

Investigation of Capsule Networks Regarding their Potential of Explainability and Image Rankings

Felizia Quetscher^a, Christof Kaufmann^b and Jörg Frochte^c

Bochum University of Applied Science, 42579 Heiligenhaus, Germany

Keywords: Explainability, Capsule Networks, Dynamic Routing, Classification, Image Recognition.

Abstract: Explainable Artificial Intelligence (AI) is a long-ranged goal, which can be approached from different viewpoints. One way is to simplify the complex AI model into an explainable one, another way uses post-processing to highlight the most important input features for the classification. In this work, we focus on the explanation of image classification using capsule networks with dynamic routing. We train a capsule network on the EMNIST letter dataset and examine the model regarding its explanatory potential. We show that the length of the class specific vectors (squash vectors) of the capsule network can be interpreted as predicted probability and it correlates with the agreement between the decoded image and the original image. We use the predicted probabilities to rank images within one class. By decoding different squash vectors, we visualize the interpretation of the image as the corresponding classes. Eventually, we create a set of modified letters to examine which features contribute to the perception of letters. We conclude that this decoding of squash vectors provides a quantifiable tool towards explainability in AI applications. The explanations are trustworthy through the relation between the capsule network's prediction and the corresponding visualization.

1 INTRODUCTION

Through the rise of machine learning applications the demand for their explainability is increasing. In the report *Guidelines for Trustworthy AI* (Ala-Pietilä et al., 2019) the explainability of an AI system is classified as part of its transparency and it consists of two elements:

the ability to explain [...] the technical processes of an AI system and the related human decisions

When we use the term explainability, we refer to the technical part of this definition. This is further specified as requirement of an AI system to be *understood [...] by human beings* (Ala-Pietilä et al., 2019). We interpret this definition as the reasons that led to a decision of an AI system.

We focus on the classification task on images. Currently, the application of convolutional neural networks (CNNs) on this task is the state of the art, see e. g. He et al. (2016) and Zoph et al. (2018). Despite the ability of CNNs for image recognition, classifica-

tion and segmentation tasks, their decisions are neither always self-explanatory for humans nor always human-understandable at all.

Multiple approaches provide methods that aim to explain the results and the vision inside CNNs by isolating or highlighting important areas of the input image. This is done either by the creation of approximated models (Ribeiro et al., 2016, 2018) or by the additional calculations based on a trained model (Simonyan et al., 2014; Selvaraju et al., 2017). However, there is no approach yet that leads to a general conclusive solution to explain the vision of CNNs.

Because of the difficult comprehensibility of standard CNNs, we use an extension to CNNs to improve the explainability of classification tasks. One model architecture, that seems especially suitable for this task, is the capsule network (CapsNet) architecture proposed by Sabour et al. (2017). Their explanatory potential results from vectors, denoted as capsules, that potentially store human-understandable features. In a specific approach called *Dynamic Routing* the information transfer between the capsules is amplified or mitigated. During training, the vectors of the last capsules are masked and then inserted into a decoder to restore the perceived features of the input image. After training, the decoder supports explaining the

^a <https://orcid.org/0000-0003-1677-5858>

^b <https://orcid.org/0000-0002-0191-3341>

^c <https://orcid.org/0000-0002-5908-5649>

perception of the CapsNet.

Subsequently, we use the term *CapsNet* for the model that consists of the capsule network itself and also the decoder. In Sabour et al. (2017), the CapsNet is trained on the MNIST dataset (Lecun et al., 1998). It was shown that by modifying the elements inside the last capsules, features such as stroke thickness, width or scale change in the digit of the decoded image. Because these features are comparatively human understandable we examine the potential of CapsNets to create explanatory results.

We train a CapsNet model on the EMNIST letters (Cohen et al., 2017) dataset. The focus is set on the ability of the CapsNet to create explanatory image rankings. The term *image ranking* refers to the order of images based on their predicted class probability. The higher the position of the image in the ranking, the more it is associated with the considered class.

