Deep neural networks for FX prediction

Designing robust models for FX trade sizing and currency positioning

Using historical spot FX rates from 30 currency pairs dating back 16 years, Mesirow Currency's neural network models address the challenges that traditional machine learning techniques can face when trying to model financial time series.

Mehryar Emambakhsh
Vice President,
Senior Research Scientist

Richard Turner
Managing Director,
Head of Research

Brief overview of classic machine learning pipelines

Traditional machine learning pipelines usually consist of four separate steps:

1. PRE-PROCESSING

Data is mapped from a noisy, irregular, and sometimes high dimensional space onto a cleaner, more regular, and lower dimensional space.

2. HANDCRAFT FEATURE EXTRACTION

Distinctive features (represented as vectors or tensors) are manually formulated and engineered from the input data. These features should be consistent, robust, and preserve information while simultaneously removing redundancy.

3. POST-PROCESSING

The features, which together form a feature space, are sometimes post-processed to achieve further dimensionality reduction or expansion.

4. CLASSIFICATION

The result of the first three steps is then given to a classification/regression algorithm. Input is mapped onto a discrete domain (class labels) and hyper-planes split the feature space into different regions, each belonging to different classes.

Some of the advantages of this traditional machine learning pipeline are as follows:

- they are (usually) fast to train and back test
- feature extraction can be robustly designed if the underlying physics from which the samples were generated is known (example: Mel-frequency cepstral coefficients (MFCC) features used in speech recognition[1])
- hand crafted feature extraction and engineering can sometimes help in understanding the failure of the system, thus facilitating *explainability*

This traditional approach does come with limitations, though, as the steps mentioned above are clearly not independent from each other.

A well performing machine learning model is dependent on strong classifiers (step 4), which requires a well-designed feature extraction (step 2). If either of these steps fail, it can deteriorate the overall performance. Therefore, to avoid this issue, steps 1 to 4 of any machine learning pipeline should be somehow simultaneously optimised.

Also, these models usually assume that the input samples are independent and identically distributed (IID)[2]. When attempting to predict financial data trends, however, non-IID input samples prove difficult to structure and model, so the task of simultaneously designing a robust feature extraction algorithm and avoiding class over-representation is extremely difficult.

One solution to make the feature extraction step more robust would be to link outputs from several different algorithms. Unfortunately, this approach also leaves us with a potential problem: when the number of feature vectors is much higher than the number of samples, this concatenation can result in an extremely high dimensional space in which there will not be enough samples to represent each class. This is known as the *curse of dimensionality*[3] and can result in poor predictive performance.

Deep neural networks

Neural networks belong to a category of machine learning algorithms that attempt to find structure and model input data by learning parameters embedded within a set of composite non-linear functions.

Neural networks are not totally new machine learning techniques, and most of their well-known architectures can be traced back to research in the 80s and 90s: recurrent neural networks (initiated) in 1982[4], convolutional neural networks in 1995[5] and long short-term memory networks (LSTM) in 1997[6].

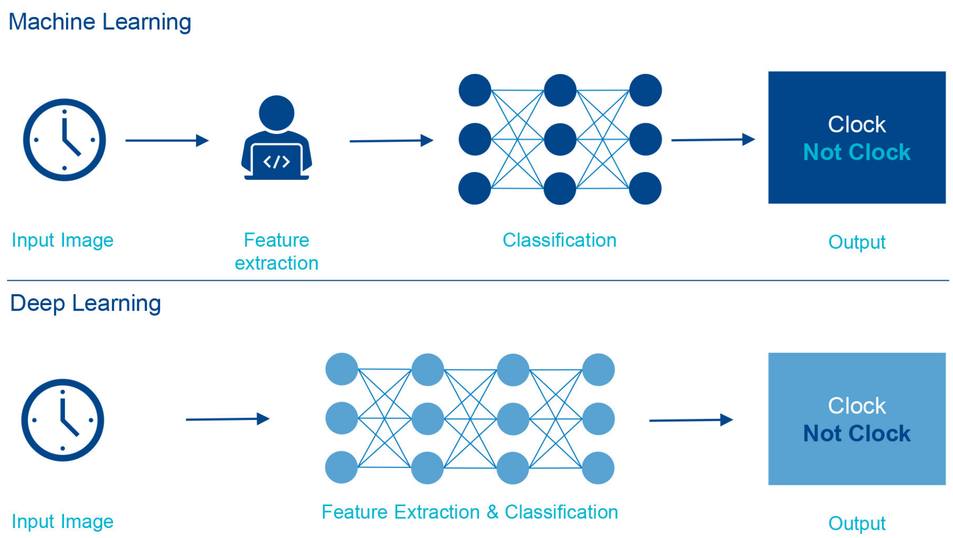
What makes these techniques widely popular in recent machine learning pipeline design are two key factors: availability of a large amount of open source data, and accessibility to fast and cheap parallelised computational power.

