Model calibration with neural networks

The speed with which the calibration of a pricing model can be performed influences the usability of that model. Andres Hernandez shows calibration can be performed significantly faster, regardless of the chosen model, using artificial neural networks; this removes calibration speed as a factor when considering a model's usability

There are several aspects that need to be taken into account when considering the applicability of a pricing model. One that is of central, practical importance is the speed with which it can be calibrated. A model might be deemed preferable to another on financial grounds, but if it takes too long to calibrate (what constitutes 'too long' is dependent on the application), then the model becomes impractical.

Of course, the decision may not always be very clear cut. In some instances, the model could be acceptable from a technical point of view, but only after compromises have been made on the financial side, eg, by limiting the number of instruments used during calibration, or by using approximations, which can lead to inaccuracies for certain parameter regions.

The purpose of this article is to showcase an alternative calibration method that will remove from consideration the time constraint imposed on a model. By using artificial neural networks (ANNs), the vast majority of the calculation workload can be offloaded to a separate training process; this, by its very nature, is allowed to run offline, while the online calibration runs quickly, and is equivalent to multiplying a few small-sized matrices.

As will be seen, the splitting of the calibration problem into a training phase and an evaluation phase is possible, with the fulcrum of the exercise being the generation of a large training set, such that the neural network can generalise from it.

As an added benefit (not addressed in this article), neural networks, being fully differentiable functions, can also provide model parameter sensitivities to market prices, potentially informing us when a model should be recalibrated.

This article is structured as follows. First, we lay out the calibration process, such that the problem to be solved is made explicit. We then detail how neural networks can be used to approach the calibration problem. Next, we provide an example use of the previously discussed methodology, applying it to the simple calibration of a Hull-White model with constant parameters. Finally, we provide some closing comments and perspectives for future work.

Calibration problem

For a given model $\boldsymbol{M}$, an instrument's theoretical quote will be denoted by:

$$Q(\tau) = \boldsymbol{M}(\theta; \tau, \phi) \tag{1}$$

where θ represents the model parameters, τ represents the identifying properties of the particular instrument, eg, maturity, day-count convention, etc, and ϕ represents other exogenous factors used for pricing, eg, an interest rate curve. Equation (1) can represent the expectation under a risk-neutral measure, or some other similar procedure, the nature of which will not be relevant for this discussion.

The model $\boldsymbol{M}$ has n parameters that need to be calibrated, and some or all of them may have constraints:

$$\theta \in S \subseteq \mathbb{R}^n \tag{2}$$

Model calibration is the process by which the model parameters are adjusted so as to 'best' describe a set of N relevant market quotes:

$$\{Q^{\text{mkt}}\} = \{Q_i^{\text{mkt}} \mid i = 1, \ldots, N\}, \qquad \{\tau\} = \{\tau_i \mid i = 1, \ldots, N\} \tag{3}$$

In general, the question of which instruments should be considered is not straightforward, and arguments can be made for the inclusion or exclusion of a particular quote, eg, liquidity concerns, or to avoid overfitting to a particular maturity region. However, it is also possible the set of instruments is reduced for the practical concern of reducing the calibration time. The purpose of this exercise is to discard this concern, and instead allow the set to be chosen on purely financial and mathematical grounds, although weights can be adjusted as deemed necessary.

It was mentioned the model parameters were chosen to best fit the given quotes. The concept of 'best' is measured by a cost function; this will be particular to a problem, and, like the decision of which quotes to include, there are different choices for it. In addition, the decision of which cost function to adopt, as in the quote set, should no longer be advised by the practical concern of reducing calibration time.

The calibration problem consists, then, in finding the parameters θ that minimise the cost function:

$$\theta = \underset{\theta^* \in S \subseteq \mathbb{R}^n}{\arg\min}\ \text{Cost}(\theta^*, \{Q^{\text{mkt}}\}; \{\tau\}, \phi) = \boldsymbol{\Theta}(\{Q^{\text{mkt}}\}; \{\tau\}, \phi) \tag{4}$$

The calibration problem is now seen as a function with N arguments and n outputs:

$$\boldsymbol{\Theta} : \mathbb{R}^N \mapsto S \in \mathbb{R}^n \tag{5}$$

It is this function, $\boldsymbol{\Theta}$, that will be approximated by a neural network.

