A Comparison of Hybrid and Statistical Forecasting Techniques for Short-Term Energy Demand Forecasting

Matthew Ingram

MSci Mathematics and Computer Science; Department of Computer Science and Engineering, University of Durham, Durham, England (e-mail: matthew.ingram@durham.ac.uk).

Abstract: Following the latest iteration of the M Time Series Forecasting Competition, this paper compares the performance of the winning (hybrid) model to classical statistical models, when applied specifically to short-term energy demand forecasting. The hybrid model combines an exponential smoothing model and a dilated Long Short-Term Memory recurrent neural network into one hybrid architecture. We extend this model through the inclusion of exogeneous, contemporaneous weather data. The statistical forecasting methods included for comparison are: the Auto-Regressive Integrated Moving Average (ARIMA) model, Holt-Winters' exponential smoothing, and the *Theta* method, among others. We find that for forecast horizons of up to 6 hours, the ARIMA model performs optimally. When forecasting at longer horizons, the naive seasonal method is dominant. For consistently strong forecasts across all lead times from 1 to 48 hours, the novel hybrid models introduced in this paper are shown to be preferable. Therefore, we conclude that the hybrid model is indeed suitable for short-term energy demand forecasting.

Keywords: Time Series Forecasting, Hybrid, Exponential Smoothing, Recurrent Neural Networks, Statistical Forecasting, Machine Learning, Short-Term Energy Demand

1. INTRODUCTION

Typically, electricity is delivered to consumers via the electric grid, where the total demand for power must be matched by the total power generated at all times. As renewable sources of power have become more prominent (such as solar panels and wind turbines), highly-polluting fossil fuel plants provide what is known as the *spinning reserve* – the unused capacity which can be activated on decision of the system operator (Rebours and Kirschen 2005) - in order to meet unexpected shortages in supply or surges in demand. An improvement in the accuracy of short-term energy demand forecasts can enable operators to reduce reliance on such standby plants (Rolnick et al. 2019), thus reducing "greenhouse gas" emissions. There is an increasingly urgent need to eliminate global "greenhouse gas" emissions (Masson-Delmotte et al. 2018), and short-term energy demand forecasting has been identified as a *high leverage* area, particularly well-suited to recent advances in machine learning and time series forecasting. This paper compares the performance of a state-of-the-art hybrid forecasting model with traditional statistical forecasting techniques, to assess the suitability of such a model to the specific task of short-term energy demand forecasting. The architecture is *hybrid* in the sense that it combines elements of both statistical and machine learning techniques.

Beyond the desire to reduce greenhouse gas emissions, the industry-needs for energy demand forecasting are numerous, including (but not limited to): the purchasing of energy, transmission and distribution planning, scheduling maintenance outages, and demand side management (Hong 2010). Furthermore, energy consumption is often used as an economic indicator for the growth and development of a country or city (Lee and Tong 2011), and is also used to influence government policy decisions. For example, with China aiming to diversify their energy production - seeking a fall in the share of coal in total generation from two-thirds today to less than 40% by 2040 (I. E. Agency 2017) - accurate energy demand prediction is of critical importance. As this paper focuses on the short-term demand for energy, we shall not be concerned with the difficulties of predicting long-term economic trends and cycles. Rather, the difficulty in prediction stems from the stochastic nature of human behaviour and the variability of the weather (Khodayar and Wu 2015).

The exponential growth in the amount of data available, significant increase in the processing power of personal computers (in particular through the widespread availability of general-purpose graphics processing units (GPGPUs)), and a huge surge in research output have led to a considerable improvement in the capabilities of machine learning algorithms in a variety of tasks, such as the *ImageNet Large Scale Visual Recognition Challenge* (Russakovsky et al. 2014). Machine learning algorithms have shown particular prowess in fields such as natural language processing (Mikolov et al. 2013; Y. Wang et al. 2016) and computer vision (Krizhevsky, Sutskever, and Geoffrey E Hinton 2012; He et al. 2015). However, when applied to the task of univariate time series prediction, machine learning techniques have typically performed disappointingly, as evidenced by the results of the recent M4 Competition (Makridakis, Spiliotis, and Assimakopoulos 2020). Five machine learning models were submitted and all were less accurate than the *Comb* method - which is just the arithmetic average of three simple exponential smoothing-based methods. In fact, only one of the five models outperformed the *Naïve2* method, which is simply the re-seasonalised naïve forecast of de-seasonalised data. Historically, other empirical studies have similarly found machine learning models to perform disappointingly (Makridakis, Spiliotis, and Assimakopoulos 2018; Crone, M. Hibon, and Nikolopoulos 2011).

Although the "pure" machine learning models generally performed poorly, the overall winner of the M4 Competition was a hybrid model which exploited the advantages of both statistical and machine learning methods, whilst seemingly avoiding their shortcomings. The model, proposed by Slawek Smyl of Uber Technologies, uses gradient descent to fit individual exponential smoothing (ES) (Robert Goodell Brown 1963) parameters to multiple time series, which are then used to de-seasonalise the time series before they are input into a recurrent neural network (RNN) (S. Smyl 2020), which produces the final prediction. Henceforth, we shall refer to this model as the ES-RNN model. The parameters of the RNN are trained simultaneously with the exponential smoothing parameters and are shared globally between all time series. The RNN architecture used in the model is a modified version of a standard RNN architecture, which introduces *dilations* - direct connections between periodically separated values in the input sequences - in an attempt to capture the seasonality of the data (Chang et al. 2017). For the individual RNN blocks, Smyl uses the standard Long Short-Term Memory (LSTM) architecture introduced in (Sepp Hochreiter and Schmidhuber 1997). The LSTM improves upon the vanilla RNN architecture, which struggles to learn long-term dependencies due to the vanishing gradient problem (Bengio, Simard, and Frasconi 1994) and the exploding gradient problem (S. Hochreiter et al. 2001). By fitting exponential smoothing parameters to each time series individually, whilst simultaneously training the parameters of the RNN across all of the time series, Smyl hypothesised that the model would be able to learn the intricate seasonal and non-seasonal patterns of the data. Slawek's model is a truly hybrid model that integrates statistical and machine learning concepts simultaneously, as opposed to several other models which simply use machine learning techniques to train the (hyper-) parameters of existing statistical models (Yang and Li 2006).

We chose to use Spanish system-level Transmission Service Operator (TSO) energy demand data (ENTSO-E 2019) to evaluate the forecasting models, as this dataset contains time series representing the total aggregated demand, as well as generation data stratified by energy type (wind, coal, etc). Smyl's original formulation of the ES-RNN model makes use of multiple time series to facilitate cross-learning in the LSTM, and this dataset provides the opportunity to replicate this functionality. We use a refactored version of the original dataset, which contains 4 years of energy demand, pricing, and weather data for Spain, as well as generation data stratified by energy type, as just mentioned (Jhana 2019). One of the novel contributions of this paper is the extension of the ES-RNN model to include this exogenous weather data. It has been shown that temperature is the key influencing factor in load forecasting (Fay and Ringwood 2010; Pitt 2000), and so we feel that any forecasting model would be incomplete if it did not utilise this correlation. The dataset includes temperature, rainfall, humidity, wind speed and pressure data for 5 major cities in Spain: Valencia, Madrid, Barcelona, Seville and Bilbao.

In this paper, we use the dataset described above to perform a comprehensive comparison of the hybrid model against multiple existing classical statistical techniques. We split the data by year and by season, giving multiple sub-datasets across which we average the models' performance. This also allows us to perform per-season analysis. The key contributions of this work are validating that Smyl's hybrid model is indeed suitable for short-term energy demand forecasting, and extending the original implementation to include contemporaneous weather information. We also modify the training procedure of the model to one more suitable for the dataset used and short-term energy demand forecasting. We find that the hybrid model is able to generate forecasts of competitive accuracy across *all* forecasting horizons from 1 to 48 hours, which none of the classical statistical techniques are able to do.

2. LITERATURE REVIEW

Over the past three decades, there have been two broad categories of energy forecasting techniques: data-driven (machine learning) techniques and statistical techniques (Hong and Fan 2016). Within the family of statistical techniques, there are broadly three main sub-groups of models: multiple linear regression models, exponential smoothing models, and auto regressive/moving average models. Historically, time series orientated techniques (both machine learning and statistical) have been computationally expensive, numerically unstable and were often overlooked as they failed to utilise the strong correlation between weather information and energy demand (Park et al. 1991). However, with the rapid increase in computational power in personal computers, along with the development of stable statistical and machine learning frameworks (Rob J Hyndman and Khandakar 2008; Paszke et al. 2019), such issues have largely been negated in modern time series forecasting. Furthermore, both machine learning and statistical models have been developed to include exogenous variables such as weather information (Chow and Leung 1996; Nengbao, Babushkin, and Afshari 2014). Whilst regression models are typically computationally inexpensive in comparison, they often model the relationship between weather and energy demand as a linear relationship, which can lead to poor forecasting accuracy (Park et al. 1991). Furthermore, the accuracy of regression based models often relies upon the accuracy of the forecasts of their regressor variables, which is often inaccurate in the case of meteorological information, or non-trivial in the case of economic indicators (such as GDP). Multivariate models that use weather variables are also typically unsuitable for short-term forecasting, as meteorological changes occur in a smooth fashion which will be captured in the series itself (J. Taylor and Mcsharry 2007).

2.1 ARIMA Models

Autoregressive integrated moving average (ARIMA) models are a family of forecasting techniques introduced by Box and Jenkins in their seminal work (G. Box and Jenkins 1976). An $ARIMA(p, d, q)$ model assumes that the current time series value is a weighted linear combination of p lagged values (the *autoregressive* terms) and q previous errors (*moving average* terms). If the time series is non-stationary, differencing of order d may be applied, hence the *integrated* in the model's name. Along with exponential smoothing models, ARIMA models have been one of the

primary time series forecasting techniques used in the past half-century. The first application of ARIMA models to short-term load forecasting was performed in (M. Hagan and Klein 1977a). Hagan and Behr later improved their model by including exogenous weather variables, thus implementing what is now commonly referred to as an ARIMAX model (M. Hagan and Klein 1977b). ARIMAX models can be considered as a special case of the more general family of models introduced by Box and Jenkins, known as *transfer function models*. Such models, also referred to as *dynamic regression models*, allow for the lagged and decaying effects of contemporaneous covariates. Hagan further refined his ARIMAX model by modifying the transfer function to include the non-linear temperature dependence of energy demand (M. T. Hagan and Behr 1987).

More recently, ARIMA and seasonal ARIMA (SARIMA) models have been used in Turkey to predict the primary energy demand (Ediger and Akar 2007). SARIMA models are a natural extension of standard ARIMA models, which facilitate the inclusion of *periodic* autoregression, error dependence and differencing to model seasonal effects. Seasonality of multiple frequencies can also be handled as demonstrated in (N. Mohamed et al. 2010), in which a double seasonal ARIMA model is used to forecast short-term load demand in Malaysia with high accuracy. Whilst the ARIMA family of models are extremely popular, they do have a number of limitations. Typically, they are far more computationally expensive to fit than the exponential smoothing family of models (discussed below). Furthermore, one key assumption in ARIMA modelling is that the underlying time series is stationary, or can be made stationary through differencing (Robin John Hyndman and Athanasopoulos 2018). In real life situations, this is typically not the case (Commandeur and Koopman 2007).

2.2 Exponential Smoothing Models

Exponential smoothing in its simplest form was first introduced by Brown in (Robert G. Brown and Little 1956). Known as *Simple Exponential Smoothing*, the forecasted value is simply a weighted combination of the previous smoothed (forecasted) value and the most recent observation. Holt later introduced the double exponential smoothing model which adds a term to take into account a linear trend in the data (Holt 2004). The smoothing is applied recursively: first to update the level of the data and then to update the slope component. Winters, a student of Holt, further extended exponential smoothing by introducing (with Holt) the Holt-Winters' model (Winters 1960). The model applies recursive smoothing three times, introducing seasonal indices which represent the amount periodically-separated data points typically lie above/below the smoothed value. The advantages of these models include being computationally inexpensive and requiring little formal specification in comparison to the ARIMA family of models (J. Taylor, De Menezes, and Mcsharry 2006).

More recently, Taylor introduced a second seasonality term and applied the model to the task of short-term electricity demand forecasting (J. Taylor 2003). The model was able to capture both intra-day and intra-week seasonality and outperformed Holt-Winters' traditional model. A comparison of the performance of six univariate models (including seasonal ARMA, a basic feed-forward neural network, and double-seasonal exponential smoothing) for short-term electricity demand forecasting was performed in (J. Taylor, De Menezes, and Mcsharry 2006). Taylor found that the double-seasonal exponential smoothing model outperformed the other models, and the accuracy of the neural network was noticeably poor. The experiments were then repeated (with additional models) on European data in (J. Taylor and Mcsharry 2007). Taylor further expanded his exponential smoothing model to include the effects of annual seasonality in (James W. Taylor 2010). The results of his experiments showed that triple seasonal models can outperform double seasonal models, as well as a univariate neural network. Although there may be intuitive appeal in modelling the annual seasonal cycle, its benefit for short-term prediction is ultimately a matter for empirical testing, as the yearly seasonal effects do not bare much influence on such small time scales (J. Taylor 2003).

In (James W. Taylor 2010), a simple average combination of the forecasts of the models led to the greatest forecast accuracy. The idea of combining forecasts is not new; indeed, (Bates and Granger 1969) was the first to show that combinations of models can improve forecast accuracy. Over 200 papers regarding the combination of forecasts were reviewed in (Clemen 1989), and the overarching conclusion was that combining forecasts leads to increased forecast accuracy. In the specific case of energy demand forecasting, a smoothing method combining multiple forecasts with changing weights was implemented in (J W Taylor and Majithia 2000) for electricity demand profiling.

2.3 Regression Models

In (multiple) linear regression (MLR) methods, the load demand is found in terms of regressor variable(s) such as temperature and humidity, which typically influence the demand for energy (Moghram and Rahman 1989). While the benefits of such models include being extremely easy to specify and being very simple to understand, their accuracy fundamentally relies on the accuracy of forecast of the regressor variables. Further, the relationship between temperature and short-term demand has been shown to be non-linear (M. T. Hagan and Behr 1987). In 1990, a sophisticated MLR-based method was developed to improve the short-term system load forecasting at Pacific Gas and Electric Company (PG&E) (Papalexopoulos and Hesterberg 1990). This model was found to significantly outperform the previous forecasting model used at PG&E. Hor et al. analyzed the impact of weather variables on monthly electricity demand in (Ching-Lai Hor, Watson, and S. Majithia 2005), and found their model achieved a mean absolute percentage error (MAPE) of 2.69% on unseen data. In their analysis they compared three different MLR models, and found the models that included more sophisticated regressor variables, such as *degree days* and *enthalpy latent days*, outperformed the model that used the temperature and humidity values directly. An MLR method was also used in (Cancelo, Espasa, and Grafe 2008) to forecast one day to one week ahead system load for a Spanish system operator.

