# Exploring Deep Neural Networks for Regression Analysis

Florian Kästner, Benedikt Janßen, Frederik Kautz, Michael Hübner
Chair for Embedded Systems of Information Technology
Ruhr-University Bochum, Bochum, Germany
Email: {Florian.Kaestner, Benedikt.Janssen, Frederik.Kautz, Michael.Huebner}@rub.de

***Abstract*—Designing artificial neural networks is a challenging task due to the vast design space. In this paper, we present our exploration on different types of deep neural networks and different shapes for a regression analysis task. The network types range from simple multi-layer perceptron networks to more complex convolutional and residual neural networks. Within the exploration, we analyzed the behavior of the different network shapes, when processing measurement data characteristic for mass spectrometers. Mass spectrometers are used to determine single substances within gaseous mixtures. By applying deep neural networks for the measurement data processing, the behavior of the measurement system can be approximated indirectly through the learning process. In addition, we evaluate the usage of reinforcement learning to design the neural network's architecture.**

***Keywords–ANN; MLP; CNN; Reinforcement Learning.***

## I. INTRODUCTION

In traditional machine learning approaches, manually designed features have to be provided for the input data. Extracting reasonable features is crucial for a successful usage of those algorithms. Moreover, the process of feature extraction is time consuming and requires expert knowledge regarding the specific applications. In contrast, Artificial Neural Networks (ANNs) are capable of automatically extracting features from given input data, superseding manually designed features. Furthermore, the increasing depth of Deep Neural Networks (DNNs) allows extracting very complex and abstract features out of several different representations with respect to prior levels. All these features are then used to form a proper output.

However, the design of ANNs is a challenging task due to the degrees of freedom for their architecture, such as depth of the ANN, width and type of the layers, as well as data flow paths. In addition, the training method has an impact on the ANN's performance, and has several degrees of freedom itself, for instance the initial parameter values, the batch size, the learning rate, and optimization algorithm.

In this paper, we present our results of the exploration of end-to-end trained ANNs for the processing of mass spectra measured with a miniaturized mass spectrometer [1]. Mass spectra allow the analysis of gaseous mixtures to determine their constituents. Within the exploration, we used a gaseous mixture with the constituents listed in Table I. In order to exclude any unknown effects of real measurement systems, we created a Python module to generate noisy mass spectra of the given mixture, based on the constituents' characteristic mass-to-charge ratio peaks. The noise is evenly distributed, and explained in Figure 1 a), Figure 1 b) shows the resulting spectra. The generated mass spectra are normalized so that the sum of the constituents adds up to one. The noise within the generated mass spectra and the mass spectra's minimal and maximal, as well as mean values are depicted in Figure 1. In summary, our goal is to extract the constituents' concentration of the generated noisy mass spectra, without assuming any pre-conditional knowledge. This type of problem definition for machine learning is called multi-output-regression.

TABLE I. DATA SET CONCENTRATION RANGE OF CONSTITUENTS

| Constituent | Concentration range |
|---|---|
| $H_2O$ | 0.0 % - 3.0 % |
| $CO_2$ | 0.2 % - 5.6 % |
| $N_2$ | 65.0 % - 80.0 % |
| $O_2$ | 15.0 % - 21.0 % |

Within the scope of the exploration, we analyzed the structure and hyper parameters of different kinds of ANNs suitable for this purpose, implemented with TensorFlow [2] without manually extracting features. Due to the time-invariant application we focus on feedforward-ANNs starting with the traditional Multi-Layer Perceptrons (MLPs). MLPs consist of at least three *fully-connected layer*, in which every neuron is connected to every neuron in the previous layer. In order to use spatial information in the signal and lower the number of parameters, we included Convolutional Neural Networks (CNNs). CNNs apply filter kernels on the input data that compute the dot product, and thus combine spatial information [3]. In addition, CNNs apply pooling layers that reduce the data dimension by down-sampling.

