Experimenting with "Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions"

Marius Arvinte, Mai Lee Chang, (Ethan) Yuqiang Heng
EE381V Minor Project
March 31, 2019

I. Introduction

In this report, we investigate the performance of the classification method in [1] on three different datasets. The original paper [1] introduces the notion of *prototype* vectors, used for classification in the trained latent space of an autoencoder, while, at the same time, being interpretable. The architecture of the classification network is shown in Figure 1.

The prototype layer outputs the Euclidean distances between its input and each trainable prototype vector, respectively, as

$$v(\mathbf{z}) = \left[\|\mathbf{z}-\mathbf{p}_1\|_2^2, \|\mathbf{z}-\mathbf{p}_2\|_2^2, \ldots, \left\|\mathbf{z}-\mathbf{p}_{N_p}\right\|_2^2\right],$$

where N_P is the number of prototype vectors and we let N_C denote the total number of classes.

At the same time, the interpretability of the prototype vectors is enforced by using two extra regularization terms, that force each prototype and input sample to have at least one close member from the other set, respectively, given by the expressions

$$R_1 = \lambda_1 \frac{1}{N_P} \sum_j \min_i \left\|\mathbf{p}_j - f(\mathbf{x}_i)\right\|_2^2,$$

$$R_2 = \lambda_2 \frac{1}{n} \sum_i \min_j \left\|\mathbf{p}_j - f(\mathbf{x}_i)\right\|_2^2.$$

Overall, the work in [1] leads to a classifier that exhibits both the high performance offered by a deep autoencoder, as well as an interpretable classification mechanism, based on distances to fixed examples in a latent space.

However, the original paper leaves some questions unanswered, such as: 1) How to select the number of prototypes for a given dataset?, 2) Does the method work for a dataset with a large number of classes? and 3) Does the method work for datasets with larger variance across the same class?

In this project, we attempt to answer the above questions by running and modifying the original source code on three datasets:

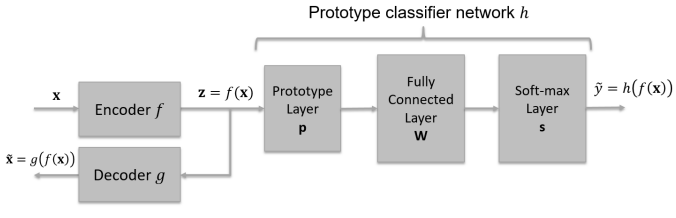

Fig. 1. The architecture of the autoencoder-based classification method in [1]. The data is encoded to the latent space and classified based on the set of distances w.r.t. the prototypes $\mathbf{p}_i$.

(i) MNIST dataset of handwritten digits [2]: We investigate the impact of reducing the number of prototypes and experimentally show that the method still classifies with good performance even when $N_P < N_C$.
(ii) EMNIST dataset of handwritten letters [3]: We investigate the performance of the method when $N_C = 47$.
(iii) CIFAR-10 dataset of natural images [4]: We investigate the performance of the method when the input images have large variance across members of each class, making classification harder, in general.

Our code, as well as extended simulation results are available online at https://github.com/mariusarvinte/FATML-MiniProject.

II. Dataset 1: MNIST

The first dataset we experiment on is the well-known MNIST [2], containing black-and-white handwritten images of the digits 0-9, with an image size of 28x28 pixels. Our goal is to investigate the impact of reducing the number of prototype vectors, since the original authors of [1] do not comment on how to choose this parameter. For this dataset, we keep the dimensions of the network and the latent space exactly as in the original paper, using four hidden layers each per encoder/decoder and prototype vectors belonging to $\mathbb{R}^{10}$. For this dataset, we do not change the dimensions of the hidden layers.

The original paper uses $N_P = 15$ prototype vectors for classification, achieving a validation set accuracy of approximately 99.2%. Our reproduced results show a similar accuracy of 99.34%, where we attribute the small increase in performance to a different initialization, noting that the original authors did not seed their experiments.

Next, we decrease the number of prototypes and investigate the impact this has on the classification performance, as well as interpretability of them. A reasonable expectation is that classification and interpretability will break down when the number of prototypes becomes less than the number of classes. We experimentally show that, for MNIST, this is **not** true and that high-accuracy, interpretable classification is possible with a low number of prototype vectors.

Figure 2 shows the validation accuracy as a function of the number of prototypes. We can clearly notice that even for $N_P = 7$, the neural network still retains an accuracy greater than 99% and, interestingly, a drastic performance loss only occurs when the number of prototype vectors is exactly 1, where the performance is equal to that of random guessing. This can be explained by the fact that the MNIST images are inherently very suited for clustering in latent spaces (e.g., as

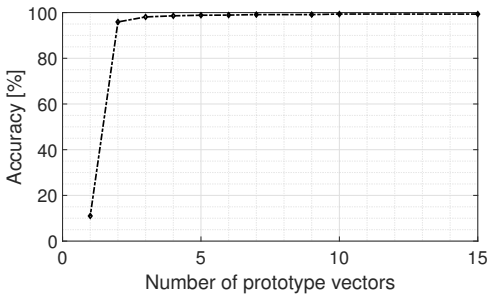

Fig. 2. Validation accuracy versus number of prototypes for the MNIST dataset.