2.4 Machine Learning and Hybrid Models

The first application of an artificial neural network (ANN) to short-term load forecasting was performed in (Park et al. 1991). More recently, an ANN was used to predict the one hour ahead load for the Western Area of Saudi Arabia (Alshareef, E. Mohamed, and Al-Judaibi 2008). The results of Alshareef et al. were highly promising, with their forecasts having an average MAPE of just 1.12%. Around the same time, Ekonomou et al. developed a similar ANN model to forecast Hellenic daily electricity demand (Ekonomou and D.S. 2008). A key difference in their model is that they did not include lagged values as inputs. Artificial neural networks have also been used to predict electricity demand in Thailand (Kandananond 2011). The performance of an ANN was compared against that of an ARIMA model and an MLR model, and found to be superior. Kong et al. were one of the first to apply LSTM networks to short-term demand forecasting (Kong et al. 2017). The LSTM is a member of the family of *recurrent neural networks* (RNN), which are particularly well-suited to working with sequential data, such as that found in natural language processing (NLP) tasks (Lipton 2015). This also makes them well-suited for time series analysis and prediction. The LSTM was introduced in 1997 by Hochreiter and Schmidhuber, to remedy the *vanishing gradients* problem that exists with *vanilla* RNNs (Sepp Hochreiter and Schmidhuber 1997).

The notion of a *hybrid* model is very loosely defined in energy demand forecasting literature. Some "hybrid" models use existing or novel optimisation algorithms to deduce the weights of neural networks (Abedinia, Amjady, and Ghadimi 2018; N. Liu et al. 2014). In (Abedinia, Amjady, and Ghadimi 2018), a novel optimisation algorithm called *improved shark smell optimisation* (ISSO) is used to find the optimal values of the weights of a three layer neural network. Abedinia et al. found that the ISSO outperformed existing optimisation algorithms, such as *particle swarm optimisation* (PSO). Empirical Mode Decomposition (EMD) is used as a preprocessing step in (N. Liu et al. 2014), before short-term load forecasts are generated using a combination of Extended Kalman Filtering (EKF) and an Extreme Learning Machine with Kernel (KELM). The PSO algorithm is used to deduce the optimal parameters of the EMD, EKF and KELM components. Other "hybrid" models combine multiple machine learning methods together. In (Oğcu, Demirel, and Zaim 2012), electricity consumption in Turkey is forecasted using a model that combines a seasonal Support Vector Regression (SVR) model with an ANN. The authors found that the combined model outperformed each of the individual models. Other combinations of existing machine learning models have been tried. An Adaptive Neuro-Fuzzy Inference System (ANFIS) is combined with a feed-forward ANN in (Okumus 2016), and achieves impressive accuracy when applied to wind energy forecasting. Finally, existing statistical methods have been combined with existing machine learning models, producing "hybrid" models that align the closest with the hybrid models discussed in this paper. For example, an ARIMA model is combined with an SVM for short-term load forecasting in (Nie et al. 2012). The linear, basic part of the load is forecasted by the ARIMA component, and the SVM is used to forecast the non-linear sensitive part of the load.

Whilst a multitude of "hybrid" models exist in energy demand forecasting literature, (to the best of our knowledge) Smyl's model is the only model that utilises machine learning techniques to train a statistical model and an LSTM concurrently, using the statistical component to transform a univariate time series into a form suitable for the LSTM.

2.5 Other Models

This review covers just a fraction of the literature on short-term energy demand forecasting that has been published in the last half-century. Several other techniques have featured prominently in the literature, including, but not limited to: fuzzy logic (Kucukali and Baris 2010; Adika and L. Wang 2012), support vector machines (SVM) (Setiawan, Koprinska, and Agelidis 2009; Fattaheian-Dehkordi et al. 2014), and grey prediction (Gao, Zhang, and X. Liu 2012; Lei and Feng 2012). For a comprehensive review of the recent literature published on energy demand forecasting, see (Ghalehkhondabi et al. 2016).

3. EXPLORATORY DATA ANALYSIS

In this section, we present a thorough analysis of the dataset that we used to compare the performance of the hybrid and statistical models.

3.1 Energy Demand Data

Whilst the dataset consists of multiple time series, the *total load actual* series is the primary series considered in this paper, particularly in the statistical forecasting models. This time series represents the total electricity demand in megawatts (MW) and consists of 35,028 individual data points, of which 36 are missing. As so few points are missing, we use simple linear interpolation to estimate them. The hourly data begins at 00:00:00 on 01/01/2015 and ends at 23:00:00 on 30/12/2018. By visually inspecting random samples of the data we identified four distinct weekly patterns, corresponding to the four seasons. Therefore, we split the data into four sections: *winter* - containing the data from each year for the months December, January and February, *spring* - containing the data for March, April and May, *summer* - containing the data for June, July and August, and *autumn* - containing the data for September, October and November. A typical weekly pattern (starting on Monday) for each season is shown in Fig. 1 below. Please note that the units of the x and y axes (*time* and MW, respectively) are omitted throughout the paper, to avoid visual clutter.

Splitting the data in this way creates 16 independent datasets (four seasons for each of the four years), across which we can average. It also enables us to perform per-season analysis. Furthermore, it is hoped that the models will benefit from splitting the data in this way, by allowing them to learn the daily and weekly "seasonal" patterns specific to each season.

We split each of the 16 individual datasets into three regions: a *training* region, a *validation* region, and a *test*

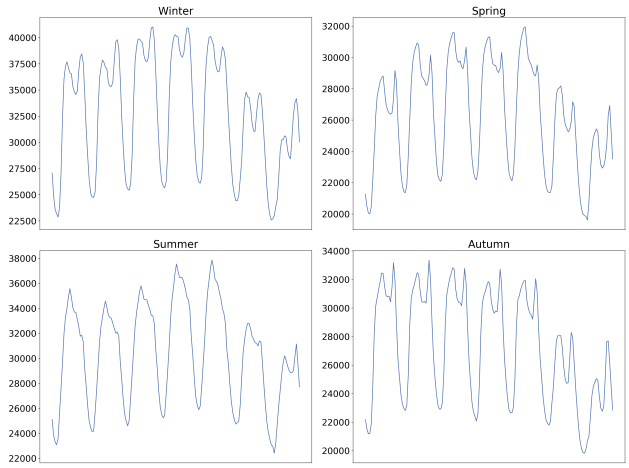

Fig. 1. Typical weekly pattern of each season

region, as is common in literature (Robin John Hyndman and Athanasopoulos 2018). The *validation* region is used to determine (hopefully) optimal values for the hyperparameters of the models, and the *test* region is reserved for the final evaluation and comparison of forecast accuracy. During the final testing, the *validation* region is included in the *training* data. As the data shows both daily and weekly seasonality (discussed in more detail shortly), we feel it prudent to ensure that the forecasting accuracy is averaged over an entire week, when testing. To achieve this, from each of the 16 datasets, 7 further datasets are created. In the first, the final 8 days of data are removed leaving the training set, and the following two 2 days are used as the test set. In the second, the final 7 days of data are removed, and again the following 2 days are used as the test set, and so on. This process is demonstrated in Fig. 2, along with the training/validation/test data split.

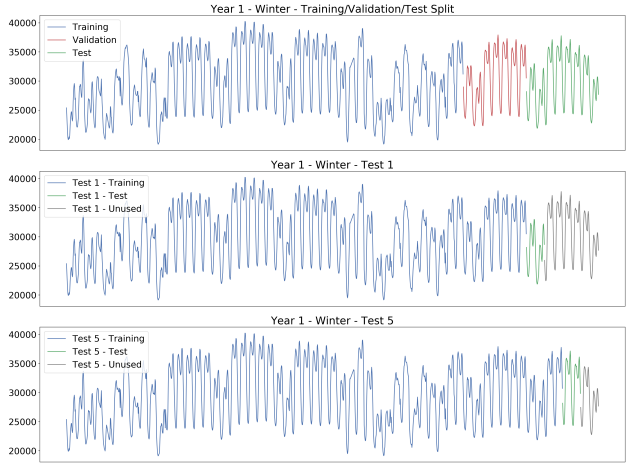

Fig. 2. Training, validation and test data split

For each of the 16 datasets, the performance of a given model is averaged across each of these 7 tests. For each season, the performance is then averaged across each of the years, and these results are recorded. Finally, to calculate an overall measure of accuracy for a model across the entire dataset, the performance of each model is averaged across all four seasons. Throughout this paper, we use $\hat{x}_t$ to denote the fitted values of our model for $t \in [0, T]$ and the forecasted values for $t \in (T, N]$, where N is the number of data points x_t in a given dataset (excluding the unused values), and the first T of these form the *training set*. For $t \in [0, T]$, the difference $x_t - \hat{x}_t$ is referred to as a *residual*, and for $t \in (T, N]$ the difference is referred to as the *(forecast/prediction) error*.

As noted earlier, the entire dataset shows a high degree of seasonality at two frequencies - daily and weekly. As many of the statistical models cannot natively handle seasonal data, the data must first be pre-processed to remove its seasonality. To identify the seasonality analytically, we plotted the autocorrelation function (ACF) of a random year for each of the four seasons. The ACF calculates the correlation between data points that are separated by a certain number of lags. From Fig. 3 the daily seasonality can be seen clearly - represented by the strong positive autocorrelation at lags that are a multiple of 24 and strong negative autocorrelation at lags that are an odd multiple of 12. The shaded blue region represents the 5% significance level, and thus the correlation at these lags is highly significant.

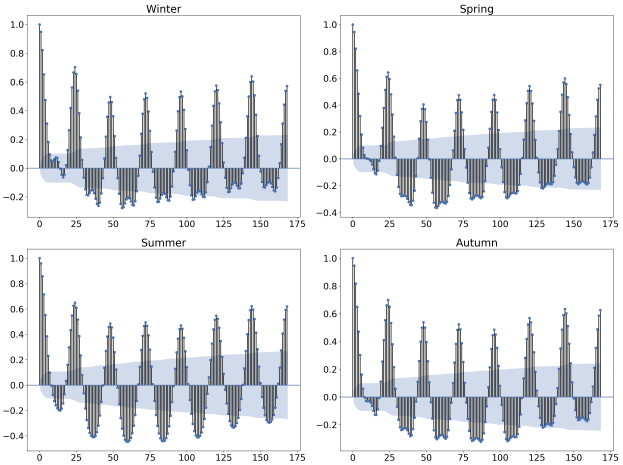

Fig. 3. Autocorrelation functions of the four seasons

To remove the seasonality of the data, we used a method called *classical decomposition*. Inspired by the ES family of models, this process attempts to decompose a time series into four components: a level, a trend, seasonality and random noise. The seasonality can then be removed from the data by dividing or subtracting the seasonal component from the data, depending on the nature of the seasonality (either *multiplicative* or *additive*). As per the M4 Competition, we decided to use *multiplicative* decomposition. We experimented with removing 24 hour and 168 hour seasonality - corresponding to daily and weekly seasonality - and found that when a frequency of 24 is used the effects of the weekly seasonality are still present. Typically, the energy demand is lower at the weekend, which is clearly shown in Fig. 4 where the weekends have been highlighted. The bottom left plot of Fig. 4 shows that using a frequency of 24 fails to account for the weekly seasonality, whereas the bottom right plot shows that such seasonal effects are almost completely removed when a frequency of 168 is used. This plot shows that, once the seasonality has been removed, the data is stationary and resembles a "white noise" process.

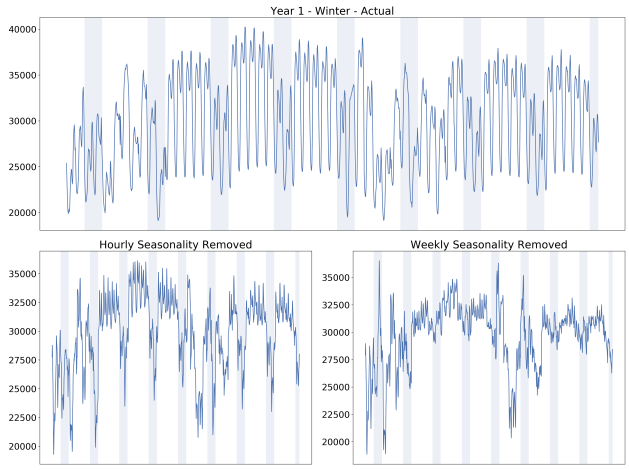

Fig. 4. 24 hour and 168 hour de-seasonalised data

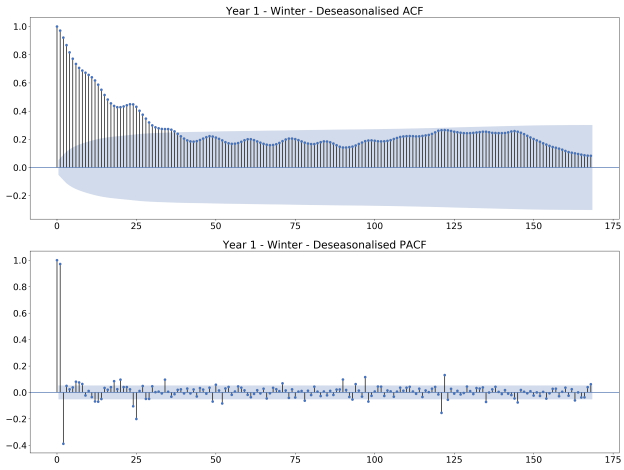

Fig. 5. ACF and PACF of the de-seasonalised data

It is interesting to note that the $1^{st}, 2^{nd}, 6^{th}$ and 7^{th} weeks of this dataset appear to show non-standard behaviour. We believe that this divergence from the norm may be due to national events and holidays, such as the *Fiesta de los Reyes* (Three Kings Festival) on January 6^{th} and the *Moros y Cristianos* (Moors and Christians) Festival celebrated in early February. Whilst possibly detrimental to the individual performance of each model, we believe that the *relative* performance of the models should not be affected, as none of the models have mechanisms to handle such annual events.

Fig. 5 shows the ACF and partial auto-correlation function (PACF) for the winter season of the first year of de-seasonalised data. The PACF gives the partial correlation of the time series with its own lagged values, where the *partial* autocorrelation of lag k is the correlation of the data points x_t and x_{t-k} with the correlation of the data points x_t and x_{t-1} through x_{t-k+1} removed. The sinusoidal behaviour generated by the seasonality has clearly been removed, although the partial autocorrelation of lag 24 is still significant, implying that the de-seasonalisation isn't perfect. This is one of the key issues that the ES-RNN model aims to solve. The significant lag 1 and lag 2 (negative) partial autocorrelation are used when determining the hyper-parameters of the ARIMA and SARIMA models. Stationarity of the input data is also required for several of the statistical models, so this was tested for once the seasonal effects were removed. Consider an autoregressive model with one lag, denoted AR(1):

$$x_t = \beta x_{t-1} + \mu \equiv x_t - \beta x_{t-1} = \mu \equiv (1 - \beta B) x_t = \mu \quad (3.1)$$

where μ is the mean of the data and B is the *backshift* operator, defined by $Bx_t := x_{t-1}$. If $\beta = 1$ the AR(1) process is clearly non-stationary, as the difference in successive terms is given by a constant μ. Such a process is said to have a *unit root*, as when represented in the following form:

$$\phi(B) x_t = \mu, \quad \text{where } \phi(B) = (1 - B\beta) \quad (3.2)$$

the polynomial $\phi(B)$ clearly has a root at $B = 1$ if $\beta = 1$. The Augmented Dickey Fuller test (ADF) assesses whether there is statistical evidence to suggest that our time series is characterised by such a unit root process, and if not we can say (with some degree of confidence) that our time series is stationary (Dickey and Fuller 1979). The ADF was performed on each of the 16 individual de-seasonalised time series, and stationarity was confirmed at the 99% significance level for all.