Neural networks superiority

Neural networks have many advantages when compared to other machine learning algorithms. One of the most important features of the neural network is their closed-form solution for feature extraction and classification. This resolves the difficult problem of first formulating a feature extraction algorithm and then optimising a compatible classifier for it. As Shown in Figure 1, neural networks do both tasks simultaneously, and these learned features can even be transferred to other (not necessarily similar) tasks, a process known as transfer learning[7].

FIGURE 1: **CLASSIC MACHINE LEARNING PIPELINES (TOP) VS. DEEP LEARNING-BASED APPROACHES**

While classic machine learning relied on an independent handcrafted feature extraction and classification, deep learning paradigms encapsulate feature extraction and classification inside a simultaneous process.

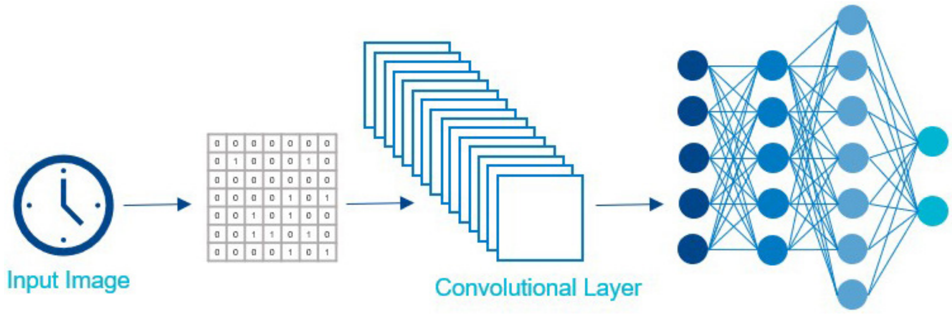


Source: Mesirow Financial

Thanks to their non-linearity, neural networks can learn more complex patterns from the data, and temporal dependencies between samples can also be learned via recurrent or convolutional architectures. As an example, Figure 2 shows a convolutional neural network which is capable of extracting spatial information from input images via learning various filters and, finally, performing classification through dense layers.

FIGURE 2: **A SIMPLE CONVOLUTIONAL NEURAL NETWORK APPLIED FOR AN IMAGE CLASSIFICATION TASK**

The input data is passed through several convolutional layers, which capture spatial information. Their output is then given to dense neural network layers to perform dimensionality reduction and classification.



Source: Mesirow Financial

General challenges in training neural networks

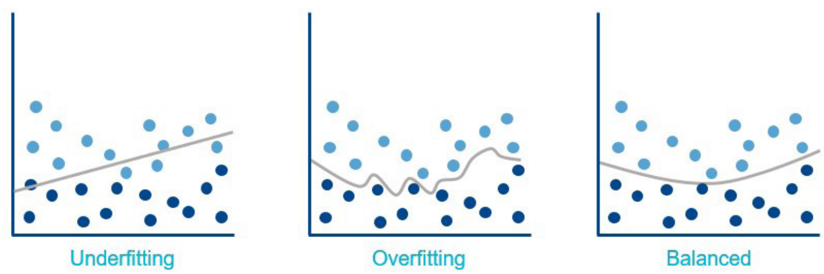
Neural networks require much more computational power to train than traditional machine learning algorithms. A neural network can very easily have thousands of trainable parameters, and to optimise these parameters over a high dimensional space, a substantial amount of data is usually needed.