TABLE I
The weights of the last layer for MNIST, $N_P = 7$. Rows represent the prototype vectors and columns the predicted classes. A smaller weight indicates a larger contribution of the distance to the prototype for the class.

	0	1	2	3	4	5	6	7	8	9
	**-14.40**	1.92	-1.09	3.71	1.12	1.93	-1.19	3.03	3.30	-1.88
	0.36	0.18	**-10.82**	**-3.86**	1.89	5.86	2.26	0.08	**-4.66**	5.33
	2.36	1.19	2.57	**-10.41**	7.91	-6.18	1.93	2.85	-1.18	-3.20
	-0.27	2.40	3.53	4.18	-1.90	**-4.92**	**-11.73**	1.58	-3.15	5.00
	2.43	**-11.23**	0.35	0.63	**-4.39**	1.84	-1.42	1.41	5.97	1.10
	4.46	2.40	2.95	2.05	**-7.45**	3.22	3.38	2.25	**-3.76**	**-9.43**
	1.58	2.38	1.74	1.44	-1.58	-1.42	4.33	**-13.61**	4.17	-1.74

proven by the VAE approach [5]), thus once a small number of prototypes are learned, the encoder can map the MNIST clusters by using the prototype vector as anchor points, since classification is done based on the Euclidean distance.

Table I shows the decoded prototypes for $N_P = 7$ on the row axis, where we can see that interpretability is preserved, but some digits are clearly missing, while still being correctly classified according to the overall performance. This is further explained by inspecting the weights of the linear, fully-connected layer used for classification. Table I contains these weights, where each line corresponds to a prototype image and each column to the predicted classes 0-9. For example, while a prototype for the digit 8 is missing it, it is correctly classified using distances from the prototypes for 2, 6 and 9, since it can be interpreted as a combination of parts of them.

III. Dataset 2: EMNIST

The Extended MNIST (EMNIST) dataset contains handwritten characters including both digits and letters derived from the NIST Special Database 19 [3], [6]. We used the balanced version of this dataset, where the number of samples for each class is the same. Although this does not reflect the frequency of each character in human language, a balanced dataset is helpful for classification purposes to reduce class bias. The dataset contains $131,600$ 28×28 images from 47 classes (10 digits and 37 letters). Both uppercase and lowercase letters are included in the dataset, but lowercase letters that look the same as their corresponding uppercase letters are removed

Fig. 3. Reconstruction results with N_P = 47 on EMNIST data. The first row is images from the training set and the second row their corresponding reconstruction.

(e.g. c for C, o for O). The EMNIST dataset is created to be directly compatible with the MNIST dataset, reproducing the conversion and processing steps used in MNIST. Hence, EMNIST provides a good comparison with MNIST when the number of classes is much larger. We use 112,800 samples as training data and 18,800 samples as testing data. We use 10% of the training samples as validation data for our models.

We first trained the model with 60 prototype vectors. We used the same hidden dimensions of $(32, 32, 32, 10)$ in the 4 layers for both encoder and decoder as in the original paper. While we achieved a testing accuracy of 85.08%, the autoencoder error decreases slowly after 200 epochs and is still significant at 5.09 after 1500 epochs. This is likely due to the small dimensions of the hidden layers, not being able to capture features in a more complex dataset. We increase the hidden dimensions to $(256, 128, 64, 32)$, and as shown in Table II, we reduce the autoencoder error from 7.31 to 2.34 with 60 prototypes. However, there isn't a significant increase in training, validation and test accuracy (84.6% to 86.5%).

We also vary the number of prototype vectors to observe its impact on classification accuracy and autoencoder performance. We trained the model with N_P = 60, 47, 20 and 10 for 200 epochs each. In all cases the model converges before 200 epochs. The train, validation and test accuracies as well as the autoencoder errors for all configurations are presented in Table II. We can observe that decreasing the number of prototypes from 60 to 10 does not reduce either testing accuracy or autoencoder error (86.5% to 86.2%). This is interesting since N_P = 10 is significantly smaller than the number of classes in EMNIST (47 classes). This shows that like in MNIST, EMNIST images are also suited for clustering in latent spaces.

Figure 3 shows the reconstruction of 10 random samples in the EMNIST dataset when we trained the model with N_P = 47. We can observe that the autoencoder can faithfully reconstruct the original characters and that, as expected, the reconstruction quality does not vary significantly with the choice of N_P. This is verified by the low AE error across different N_P in Table II.

Figure 4 shows 20 of the 60 prototypes obtained from our model trained on EMNIST data with N_P = 60. We can observe that similar to the MNIST data, some of the prototypes resemble particular characters. Unlike in the MNIST data, there are prototypes that do not resemble any one character but instead look like combination of several characters. Comparing to the prototypes obtained when we use N_P = 10, there is no obvious difference in the interpretability of prototype images. This demonstrates a potential limitation of this method: in classification problems with a large number of classes, the prototypes obtained might not be very interpretable. Although

TABLE II
EMNIST dataset: Training results for various parameters. Validation and testing accuracies are both measured at the end of training.