The different models and cost functions used in quantitative finance present a great variety of conditions, from convex cost functions to more complex structures with multiple local optima present, and for which a global optimiser may be required in order to reach an acceptable solution. As with almost every other aspect of model calibration, the choice of optimiser and the acceptability of an optimisation procedure may be informed by financial considerations (eg, the error being below a financially acceptable threshold), but it may also be influenced by computation time. As with selecting the model, the set of calibration instruments and the form of the cost function, the consideration of computation time should be discarded in the current exercise.

For particular cases, the optimisation problem might be ill defined or result in parameters that vary wildly. This can be grounds for concern, but it will not be addressed in this article. That is because it is not particular to the approach outlined here, nor is it relevant to the purpose of this article, which, once again, is to promote the move away from obtaining calibrated parameters via an optimisation – which can be very time consuming – and replace it with a neural network, which, regardless of the model being represented, should be a quick operation.

The function $\boldsymbol{\Theta}$ need not be smooth everywhere, and it may in fact contain a finite number of discontinuities; but, in general, it is expected to be continuous and smooth almost everywhere. As will be mentioned later, the mathematical basis for the representation of a function by a neural network is the continuity of such a function (Hornik 1991).

Calibration with neural networks

ANNs are a family of machine-learning techniques that have now become ubiquitous in many state-of-the-art solutions for image and speech recognition. In finance, they are widely used for time series forecasting (Connor *et al* 1994; Frank *et al* 2001) and sentiment analysis (Bollen *et al* 2010). While the uses to which they are put are now very diverse, they are, in the end, simply approximating a function with many inputs and some outputs.

ANNs have been used for model calibration as well (see, for example, Klos & Waszczyszyn 2011; Mareš *et al* 2015; Novák & Lehký 2006; Zaw *et al* 2009; Zhang *et al* 2010). However, as far as the author is aware, ANNs have not been used to calibrate financial models. In this context, we face the difficulty of not modelling a physical system, which we could directly sample as often as required in order to build a big enough sample set. In addition, we do not have a mathematical description of the system, which empirical evidence could show to be accurate to the desired degree, and which we could confidently use to generate new samples. It is the acquisition of that bigger training set via statistical sampling that is the main contribution of this article.

There are, of course, other approaches available to approximate a function, even within machine learning. However, neural networks have been very successful in approximating the functions of many inputs. In terms of (5), when N is large, neural networks excel; this is where other methods can falter, eg, interpolation tables.

We will be very pragmatic and not attempt to find the set of financial models whose $\boldsymbol{\Theta}$ function can be approximated via a neural network. Instead, with the confidence bestowed by the universal approximation theorem (Cybenko 1989; Hornik 1991), we will assume it can, in fact, be done, and then attempt to do it.

It is expected different models will be best proxied by different neural networks. In particular, the number of layers and neurons per layer will change from application to application. However, regardless of the topology being used, once a suitable neural network has been found, the calibration problem will be split into two phases: a supervised training phase, which will be performed offline, and an evaluation phase, which will give the model parameters for a given input, and which will be extremely fast.

What is gained by calibrating a model with neural networks is the offloading of the computationally intensive part into an offline process, which in financial cases can take hours or days, depending on the available resources and the model itself. The calculations performed live, ie, when the calibrated parameters are required, are akin to multiplying a few matrices, which any computer will be able to handle quickly.

For the supervised training phase, the task is again split into two parts: (1) the collection of a large enough training set and (2) the actual training, validation and testing. The preparation of the training set will be the crucial part of this exercise. The training set needs to be large enough to allow learning of the model's peculiarities, ie, so as not to overfit. The size and quality of the example set will directly affect the neural network's ability to generalise the obviously limited set of examples into new situations.