For every layer we apply the Rectifier Linear Unit *ReLU*-activation function, due to its properties of smoothing the issue of vanishing or exploding gradients as shown by Glorot et al. [4]. Another important property of this activation function applied to ANNs is the *sparse activation*, meaning that a certain number of neurons within a network will never fire. Although, this feature is desirable when designing ANNs, our exploration results of the network size could be influenced, due to the varying number of dead neurons. However, in order to not further increase the exploration complexity, we assume that this property does not influence the exploration, if the weight initialization is uniformly distributed.

To speed-up training, smooth the issue of exploding gradients, and to avoid over-fitting we use batch normalization with trainable scale and shift parameters. Within the scope of this work, we relinquish further methods avoiding over-fitting like dropout.

Although, applying batch normalization and ReLU-activation function relieves the issue of vanishing or exploding gradients, the problem still exists and becomes more crucial when the network is deeper. Therefore, we extend our exploration with *Residual neural Networks* (ResNets), and *highway networks*. The distinctive property of ResNets are *shortcuts*. Shortcuts implement the possibility to route the data flow around layers, and afterwards recombine the processed and

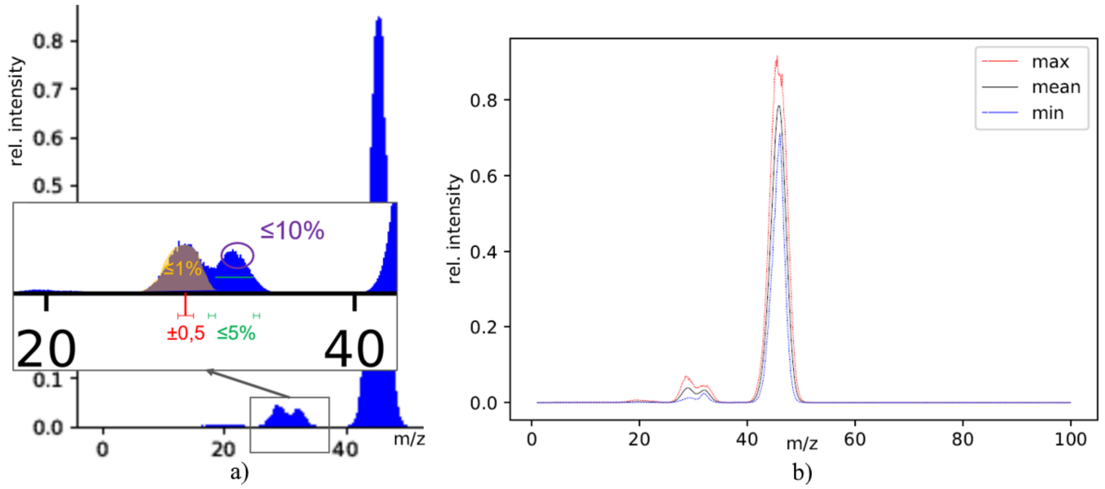

Figure 1. Mass spectra: area of each constituent peak varies by $\leq 1\,\%$, position by $\pm 0.5\frac{m}{z}$, variance by $\leq 5\,\%$, and each measurement value by $\leq 10\,\%$.

bypassed data [5]. In particular, the bypassing is done by an identity shortcut connection. Thus, the original residual block simply adds the outcome of the convolution layers with the original input. If the dimensions do not match after applying convolution due to stride, padding and amount of feature map parametrization, the input can be compressed, preferably with not trainable methods, or extended through padding. Shortcut methodologies are subject to current research. Due to the shortcut connections, the influence of vanishing gradients is far lower than with traditional ANNs. Moreover, the back-propagation can be applied much more efficient and simpler. Highway networks follow a similar approach, however, they employ a gating function optimized within the training phase to filter the data through the shortcut [6].

The related work within this field of research is presented in Section II. The method of our exploration is twofold. We further used Reinforcement Learning (RL) to automate the exploration for the MLP networks. The results of the RL based exploration can be found in Section IV. A discussion and summary of the current results, as well as an outlook to future work can be found in Section V.

## II. Related Work

An early work within the direction of our approach was published by Massicotte et al. [7], who investigated into the calibration of high-pressure measurement systems with MLP networks. The authors compared an ANN-based method to a spline-based method, and state that the ANN-based method achieves better results with lower quality reference data, and thus, enables a reduction of the time necessary for calibration data acquisition. Moreover, the spline-based method requires the parallel measurement of the temperature to consider temperature effects, which is not the case for the Ann-based method.

