Reproduction of DL for Case-Based Reasoning: Results Replicated

Tom Lotze
11161159
University of Amsterdam
Tom.Lotze@gmail.com

Berend Jansen
11051949
University of Amsterdam
bereeend@gmail.com

Stan Lochtenberg
12766739
University of Amsterdam
Stan.Lochtenberg@student.uva.nl

Cees Kaandorp
11061642
University of Amsterdam
Cees.Kaandorp@hotmail.nl

ABSTRACT

The aim of this paper is to reproduce the results of Li et al. [9], who tried to make neural network decision-making transparent by using image similarity to prototypes. In this paper, we successfully replicate their results by using artifacts of the original authors, and hence award the paper with the *Results Replicated* ACM badge. Moreover, we further test the capabilities of the model by trying to classify a custom made colored MNIST dataset, with random realistic backgrounds, while keeping interpretable prototypes. While the obtained classification accuracy of 0.947 is high, the prototypes cannot be considered meaningful. When converting this dataset to grayscale, more meaningful prototypes can be produced, indicating that three color channels are too noisy for realistic reconstruction of prototypes. We also tested the model on CIFAR-10, but meaningful prototypes are lacking because of high intra-class variance in the dataset. More complex models or partial prototypes could be a solution for some of the problems, provided that they are noise invariant and adaptive to color range shifting, a problem that is left for future research.

1 INTRODUCTION

With machine learning models being used for increasingly important decision-making, so grows the urgency of making these systems transparent. The reasoning process behind a prediction can be determined by the use of an 'explainable' machine learning model in which one can obtain an interpretable explanation of the prediction. The production of interpretable models can be a great challenge however, especially for neural networks. Neural architectures are characterized by their "black-box" decision-making, meaning it is often hard for humans to truly grasp the inner workings of the network.

Explanations for black-box models can be generated in a several ways. First, a post hoc interpretability analysis can be performed, where the objective is to interpret a model or its features after the training process is completed. Hence, the interpretability analysis becomes a separate modelling effort. A few methods have been introduced that produce feature importance summaries after the training process is completed to explain the model, for example, LIME [13], SHAP Lundberg and Lee [10] and Integrated Gradients [16]. Another group of techniques for obtaining explanations of model predictions is the use of example-based explanations. Rather than focusing on explaining the black box with feature summaries of the model, these methods choose to concentrate on exemplary data instances. A few methods which belong to this group are Counterfactual Examples [7], Adversarial Examples [15] and Influential Instances [5].

Li et al. [9] proposed another example-based explanation method: the use of prototypes. In this type of interpretability modelling, the predictions for a data instance are made based on similarity with learned prototypes. The definition of a "prototype" can vary across the literature, in this context a prototype is a learned representation of a class that is near or identical to an actual instance of the training dataset. The model will assign a probability to every class based on the similarity between the input image and the different prototypes. The weights between different prototypes and the classes, and the similarity score for each prototype make the decision making process more transparent.

Li et al. [9] tested their architecture on three datasets: MNIST handwritten digits [8], color images of cars from different angles [3] and the MNIST fashion dataset [17]. These datasets provide relatively simple classification problems since they all have uniform backgrounds, and exhibit relatively low intra-class variation. Consequently, a single prototype can represent the average of a class, something that gets more difficult if the intra-class variation increases. Moreover, the class instances will be clustered in the latent space. However, when dealing with more complicated datasets, class regions might overlap instead. The prototypes in latent space will learn to be in the non-overlapping class-regions, failing to be representative for the whole class, including the overlapping instances Gee et al. [4].

In this paper, we aim to reproduce the prototype architecture and test it on a more challenging and realistic dataset: the MNIST dataset with a non-uniform colored background. Our objective is to ensure high accuracy on the more complex images, while using an identical network structure with interpretable and meaningful prototypes to keep the decision making process transparent.

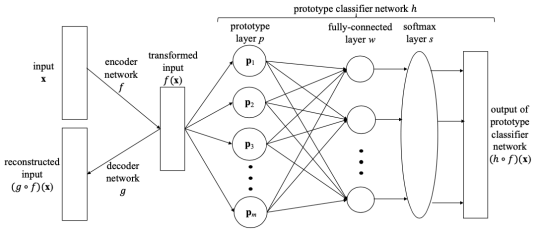


Figure 1: Architecture of the model. Image taken from Li et al. [9]

2 METHODS

2.1 Model description

An existing implementation of the model has been made available by Li et al. [9], which makes use of the TensorFlow package. While keeping the architecture of the model identical, our implementation makes use of the PyTorch framework. The model consists of two main parts: an autoencoder model (encoder, decoder), and a classification network to predict the class of the input images. A visual overview of the architecture is shown in figure 1.