3.2 Weather Data

The dataset under consideration includes weather information for 5 major Spanish cities. As we only have access to aggregated demand data, we feel that the most viable way to include this information is to simply take the average of the data across the 5 cities. The upper left plot of Fig. 6 shows the relationship between aggregated demand and temperature. The superimposed orange line shows the result of a quadratic polynomial regression on the data. This line highlights the general relationship between temperature and demand: below a certain inflection point energy is used for heating, and above this point energy is used for air conditioning and dehumidification. However, even to the human eye this relationship is non-obvious. Therefore, we extract from the weather data three further metrics: *heating degree* (HD), *cooling degree* (CD) and *latent enthalpy* (LE) which have stronger linear relationships with demand (Ching-Lai Hor, Watson, and S. Majithia 2005). These measures represent the energy required to heat, cool and dehumidify a building, respectively. The HD is the positive difference between the temperature T and a heating reference temperature H_{ref} if $T < H_{ref}$, and 0 if not. Similarly, the CD is the positive difference between the temperature T and a cooling reference temperature C_{ref} if $T > C_{ref}$, and 0 if not. The following formulae summarise this:

$$\begin{aligned} HD &= (H_{ref} - T)\mathbb{1}_{HD},\ \mathbb{1}_{HD} := \begin{cases} 1, & \text{if } T < H_{ref} \\ 0, & \text{otherwise} \end{cases} \\ CD &= (T - C_{ref})\mathbb{1}_{CD},\ \mathbb{1}_{CD} := \begin{cases} 1, & \text{if } T > C_{ref} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \quad (3.3)$$

The values H_{ref} and C_{ref} for Spain are given by the European Environment Agency as 15.5°C and 22°C (E. E. Agency 2019), respectively. Given only the temperature, pressure and relative humidity of moist air, calculating the *specific enthalpy* of moist air - required to calculate the LE - is much more challenging. The calculation itself relies on knowledge of the *saturation vapour pressure* of moist air for the given temperature $P_s(t)$, for which no analytic equation exists. Therefore, we opted to use the

Arden Buck approximation (Buck Research Instruments 2012), which gives $P_s(T)$ in Pascals (Pa) (for $T \geq 0$) as:

$$611.21 \exp\left(\left(18.678 - \frac{T}{234.5}\right)\left(\frac{T}{257.14+T}\right)\right) \tag{3.4}$$

The specific enthalpy E in $\frac{J}{kg}$ of moist air, as a function of the temperature T in °C, pressure ρ in Pa, and relative humidity of the air r as a percentage, is then given by the following formulae:

$$\begin{aligned} E(T,\rho,r) &= c_{pa}T + x(c_{pw}T + h_{we}), \quad \text{where} \\ x &= 0.62198\frac{rP_s(T)}{\rho - rP_s(T)}, \quad \text{and} \\ c_{pa} &= \text{spec. heat of air} = 1.006\frac{kJ}{kg}°C \\ c_{pw} &= \text{spec. heat of vapour} = 1.84\frac{kJ}{kg}°C \\ h_{we} &= \text{evap. heat of water} = 2501\frac{kJ}{kg} \end{aligned} \tag{3.5}$$

Finally, the LE for moist air at a given temperature, pressure and relative humidity can be calculated as the positive difference between the specific enthalpy of the moist air at the given temperature, and the specific enthalpy of the moist air at the reference temperature E_{ref}:

$$\begin{aligned} LE &= (E(T,\rho,r) - E(E_{ref},\rho,r))\mathbb{1}_{LE}, \quad \text{where} \\ \mathbb{1}_{LE} &:= \begin{cases} 1, & \text{if } E(T,\rho,r) > E(E_{ref},\rho,r) \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3.6}$$

E_{ref} is taken to be 25.6°C, as in (Sailor and Muñoz 1997). Working clockwise from the upper left plot, the second, third and fourth plots in Fig. 6 show the aggregated demand plotted against the HD, LE and CD, respectively. As is evident, there is a strong positive correlation between demand and both the CD and the LE, and there is a negative correlation between demand and the HD, albeit a weaker one. We theorised that by including these metrics, which have an emphasised correlation with the aggregated demand, the hybrid ES-RNN model would be able to learn the relationship between weather and demand. The final weather features included in our model are: temperature, humidity, wind speed, HD, CD and LE. Other weather information (such as rainfall and cloud cover) is not included, either due to insufficient data or lack of a statistically significant correlation with the aggregated demand.

4. STATISTICAL BENCHMARK METHODS

In this paper, the performance of the ES-RNN model is compared against multiple statistical benchmark models. These models are:

(1) Naive1
(2) Naive2 **(d)**
(3) NaiveS
(4) Simple Exponential Smoothing (SES) **(d)**
(5) Holt's Method (Holt) **(d)**
(6) Damped Holt's Method (Damped) **(d)**
(7) Holt-Winters' Method (Holt-Winters)
(8) The Combined Method (Comb) **(d)**
(9) ARIMA **(d)**
(10) Seasonal ARIMA (SARIMA)
(11) The Theta Method (Theta) **(d)**

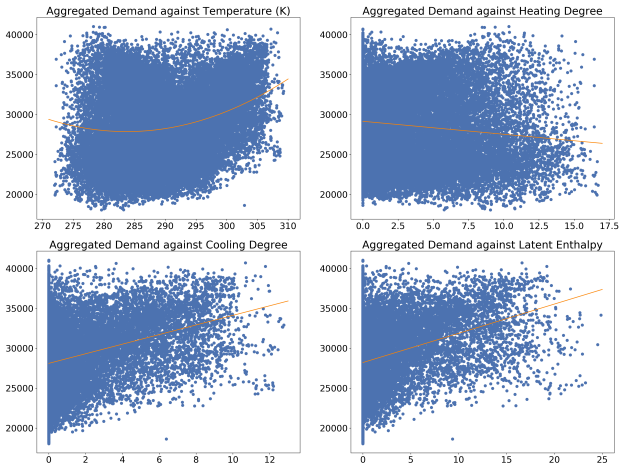

Fig. 6. Relationship of demand and various weather metrics

The decision to use these methods as benchmarks was largely driven by the fact that an almost identical set of methods was used in the M4 Competition. Further methods were included to provide a broader range of forecasts to compare against. The methods marked with **(d)** are the methods which require the input data to be deseasonalised using the classical decomposition method described previously. The forecasted outputs of these methods are then re-seasonalised to give the final output.

4.1 Naive Methods

(1) **Naive1**
The Naive1 method is equivalent to the random walk method. That is, if we are at time step t, the previous values of the time series $x_{t-1}, x_{t-2}, ..., x_0$ give no information about the forecasted time series value $\hat{x}_{t+1}$. Our best estimate for future values is simply the most recent observed time series value. Therefore, this simple model is given by:

$$\hat{x}_{t+h} = x_t \tag{4.1}$$

where h is the forecast horizon.

(2) **Naive2**
The Naive2 method is almost identical to the Naive1 method. The only difference is that the data is deseasonalised first, using the procedure described previously, and then re-seasonalised after the forecast has been made.

(3) **NaiveS**
The NaiveS method is also very similar to the Naive1 method, except that forecasted values are set equal to the last known value of the same period within the seasonal cycle:

$$\hat{x}_{t+h} = x_{t-(s-(h \bmod s))} \tag{4.2}$$

where s is the seasonality of the data.

4.2 Exponential Smoothing Methods

(4) **Simple Exponential Smoothing**
There are two derivations and interpretations of the SES model (Robin John Hyndman and Athanasopoulos 2018), both of which we shall give here. The first is the *weighted*

average form. The forecast at time t for time $t+1$ is given by:

$$\hat{x}_{t+1} = \alpha x_t + (1-\alpha)\hat{x}_t \tag{4.3}$$

where α is the smoothing coefficient. Therefore, the forecasted value is equal to the weighted sum of the current time series value and the previous forecasted value. "Smoothing coefficient" is somewhat of a misnomer, in that the *larger* the value of α the *less* the series is smoothed, as more weight is given to the most recent observation. By recursively substituting the equation into itself, we observe the following:

$$\begin{aligned}\hat{x}_{t+2} &= \alpha x_{t+1} + (1-\alpha)\hat{x}_{t+1} \\ &= \alpha x_{t+1} + (1-\alpha)(\alpha x_t + (1-\alpha)\hat{x}_t) \\ &= \alpha x_{t+1} + \alpha(1-\alpha)x_t + (1-\alpha)^2\hat{x}_t\end{aligned} \tag{4.4}$$

We can prove inductively that this holds generally, and thus:

$$\hat{x}_{t+1} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + \ldots \tag{4.5}$$

In this form, one can see that exponentially decreasing weights are given to less recent time series values, which gives rise to the method's name.

The second representation of the method is the *component form*. The benefit of using this representation is that it can be easily extended to model the more advanced methods in the exponential smoothing family. In this form, SES has only one component which is known as the *level*, denoted l_t. The model is then given by the two equations:

$$\begin{aligned}Forecast: &\quad \hat{x}_{t+h} = l_t \\ Level: &\quad l_t = \alpha x_t + (1-\alpha)l_{t-1}\end{aligned} \tag{4.6}$$

It is not difficult to see that this representation of the model is equivalent to the previous representation. The *forecast* equation sets the forecasted value equal to the current level, and the *level* equation calculates the new level. The new level is a weighted average of the most recent time series value (x_t) and the previous level (l_{t-1}). For multi-step forecasts the forecasted values are all equal to the final level of the training data, and so when plotted the forecast will be a horizontal line. As a consequence, SES is only suitable for time series without seasonality or an underlying trend, and so the input data must be de-seasonalised before this method can be applied.

(5) **Holt's Method**
Holt extended SES by including an equation to model an underlying linear trend in the data:

$$\begin{aligned}Forecast: &\quad \hat{x}_{t+h} = l_t + hb_t \\ Level: &\quad l_t = \alpha x_t + (1-\alpha)(l_{t-1} + b_{t-1}) \\ Trend: &\quad b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}\end{aligned} \tag{4.7}$$

In Holt's model, the *level* equation first updates the level, with the new level set equal to a weighted combination of the most recent time series value (x_t) and the previous level plus a single time step in the direction of the previous trend ($l_{t-1} + b_{t-1}$). The *trend* equation then updates the trend, by setting the new trend equal to a weighted combination of the difference of the most recent two levels ($l_t - l_{t-1}$, an estimate of the local trend) and the previous trend (b_{t-1}). Two smoothing coefficients are now required, α for the level and β for the trend. To generate a h-step forecast, h multiplied by the final trend is added to the final level, to produce a linear forecast with gradient b_t. Similarly to SES, Holt's method is unable to model seasonality, and so the input data must be de-seasonalised before this method can be applied.

(6) **Damped Holt's Method**
A variation of Holt's method was introduced in (Gardner and McKenzie 1985), which gradually dampens the linear trend to a horizontal line at some point in the future. This modification was motivated by empirical evidence that indicated that a constant linear trend was unsuitable for longer term forecasts. The damped method introduces a *damping* constant $0 < \phi < 1$, which controls the rate at which the linear trend decays to the horizontal.

$$\begin{aligned}Forecast: &\quad \hat{x}_{t+h} = l_t + (\phi + \phi^2 + \ldots + \phi^h)b_t \\ Level: &\quad l_t = \alpha x_t + (1-\alpha)(l_{t-1} + \phi b_{t-1}) \\ Trend: &\quad b_t = \beta(l_t - l_{t-1}) + (1-\beta)\phi b_{t-1}\end{aligned} \tag{4.8}$$

As $h \to \infty$, standard mathematical theory tells us that the geometric series $\phi + \phi^2 + \ldots + \phi^h$ converges to $\frac{\phi}{1-\phi}$ for $0 < \phi < 1$. Therefore, as the forecasting horizon increases, the linear trend does indeed decay to a constant. Typically $0.8 < \phi < 0.98$.

(7) **Holt-Winters' Method**
Holt-Winters' method further expands the exponential smoothing family to handle time series with distinct seasonality (at one frequency). There are in fact two variants of this model, an *additive* variant and a *multiplicative* variant. The naming of these two variants pertains to how the seasonality is handled in the model. We implement the *multiplicative* variant, as Holt-Winters' method with multiplicative seasonality has been widely used historically (James W. Taylor 2003). This variant of the model is given by the following equations:

$$\begin{aligned}Forecast: &\quad \hat{x}_{t+h} = (l_t + hb_t)s_{t+h-m(k+1)} \\ Level: &\quad l_t = \alpha\frac{x_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \\ Trend: &\quad b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ Seasonal: &\quad s_t = \gamma\frac{x_t}{l_{t-1} + b_{t-1}} + (1-\gamma)s_{t-m}\end{aligned} \tag{4.9}$$

where m is the frequency of the seasonality, and $k = \lfloor\frac{h-1}{m}\rfloor$. The *level* equation updates the level, with the new level set equal to a weighted linear combination of the previous level plus the linear trend ($l_{t-1} + b_{t-1}$) and the most recent seasonally-adjusted time series value ($\frac{x_t}{s_{t-m}}$). The trend is updated as in Holt's method. Finally, the *seasonal* equation updates the seasonality value, and sets the new seasonality equal to a weighted average of the seasonal component of the most recent time series value ($\frac{x_t}{l_{t-1}+b_{t-1}}$) and the corresponding seasonal value of the previous cycle (s_{t-m}, the seasonality value m time periods ago). It is worth noting that it is simple to include a damped linear trend in Holt-Winters' method too.

(8) **The *Comb* Method**
The Comb method is simply the arithmetic average of the SES, Holt and Damped Holt exponential smoothing models. As has been demonstrated empirically in numerous studies, combining the forecasts of multiple methods often improves the accuracy of the overall forecast (Clemen 1989).