Several different classification methods for analyzing mass spectra data were analyzed by the authors of [8]. Those classification methods include linear discriminant analysis, quadratic and discriminant analysis, k-nearest neighbor classifier, classification trees, Support Vector Machines (SVMs), and random forest. Wu et al. found that Random Forrest outperforms the other classification methods in overall misclassification as well as in stable assessment of classification errors.

The authors of [9] used an unsupervised method for extracting features from mass spectra data followed by classification realized with SVMs. Their methodology results in greater 95 % correctly classified samples.

The work described in [10] uses a RL method (Q-learning) based algorithm capable of generating high performance standard CNN. Created CNNs are outperforming existent networks with similar layer types and are competitive with state of the art networks making use of more complex layer types.

Referring to the screening methodology in the medical and genetic fields, the authors of [11] developed an approach which randomly generates a high number of networks with different parameter initialization and architectures. Configurations that show good results are used for further training. Regarding the steady growth of computational power Pinto et al. [11] claim that this approach can speed up development success and understanding of biological vision.

With the use of Cartesian Genetic Programming (CGP), Suganuma et al. [12] automatically construct a CNN for an image classification task with CIFAR-10 data set. Within the process the CNN structure is represented by CGP encoding method and is further optimized to reach best possible results. With this approach the authors claim to automatically find network architectures which are comparable with common state of the art CNNs.

## III. Empirical Exploration

Within this Section we demonstrate our approach to explore the shape of four different types of ANNs. The goal is to investigate, which structure suits best for the given qualitative analysis. This task represents a complex empirical optimization due to the high amount of adjustable hyperparameters, including among others batch-size, optimizer, and learning rate. Thus, the empirical approach can be seen as a starting point for further explorations preferably using optimization method such as RL described in Section IV.

The input vector consists of 990 floating point values representing the mass spectra from 0 to 100$\frac{m}{z}$. The last layer of every ANN within this work is built as a fully-connected layer with four neurons representing the constituents concentrations, where no activation function is applied. Due to the regression task we define the cost function as the Mean Squared Error (MSE) between the labeled constituents concentrations and the outcome of the last layer. The corresponding loss function is defined in Equation 1, with the true value $y$ and the prediction $\hat{y}$ for each of the four constituent.

$$MSE = \frac{1}{4}\sum_{i=0}^{4}(y_i - \hat{y_i})^2 \tag{1}$$

We apply batch normalization with trainable scale and shift parameters for all networks at certain points in hierarchical structure. For training we use a fixed batch size consisting of 25 randomly picked samples and Adaptive Moment Estimation (Adam) as the optimization method. The Adam optimizer was chosen due to the adaptive learning rate for every parameter and its good results dealing with sparse gradients. For further information, we refer to [13]. Those sparse gradients can result from the properties of ReLU activation function. As a weight initialization we choose the *Xavier method* [14]. To avoid overfitting, we apply early stopping. The break condition is an increasing deviation of the prediction from the label of the last three to the previous three predictions of the verification dataset, after the network has been trained with all samples in the dataset consisting of 100000 mass spectra. After the training, we verify the accuracy of the networks based on the deviation of the output of the last layer and the corresponding labels using a new dataset of 10000 mass spectra.

## A. Multi-Layer Perceptron Network

For the exploration of MLP networks, we created networks with different depths, starting from three up to 13 layers. The choice of the layer sizes is based on NumPy's *logspace()* function with base 10.0, generating a list of layers defined by the input layer size and the output layer size. The input layer size has been set within the range of $1.1\times$ to $0.05\times$ of the length of the input vector. We assume that the resulting funnel-shape of the networks is a suitable approximation to follow the feature extracting policy in order to raise the depth of the network with a sufficient number of neurons within the layers. This implies that we assume the data to be compressed and the function of the network is more dependable on the depth than on the width of the layers. We further save a significant number of parameters when downsizing the width of the fully-connected layers. The observed deviation on the test dataset is depicted in Figure 2. With an increasing depth of the network, the deviation tends to be larger. The best overall result for this exploration is achieved with a depth of four fully-connected layers with a mean of 1.5 %. The result of the exploration matches our expectations. The vanishing gradient problem prevent the network to perform better with increasing network depth. This represents a well-known problem in the deep learning domain. In Section III-C, we tackle this problem with the introduction of residual blocks allowing us to build deeper networks.