Taking all the available historical values and calibrating them would give a set of examples for which the output is known for a given input, and which can then be used for training. The disadvantage of this approach is twofold: first, the training set is unlikely to be large enough, and second, the possibility of backtesting to historical data will be compromised.

Since a good approximation for the inverse of $\boldsymbol{\Theta}$ is known (this is simply the regular valuation of the instruments under a given set of parameters), it is the model itself that will serve as the basis of the training set:

$$\boldsymbol{\Theta}^{-1}(\theta; \{\tau\}, \phi) \approx \boldsymbol{M}(\theta; \{\tau\}, \phi) = \{Q\} \tag{6}$$

This means we can generate new examples simply by generating random parameters θ, albeit with some complications that need to be considered. The first complication is that, if the model requires exogenous factors ϕ, then these will also need to be generated randomly, and likely correlated properly with the parameters for the example set in order to be meaningful.

A second complication may arise from the inability of a model to perfectly fit all relevant market quotes. In the example presented below, it is observed that generating random noise, properly correlated with the exogenous ϕ and parameters θ, improves performance during in-sample backtesting.

Next, the historical data needs to be collected. A portion of it will be used purely to serve as an out-of-sample backtest, while another part will be needed to estimate the correlation between the parameters and, if required, the exogenous factors and errors.

The process described above already assumes a neural network topology has been proposed. It is difficult *a priori* to determine what makes a good topology. So, we first collect a large training set, and then we try out different configurations.

Even if we limit the discussion to feed-forward networks (FNNs) and convolutional neural networks (CNNs), the number of options is large: number of layers, neurons per layer, regularisation procedure, optimisation method, activation function, number of training iterations, etc. Setting these so-called hyper-parameters judiciously is important because they can have a significant impact on the learning process.

The process of hyper-parameter optimisation is a recurring problem in neural networks, and there are standard approaches to it that can give good results. The simplest examples are grid search and manual search. In the former, a multidimensional grid of parameters is prepared and the configuration with the best test results is picked.

In the current exercise, a mixture of grid and manual search is used. As each set of hyper-parameters requires a new neural network to be trained, a grid search can spend a significant amount of time pursuing avenues unlikely to bear fruit. After a limited grid search, a manual inspection selects the areas on which a further grid search will concentrate. This is repeated in an iterative process.

There exist, of course, more developed approaches, eg, random search (Bergstra & Bengio 2012), and more formal search strategies (Snoek *et al* 2012). In the following example, the combination of grid and manual search mentioned above produces good results; however, in future applications these more sophisticated approaches could pay off.

Example: Hull-White

As an example, the single-factor Hull-White model calibrated to 156 sterling at-the-money (ATM) swaptions will be used:

$$\mathrm{d}r_t = (\theta(t) - \alpha r_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t \tag{7}$$

where α and σ are constant and shared across all option maturities. $\theta(t)$ is picked to replicate the current yield curve $y(t)$.

The problem is then:

$$(\alpha, \sigma) = \boldsymbol{\Theta}(\{Q^{\text{mkt}}\}; \{\tau\}, y(t)) \tag{8}$$

This is a problem shown in QuantLib's Bermudan Swaption example,[1] available both in C++ and Python.

The collected historical data comprises lognormal ATM volatility quotes for sterling from January 2, 2013 to June 1, 2016. The option maturities are 1 to 10 years, plus 15 and 20 years, while the swap terms are from 1 to 10 years, plus 15, 20 and 25 years. This gives a total of 156 swaptions to be used, equally weighted, as calibration instruments.

For the yield curve, the six-month (6M) tenor Libor curve is used. This is bootstrapped using a monotonic cubic spline interpolation of the zero curve and built on top of the overnight index swap curve. Only forward rate agreements and swaps are used to bootstrap the curve. When serving as an input to the neural network, the yield curve is discretely sampled at 44 points: 0, 1, 2, 7 and 14 days; 1 to 24 months; 3 to 10 years; plus 12, 15, 20, 25, 30, 40 and 50 years.