To fit an exponential smoothing model to the time series, we assign values to the smoothing coefficients such that the

sum of the square of the residuals (denoted SSE, for *sum of the squared errors*) is minimised. That is, we perform the following non-linear optimisation:

$$\begin{aligned} \text{minimise } SSE &= \sum_{t=0}^{T} (x_t - \hat{x_t})^2 \\ s.t \quad & 0 \leq \alpha, \beta, \gamma \leq 1 \end{aligned} \tag{4.10}$$

As this is a non-linear optimisation problem, numerical methods must be employed and often a non-optimum solution will be found. How to initialise exponential smoothing models is a source of contention in forecasting literature (Makridakis, Wheelwright, and Rob Hyndman 1984). We use an initialisation method based on the classical seasonal decomposition method discussed in Section 3. A moving average is fitted to the first three cycles of data, and then each data point is divided by its corresponding smoothed value to get a de-trended value. The first m seasonal values are then calculated as the average of the first 3 values in their corresponding season. We then divide the first m time series values by the first m seasonal values to remove the seasonal effects, and fit a line $y = m^*x + c^*$ through these de-seasonalised values. Finally, we let $l_0 = c^*$, $b_0 = m^*$. This method is simple, initialises only the values that are required, and prevents the seasonal patterns in the data from affecting the initial values.

4.3 ARIMA Methods

(9) **ARIMA**

To keep the model description terse, we shall use the standard *backshift* notation seen in many text books. The backshift operator, B, has the effect of changing the time period t to time period $t-1$. Thus $Bx_t = x_{t-1}$ and $B^2x_t = x_{t-2}$. Using the backshift operator, first order differencing can be written as: $x'_t := x_t - x_{t-1} = x_t - Bx_t = (1-B)x_t$. It is easy to show inductively that a dth-order difference can be written as $(1-B)^d x_t$. If seasonal differencing is required, then the seasonal backshift operator can be used: $x^s_t := x_t - x_{t-s} = x_t - B^s x_t = (1-B^s)x_t$. The most general form of an ARIMA model is:

$$\begin{aligned} x'_t = c &+ \phi_1 x'_{t-1} + \ldots + \phi_p x'_{t-p} \\ &+ \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} \end{aligned} \tag{4.11}$$

where x'_t is the differenced series (possibly d times) and c is a constant. The model assumes that the current (differenced) time series value (x'_t) is a linear combination of the p previous time series values ($\phi_1 x'_{t-1} + \ldots + \phi_p x'_{t-p}$, the *autoregressive* part) and the previous q residuals ($\theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q}$, the *moving average* part). As the model is applied to (possibly) differenced data, the model is said to be *integrated*. Using backshift notation, it can be shown equation (4.11) can be written as:

$$\begin{aligned} &(1 - \phi_1 B - \ldots - \phi_p B^P)(1-B)^d x_t \\ &\quad = (1 + \theta_1 B + \ldots + \theta_q B^q)\epsilon_t + c \end{aligned} \tag{4.12}$$

or even more succinctly:

$$\phi_p(B)\nabla^d(B)x_t = \theta_q(B)\epsilon_t + c \tag{4.13}$$

where:

$$\begin{aligned} \phi_p(B) &= (1 - \phi_1 B - \ldots - \phi_p B^P) \\ \nabla^d(B) &= (1-B)^d \\ \theta_q(B) &= (1 + \theta_1 B + \ldots + \theta_q B^q) \end{aligned} \tag{4.14}$$

We call this model an ARIMA(p, d, q) model, where p is the order of the autoregressive component, d is the degree of differencing required, and q is the order of the moving average component. The three-step process for "correctly" specifying an ARIMA model is rigorous (Barak and Sadegh 2016):

(1) *Model Identification*
To determine d, the order of differencing required, either the ACF plot can be consulted as described in Section 3, or the (augmented) Dicky-Fuller unit root test can be performed (Dickey and Fuller 1979). To determine p and q, the number of autoregressive and moving average terms to use, the ACF and PACF plots can be used as guidance. In (Robin John Hyndman and Athanasopoulos 2018), Hyndman suggests using: an ARIMA$(p, d, 0)$ model if the ACF is exponentially decaying or sinusoidal, and the PACF is significant up to (but not after) lag p, or an ARIMA$(0, d, q)$ model if the PACF is exponentially decaying or sinusoidal, and the ACF is significant up to (but not after) lag q. The motivation behind these suggestions comes from looking at the ACF/PACF plots of pure $AR(p)$ and $MA(q)$ processes, and noting their structure. Note that it is difficult to identify a mixed model ($p, q > 0$) directly from the ACF and PACF (Hintze 2007), due to their overlapping effects.

(2) *Parameter Estimation*
Maximum likelihood estimation (MLE) is then used to fit the model to the data, by determining the values of the parameters $\phi_i, i \in \{1, p\}$ and $\theta_j, j \in \{1, q\}$ that maximise the probability of the model generating the data that we have observed. Once the model's parameters have been determined, the Akaike's Information Criterion (AIC) of the model is calculated, which is a measure of the relative quality of the model for the given time series. The values of p and q are systematically modified, the model refitted, and the AIC recalculated until the AIC no longer decreases. Note that in this way a mixed model may be selected.

(3) *Diagnostic Checking*
Once we have settled on values for p, d, and q and fitted the parameters of the model, we must check the residuals of the model by plotting the ACF of the residuals, or by performing the Llung-Box test (Ljung and G. E. P. Box 1978). If the ACF plot of the residuals looks like a "white noise" process, or if we accept the null hypothesis of the Llung-Box test, then we can be confident that no additional information remains in the residuals of the model.

At this point we can be confident in our model, and generate our forecasts. Note that in its *vanilla* form, ARIMA cannot handle seasonal effects and thus the input data must be de-seasonalised before the model can be used. To generate a forecast, we simply rearrange the equation so that only x_t remains on the left hand side, replace t with $t+1$, and drop any "future" residual values.

(10) **Seasonal ARIMA**

The ARIMA model can be extended easily to handle seasonality natively. In the seasonal ARIMA model, seasonal differencing can be performed to make the data stationary, and seasonal autoregressive and moving average terms can be included. The general form of the model is thus:

$$\begin{aligned}\phi_p(B)\Phi_P(B^s)\nabla^d(B)\nabla^D(B^s)x_t \\ = \theta_q(B)\Theta_Q(B^s)\epsilon_t + c\end{aligned} \tag{4.15}$$

where:

$$\begin{aligned}\phi_p(B) &= (1 - \phi_1 B - \ldots - \phi_p B^P) \\ \nabla^d(B) &= (1 - B)^d \\ \theta_q(B) &= (1 + \theta_1 B + \ldots + \theta_q B^q) \\ \Phi_P(B^s) &= (1 - \Phi_1 B^s - \ldots - \Phi_P B^{sP}) \\ \nabla^D(B^s) &= (1 - B^s)^D \\ \Theta_Q(B^s) &= (1 + \Theta_1 B^s + \ldots + \Theta_Q B^{sQ})\end{aligned} \tag{4.16}$$

We call this model an ARIMA$(p, d, q)(P, D, Q)_s$ model, where p, d, q are as before, P is the number of seasonal autoregressive terms to include, D is the degree of seasonal differencing required, Q is the number of seasonal moving average terms to include, and s is the frequency of the seasonality. One important detail to note is that Box and Jenkins explain that the maximum value typically required for any of the hyperparameters p, d, q, P, D, and Q is 2 (G. Box and Jenkins 1976), which ensures that the model retains a heuristic interpretation. As the three step processing of specifying (S)ARIMA models can be time consuming, computational techniques have been developed to automate the process systematically, such as the Hyndman-Khandakar algorithm (Rob J Hyndman and Khandakar 2008). The algorithm uses a stepwise search to traverse the model space, aiming to find the model which minimises the AIC. We use the pmdarima module (Smith et al. 2017), which implements the above algorithm, to automatically identify an optimum SARIMA model.

4.4 Other Statistical Models

(11) **The Theta Method**
The Theta method has been included as a benchmark method as it was the winning entry to the M3 competition (Makridakis and Michèle Hibon 2000), the previous iteration of M competitions organised primarily by Spyros Makridakis. The intuition behind the model given in (Makridakis and Michèle Hibon 2000) is that by modifying the local curvature of the time series one can magnify the effects of long-term trends and minimise short-term noise. The theta-coefficient is applied directly to the second difference of our time series (x_t) to generate so-called *theta lines* $L(\theta)$, with:

$$L''(\theta) = \theta x'', \quad \text{where } x'' = x_t - 2x_{t-1} + x_{t-2} \tag{4.17}$$

Increasing the value of θ increases the local curvature of the time series, thus magnifying short-term fluctuations. Conversely, decreasing the value of θ dampens short-term fluctuations, to a minimum at $\theta = 0$. It can be shown that $L(\theta = 0)$ is simply the line given by standard least squares linear regression. Ignoring de/re-seasonalisation, forecasting using the Theta method has three distinct stages:

(1) *Decomposition*: The time series is decomposed into two theta lines, $L(\theta = 0)$ and $L(\theta = 2)$.
(2) *Extrapolation*: $L(\theta = 0)$ is extrapolated in the standard way, and $L(\theta = 2)$ is extrapolated using simple exponential smoothing.
(3) *Combination*: Forecasts are produced by taking the simple arithmetic average of the extrapolated $L(\theta = 0)$ and $L(\theta = 2)$ *theta lines*.

Whilst relatively intuitive to understand, specifying the model analytically required several pages of convoluted mathematical derivation in the original paper. Hyndman et al. later showed that the model has an equivalent, much simpler form (R.J. Hyndman and Billah 2001). The Theta model is actually just SES with an added trend and constant, where the slope of the trend is half that of the fitted trend line through the original time series.

5. THE HYBRID ES-RNN MODEL

We now introduce the hybrid ES-RNN model that we are investigating in this paper, developed by Slawek Smyl of Uber Technologies (S. Smyl 2020). As mentioned previously, this model was the winning entrant of the recent M4 Competition. Before continuing, we feel it is pertinent to discuss the Spanish TSO dataset further, on which we perform our analysis. Within the TSO dataset, the *total load actual* time series is the primary time series considered - particularly for the univariate statistical models - as this represents the actual hourly demand for energy. However, there are multiple other time series in the dataset, which represent: the amount of energy generated, stratified by energy type (wind, coal, etc); the forecasts for wind and solar generation; and the day-ahead and actual price of the energy. The existence of multiple time series in the dataset was one of the driving factors in choosing to investigate Smyl's hybrid model, as the model makes use of multiple time series to facilitate cross-learning in the dilated LSTM. The key idea of the hybrid model is that LSTM learns across multiple time series, in the hope that these related time series (such as the generation and pricing data) contain information that will aid the prediction of the total energy demand.

5.1 Model Overview

We first present a "high-level" overview of the model, before discussing the finer implementation details in the following section. The model has three distinct features:

(1) **Exponential Smoothing Component**
It has been shown that univariate machine learning models are often unable to learn complex seasonal patterns, such as those present in hourly energy data (Makridakis, Spiliotis, and Assimakopoulos 2018). This necessitates de-seasonalising the data, which is often done through classical seasonal decomposition, as discussed in Section 3. Classical seasonal decomposition separates the pre-processing from the forecasting, and averages the seasonal effects across the entire input dataset. Whilst often an adequate solution, this averaging process fails to take into account local variations in the seasonality towards the end of the training set, which tend to have the highest correlation with the actual values we wish to forecast. Instead, the ES-RNN model draws inspiration from the ES family of statistical models to de-seasonalise the data - in particular Holt-Winters' method, which has the inate ability to handle seasonal effects. De-seasonalisation is performed by first fitting an *individual* ES model to each of the time series in the dataset. In Slawek's paper, the following simplified version of Holt-Winters' method (see Section 4) is used:

$$\begin{aligned} \textit{Level}: \quad & l_t = \alpha \frac{x_t}{s_t} + (1-\alpha) l_{t-1} \\ \textit{Seasonal}: \quad & s_{t+m} = \gamma \frac{x_t}{l_{t-1}} + (1-\gamma) s_t \end{aligned} \tag{5.1}$$

where m is the seasonality of the data. As each of the 16 sub-datasets were statistically proven to be stationary, we also deem it unnecessary to include a trend component. To fit the model, the values of the smoothing coefficients α and β must be determined, along with the initial seasonality values $s_1, s_2, ..., s_m$. By fitting this model to the dataset, a level value l_t and seasonality value s_{t+m} is generated for each data point x_t within the time series. Note that the seasonal value corresponding to the data point x_t is s_{t+m}, as the method requires m initial seasonality values. To de-seasonalise the data, each data point is simply divided by its corresponding seasonality value. A non-linear optimisation algorithm such as the limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) (D. C. Liu and Nocedal 1989) algorithm would typically be used to fit the model in a statistical setting. In the ES-RNN model, the parameters are deduced using stochastic gradient descent in parallel with the parameters of the LSTM. Each of the time series in the dataset is allocated individual smoothing coefficients and initial seasonality values, and thus these parameters are henceforth referred to as the *local* (i.e to each time series) parameters of the model. Hopefully it is apparent that de-seasonalisation is an integral component of the model, as opposed to a pre-processing step which may adversely affect the quality of the forecasts.

(2) **Long Short-Term Memory Component**
Following de-seasonalisation, the time series are inputted into the RNN. It is hoped that the RNN will learn any underlying seasonal patterns in the data that were not removed in the de-seasonalising process, as well as any short-term trends that may be present locally. The specific RNN architecture used is a multi-layer dilated LSTM network (Chang et al. 2017), with residual connections between certain pairs of layers (He et al. 2015). A dilated LSTM (dLSTM) is used to aid the model in detecting seasonal patterns of different frequencies, and the details of the architecture are discussed in the following section. Whilst the ES smoothing coefficients are *local* to each time series, the parameters of the dLSTM are *global*, in the sense that they are updated every time each of the time series is inputted into the network. The output of the dLSTM then passes through a simple *tanh* non-linear layer, before passing through a final linear layer. The linear layer serves to map the output from the *tanh* layer, which has the same dimension as the output of the dLSTM, to the correct output dimension - the forecast horizon.

(3) **Ensembling**
Ensembling is the practice of combining the forecasts of multiple models in order to improve the overall accuracy of the final forecast. It has been proven theoretically that the mean squared forecast error (MSFE) of the arithmetic average of multiple forecasts decreases as the number of models increases (Chan and Pauwels 2018). For the M4 Competition, Smyl used an ensembling technique called *Ensemble of Specialists*, which he developed for datasets containing a number of series from unknown sources (Slawek Smyl 2017). As the dataset under consideration in this paper has far fewer time series than the one used in the M4 Competition, and all of which are inherently related, we believe that such a complex ensembling procedure is unnecessary. Instead, we simply train the model multiple times and take an average of the forecasts. The random initialisation of some of the model's parameters effectively guarantees at least a small amount of variation in the model's forecasts. A second level of ensembling was also used by Smyl for the M4 Competition, which we do use. In order to mitigate the effects of overfitting, forecasts are generated by the model after each of the final five training epochs. The final forecast produced by the model after a complete training cycle is the average of these five forecasts.

From this description, it is hopefully evident that the model is both hybrid - as it combines elements of existing statistical and machine learning methods, and hierarchical - as it combines a global component, learnt across multiple time series, with a local component that is individual to each time series. Furthermore, the model also makes extensive use of the proven benefits of ensembling.