## B. Convolutional Neural Network

CNNs are a famous type of ANNs in the computer vision domain, especially in object detection, segmentation and tracking applications [15]. They are mainly responsible for

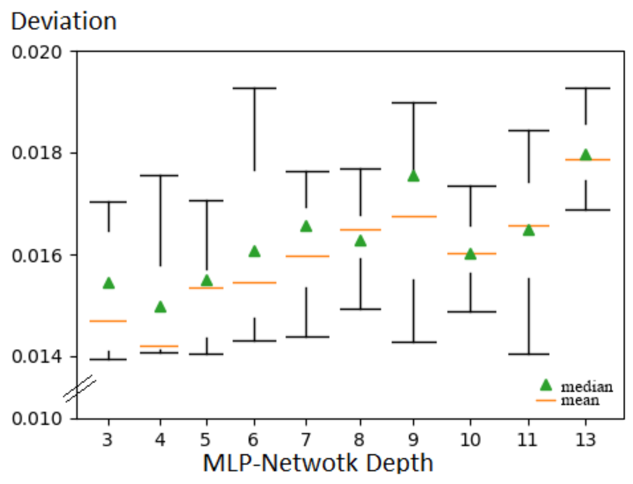

Figure 2. Deviations for different MLP configurations.

the today's popularity of ANNs due to their success in this field starting with ImageNet in 2011. One reason for their success is the property of the corresponding convolutional layer to combine spatial information applying 2-dimensional filter kernels to the input followed by pooling to reduce dimensions along the depth of the network. We also want to take advantage of these feature assuming the existence of spatial relationships.

Instead of reshaping the input and perform a 2-dimensional convolution we apply a 1-dimensional convolution with different kernel-sizes and filter-depths. The output of this convolution is a 2-dimensional feature map, which consist of corresponding height and depth. We further could apply a 2-dimensional convolution. However, we still assume that the spatial relationships only exist among the first dimension. Therefore, we decide to use 2-dimensional 1x1 convolution to reduce the shape back to a 1-dimensional outcome. To also reduce the shape among the first dimension we can apply the 1-dimensional convolution with a specific stride. The feature map shape of the 1 dimension is then given by the quotient of the 1 dimension of the input shape and the stride. These two convolutions represent the basic block of our CNN. The basic principle is visualized in Figure 3.

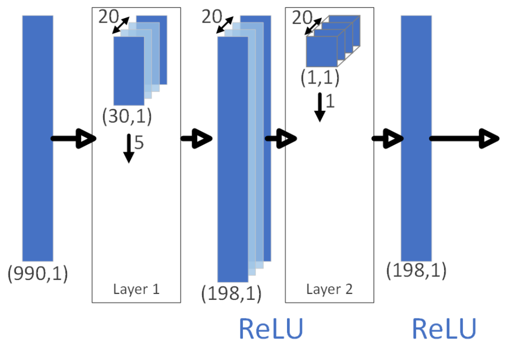

Figure 3. Visualization of the basic CNN principle.

We design the CNN by simply stacking those basic blocks and adjusting the hyperparameters, which are the stride, the number of feature maps, and the kernel size. The output of the last basic block is fed to a fully-connected layer with a fixed number of neurons of 99 followed by the output layer. The exploration is done with five CNNs ranging from two to six stacked basic blocks. We reduce the shape of the first dimension by using a stride of five in the first basic block. We further use a stride of two in the last basic block except for the first CNN, where just two basic blocks are involved. The stride parameter for all other basic blocks is set to one. The kernel size and number of kernels within a layer varies, starting with a wide kernel and a low number of kernels. Table II lists the chosen CNN configurations. While going deeper, we downscale the kernel size and increase the kernel depth. The result of this evaluation can be seen in Figure 4. Contrary to the MLP exploration, the accuracy does not drop rapidly as the network depth's increases, instead the deviation on the test dataset is approximately the same. Therefore, we follow that the influence of vanishing gradient is intensified in MLP networks. CNNs can be deeper, as the convolution requires less parameters, and owns an aggravated forward- and backpropagation path, due to the spatial connections, compared with the MLP network. The overall accuracy is slightly lower compared to those of the MLP exploration. To design deeper networks, we add block-wise shortcuts to the MLP and CNN as introduced in the next Section.