Firstly, we show that the vectors produced by the CapsNet are applicable for the creation of image rankings. Secondly, we create and explain the image rankings. The explanation is performed by the visualization of those areas that contributed to the prediction of the correct class. We extend the explanation by visualizing those features that contributed to the prediction of other classes. Finally, we explore the specific characteristics of letters that are displayed in an image.

Overall, the main contributions of our work is the examination of a CapsNet's potential and usability to

- create comprehensible image rankings for images of the same label and
- improve investigation techniques regarding the explainability.

2 EXPLANATORY APPROACHES OF CNNs

As mentioned above, there are in fact explanatory approaches for CNNs. In this chapter, we provide a brief overview about the properties of three fundamental explanatory approaches of CNNs: We cover the *LIME* approach (Ribeiro et al., 2016), *occlusion maps* (Zeiler and Fergus, 2014), *saliency maps* (Simonyan et al., 2014) and the *Grad-CAM* algorithm (Selvaraju et al., 2017).

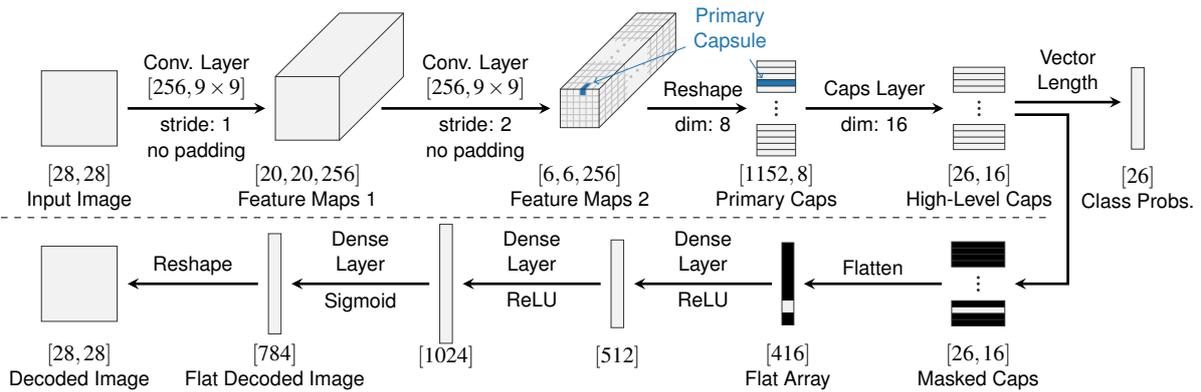
The LIME (Local Interpretable Model"-Agnostic Explanations) approach is a general method to explain single results of an AI model. It is not limited to any specific model architecture. The core idea of the LIME approach is the substitution of a multidimensional non"-human"-understandable model with

an easier interpretable but linear model as approximation. It is extended to non-linear approximations by *anchors* (Ribeiro et al., 2018). Both approaches result in the examination and isolation of those image areas that highly impact the class probability. However, the results of both approaches show that the isolated areas differ from those features that humans would use for their perception.

Occlusion maps as first proposed in Zeiler and Fergus (2014) are created by occluding different parts of the input image and hence this approach is model-agnostic as well. Rectangles filled with gray or random noise are often used as occluder. By shifting it through the image and recording the predicted class probability, it can provide insights which parts of the image are important for a specific class. However, a drawback is that the size of the occluder can influence the quality of the map. Also when different objects of the same or different classes are visible and a softmax output is used, occluding other objects can decrease or increase the class probability, respectively, which might lead to a wrong impression.

The approach to create saliency maps and the Grad-CAM algorithm are model-specific and directly applied to an available trained CNN model. Saliency maps visualize prominent pixels from a specified layer of a CNN by either using guided backpropagation (Springenberg et al., 2015) or inserting the output of a layer into the inverted model structure (Zeiler and Fergus, 2014). Both methods provide a rough orientation for the important features of a class. However, due to the evaluation of single outputs, the resulting features are not related to each other and no explanation for the decision-making of the CNN is included.

