An Architecture to Quantify the Risk of AI-Models

Alexander Melde[1], Astrid Laubenheimer[1], Norbert Link[2], and Christoph Schauer[2]

[1] Karlsruhe University of Applied Sciences
alexander.melde@h-ka.de astrid.laubenheimer@h-ka.de
[2] Inferics GmbH
norbert.links@inferics.com christoph.schauer@inferics.com

Abstract. In this paper we propose a multi-step approach to quantify the risk of AI-models. To evaluate the quality of a learned AI-model for image classification, a previously unseen part of a dataset is classified and the predictions are compared with their groundtruth to measure the accuracy of a model. In contrary, we first split the test dataset into two parts based on how unambiguous each sample can be assigned to a class. Samples that are close to the class decision boundary of multiple learned models are considered particularly difficult to classify. Second, we create a quantification of the model's ability to extrapolate on hard-to-classify or unseen data by training the model on "easy" data and evaluating it on the "difficult" split. Inside our models, we project the data into a 3-dimensional space using a neural network. We analyze this projection using the histogram of mutual distances, the silhouette measure [1] and the entropy of it to assess the extrapolation quality and thus robustness of the model. Subsequently, we apply our approach to the MNIST dataset [2] to prove its effectiveness. We see that models trained only on "easy" data are less robust than models trained on mixed data, which includes "difficult" data that lies in-between classes. This behavior is evident in both our quantitative measurements and qualitative evaluation In this paper, after an introduction to the topic and scope, related work is presented and the approach is explained in general terms. Subsequently, the application of the approach to the MNIST dataset is described and the results of these experiments are presented. Finally, a conclusion is drawn and options for future work are given.

Keywords: quality assurance, explainable AI, explainability, artificial intelligence, machine learning, explainable artificial intelligence, human activity recognition, action recognition, evaluation of AI systems, applications of AI in life sciences

1 Introduction

In recent years, there has been increasing research on artificial intelligence (AI) methods in order to make our everyday lives easier and safer. For instance, assisted living in combination with outpatient care services has become increasingly popular as an alternative to nursing homes. In addition to the established emergency call systems, more and more sensor-based AI systems are entering the market. These systems can inform nursing staff or trigger an alarm when they detect dangerous situations or unusual activities involving residents. Further increases in popularity are to be expected for these systems as new AI-based technologies support safety and security for self-determined living in familiar surroundings. In general, these technologies are based on machine learning (ML) models trained on datasets whose quality of being representative for real world scenarios is unknown.

For providers of such systems the introduction of new ML models into their products is of a high risk. The set of data on which the model can be evaluated in advance of the product launch commonly is not numerous enough and often generated under laboratory conditions which do not represent the true conditions for the product in real use. Furthermore, the performance of the models (from sight of economic efficiency) can not securely be predicted in advance because well introduced measures such as precision and recall [3, 415] do not lead to a reliable estimation of the risk of failures such as false alarms or missed detections and the associated financial cost.

In the context of systems that rely on activity recognition in domestic environments, this leads to problems, when closely related activities have to be distinguished. As one example "drinking a glass of water" and "brushing your teeth" are hard to distinguish, especially when the decision has to be made in the absence of a semantic context. In practice, classifiers usually are trained and tested on data that represents both activities in a clearly distinguishable way, which often leads to models with high accuracy.

In the wild however, one has to expect situations that are positioned fuzzy between the two classes and therefore hard to recognize. If such situations are not only underrepresented in the training data but also in the test data, they are not sufficiently considered in the quality assessment as well. Nevertheless, the behavior of the model on exactly these situations defines the risk of the use of the model in a real-life system.

In this paper, we propose a framework to evaluate how models which are trained on "easy" data only perform on "difficult" data which lies in-between classes. We demonstrate its effectiveness on the MNIST dataset [2].

