Benchmarking Lightweight User Mobility Predictors on Operational WLAN Data

Miriam Leopoldseder, Philipp Svoboda, Lukas Eller, Markus Rupp
Institute of Telecommunications
Technische Universität Wien
Vienna, Austria
firstname.lastname@tuwien.ac.at

***Abstract*—In 5G cellular networks, a lightweight, online, real-time, and precise user flow prediction can improve the scheduling of adaptive radio and core network resources. We compare six different methods for short-to-medium-time prediction that fulfill these requirements. In our scenario, we predict the flow of users in a small cell indoor network. We then compare the performance of each method to predict the number of users per access point for several discrete time intervals without any intermediate information. To benchmark the performance of each method in live networks, we collected data from an operational WLAN network at the university over a period of one semester. The results show that common Markov-based solutions perform well only on very short prediction horizons. Beyond that, they are even outperformed by naive predictors. Solely the machine learning approach based on a neural network outperforms all other methods for any prediction horizon. This method enables a real-time and precise user flow prediction at the edge nodes of the network, as the complex task of training can be performed centralized and offline.**

***Index Terms*—machine learning, Markov, Kalman, naive predictor, neural networks, WLAN, Wi-Fi, 5G, user flow prediction, benchmark**

I. Introduction

Future 5G cellular networks have to satisfy challenging demands such as high data rate, low latency, energy constraints, and the diverse needs of a growing number of users. Information-driven networking protocols, on top of software-defined networks that optimize the resources on demand, can help to address these challenges. The online optimization of resources is tightly connected to understanding the current number of active users in a mobile cell. A real-time, and precise short-time user flow prediction enables networks to have well-orchestrated scheduling of radio and core resources. However, as the data ages quickly, the solution is required to run on nodes at the very edge of the network.

Based on real-world data collected from a small cell environment, we investigate six different location prediction methods, benchmarking individual estimators adopted from the existing literature and access point (AP) aggregated methods. We consider the small cell scenario to be especially relevant, as it is one of the solutions that have already been studied to solve the increasing bandwidth demands in buildings.

Opposed to existing work, we also include naive predictors as a performance baseline.

II. Location prediction in Mobile Communication

Matching the current demand with the available resources is a fundamental challenge in a fast-paced modern society. Predicting traffic flows allows to match the demand for mobility with the required resources. It is therefore not surprising that early researchers have already investigated the methods of predicting where people will be and how they will use the network infrastructure at a specific location.

A. System Model

Historically, a common goal in mobile communication is to predict the hand-over of user connections between cells. Subsequently, researchers became more focused on predicting the location of a single user than on tracking the movement of a group of users. To be able to predict the next user location, many algorithms define the movement history of a single user, $\mathcal{H}_m = \langle x_1 = a_i, \ldots, x_m = a_j \rangle$, where $i, j \in \{1, \ldots, N\}$, as a base for the prediction. Here, x represents an abstract location (e.g., a cell) or a precise location (e.g., GPS coordinates) at time or event i. We assume that the locations x can be modeled as random variables from a finite alphabet $\mathcal{A} = \{a_1, \ldots, a_N\}$, which describes all possible locations. The update of $\mathcal{H}_m$ can be movement-based, with an entry added event-driven (e.g., updating the user history at each cell crossing). Alternatively, the updates can happen at fixed time intervals. Combining both approaches is possible, if $\mathcal{H}$ is updated after some time units and at every cell crossing [1]. $\mathcal{H}_m$ is also called user sequence, trajectory, or trace.

We define prediction horizon p as the number of prediction steps getting predicted. The length of each prediction step is defined by the data and, in our case, is equivalent to 30 seconds. Depending on the prediction horizon, we only get new measurements of the current state of the network each pth step, as illustrated in Fig. 1. All intermediate time-steps are predicted using the selected prediction method.

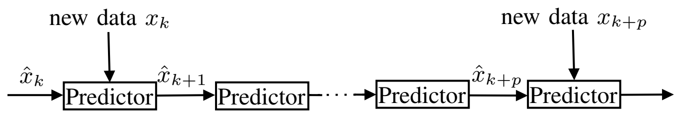

Fig. 1. Prediction process with data updated every pth step.