The Grad-CAM (Gradient-based Class Activation Map) algorithm (Selvaraju et al., 2017) computes the gradient of the last feature maps w.r.t. a specific class. The mean gradient of a feature map is used as its weighting, because it describes its importance for class. The positive values of the weighted average of the feature maps yields the class activation map. It highlights areas in the original image that increased the predicted class probability. Similar as saliency maps the results of the Grad-CAM algorithm are reasonable for a rough orientation for the CNN's decision. However, they primarily show that CNNs rely on different features for the classification than humans.

Figure 1: Architecture of a CapsNet with an image input size of 28×28 pixels and 26 output classes.

3 ARCHITECTURE OF THE USED CapsNet

The architecture of our CapsNet is illustrated in Figure 1. It is based on the architecture proposed by Sabour et al. (2017). The CapsNet model starts with two subsequent convolutional layers. We use a Leaky-ReLU activation function with a leak of $a = 0.01$ for both convolutional layers. Gagana et al. (2018) showed that Leaky-ReLU improves the performance compared to plain ReLU. From the output array of the second convolutional layer (named *Feature Maps 2* in Figure 1) the primary (also: low-level) capsules are formed by reshaping them. Each primary capsule consists of a group of n_p feature maps (here: $n_p = 8$) at a specific location. The number of values inside a capsule n_p must be a divisor of the number of feature maps n_m (here: $n_m = 256$), such that $\frac{n_m}{n_p}$ is the number of primary capsules per location (here: 32). Together with the dimensions $[h_m, w_m, n_m]$ of the feature maps (here: $[6, 6, 256]$) it results in the number N_p of primary capsules (here: $N_p = 1152$).

$$N_p = h_m \cdot w_m \cdot \frac{n_m}{n_p} \quad (1)$$

Similar to common neurons, capsules also have an activation function. As in Sabour et al. (2017), we use the squashing function

$$\hat{\mathbf{g}} = \frac{\|\mathbf{g}\|_2^2}{(1 + \|\mathbf{g}\|_2^2)} \frac{\mathbf{g}}{\|\mathbf{g}\|_2} \quad (2)$$

where \mathbf{g} is the vector of a capsule. It squashes the length of the output vector $\hat{\mathbf{g}}$ of a capsule between 0 and 1. The resulting length depends non-linearly on \mathbf{g} . The squashing function is performed to both primary and the subsequent high-level capsules.

The values in a capsule can be interpreted as neurons, since each primary capsule i is fully connected

to each high-level capsule j by a weight matrix \mathbf{W}_{ij} . However, there is no bias vector. The Dynamic Routing algorithm (Sabour et al., 2017) is executed between the primary capsules and the high-level capsules. It adds an additional coupling coefficient c_{ij} between each primary capsule i and each high-level capsule j , which stems from a routing logit b_{ij} by applying a softmax across j . The routing logits b_{ij} are initialized with zeros for each forward pass and updated within the routing iterations. Their values result from the relevance of the prediction of a primary capsule i to the prediction of a high-level capsule j , see Procedure 1 in Sabour et al. (2017). As a result, the connection between both capsules is amplified or mitigated.

The number of high-level capsules is equal to the number of classes. We use the EMNIST letter dataset (Cohen et al., 2017), which has 26 classes. The number of values per high-level capsule is arbitrary (here: 16). We refer to the output of a high-level capsule as *high-level squash vector* and for the array of all 26 high-level squash vectors as *high-level squash array*. On one hand the length of the high-level squash vectors is directly used as predicted class probability. On the other hand the high-level squash array is passed to the decoder for the reconstruction of the image.

For the decoder we also use the architecture proposed in Sabour et al. (2017), see the bottom part of Figure 1. Before the high-level squash array is inserted into the dense layers of the decoder, it is masked and flattened. During the training, the masking is executed for the true class of the input image. As a result, the values of the high-level squash vector for the true class stay while the values of all other high-level squash vectors are set to zero. When we evaluate the model after training, the masking is done for one or multiple arbitrary classes, depending on the purpose of the evaluation.

Like in Sabour et al. (2017), we use the margin loss for the predicted class probabilities L_C and the the mean squared error (MSE) loss for the decoder L_D . Both loss terms are combined with the weight d :

$$L_{LC} = L_C + d \cdot L_D \quad (3)$$

This also means, that the decoder is not only responsible to reconstruct the images, rather it provides a *regularization* for the CapsNet to learn the class representations inside the capsules.

