

CHARLES UNIVERSITY
FACULTY OF SOCIAL SCIENCES

Institute of Economic Studies



**Recurrent Neural Networks use for Value
at Risk estimation. Application to the
Visegrád Four stock market indexes.**

Bachelor's thesis

Author: Nikanor Goreglyad

Study program: Economics and Finance

Supervisor: Mgr. Marek Hauzr

Year of defense: 2020

Declaration of Authorship

The author hereby declares that he or she compiled this thesis independently, using only the listed resources and literature, and the thesis has not been used to obtain any other academic title.

The author grants to Charles University permission to reproduce and to distribute copies of this thesis in whole or in part and agrees with the thesis being used for study and scientific purposes.

Prague, July 26, 2020

Nikanor Goreglyad

Abstract

abstract

JEL Classification C22, C45, C51

Keywords Value at risk, VaR, Recurrent Neural Networks,
Value at risk estimation, Visegrád Four, V4.

Title Recurrent Neural Networks use for Value at Risk
estimation. Application to the Visegrád Four
stock market indexes.

Author's e-mail 40038969@fsv.cuni.cz

Supervisor's e-mail reader@fsv.cuni.cz

Abstrakt

Klasifikace JEL C22, C45, C51

KlÁšterovÁ slova Value at risk, VaR, Rekurentní neuronové
sítě, Value at risk estimation, Visegrádská
skupina, V4.

NÁzev práce Použití rekurentních neuronových sítí pro
odhad Value at Risk. Aplikace na akciové
indexy Visegrádské skupiny.

E-mail autora 40038969@fsv.cuni.cz

E-mail vedoucího práce reader@fsv.cuni.cz

Acknowledgments

The author is grateful especially to Mgr. Marek Hauzr,

Typeset in L^AT_EX using the IES Thesis Template.

Bibliographic Record

Goreglyad, Nikanor: *Recurrent Neural Networks use for Value at Risk estimation. Application to the Visegrád Four stock market indexes..* Bachelor's thesis. Charles University, Faculty of Social Sciences, Institute of Economic Studies, Prague. 2020, pages 38. Advisor: Mgr. Marek Hauzr

Contents

List of Tables	vii
List of Figures	viii
Acronyms	ix
Thesis Proposal	x
1 Introduction	1
2 Literature review	3
2.1 On Recurrent Neural Networks	3
2.2 On chosen benchmark models and their predecessors	4
2.2.1 ARCH family	4
2.2.2 Extreme Value Theory	4
3 Methodology	5
3.1 Benchmark models	5
3.1.1 FIAPARCH model	5
3.1.2 EVT-POT model	5
3.2 Recurrent Neural Networks	7
3.2.1 Feed-forward Neural networks	7
3.2.2 Recurrent Neural Network Architecture.	9
3.2.3 Backpropagation.	11
3.2.4 Learning specifics of RNN.	11
3.3 Backtesting	12
3.3.1 Kupiec's Proportion of Failures test	13
3.3.2 Christoffersen's Joint Test of Coverage and Independence	14

4 Data	15
4.1 Description	15
4.2 Preprocessing	17
5 Results and conclusion	18
5.1 Empirical results	18
5.2 Conclusion	19
Bibliography	23
A Title of Appendix A	I
B Title of Appendix B	II

List of Tables

4.1 Descriptive statistics of daily index returns	16
---	----

List of Figures

3.1	Single neuron	8
3.2	General architecture of feed-forward neural network.	8
3.3	10
4.1	Histogram of daily index returns	16

Acronyms

VaR Value at Risk

RNN Recurrent Neural Network

POF-test Proportion of failure test

Bachelor's Thesis Proposal

Author	Nikanor Goreglyad
Supervisor	Mgr. Marek Hauzr
Proposed topic	Recurrent Neural Networks use for Value at Risk estimation. Application to the Visegrád Four stock market indexes.

Motivation Value at Risk (VaR) is a comprehensive tail-related market risk quantitative measure. In recent years it has become the most widely used technique for measuring future expected risk in both financial and commercial institutions. Unfortunately, due to certain properties of financial time series data (i.e. heteroscedasticity, heavy-tailedness, skewness, etc.), VaR estimation appeared to be a tough statistical proposition and none of the traditional methods has achieved convincing results. Recurrent Neural Networks have confirmed to be one of the most efficient instruments to process time series data. While the traditional neural networks assume, that inputs are pairwise independent, Recurrent NNs apply the same task on every element of sequential data with output dependent on previous operations. This characteristic and capability to overcome limitations of the traditional forecasting techniques make RNNs a promising alternative approach for risk management purposes. This paper aims to investigate the efficiency of Recurrent Neural Networks for predicting Value at Risk and their ability to overperform traditional statistical models. For comparison EVT-POT model and GARCH(p, q) have been chosen. According to several empirical papers, the approach based on the EVT could be considered as the most accurate for estimating VaR. (Abad, P., et al., 2013) Among of the GARCH family, the GARCH(p, q) model overperforms others in terms of VaR estimation. (So and Yu, 2006)

Methodology Recurrent Neural Network will be applied to the forecasting of 1%, 2.5% and 5% Value at Risk of PX, BUX, WIG20, and SAX indexes. The model will be trained on historical daily market data. The evaluation of RNN forecast will be

done by comparing with those by GARCH(p, q) and EVT-based dynamic approach using tests of Kupiec (1995) and Christoffersen. (1998)

Expected Contribution Despite the vast amount of theoretical and applied research on the neural networks approach in finance (White, 1988; Enke & Thawornwong, 2005; Pham & Liu, 1995), there is little in terms of risk management. This study advances the existing literature by providing empirical evidence of the adequacy of RNN use in modeling and forecasting Value at Risk in emerging stock markets.