The encoder and decoder both consist of four convolutional layers, using ReLUs in between each layer, except for the last layer of the decoder, where a sigmoid is used to ensure pixel values in $[0, 1]$. The latent representation is the input for the classification network consisting of a prototype layer and a fully connected layer, followed by a softmax module. The prototype layer stores the prototypes (in latent space) and outputs the L^2 distance between the latent representation of the input and the different learned prototypes. The fully connected layer computes the weighted sum of these distances. A softmax module can be added to get a probability distribution over the classes.

The achieved interpretability is three-fold: prototypes can be decoded and mapped to the pixel space to visualize prototypical examples for the classes, the distances to prototypes can be inspected to determine the most similar prototype per input image, and the weights of the final layer indicate the contribution of individual prototype distances to the classes.

To ensure proper functioning of this architecture, the cost function is four-fold and can be seen in equation 1. The four factors are weighted by the hyperparameter $\boldsymbol{\lambda} = \{\lambda_{class}, \lambda_{ae}, \lambda_1, \lambda_2\}$ and have the following roles: the classification error (E), which is the cross entropy loss between the predictions and targets, the reconstruction loss of the autoencoder (R), defined by the mean of the norms of the differences between the input and the reconstructed image, a regularization term (R1) pushing the prototype vectors to have meaningful decodings in pixel space, and a regularization term (R2) to cluster the training examples around prototypes in latent space. The full mathematical definition for each of the terms can be found in Li et al. [9]. Also, an overview of all the hyperparameters can be found in appendix C.

$$L = \lambda_{class} * E + \lambda_{ae} * R + \lambda_1 * R1 + \lambda_2 * R2 \tag{1}$$

3 EXPERIMENTAL SETUP

3.1 Reproduction of Li et al. (2018)

First of all, we tested our implementation on the standard MNIST digits dataset to reproduce the results of the original paper [9]. This dataset consists of grayscale handwritten digit images of 28×28 pixels, divided in 55,000 training images, 5,000 validation images and 10,000 test images. After confirming our implementation was correct and consistent with the original implementation, we tested the model on more complicated datasets.

3.2 CIFAR-10 dataset

To test the model performance on more complex and realistic datasets, we used the CIFAR-10 dataset, consisting of $60,000$ RGB images of 32×32 pixels in 10 classes [6]. The ratio between train/validation/test is the same as for the standard MNIST. However, as already stated by Li et al. [9], the model does not produce meaningful prototypes on natural images, such as the images in CIFAR-10.

3.3 Colored MNIST

To find a middle ground between natural images and the simple MNIST digits, we created a new dataset using the MNIST digits, with a part of the standard "Lena" test image (see appendix, fig 7) as an artificial background.[1] Examples of this new dataset are shown in the appendix (fig 8). We aim for this new dataset to be positioned in between the standard MNIST and CIFAR-10 dataset in terms of complexity. The new dataset shows clearly visible digits, while the background for every image is different, increasing the complexity compared to standard MNIST. The number of examples used for training and the dimensions of the images are unaltered. Using the default parameters and architecture as mentioned in section 2, the decoded prototypes were not interpretable. To find values for λ yielding better prototypes, we performed a grid search.

3.4 Lambda Grid search

To find the models producing the best prototypes, we performed a grid search over multiple values for λ. In the first run, we kept λ_{class} and λ_1 fixed at 20 and 1 respectively, while testing all combinations for λ_{ae} and λ_2 taking on values $1, 5, 10,$ and 20. In the second run, we kept λ_{class}, λ_{ae}, and λ_2 fixed at 20, 1, and 1, respectively, while testing values $1, 5, 10,$ *and* 20 for λ_1.

3.5 Grayscale colored MNIST

Since none of the lambda combinations led to interpretable prototypes (see section 4), we investigated the effect of color by converting the colored dataset, including background, to grayscale, making it less complex. However, all of the digits still have a different (random) background, and since the color distribution of every digit is changed, they remain in different shades of gray, ruling out the possibility of the model overfitting on a single shade of gray. For the grayscale dataset, we performed the same grid search as described above.