B. State of the art

There are numerous examples of location prediction methods in the literature. A common basis for the predictors is to assume a Markov condition. For example, [2] evaluated mobility prediction based on different enhanced Markov model schemes. In [3], the authors introduced an improved Markov-based prediction and compared it among others to a time-based Markov approach. They comment that the prediction is more accurate for sequences with less locations and less changes.

In [4], the authors compared different domain-independent prediction methods in terms of their prediction performance for WLAN data. They concluded that entropy, as an indicator of movement randomness, correlates with the performance of Markov-based predictors. Surprisingly, the lower-order Markov predictors showed the best overall performance when compared to more complex compression-based methods. The authors further found that methods relying on online learning need a long trace length until they deliver reliable results.

Hidden Markov models (HMMs) offer a more complex underlying model, which are used for a wide array of tasks such as speech recognition [5] and gene finding [6]. The authors in [7] employed HMMs to predict movement in buildings. In [8], a similar model is applied to a cellular mobile network. They then compared the prediction performance of an order-1 Markov chain, order-2 Markov chain, and an HMM.

III. Methods

Our objective is to benchmark several prediction methods for varying prediction horizons and to evaluate their use in future network generations. To quantize the prediction performance, we chose the mean square error (MSE) between the correct number of users per AP and the predicted number of users per AP for one week. We compare six different methods, two of which are based on predicting the location for each individual users, two directly predict the number of users per AP, and finally two naive predictors.

1) **Order-L Markov:** This method is introduced in [4] for handover prediction of individual users. It offers a low complexity prediction based on a maximum likelihood estimator of the next location.
2) **HMM:** This approach is widely used and is an extension of the Markov model before [9]. The model includes non-observable hidden states that help to characterize hidden relationships between the observed data [10].
3) **Kalman filter:** This is based on a linear state space model, in which each state represents one AP. This method directly predicts the number of users per AP [11].
4) **Neural network:** As a representative of a general machine learning method, we use neural networks to predict the number of users at each AP [12].
5) **Use last state (ULS):** ULS is a naive predictor that uses the last-known state for all prediction steps of each individual user.
6) **ULS AP:** This AP-based naive method keeps the number of users per AP constant over the prediction horizon.

A. Order-L Markov

This method utilizes a Markov assumption for an individual user based next step prediction [4].

At each point in time, $x \in \mathcal{A} = (a_1, \dots, a_N)$ represents the user location. We assume that we know the location history $\mathcal{H}_m = \langle x_1 = a_i, \dots x_m = a_j \rangle$, where $i, j \in \{1, \dots, N\}$, of each user up to the point of the estimation. Assuming stationarity, the maximum probability of the next step x_{m+1}, considering the history $\mathcal{H}_m$, gives us the maximum likelihood estimate for the prediction,

$$\hat{x}_{m+1} = \arg\max_{a \in \mathcal{A}} \frac{\mathcal{N}(a|c_m; \mathcal{H}_m)}{\mathcal{N}(c_m; \mathcal{H}_m)}. \quad (1)$$

B. Hidden Markov Model (HMM)

The HMM consists of hidden states $h_t \in \{1, \dots, H\}$ and the observed variables $v_t \in \{1, \dots, V\}$, which are connected according to an observation model $p(v_t|h_t)$ [9]. This leads to the joint distribution of hidden and visible states,

$$p(h_{1:T}, v_{1:T}) = p(v_1|h_1)p(h_1) \prod_{t=2}^{T} p(v_t|h_t)p(h_t|h_{t-1}). \quad (2)$$

Assuming that the HMM is stationary, the transition probability can be represented by a $H \times H$ transition matrix $\mathbf{A}$, with $\mathbf{A}(i,j) = A_{ij} = p(h_{t+1} = i|h_t = j)$. The emission probability is represented as $V \times H$ emission matrix $\mathbf{B}$, with $\mathbf{B}(i,j) = B_{ij} = p(v_t = i|h_t = j)$ [9], [10].