Outline

1. Abstract
2. Introduction and summary
3. Literature review
 - 3.1. On Recurrent Neural Networks
 - 3.2. On traditional VaR estimation models
4. Methodology
 - 4.1. GARCH(p,q) model
 - 4.2. EVT-POT model
 - 4.3. RNNs
5. Data
 - 5.1. Description
 - 5.2. Preprocessing
6. Results
 - 6.1. GARCH(p,q) model
 - 6.2. EVT-POT model
 - 6.3. RNNs
 - 6.4. Comparison
7. Conclusion

Core bibliography

- Abad, P., et al. (2013). A comprehensive review of Value at Risk methodologies. *The Spanish Review of Financial Economics*, 12(1), 15-32. doi: 10.1016/j.srfe.2013.06.001
- Christoffersen P.F. (1998). Evaluating Interval Forecasts. *International Economic Review*, 39(4), 841-862. doi: 10.2307/2527341.
- Donaldson, R. G. and Kamstra, M. (1996) Forecast combining with neural networks. *Journal of Forecasting*, 15, 49-61. doi:10.1002/(SICI)1099-131X(199601)15:1<49::AID-FOR604>3.0.CO;2-2
- Kupiec P.H. (1995). Techniques for Verifying the Accuracy of Risk Measurement Models. *The Journal of Derivatives*, 3(2), 73-84. doi: 10.3905/jod.1995.407942
- Olah, C. (2015, August 27). Understanding LSTM Networks. Retrieved from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- So, M., Yu, P. (2006). Empirical analysis of GARCH models in value at risk estimation. *Journal of International Financial Markets, Institutions & Money*, 16, 180â€“197.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, 1929â€“1958. Retrieved from: <http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf>
- Taylor, J. W. (2000): â€śA Quantile Regression Neural Network Approach to Estimating the Conditional Density of Multiperiod Returns.â€ť *Journal of Forecasting* 19: pp. 299â€“311
- Widrow, B., Rumelhart, D. E., and Lehr, M. A. (1997). Neural networks: Applications in industry, business and science. *Communications of the ACM*, 37(3).
- White, H. (1988). Economic Prediction using Neural Networks: The Case of IBM Daily Stock Returns. *Proceedings of the IEEE International Conference on Neural Networks*, 2(2), 451 - 458.
- Enke, D., Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, 29, 927-940.
- Pham D.T., Liu X. (1995) Financial Prediction Using Neural Networks. In: *Neural Networks for Identification, Prediction and Control*. Springer, London

Author

Supervisor

Chapter 1

Introduction

Value-at-Risk (VaR) is a relatively simple yet very convenient risk measurement technique in finance. VaR measures the loss in value of asset or portfolio over a certain period for a given confidence interval. Appearing in the 1980's VaR has come a long way to become one of the most common risk assessment tools used by financial institutions and investors. The general simplicity of concept and the Basel Committee of Bank Supervision contributed to its popularity a lot by the imposition of VaR use on financial institutions. Also, in 1997 the Securities and Exchange Commission suggested using VaR to report their market risk exposure.

Despite VaR's popularity, VaR models have frequently been criticized for being dependent on assumptions that are not necessarily valid, regarding specific properties of financial data, such as skewness, heteroskedasticity, and heavy-tailedness. Under these circumstances, we can expect machine learning, the study that has gained tremendous popularity in recent times, to be useful due to the fact that most machine learning algorithms do not require any assumption to be made, i.e., are non-parametric. One particular set of algorithms, Neural networks, has already proved their reliability when applied to financial data.

This thesis aims to examine the ability of neural networks with recurrent architecture to forecast the value-at-risk market indices of developing countries. For this purpose, Long Short-term Memory (LSTM) architecture, as one already proofed its capability of Value-at-Risk forecasting of market indices of developed countries, is chosen. The assessment is made by comparing it with one parametric model and one semi-parametric. Chosen benchmark models are the fractionally integrated asymmetric power autoregressive conditional

heteroskedasticity (FIAPARCH) model and the Extreme Value Theory Peak Over Threshold (EVT-POT) model. Both models are considered top performers in their classes (Abad *et al.* 2014).

Chapter 2

Literature review

2.1 On Recurrent Neural Networks

Over the past decade, a vast number of machine learning techniques have been developed and widely adopted to extract information from various kinds of data (Carrio *et al.* (2017), Bengio (2009), Khan & Yairi (2018)). There are several types of architectures for machine learning, which are more or less appropriate for different characteristics of input data. In research areas where researchers have to deal with sequential data, such as text, audio or financial time-series, the recurrent neural networks (RNN) thoroughly prevail (Werbos (1988), Robinson & Fallside (1987), Pearlmutter (1989), Gallagher *et al.* (2005)),

The typical feature of the RNN architecture is a recurrent connection, which enables the RNN to take into account not only input data but also past states of hidden layers. These networks, such as fully recurrent NNs (Jordan, Chen & Soo (1996)) and selective RNNs (Šter (2013)), consisting of standard recurrent cells, are unable to connect the relevant information when dealing with relatively big gaps between relevant input data. To overcome this issue Hochreiter & Schmidhuber (1997) proposed long short-term memory (LSTM) architecture.

After LSTM was introduced, its learning capacity was deservedly appreciated by researchers from different fields and has been employed in tremendous number of studies, such as sentence embedding (Palangi *et al.* (2016)), speech recognition (Hsu *et al.* (2016), Soltau *et al.* (2016), He & Droppo (2016)), acoustic modeling (Zeyer *et al.* (2017) and last but not least finances (Siami-Namini & Namin (2018), Fischer & Krauss (2018), Hwang (2020))

«« add more about finance and financial risks related studies, consider

gibryd models as well, at least one paper on VaR (emerging market: Reghin & Lopes (2019) and developed: ...)»»

2.2 On chosen benchmark models and their predecessors

2.2.1 ARCH family

2.2.2 Extreme Value Theory

Main purpose of Extreme Value Theory (EVT) is providing models estimating tail behavior of statistical distributions. Initially theory found multiple application in hydrology (Rossi *et al.* (1984), Frances *et al.* (1994)) and only in last couple decades there has been an increasing number of studies, where theory was applied to risk management (Gilli *et al.* (2006), Diebold *et al.* (1998)) and Value at Risk in particular (Bali (2003), Gencay & Selcuk (2004)).