[1] Inspired by https://github.com/wouterbulten/deeplearning-resources

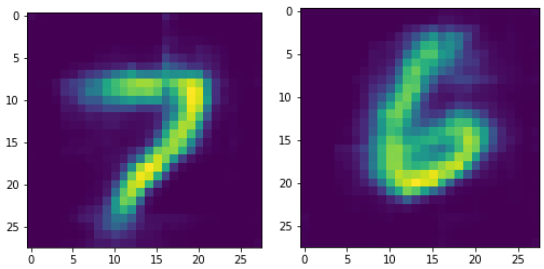


Figure 2: Two examples of prototypes for standard MNIST dataset. Prototype generated by model as described in original paper

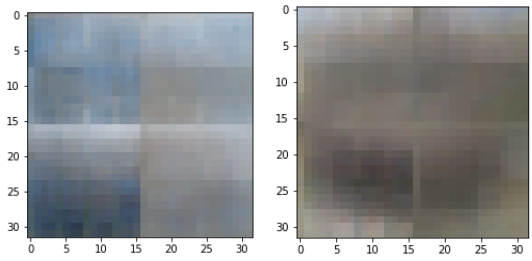


Figure 3: Two examples of typical prototypes for CIFAR-10. Prototype generated by standard model adapted for CIFAR-10.

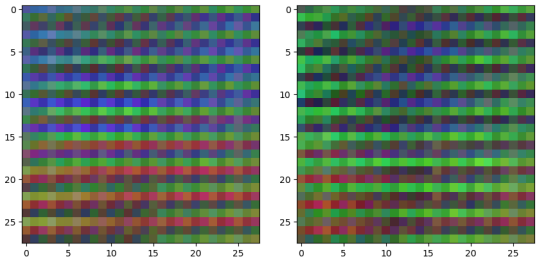


Figure 4: Two examples of typical prototypes for colored MNIST dataset. Prototypes generated by PM with $\lambda_{class} = 20, \lambda_{ae} = 10, \lambda_1 = 1, \lambda_2 = 1$ after 25 epochs.

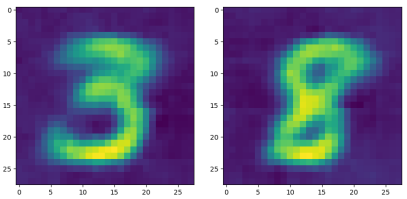


Figure 5: Two examples of typical prototypes for gray MNIST dataset. Prototype generated by model with $\lambda_{class} = 20, \lambda_{ae} = 10, \lambda_1 = 1, \lambda_2 = 1$ after 25 epochs.

3.6 Linear Model

To measure the effect of the prototype layer on performance, we also constructed a Linear Model (LM), in which the prototype layer is replaced by a fully connected layer, followed by a ReLU (the decoder is removed from the LM). We tested this LM on the colored MNIST and the grayscale MNIST dataset, as no other models have ever been tested on this data, to see the difference in accuracy compared to the Prototype Model (PM). This shows the cost of interpretability in terms of accuracy loss.

4 RESULTS

4.1 Standard MNIST

First of all, we reproduced the results of Li et al. [9]. Examples of our prototypes for the standard MNIST dataset are shown in figure 2 and qualitatively match the prototypes reported by Li et al. [9]. The test accuracy on this standard MNIST dataset was 0.991 and can be compared to the other datasets in table 2.

4.2 CIFAR-10

The CIFAR-10 dataset [6] is too complex for this architecture; the accuracy on the test set did not exceed 0.6431 and the prototypes did not resemble instances of any of the classes. However, we hypothesize that the different colored sections (for example, blue and brown) in the prototypes (figure 3) are the result of the absence or presence of ground, air, and or water in the different classes. Moreover, the lack of interpretable prototypes is the result of relatively large intra-class variation compared to the MNIST dataset. In other words, there is no prototypical image that resembles all the different images of e.g. airplanes or ships in this dataset, and hence there is no meaningful prototype.

4.3 Variants of MNIST

When running the same model on the colored MNIST with background, typical prototypes after training convergence (30 epochs) are shown in figure 4. Despite these prototypes not being interpretable, the PM achieved an accuracy on the test set of 0.947, which is slightly worse than the LM (0.956).