For the prediction, we maximize the posterior probability over all possible states v_t,

$$\begin{aligned} v_t^* &= \arg\max_{v_t} p(v_t|v_{1:t-1}) \\ &= \arg\max_{v_t} \left\{ \sum_{h_t} p(v_t|h_t)p(h_t|v_{1:t-1}) \right\}. \end{aligned} \quad (3)$$

Training the Parameters of HMMs: We apply the Baum-Welch algorithm to the training data to estimate the parameters of the HMM [10]. The method itself is often numerically unstable and therefore some measures need to be taken to deal with the very small numbers. We chose to apply the logarithm to each parameter as suggested by several authors such as [9], [10].

C. Kalman

Assuming that the system we want to characterize can be described by a linear state space model,

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{G}_k\mathbf{w}_k \qquad \mathbf{x}(0) = \mathbf{x}_0 \quad (4a)$$

$$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{D}_k\mathbf{u}_k + \mathbf{H}_k\mathbf{w}_k + \mathbf{v}_k \quad (4b)$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state representing the number of users at each AP, $\mathbf{u} \in \mathbb{R}^p$ is the number of users leaving or entering the system, $\mathbf{y} \in \mathbb{R}^q$ is the measured number of users per AP, $\mathbf{w} \in \mathbb{R}^r$ is the driving noise, and $\mathbf{v} \in \mathbb{R}^q$ is the measurement noise. Estimating the future state in the next time step is equal to minimizing the error between the estimated state $\hat{\mathbf{x}}_{k+1}$ and the real state $\mathbf{x}_{k+1}$. The Kalman filter is a recursive formulation of a sequential linear minimum mean square error estimation for the stated model [13]. It consists of three steps: calculating the Kalman gain matrix, updating the estimate, and then determining the new error covariance [11].

Prediction implementation: We know that the number of users per AP cannot be smaller than zero; thus, we add a step to ensure only those results that are greater than zero. Two methods are suggested in [14] to account for the inequality constraints when working with a Kalman filter. The first method is to project the unconstrained estimate $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}(k+1|k)$ into a space that fulfills the constraints,

$$\hat{\mathbf{x}}^{(p)}_{k+1} = \arg\min_{\mathbf{x}} \left\{ (\mathbf{x} - \hat{\mathbf{x}}_{k+1})^{\mathrm{T}} (\mathbf{x} - \hat{\mathbf{x}}_{k+1}) \right\} \tag{5}$$

$$\text{s.t. } \mathbf{Cx} \leq \mathbf{d}. \tag{6}$$

The second approach is to restrict the optimal Kalman gain to allow only for those solutions that fit the constraints.

In the implementation, we first calculate the standard Kalman estimate. Then, we modify the result by solving the minimization problem outlined in (7) and (8) in order to fulfill the constraint $\mathbf{x} \geq 0$ and optimize with Matlab's function `fmincon` [15].

D. Neural Network

Neural networks, when applied to regression tasks, are basically universal nonlinear function approximators that learn directly from data. Consequently, they are by design black box approaches, applicable to a wide variety of problems. This also means that the parameters learned by the network do not have a direct intuitive representation in the real world, therefore, they are not as insightful as model-based approaches.

In a neural network, the input $\mathbf{x}$ - output $\mathbf{y}$ relation can be seen as a special case of a generalized linear model with a nonlinear function applied to the output [12]:

$$y(\mathbf{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) \right) \tag{7}$$

where the weights w_j are the training parameters of the model, M is the number of inputs for a neuron, $f(\cdot)$ is a nonlinear activation function, and the basis functions are given by $\phi(\cdot)$. In the case of a feed-forward neural network, $\phi(\cdot)$ themselves are nonlinear functions of a linear combination of the inputs x.

A complete structure of a neural network is given in Fig. 2. It consists of multiple layers of connected neurons — the output of one layer always acts as the input of the following one. In regressions tasks, the output layer always uses a linear activation function $f(\cdot)$, whereas the nonlinearity is introduced by the activation functions in the hidden layers. A network that consists solely of linear $f(\cdot)$ would collapse into a simple linear regressor.