Neural networks also have several hyper-parameters, which significantly influence their performance. These cover a wide range of parameters, from the number of hidden layers and neurons per layer, to learning rate and optimisation parameters. Neural network architecture search, a process that can automatically search for the best network architecture and hyper-parameter setting, can be a possible solution but is still an ongoing research problem.[8,9,10]

FIGURE 3: **UNDERFITTING VS. OVERFITTING**

Failure to properly model input data (due to low model complexity and sometimes, class over-representation within the training data) increases bias and results in underfitting.

On the other hand, models that are too complex can cause high variance, low generalisation capability, and can cause the model to overfit. Ideally, there should be a balance between bias and variance[3], which is a fundamental challenge in machine learning.



Source: Mesirow Financial

Although underfitting is not often problematic, due to the high complexity of these networks, one problem that could result (and needs to be avoided) during model design is overfitting (Figure 3). This shows itself in a large gap between train, validation, and test performance and in drastic fluctuations in output (high variance) when there is any small alteration to the input data.

Challenges in training neural networks over financial data

Non-stationary and non-IID inputs (e.g. financial time series) can be problematic for training. Non-stationarity causes significant statistical differences between the train, validation, and test sets. For complex models, which are already vulnerable to input variations this can, of course, cause a significant drop in performance during the live prediction.

One solution to this is to limit the number of training samples by using more recent data points (in the hopes of achieving higher stationarity while simultaneously capturing recent market trends), but this undermines the necessity of incorporating a substantial amount of data in order to effectively train the neural networks.

In short, using too many training samples can result in class over-representation, but using too few can cause the neural network model to overfit. Hence, the challenge of using neural networks to their highest potential over a complex task can sometimes force quantitative researchers to discard neural networks completely and utilise old-fashioned solutions instead.

Designing a classification model over financial data relies on first labelling the samples, which could be quite challenging. Such data labelling should perfectly represent immediate market trends for each sample. Only utilising immediate next samples for label assignment can result in high class overlap as in a short forward-looking window, financial series tend to act as random walk. However, a long-term forward-looking window for labelling can cause mid-term losses or a missing out on profit-making opportunities. Ideally, a financial data labelling strategy should consider all of these.

Our neural network-based solution

The neural network models used in Mesirow Currency Management's alpha strategies use historical daily spot FX rates from 30 currency pairs as the only input to the network; implemented as a binary classifier, the models learn to label each sample into going long or short. All the models are back tested over a period of 16 years.

During model development, we considered the limitations of classic machine learning techniques and challenges in utilising neural networks over financial time series and addressed those issues (overfitting, hyper-parameter optimisation, stationarity) in the design of our deep neural network strategy.

Network architectures

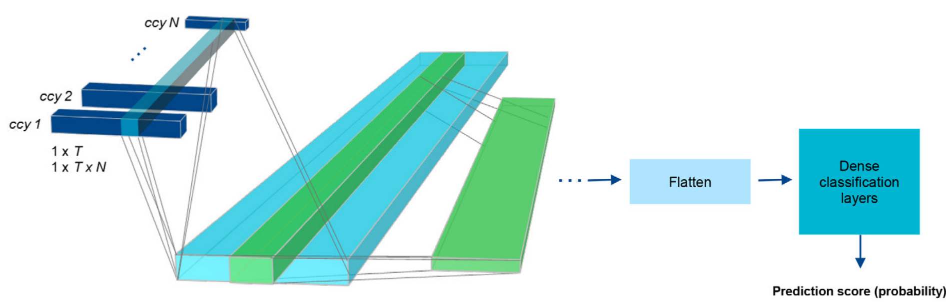
Our networks are based on three main architectures:

1. Convolutional
2. Recurrent
3. Convolutional Recurrent (or Mixture)

Convolutional networks model the temporal dependencies by learning filters which are convolved with the input data. An example of this approach is shown in Figure 4.

FIGURE 4: **A CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE OVER FX DATA**

Given a look-back window size *T*, data from multiple *N* currencies can be arranged as tensors, which are given to several convolutional layers to capture temporal dependencies. After flattening their outputs, the filtered data is fed into several dense layers for dimensionality reduction and final classification. The output probability score is used to determine the label.