On the gray dataset, the prototypes could be interpreted (figure 5), and the quality is comparable to the standard MNIST dataset. Furthermore, an accuracy of 0.977 was achieved. This supports the claim that natural, colored images are too complex for this model to classify based on meaningful prototypes. On this dataset, the difference between the PM and the LM is marginal (PM: 0.977 vs. LM: 0.980).

To validate the prototypes and their effect on decision making, the weights to every class for every prototype are shown in table 1. In every row, the lowest weight (in bold) corresponds to the visual class of the prototype shown, as indicated in the last column. The reason for the negative weights is that classification is done through the dissimilarity measure in the prototype layer. In case the dissimilarity between a prototype of a certain visual class and an input example is high, the negative weight leads to a low activation for this particular visual class. In case the input example and the prototype are similar (small dissimilarity value), the negative weight is multiplied by a small value, and its effect diminishes. Essentially, all wrong classes get low activations, and the correct class is predicted.

	0	1	2	3	4	5	6	7	8	9	argmin
	0.36	0.33	-0.2	-0.72	0.67	-0.13	0.81	**-0.89**	0.3	0.14	7
	-0.22	-0.3	**-1.58**	-0.22	0.28	0.77	0.12	0.16	0.47	0.04	2
	0.14	-0.32	0.15	-0.23	0.28	0.47	0.35	0.16	**-1.64**	0.08	8
	**-1.03**	0.47	-0.25	0.48	0.23	-0.31	0.18	0.07	-0.2	0.49	0
	0.2	-0.51	0.36	0.28	**-1.24**	0.3	-0.23	0.06	0.24	0.12	4
	-0.03	-0.08	-0.24	-0.08	0.16	-0.15	**-0.84**	0.55	0.23	0.16	6
	**-0.99**	0.49	0.02	0.37	-0.12	0.19	-0.2	-0.48	0.05	0.64	0
	0.7	**-1.4**	0.21	-0.68	0.28	0.85	0.41	-0.02	-0.46	0.46	1
	-0.34	0.57	0.45	0.34	-0.04	-0.87	**-1.17**	0.39	-0.75	-0.05	6
	-0.02	-0.37	0.25	0.52	**-1.08**	-0.1	-0.1	-0.03	0.57	0.24	4
	0.11	0.05	-0.16	-0.34	0.41	-0.17	**-1.15**	0.48	0.49	-0.36	6
	0.4	0.19	-0.13	0.37	0.17	0.12	0.22	-0.27	0.31	**-0.93**	9
	0.38	0.26	0.13	0.57	-0.01	0.12	0.22	0.22	-0.51	**-1.07**	9
	0.13	0.29	0.51	0.18	-0.07	-0.3	0.48	-0.31	0.14	**-0.91**	9
	0.26	-0.0	0.71	**-1.65**	0.22	-0.9	0.37	0.07	0.54	0.28	3

Table 1: Prototype weights for the Gray MNIST dataset. Prototypes after 30 epochs with lambda values: 20, 10, 1, 1.

Model on dataset	Accuracy	Nr. epochs
LM on MNIST color	0.956	40
LM on MNIST rgb2gray	0.980	20
Li et al. [9] on standard MNIST	0.995	1500
PM on standard MNIST	0.991	20
PM on MNIST color	0.947	20
PM on MNIST rgb2gray	0.977	25
PM on CIFAR-10	0.643	30

Table 2: Performance of the tuned model on different datasets. Accuracy is computed on hold-out test set. PM **refers to the Prototype Model,** LM **refers to the Linear Model**

λ_{class}	λ_{ae}	λ_1	λ_2	Accuracy
20	1	1	1	**0.951**
20	5	1	1	0.951
20	20	1	1	0.950
20	10	1	1	0.949
20	1	1	10	0.947

Table 3: Colored MNIST: Top 5 tuned models and their classification accuracy on test set. Accuracy rounded to 3 decimals.

λ_{class}	λ_{ae}	λ_1	λ_2	Accuracy
20	10	1	1	**0.981**
20	1	1	1	0.980
20	5	1	1	0.979
20	5	1	5	0.979
20	1	1	5	0.972

Table 4: Gray MNIST: Top 5 tuned models and their classification accuracy on test set. Accuracy rounded to 3 decimals.