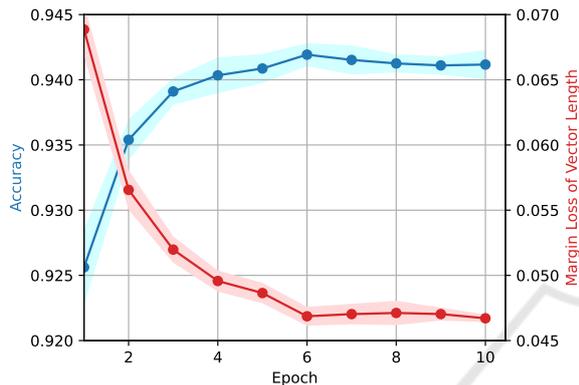


Figure 2: Accuracy and margin loss of the predicted class probabilities on the test dataset. Solid lines are mean values and areas are std. dev. of ten runs.

4 TRAINING OF THE CapsNet

The model is trained on the EMNIST letter dataset (Cohen et al., 2017) that contains 26 classes of handwritten white letters on a black background. Each class contains 4800 samples in the training set and 800 in the test set. To train the network we used the parameters summarized in Table 1.

During the training, the accuracy and the margin loss L_C is recorded for each epoch with a test dataset, see Figure 2. The loss of class probabilities L_C converges faster than the loss of the decoder L_D . To avoid

Table 1: Summary of applied parameters to train the CapsNet.

Training Parameter	Value
Epochs CapsNet (incl. Decoder)	10
Additional Epochs Decoder	20
Batch Size	100
Routing Iterations r	3
High-Level Capsule Dimension	16
Learning Rate	10^{-3}
Decay Rate per Epoch	0.9
Decoder Loss Weighting d	0.392

overfitting the decoder was trained separately for additional 20 epochs by providing the masked squash arrays as input data.

5 PERFORMING IMAGE RANKINGS WITH A CapsNet

To provide an impression about the appearance of the high-level squash array, an example for class A is displayed in Figure 3. The rows of the squash array contain the individual squash vectors for class A to class Z of the EMNIST letters dataset. The first row for class A contains the values with the largest deviation from 0 in both positive and negative direction. The length of this high-level squash vector, calculated by the euclidean norm, is indeed 0.95. This value is significantly larger than the lengths of the remaining high-level squash vectors for the other classes. Consequently, class A is predicted based on this high-level squash array.

5.1 Validation of the Squash Vector Length for its Usage in Image Rankings

Before ranking images by the length of the high-level squash vector, we confirm that the stored features in a high-level squash vector are able to represent the image. For that we visualize the stored features by decoding the high-level squash array masked for the true class. Then we measure the quality of the restored image using the mean structural similarity (SSIM) index (Wang et al., 2004) between the restored image and the original image. A scatter plot is created for all images from the test set relating the SSIM index with the length of the high-level squash vector for the true class, see Figure 4. We also compute the Pearson-Correlation coefficient between the lengths of the high-level squash vector and the SSIM indices of the restored images.

A positive correlation coefficient was found for each class ranging from 0.53 for class I to 0.91 for class V. The overview of all correlation values is shown in Figure 5. This relation supports the assumption that a larger high-level squash vector stores more features which result in a clearer reconstruction of the letter. It seems reasonable to evaluate the predicted class probability of an image by the high-level squash vector length and to use it for the creation of image rankings within one class.

Figure 6 shows the calibration plot for all predicted classes. The calibration curve is monotonically

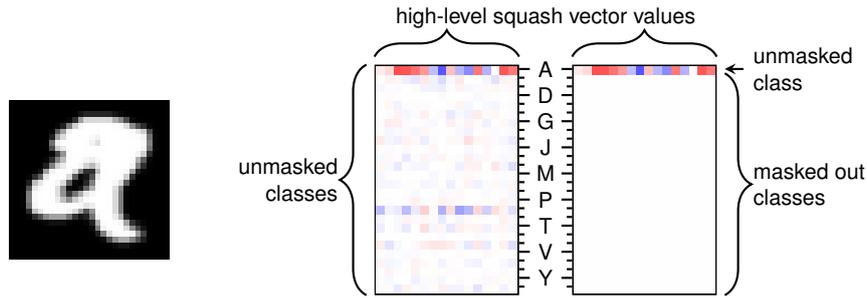


Figure 3: Visualization of a $[26 \times 16]$ -dimensional high-level squash array for the recognition of a sample image of class A. Left: unmasked squash array, right: squash array masked for class A. A color map from blue to white to red is applied, in which blue represents negative values, white zero and red positive values.

