Method Description

General Information

Type of Entry (*Academic, Practitioner, Researcher, Student*)	Practitioner
First Name	Belinda
Last Name	Trotta
Country	Australia
Type of Affiliation (*University, Company-Organization, Individual*)	Individual
Affiliation	

Team Members: none

Information about the method utilized

Name of Method	CNN-TS (Convolutional Neural Networks for Time Series)
Type of Method (*Statistical, Machine Learning, Combination, Other*)	Machine Learning
Short Description (up to 200 words)	The approach is to use *all* the given sequences in each frequency type (yearly, weekly, etc) to train our neural networks. This avoids the usual problem with using machine learning for time series forecasting, i.e., that there isn't enough data in each time series to train a model. Also, using a machine learning approach rather than traditional statistical models means that we don't need to explicitly model the mathematical behaviour of each series (linear, exponential, etc). Instead, the machine learning algorithm can implicitly learn the different behaviours and how best to forecast them, by looking at thousands of different examples.

Extended description: see following pages

Convolutional neural networks for time series forecasting

The approach is to use *all* the given sequences in each frequency type (yearly, weekly, etc) to train our neural networks. This avoids the usual problem with using machine learning for time series forecasting, i.e., that there isn't enough data in each time series to train a model. Also, using a machine learning approach rather than traditional statistical models means that we don't need to explicitly model the mathematical behaviour of each series (linear, exponential, etc). Instead, the machine learning algorithm can implicitly learn the different behaviours and how best to forecast them, by looking at thousands of different examples.

Instructions for running the code

The code is written in Python 3.6.1, and uses the Keras package for building the neural network models, with the Tensorflow backend. We use the pandas and numpy packages for reading, writing, and preparing the data. A full specification of the environment is given in environment.txt.

Copy all the python files into a directory. Make a subdirectory called data, and copy the csv data files into it.

Training

The trained model files are provided in this repository (in the zipped trained_models folder), so retraining the models is not necessary if you just want to generate the forecasts; in this case, skip ahead to the next section. I have provided the model training code to show how the trained model coefficients are generated, and in case anyone wants to reproduce the whole process from scratch.

Run shuffle.py; this randomises the order of the data in each file and saves a copy with "shuffle" appended to the name. This is necessary for doing proper validation, since in the original files the series are ordered by type: finance, macro, etc.

To train all the models, run train_models.py. This script will run the training code twice. In the first run, the first 20% of the rows in each file are excluded from training and used to measure the model's performance. A log file is written containing the calculated metrics on this validation set. The second run trains the models using all the data and saves the models in a subfolder called trained_models. The total runtime of the script is around 5 hours on a fairly powerful Windows laptop (i7 processor, 24Gb RAM).

Prediction

Run predict.py. This will load the previously saved trained models (that you have either downloaded or generated in the previous step) and make predictions for all series. The

result is saved is a csv file called Submission_fc_YYMMDD_HHMM.csv. The lower and upper bounds are saved in Submission_lower_YYMMDD_HHMM.csv and Submission_upper_YYMMDD_HHMM.csv. This takes around 10 minutes.

Model structure

We use separate models (with different architectures) for each frequency (yearly, monthly, etc). For all frequencies except yearly, we expect that there will be cyclic effects (for example, with the hourly series we expect that there may be daily and weekly patterns). To capture these effects we use convolutional neural networks. This means we pass a filter having window size equal to the frequency over the inputs. This filter is essentially a set of linear functions (in our case 3), whose coefficients are found during the training. Since the same filter, with the same coefficients, is applied to each cycle (e.g. each 12-month period for the monthly models), we can capture the seasonal patterns.

We train separate models for each period of the forecast horizon; for example, when modelling the quarterly series, we train 8 versions of each neural network, using the same x inputs, but different y values (leaving gaps of 0 to 7 periods between the end of the x data and the y value to be forecast).

The loss function for the neural networks is mean squared error.

Training data

Within each frequency type, there are many different series lengths. For example, the monthly series range in length from 42 to 2794, with 95% lying between 68 and 450. We expect that in general, the more historical data the model can use, the better the accuracy. The neural networks architecture we use requires fixed-length inputs, so we train multiple models for each frequency type. In the monthly case, we train three models using training periods of 48, 120, and 240, in each case using the most recent data available, i.e. the periods at the end of the series. (Since the forecast horizon in this case is 18, as explained above, we actually train 18 versions of each of these models, so we require an additional 18 periods at the end of the training x data, i.e., these models require 66, 138 and 258 periods of data.) For the longer series, this means that we actually have multiple models to choose from. For example, for a series of length 200 we could use the model with 48 inputs or the model with 120 inputs.

We choose the training lengths to be multiples of all the frequencies which we expect the series to exhibit (e.g. for the hourly series, our training period is a multiple of 24 and 7). This makes the calculation of the convolutional filters straightforward, since the filter length will divide the input length, meaning that we repeat the filter an integer number of times over the input.

Note that although the models with longer inputs have more features per training sample, on the other hand there are fewer training samples, so these models do not necessarily

outperform the ones with shorter inputs. We obtained good results by blending (with equal weights) all the available models for a series.