2 Related Work

Approaches to assess model quality regardless of the distribution of the test data are provided by the field of Explainable Artificial Intelligence (XAI), which is a term to describe methods that explain decisions made by AI algorithms. Many deep neural network architectures are fundamentally black boxes whose decision-making is not comprehensible. Explainable AI attempts to make individual model executions or the entire model decision strategy more transparent. By understanding the decisions of a model, its quality can be assessed more independently from the particular test data and misconceptions arising from a limited scope of test data can be avoided. In recent years, numerous methods for XAI have been published. So far, these usually follow one of the following three approaches: First, the model can be built in a way that it is explainable naturally [4] (or by design, e.g. by using decision trees [5]), second original models can be replaced by fitted surrogate models that allow local or global interpretations [6, 7] and third, explanations can be generated using a direct process for either local or global explanations by putting a model into a more explainable state during training or by explaining single predictions, e.g. by determining the most important features for a certain decision [8, 9]. An extensive survey of XAI methods is given by Burkart and Huber in 2021 [9]. XAI tries to identify the risk of an AI model's decision being wrong and questions decisions made by the model. However, our approach does not explain the model itself, but instead attempts to quantify this risk. It describes the set of training data and how well it can be used to describe a real-world problem, and therefore does not fall into any of these categories.

Another related field of research is coverage testing, a technique determining whether the test cases used are actually covering the application area. Mani et. al. [10] explain the necessity to measure the quality of a dataset beyond the standard accuracy measure

(proposing a set of four metrics to measure the coverage quality of a test dataset in the feature space of a model) and propose and demonstrate the effectiveness of a systematic test case generation system (samples additional test cases from the feature space) [10, 1]. They propose four different test quality dimensions that (1) measure the distribution of test data across individual classes, (2) measure the percentage of test data for each class that lies close to the centroid of the trained cluster or (3) near the boundary with respect to every other class of trained class clusters and (4) measure for each pair of classes the percentage of the boundary-condition (3) [10, 2]. This proposed system is evaluated on the MNIST dataset as well. This approach is closely related to active learning, a term to describe methods that find areas of the feature space that are not sufficiently sampled and ask the user to add test data for this specific areas [11]. This might also be done by evaluating the density function in the feature space. In contrary to these approaches, we do not measure the quality of coverage in the feature space but the quality of data distribution in the latent space.

A related approach that can be applied in both the latent and feature space is outlier detection, which looks for data points outside the distribution of the dataset [12]. In contrast, we evaluate quality using only the data points that lie between classes.

3 Approach

Our proposed approach starts by splitting the test split of a dataset into two parts: one of them representing data samples that can clearly be assigned to one class ("easy data") and the other one representing data samples that cannot be assigned to a class unambiguously due to its proximity to the class decision border in the latent space of the network ("difficult data"). The split can be found by means of a majority voting approach across several proven model architectures.

In the second step, we quantify the ability of arbitrary models to extrapolate from "easy" to "difficult" data, by training the model on "easy" data and evaluating it on the "difficult" split, which then leads to a quantification of the models ability to extrapolate onto hard to classify or unseen data. For our approach we assume that the model consists of an embedder backbone network which projects the data into the latent space and a classifier head evaluating the projections. The extrapolation quality of the model is assessed by analyzing the projections in the latent space, where the histogram of mutual distances is analyzed. The silhouette measure [1] and the entropy are used to compare the model performance on the "easy" and "difficult" datasets as well as to measure the extrapolation power of the model.

The steps are visually summarized in Fig. 1 and described in detail below.

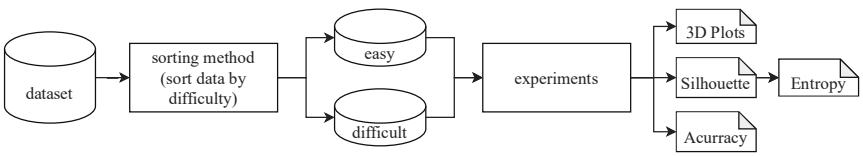


Fig. 1: Simple Overview of our architecture

To split the dataset by difficulty, we train seven different models using the same original training split but each time a different network architecture. We are using convolutional neural networks (CNNs) [13] of various complexity.

For each learned model, the test set is predicted and compared with the ground truth. We count the number of incorrect predictions per sample image of the test set to quantify the difficulty of each sample. The higher this number is, the more difficult this sample was to predict. We then group all of these samples into the two datasets which we consider "easy" and "difficult".

We train and test different combinations of dataset splits for test and training, loss functions, network architectures and number of epochs to measure the accuracy [3, 101] of different combinations to finally be able to find the "best working" combination in the traditional sense.

In order to measure robustness, we first create a visual representation of the model. For this we train multiple deep neural networks (with different dataset difficulties). We analyze the latent space of a model, which represents the training quality of a model and will serve as an important indicator for its robustness. The samples of each class are expected to form clusters in this space. We use both this visual representation of the 3-dimensional latent space and the silhouette histogram [1] and its entropy to measure the separability of these clusters.