5.2 Model Details

Fitting the ES Component and Training the Model
Consider a single time series x_t in the dataset, with smoothing coefficients α and γ, and initial seasonality values $s_1, s_2, ..., s_m$, where m is the seasonality of the data. As explained in the previous section, these are the *local* parameters which are unique to this time series. By looping through the time series and applying equation (5.1) given in the previous sub-section, a level value l_t and seasonal value s_{t+m} are calculated for each data point in the time series. The first, second and third plots in Fig. 7 show the original data, the level values and the seasonal values, respectively.

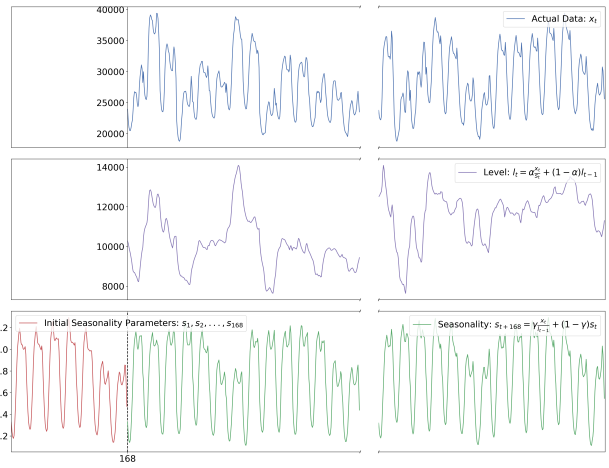

Fig. 7. Fitting the ES component

The model utilises *supervised learning* to deduce the optimal values of its parameters. Therefore, the input time series x_t, following de-seasonalisation, must be converted into a batch of $(input, target)$ pairs. A *sliding window* approach is employed to generate such pairs. An input window of two weeks (336 hours) is placed over the data, and the following two days (48 hours) of data forms the

corresponding target. Each data point x_t within the input and target windows is then divided by its corresponding seasonal value to remove the seasonal effects. Each data point is then divided by the level corresponding to the *final* value x_t of the input window, to normalise the input and target with respect to this final level. We denote these de-seasonalised and normalised input and target values y_t. Note that each data point y_t is simply a function of the data points $x_k, k \in 1, 2, ..., t$ and the parameters α, γ and $s_1, ..., s_m$. This is very important, as it is enables us to update the values of the parameters through backpropagation (Rumelhart, Geoffrey E. Hinton, and Williams 1986). The input and target windows are then moved forward one data point (one hour), and the next $(input, target)$ pair is formed. This process repeats across the length of the time series. The input sequence of each pair is then fed into the dLSTM, which produces an output that, after passing through a non-linear *tanh* layer and then a fully connected linear layer, is of the correct length (48 hours). For simplicity, we shall refer to this output as simply the dLSTM output. This process is demonstrated in Fig. 8. The first of the two plots shows how the sliding window is used to generate an $(input, target)$ pair. The second plot shows the input and target after they have been de-seasonalised and normalised, and also the output of the dLSTM. The dLSTM output is then compared to the (de-seasonalised and normalised) target, and a loss value for the training example is calculated. During backtesting, Smyl found that the model tended to have a positive bias. To counter this, he used a *pinball loss* function:

$$L_t = \begin{cases} \tau(y_t - \hat{y}_t), & \text{if } y_t \geq \hat{y}_t \\ \tau(\hat{y}_t - y_t), & \text{if } y_t < \hat{y}_t \end{cases} \tag{5.2}$$

with a τ value a little smaller than 0.5. We also use this *pinball loss* function, and in our experiments opt for $\tau = 0.49$. The loss for the entire batch of $(input, target)$ pairs is then calculated, and the values of the local parameters α, γ and $s_1, s_2, ..., s_m$ are updated, along with the global parameters of the dLSTM, using the standard backpropagation of gradients algorithm (Rumelhart, Geoffrey E. Hinton, and Williams 1986). This process is then repeated for each of the individual time series in the data set. Therefore, the dLSTM is trained across all of the time series, whereas the ES parameters remain local to each individual time series. Please note that in Fig. 7, Fig. 8 and Fig. 12 the model has not been trained.

LSTM Architecture

Traditional *feed-forward* networks lack any notion of *state*; other than the information inherent in the value of the weights of the network, no information persists between inputs. RNNs address this issue by maintaining a *hidden state* within the network, which gives the network temporal awareness. Consider the first 8 values of an input sequence from a given $(input, target)$ pair, denoted x_t for $t \in 1, 2, ..., 8$. We can visualise a single layer of an RNN as a block B, which accepts the input values of our sequence x_t one at a time, and for each input value produces an output y_t. With each new input x_t, the hidden state h_t of the block is updated and this value is fed back into the block when the next input value is inputted. The diagram to the left in Fig. 9 visualises this succinctly. It is perhaps more natural, however, to *unroll* the network, and think of a layer of an RNN as multiple copies of the same block B,

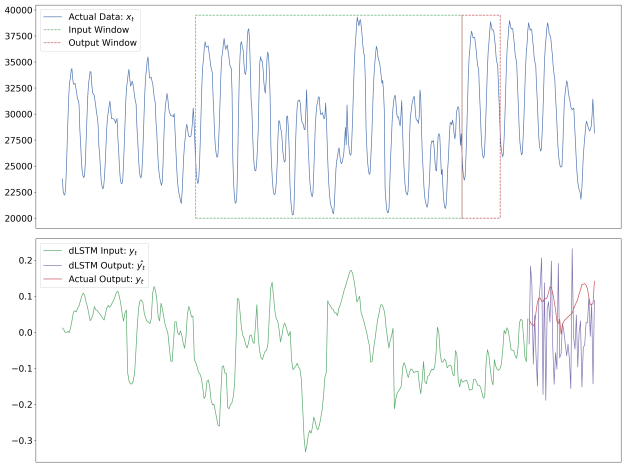

Fig. 8. Generating $(input, target)$ pairs using a sliding window

each passing a message to its successor as the individual values of the sequence are inputted. The diagram to the right in Fig. 9 demonstrates this concept.

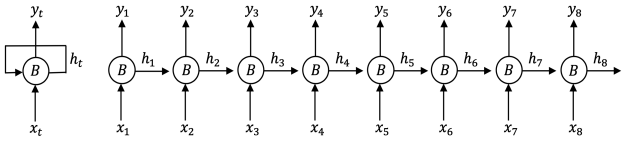

Fig. 9. Unrolling a recurrent neural network

Vanilla RNNs are often unable to learn long-term dependencies in input sequences, due to the *vanishing gradients* problem (Bengio, Simard, and Frasconi 1994). The LSTM is a type of RNN with a specific architecture for the block B that aims to remedy this issue. In an LSTM layer, the block B maintains two states: the hidden state h_t and the cell state c_t. The intuition behind the block is as follows: the cell state c_t allows information to pass easily from early inputs to later outputs, and the various *gates* - *sigmoid* and *tanh* layers - in the block control what is "remembered" or "forgotten" from the cell state with each new input x_t. One of the gates is also used to control how much information from the cell state and input is assigned to the hidden state h_t at each time step, and the value of the hidden state h_t is assigned to the output value y_t at each time step. A superb blog post that explains the intricacies of the LSTM block can be found at (Olah 2015).

The dilated LSTM (dLSTM) aims to further assist the network in learning long-term and seasonal dependencies in the input sequence (Chang et al. 2017). Suppose, for simplicity, that our input sequence x_t has seasonality of order 2. That is, every value x_t is highly correlated with the value two time steps ago x_{t-2}. Now also suppose that, instead of feeding the hidden state h_t of every block into the following block as usual (as shown at the bottom of Fig. 10), we instead skip a block and feed it directly into the block two after, in order to assist the network in learning the dependence between the values. Such a connection is referred to as a *dilated* connection of order $d = 2$, and is shown in the middle of Fig. 10. This functionality can be achieved through manipulation of the input series,

requiring no direct modification to the structure of the LSTM block itself. We simply convert our input sequence x_t for $t \in 1, 2, ..., 8$ into two input sequences x_1, x_3, x_5, x_7 and x_2, x_4, x_6, x_8, and use a standard LSTM block. This equivalence is shown at the top of Fig. 10.

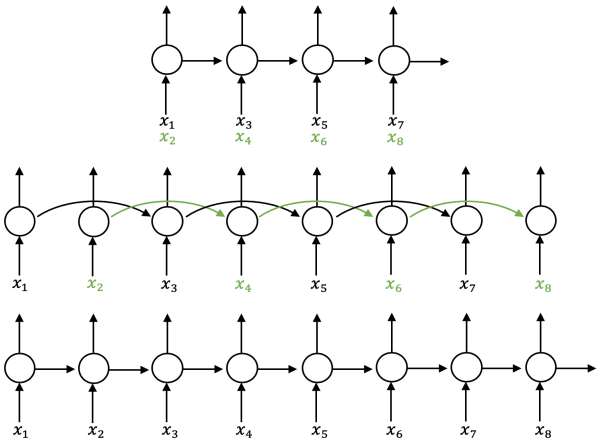

Fig. 10. Visualisation of the dilated LSTM

In the hybrid model, we stack four dLSTM layers together, with dilations of $1, 4, 24$ and 168, respectively. The intuition behind this decision is that we hoped each layer would be able to learn the seasonal dependencies corresponding to its given dilation. The final feature in our implementation of the dLSTM network is the inclusion of *ResNet*-style shortcuts between certain layers (He et al. 2015). Counter-intuitively, it has been noted that the training accuracy of *deeper* neural networks - neural networks with more layers - is often worse than those of neural networks with fewer layers (He et al. 2015). This is unexpected, as deeper neural networks should simply learn the identity mapping for any redundant layers, and thus achieve at least the same accuracy as a network with fewer layers. He et al. hypothesised that it may be easier for a complicated non-linear layer (such as an LSTM block) to learn the zero-mapping rather than the identity mapping, if that is indeed the "correct" mapping the layer should learn. We can force a layer to learn the zero mapping instead of the identity mapping by simply adding the input x directly to the output of the layer *as well as* passing it through the layer itself. Such a connection is known as a *residual* connection, and He et al. proved empirically that adding such connections improves the training accuracy of deep networks.

Complete Neural Network Architecture

For clarity, Fig. 11 shows a slightly simplified version of the entire neural network used in the ES-RNN model, but still demonstrates the necessary details. The network shown here is a stack of three dLSTM layers, with dilations of $1, 2$ and 4 respectively, whereas four layers with dilations of $1, 4, 24$ and 168 are used in the actual network. The network in Fig. 11 also demonstrates the addition of a *residual* connection between the input to the 1st layer and the output of the 2nd layer, whereas in our network we implement a residual connection between the input to the 3rd layer and the output of the 4th layer. Finally, Fig. 11 shows how the output from the dLSTM (a vector of length 40, corresponding to the size of the hidden state of the network) passes through a non-linear *tanh* layer (labelled T), and then through a fully-connected linear layer (labelled L), to generate a 48-hour output vector.

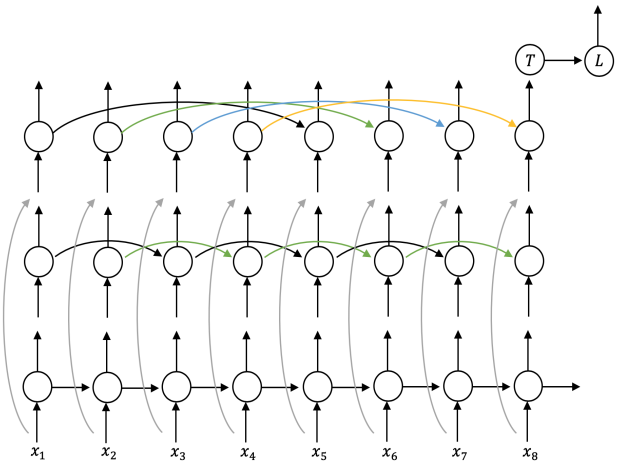

Fig. 11. Simplified neural network architecture

Generating Forecasts

At the end of every training epoch, the vectors of levels and seasonality values for the *total load actual* time series are saved internally in the model. To generate a forecast, the final two weeks of training data for this series are first divided by their corresponding saved seasonality values, and then by the final level value, and then input into the dLSTM. The final level value is extrapolated, and the most recent corresponding seasonality values are repeated. The output of the dLSTM is then multiplied by the extrapolated level and repeated seasonality values to generate a final forecast. This process is demonstrated in Fig. 12. The first plot shows the final two weeks of training data (the input) and the first 48 hours of test data (the actual output). The second plot shows the level values corresponding to the input, and the extrapolated level values. The third plot shows the seasonality values corresponding to the input, and which of these values are repeated to provide the seasonality values for the forecast. As we opt to use 168 hour seasonality, the seasonality values from the previous week are the values that are repeated - shown in the green dashed box. Finally, the fourth plot shows the input into and output of the dLSTM. The dLSTM output is multiplied by the corresponding level and seasonality values to give the final forecast, which is shown in the first plot.

6. EXTENSION OF THE ES-RNN MODEL

In order to differentiate between different variants of the model, we now refer to Smyl's original formulation of the model as **ES-RNN-S**. As discussed in Section 1, the key contribution of this paper is the inclusion of the weather information in the model. Once the weather data has been aggregated and the additional metrics (*heating degree*, *cooling degree* and *latent enthalpy*) calculated, including the weather information as an additional *feature* of the input into the dLSTM is very straightforward. For each data point in the time series, we simply extend the scalar value to a multi-dimensional vector, where the first

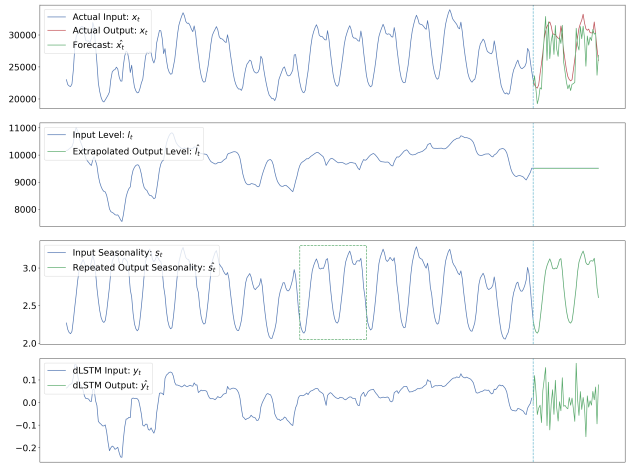

Fig. 12. Procedure for generating a forecast

dimension of the vector is the demand data itself and the following dimensions are the various (scaled) weather metrics. This process is shown in Fig. 13, where t_n and h_n represent the temperature and humidity, respectively (the other weather features have been omitted for simplicity). The diagram at the bottom of the figure represents the original input, and the diagram in the middle represents the data with the weather information added.