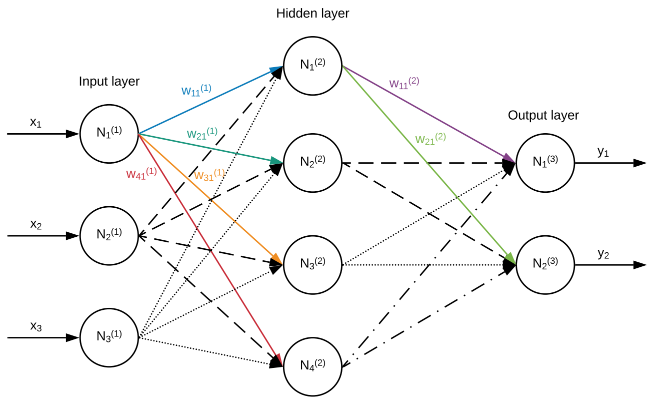

Fig. 2. Example topology of a neural network with one hidden layer.

Implementation: Working with state-of-the-art neural network framework [16], we resorted to Hyperas [17], a Keras-adapted wrapper around Hyperopt [18], to optimize the hyperparameters and overall network topology. In the optimization, our focus was on finding a sweet spot between model complexity and performance.

We used neural networks for AP-aggregated prediction for different prediction horizons. Therefore, the input vector $\mathbf{x}_k$ consists of the number of users assigned per AP at time k, whereas the output is given by a matrix of p stacked $\hat{\mathbf{x}}_{k+n}$ each for a different n. Each of those $\hat{\mathbf{x}}_{k+n}$ is estimated by a one-step neural network predictor, because for multi-step predictors, the model needs to be optimized repeatedly for different prediction-horizons as the number of neurons in the output layer is proportional to p. This would make the comparison between different horizons p unreliable.

Optimizing the one-step predictors results in a model with two densely connected hidden layers and 256 neurons each. We select *tanh* as activation functions for the hidden layers. The size of the output layer is equivalent to the number of APs. During training, we incorporate a small *dropout* in each hidden layer to prevent overfitting [19]. We choose the algorithm Adam, with the MSE as a loss function, for the optimization procedure [20].

E. Naive Methods

To get a good benchmark of the different methods, we wanted to compare the results to a simple unbiased technique. We therefore decided to include the results achieved with a ULS predictor, which uses the last-known state for all prediction steps for each individual user. Additionally, we evaluate results achieved by the ULS AP, for which we keep the value of the number of users constant for the length of the prediction horizon.

IV. Results

In this section, we focus on benchmarking the different methods while working with real-world data collected from an operational environment. A more detailed analysis, including artificial data sets and parameter analysis, can be found in [15].

A. Experimental Setup

We collected data from the WLAN network of TU Wien, specifically from one part of Campus Gußhaus. In total, we monitored four floors that encompass six lecture rooms. The location of the APs of the most-frequented floor can be seen in Fig. 3.

The data collection took place between April 2018 and February 2019. Over one semester, we gathered 4 883 603 data entries in total, including 11 weeks measured continuously every 30 seconds. Every entry includes a timestamp, anonymized client ID, AP MAC, AP name, and AP ID.

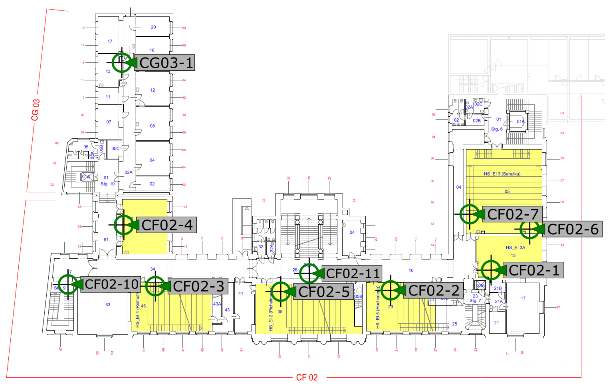

Fig. 3. Third floor of Campus Gußhaus, displaying APs in green and lecture rooms in yellow [21].

A clear day of time effect in the number of users over a day can be seen in the data. To illustrate, Fig. 4 plots 24 hours of measured data on Monday 22th of January 2019. Each color represents a different AP, and all APs are stacked on top of each other.