Source: Mesirow Financial

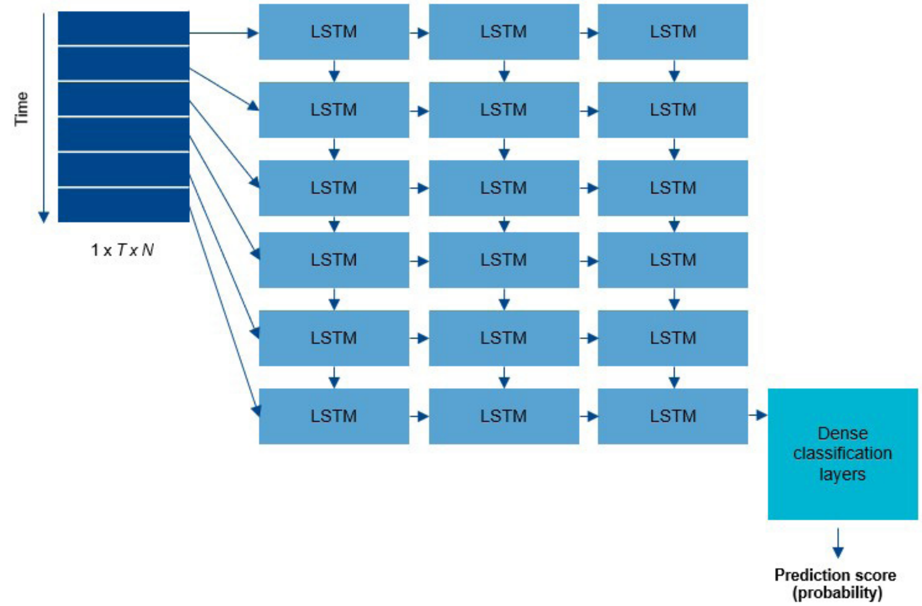
The financial data samples are intrinsically temporally dependent. Such dependency can be modelled through defining a (hidden) state space, which stores information about previous samples (observations). As shown in Figure 5, recurrent neural networks can learn from past data samples and generate this state space during their training process.

The recurrent architecture models the underlying temporal dependencies between the samples by propagating the neural hidden states. The recurrent units are implemented as simple recurrent layers, long short-term memory networks (LSTM)[6] and gated recurrent units (GRUs)[11]. The choice of recurrent layers is determined fully automatically from a separate optimisation procedure.

Finally, the Mixture architecture utilises the convolutional filters to smooth out, denoise, and simplify the input data by making important features more prominent; the outputs of these convolutional layers are then given to a recurrent neural network to capture the temporal trends, as depicted in Figure 6.

FIGURE 5: **A RECURRENT NEURAL NETWORK USING LSTM RECURRENT BLOCKS**

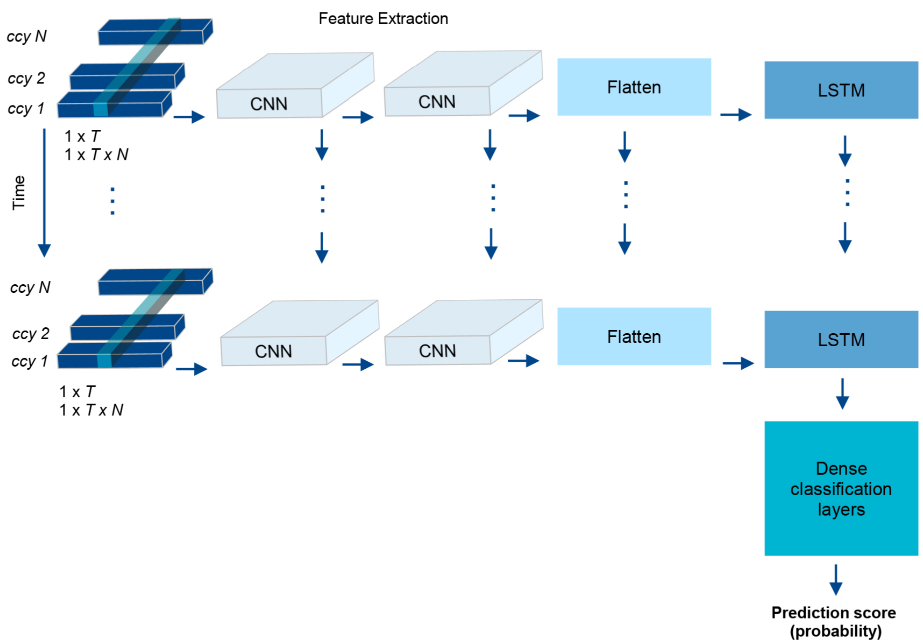
Given a look-back window size, the data is chronologically ordered and is given to a deep recurrent neural network. This model learns the temporal dependencies within the data samples by conveying the network's latent state.