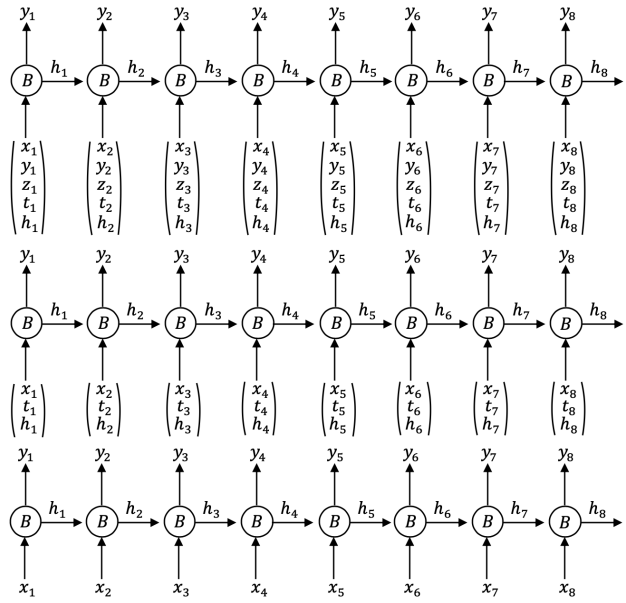

Fig. 13. Adding extra features to the demand data

The second modification we make to the model is to change how the additional related time series in the dataset are utilised - such as the generation and pricing information. In the **ES-RNN-S** model, in every training epoch, each of the time series is inputted one at a time, and the parameters of the network are updated with respect to the loss of this time series. In the context of the M4 Competition this makes sense, as the model was required to generate forecasts for each time series and its overall performance was calculated as the average error of these forecasts. However, we are only interested in the forecasting accuracy of the aggregated demand, and not of the other time series in the dataset. Therefore, we believe a more natural method (and arguably the more "standard" method) of using this information is to include it as additional features of the input into the dLSTM, mirroring how the weather information is included. In every training epoch, an individual ES model is still fitted to each of the time series in the dataset in order to deseasonalise and normalise the data. However, instead of being inputted into the dLSTM individually, the time series are concatenated into a single vector, along with the weather information, to form a multi-feature input into the dLSTM. This process is represented in the top diagram of Fig. 13, where y_n and z_n represent other time series within the dataset. In this way, de-seasonalisation through exponential smoothing still remains a fundamental component of the model, but the dLSTM is focused towards learning the mapping required to generate good forecasts of the aggregated energy demand only. We denote this variation of the model - with weather information and the additional time series included as extra features into the dLSTM - as **ES-RNN-IW**, where the **I** denotes "Ingram" and the **W** denotes the inclusion of weather information.

We also implement another variant of the model, to assess whether or not the other time series in the dataset (such as the generation data) actually contain any information that improves the accuracy of the forecast. In this simple variation, which we denote **ES-RNN-D** (where **D** denotes "demand"), only the *total load actual* time series is used during training. For completeness, we also include the variants of each of the three models with/without the weather information, to assess the predictive power of the weather information itself. In total, this gives 6 independent models:

(1) **ES-RNN-S** - Smyl's original model, where the time series are individually inputted into the dLSTM, without weather information
(2) **ES-RNN-SW** - Smyl's original model, where the time series are individually inputted into the dLSTM, with weather information
(3) **ES-RNN-D** - Smyl's original model, trained only using the *total load actual* time series, without weather information
(4) **ES-RNN-DW** - Smyl's original model, trained only using the *total load actual* time series, with weather information
(5) **ES-RNN-I** - our modified model, where the other related time series are included as additional features of the input into the dLSTM, without weather information
(6) **ES-RNN-IW** - our modified model, where the other related time series are included as additional features of the input into the dLSTM, with weather information

7. EXPERIMENTS AND RESULTS

To implement the models, we use Python 3.7.3 (Van Rossum and Drake 2009). For the statistical models, the statsmodels module (Seabold and Perktold 2010) is used extensively. For the machine learning and hybrid models, the PyTorch module (Paszke et al. 2019) is used. PyTorch was chosen over other prominent machine-learning libraries due to its use of a *dynamic computation graph*

under the hood. In essence, this means that the computation graph is re-generated with every forward pass through the network, which facilitates the implementation of networks such as the one in the ES-RNN-S model, which optimises a different set of parameters with each iteration. Before continuing with the experimental results, we want to briefly explain our choice of error metrics.

7.1 Error Measures

Historically, error measurements were typically reported in terms of the MAPE. However, four issues with the MAPE were highlighted in (Makridakis 1993). These issues include the non-symmetric nature of the MAPE - the MAPE penalises over-estimations more harshly than under-estimations, and also the tendency for the MAPE to "explode" if the observed values are small. The paper introduces what is now referred to as the *symmetric* mean absolute percentage error (sMAPE) and advocated for its widespread adoption. However, in 2006, Hyndman and Koehler showed that many widely used error measurements are degenerate (Rob J. Hyndman and Koehler 2006), including the sMAPE - which can (unsatisfyingly) take on negative values even though supposedly being an "absolute" error. Therefore, the authors introduced the mean absolute scaled error (MASE), which utilizes the in-sample Mean Absolute Error (MAE) of the naïve forecast method. The MASE is easily interpreted as the relative improvement of the proposed forecasting method over the naïve forecasting method. The M4 Competition combined both the sMAPE and the MASE into one overall error measure, which the authors call the overall weighted average (OWA) (Makridakis, Spiliotis, and Assimakopoulos 2020). The OWA is calculated by first dividing the sMAPE and MASE of the forecast by the corresponding sMAPE and MASE values of the Naïve2 method, to obtain a relative value for both error measures. Their mean is then taken, giving the OWA. We too shall use the OWA in our experiments, to allow direct comparison with the results of the M4 Competition. The sMAPE and MASE are given by:

$$MASE = \frac{\frac{1}{h}\sum_{n+h}^{t=n+1}|x_t - \hat{x_t}|}{\frac{1}{n-s}\sum_{t=s+1}^{n}|x_t - x_{t-s}|}, \qquad (7.1)$$

$$sMAPE = \left(\frac{2}{h}\sum_{t=n+1}^{n+h}\frac{|x_t - \hat{x_t}|}{|x_t| + |\hat{x_t}|}\right) \times 100\%$$

where h is the forecasting horizon, s is the seasonality of the data and n is the number of in-sample data points. If M_f is the MASE of the forecast method under consideration, M_n the MASE of the Naïve2 method, sM_f the sMAPE of the forecast method under consideration, and sM_n the sMAPE of the Naïve2 method, then the OWA is given by:

$$OWA = \frac{1}{2}\left(\frac{M_f}{M_n} + \frac{sM_f}{sM_n}\right) \qquad (7.2)$$

7.2 Results

In Section 3 the testing methodology is discussed extensively, so we shall only quickly summarise it here. For each season, 7 forecasts are generated for each of the four years of data, corresponding to 48 hour forecasts for each of the final 7 pairs of days. This gives 28 forecasts for each model for each season, across which the performance of each model is averaged. To calculate the final OWA score of each model, the average across all 4 seasons is then calculated. The standard deviation of the OWA of the forecasts is also calculated. We believe this is useful to analyse, as it represents the stability of the forecasts generated by the model throughout the four years. For two methods with a similar average OWA, a lower standard deviation implies that, regardless of the day of the week and year, there is a greater chance of producing an accurate forecast.

Analysis of 48 Hour Forecasts

The table on the left in Fig. 14 shows the final results for the 48 hour forecasts. Averaged across the four seasons and four years, the naïve seasonal (NaiveS) method performs the best, with an average OWA of 0.79 - representing an improvement of 21% on average over the Naive2 method. Whilst the superiority of such a simple method may be surprising, our view is that over such a long forecasting horizon this is almost to be expected, given the strong seasonality of the data, yet still unpredictable nature of short-term energy demand. Following closely behind the NaiveS method are the 6 variants of the hybrid ES-RNN model. The superiority of the two ES-RNN-D variants (the variants trained only using the *total load actual* time series) over the other variants is perhaps evidence that, in general, there is little predictive power in the other time series in the dataset - such as the generation and pricing information. However, in the case of the autumn season, the two ES-RNN-S variants outperform all of the other variants by 3.6% - 12.5%, which would suggest that in the autumnal months the hybrid models benefit from cross-learning with the other information. The effectiveness of the inclusion of the weather information is also debatable. Although the ES-RNN-DW performs the best out of the 6 variants on average, it only outperforms its weather-less counterpart (ES-RNN-D) by 0.2%, and does so only on the summer and autumn months. Furthermore, the ES-RNN-IW and and ES-RRN-SW are outperformed by their weather-less counterparts by ~5% and ~9% on average, respectively. Whilst we showed that there is a relatively strong linear relationship between CD/LD and temperature in Section 3, without predictions for future temperatures such historical information likely holds little predictive power in the short-term. It is interesting to note that the ES-RNN models in this paper achieved a very similar OWA to the OWA achieved by Smyl's original ES-RNN model in the M4 Competition (0.821), which we believe validates our implementation of the model. Furthermore, the standard deviation of the OWA of the two ES-RNN-D models is significantly lower than that of the NaiveS model. As the three models have very similar forecasting performance, it could be argued that use of the ES-RNN-D models is preferable.

The results for the summer season are of particular interest. Only three of the models (the two ES-RNN-D variants and the ES-RNN-I model) outperform the Naive2 method, and the NaiveS method performs very poorly relative to the other 3 seasons. We believe this is due to the weekly seasonality breaking down more frequently in the summer

48 Hour	Spring		Summer		Autumn		Winter		Average		
Method	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	*Rank*
NaiveS	0.469	0.397	1.364	0.672	0.642	0.384	0.683	0.383	0.790	0.459	1
ES-RNN-DW	0.679	0.228	0.938	0.404	0.796	0.262	0.793	0.339	0.802	0.308	2
ES-RNN-D	0.663	0.218	0.960	0.396	0.831	0.292	0.787	0.340	0.810	0.312	3
ES-RNN-I	0.792	0.354	0.894	0.370	0.772	0.234	0.870	0.413	0.832	0.343	4
ES-RNN-S	0.710	0.368	1.234	0.740	0.736	0.244	0.788	0.268	0.867	0.405	5
ES-RNN-SW	0.745	0.345	1.223	0.709	0.729	0.239	0.968	0.402	0.916	0.424	6
ES-RNN-IW	0.842	0.394	1.119	0.600	0.861	0.333	0.862	0.367	0.921	0.424	7
Comb	0.996	0.033	1.006	0.049	0.965	0.043	1.002	0.032	0.992	0.039	8
Holt	1.000	0.001	1.005	0.016	1.006	0.011	0.984	0.064	0.999	0.023	9
SES	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	10
Naive2	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	10
Theta	1.002	0.006	1.000	0.003	1.000	0.001	0.997	0.008	1.000	0.005	10
Damped	0.994	0.095	1.061	0.184	0.922	0.111	1.032	0.071	1.002	0.115	11
Holt-Winters	1.128	0.252	1.207	0.351	1.106	0.201	0.852	0.155	1.073	0.240	12
ARIMA	1.073	0.667	1.178	0.592	1.254	0.552	1.180	0.444	1.171	0.564	13
SARIMA	1.415	0.662	3.628	2.327	1.878	0.919	1.862	1.041	2.196	1.237	14
Naive1	3.241	1.699	5.036	2.965	3.012	1.179	3.957	1.625	3.811	1.867	15

2 Hour	Spring		Summer		Autumn		Winter		Average		
Method	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	*Rank*
ARIMA	0.653	0.844	1.561	1.308	0.550	0.831	0.915	0.952	0.714	0.919	1
ES-RNN-I	0.632	0.772	2.327	1.797	0.533	0.801	0.809	0.948	0.723	0.945	2
ES-RNN-IW	0.626	0.770	2.665	1.900	0.574	1.044	0.817	0.877	0.763	1.012	3
ES-RNN-S	0.702	0.960	2.762	2.388	0.557	0.948	0.789	0.917	0.771	1.084	4
ES-RNN-SW	0.727	0.910	2.966	2.354	0.550	0.893	0.878	0.876	0.805	1.037	5
ES-RNN-D	0.787	1.027	3.112	2.100	0.619	1.033	0.864	0.943	0.857	1.111	6
ES-RNN-DW	0.815	1.093	3.050	1.977	0.605	0.996	0.891	0.956	0.858	1.103	7
Damped	0.873	1.063	1.733	1.481	0.773	1.027	0.978	0.971	0.897	1.062	8
Comb	0.948	1.014	1.158	0.940	0.922	0.992	0.990	0.986	0.957	0.992	9
Holt-Winters	1.146	1.204	1.006	1.042	0.971	0.980	0.904	0.891	0.995	1.014	10
SES	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	11
Holt-Winters	1.000	1.000	0.999	1.000	1.002	1.000	0.996	0.994	1.000	1.000	12
Naive2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	13
Theta	1.001	1.000	1.001	1.000	1.000	1.000	1.000	1.000	1.000	1.000	14
SARIMA	0.595	0.755	3.655	3.241	0.866	1.407	1.021	0.990	1.007	1.300	15
NaiveS	0.855	1.261	5.779	4.144	0.649	1.445	0.733	0.853	1.009	1.477	16
Naive1	4.607	0.887	18.997	1.847	3.544	0.920	6.576	1.100	5.366	1.065	17

Fig. 14. Overall results for 48 hour and 2 hour forecasts

months than in other seasons. Of the approximately 48 summer weeks across the four years, only around half show the distinct weekly pattern displayed in Fig. 1, and many include individual days of abnormal demand. The statistical models that employ classical seasonal decomposition (such as the Holt and SES methods) outperform many of the more complicated models in the summer months. We believe this can be attributed to the fact that classical seasonal decomposition averages the seasonal effects over the entire training dataset, and the models which use it are therefore more resistant to significant changes in demand from one week to the next. Across the four seasons, the SES model performs exactly the same as the Naive2 method. This implies that the model learnt a smoothing coefficient value of $\alpha = 1$ in all of the tests, and thus the forecasts it generated were simply the most recent observed value - identical to the Naive2 model. We believe that this is predictable behaviour, as the SES model does not natively support seasonal data. Finally, we note the surprisingly poor performance of the SARIMA and (to an extent) Holt-Winters' models, which we discuss in detail in the following subsection.