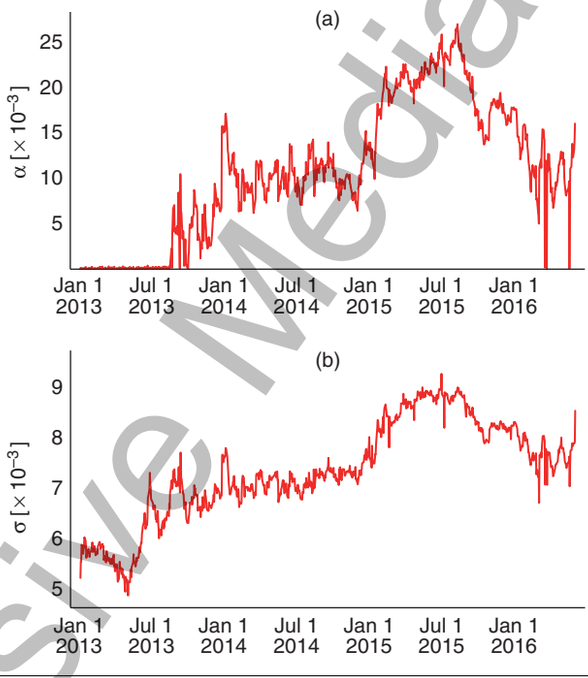
A Levenberg-Marquardt local optimiser is used to minimise the equally weighted average of squared net present value (NPV) differences.[2] The calibration is done twice, with different starting points: first, a default starting point with $\alpha = 0.1$ and $\sigma = 0.01$, and second, the calibrated parameters from the previous day using the default starting point. The results for the default calibration are shown in figure 1.

Once the calibration history is available, the next task is to produce a large enough data set to train a neural network. We start with three-and-a-half years of historical daily data, which comprises only 891 individual examples. What is needed is about two orders of magnitude larger, so just approaching a data provider will not be sufficient. Instead, a statistical model of the sample set is created, and then an artificial training set is generated by sampling from this model. While forming the statistical model, it is observed that scaling the input to have zero mean and unit variance improves the training.

The training set is generated in the following manner:

- Obtain errors for each calibration instrument for each day.
- As parameters are positive, take the natural logarithm on the calibrated parameters.
- Rescale yield curves, parameters and errors to have zero mean and variance 1.
- Apply dimensional reduction via principal component analysis (PCA) to the yield curve, and keep parameters for given explained variance (99.5%).
- Calculate covariance of rescaled log parameters, PCA yield curve values and errors.
- Generate random, normally distributed vectors consistent with given covariance.
- Apply inverse transformations: rescale to original mean, variance and dimensionality, and take exponential of parameters.
- For each sample, select a reference date randomly from the set used for covariance estimation.
- Obtain implied volatility for all calibration instruments and apply random errors to results.

1 Calibrated (a) α and (b) σ using the default starting point



For the hyper-parameter optimisation, the sample set was divided into three parts: 60% served as the training set, 20% served as the cross-validation set and the remaining 20% served as a testing set. The training set was used during the intra-epoch training. The cross-validation set was used to measure the inter-epoch improvement, including the possibility of stopping early. The testing set was used to pick the best configuration.

As mentioned above, the hyper-parameter optimisation was a mixture of grid and manual search, with a truncated grid search over the number of layers, neurons per layer (with all hidden layers having the same number of neurons), the learning rate, dropout rate, the activation function and even the optimiser.

The hyper-parameter optimisation led to the FNN detailed in figure 2. It has four hidden layers, each with 64 neurons using an exponential linear units activation function (Clevert *et al* 2015). A Nesterov Adam optimiser (Dozat see 2015) was used to train it, with a learning rate of 0.001. A standard 500 epochs were used for training, with a sample set of 150,000 samples; however, an early stopping strategy was applied, whereby if no improvement on the cross-validation loss objective was detected for 50 epochs, then learning would be stopped. A dropout rate of $\frac{1}{5}$ was used in all layers.