TABLE II. CNN CONFIGURATIONS EXPLORED

| number of stacked basic blocks | List of layer configurations with [(kernelsize,number of kernels)] in hierarchical order |
|---|---|
| 2 | [(15,30),(10,60)] |
| 3 | [(15,20),(10,30),(5,60)] |
| 4 | [(15,20),(10,30),(10,60),(5,300)] |
| 5 | [(30,20),(20,30),(20,80),(10,200),(5,400)] |
| 6 | [(30,20),(20,30),(10,60),(5,180),(1,300),(3,500)] |

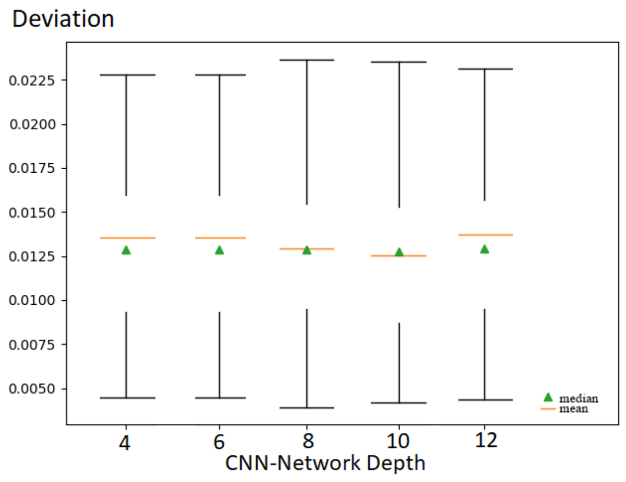

Figure 4. Deviations for different CNN configurations.

## C. Residual and Highway Network

Residual Networks or ResNets are the state-of-the-art networks for various applications and especially famous in image recognition tasks. He et al. [16] reformulated the layers to learn residual functions with respect to the input of the layer. This basic principle eases the learning due to the stepwise replacement of the product with the sum of the output of the layers in the backward path, and thereby reduces the problem of vanishing gradients rapidly. Thus, He et al. were able to design a ResNet with 152 stacked layers which outperformed all previous plain networks. We adopt this principle to our needs extending our basic convolution pair described in Section III-B with an identity mapping. The resulting principle residual block is shown in Figure 6. For simplicity reasons, we forego to reduce the shape of the first dimension. Thus, the dimensions of the input and the output of the block are the same. If this would be not the case the input has to be downscaled or upscaled, preferable without or a low number of trainable parameters. To downscale the first dimension among the depth of the network at certain points, we apply the basic convolution pair without shortcuts with a specific stride after a significant number of residual blocks. This could also be replaced with a pooling layer in the future.

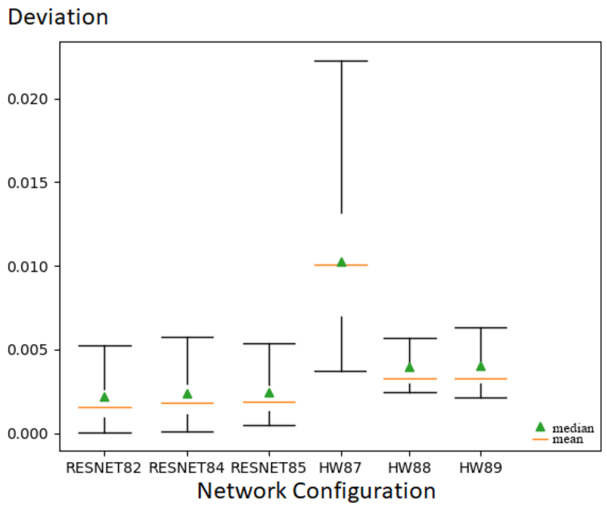

Figure 5. Deviations for different ResNet and Highway configurations.