Analysis of Forecasts at Other Lead Times

Whilst the NaiveS prevails for 48 hour forecasts, the results for shorter lead-times are more varied. We believe it to be more instructive to plot the absolute error (sMAPE) against the lead time, as opposed to the OWA which is a measure of relative performance. The choice of the sMAPE over the MASE was arbitrary, as the results differ insignificantly when either are used. In Fig. 15 the average sMAPE across the four seasons is plotted against the lead time for each of the forecasting models. The hybrid models are plotted with circles and the statistical models are plotted with crosses. Note that the Naive1 method is excluded completely, as well as the SARIMA method for longer lead times, due to their poor performance. Note also that the Naive2, Holt, SES and Theta models generated almost identical forecasts across all the tests, and thus the cyan line representing the Theta method should be taken to represent the performance of all four of these models. Excluding the NaiveS model, a common feature to all of the methods is the sharp increase in the sMAPE as the lead time increases, up until a lead time of 10 to 12 hours, where the errors for each method typically plateau. The errors of all the methods then typically decrease by

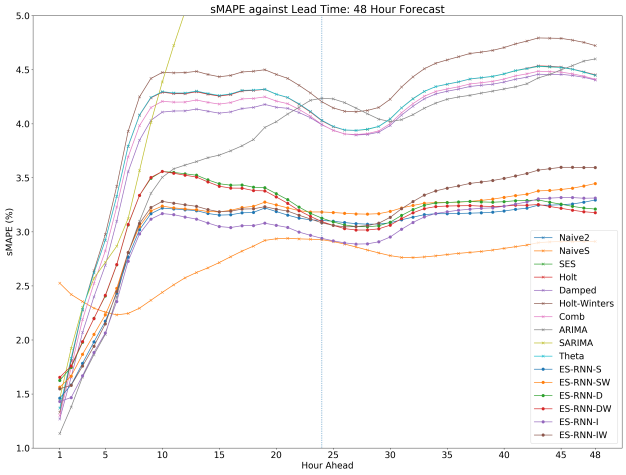

Fig. 15. sMAPE against lead time for the 48 hour forecast

~0.25% at a lead time of around 24 hours (emphasised by the dotted blue vertical line), which is likely to be a consequence of the strong daily seasonality of the data. The hybrid methods unanimously plateau to a lower region than that of the statistical methods (again excluding the NaiveS model), with an sMAPE of ~3.0% - ~3.5% compared to ~4.0% - ~4.5% for the statistical methods. We believe that this improved performance is a consequence of the hybrid models' use of recent seasonal values in their forecast, which will typically be more accurate than the seasonal values generated by classical decomposition that are used by the statistical methods. The NaiveS model is the only model with a unique sMAPE profile. There is little variation in the model's performance over the 48 hour forecast: starting at 2.525% for the hour-ahead forecast, dropping to a minimum of 2.245% after 6 hours and then plateauing just below 2.912% - the optimum out of all of the methods. As noted in the previous subsection, we believe that this consistent performance of the NaiveS model is due to the extremely strong seasonality of the data. The peak performance for a lead time of 6 hours possibly implies the existence of 6-hour seasonality that we had not previously accounted for.

However, as the NaiveS model is outperformed by most of the models for lead times of up to 6 hours, we further analyse these (very) short-term forecasts. In Fig. 16, the

average sMAPE is plotted against the lead times for the first 12 hours. For the single hour-ahead forecasts, the sta-

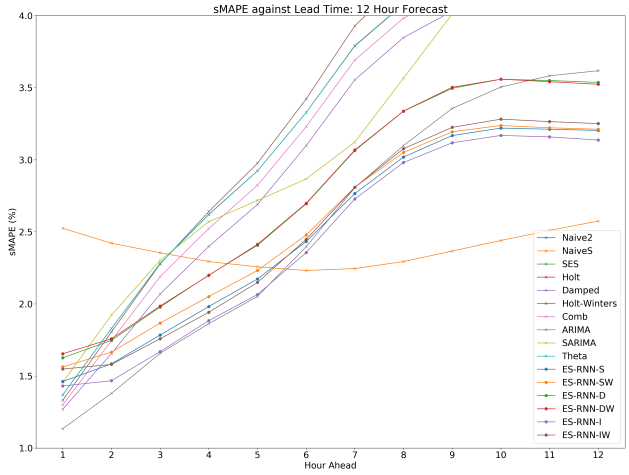

Fig. 16. sMAPE against lead time for the first 12 hour forecast

tistical benchmark methods - including the Naive2 method - typically outperform the hybrid models. However, except for the ARIMA model, the performance of the statistical models drops dramatically after the first hour. This is further evidenced in the table on the right in Fig. 14, which gives the OWA results for the 2 hour forecasts. As is shown, other than the ARIMA model, the hybrid models perform optimally. The strong performance of the ARIMA model is particularly impressive as the model does not natively handle seasonal data, and the model achieves sMAPE of just 1.134% for the hour ahead forecast. The ARIMA model continues to perform superiorly up until a lead time of 6 hours, and for longer lead times the NaiveS method then dominates. We systematically identified the optimal ARIMA model (in terms of the AIC) for each season within each year, and the improvement in performance of the ARIMA model over the other models can perhaps be attributed to this. In particular, the ES family of models require no formal identification prior to being fitted to the training data, and thus it is feasible that the well-specified ARIMA models are better tuned to the dataset. However, this reasoning leads us to find the relative weakness of the SARIMA model surprising, for forecasts of lead times greater than one hour. We believe that this weakness in performance, particularly for lead times greater than 10 hours, may be attributed to the high frequency and 168-hour seasonality of the data. An ARIMA$(p, 0, q)(P, 0, Q)_{168}$ model, such as the one used in this paper, has $p + q + 168(P + Q)$ free parameters; the values of which must be deduced by a non-linear optimisation algorithm. Therefore, it is very feasible that the values found for these parameters were sub-obtimal, leading to the poor results. The Holt-Winters method also performs relatively poorly, beating the Naive2 model in only the first two lead times. We hypothesise that this may be due to the absence of an auto-regressive error correction term within the model's formulation, which when added appears to greatly improve the model's performance (J. Taylor and Mcsharry 2007).

We believe that the strength of the hybrid models lies in their consistency. Neither the ARIMA model nor the NaiveS model are able to generate consistently good forecasts for all lead times. The performance of the ARIMA model deteriorates for longer lead times, and the NaiveS model performs poorly for shorter lead times. In particular, our novel ES-RNN-I model performs well across all lead times, as shown in Fig. 15. For lead times of 3 to 5 hours its performance is almost identical to that of the ARIMA model, and for lead times of 20 to 30 hours its performance almost matches that of the NaiveS model. The model achieves an sMAPE of just 1.431% for one hour ahead forecasts, and a relatively strong sMAPE of 3.313% for 48 hour forecasts.

7.3 Other Experimental Findings

Hybrid Models
We found that the 6 hybrid models are all particularly sensitive to the initialisation of the initial seasonality parameters of the ES component, and the learning rate used whilst training. When the initial seasonality parameters are randomly initialised, the models are rarely able to converge to a performant local minimum. Therefore, we initialise these values with the indices calculated by classical seasonal decomposition of the training data. Our findings reiterate those of Smyl - that the key to successful forecasts is fitting a smooth level to the data so that the seasonal effects are absorbed in the seasonality values. Initialising the indices using classical seasonal decomposition enables this to happen. The initial learning rate used by Smyl was found to cause non-convergent behaviour, and so we modified the schedule accordingly. We experimented with removing the dLSTM component entirely, to assess if it actually provides any additional predictive ability, and found that the model performs worse without the dLSTM component. Finally, we experimented with the architecture of the dLSTM block. Similar to the findings of (Greff et al. 2015), we found that the use of other architectures (such as the *Gated Recurrent Unit* (Cho et al. 2014)) does not improve the accuracy of the forecasts over the standard LSTM block.

Statistical Benchmark Models
We found that the statistical models are particularly sensitive to the seasonality (either 168 hour or 24 hour) and the optimisation algorithm used. When fitting the SARIMA model, we found that using quasi-Newtonian methods such as the L-BFGS algorithm led to high memory usage, long training times and convergence to poor parameter values, particularly when using 168 hour seasonality. This is likely to be due to the large number of degrees of freedom in the model, which makes approximating the Hessian matrix computationally expensive. During the final testing we opted to use the Powell method instead, as this method does not require any derivatives to be taken. When fitting the Holt-Winters model during testing we also used a non-Newtonian global optimisation algorithm - the so-called *Basin-hopping algorithm* (Wales and Doye 1997) - as opposed to the L-BFGS algorithm. Doing so improved the method's OWA significantly, and using 168 hour seasonality instead of 24 hour seasonality improved the method's OWA further. We experimented with the inclusion of a damped/non-damped additive/multiplicative trend, but

found that when a trend is included the fitting algorithm is often unable to converge. This reaffirms that the data is indeed stationary, as proved statistically in Section 3. In the interest of fairness we also experimented with initialising the seasonal indices of the Holt-Winters model with the indices found using classical seasonal decomposition (mirroring the ES-RNN models), but found no noticeable improvement when doing so. Finally, we experimented with the seasonality used in the NaiveS model. An OWA of 0.79 is achieved when using 168 hour seasonality, which is a significant improvement on the OWA of 1.83 achieved when 24 hour seasonality is used.

8. CONCLUSION AND FURTHER WORK

In this paper, we have used four years of hourly energy demand data to compare empirically the short-term forecasting accuracy of six hybrid and eleven statistical forecasting models. The six hybrid models are all variants of the ES-RNN model - the winner of the M4 Competition - developed by Slawek Smyl of Uber Technologies (S. Smyl 2020). The success of the ARIMA model for lead times of up to 6 hours is impressive, particularly for a model without native support for seasonal data. Of the 17 models, the ARIMA model achieves an optimal sMAPE of 1.134%. For forecasts further into the future, the performance of the ARIMA model deteriorates. The NaiveS model was found to perform superiorly for lead times of 6 to 48 hours, achieving an OWA of 0.79 for the 48 hour forecast - representing a 21% improvement over the Naive2 model. Whilst the ARIMA and NaiveS models are able to generate accurate short-term and long-term forecasts respectively, neither are able to do both. For consistent performance in both short-term and longer-term forecasts, the hybrid ES-RNN models appear superior. The performance of the novel ES-RNN-I model is of particular note - achieving an sMAPE of 1.431% for one hour forecasts and an OWA of 0.832 for 48 hour forecasts - representing a 16.8% improvement over the Naive2 forecast. For longer-term forecasts, all of the hybrid models converge to sMAPEs lower than that of the statistical benchmark models, excluding the NaiveS model. Therefore, we conclude that Smyl's hybrid ES-RNN model, and our modifications to it, are indeed suitable for the specific task of short-term energy demand forecasting.

One limitation of this study is that we only evaluate the performance of the models on a single dataset. For a more complete reflection of the performance of the models, we would like to repeat the same experiment methodology on a different dataset, perhaps focusing on household energy demand rather than system-level demand. In future work, we would also like to further refine the hybrid model in several ways. These include modifying the ES component to handle seasonality at multiple frequencies, and encoding predictable annual events as features in the dLSTM, to assist the model in accounting for variations in demand that occur as a consequence of such events. We believe that the inclusion of forecasted weather information likely holds the key to generating more accurate long-term forecasts. We also believe that using a more sophisticated technique to extrapolate the *level* of the ES component could improve the accuracy of the model's short-term forecasts. Finally, we believe that pre-training the dLSTM component across a larger dataset, such as the one used in the M4 Competition, may further improve the performance of the model, and offers another experimental avenue to explore.

REFERENCES

Abedinia, Oveis, Nima Amjady, and Noradin Ghadimi. "Solar energy forecasting based on hybrid neural network and improved metaheuristic algorithm". In: *Computational Intelligence* 34.1 (2018), pp. 241–260. DOI: 10.1111/coin.12145.

Adika, C. O. and L. Wang. "Short term energy consumption prediction using bio-inspired fuzzy systems". In: *2012 North American Power Symposium (NAPS)*. Sept. 2012, pp. 1–6. DOI: 10.1109/NAPS.2012.6336358.

Agency, European Environment. *Heating and Cooling Degree Days*. June 2019. URL: https://www.eea.europa.eu/data-and-maps/indicators/heating-degree-days-2/assessment (visited on 04/21/2020).

Agency, International Energy. *World Energy Outlook 2017*. 2017, p. 763. DOI: 10.1787/weo-2017-en.

Alshareef, Abdulaziz, E. Mohamed, and E. Al-Judaibi. "One Hour Ahead Load Forecasting Using Artificial Neural Network for the Western Area of Saudi Arabia". In: *Proccedings of the World Academy of Science, Engineering and Technology* 27 (Jan. 2008).

Barak, Sasan and S. Saeedeh Sadegh. "Forecasting energy consumption using ensemble ARIMA-ANFIS hybrid algorithm". In: *International Journal of Electrical Power & Energy Systems* 82 (2016), pp. 92–104. DOI: 10.1016/j.ijepes.2016.03.012.

Bates, J. M. and C. W. J. Granger. "The Combination of Forecasts". In: *OR* 20.4 (1969), pp. 451–468. ISSN: 14732858. URL: http://www.jstor.org/stable/3008764.

Bengio, Y., P. Simard, and P. Frasconi. "Learning Long-Term Dependencies with Gradient Descent is Difficult". In: *Trans. Neur. Netw.* 5.2 (Mar. 1994), pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181.

Box, George.E.P. and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

Brown, Robert G. and Arthur D. Little. "Exponential Smoothing for Predicting Demand". In: (Oct. 1956). URL: https://www.industrydocuments.ucsf.edu/docs/jzlc0130.

Brown, Robert Goodell. *Smoothing, forecasting and prediction of discrete time series*. Englewood Cliffs, N.J.: Prentice-Hall, 1963, p. 468. ISBN: 9780486495927.

Buck Research Instruments, LLC. *Buck Research Manual*. May 2012. URL: http://www.hygrometers.com/wp-content/uploads/CR-1A-users-manual-2009-12.pdf (visited on 04/21/2020).

Cancelo, José, Antoni Espasa, and Rosmarie Grafe. "Forecasting the electricity load from one day to one week ahead for the Spanish system operator". In: *International Journal of Forecasting* 24 (Oct. 2008), pp. 588–602. DOI: 10.1016/j.ijforecast.2008.07.005.

Chan, Felix and Laurent L. Pauwels. "Some theoretical results on forecast combinations". In: *International Journal of Forecasting* 34.1 (2018), pp. 64–74. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2017.08.005.

Chang, Shiyu et al. "Dilated Recurrent Neural Networks". In: *CoRR* abs/1710.02224 (2017). arXiv: 1710.02224.

Ching-Lai Hor, S. J. Watson, and S. Majithia. "Analyzing the impact of weather variables on monthly electricity demand". In: *IEEE Transactions on Power Systems* 20.4 (Nov. 2005), pp. 2078–2085. ISSN: 1558-0679. DOI: 10.1109/TPWRS.2005.857397.

Cho, Kyunghyun et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078.

Chow, Tommy and C.T. Leung. "Neural Network based Short-Term Load Forecasting Using Weather Compensation". In: *Power Systems, IEEE Transactions on* 11 (Dec. 1996), pp. 1736–1742. DOI: 10.1109/59.544636.

Clemen, Robert T. "Combining forecasts: A review and annotated bibliography". In: *International Journal of Forecasting* 5.4 (1989), pp. 559–583. DOI: 10.1016/0169-2070(89)90012-5.