There two main methods to identify extremes in real data, the method of block over maxima (BM), proposed by Fisher & Tippett (1928), and the method of peaks over threshold (POT), approach based on theorem stated by Balkema & De Haan in 1974. In this paper for choosing threshold graphical tool, proposed by Davison & Smith (1990), namely Mean Residual Life plot, is used. following the methodology employed by Bali (2003).

Our motivation to use EVT-POT model as benchmark in this paper is based on "A comprehensive review of VaR methodologies" by Abad *et al.* (2014), where its performance has been ranked as one of the best compared to a wide variety of statistical models.

Chapter 3

Methodology

3.1 Benchmark models

3.1.1 FIAPARCH model

This section is organized as follows. Subsection 3.1.1 is dedicated to methodology of parametric benchmark model. In Section 3.1.1 fundamentals of parametric models family ARCH and specifically generalized ARCH model. In Section 3.1.1 FIAPARCH model is explained together with its derivation from ancestors. Explanation of density function chosen for Value at Risk modeling could be found in Section 3.1.1. to be completed

ARCH Family

FIA-PARCH

Density function

3.1.2 EVT-POT model

The central assumption made within The Extreme Value Theory (EVT) method is that distribution of extreme returns is observed over a long time, independent of the distribution of the returns themselves. There are two main approaches for determination what returns are extreme: Block Maxima (BM), where time series data split into sections and maximum values from each section are considered as extreme returns, and more modern approach Peak Over Threshold (POT). According to this method all values over fixed threshold are qualified as extreme returns. In this study, POT approach is employed as more efficient in using of the financial data at extreme values.

Generalized Pareto Distribution

According to Balkema & De Haan (1974), the distribution of exceedances over threshold is Generalized Pareto Distribution.

$$G_{\sigma,\xi}(y) = \begin{cases} 1 - (1 + \frac{\xi}{\sigma}y)^{-1/\xi}, & \text{if } \xi \neq 0 \\ 1 - \exp(-y/\sigma), & \text{if } \xi = 0 \end{cases} \quad (3.1)$$

For vector of exceedances \mathbf{y} of length N_u defined as $y_i = r_i - u$, where r_i represents financial returns at time i and u is threshold, the distribution can be defined as follows

$$F_u(y) = P(r - u < y | r > u) = \frac{F(y + u) - F(u)}{1 - F(u)} \quad (3.2)$$

By substituting $F(y)$ with GPD from equation 3.1 in equation 3.2, we get

$$F(r) = F(y + u) = (1 - F(u))(1 - (1 + \frac{\xi}{\sigma}(r - u))^{-1/\xi}) + F(u), r > u \quad (3.3)$$

As we substitute $F(u)$ with proportion of data in tail $(1 - \frac{N_u}{n})$, tail estimator is defined as follows

$$F(r) = 1 - \frac{N_u}{n}(1 + \frac{\xi}{\sigma}(r - u))^{-1/\xi}, r > u \quad (3.4)$$

For a given probability $(1 - \alpha) \rightarrow F(u)$, the VaR_α estimate is calculated by inverting the tail estimation formula

$$VaR_\alpha = u + \frac{\sigma}{\xi} \left(\left(\frac{n\alpha}{N_u} \right)^{-\xi} - 1 \right) \quad (3.5)$$

Parameters σ and ξ could be estimated using Maximum Likelihood Estimation:

$$\mathcal{L}(\xi, \sigma | y) = \begin{cases} -n \log \sigma - (\frac{1}{\xi} + 1) \sum_{i=1}^n \log(1 + \frac{\xi}{\sigma} y_i), & \text{if } \xi \neq 0 \\ -n \log \sigma - \frac{1}{\sigma} \sum_{i=1}^n \log(1 + \frac{\xi}{\sigma} y_i), & \text{if } \xi = 0 \end{cases} \quad (3.6)$$

Choosing threshold

Setting an appropriate threshold u is not that trivial. If threshold set too large there will be too few values to model the tail of the distribution correctly as the variance is likely to be large due to only very extreme observations. On the other hand a low threshold will include too many values, therefore giving a high

bias. Davison & Smith (1990) suggest Mean Residual Life plot (MLR-plot) as efficient graphical tool to find an appropriate balance between the variance and the bias, when setting threshold. The MRL-plot uses the expectation value of the GPD excesses.

$$E(X - u|X > u) = \frac{\beta_u}{1 - \gamma}, \quad (3.7)$$

where β_u is the scale parameter given threshold u and γ is the shape parameter which needs to be defined. Then for any $v > u$ expected value of excesses is

$$E(X - v|X > v) = \frac{\beta_u + \gamma v}{1 - \gamma} \quad (3.8)$$

On the plot mean excesses based on equation 3.8 appropriate threshold can be set, plot starts showing a linear behavior. Also plot is likely to lose its linear behavior when the threshold gets too large due to the variance of the few extremes left will cause the plot to jump. «<add MLR plots for indexes»»

3.2 Recurrent Neural Networks

In the following section, the basic concept of neural networks is explained. Then differences between feed-forward and recurrent architecture are discussed. After it, the training algorithm is explained. Lastly, learning specifics of RNN are mentioned, and the concept of gated unit explained.

3.2.1 Feed-forward Neural networks

The neural network is a learning algorithm that consists of a large number of simple elements, called neurons. These neurons are organized in interconnected layers. The number of neurons per layer, as well as the number of layers, can be changed depending on the tasks facing the network. This adaptive structure allows NN to emulate almost any function. Its emulation, nevertheless, is limited mostly by the availability of computing power and training data. That explains why recently developed accessibility of graphic processing units and cloud computing has contributed to the development of public interest in neural networks. The first layer represents inputs of the network, and the last layer represents outputs. They are known as the input layer and output layer, respectively. Layers between input and output layers are hidden layers. NN contains at least one hidden layer.