The silhouette value of an object ranges between -1 and 1. Higher values mean the object is positioned better in the clustered space. The distribution of these values for a certain set of objects can serve as an indicator of the quality of the clustering. The silhouettes are calculated as

$$s(\vec{x}) = \frac{b(\vec{x}) - a(\vec{x})}{\max\{a\,(\vec{x})\,, b\,(\vec{x})\}} \tag{1}$$

with $a(\vec{x})$ being the average dissimilarity (distance) of an object $\vec{x}$ to all other objects of its own cluster X and $b(\vec{x})$ being the minimum average dissimilarity of $\vec{x}$ to all objects of all other clusters C_x [1, 55]. To quantize this primarily visual measurement, we calculate the entropy of the frequency distribution of the calculated silhouette using the following formula:

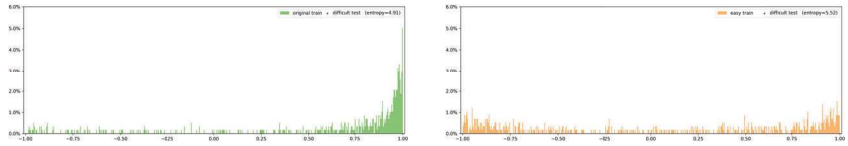
$$S = \sum_i (p_i * log_2(p_i)) \tag{2}$$

with each p_i being a bin of the histogram representing the silhouette values. In our examples we used 500 bins because it provided well distinguishable visual results, neither being too abstract nor too detailed to visually assess separation quality.

A well separated cluster will lead to a more robust implementation and a lower risk when applied to a real-world use case. These "easy" situations are characterized by a silhouette diagram where most values are concentrated in a peak on the right hand side (see Fig. 2a), leading to a low entropy value. In contrast, for "difficult" situations the silhouettes values are spread across the histogram (see Fig. 2b) leading to a higher entropy.

4 Application to MNIST

In this paper our method is applied on MNIST, a dataset of 70.000 fixed-size images of size-normalized and centered handwritten digits. It was created in 1998 by LeCun et. al. [2] based on a subset of the NIST handprinted forms and characters dataset [14].



(a) The model is trained with both “easy” and “difficult” data, the silhouette has a peak on the right side.

(b) The model is trained with “easy” data, the silhouette values are spreaded across the cosine similarity spectrum.

Fig. 2: Visual distinction of training dataset difficulty based on the silhouette. The same network was trained using differently difficult dataset splits and each time tested with the same previously unseen “difficult” data.

The complete process of our approach applied to the MNIST dataset is shown in the architecture diagram (see Fig. 3).

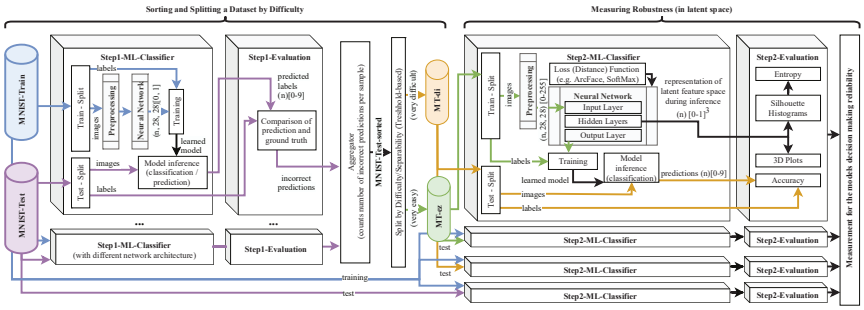
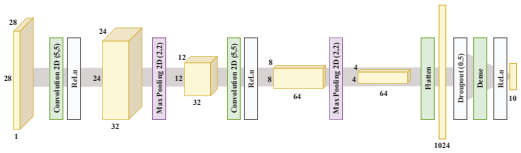


Fig. 3: Detailed overview of our architecture applied to the MNIST dataset

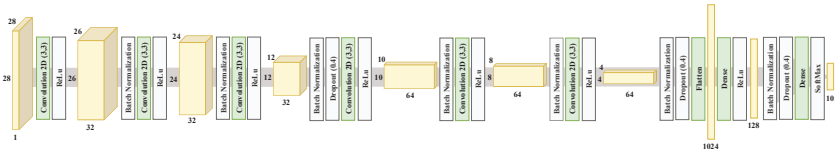
We use the original MNIST training and test split from the ‘keras.datasets‘ library and apply common preprocessing steps to reshape the list of pixels to a multidimensional array of shape (28,28,1), then rescale all values between 0 and 1 and finally one-hot encode the class labels.