Source: Mesirow Financial

FIGURE 6: **A CONVOLUTIONAL RECURRENT (MIXTURE) ARCHITECTURE.**

During its training phase, the networks learn filters to smooth the data and extract features, while the outputs are given to a recurrent neural network (here shown as LSTM blocks) to learn temporal dependencies.



Source: Mesirow Financial

Stationarity vs. memory preservation

One approach to provide stationary input data is to compute the first order differencing. This, however, can completely remove the temporal dependencies (or memory) between the input samples, which is extremely informative when designing predictive models. Therefore, there should be a trade-off between stationarity and memory preservation.

For our models, stationarity is maintained using a controlled adaptive fractional differencing procedure[12] and memorisation is achieved by incorporating recurrent and convolutional neural architectures.

Underfitting vs. overfitting

While underfitting is easily avoided by discarding models with low train accuracy, extra care has been taken to reduce the risk of overfitting. Each of our models can easily have tens of thousands of parameters, and a robust validation step is embedded within the network training process in order to avoid overfitting. This is achieved by incorporating regularisation methods, adding spatial and temporal dropout layers, early stopping procedures, and automated evaluation of validation loss and accuracy.

Data labelling and training one model per currency pair

Data labelling is performed in such a way to capture immediate data trends. To detect possible cross-currency correlations, the network inputs are tensors containing data from all 30 of the currency pairs traded. Each model performs prediction for one currency pair. Therefore, for each re-tuning interval we will have 30 deep neural networks optimised for the prediction task.

Temporal sample weighting and model selection

The class over-representation vs. lack of data dilemma is addressed via a temporal sample weighting procedure. The sample weighting is also used during the model execution: the models only generate signals if they perform better than a minimal threshold when their inputs are temporally weighted. This step significantly improves the robustness of the system by avoiding trading when the market is very volatile.

Hyper-parameter optimisation and implementation for back testing and live signalling

The model architectures are implemented in a decoupled way in our machine learning pipeline. This facilitates the utilisation of a separate hyper-parameter optimisation and neural architecture search. In other words, the model architecture constantly evolves based on the most recent FX rates.

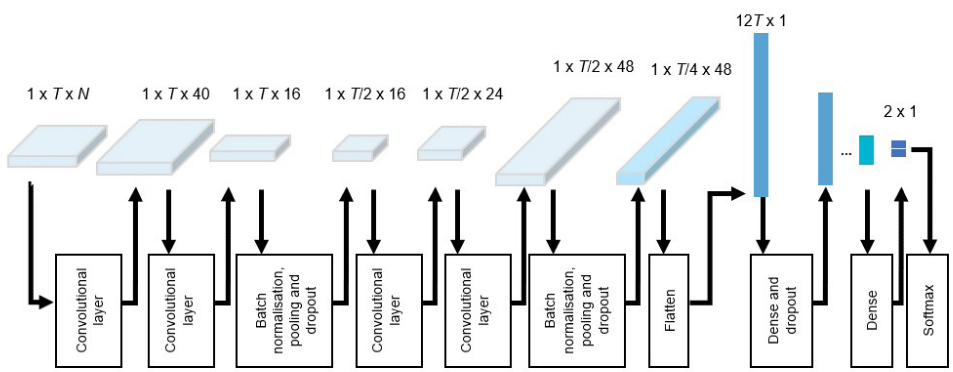
All the computations are performed over our in-house GPU servers. The models are extensively back tested over a period of 16 years. To speed up these highly computationally demanding back testing operations, the process is parallelised. This reduces the tuning time (including architecture search and training) to less than 5 hours over all 30 currencies and enables daily model re-calibration.

