 0.5375 | 0.3982 | 78.02% | 0.54 | 6 | 3 |
| Test 6.4 – House, No date inputs | 0.2677 | 0.5352 | 0.4126 | 84.67% | 0.50 | 6 | 7 |

*Table 6.5: Single house, inputs strategies best test results overview*

The results are very blatantly abysmal. Even the best performance found throughout all the tests had an average error (MAPE) of 78.02%, and that was only achieved with perfect knowledge of future temperature. Even that not usable at all, having to deal with an error margin that high.

From the best plots shown on Figure 6.8, Figure 6.10, Figure 6.12 and Figure 6.14 the NNs did a poor job to forecast both the low demand and demand spikes. Therefore, at least with the amount of data and selection here used, there is not much to salvage here. Some adjustments could still be made but nothing that would make the results usable in any shape or form.

# 7 Discussion

In this section the results found on sections 4, 5 and 6 are discussed along with the potential application of neural networks in demand forecasting.

## 7.1 Energy Grid Levels

Outside some unforeseeable event, the macrogrid's demand is fairly easy to predict, given that even a date/time only based NN produced very good results. Macrogrid is not only the easiest, but also the one with the most data readily available. In some cases even commonly available on the Internet, as G.B. Grid Watch proves.

The community level is still quite forecastable. The error is of course larger than that of the macrogrid but it is still very reasonable. Although, there are quite a few spikes in demand that are hard to detect, at least without environmental data.

On the house level results got grim. The issue with data holes of course did not help, but such low level of accuracy and even demand shape left a lot to desire. Forecasting a single house did not produce a usable system, non-averaged human behaviour was just too chaotic. Although, it is not necessarily impossible.

## 7.2 Inputs

Obviously, the networks would greatly benefit from cleaner data sets, or sets without unreasonable samples (i.e. samples taken during maintenance or blackouts). But, cleaning it up is a time consuming and often a manual labour, therefore, not always possible. One possible strategy to counter that is to find outliers and substitute them for the value found on the last sample on the same time period.

One of the more frequently used strategies is to use a whole day's worth of demand with one single neural network. During the tests performed, for the neuron count limits here established the performance was lesser than with less, more targeted inputs and one single demand being forecast at a time. This should be better expanded upon to find the real answer. But like with many things in neural networks it might not be as straightforward as everyone would hope.

During the tests it was shown that using dates or even the day of the year approach was indeed beneficial. But it comes with a caveat the training dataset

available must cover the whole sample space. Otherwise when a forecast is made with values beyond the ones used during training the network will very unlikely be able to deal with them. Therefore, they should only be used when a whole year's worth of data is available for training. Similarly, the year should be avoided. It won't have a huge effect (since the value will vary proportionally little) but often there wont be enough data for the network to appropriately grasp.

It would be very beneficial to have more environmental data like outside temperature or humidity. Unfortunately, not every variable of interest (as the ones described in section 2.2) was available. As with this kind of problem, consistent data is not trivial to come by. But, if possible, to acquire or even decently estimate would generate better results.

Additionally, techniques can be applied such as using a multi-output NN to determine upper and lower bounds of possible demand (Quan, Srinivasan and Khosravi 2014). This could be specifically interesting when using inputs that have similar strategies, such as the high and low temperatures used by weather forecasters.

Input switch could be used depending on the size of the dataset. There are better strategies that require a fuller set and creating a network when it is in a more complete state would not be an issue. One interesting idea would be to have multiple networks trained with different strategies and keep watch on which performs better on each scenario. Of course there would be a dominant one, or used for the actual forecasting, but the others would simply be tested in parallel.

## 7.3 Dataset Requirements

During initial attempts on the tests with a one-month database for training the "unclean" samples were a big issue, since outliers skewed the results by a large margin. One test a 388% MAPE was reached, which on paper is a horrible result. But, as seen on Figure 5.32 the problem was not the algorithm, it was near zero samples on the final time steps of the test week.

Therefore, the algorithm itself resisted bad samples within the **Training** pool, but it would return theoretically bad results on the field if for example there was a black-out. In this scenario the system (on battery support) would indicate normal levels of demand on that time frame while no power consumption would register therefore giving out big error measurements.