Commandeur, Jacques J.F. and Siem Jan Koopman. *An Introduction to State Space Time Series Analysis*. OUP Catalogue 9780199228874. Oxford University Press, 2007. URL: https://ideas.repec.org/b/oxp/obooks/9780199228874.html.

Crone, S.F., M. Hibon, and K. Nikolopoulos. "Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction". In: *International Journal of Forecasting* 27.3 (2011). cited By 131, pp. 635–660. DOI: 10.1016/j.ijforecast.2011.04.001.

Dickey, David A. and Wayne A. Fuller. "Distribution of the Estimators for Autoregressive Time Series With a Unit Root". In: *Journal of the American Statistical Association* 74.366 (1979), pp. 427–431. URL: http://www.jstor.org/stable/2286348.

Ediger, Volkan and Sertac Akar. "ARIMA forecasting of primary energy demand by fuel in Turkey". In: *Energy Policy* 35 (Feb. 2007), pp. 1701–1708. DOI: 10.1016/j.enpol.2006.05.009.

Ekonomou, L. and Oikonomou D.S. "Application and Comparison of Several Artificial Neural Networks for Forecasting the Hellenic Daily Electricity Demand Load". In: Jan. 2008.

ENTSO-E. *Online TSO Energy Data*. 2019. URL: https://transparency.entsoe.eu/ (visited on 10/11/2019).

Fattaheian-Dehkordi, Sajjad et al. "Hour-ahead demand forecasting in smart grid using support vector regression (SVR)". In: *International Transactions on Electrical Energy Systems* 24 (Dec. 2014). DOI: 10.1002/etep.1791.

Fay, D. and J. V. Ringwood. "On the Influence of Weather Forecast Errors in Short-Term Load Forecasting Models". In: *IEEE Transactions on Power Systems* 25.3 (2010), pp. 1751–1758.

Gao, Rong, Liyuan Zhang, and Xiaohua Liu. "Short-term load forecasting based on least square support vector machine combined with fuzzy control". In: July 2012, pp. 1048–1051. ISBN: 978-1-4673-1397-1. DOI: 10.1109/WCICA.2012.6358034.

Gardner, Everette S. and Ed. McKenzie. "Forecasting Trends in Time Series". In: *Management Science* 31.10 (1985), pp. 1237–1246. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2631713.

Ghalehkhondabi, Iman et al. "An overview of energy demand forecasting methods published in 2005–2015". In: *Energy Systems* 8 (Apr. 2016). DOI: 10.1007/s12667-016-0203-y.

Greff, Klaus et al. "LSTM: A Search Space Odyssey". In: *CoRR* abs/1503.04069 (2015). arXiv: 1503.04069.

Hagan, M. T. and S. M. Behr. "The Time Series Approach to Short Term Load Forecasting". In: *IEEE Transactions on Power Systems* 2.3 (Aug. 1987), pp. 785–791. ISSN: 1558-0679. DOI: 10.1109/TPWRS.1987.4335210.

Hagan, Martin and Ronald Klein. "Identification Techniques of Box and Jenkins Applied to the Problem of Short Term Load Forecasting". In: *IEEE Summer Power Meeting* A77 168-2 (July 1977).

– "Off-Line and Adaptive Box and Jenkins Models for Load Forecasting". In: *Proceedings of the Lawrence Symposium on Systems and Decision Sciences, Berkeley, CA, October* (Oct. 1977).

He, Kaiming et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385.

Hintze, J. https://ncss-wpengine.netdna-ssl.com/wp-content/uploads/2012/09/NCSSUG4.pdf. Kaysville, USA: NCSS, 2007.

Hochreiter, S. et al. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In: *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by S. C. Kremer and J. F. Kolen. IEEE Press, 2001.

Hochreiter, Sepp and JÃŒrgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Holt, Charles C. "Forecasting seasonals and trends by exponentially weighted moving averages". In: *International Journal of Forecasting* 20.1 (2004), pp. 5–10. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2003.09.015.

Hong, Tao. "Short Term Electric Load Forecasting". PhD thesis. North Carolina State University, 2010.

Hong, Tao and Shu Fan. "Probabilistic electric load forecasting: A tutorial review". In: *International Journal of Forecasting* 32.3 (2016), pp. 914–938. DOI: 10.1016/j.ijforecast.2015.

Hyndman, R.J. and B. Billah. *Unmasking the Theta Method*. Monash Econometrics and Business Statistics Working Papers 5/01. Monash University, Department of Econometrics and Business Statistics, June 2001. URL: https://ideas.repec.org/p/msh/ebswps/2001-5.html.

Hyndman, Rob J. and Anne B. Koehler. "Another look at measures of forecast accuracy". In: *International Journal of Forecasting* 22.4 (2006), pp. 679–688. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2006.03.001.

Hyndman, Rob J and Yeasmin Khandakar. "Automatic time series forecasting: the forecast package for R". In: *Journal of Statistical Software* 26.3 (2008), pp. 1–22. URL: http://www.jstatsoft.org/article/view/v027i03.

Hyndman, Robin John and George Athanasopoulos. *Forecasting: Principles and Practice*. English. 2nd. Australia: OTexts, 2018. URL: https://otexts.com/fpp2/ (visited on 12/22/2019).

Jhana, Nicholas. *Hourly energy demand generation and weather*. 2019. URL: https://www.kaggle.com/nicholasjhana/energy-consumption-generation-prices-and-weather (visited on 10/11/2019).

Kandananond, Karin. "Forecasting Electricity Demand in Thailand with an Artificial Neural Network Approach". In: *Energies* 4 (Dec. 2011), pp. 1246–1257. DOI: 10.3390/en4081246.

Khodayar, Mohammad E. and Hongyu Wu. "Demand Forecasting in the Smart Grid Paradigm: Features and Challenges". In: *The Electricity Journal* 28.6 (2015), pp. 51–62. ISSN: 1040-6190. DOI: doi.org/10.1016/j.tej.2015.06.001.

Kong, Weicong et al. "Short-Term Residential Load Forecasting based on LSTM Recurrent Neural Network". In: *IEEE Transactions on Smart Grid* PP (Sept. 2017), pp. 1–1. DOI: 10.1109/TSG.2017.2753802.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://www.cs.toronto.edu/~hinton/absps/imagenet.pdf.

Kucukali, Serhat and Kemal Baris. "Turkey's short-term gross annual electricity demand forecast by fuzzy logic approach". In: *Energy Policy* 38.5 (May 2010), pp. 2438–2445. URL: https://ideas.repec.org/a/eee/enepol/v38y2010i5p2438-2445.html.

Lee, Yi-Shian and Lee-Ing Tong. "Forecasting energy consumption using a grey model improved by incorporating genetic programming". In: *Energy Conversion and Management* 52.1 (2011), pp. 147–152. DOI: 10.1016/j.enconman.2010.06.053.

Lei, Mingli and Zuren Feng. "A proposed grey model for short-term electricity price forecasting in competitive power markets". In: *International Journal of Electrical Power & Energy Systems* 43.1 (2012), pp. 531–538. DOI: 10.1016/j.ijepes.2012.06.001.

Lipton, Zachary Chase. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *CoRR* abs/1506.00019 (2015). arXiv: 1506.00019.

Liu, Dong C. and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* 45.1 (1989), pp. 503–528. DOI: 10.1007/BF01589116.

Liu, Nian et al. "A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids". In: *Applied Energy* 129.C (2014), pp. 336–345. DOI: 10.1016/j.apenergy.2014.0.

Ljung, G. M. and G. E. P. Box. "On a Measure of Lack of Fit in Time Series Models". In: *Biometrika* 65.2 (1978), pp. 297–303. ISSN: 00063444. URL: http://www.jstor.org/stable/2335207.

Makridakis, Spyros. "Accuracy measures: theoretical and practical concerns". In: *International Journal of Forecasting* 9.4 (1993), pp. 527–529. DOI: 10.1016/0169-2070(93)90079-3.

Makridakis, Spyros and Michèle Hibon. "The M3-Competition: results, conclusions and implications". In: *International Journal of*

Forecasting 16.4 (2000). The M3- Competition, pp. 451–476. ISSN: 0169-2070. DOI: 10.1016/S0169-2070(00)00057-1.

Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Statistical and Machine Learning forecasting methods: Concerns and ways forward". In: *PLOS ONE* 13.3 (Mar. 2018), pp. 1–26. DOI: 10.1371/journal.pone.0194889.

– "The M4 Competition: 100,000 time series and 61 forecasting methods". In: *International Journal of Forecasting* 36.1 (2020). M4 Competition, pp. 54–74. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.04.014.

Makridakis, Spyros, S. Wheelwright, and Rob Hyndman. "Forecasting: Methods and Applications". In: vol. 35. Jan. 1984. DOI: 10.2307/2581936.

Masson-Delmotte, V. et al. "Global warming of 1.5 C. An IPCC special report on the impacts of global warming of 1.5 C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty". In: (2018).

Mikolov, Tomas et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *CoRR* abs/1310.4546 (2013). arXiv: 1310.4546.

Moghram, I. and S. Rahman. "Analysis and evaluation of five short-term load forecasting techniques". In: *IEEE Transactions on Power Systems* 4.4 (Oct. 1989), pp. 1484–1491. ISSN: 1558-0679. DOI: 10.1109/59.41700.

Mohamed, Norizan et al. "Short Term Load Forecasting Using Double Seasonal ARIMA Model". In: *Statistics Faculty of Computer and Mathematical Sciences* 15 (July 2010), pp. 57–73.

Nengbao, Liu, Vahan Babushkin, and Afshin Afshari. "Short-Term Forecasting of Temperature Driven Electricity Load Using Time Series and Neural Network Model". In: *Journal of Clean Energy Technologies* 2 (Jan. 2014). DOI: 10.7763/JOCET.2014.V2.149.

Nie, Hongzhan et al. "Hybrid of ARIMA and SVMs for short-term load forecasting". In: *Energy Procedia* 16 (Dec. 2012), pp. 1455–1460. DOI: 10.1016/j.egypro.2012.01.229.

Oğcu, Gamze, Omer F. Demirel, and Selim Zaim. "Forecasting Electricity Consumption with Neural Networks and Support Vector Regression". In: *Procedia - Social and Behavioral Sciences* 58 (2012). 8th International Strategic Management Conference, pp. 1576–1585. DOI: 10.1016/j.sbspro.2012.09.1144.

Okumus, Inci. "Current status of wind energy forecasting and a hybrid method for hourly predictions". In: *Energy conversion and management* v. 123 (2016), pp. 362-371–2016 v.123. DOI: 10.1016/j.enconman.2016.06.053.

Olah, Christopher. *Understanding LSTM Networks*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Papalexopoulos, A. D. and T. C. Hesterberg. "A regression-based approach to short-term system load forecasting". In: *IEEE Transactions on Power Systems* 5.4 (Nov. 1990), pp. 1535–1547. ISSN: 1558-0679. DOI: 10.1109/59.99410.

Park, D. C. et al. "ARIMA forecasting of primary energy demand by fuel in Turkey". In: *IEEE Transactions on Power Systems* 6 (May 1991), pp. 442–449. DOI: 10.1109/59.76685.

Paszke, Adam et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pitt, Barnaby. "Applications of data mining techniques to electric load profiling". In: 2000.

Rebours, Yann and Daniel Kirschen. *What is Spinning Reserve?* 2005.

Rolnick, David et al. "Tackling Climate Change with Machine Learning". In: *CoRR* abs/1906.05433 (2019). arXiv: 1906.05433.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

Russakovsky, Olga et al. *ImageNet Large Scale Visual Recognition Challenge*. 2014. arXiv: 1409.0575 [cs.CV].

Sailor, David J. and J.Ricardo Muñoz. "Sensitivity of electricity and natural gas consumption to climate in the U.S.A - Methodology and results for eight states". In: *Energy* 22.10 (1997), pp. 987–998. ISSN: 0360-5442. DOI: 10.1016/S0360-5442(97)00034-0.

Seabold, Skipper and Josef Perktold. "statsmodels: Econometric and statistical modeling with python". In: *9th Python in Science Conference*. 2010.

Setiawan, A., I. Koprinska, and V. G. Agelidis. "Very short-term electricity load demand forecasting using support vector regression". In: *2009 International Joint Conference on Neural Networks*. June 2009, pp. 2888–2894. DOI: 10.1109/IJCNN.2009.5179063.

Smith, Taylor G. et al. *pmdarima: ARIMA estimators for Python*. [Online; accessed 20/03/20]. 2017. URL: http://www.alkaline-ml.com/pmdarima.

Smyl, S. "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting". In: *International Journal of Forecasting* 36.1 (2020). cited By 7, pp. 75–85. DOI: 10.1016/j.ijforecast.2019.03.017.

Smyl, Slawek. "Ensemble of Specialized Neural Networks for Time Series Forecasting". In: June 2017.

Taylor, J W and S Majithia. "Using combined forecasts with changing weights for electricity demand profiling". In: *Journal of the Operational Research Society* 51.1 (2000), pp. 72–82. DOI: 10.1057/palgrave.jors.2600856.

Taylor, James. "Short-term electricity demand forecasting using double seasonal exponential smoothing". In: *Journal of Operational Research Society* 54 (Aug. 2003), pp. 799–805. DOI: 10.1057/palgrave.jors.2601589.

Taylor, James W. "Exponential smoothing with a damped multiplicative trend". In: *International Journal of Forecasting* 19.4 (2003), pp. 715–725. URL: https://ideas.repec.org/a/eee/intfor/v19y2003i4p715-725.html.

– "Triple seasonal methods for short-term electricity demand forecasting." In: *European Journal of Operational Research* 204.1 (2010).

Taylor, James, Lilian De Menezes, and Patrick Mcsharry. "A Comparison of univariate methods for forecasting electricity demand up to a day ahead". In: *International Journal of Forecasting* 22 (Feb. 2006), pp. 1–16. DOI: 10.1016/j.ijforecast.2005.06.006.

Taylor, James and Patrick Mcsharry. "Short-Term Load Forecasting Methods: An Evaluation Based on European Data". In: *Power Systems, IEEE Transactions on* 22 (Dec. 2007), pp. 2213–2219. DOI: 10.1109/TPWRS.2007.907583.

Van Rossum, Guido and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

Wales, David J. and Jonathan P. K. Doye. "Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms". In: *The Journal of Physical Chemistry A* 101.28 (July 1997), pp. 5111–5116. ISSN: 1520-5215. DOI: 10.1021/jp970984n.

Wang, Yequan et al. "Attention-based LSTM for Aspect-level Sentiment Classification". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 606–615. DOI: 10.18653/v1/D16-1058.

Winters, Peter R. "Forecasting Sales by Exponentially Weighted Moving Averages". In: *Management Science* 6.3 (1960), pp. 324–342. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2627346.

Yang, Shu-Xia and Ning Li. "Power Demand Forecast Based on Optimized Neural Networks by Improved Genetic Algorithm". In: Sept. 2006, pp. 2877–2881. DOI: 10.1109/ICMLC.2006.259073.