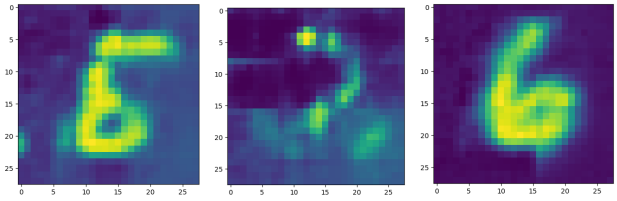


Figure 6: Prototypes learned in grid search; left: $\lambda_{class} = 20$, $\lambda_{ae} = 10$, $\lambda_1 = 1$, $\lambda_2 = 10$; **middle:** $\lambda_{class} = 20$, $\lambda_{ae} = 10$, $\lambda_1 = 1$, $\lambda_2 = 20$; **right:** $\lambda_{class} = 20$, $\lambda_{ae} = 1$, $\lambda_1 = 10$, $\lambda_2 = 1$

4.4 Lambda grid search

We observe little variation in accuracy between the different lambda combinations for both the colored and gray dataset, as shown in tables 3 and 4. On the colored dataset, the combination $\lambda_{class} = 20, \lambda_{ae} = 1, \lambda_1 = 1, \lambda_2 = 1$ performs best by a small margin, with an accuracy of 0.951. On the gray dataset, the combination $\lambda_{class} = 20, \lambda_{ae} = 10, \lambda_1 = 1, \lambda_2 = 1$ performs best by a small margin, with an accuracy of 0.981. However, the quality of the prototypes varies drastically between the different lambda settings. For the colored MNIST the prototypes are less noisy, but for any setting still not interpretable. For the gray dataset we see large differences in the prototypes, as can be seen in figure 6. Increasing λ_2 has a negative impact on the interpretability of the prototypes. Additionally, increasing λ_1 does not lead to better prototypes. For completeness, all prototypes for the lambda settings in figure 6 are shown in appendix E.

5 DISCUSSION

We have shown that this model can work on more complex datasets than the original paper presented. However, the complexity of color images still cannot be represented by meaningful prototypes by this model. This problem is therefore left to future research.

There are some remarks to be made about the original paper [9], concerning transparency and reproducibility. First of all, there are some discrepancies between the model described in the paper

and in the available code. For example, in code provided by the authors, ReLUs are used in the autoencoder, whereas in the paper it is indicated that sigmoid non-linearities are used in every layer. Moreover, only the implementation on the MNIST digits was made available, excluding code for the cars and fashion MNIST dataset.

To deal with the problem of intra-class variance within the dataset, partial prototypes could be further investigated, as the authors of the original paper are doing in the follow up paper [1]. The essence of partial prototypes would be matching specific parts of images to prototypes, e.g. the shape of a cockpit, or a horse its head, etc. Furthermore, different distance metrics (other then L^2) could be used when comparing the latent representations with the prototypes.

On the other hand, one can approach the lack of interpretable prototypes as the result of a lack of expressive power of the autoencoder. Future research could look into applying the prototype layer to more complex autoencoder architectures, to increase this expressive power. However, this method of prototypes can only work if all the members of a certain class have characteristics in common, which are displayed in the same way in every example (i.e. same viewpoint, rotation and scale). Using partial prototypes might partly alleviate this problem, but there still needs to be a basis within the class for the (partial) prototypes.

6 BROADER IMPLICATIONS

Machine learning models acquire an increasing amount of responsibilities in today's society. Therefore, it is of great essence that these models can be trusted. A large variety of research has been performed to realize trustworthy models, by understanding the reasoning behind these complex models.

Transparency is closely connected to the concept of Fairness. Mehrabi et al. [11] define fairness as "the absence of any prejudice or favoritism toward an individual or a group based on their inherent or acquired characteristics". Unfair machine learning can have different causes. One of these causes is that existing bias from the data might be encoded into the prediction model. Transparency can help to track down these biases by providing an explanation for the produced output. Prototypes can help to shine light on biases that the data has created, as they are representations of the data. As the model produces an output, we can observe based on which prototype the prediction has been made. Rudin [14] argues that in addition to helping to detect data biases, interpretable models can also help to detect possible data privacy issues. Transparency can provide an insight to what personal data is ultimately used in the decision-making. In general, transparency can not solve fairness issues, but it is able to provide a good insight into the origin of the bias. Subsequently, necessary procedures can be taken based on the information that the interpretable model provides.