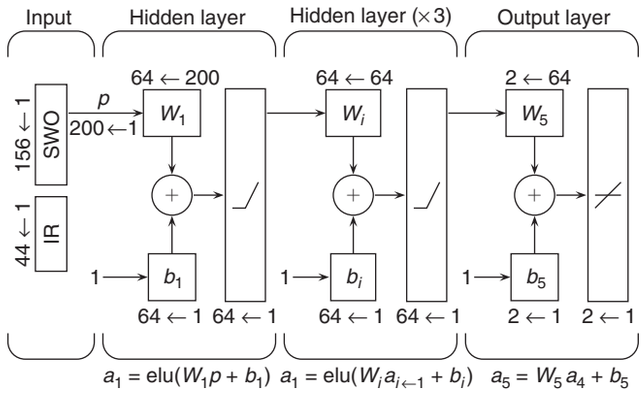
Different variations of CNNs were tried, but as the input matrix of 13×12 is relatively small, no significant improvement was obtained over the best FNN and training time increased significantly. CNNs could become useful for other problems, eg, involving a volatility surface.

Two different neural networks were trained using (1) a sample set produced with a covariance matrix and estimated with historical data from January 2013 to June 2014 (40% of the historical data), and (2) a second sample set using data from January 2013 to June 2015 (~73% of the

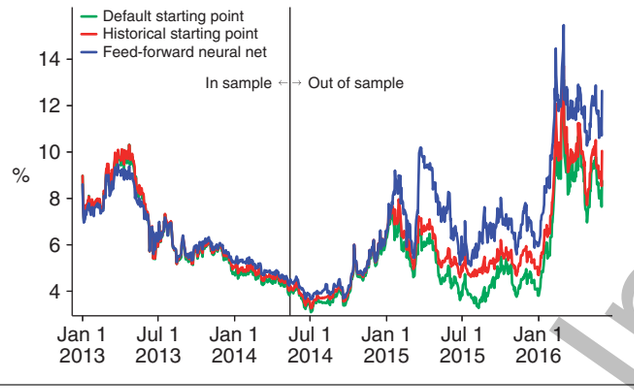
[1] *QuantLib: a free/open-source library for quantitative finance (www.quantlib.org).*

[2] *This is the default calibration objective in QuantLib, but it can be easily changed to have different weights or a different value indicator*

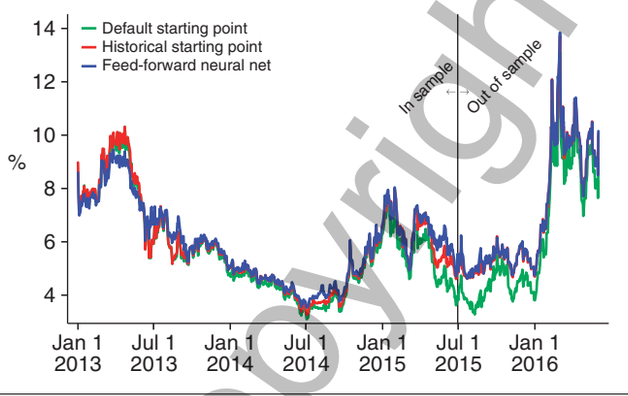
2 FNN used for Hull-White calibration



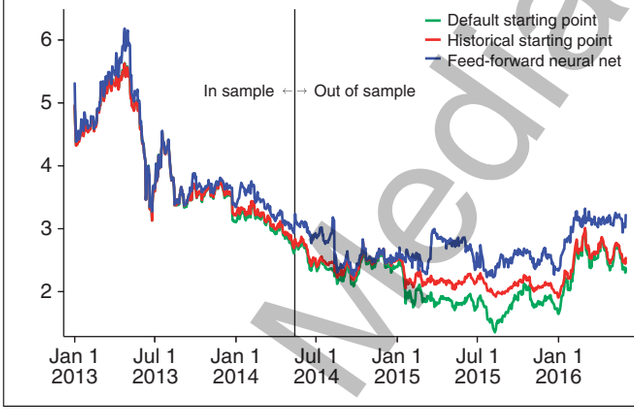
3 Correlation up to June 2014 (average volatility error)