Table 5.12 compares side-by-side the best neural network found for their training time. But, as shown on Figure 5.20, a 7 day training period, with the best topology, produced in average a ~17% error. Figure 5.22 shows a ~11% average error on a 30 day training period. This informs how much more reliable the network gets with a greater training time. Therefore, it can produce far better results without incurring in a too long waiting period. That is why at least one month worth of data should be produced before any control scheme is put into effect.

The performance results, even with more limited training times was valid in general. A 10% error margin could potentially be dealt with.

## 7.4 Neural Network Topology

There is no definitive way to prove which exact topology, or how many neurons in each layer, is the correct one for any specific problem. But, a general idea of which groups of topologies are better definitely can be identified.

In general at least two layers were superior to a single one. This in in compass with the understanding that two layers offers the network the capability to simulate more complex logical operators such as the XOR (exclusive or). Although not strictly true for non-perceptron neural networks the concept still stands.

It was very interesting to note that in general, the inclusion of a single neuron second hidden layer maimed the learning capacity of the neural network and held consistently worse results throughout most of the tests. That lone neuron had the task of compressing the information of the whole network and it resulted in it being more of a hindrance than if the network was left with a single hidden layer, in which all neurons (in the hidden layer) did not communicate with each other. This reinforces the idea to switch networks depending on the dataset size.

Most of the time the second hidden layer only had a positive impact once it had four or more neurons. A notable exception of this was the network in *Test 5.3*, the one that included the demand from the previous 168-hour (or the same time of day one week prior) likely due to a good match between the weeks, therefore not too much was needed. But still, this proved inferior to other input sets with more neurons on the second hidden layer.

Regardless, the results matrixes found give a general idea of how to choose a network topology. A concentration of good results merely indicates the best

performing topology concept (e.g. many neurons on the first layer and few neurons on the second). This occurs due to the uncertainty attached to the algorithm that can occasionally perform abnormally making it hard to pinpoint the exact optimal topology.

Curiously, a smaller data set sprung the notion that simpler topologies could be better, at least in average, than more complex ones. This is seen on Figure 5.22, Figure 5.24, Figure 5.27 and where a shift on the best topology for average results goes from simple (bottom left corner of the result matrixes) to complex (top right corner of the result matrixes).

## 7.5 Implementation

The test computer took a few hours (overall averaging on 3 hours) to sweep from 5 to 10 on the first hidden layer and from 0 to 10 on the second hidden layer 50 times each.

When trained, a neural network is not much more beyond the weight values found. One single neuron (including in the output layer) requires one weight value for each previous neuron connected to it. Even adding the code on top of it wont break the MB threshold.

Additionally, it is required to store all the information to create the database. Given a sample with 5 inputs (e.g.: year, month, day, time, demand, temperature), a 30 minute time step and even using solely high precision 8 byte double types (for every value) will consume take little more than 500 kB for a four year period.

The test computer used a 2.5 GHz quad-core CPU and benefited from MATLAB's parallel processing. Each individual test often kept the duration at a few seconds. Depending on the hardware being employed on an actual application of this system, the short training time found here might not be as realistic. Retraining sections will be infrequent, at maximum once a week, therefore not much of an issue is expected as long as the demand readings are acquired and up-to-date. Even so, a 100 times slower processor would still take only a few minutes in average to finish one training section. In comparison, a hardware called Raspberry Pi 3 Model B (RASPBERRY PI 3 MODEL B 2016) that is basically a computer costing mere U$ 30,00 on Amazon (Amazon.com Raspberry PI 3 Model B Motherboard 2016) has

a 1.2GHz 64-bit quad-core processor while the test computer had a 2.5 GHz Intel Core i7 processor. Therefore, processing power is seldom an issue.

The usage of the **Testing** data pool (as discussed on section 2.1) was seen as innocuous as it seldom detected a good or bad NN forecasting prowess. It usually performs similarly as the **Training** and **Validation** data pools when they are randomly chosen from the original dataset. Very large error measurements on these pools usually lead to bad performing NNs, but low error measurements don't correspond necessarily to good performances, therefore it can be used as an indicator at best but not much more.

The goal of Test 5.11 was to find if there was parts of the day that were easier or harder to forecast. A little before noon and early afternoon induced the most error due to hard to predict demand spikes. This is very unfortunate since that is the most important period to forecast.