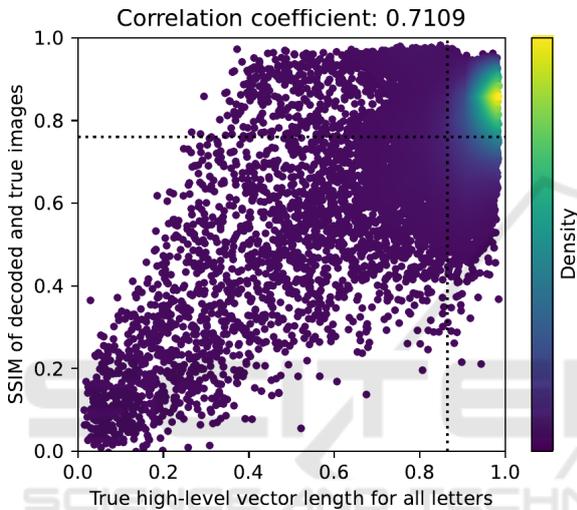


Figure 4: Correlation of the high-level squash vectors length for the complete test set with all classes and the corresponding mean SSIM index between the original image and the decoded image from a squash array masked for the true class. The dotted lines show the mean values.

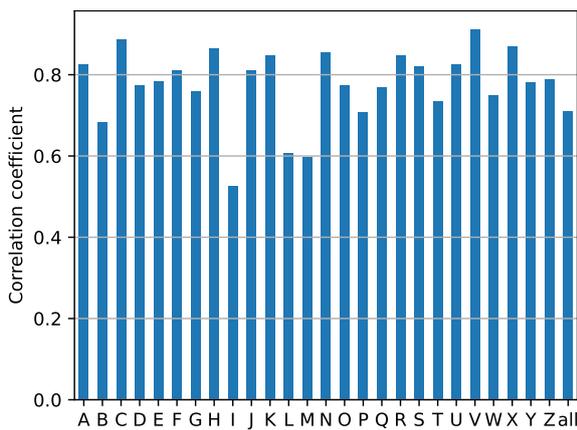


Figure 5: Correlation coefficients of high-level squash vector lengths and SSIM indices for each letter and once for all letters together.

increasing. This strongly supports the application of the high-level squash vector length for an image ranking, despite that there is a deviation from a perfectly calibrated curve. Because we do not calibrate the models, we denote the high-level vector lengths as *predicted probabilities*.

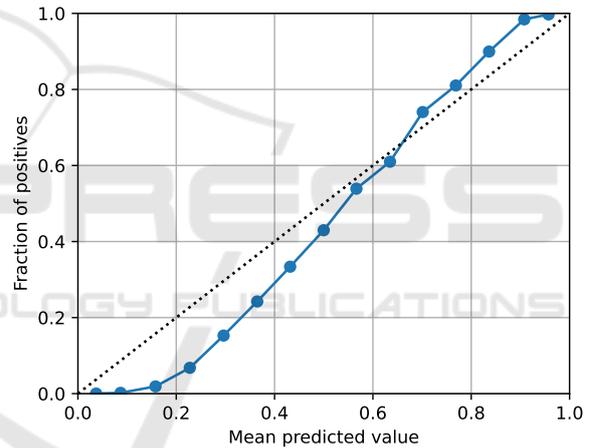


Figure 6: Probability calibration plot. The high-level squash vector length is used as probability. The plot uses the test dataset and aggregates all classes, where each class is handled in a one vs. rest manner.