For splitting the dataset by difficulty we use various architectures taken from popular publications and blog posts for image classification [15–17]. An exemplary selection of these is shown in Fig. 4 to visualize the differences in their approaches and complexity (different amount, selection, arrangement and sizes of layers and filters).

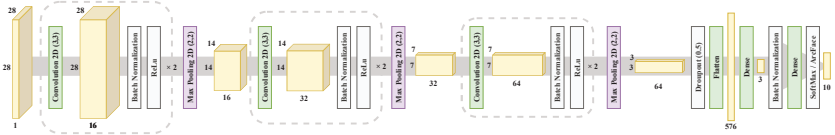
The trained deep neural networks for determining robustness are based on a version of the established VGG architecture [17]. We use the VGG8 architecture shown in Fig. 4c, which, unlike VGG16, allows us to process images smaller than 32x32px [17], which is important when applying this method to the MNIST dataset, of which the images have a size of 28x28px [2]. The architecture is retrieved from a repository containing an implementation of the ArcFace loss function [18], which we will use to visualize what the model has learned. By removing the output layer of the tested deep neural network architecture (SoftMax/ArcFace Layer in Fig. 4c), we can directly access the last dense layer representing the latent space of the model.



(a) CNN1 [15]



(b) CNN2 [16]



(c) VGG8 (VGG16: [17])

Fig. 4: Selected network architectures used in our experiments. Network layers are colored green, the dimension of data between layers is visualized by yellow boxes. The size of the convolutional filters is given in parentheses after the layer name (width, height), similar to the percentage of the drop-out in the corresponding layers.

In our experiments, we are measuring the dissimilarity of two objects $d(\vec{x}, \vec{y})$ using the cosine function where $\vec{x} \cdot \vec{y}$ is the dot product of $\vec{x}$ and $\vec{y}$:

$$d(\vec{x}, \vec{y}) = 1 - \frac{\vec{x} \cdot \vec{y}}{||\vec{x}||_2 \, ||\vec{y}||_2} \,. \tag{3}$$

Consisting of very basic mathematical functions, it is computationally simple and in our experiments provided well distinguishable results in the silhouette graphs.

5 Results

We apply the proposed method on the well-known MNIST dataset of handwritten digits [2] to demonstrate the effectiveness of our proposed approach.

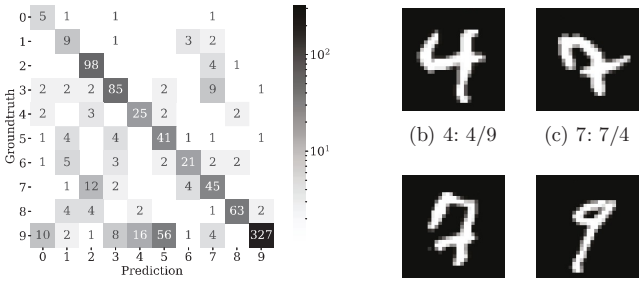
To determine the difficulty of the image samples, seven different CNN-based classificators were trained and used to classify the test dataset. We consider every element that was incorrectly classified at least once as part of the "difficult" data split, leading to a final split of 9632 "easy" and 368 "difficult" images (colored green and orange in Fig. 3).

Next, the classifiers shown in Fig. 4 are trained with the "easy" data and then were validated using the "difficult" data split and the standard accuracy measurement. The results in Table 1 indicate that the different network architectures have different learning curves and in general perform differently well, pointing out they differ enough to ensure a reliable selection of network architectures to measure dataset sample difficulty.

Table 1: Testing accuracy when training with “easy” and testing with “difficult” data.

Network	Validation Accuracy after 7 Epochs	Validation Accuracy after 100 Epochs	Training Accuracy after 7 Epochs	Training Accuracy after 100 Epochs
CNN1	0.56739	0.78190	0.99686	0.99956
CNN2	0.37174	0.84733	0.99476	0.99967
VGG8	0.45435	0.78408	0.97935	0.99978

To visualize the classification confusion, the predicted classes of the VGG8 classifier are grouped by their groundtruth classes in Fig. 5a) in the form of a confusion matrix. A confusion matrix representing a classifier with high accuracy shows high values on the main diagonal and small values outside of it. We consider training with the “easy” data and testing the “difficult” data as a difficult task, which explains why in this case high numbers appear outside the diagonal. The highest number of confusions is given for the case that a handwritten digit 9 is classified as 5, but we also see that the number 9 is generally over-represented in the “difficult” dataset. Example images of these confused MNIST digits can be seen in the other images of Fig. 5, labeled with the groundtruth followed by predictions of two classificators used in our experiments.