The method of using prototypes to explain the model's workings could pose some problems regarding the confidentiality. Namely, saved prototypes could have close resemblance to a data instance and contain personal information. Removing the decoder is not an option, as this would defeat the purpose of having prototypes in the first place. A solution would be to force the prototypes to not closely resemble specific data instances in order to provide *differential privacy* [2]. However, within the prototype model this is a difficult task, since the minimization of the R1 regularization term pushes each prototype vector to correspond to at least one of the training examples in the latent space.

7 CONCLUSION

This project was undertaken to reproduce the paper of Li et al. [9] and evaluate their findings. While the prototype based model works almost perfectly on relatively simple datasets with low intra-class variance, realistic image classes cannot be captured well by input-sized prototypes. In this paper, we reproduced the original results [9], and explored the model's performance on new datasets, such as colored and grayscale MNIST digits. We show that the colored dataset is too complex for the current model architecture, something that might be improved in future research.

For the replication objective, we award the paper of Li et al. [9] with an "results replicated" badge from ACM, as "[t]he main results of the paper have been obtained in a subsequent study by a person or team other than the authors, using, in part, artifacts provided by the author" [12]. The code provided by Li et al. [9] was used as starting point for the code that we used to produce the results in this paper.

Code: All of the code used to generate the results in this paper is available on https://github.com/Tom-Lotze/FACT.

REFERENCES

[1] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. 2019. This Looks Like That: Deep Learning for Interpretable Image Recognition. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8928–8939. http://papers.nips.cc/paper/9095-this-looks-like-that-deep-learning-for-interpretable-image-recognition.pdf

[2] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[3] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. 2012. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *Advances in neural information processing systems*. 611–619.

[4] Alan H. Gee, Diego García-Olano, Joydeep Ghosh, and David Paydarfar. 2019. Explaining Deep Classification of Time-Series Data with Learned Prototypes. *CoRR* abs/1904.08935 (2019). arXiv:1904.08935 http://arxiv.org/abs/1904.08935

[5] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1885–1894.

[6] Alex Krizhevsky. 2012. Learning Multiple Layers of Features from Tiny Images. *University of Toronto* (05 2012).

[7] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2019. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *arXiv preprint arXiv:1907.09294* (2019).

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[9] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[10] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*. 4765–4774.

[11] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635* (2019).

[12] A. of Computing Machinery. 2018. *Artifact review and badging*.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 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*. ACM, 1135–1144.

[14] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (2019), 206–215.

[15] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* (2019).
[16] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3319–3328.
[17] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

A CONTRIBUTION OF AUTHORS

Berend and Tom converted the original model to PyTorch implementation and are responsible for all the code and results. Contributions to the code can be seen in the commit history on Github. Furthermore, Tom and Berend wrote the Methods, Experimental setup, Results and Discussion section.

Cees and Stan wrote the introduction and broader implications sections.

B LENA IMAGE

Crops from the following figure were used as background.

Figure 7: The standard Lena image used for the background of our MNIST digits dataset

C HYPERPARAMETERS

Hyperparameter	Value
Learning rate	0.002
Optimizer	Adam
Batch size	250
Nr. Prototypes	15
Nr. Layers (decoder & encoder)	4
Kernel size convolutional layers	3×3
Stride	2