Figure 3.1: Single neuron

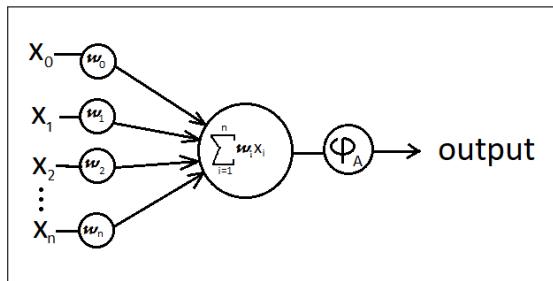


Figure 3.1 shows the structure of a single neuron. Each neuron excepts neuron in the input layer receives outputs of each neuron of the previous layer. Each input is multiplied by the weighted sum of inputs is then transformed by an activation function. The activation function is supposed to be continuous and differentiable. The most common are linear, sigmoid, Hyperbolic tangent (TanH), and Rectified Linear Unit (ReLU). The explanation of different activation functions lies outside the scope of this work and could be found in “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning.”(Nwankpa *et al.* 2018) Therefore, output of neuron is defined as

$$y = \varphi_A(\sum_{i=0}^n w_i x_i), \quad (3.9)$$

where \$x_i \in \mathbb{R}\$ for \$i \in \{1, 2, \dots, n\}\$ is output of neuron \$i\$ from previous layer of size \$n\$, \$x_0\$ is constant, \$w_i \in \mathbb{R}\$ for \$i \in \{0, 1, \dots, n\}\$ is respective synaptic weight of input and, finally, \$\varphi_A\$ is an activation function.

Figure 3.2: General architecture of feed-forward neural network.

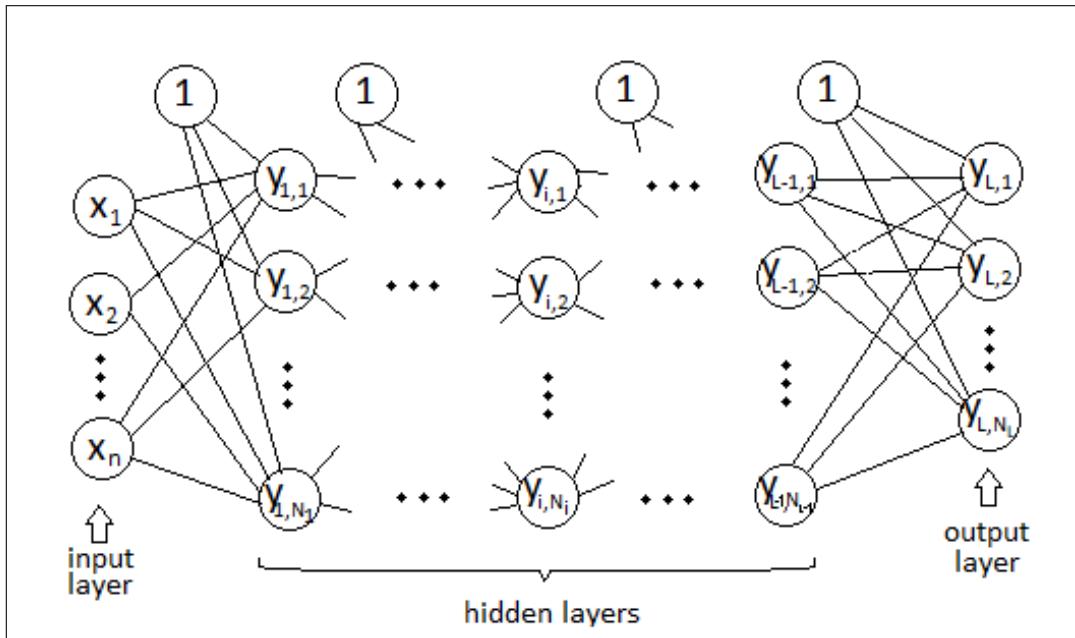


Figure 3.2 illustrates structure of feedforward neural network with $L - 1$ hidden layers, input vector $\mathbf{x} \in \mathbb{R}^n$ and vector of output $\mathbf{y}_L \in \mathbb{R}^{N_L}$. Number of neurons in i^{th} layer is N_i for $i \in \{1, 2, \dots, L\}$. Then output of $(i + 1)^{\text{th}}$ layer for $i \in \{1, 2, \dots, L - 1\}$ is defined as

$$\mathbf{y}_{i+1} = \varphi_{i+1}((1, \mathbf{y}_i) \mathbf{W}_{i+1}), \quad (3.10)$$

where

$$\mathbf{y}_{i+1} \in \mathbb{R}^{N_{i+1}},$$

$$\mathbf{y}_i \in \mathbb{R}^{N_i} \text{ is output vector of layer } i,$$

$$\mathbf{W}_{i+1} \in M(N_i + 1 \times N_{i+1}) = \begin{pmatrix} w_{10}^{i+1} & w_{20}^{i+1} & \dots & w_{N_{i+1}0}^{i+1} \\ w_{11}^{i+1} & \ddots & & w_{N_{i+1}1}^{i+1} \\ \vdots & & \ddots & \vdots \\ w_{1N_i}^{i+1} & w_{2N_i}^{i+1} & \dots & w_{N_{i+1}N_i}^{i+1} \end{pmatrix},$$

where $w_{js}^{i+1} \in \mathbb{R}$ is synoptic weight between j^{th} neuron of i^{th} layer and s^{th} neuron of $(i + 1)^{\text{th}}$ layer,

$\varphi_{i+1} = (\underbrace{\varphi_{i+1}, \varphi_{i+1}, \dots, \varphi_{i+1}}_{N_{i+1} \text{ times}})$, where φ_{i+1} is activation function of $(i + 1)^{\text{th}}$ layer.

3.2.2 Recurrent Neural Network Architecture.

When applying machine learning techniques to financial data, the feed-forward neural network's main disadvantage is the inability to employ temporal dynamic behavior of data. In feed-forward architecture, all samples are considered to be independent. In light of this fact, recurrent neural network (RNN), a special type of neural network architecture that can capture patterns in data over time, could be considered more suitable for sequential data. In recurrent architecture output of neurons is transmitted not only into the neurons in the next layer but also back into its own input. Therefore, information from previous time steps is captured encoded into hidden state variables and used to influence past computational processes.