The generated signals are post-processed further to reduce the effects of carry and transaction costs, while reducing the absolute drawdown. The signal generated by our neural network model shows very low correlation with our other models, indicating it is a novel source of alpha and so improving the overall return of the system per unit of risk.

We have used TensorFlow v2.0 as our machine learning pipeline framework. An example of the trained convolutional architectures to predict for one of our currency pairs is shown in Figure 7. Depending on the number of filters, the convolutional layers generate tensors with different dimensionalities. While the stability of these layers is controlled using the batch normalisation layers, the average and max pooling layers extract the most prominent features from the input tensors. The dropout layers are used to improve the networks' generalisation and to reduce the risk of overfitting. The final layer applies the SoftMax function, which generates class dependency probabilities.

FIGURE 7: **ONE OF OUR TRAINED DEEP CONVOLUTIONAL NEURAL NETWORK ARCHITECTURES**

T and *N* are the number of look-back days and number of input currencies, respectively. The input data is passed through several convolutional, batch normalisation, pooling, dropout and dense fully connected layers, while the final decision making is performed over the output of the last layer (the SoftMax layer).



Source: Mesirow Financial

Conclusions

In this paper, we briefly discussed the limitations of classic machine learning-based techniques and challenges in utilising neural networks over financial time series. We then explained how we addressed those issues to design our deep neural network strategy for FX trade sizing. Our fast parallelised re-tuning framework enables even daily model re-calibration, preparing the networks to learn from the latest data and market trends.

In addition to FX trade sizing, our model design paradigm can be applied to other applications: asset allocation, market volatility analysis, regression, outlier detection, and portfolio management – some of which we have already started research and algorithm development. Our implemented framework facilitates straightforward integration and back testing for these applications. As our approach is completely data driven, capable of detecting underlying correlations between various data modalities, we believe it can also be applied to other types of financial time series (e.g. stock market data) and allows other data sources, for example textual news data, to be easily incorporated.

While this strategy has been part of our live trading platform since July 2020, we continuously inspect and analyse the performance of our models and actively research more advanced neural architectures and machine learning models. Our decoupled implementation facilitates fast historical back testing, replacing models with improved versions and immediate usage of any new model for live trading.

About Mesirow

Mesirow is an independent, employee-owned financial services firm founded in 1937. Headquartered in Chicago, with locations around the world, we serve clients through a personal, custom approach to reaching financial goals and acting as a force for social good. With capabilities spanning Global Investment Management, Capital Markets & Investment Banking, and Advisory Services, we invest in what matters: our clients, our communities and our culture. To learn more, visit mesirow.com and follow us on LinkedIn.

References

1. Steven Davis and Paul Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences." *IEEE Transactions on Acoustics, Speech, and Signal Processing 28.4 (1980): 357-366.*

2. Marcos Lopez De Prado. *Advances in Financial Machine Learning.* John Wiley & Sons, 2018.

3. Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

4. John J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences* 79.8 (1982): 2554-2558.

5. Yann LeCun and Yoshua Bengio. "Convolutional networks for images, speech, and time series." *The Handbook of Brain Theory and Neural Networks* 3361.10 (1995): 1995.

6. Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." *Neural Computation* 9.8 (1997): 1735-1780.

7. Michael J. Prince and Richard M. Felder. "Inductive teaching and learning methods: Definitions, comparisons, and research bases." *Journal of Engineering Education* 95.2 (2006): 123-138.

8. Liu, Chenxi, et al. "Progressive neural architecture search." *Proceedings of the European Conference on Computer Vision (*ECCV). 2018.

9. Pham, Hieu, et al. "Efficient neural architecture search via parameter sharing." *International Conference on Machine Learning* (2018).

10. B. Zoph and Q. V. Le. "Neural architecture search with reinforcement learning." I*nternational Conference on Learning* Representations (2017).

11. Cho, Kyunghyun, *et al.* "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259 (2014).

12. Clive WJ Granger and Roselyne Joyeux. "An introduction to long-memory time series models and fractional differencing." *Journal of Time Series Analysis* 1.1 (1980): 15-29.