(a) Confusion matrix of final VGG8 classifier

(b) 4: 4/9

(c) 7: 7/4

(d) 7: 7/9

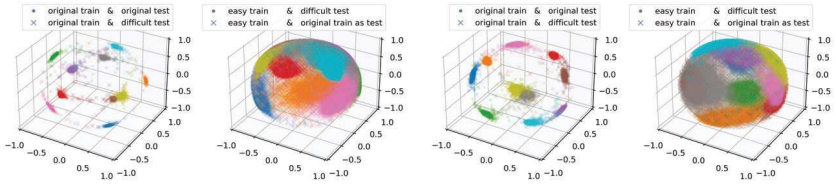
(e) 9: 9/7

Fig. 5: Confusion matrix of MNIST images and sample images [2], their groundtruths and predictions made by one (a) or two (b – e) classificators. The label of images b – e show the ground truth followed by the different predictions of the two classificators.

For all following experiments, we will use both the traditional SoftMax loss [3, 181] as well as the ArcFace loss [18].

To visualize what the model has learned, the latent space is plotted in a 3D space. As expected, the different MNIST classes form clusters in this space. Datasets varying in difficulty will lead to differently well separated clusters. The better the clusters are separated, the more robust the model should be. This 3D visualization of the learned clusters is shown in Fig. 6. For both losses, different combinations of training and test datasets are shown. This overlapped view is useful to detect outliers arising from difficult data. From this visualization alone, we can see that when training with not only the “easy” images, but the complete train dataset (which also includes more difficult elements), a far better separability could be achieved. We can also notice that in our experiments,

using the ArcFace loss [18] resulted in better separated clusters than using the SoftMax loss. This pattern is observed independently of the number of training epochs.



(a) ArcFace [18] loss function (b) SoftMax [3, 181] loss function

Fig. 6: Visualization of learned features in the latent space when trained for 100 epochs using different loss functions and different combinations of train and test datasets. Different colors represent different classes of the MNIST [2] dataset, different symbols are used to represent the different dataset combination.

Different Splits of difficulty lead to the distributions of silhouettes shown in Fig. 7. A high silhouette means an item is located near the center of the learned cluster.

The quantitative results of our experiments (shown in Table 2) and the visualized silhouette values (see Fig. 7) show the generally better performance of ArcFace over SoftMax and the benefits of training more epochs.

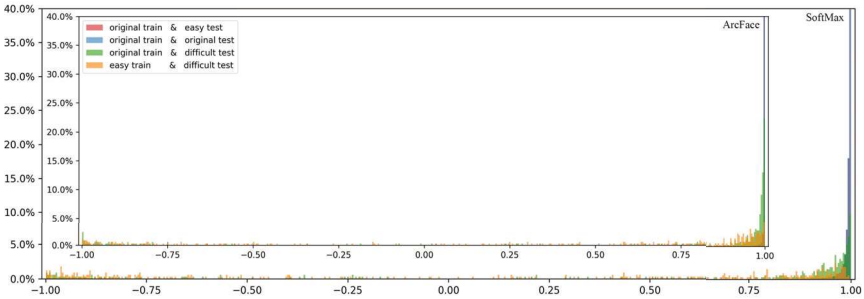


Fig. 7: Histograms of the silhouette values after training for 200 epochs with the SoftMax and ArcFace loss. Using ArcFace leads to more high silhouette values (especially in the rightmost bin) representing a better separation for all combinations of test and training datasets.

In general, if an image sample has a high silhouette value, it is well positioned in the clustered feature space. We can see that combinations that are deemed "difficult" lead to a wider distribution, e.g. training with easy images and testing difficult images, while "easy" combinations lead to a peak near the high end of the histogram. A wide distribution shows that there is no clear tendency to how good the elements are clustered, but a peak on the right side shows that most elements are clustered well. This is also

Table 2: Entropy values measured in our experiments. A low entropy is an indicator of good cluster separation in the latent space.