Figure 3.3

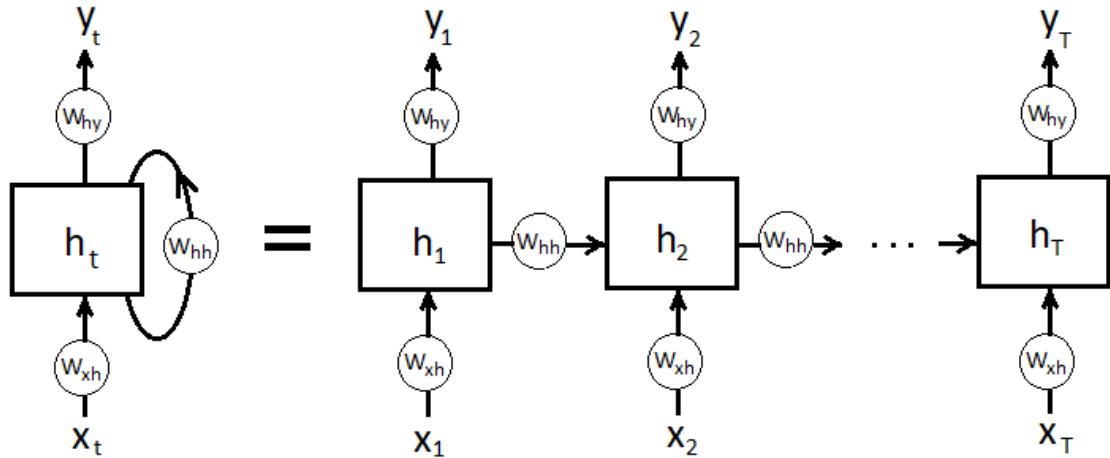


Figure 3.3 shows how layers for each time step share weights across time. h_t represents hidden state of RNN in time t . It is defined as

$$h_t = \varphi(\mathbf{W}_{xh}x_t + \mathbf{W}_{hh}h_{t-1} + b_h), \quad (3.11)$$

where

- $x_t \in \mathbb{R}^n$ is input vector at time t ,
- $h_{t-1} \in \mathbb{R}^m$ is hidden state at time $t - 1$,
- $b_h \in \mathbb{R}^m$ is bias vector,
- $\mathbf{W}_{xh} \in M(m \times n)$ is matrix of weights at input,
- $\mathbf{W}_{hh} \in M(m \times m)$ is matrix of weights at hidden state,
- for $n \in \mathbb{N} \setminus \{0\}$ and $m \in \mathbb{N} \setminus \{0\}$,
- φ is an activation function applied to hidden neurons.

Here Then output of RNN is defined as

$$y_t = \psi(\mathbf{W}_{hy}h_t + b_y), \quad (3.12)$$

where

- $y_t \in \mathbb{R}^k$ is output vector at time t ,
- $b_y \in \mathbb{R}^k$ is bias vector,
- $\mathbf{W}_{hy} \in M(m \times k)$ is matrix of weights at output,
- for $k \in \mathbb{N} \setminus \{0\}$ and $k \in \mathbb{N} \setminus \{0\}$,
- ψ is activation function applied to output.

3.2.3 Backpropagation.

At the moment it is worth mentioning that the training of neural networks is of two types: supervised and unsupervised. In supervised learning, the network is trained on data set with known correct outputs, while in unsupervised learning network is expected to discover information itself, without output provided. Unsupervised learning is less limited by the quantity and quality of available data and might be more efficient in solving classification problems. (Sathya & Abraham 2013) On the other hand supervised learning provides more tools to optimize the network's performance. Since the supervised learning approach was chosen for this work, unsupervised learning is not discussed.

The training process in supervised learning is, in point of fact, deduction of function composition able to map inputs on outputs. The training process can be described as the 5 following steps. The first step is assigning random weights to each neuron. The second step is forward propagation of training set into NN to compute output. By training set, we mean set of the inputs with the known true values of the outputs. Therefore, the next step error produced by NN with randomly assigned weights is calculated based on a difference of just obtained output and correct output. In the fourth step, the error is passed back into the network to determine the error for each particular neuron. This process, known as back-propagation, will be discussed in the next paragraph. The last step is updating the weights of each neuron to minimize its error. While the rest of the steps are quite obvious, understanding of backpropagation might be a bit challenging.

««BAckprapogation»>

3.2.4 Learning specifics of RNN.

Backpropagation could be applied on any network with an ordered structure including RNN. In the case of RNN derivatives are computed and propagated back in time algorithm to change weights at hidden state. Therefore, for RNN training the same backpropagation algorithm could be used.

Nevertheless, derivatives of activation functions of certain neurons could be close to 0 and in case of long sequences when it happens on several layers gradient values rapidly shrink and gradient may completely vanish. This problem is known as the gradient vanishing problem. A similar case is the exploding gradient problem when gradient has large values and the training process produces unstable parameters. This problem, however, could be solved by setting an

appropriate threshold for gradient values. To overcome the vanishing gradient problem the concept of gated RNN units was proposed.

This concept was implemented in several RNN architectures, the most well known are Long Short-term Memory (LSTM) and Gated Recurrent Units. For purposes of this paper, LSTM architecture was chosen. LSTM neurons, commonly called "Memory blocks", each have three gate vectors. The input gate multiplies the input vector of neuron, therefore it determines what proportion of the current input vector is relevant and will affect the hidden state of neurons. The output gate controls what information in the hidden state is relevant for current output. The forget gate is responsible for the determination of relevant information coming from the previous hidden state. This gated structure allows LSTM to overcome vanishing gradient problem and produce stable results.