Beyond everything there is a certain luck element to a good performing neural network. Neural networks is a quite hard to debug. Even on good configurations, bad performing outliers still appear. Therefore, its results should not be treated as a hard fact, uncertainty treatments must be considered.

There is potential to design control schemes that revolve around this type of forecasting. Naturally, the other part of the problem would be to actually develop a good control strategy, one that takes into account the probabilistic nature of the forecasting. It is very clear that it should not be done in the house level just yet, but a community's microgrid would fit quite nicely. Combining stochastic power sources with some form of energy storage could potentially be greatly optimized if the system were to have an idea of what will be demanded of it in the near future. In terms of hardware there is no limitation but a good deal of tests are still required to be performed on both the forecasting and control sides.

## 7.6 Future Work

Both clustering and time series neural network types (discussed in section 3.1.1) within MATLAB are promising in the profiling of daily curves and the actual demand forecasting. Although this paper limited itself on the more direct data fitting type the other options are quite valid and deserve consideration on dedicated future works.

It is plausible to think that a dedicated network for specific times of the day can generate better results. Thereby, shifting attention to more critical periods. An investigation on that should be an interesting endeavour.

Few tests with a third hidden layer or larger neuron count were done here with less than impressive results. Although very time consuming, since the time required increases quite a lot, checking larger network topologies would be very welcome.

Also, the differing results depending on dataset size would be very interesting to further study. Finding the optimized configuration depends on many factors and they should be further explored.

Additionally, each study case has its particularities and slightly different results are expected. But more should be tested to better certify base concepts. It is plausible that different scenarios perform better with different configurations.

Since individual houses create such less accurate results and microgrids consist of communities it is not necessary to deploy the system on cheap hardware, but it is quite feasible to implement it on microcontrollers and other small hardware, such as Arduino and Raspberry PI. It would be a good proof of concept to deploy and field-test it.

During the tests, on the community level the average error could get as low as 8%. Beyond just predicting values it is important to establish how to use the predictions having in mind the uncertainty that is still present. So, developing said control strategies goes hand-in-hand with forecasting.

# 8 Conclusion

In order to optimize the use of energy, and specially the use of renewable energy, some case studies on different grid levels were attempted.

On both the macrogrid and community levels the results can be considered useful. Sadly, on a single house that is not true.

Utilizing neural networks is far from being a perfect tool with strong pre-established techniques. A lot of the hardships come from having a good pre-existing knowledge of how the system being analysed works and even then the results can be surprising. Regardless, more documentation must be created on each the specific subject to better guide their application.

Nevertheless, neural networks is an important tool that, if applied with enough care, can have a lasting effect in how uncertain systems are modelled and predicted.

Hopefully, with the advent of smart metering options for the end user more data will be available to better calibrate future models.

# 9 References

*Amazon.com Raspberry PI 3 Model B Motherboard.* 2016. https://www.amazon.com/Raspberry-Pi-RASP-PI-3-Model-Motherboard/dp/B01CD5VC92/ref=sr_1_cc_1?s=aps&ie=UTF8&qid=1471105269&sr=1-1-catcorr&keywords=RASPBERRY+PI+3+MODEL+B (accessed on 13 of August, 2016).

Benedetti, Miriam, Vittorio Cesarotti, Vito Introna, e Jacopo Serranti. "Energy consumption control automation using Artificial Neural Networks and adaptive algorithms Proposal of a new methodology and case study." *Applied Energy* , 2015.

Dorofki, Mohammad, Ahmed Elshafie, Othman Jaafar, Othman Karim, e Sharifah Mastura. "Comparison of Artificial Neural Network Transfer Functions Abilities to Simulate Extreme Runoff Data." *International Conference on Environment, Energy and Biotechnology* . Bangi, 2012.

Dudek, Grzegorz. "Neural networks for pattern-based short-term load forecasting: A comparative study." *Neurocomputing* , 2016.

Durek, Grzegorz. "Pattern-based local linear regression models for short-term load forecasting ." *Electric Power Systems Research* (Elsevier), 2015.

"G.B. National Grid Status." http://www.gridwatch.templar.co.uk (accessed on 13 of June, 2016).

GreenAnt. *GreenAnt Web page.* http://www.greenant.com.br/en/home_en/ (accessed on 28 of June, 2016).