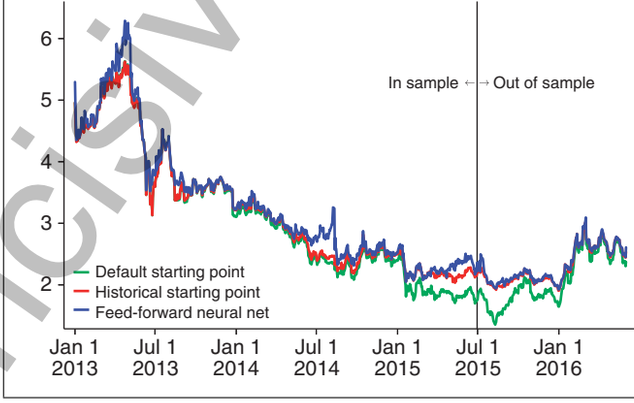
4 Correlation up to June 2015 (average volatility error)



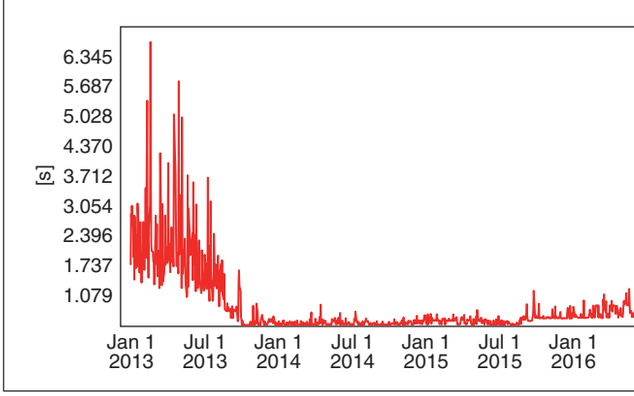
5 Correlation up to June 2014 (NPV mean square error)



6 Correlation up to June 2015 (NPV mean square error)



7 Calibration time with local optimiser and default starting point



historical data). For training, the sample set was split into 80% training set and 20% cross-validation. The testing set will be the historical data itself, which will then serve as backtesting for the model.

The comparison between the model and the calibrations with an optimiser and the two starting points are presented in figures 3–6. The covariance matrix used in the production of the training set has important consequences during backtesting. It is observed that a trained model will behave well out-of-sample only for a period of about six months.

Until now, the speed of the calibration has not been mentioned. The reason for this is the one-factor Hull-White model was picked not because it is considered slow to calibrate, but rather for the ease of calibration. However, while the Hull-White model is generally considered fast to calibrate, it is still instructive to appreciate the difference in calibration speed. The calibration starting from default parameters is presented in figure 7. The average time required is 0.864 seconds, with a standard deviation of 0.738 seconds. However, if one only takes from October 1, 2013 onwards into

consideration, the average drops to 0.563 seconds, with a standard deviation of 0.124 seconds. In contrast, the neural network on average requires 0.0025 seconds, with a standard deviation of 0.0004 seconds. That is over 225 times faster. Of course, the timing will vary depending on the actual implementations and hardware, but it is expected the neural network will be faster. The timing of the neural network is not displayed in figure 7 because it is so much smaller than the local optimiser, and exhibits such small variation, that it is practically indistinguishable from the horizontal axis. However, nothing is free: the time saved during calibration is gained by investing in training the network, which takes about one hour. The configuration is detailed in figure 2.

Conclusions

The use of pricing models in quantitative finance necessitates their calibration to relevant market data in an optimisation process, which can be time consuming. One's choice of model is informed by many considerations, and the time required to perform the calibration is certainly an important one, particularly in settings such as trading desks.

We propose a novel approach for calibration that can offer significant time savings during calibration when compared with the traditional optimisation-based approach. The improvement in performance is achieved by utilising neural networks to approximate the calibration problem, offloading the bulk of the calculations to a training phase, which can be done offline and at a convenient time.