3.3 Backtesting

After building the predictive models for Value at Risk, the only step missing is to evaluate and compare its accuracy. Backtesting is a statistical procedure where actual profits and losses are compared with model forecasts for test data. There are two main types of backtesting procedures. The first one is Unconditional coverage tests that examine whether the frequency of exception (events when actual returns appear to be less than Value at Risk estimate) over a certain period corresponds to the selected confidence level. E.g., if 95% confidence level of VaR is chosen, the exception is expected to occur five times in 100 days. Unconditional coverage tests take into account the number of exceptions but ignore when exactly during this specified period exceptions occur. Another type of backtesting is conditional coverage tests. Conditional coverage tests assess not the only number of exceptions but whether predicted exceptions are independent. As there is no direct way to test independence, tests address specific properties of independence, such as autocorrelation or clustering. In the empirical part of this paper Kupiec's (1995) proportion of failure test (POF-test) is used for assessment of unconditional coverage of models. Assessment of conditional coverage is covered with Christoffersen's (1998) Joint Test of Coverage and Independence. In the following section, a brief insight into two methods used in the empirical part is provided.

3.3.1 Kupiec's Proportion of Failures test

As mentioned above, Kupiec's POF-test assesses whether the frequency of exceptions is consistent with the chosen quantile of loss Value at Risk measure. Consider quantile of loss $(1 - \alpha)$, exceptions indicator can be defined as follows:

$$I_t = \begin{cases} 1 & \text{if } r_t < VaR_\alpha \\ 0 & \text{if } r_t \geq VaR_\alpha \end{cases} \quad (3.13)$$

Under assumption that probability of exception is constant, number of exceptions x could be defined as follows:

$$x = \sum_{t=1}^N I_t \sim B(N, \alpha), \text{ where} \quad (3.14)$$

N is total number of observations. Estimator for quantile of loss then defines as

$$\hat{\alpha} = \frac{\sum_{t=1}^N I_t}{N} \quad (3.15)$$

To find out whether the estimated rate is significantly different from theoretical α , null hypothesis for POF-test is

$$H_0 : \hat{\alpha} = \alpha \quad (3.16)$$

Kupiec (1995) proposes that the POF-test is best conducted as a likelihood-ratio test. Therefore statistic takes the form

$$LR_{uc} = -2 \ln \left(\frac{(1 - \alpha)^{T-x} \alpha^x}{(1 - \hat{\alpha})^{T-x} \hat{\alpha}^x} \right) \quad (3.17)$$

and asymptotically χ^2 distributed with one degree of freedom. The null hypothesis will then be rejected (meaning model is rejected) if the statistic exceeds the critical value of the χ^2 distribution. The following table contains critical values for sample sizes and confidence levels studied in the empirical part of the present paper.

«<table»>

3.3.2 Christoffersen's Joint Test of Coverage and Independence

As it is suggested by name Christoffersen's (1998) Joint Test consists of two parts, test of coverage and test of independence. For coverage Christoffersen proposes a test similar to that of Kupiec. His independence test examines the frequency of consecutive exceptions. Using same definition of exceptions indicator as in (3.13), we define $n_{ij}, i, j \in \{0, 1\}$ as the number of observations when condition j occurred assuming that condition i occurred in the previous observation. Then using this notation estimator for the probability of exception conditional on state i of the previous observation can be defined as

$$\hat{\pi}_i = \frac{n_{i1}}{n_{i0} + n_{i1}} \quad (3.18)$$

and estimator for probability of exception to occur is

$$\hat{\pi} = \frac{n_{01} + n_{11}}{n_{01} + n_{11} + n_{10} + n_{00}} \quad (3.19)$$

If the model predicts exceptions accurately, occurrences of exceptions should be independent of previous observation's state. Otherworlds, probabilities π_0 and π_1 have to be equal. Therefore the null hypothesis for independence test is defined as

$$H_0 : \hat{\pi}_0 = \hat{\pi}_1 \quad (3.20)$$

Then test statistic takes form

$$LR_{ind} = -2 \ln \left(\frac{(1 - \pi)^{n_{00} + n_{10}} \pi^{n_{01} + n_{11}}}{(1 - \pi_0)^{n_{00}} \pi_0^{n_{01}} (1 - \pi_1)^{n_{10}} \pi^{n_{11}}} \right) \stackrel{asy}{\sim} \chi^2_1 \quad (3.21)$$

Finally, to form conditional coverage test we combine test of independence (3.21) and test of unconditional coverage (3.17)

$$LR_{cc} = LR_{uc} + LR_{ind} \stackrel{asy}{\sim} \chi^2_2 \quad (3.22)$$

Then the model is rejected if statistic exceeds the critical value of the χ^2 distribution. It is worth noting that the model could be accepted even though it was rejected by one of the constituent tests.

Chapter 4

Data

Empirical study is performed on daily data on financial indices of emerging markets. Distribution of returns on emerging markets commonly characterized by heavier tails, i. e. extreme events happened there more frequently. Therefore, precise estimate of Value at Risk might be more critical for emerging markets than elsewhere. One of the first analyses recognizing inefficiency of traditional VaR methodologies in emerging markets was done by Parrondo (1997). Aragonés *et al.* (2000) suggest that focus of traditional methodologies on whole distribution and, therefore ignoring rare events, as well as limitation made by VaR assumptions, lead to insufficient performance in terms of VaR modeling.

4.1 Description

For current research leading indices of Visegrad Four stock exchanges were chosen:

- **PX** is the leading index of Prague Stock Exchange. It is calculated as a market-capitalization-weighted price index of 12 stocks with base date April 5th, 1994.
- **SAX** is the leading index of Bratislava Stock Exchange. It is calculated as a market capitalization-weighted total return index of (currently) 7 stocks with base date September 14th, 1993.
- **BUX** is the leading index of Budapest Stock Exchange. It is calculated as a market-capitalization-weighted performance index of 12 to 25 stocks with base date January 2nd, 1991.

- **WIG20** is the leading index of Warsaw Stock Exchange. It is calculated as a market-capitalization-weighted price index of 20 stocks with base date April 16th, 1994.