5.2 Creation and Explanation of Image Rankings

In this section we provide an example for an image ranking based on high-level squash vectors and their explanation. In Figure 7 we rank eleven test images of class B with high-level squash vector lengths between 0.95 and 0.11.

One can observe that a small predicted probability yields a low intensity in the decoded image. Also, the letters in these reconstructed images appear smoothed in comparison to the original images. Irregularities, such as line breaks and additional serifs are reconstructed only to a small degree. These missing details

indicate that the CapsNet tends to learn a generalized representation of the class. One reason for that might lay in the rather large $[9 \times 9]$ convolutional filters that are applied in two convolutional layers.

As discussed in Sabour et al. (2017), when feeding the decoder with manipulated high-level squash vectors, the characteristics in the decoded image change in a certain way according to the modified values. This provides a way to explainability, but a tedious one, since the features are different for each class and might be hard to interpret. We use a different way. By masking the high-level squash array for different classes, the decoder reconstructs images of the corresponding classes. This allows to view the image interpreted as different classes. This is most interesting for classes with large predicted probabilities to visualize which part of the image contributed to the prediction.

The third image row in Figure 7 shows the decoded images based on the high-level squash vector masked for the class with the highest predicted probability, while ignoring the predicted probability for the true class B . The quality of these reconstructed letters depends strongly on the level of the predicted probability. Above a predicted probability of 0.80, the decoded letter is recognizable in all instances, while below 0.80 the letter is recognizable in some cases.

By the decoded images for the true class and for the class with the remaining largest squash vector we show how the letters are perceived by the CapsNet. Their areas cannot be transferred directly to the input image but rather they indicate why a letter was rated with a high probability. Good examples are the images in the sixth and ninth columns in Figure 7. These might be a small b with a missing part in the bottom

or a small h . The highest predicted probability is at the class H , but the class B also gets a high probability. The decoder together with the masking provides a method to see how the image can be interpreted as small b or h . This might also work for occluded image parts. The restoring of the missing characteristics of a class also supports the hypothesis that the CapsNet learns generalized shapes of the classes.

When both high-level squash vectors of one image have a large difference to each other, the recognition by the CapsNet is clear. However, the smaller the difference between the lengths of both high-level squash vectors, the larger is the ambiguity found in the original image. This is often the case for the combination of two high-level squash vector lengths between 0.30 and 0.80.

The last image row in Figure 7 shows the decoded image based on a high-level squash array masked for both classes included above. Through those images the interaction between the high-level squash vectors is visualized. The closer the lengths of both high-level squash vectors, the stronger is their mutual impact on the dual masked image. The impact is especially high in the range if the difference between both lengths is small.

As a result, the certainty for recognized features correlates with the length of the high-level squash vector. Through this relation a connection between the predicted class probability and the explanations is created. Thereby, the explainability results directly from the predictions of the CapsNet which leads to trustworthy results.

To examine which letters are frequently additionally detected in specific classes, the number of largest

Original	
Masked Prediction	
$\ v_B\ $	0.95 0.85 0.75 0.65 0.55 0.46 0.36 0.29 0.21 0.18 0.11
Class ⁻	H O O D H H O H H Z G
Masked Prediction	
$\ v^-\ $	0.13 0.24 0.15 0.35 0.42 0.59 0.21 0.23 0.81 0.72 0.74
Dual Masked Prediction	

Figure 7: Image ranking samples from test dataset for class B based on the high-level squash vector length. The decoded images, masked for class B , are shown together with the predicted probability of the original sample $\|v_B\|$ in the second row. The third row shows the decoded images, masked for the class with the largest high-level squash vector that is not B . In the bottom row the decoded images, masked for both classes is shown.