As an example, our approach was tested using a Hull-White model calibrated to 156 swaptions. The out-of-sample back-test showed good behaviour for between six months and one year after the end of the period used in the correlation estimation. After that, a significant degradation of the calibration performance of the neural network was observed.

The performance degradation beyond six months is not problematic, as one could simply retrain the model periodically in order for it to remain within acceptable bounds. Also, in critical environments, safeguards could be set up to monitor the reliability of the solution. For example, a periodic parallel calibration using the traditional approach could be used to ascertain the reliability of the neural network *vis-a-vis* the current market state.

Overall, it can be seen that the methodology described in this article presents a substantial time improvement. Moreover, the time required to calibrate the model upon being presented with new inputs is quite uniform, as shown by the very low standard deviation in the example presented. This is a direct consequence of figure 2. While the training might be time consuming, the actual calibration involves multiplying only a few matrices and evaluating a handful of exponentials. The speed improvement will carry over directly to more complicated models.

Further points of study could involve using a parameterised correlation. Such a possibility would enable training with states that reflect not only the current market environment but also possible future developments. This could extend the lifespan of a trained model. It is important to mention at this point that here, unlike in Monte Carlo calculations, it is not necessary for the sampling set to be able to correctly predict how future market conditions will look statistically; it must simply contain samples in sufficient quantities that mimic such possible conditions. ■

Andres Hernandez is a manager at PwC Deutschland's financial services practice in Frankfurt. **The author would like to thank the referees for their helpful comments.**
Email: andres.hernandez@gmx.net.

REFERENCES

Bergstra J and Y Bengio, 2012
Random search for hyper-parameter optimization
Journal of Machine Learning Research 13, pages 281–305

Bollen J, H Mao and X-J Zeng, 2010
Twitter mood predicts the stock market
Preprint, available at https://arxiv.org/abs/1010.3003

Clevert D-A, T Unterthiner and S Hochreiter, 2015
Fast and accurate deep network learning by exponential linear units (ELUs)
Preprint, available at http://arxiv.org/abs/1511.07289

Connor JT, RD Martin and LE Atlas, 1994
Recurrent neural networks and robust time series prediction
IEEE Transactions on Neural Networks 5(2), pages 240–254

Cybenko G, 1989
Approximations by superpositions of sigmoidal functions
Mathematics of Control, Signals, and Systems 2(4), pages 303–314

Dozat T, 2015
Incorporating Nesterov momentum into Adam
Preprint, available at http://cs229.stanford.edu/proj2015/054_report.pdf

Frank RJ, N Davey and SP Hunt, 2011
Time series prediction and neural networks
Journal of Intelligent and Robotic Systems 31(1), pages 91–103

Hornik K, 1991
Approximation capabilities of multilayer feedforward networks
Neural Networks 4(2), pages 251–257

Klos M and Z Waszczyszyn, 2011
Modal analysis and modified cascade neural networks in identification of geometrical parameters of circular arches
Computers and Structures 89, pages 581–589

Mareš T, E Janouchová and A Kucerová, 2015
Artificial neural networks in calibration of nonlinear mechanical models
Preprint, available at https://arxiv.org/abs/1502.01380

Novák D and D Lehký, 2006
ANN inverse analysis based on stochastic small-sample training set simulation
Engineering Applications of Artificial Intelligence 19, pages 731–740

Snoek J, H Larochelle and R Prescott Adams, 2012
Practical Bayesian optimization of machine learning algorithms
In *Advances in Neural Information Processing Systems*, volume 528
NIPS

Zaw K, GR Liu, B Deng and KBC Tan, 2009
Rapid identification of elastic modulus of the interface tissue on dental implants surfaces using reduced-basis method and a neural network
Journal of Biomechanics 42, pages 634–641

Zhang L, L Li, H Ju and B Zhu, 2010
Inverse identification of interfacial heat transfer coefficient between the casting and metal mold using neural network
Energy Conversion and Management 51, pages 1898–1904