A similar principle is used with Highway Networks. The main difference is the use of a gating function for identity mapping. However, this difference is only valid considering the original ResNet and the original Highway Network. Recent developments regarding different types of ResNet blurred these difference [17]. In the original highway network, developed by Srivastava et al. [18], the output $y$ of a basic highway block is defined as follows:

$$y = F(x, \boldsymbol{W_F}) \cdot T(x, \boldsymbol{W_T}) + x \cdot (1 - T(x, \boldsymbol{W_T})) \tag{2}$$

Where $x$ represents the input of the layer and $F$ represents the nonlinear activation function with the weight parameters $\boldsymbol{W_F}$. $T$ is defined as the transform gate, while the term $1 - T$ is depicted as the carry gate. This gating units can be seen as a learnable dataflow control unit through the network. We use this principle to continue our exploration. Therefore, we extend fully-connected layers with this gating unit. Similar to the ResNet approach we keep the dimensions equal inside the block units. We further apply fully-connected layers without gating units after a significant amount of highway blocks in order to reduce the shape of the first dimension.

Figure 5 shows the deviation result of 3 different configurations based on our residual and highway basic blocks. The residual blocks consist of a 1-dimensional convolution with kernel size 15, and 5 or 10 kernels, as well as a 2-dimensional $1\times1$ convolution. For the ResNet exploration we used 120, 80, and 40 layers. For the exploration of the highway network, we analyzed configurations with $24\times500$ and $24\times20$, $35\times100$ and $35\times50$, as well as $30\times100$ and $30\times50$ highway layers.

*ResNet82* performs best on the given test dataset with an mean deviation of 0.24 %. This network represents also the deepest one in our exploration containing 120 layers. However, the deviation difference to the other *ResNet* configurations are negligible. For instance, *ResNet85* consists of only 40 layers with mean deviation of 0.238 % on the test dataset. The highway network *HW87* is built with 48 highway blocks and owns a very low overall accuracy. In comparison, *HW88*, consisting 70 blocks, and *HW89*, consisting 60 blocks, perform better on the test dataset. This corresponds to our expectations as the results of the exploration of Srivastava et al. [18] also show a similar tends, where the accuracy is not sufficient for smaller highway Networks and increases with increasing depth.

Figure 8 shows the validation error during training of the best performing ANN of every type of ANN we explored over 500000 epochs. As can be seen, the ResNet82 converges very fast and without high fluctuations till break conditions. The MLP network with four layers is comparable to the other ANN types. The CNN with 10 convolution pairs started to converge faster but did not improve the prediction after 20000 epochs. HW88 shows a very fluctuating convergence, especially at the early training epochs. While parametrization of the gating units starts to work properly, the deviation with the validation dataset drops.

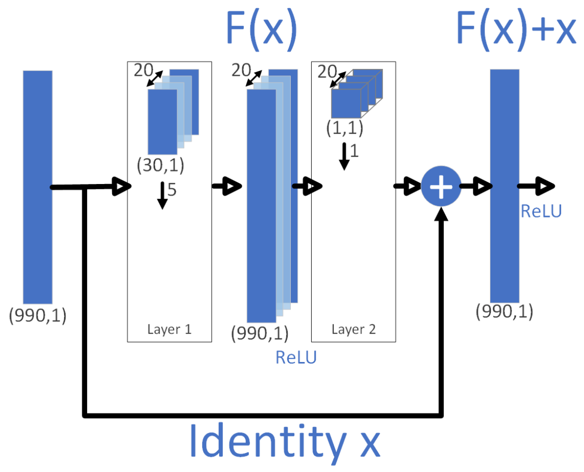

Figure 6. Visualization of the residual block principle.