Table 5: Hyperparameter settings used for experiments

D EXAMPLES OF COLORED MNIST DATASET

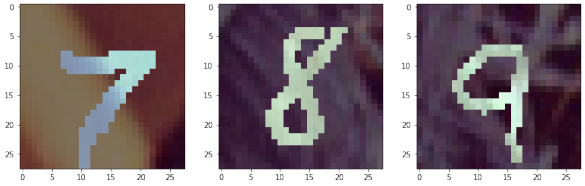


Figure 8: Examples of colored MNIST dataset

E PROTOTYPES FOR LAMBDA GRID SEARCH

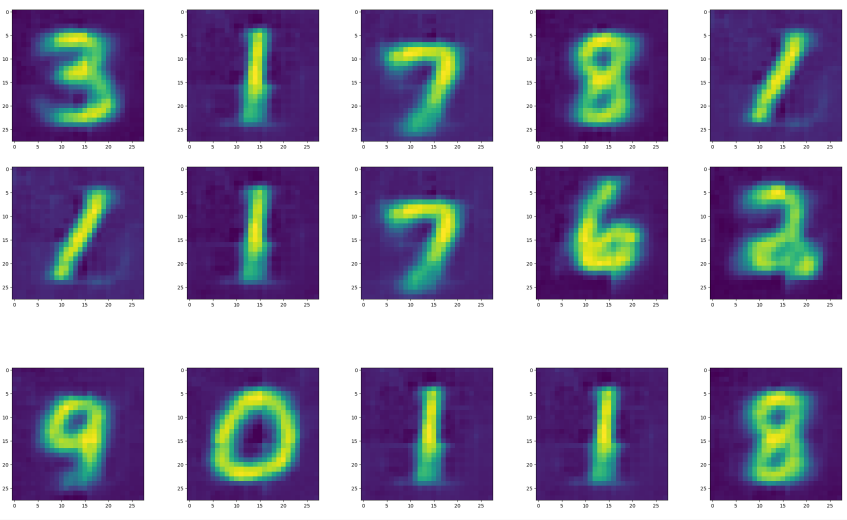
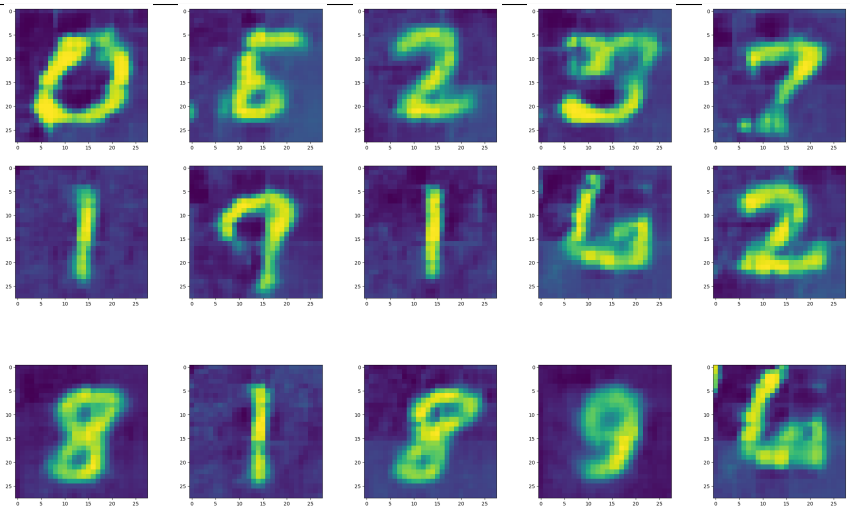
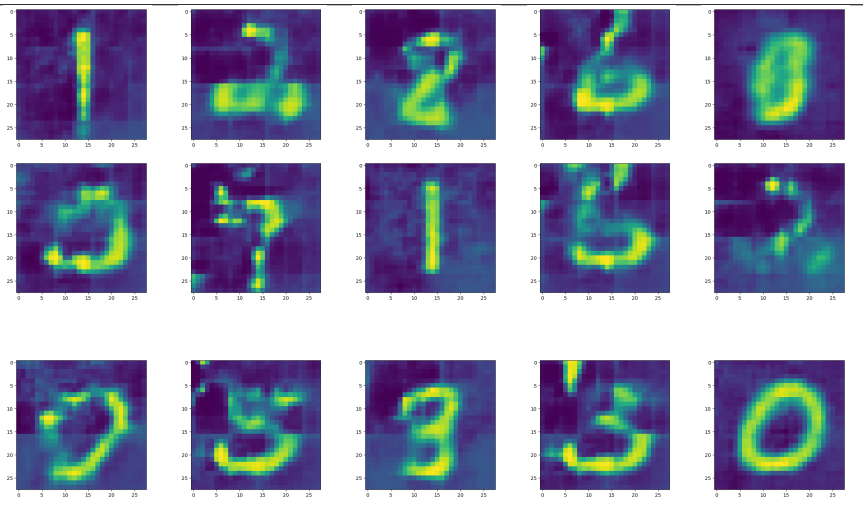
$\lambda_{class} = 20$ $\lambda_{ae} = 1$ $\lambda_1 = 10$ $\lambda_2 = 1$	
$\lambda_{class} = 20$ $\lambda_{ae} = 10$ $\lambda_1 = 1$ $\lambda_2 = 10$	
$\lambda_{class} = 20$ $\lambda_{ae} = 1$ $\lambda_1 = 10$ $\lambda_2 = 20$	

Table 6: Prototype examples for different lambda settings