		Largest non-true predicted vector																												
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z			
Test data	A		18	23	46	21	17	101	76	40	12	21	13	14	43	41	43	120	25	10	8	29	12	19	7	17	24			
	B			44	59	17	10	92	161	34	10	25	31	7	18	17	31	86	29	17	12	5	24	31	6	5	11			
	C				12	135	14	113	19	66	7	44	55	9	10	44	35	63	24	15	14	14	9	21	2	10	37			
	D					14	14	60	87	35	33	18	22	9	32	85	36	43	25	5	16	16	25	20	12	12	49			
	E						30	87	23	35	8	68	12	7	13	22	115	55	39	6	11	17	16	18	0	7	33			
	F							65	29	82	24	52	19	3	13	4	106	93	38	24	80	4	9	12	9	14	13			
	G								27	41	30	5	9	3	16	8	14	275	11	44	4	4	10	13	5	23	28			
	H									67	8	72	34	38	76	16	23	39	19	6	28	25	25	45	9	12	7			
	I										53	2	622	4	8	1	2	9	4	0	9	2	2	4	2	4	22			
	J											9	96	4	10	18	16	33	2	36	37	13	31	18	9	10	85			
	K												25	12	14	12	42	46	91	9	28	24	18	35	62	18	15			
	L													0	2	5	9	9	5	0	17	10	18	5	4	5	7			
	M															98	6	39	23	21	6	8	13	53	86	11	18	47		
	N																17	35	29	66	5	17	30	33	112	35	14	16		
	O																	33	55	7	8	23	35	12	21	3	5	27		
	P																		115	59	2	19	17	22	15	7	25	36		
	Q																			6	8	1	7	13	16	9	19	13		
	R																				6	44	6	76	12	22	38	38		
	S																					7	5	7	106	3	10	75		
	T																						3	4	15	28	23	75		
	U																							186	54	5	32	10		
	V																							20	147	24	22	56	17	
	W																								10	59	64	9	12	15
	X																										98	24		
	Y																										77	27		
	Z																												13	

Figure 8: Counts of the highest predicted class probability that is not the true class using the full test dataset.

high-level squash vectors is accumulated for the non-true class, see Figure 8. We see that the feature of class *L* is often found in images of class *I* (622 times) and vice versa (599 times). This explains why the correlation coefficient between the high-level squash vector length and the SSIM index was low for both classes. A similar but less distinct phenomenon could appear at the next high combinations, such as *G* and *Q* (275 and 349 times) as well as *U* and *V* (186 and 147 times). The matrix provides an insight to the perception of the CapsNet because it shows which classes are found most frequently within other classes.

5.3 Exploration of Perceived Features

To explore the characteristics perceived by the CapsNet in more detail we create a set of modified images that contains the letter pair (*R*, *K*) from the test data. The letter *R* is gradually transformed to the letter *K* by hand, see Figure 9. The images are inserted into the CapsNet and the resulting high-level squash arrays are masked for both classes separately. The corresponding decoder outputs and the high-level squash vector lengths for classes *R* and *K* are shown. The lengths

are mostly decreasing for class *R* while increasing for class *K*. As Figure 5 proved, often a larger high-level squash vector leads to a clearer reconstruction of the letter. This is confirmed by Figure 9.

The length of the high-level squash vector changes non-linearly between the samples in Figure 9. There are one or two images in which the squash vector length together with the decoded image quality rises or falls abruptly. According to Figure 6, this behavior might be a sign for overconfidence. The threshold in the squash vector length could work as a support to investigate the features that are crucial for the CapsNet to detect a class. Apparently in this specific case, the top line of the original letter is a decisive factor of the CapsNet for or against class *K*. Equally, the connection of the loop for the letter *R* must have a certain intensity for the CapsNet to find the class *R*. Several of the missing features are interpolated, we suspect, towards generalized letters which maximizes activation. This assumption is supported by the rise of the squash vector length from the fifth to the sixth image. We assume, this occurs because the sixth image resembles one of the generalized letters for class *R* more than the fifth image.