## IV. Reinforcement Learning

As shown in Section III, the exploration of ANNs is a very complex optimization problem. Therefore, we start to explore the shape of ANNs with RL. The first results of this heuristic exploration are described in this Section. In general, a RL problem is based on the Markov decision process, in which an agent is taking actions, measuring the reaction of the environment, updating the policy and taking a new action based on the previous information. For further investigations we refer to [19]. For our purpose, we use a Q-learning methodology, which is a model-free, off-policy, temporal difference RL method. We focused the RL-based exploration on the MLP networks shape, in order to receive a MLP configuration with a sufficient accuracy. More precisely, we train an agent to the design an MLP network for our application. Thus, we first have to define a discrete state-action space. Starting from one hidden layer after the fixed size of the input layer, the agent can choose at every step either to add 10 neurons to the current layer or to extend the network with another hidden layer starting with 10 neurons reaching a new state. At each step the network is trained with the same hyperparameters outlined in Section III. The maximum width of every hidden layer consists of 400 neurons, while the maximum depth is set to 10 layers. If the agent is reaching a depth of 10 layers, or a deviation on the validation dataset below 0.1 %, the episode is stopped. The reward is defined as the reciprocal deviation on the validation dataset.

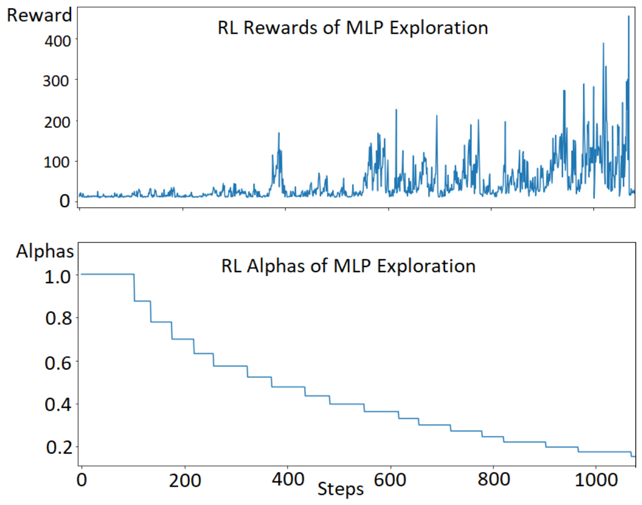

Figure 7. Reward and Alpha of Reinforcement Learning approach

We choose an $\epsilon$-greedy policy, where a random action is chosen with the probability of $\alpha$, otherwise the currently optimal action is chosen, which owns the highest value in the Q-Table. After every step, the Q-Table is updated with the learning rate $\beta$, following the temporal difference mechanism. $\alpha$ and $\beta$ are decreasing over time in a logarithmic manner, after each episode. We verify the functionality of this method by using 1000 steps, including 70 episodes, to train the agent. Figure 7 shows the progress of the obtained reward over time as well as the logarithmic decrease of $\alpha$. As can be seen, the reward increases and reaches a maximum the last episodes, which corresponds to a deviation of 0.22 % on the verification dataset. However, most interesting is the evolution of the Q-values. As we expected, for the first layers the agent is more likely to add neurons to the current layer till a size of 100 neurons. However, the agent does not want to drop the width of the sizes rapidly as we have expected. Instead the optimal policy of the agent extends the network in a much smoother way for the following layers. The best configuration on the verification dataset is: 90, 100, 70, 100, 100, 100, 60, 20, 50.