For the small number of series with less than the minimum amount, we add some synthetic data at the beginning by filling the data backwards from the start of the series. To preserve the frequency patterns, we copy from the first available period that is at the same point in the cycle.

Since the training set consists of unrelated series with different magnitudes, we standardise each series (i.e. each row of the training set) by subtracting the mean and dividing by the standard deviation (where the mean and standard deviation are calculated over the x values for each series). Also, we remove outliers from the y values by restricting each series' y value to be at most +/-5 times the standard deviation of the series' x values.

For the series with fewer training samples (i.e. those with weekly or high frequency), we augment the training data by shifting the training data by 1, 2, ..., n-1 periods where n is the frequency as follows:
Yearly (no augmentation); Quarterly (no augmentation); Monthly (no augmentation); Weekly (52); Daily (7; although there are also yearly patterns, adding them would result in an unfeasibly large training set); Hourly (7 x 24 = 168).

Details of model structure

All neural networks use a learning rate of 0.001, batch size of 1000, and 250 epochs. The loss function is mean squared error and the model is trained using the Adam optmizer implementation in Keras. Early stopping is enabled using a validation set consisting of 10% of the training data. If the validation loss fails to improve for 10 epochs, training is stopped. This decreases the training time and helps control overfitting.

The inner layers of each neural network use the relu (rectified linear) activation.

Yearly

Training periods: [10, 20, 30]
NN structure:

- 2 dense hidden layers of 50 relu units feeding into linear output cell.

Quarterly

Training periods: [20, 48, 100]
NN structure:

- Yearly averages feeding into a 2 successive densely connected hidden layers of 50 relu units then a single linear output cell representing the yearly average forecast.
- Quarterly differences from the yearly average are run through a trainable convolution filter of length 4 with 3 channels.

- Convolution outputs are fed into 2 successive densely connected hidden layers of 50 relu units then a single linear output cell representing the forecast difference from the yearly output.
- Yearly forecast and quarterly difference forecast are summed to produce the final output.

Monthly

Training periods: [48, 120 (= 12 * 10), 240 (= 12 * 20)]
NN structure

- Yearly averages feeding into 2 successive densely connected hidden layers of 20 relu units then a single linear output cell representing the yearly average forecast.
- Monthly differences from the yearly average are run through a trainable convolution filter of length 12 with 3 channels.
- Convolution outputs are fed into 2 successive densely connected hidden layers of 20 relu units then a single linear output cell representing the forecast difference from the yearly output.
- Yearly forecast and monthly difference forecast are summed to produce the final output.

Weekly

Training periods: [52, 520 (= 52 * 10), 1040 (= 52 * 20)]
NN structure for training length 52:

- Weekly inputs fed into 2 successive densely connected hidden relu layers of 20 relu units, then a single linear output cell.

NN structure for training lengths > 52:

- Yearly averages feeding into 2 successive densely connected hidden relu layers of 20 relu units then a single linear output cell representing the yearly average forecast.
- Weekly differences from the yearly average are run through a trainable convolution filter of length 52 with 3 channels. Convolution outputs are fed into 2 successive densely connected hidden relu layers of 20 relu units then a single linear output cell representing the forecast difference from the yearly output.
- Yearly forecast and monthly difference forecast are summed to produce the final output.

Daily

Training periods: [98 (= 14 * 7)]
NN structure:

- Weekly averages feeding into 2 successive densely connected hidden relu layers of 20 relu units then a single linear output cell representing the yearly average forecast.
- Daily differences from the weekly average are run through a trainable convolution filter of length 7 with 3 channels. Convolution outputs are fed into 2 successive densely connected hidden relu layers of 20 relu units then a single linear output cell representing the forecast difference from the weekly output.

- Yearly forecast and weekly difference forecast are summed to produce the final output.

I experimented with adding longer training lengths, and adding a yearly convolution for these, but these models produced poor results during testing, which seemed to be mostly caused by some problematic series having very large step changes.

Hourly

Training periods: [672 (= 96 * 7 * 24)]
NN structure:

- Weekly averages feeding into 2 successive densely connected hidden relu layers of 20 relu units then a single linear output cell representing the weekly average forecast.
- Daily differences from the weekly average are run through a trainable convolution filter of length 7 with 3 channels. Convolution outputs are fed into 2 successive densely connected hidden relu layers of 20 relu units then a single linear output cell representing the forecast difference from the weekly output.
- Hourly differences from the daily average are run through a trainable convolution filter of length 24 with 3 channels. Convolution outputs are fed into 2 successive densely connected hidden relu layers of 20 relu units then a single linear output cell representing the forecast difference from the daily output.
- Weekly forecast and daily difference and hourly forecast are summed to produce the final output.

Prediction intervals

To calculate the prediction interval, we consider all series of a given frequency type (yearly, monthly, etc) together. We calculate the error over the the last h periods of the training data, where h is the forecast horizon, and normalise it by dividing by the prediction over this period. For each period of the forecast horizon, we calculate the 0.025th and 0.975th quantiles, and use these to calculate the forecast intervals.