Ismailov, Vugar. "On the approximation by neural networks with bounded number of neurons in hidden layers." *Journal of Mathematical Analysis and Applications* , 2014.

MacKay, Davic. "Information Theory, Inference, and Learning Algorithms." C.U.P., 2005.

MathWorks. *MATLAB Webpage.* http://www.mathworks.com/products/matlab/ (accessed on 28 of June, 2016).

ORIGIN. *ORIGIN Project Web Site.* http://www.origin-energy.eu (acesso em 29 de June de 2016).

Owens, Edward. "ORIGIN Final Report." 2015.

Quan, Hao, Dipti Srinivasan, e Abbas Khosravi. "Uncertainty handling using neural network-based prediction intervals for electrical load forecasting ." *Energy* (Elsevier), July 2014.

*RASPBERRY PI 3 MODEL B.* 2016. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (acesso em 13 de August de 2016).

Rowe, Matthew, Timur Yunusov, Stephen Haben, William Holderbaum, e Ben Potter. "The Real-Time Optimisation of DNO Owned Storage Devices on the LV Network for Peak Reduction." *energies*, 2014.

Setlhaolo, Ditiro, Xiaohua Xia, e Jiangfeng Zhang. "Optimal scheduling of household appliances for demand response." (Elsevier) 116 (November 2014): 24-28.

Stephen, Bruce, Xiaoqing Tang, Poppy R. Harvey, Stuart Galloway, e Kyle I. Jennett. "Incorporating Practice Theory in Sub-Profile Models for Short Term Aggregated Residential Load Forecasting." *IEEE Transactions on Smart Grid*, 2015.

Tao, Hong. "Short Term Electric Load Forecasting." Operations Research and Electrical Engineering, North Carolina State University, 2010, 175.

Wikimedia. *Wikipedia.* https://en.wikipedia.org/ (accessed on June, 2016).

Zhao, Hai-xiang, e Frédéric Magoulès. "Renewable and Sustainable Energy Reviews." *Elsevier*, 2012.