Dataset for		SoftMax Entropy after				ArcFace Entropy after			
training	testing	7	100	200	Epochs	7	100	200	Epochs
original train	easy test	2.09	1.65	1.57		2.13	0.39	0.30	
original train	original test	2.27	1.80	1.71		2.37	0.57	0.40	
original train	difficult test	4.89	4.45	4.21		4.91	3.24	3.38	
easy train	difficult test	5.39	4.96	5.09		5.52	4.99	4.81	

represented in the entropy value of each graph, shown in Table 2. The wider the graph, the higher the entropy. The smallest value for the "easy" train and "difficult" test scenario is given when training with ArcFace for 200 epochs. Overall, we see that in most cases training using the SoftMax Loss is less effective than training with the ArcFace Loss. This is also shown in the histogram in Fig. 7, as with SoftMax the classic "blue" combination of training and testing using the official dataset splits shows a wider peak at the right side than with ArcFace, visualizing that more elements are worse positioned in the feature space.

6 Conclusion

We proposed a framework to separate a dataset into different levels of difficulty using a majority voting approach and evaluated how models which are trained on the "easy" data split only perform when the "difficult" data split is used for testing. We ensure a both qualitative and quantitative measurement by evaluating plots of the latent space and silhouette histograms as well as the entropy value of the silhouette histogram.

This theoretical approach has been applied to the MNIST dataset [2] to prove its efficiency. The results of the experiments are as expected: training with a dataset consisting of only "easy" data leads to less robust models than training with the full dataset that also contains "difficult" samples. We have also proven that the entropy of the silhouette measure histogram and both the visualizations are useful to determine the robustness of an AI-model. During these experiments, we also measured that using the ArcFace loss instead of SoftMax leads to a better clustering and therefore more robust models in most cases.

7 Future Work

In future work, this approach will be transferred to more complex applications using action recognition to support the use cases in the area of life sciences. For this we will use an activity recognition video dataset and matching machine learning models for video action classification.

Independently, in future projects, the majority voting approach can be replaced by using the silhouette coefficient to separate "easy" and "difficult" samples. The assumption that a high silhouette value implies that the data sample is easy to classify will result in shorter training time, as the number of required classificators is reduced.

References

1. Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Journal of computational and applied mathematics **20** (1987) 53–65
2. LeCun, Y.: The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
3. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016) http://www.deeplearningbook.org.
4. Loh, W.Y.: Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1**(1) (2011) 14–23
5. Yang, Y., Morillo, I.G., Hospedales, T.M.: Deep neural decision trees. arXiv preprint arXiv:1806.06988 (2018)
6. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should i trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. (2016) 1135–1144
7. Lundberg, S.M., Lee, S.I.: A Unified Approach to Interpreting Model Predictions. In: Proceedings of the 31st international conference on neural information processing systems. (2017) 4768–4777
8. Štrumbelj, E., Kononenko, I.: Explaining prediction models and individual predictions with feature contributions. Knowledge and information systems **41**(3) (2014) 647–665
9. Burkart, N., Huber, M.F.: A Survey on the Explainability of Supervised Machine Learning. Journal of Artificial Intelligence Research **70** (2021) 245–317
10. Mani, S., Sankaran, A., Tamilselvam, S., Sethi, A.: Coverage Testing of Deep Learning Models using Dataset Characterization. arXiv preprint arXiv:1911.07309 (2019)
11. Settles, B.: Active Learning Literature Survey, University of Wisconsin-Madison Department of Computer Sciences (2009)
12. Wang, H., Bah, M.J., Hammad, M.: Progress in Outlier Detection Techniques: A Survey. Ieee Access **7** (2019) 107964–108000
13. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation Applied to Handwritten Zip Code Recognition. Neural computation **1**(4) (1989) 541–551
14. Grother, P.J.: NIST special database 19-Hand-printed forms and characters database. Technical Report, National Institute of Standards and Technology (1995)
15. Leban, J.: Image recognition with Machine Learning on Python, Convolutional Neural Network. Towards Data Science (2020) retrieved from https://towardsdatascience.com/363073020588 on 03.08.2021.
16. Deotte, C.: What is the best CNN architecture for MNIST? Kaggle notebook (2018) retrieved from https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist on 03.08.2021.
17. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556 (2014)
18. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 4690–4699

Acknowledgements

We thank J. Wetzel and T. Iraki for helpful comments and discussion. This work was funded by the Ministry of Science, Research and Arts of Baden-Württemberg (MWK) as part of the project Q-AMeLiA (Quality Assurance of Machine Learning Applications).