B. Order-L Markov Predictor

The order of the Markov predictor significantly affects the method's performance. As such, we first test several variations of L to determine which variation will deliver the best MSE

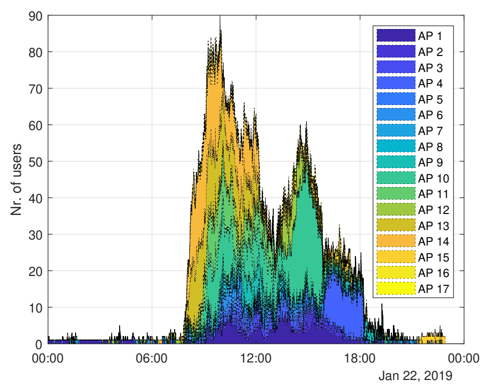

Fig. 4. Number of users over time per AP during Monday, 22nd of January 2019, measured every 30 seconds at Campus Gußhaus.

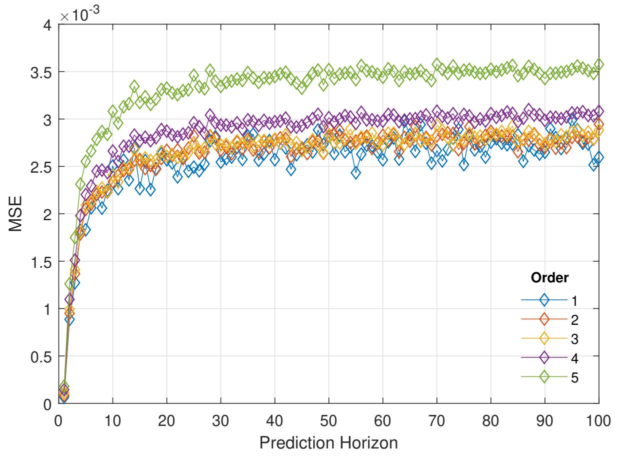

Fig. 5. MSE results for different order Markov predictors applied to the measured WLAN data.

results. Fig. 5 shows the results of the Markov predictor for the WLAN data with different context lengths. For one prediction step, the results of all orders are very close; in the higher prediction steps, the lower-order predictors outperform the higher-order methods. We then choose order-1 for the later evaluations because it delivers slightly better results for prediction horizons 2, 3, and 5 and because it is the least complex.

C. Benchmark of Different Prediction Methods

The goal of our approach is to identify possible candidates for network edge computing solutions for predicting user flow. Therefore, we consider two aspects in our evaluation, namely, prediction accuracy and error propagation due to noisy predictor parameters. In Fig. 6, the prediction MSE for the WLAN data is depicted for different prediction horizons.

The results of the short-term user prediction are in line with the literature, which often utilizes simple Markov models for prediction. However, as our prediction horizon extends beyond one minute, it is clearly outperformed by the methods that directly predict the number of users per AP, even by the naive ULS AP. The machine learning approach based on neural networks outperforms all other methods by a substantial margin. This indicates that even after optimizing the model parameters extensively, the underlying predictor models do not match the WLAN data well. A detailed overview of the prediction performance can be seen in Table I.

Additionally, we conducted a preliminary analysis using data generated from a theoretical HMM model. With the ground truth known, we were able to analyze the impact of noisy parameter estimations. Accordingly, we were able to understand the error propagation in Order-L Markov, HMM and Kalman predictors [15]. This could be a limitation for practical implementations in live networks, where input data is prone to measurement errors, potentially resulting in biased model parameters.

TABLE I

MSE RESULTS OF THE DIFFERENT PREDICTION METHODS APPLIED TO THE WLAN DATA FOR PREDICTION HORIZONS 1, 10, 20, AND 40. ALL VALUES ARE TO THE POWER OF −5.