# 10 Appendix: MATLAB Scripts

```
% Neural Network test battery script
% for 2 layers
% This script assumes these variables are defined:
%
% ti = training inputs
% to = training outputs
% ei = test inputs
% eo = test outputs

tic % Algorithm chronometer start

% Initialize Variables
trainFcn = 'trainlm'; % Training Function Levenberg-Marquardt
backpropagation.
hl1low = 5;    hl1high = 10; % Hidden layer 1 range
hl2low = 0;    hl2high = 10; % Hidden layer 2 range
iend = 50; % Number of tests per topology
MMSE = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % MSE result per
topology
MR = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % R result per
topology
MMAE = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % MAE result per
topology
MMAPE = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % MAPE result
per topology
Mtime = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % time result
per topology
net1 = fitnet([1],trainFcn); % Number 1 best result network
net2 = fitnet([1],trainFcn); % Number 2 best result network
net3 = fitnet([1],trainFcn); % Number 3 best result network
BRM = zeros(3,7); BRM(:,1) = Inf; % Best Results Matrix
sz = size(eo);
x = [1:1:sz(1,1)];

% Test loops
for hl2 = hl2low:hl2high
    for hl1 = hl1low:hl1high
        for i = 1:iend
            % Define network topology
            if (hl2 == 0)
                topology = hl1;
            else
                topology = [hl1 hl2];
            end

            % Innitialize a Fitting network
            net = fitnet(topology,trainFcn);
            % Setup Division of Data for Training, Validation, Testing
            net.divideParam.trainRatio = 75/100;
            net.divideParam.valRatio = 25/100;
            net.divideParam.testRatio = 0/100;
            % Train the network
            [net,tr] = train(net,ti',to','useParallel','yes');
```

```
            % Test the network
            res = net(ei')'

            % Auxiliary function for performance results
            error = gsubtract(res,eo);
            aux = rdivide(abs(eo-res),eo);
            aux(isinf(aux)) = 1;
            % Performance results
            MSE = mse(res,eo);
            R = regression(eo,res,'one');
            MAE = mae(error);
            MAPE = mae(aux);
            time = max(tr.time);

            % Add result to Result Matrixes
            MMSE(hl2high-hl2+1, hl1-hl1low+1) = MMSE(hl2high-hl2+1, hl1-
hl1low+1) + MSE;
            MR(hl2high-hl2+1, hl1-hl1low+1) = MR(hl2high-hl2+1, hl1-
hl1low+1) + R;
            MMAE(hl2high-hl2+1, hl1-hl1low+1) = MMAE(hl2high-hl2+1, hl1-
hl1low+1) + MAE;
            MMAPE(hl2high-hl2+1, hl1-hl1low+1) = MMAPE(hl2high-hl2+1, hl1-
hl1low+1) + MAPE;
            Mtime(hl2high-hl2+1, hl1-hl1low+1) = Mtime(hl2high-hl2+1, hl1-
hl1low+1) + time;

            % Check and update if new best result was found
            if (MSE < BRM(3,1)) % If new is in the  top 3 (included for
speed)
                if (MSE < BRM(1,1))
                    BRM(3,:) = BRM(2,:);
                    net3 = net2;
                    BRM(2,:) = BRM(1,:);
                    net2 = net1;
                    BRM(1,:) = [MSE R MAE MAPE time hl1 hl2];
                    net1 = net;
                else
                    if (MSE < BRM(2,1))
                        BRM(3,:) = BRM(2,:);
                        net3 = net2;
                        BRM(2,:) = [MSE R MAE MAPE time hl1 hl2];
                        net2 = net;
                    else
                        if (MSE < BRM(3,1))
                            BRM(3,:) = [MSE R MAE MAPE time hl1 hl2];
                            net3 = net;
                        end
                    end
                end
            end

        end
    end
end

% Average Result Matrixes
MMSE = MMSE/iend;
MR = MR/iend;
```

```
MMAE = MMAE/iend;
MMAPE = MMAPE/iend;
Mtime = Mtime/iend;

% Error histogram for Net1
res = net1(ei')'
error = gsubtract(res,eo);
histogram(error);
% Plot Measured, Predicted for Net1
plot(x,eo, x,res), legend('Measured', 'Predicted');

toc % Algorithm chronometer stop

% END OF Neural Network test battery script
```

```
% Neural Network with moving test battery script
% for 2 layers
% This script skips test periods that contain near zero (<4) values in it
% the <4 value was selected for one specific database and should be
% changed according to what can be identified as a near zero value
%
% This script assumes these variables are defined:
%
% ti = inputs
% to = outputs

tic % Algorithm chronometer start

% Initialize Variables
trainFcn = 'trainlm'; % Training Function Levenberg-Marquardt
backpropagation.
hl1low = 5;    hl1high = 10; % Hidden layer 1 range
hl2low = 0;    hl2high = 10; % Hidden layer 2 range
iend = 50; % Number of tests per topology
train_days = 7;
test_days = 7;
trainsize = train_days*48; % Size of training pool
testsize = test_days*48; % Size of test pool
MMSE = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % MSE result per
topology
MR = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % R result per
topology
MMAE = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % MAE result per