The descriptive statistics of these indices daily return time series, calculated as

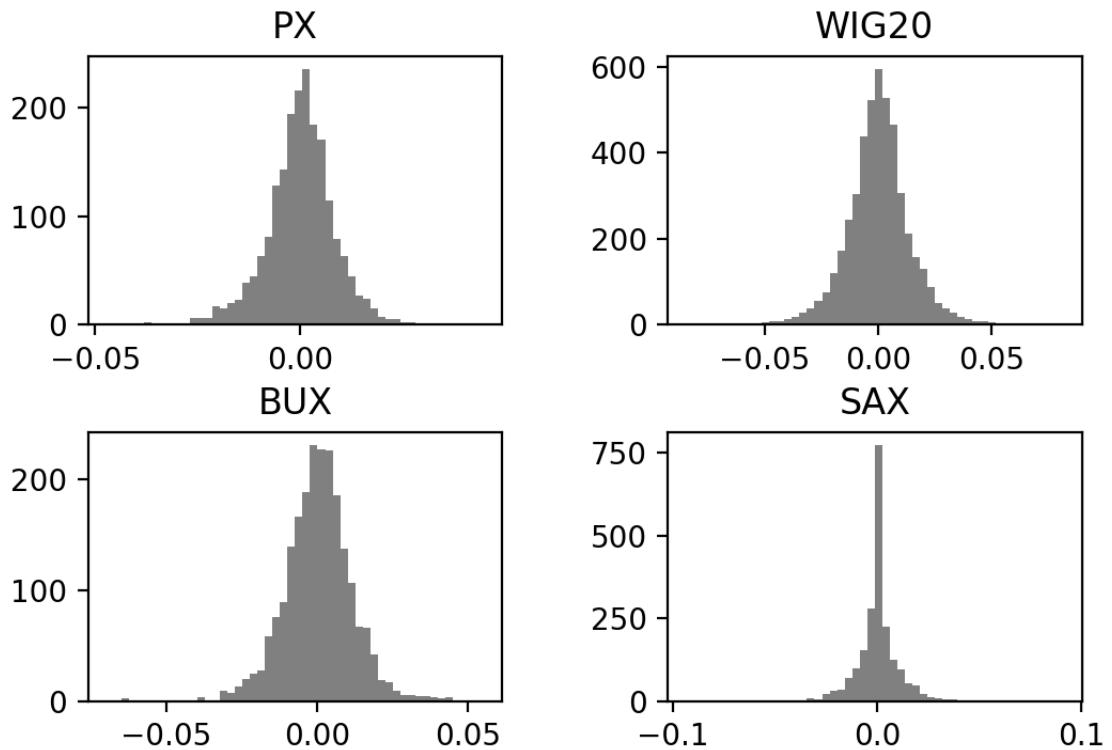
$$r_t = \ln(P_t/P_{t-1}) \quad (4.1)$$

where P_t is the index value on the day t , are summarized below.

Table 4.1: Descriptive statistics of daily index returns

	PX	WIG20	BUX	SAX
sample size	1995	4718	2192	2089
mean	0.000119	0.000073	0.000312	0.000202
maximum	0.0447	0.0815	0.0551	0.0912
minimum	-0.0471	-0.0844	-0.0698	-0.0933
std	0.0085	0.0142	0.0115	0.0104
skewness	-0.347	-0.153	-0.232	-0.076
kurtosis	2.510	2.814	2.943	9.026

Figure 4.1: Histogram of daily index returns



4.2 Preprocessing

To build our model LSTM architecture with 2 hidden layers is used. 90% of data is utilized for training and the other 10% of data left for testing. As inputs we use sequences of 20 consecutive log returns. As outputs we use realized Value at Risk computed with 20-day realized historical volatility of returns and 20-day realized average return. Student's t distribution of financial returns is assumed. Therefore, realized Value at Risk is defined as follows

$$\widehat{VaR}_t^\alpha = \hat{\mu}_{20d.r.,t} + z_{st,t}^\alpha \hat{\sigma}_{20.r.,t}, \text{ where} \quad (4.2)$$

$\hat{\mu}_{20d.r.,t}$ is 20-day realized average return,

$\hat{\sigma}_{20.r.,t}$ is 20-day realized historical variance,

$z_{st,t}^\alpha$ is α -percentile of Student's t distribution.

For training we use mean squared error to optimize our model.

Chapter 5

Results and conclusion

5.1 Empirical results

index	α	stats	LSTM			FIAPARCH			EVT-POT		
			UC*	ind**	CC***	UC*	ind**	CC***	UC*	ind**	CC***
WIG20	0.01	<i>LR</i> <i>p – value</i>	4.271 0.0275 (acc.)	4.275							
	0.05	10.078	10.517								
	0.1	5.647	6.566								
PX	0.01	0.559	0.569								
	0.05	0.332	1.021								
	0.1	0.699	3.580								
BUX	0.01	0.754	0.763								
	0.05	3.895	4.135								
	0.1	0.089	0.673								
SAX	0.01	4.363	4.768								
	0.05	10.142	10.182								
	0.1	15.065	15.430								