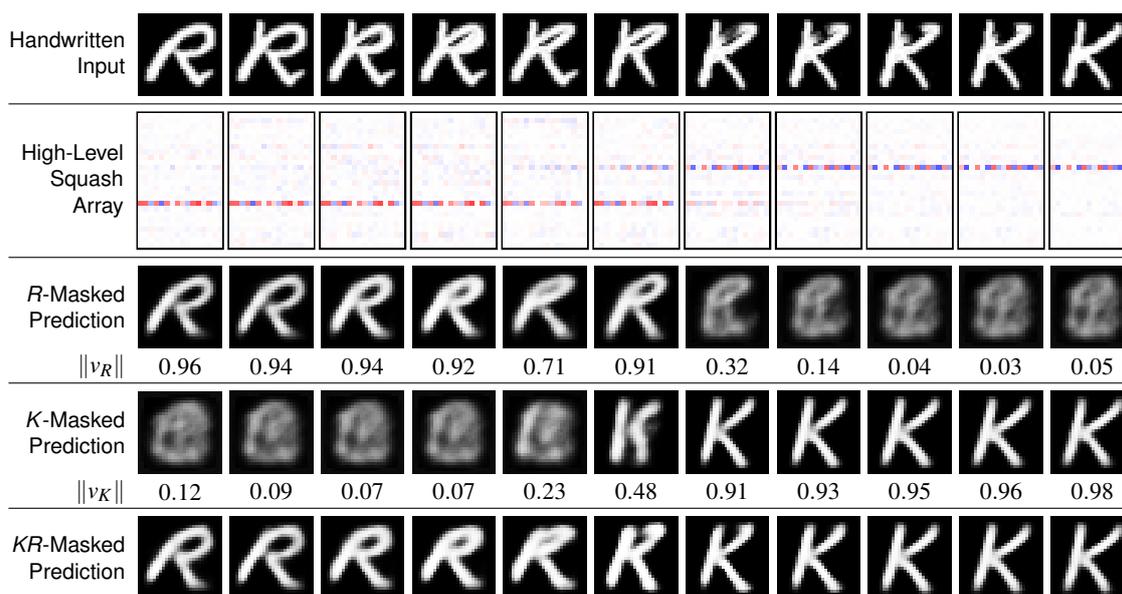


Figure 9: Predictions of the decoder for the image set containing morphed images between the classes K and R .

The explored threshold contributes to the explanatory approach with the CapsNet. On this basis, the characteristics that are important to the recognition of a class can be extracted. The threshold of two classes is not necessarily on the same point. This results in the capability to recognize ambiguous images. Through the decoding of the squash array masked for one class it is explainable which characteristics of the input image were perceived by the CapsNet. Furthermore, the ambiguous letters are also letters that are ambiguous for the human perception.

6 CONCLUSION AND FUTURE PROSPECTS

In the introduction we referred to the term *explainability* as an AI system that is *understood [...] by human beings* (Ala-Pietilä et al., 2019). With the high-level squash array of the CapsNet together with the decoder we examined a strong explanatory tool. We showed that the length of the high-level squash vector is applicable as predicted class probability and that a ranking based on its length is reasonable.

The image rankings were explained by decoding specific high-level squash vectors. The resulting characteristics showed how the images can be interpreted as the true class and as another class. Thereby, we could explain which areas were misrecognized by the CapsNet. Based on the high-level squash vector length we could explain the degree of the misinterpretation. Finally we showed, based on the transfor-

mation of specific images that the features used by the CapsNet are comparable to the human recognition.

In conclusion, the length of squash vector provides an explainable and quantifiable tool for image classification. Its advantage above post-hoc explanatory approaches is the connection of the class probability and the explanation by visualizing the features of the high-level squash array. Both outputs rely on the values stored in the high-level squash array resulting in a high trustworthiness of the explanations.

ACKNOWLEDGEMENTS

The authors of this work were funded by SUMA e. V. as well as the federal state of North Rhine-Westphalia and the European Regional Development Fund FKZ: ERFE-040021.

REFERENCES

- Ala-Pietilä, P., Bauer, W., Bergmann, U., and Bietliková, M. (2019). *Ethics guidelines for trustworthy AI*. EU Publications.
- Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926.
- Gagana, B., Athri, H. U., and Natarajan, S. (2018). Activation function optimizations for capsule networks. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1172–1178. IEEE.

- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). “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*, pages 1135–1144.
- Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-Precision Model-Agnostic Explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).
- Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 3859–3869, Red Hook, NY, USA. Curran Associates Inc.
- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626.
- Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*.
- Springenberg, J., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*.
- Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.
- Zeiler, M. D. and Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.
- Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.