topology
MMAPE = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % MAPE result
per topology
Mtime = zeros(hl2high - hl2low + 1, hl1high - hl1low + 1); % time result
per topology
net1 = fitnet([1],trainFcn); % Number 1 best result network
net2 = fitnet([1],trainFcn); % Number 2 best result network
net3 = fitnet([1],trainFcn); % Number 3 best result network
BRM = zeros(3,8); BRM(:,1) = Inf; % Best Results Matrix
st = 1; % Start position
szi = size(ti); szo = size(to);
mov = (szo(1,1) - trainsize - testsize) / (iend - 1); % Movement
tix = zeros(trainsize,szi(1,2)); % Moving training input pool
tox = zeros(trainsize,szo(1,2)); % Moving training output pool
eix = zeros(testsize,szi(1,2)); % Moving test input pool
eox = zeros(testsize,szo(1,2)); % Moving test output pool
list = zeros(iend*(hl2high - hl2low + 1)*(hl1high - hl1low + 1),10);
ind = 1;

% Test loops
for hl2 = hl2low:hl2high
    for hl1 = hl1low:hl1high
        for i = 1:iend
            % Moving test routine
            if (floor(st)+trainsize+testsize > szo(1,1)) % Ensure dataset
is big enough
                st = 1;
            end

            eox = to(floor(st)+trainsize:floor(st)+trainsize+testsize-1,:);
% Test Outputs
```

```
            while sum(find(eox<4))>0 % Check for near-zero values
                st = st + mov; % Move test pool
                if (floor(st)+trainsize+testsize > szo(1,1)) % Ensure
dataset limit is not exceeded
                    st = 1;
                end
                eox = to(floor(st)+trainsize:floor(st)+trainsize+testsize-
1,:); % Test Outputs
            end

            tix = ti(floor(st):floor(st)+trainsize-1,:); % Training inputs
            tox = to(floor(st):floor(st)+trainsize-1,:); % Training outputs
            eix = ti(floor(st)+trainsize:floor(st)+trainsize+testsize-1,:);
% Test Inputs
            eox = to(floor(st)+trainsize:floor(st)+trainsize+testsize-1,:);
% Test Outputs

            % Define network topology
            if (hl2 == 0)
                topology = hl1;
            else
                topology = [hl1 hl2];
            end

            % Innitialize a Fitting network
            net = fitnet(topology,trainFcn);
            % Setup Division of Data for Training, Validation, Testing
            net.divideParam.trainRatio = 75/100;
            net.divideParam.valRatio = 25/100;
            net.divideParam.testRatio = 0/100;
            % Train the network
            [net,tr] = train(net,tix',tox','useParallel','yes');

            % Test the network
            res = net(eix')'

            % Auxiliary function for performance results
            error = gsubtract(res,eox);
            aux = rdivide(abs(eox-res),eox);
            aux(isinf(aux)) = 1;
            % Performance results
            MSE = mse(res,eox);
            R = regression(eox,res,'one');
            MAE = mae(error);
            MAPE = mae(aux);
            time = max(tr.time);

            list(ind,:) = [MSE R MAE MAPE time hl1 hl2 st tr.best_perf
tr.best_vperf];
            ind = ind + 1;

            % Add result to Result Matrixes
            MMSE(hl2high-hl2+1, hl1-hl1low+1) = MMSE(hl2high-hl2+1, hl1-
hl1low+1) + MSE;
            MR(hl2high-hl2+1, hl1-hl1low+1) = MR(hl2high-hl2+1, hl1-
hl1low+1) + R;
            MMAE(hl2high-hl2+1, hl1-hl1low+1) = MMAE(hl2high-hl2+1, hl1-
hl1low+1) + MAE;
```

```
            MMAPE(hl2high-hl2+1, hl1-hl1low+1) = MMAPE(hl2high-hl2+1, hl1-
hl1low+1) + MAPE;
            Mtime(hl2high-hl2+1, hl1-hl1low+1) = Mtime(hl2high-hl2+1, hl1-
hl1low+1) + time;

            % Check and update if new best result was found
            if (MSE < BRM(3,1)) % If new is in the  top 3 (included for
speed)
                if (MSE < BRM(1,1))
                    BRM(3,:) = BRM(2,:);
                    net3 = net2;
                    BRM(2,:) = BRM(1,:);
                    net2 = net1;
                    BRM(1,:) = [MSE R MAE MAPE time hl1 hl2 st];
                    net1 = net;
                else
                    if (MSE < BRM(2,1))
                        BRM(3,:) = BRM(2,:);
                        net3 = net2;
                        BRM(2,:) = [MSE R MAE MAPE time hl1 hl2 st];
                        net2 = net;
                    else
                        if (MSE < BRM(3,1))
                            BRM(3,:) = [MSE R MAE MAPE time hl1 hl2 st];
                            net3 = net;
                        end
                    end
                end
            end

            st = st + mov; % Move test pool

        end
    end
end

% Average Result Matrixes
MMSE = MMSE/iend;
MR = MR/iend;
MMAE = MMAE/iend;
MMAPE = MMAPE/iend;
Mtime = Mtime/iend;

% Re-aquiring best result vectors
st = BRM(1,8);
eix = ti(floor(st)+trainsize:floor(st)+trainsize+testsize-1,:);
eox = to(floor(st)+trainsize:floor(st)+trainsize+testsize-1,:);
% Error histogram for Net1
res = net1(eix')'
error = gsubtract(res,eox);
histogram(error);
% Plot Measured, Predicted for Net1
plot([1:1:testsize],eox, [1:1:testsize],res), legend('Measured',
'Predicted');

toc % Algorithm chronometer stop

% END OF Neural Network with moving test battery script
```