Prediction horizon p	**1**	**10**	**20**	**40**
Order-1 Markov	6.68	243.52	244.91	271.31
HMM	6.67	51.16	197.98	747.32
Kalman	25.06	41.41	61.84	93.22
ML	1.24	4.81	9.14	20.08
ULS	6.37	436.68	574.14	807.68
ULS AP	5.86	20.33	34.05	55.52

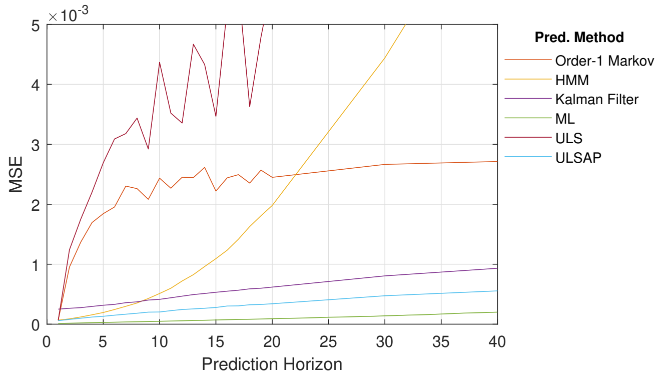

Fig. 6. MSE over prediction horizon of the different prediction methods.

V. Conclusion

In this paper, we present an in-depth benchmark of six different methods to forecast user-flow patterns inside a university building. All methods show promising results for single-step predictions. However, medium-term predictions, as required for resource scheduling of radio elements in 5G, fail with state of the art solutions. We conducted the benchmark using real-world data collected from a university campus. In fact, for a prediction horizon of several minutes naive approaches even outperforms state-of-the-art methods. The only method capable of coming out on top is a neural-network-based predictor.

References

[1] C. Cheng, R. Jain, and E. van der Berg, "Location Prediction Algorithms for Mobile Wireless Systems," in *Handbook of Wireless Internet*, CRC Press, 2003, ch. 11, pp. 245–261.

[2] A. Hadachi, O. Batrashev, A. Lind, G. Singer, and E. Vainikko, "Cell phone subscribers mobility prediction using enhanced Markov Chain algorithm," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 1049–1054, 2014.

[3] Y. Cheng, Y. Qiao, and J. Yang, "An improved Markov method for prediction of user mobility," in *Proc. 2016 12th International Conference on Network and Service Management (CNSM)*, IEEE, pp. 394–399.

[4] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating Location Predictors with Extensive Wi-Fi Mobility Data," vol. 5, pp. 1633–1649, 2003.

[5] P. Kenny, M. Lennig, and P. Mermelstein, "A Linear Predictive HMM for Vector-Valued Observations with Applications to Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 2, pp. 220–225, Feb. 1990.

[6] S. L. Cawley and L. Pachter, "HMM sampling and applications to gene finding and alternative splicing," *Bioinformatics*, vol. 19, no. SUPPL. 2, 2003.

[7] A. Gellert and L. Vintan, "Person Movement Prediction Using Hidden Markov Models," *Studies in Informatics and Control*, vol. 15, no. 1, pp. 17–30, 2006.

[8] H. Si, Y. Wang, J. Yuan, and X. Shan, "Mobility Prediction in Cellular Network Using Hidden Markov Model," *2010 7th IEEE Consumer Communications and Networking Conference*, pp. 1–5, 2010.

[9] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. 1991, pp. 73–78, 216–244.

[10] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge: Cambridge University Press, 2011, p. 646.

[11] W. Kemmetmüller and A. Kugi, *Regelungssysteme 1*. 2018. [Online]. Available: https://www.acin.tuwien.ac.at/master/regelungssysteme-2/.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[13] F. Hlawatsch, "Parameter estimation Methods," no. March, pp. 12–15, 2010, ISSN: 00051098.

[14] N. Gupta and R. Hauser, "Kalman Filtering with Equality and Inequality State Constraints," no. 07, pp. 1–26, 2007.

[15] M. Leopoldseder, "Data driven prediction of crowd mobility in small cell environments," Technische Universität Wien, 2019, p. 90. [Online]. Available: http://katalog.ub.tuwien.ac.at/AC15373966.

[16] F. Chollet *et al.*, *Keras*, 2015. [Online]. Available: https://keras.io.

[17] *Hyperas*, https://github.com/maxpumperla/hyperas.

[18] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: A python library for model selection and hyperparameter optimization," *Computational Science & Discovery*, vol. 8, no. 1, 2015.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014.

[21] GUT, *GUT Grundrisspläne*, 2019. [Online]. Available: https://www.tuwien.at/fileadmin/Assets/dienstleister/gebaeude_und_technik/FS/Plaene/Gusshausstrasse_25-25a_1040_Wien_CF-CG.pdf.