* Unconditional coverage according to Kupiec's POF test

** Independence test by Christoffersen

*** Conditional coverage according by Christoffersen

5.2 Conclusion

to be completed

Bibliography

- ABAD, P., S. BENITO, & C. LÓPEZ (2014): “A comprehensive review of value at risk methodologies.” *The Spanish Review of Financial Economics* **12**(1): pp. 15–32.
- ARAGONÉS, J. R., C. BLANCO, & K. DOWD (2000): “Extreme value var.” *Derivatives week* pp. 7–8.
- BALI, T. G. (2003): “An extreme value approach to estimating volatility and value at risk.” *The Journal of Business* **76**(1): pp. 83–108.
- BALKEMA, A. A. & L. DE HAAN (1974): “Residual life time at great age.” *The Annals of probability* pp. 792–804.
- BENGIO, Y. (2009): *Learning deep architectures for AI*. Now Publishers Inc.
- CARRIO, A., C. SAMPEDRO, A. RODRIGUEZ-RAMOS, & P. CAMPOY (2017): “A review of deep learning methods and applications for unmanned aerial vehicles.” *Journal of Sensors* **2017**.
- CHEN, T.-B. & V.-W. SOO (1996): “A comparative study of recurrent neural network architectures on learning temporal sequences.” In “Proceedings of International Conference on Neural Networks (ICNN’96),” volume 4, pp. 1945–1950. IEEE.
- CHRISTOFFERSEN, P. F. (1998): “Evaluating interval forecasts.” *International Economic Review* **39**(4): pp. 841–862.
- DAVISON, A. C. & R. L. SMITH (1990): “Models for exceedances over high thresholds.” *Journal of the Royal Statistical Society: Series B (Methodological)* **52**(3): pp. 393–425.
- DIEBOLD, F. X., T. SCHUERMANN, & J. D. STROUGHAIR (1998): “Pitfalls and opportunities in the use of extreme value theory in risk management.” In “Decision technologies for computational finance,” pp. 3–12. Springer.

- FISCHER, T. & C. KRAUSS (2018): “Deep learning with long short-term memory networks for financial market predictions.” *European Journal of Operational Research* **270**(2): pp. 654–669.
- FISHER, R. A. & L. H. C. TIPPETT (1928): “Limiting forms of the frequency distribution of the largest or smallest member of a sample.” In “Mathematical Proceedings of the Cambridge Philosophical Society,” volume 24, pp. 180–190. Cambridge University Press.
- FRANCES, F., J. D. SALAS, & D. C. BOES (1994): “Flood frequency analysis with systematic and historical or paleoflood data based on the two-parameter general extreme value models.” *Water resources research* **30**(6): pp. 1653–1664.
- GALLAGHER, J. C., S. K. BODDHU, & S. VIGRAHAM (2005): “A reconfigurable continuous time recurrent neural network for evolvable hardware applications.” In “2005 IEEE Congress on Evolutionary Computation,” volume 3, pp. 2461–2468. IEEE.
- GENCAY, R. & F. SELCUK (2004): “Extreme value theory and value-at-risk: Relative performance in emerging markets.” *International Journal of forecasting* **20**(2): pp. 287–303.
- GILLI, M. *et al.* (2006): “An application of extreme value theory for measuring financial risk.” *Computational Economics* **27**(2-3): pp. 207–228.
- HE, T. & J. DROPOPO (2016): “Exploiting lstm structure in deep neural networks for speech recognition.” In “2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),” pp. 5445–5449. IEEE.
- HOCHREITER, S. & J. SCHMIDHUBER (1997): “Long short-term memory.” *Neural computation* **9**(8): pp. 1735–1780.
- HSU, W.-N., Y. ZHANG, A. LEE, & J. GLASS (2016): “Exploiting depth and highway connections in convolutional recurrent deep neural networks for speech recognition.” *cell* **50**: p. 1.
- HWANG, J. (2020): “Modeling financial time series using lstm with trainable initial hidden states.” *arXiv preprint arXiv:2007.06848* .

- JORDAN, M. (????): “Attractor dynamics and parallelism in a connectionist sequential machine. 1986.” In “Proceedings of Eighth Annual Conference of Cognitive Science Society Hillsdale, NJ, Erlbaum,” pp. 531–546.
- KHAN, S. & T. YAIRI (2018): “A review on the application of deep learning in system health management.” *Mechanical Systems and Signal Processing* **107**: pp. 241–265.
- KUPIEC, P. (1995): “Techniques for verifying the accuracy of risk measurement models.” *The J. of Derivatives* **3(2)**.
- NWANKPA, C., W. IJOMAH, A. GACHAGAN, & S. MARSHALL (2018): “Activation functions: Comparison of trends in practice and research for deep learning.” .
- PALANGI, H., L. DENG, Y. SHEN, J. GAO, X. HE, J. CHEN, X. SONG, & R. WARD (2016): “Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval.” *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **24(4)**: pp. 694–707.
- PARRONDO, J. M. (1997): “Calculation of the value at risk in emerging markets.” .
- PEARLMUTTER, B. A. (1989): “Learning state space trajectories in recurrent neural networks.” *Neural Computation* **1(2)**: pp. 263–269.
- REGHIN, D. & F. LOPES (2019): “Value-at-risk prediction for the brazilian stock market: A comparative study between parametric method, feedforward and lstm neural network.” In “2019 XLV Latin American Computing Conference (CLEI),” pp. 1–11. IEEE.
- ROBINSON, A. & F. FALLSIDE (1987): *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering Cambridge, MA.
- ROSSI, F., M. FIORENTINO, & P. VERSACE (1984): “Two-component extreme value distribution for flood frequency analysis.” *Water Resources Research* **20(7)**: pp. 847–856.

- SATHYA, R. & A. ABRAHAM (2013): “Comparison of supervised and unsupervised learning algorithms for pattern classification.” *International Journal of Advanced Research in Artificial Intelligence* **2**: p. 36.
- SIAMI-NAMINI, S. & A. S. NAMIN (2018): “Forecasting economics and financial time series: Arima vs. lstm.” *arXiv preprint arXiv:1803.06386* .
- SOLTAU, H., H. LIAO, & H. SAK (2016): “Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition.” *arXiv preprint arXiv:1610.09975* .
- ŠTER, B. (2013): “Selective recurrent neural network.” *Neural processing letters* **38(1)**: pp. 1–15.
- WERBOS, P. J. (1988): “Generalization of backpropagation with application to a recurrent gas market model.” *Neural networks* **1(4)**: pp. 339–356.
- ZEYER, A., P. DOETSCH, P. VOIGTLAENDER, R. SCHLÜTER, & H. NEY (2017): “A comprehensive study of deep bidirectional lstm rnns for acoustic modeling in speech recognition.” In “2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),” pp. 2462–2466. IEEE.

Appendix A

Title of Appendix A

Appendix B

Title of Appendix B