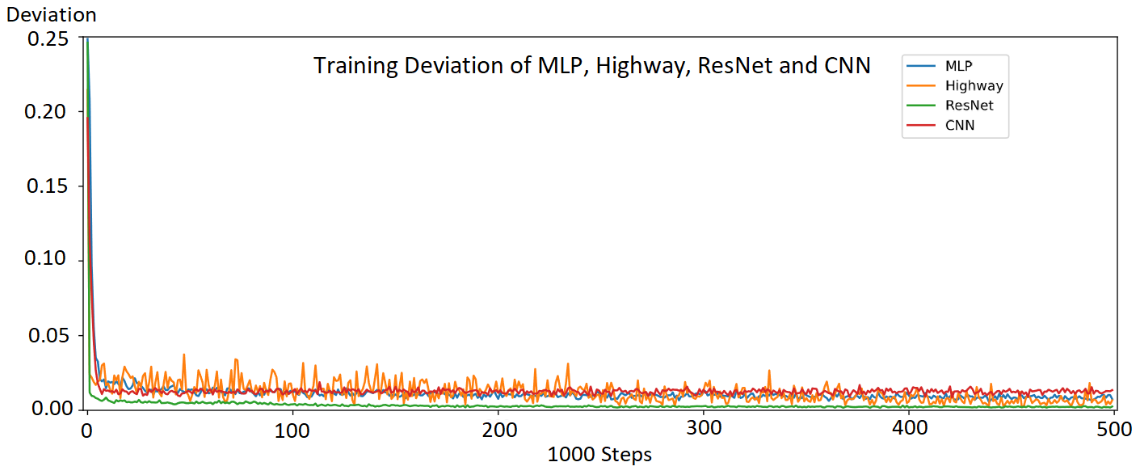

Figure 8. Validation error for best ANN of each class within training process.

## V. Conclusion and Outlook

In this paper, we presented an empirical exploration of different types of feedforward ANNs, namely MLP networks, CNNs, ResNets, and highway networks. Our goal is to find a suitable type and shape of ANNs to predict the concentration of four mixture constituents. The results represent the starting point for further explorations for this multi-target-regression task. Therefore, we explored the shape of a traditional MLP network first and investigated the effect of applying convolution layers to extract spatial relationships on the generated dataset. We further extend the exploration by increasing the depth of both network types with the help of shortcuts, namely residual blocks and highway blocks. The best result was achieved with a ResNet configuration containing 120 layers, which also represents the biggest network created in this research. Due to the high complexity of the exploration task, we simultaneously develop a Q-learning strategy to automatically explore the shape of a MLP network in order to find a configuration owning a sufficient accuracy. In future work, we want to combine the results from both approaches. Thus, the most promising approach is the automatic exploration with RL using residual blocks. We also want to extend our approach with recurrent units, to take aging effects of the mass spectra into account.

## Acknowledgment

This work was done under the support of BMWi project MiMEP (03ET1314A).

## References

[1] W. Kuipers et al., "Realization of a miniaturized mass spectrometer based on a microfluidic device," in Proceedings of the 1. International Conference on Microfluidic Handling Systems, 10 2012.

[2] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[3] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, Object Recognition with Gradient-Based Learning. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345.

[4] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, ser. Proceedings of Machine Learning Research, vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015.

[6] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," CoRR, vol. abs/1505.00387, 2015.

[7] D. Massicotte, S. Legendre, and A. Barwicz, "Neural-network-based method of calibration and measurand reconstruction for a high-pressure measuring system," IEEE Transactions on Instrumentation and Measurement, vol. 47, no. 2, Apr 1998, pp. 362–370.

[8] B. Wu et al., "Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data," Bioinformatics, vol. 19, 2003, pp. 1636–1643.

[9] M. Ceccarelli, A. d'Acierno, and A. Facchiano, "A machine learning approach to mass spectra classification with unsupervised feature selection," in Computational Intelligence Methods for Bioinformatics and Biostatistics, F. Masulli, R. Tagliaferri, and G. M. Verkhivker, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 242–252.

[10] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," CoRR, vol. abs/1611.02167, 2016.

[11] N. Pinto, D. Doukhan, J. J. DiCarlo, and D. D. Cox, "A high-throughput screening approach to discovering good forms of biologically inspired visual representation," PLOS Computational Biology, vol. 5, no. 11, 11 2009, pp. 1–12.

[12] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in Proceedings of the Genetic and Evolutionary Computation Conference, ser. GECCO '17. New York, NY, USA: ACM, 2017, pp. 497–504.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," CoRR, vol. abs/1412.6980, 2014.

[14] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), vol. 9, May 2010, pp. 249–256.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015.

[17] ——, "Identity mappings in deep residual networks," CoRR, vol. abs/1603.05027, 2016.

[18] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," CoRR, vol. abs/1505.00387, 2015.

[19] R. S. Sutton and A. G. Barto, Introduction to Reinforcement Learning, 1st ed. Cambridge, MA, USA: MIT Press, 1998.