Hidden layer dimensions	N_P	Train accuracy %	Validation accuracy %	Test accuracy %	AE error
32, 32, 32, 10	60	83.2	84.8	84.6	6.86
256, 128, 64, 32	60	86.9	86.5	86.5	2.34
256, 128, 64, 32	47	87.1	86.5	86.4	2.22
256, 128, 64, 32	20	87.1	86.0	86.1	2.38
256, 128, 64, 32	10	86.3	86.3	86.2	2.01

Fig. 4. Twenty examples of prototypes for EMNIST data, with $N_P = 60$.

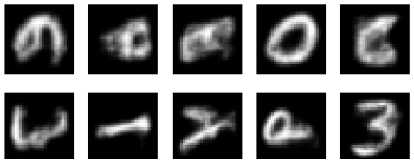

Fig. 5. Ten prototypes of EMNIST data, with $N_P = 10$.

EMNIST has significantly more classes than MNIST, its characters are still relatively simple and comparable to the digits in MNIST. While we can reconstruct explainable prototypes from the latent space of training images, the reconstructions might not be explainable when the number of classes increases. Table III shows the contribution of prototypes to a subset of the classes. A few classes such as **0** and **O** have a single contributing prototype that looks like both classes and is interpretable. However, most classes have several equally contributing prototypes, which themselves look like combinations of characters and are not interpretable.

IV. Dataset 3: CIFAR-10

The last dataset that we experiment with is the CIFAR-10 dataset [4]. It consists of 60,000, 32x32 colour natural images, with a balanced split across 10 classes. There are 50,000 training images and 10,000 test images. We use this dataset because it is more challenging than MNIST and EMNIST, since it consists of natural images. Our approach consists of varying the following parameters: 1) N_P, 2) hidden layer dimensions, 3) black-and-white vs. color, 4) elastic deformation and 5) weight initialization. Table IV provides a summary of the results.

Initially, we convert the data to black-and-white using the formula recommended by ITU [7] for colour-to-grayscale conversion:

TABLE III
The weights of the last layer used for classification based on distances for $N_P = 10$ on EMNIST. The rows represent the prototype vectors and the columns the predicted classes. A smaller weight indicates a larger contribution of the distance to the prototype for that particular class.

	0	5	8	A	B	E	M	O	S	V	a	h
0	0.551	-0.783	2.91	-0.074	**-1.18**	**-3.82**	**-1.39**	-0.440	-0.205	2.19	1.05	0.807
5	1.23	1.59	0.681	-3.16	-0.693	0.0844	0.249	1.64	0.588	3.10	**-1.39**	**-4.24**
8	0.527	**-1.30**	**-3.49**	-0.440	-0.369	-0.640	5.28	0.599	**-1.67**	0.628	1.08	1.61
A	**-5.36**	**-1.27**	0.687	1.25	1.15	1.91	1.83	**-5.23**	**-2.59**	**-1.38**	**-2.36**	1.56
B	0.955	4.02	3.10	**-2.31**	1.05	3.29	**-4.49**	0.0845	3.80	-0.618	**-1.14**	**-2.74**
E	-0.154	-1.54	0.361	-0.070	**-2.88**	2.09	1.86	0.140	-1.01	1.17	1.35	1.82
M	1.09	-0.02	-0.562	4.70	2.17	**-2.56**	3.46	1.76	**-1.74**	-0.999	1.79	-0.328
O	1.41	**-1.80**	1.207	0.693	2.43	0.650	-0.749	1.17	2.22	**-3.27**	3.01	0.741
S	1.50	1.98	-0.598	-0.262	2.90	0.394	-0.770	1.28	0.194	2.48	**-1.90**	1.05
V	-0.24	0.427	**-2.18**	1.05	**-2.08**	-0.310	**-1.06**	0.584	1.20	-0.664	1.57	1.48

Fig. 6. Ten examples of the CIFAR-10 autoencoder reconstruction when we use grayscale conversion and $N_P = 15$. Validation accuracy of 58.6%.

$$0.3 \times R + 0.59 \times G + 0.11 \times B \tag{1}$$

Using grayscale, a number of $N_P = 15$ prototypes and training for 1,500 epochs, the model achieves an accuracy of 60.0% on the training set, 58.7% on the validation set, and 57.9% on the test set. Examples of reconstructed images and learned prototypes are shown in Figures 6 and 7, respectively.

Next, we increase the number of prototypes to 30 and also increase training duration from 1,500 to 3,000 epochs. However, we this does not improve the mode's performance (training accuracy of 58.7% and validation accuracy of 59.0%). We noticed that the accuracy increases to the maximum value after training for as little as 30 epochs, after which we enter an overfitting regime where training accuracy increases, but validation accuracy decreases. Thus, we reduce the number of training epochs for the rest of the experiments. Even with N_P = 60, after training for 300 epochs, the model's validation accuracy still did not significantly improve.

TABLE IV
CIFAR-10 dataset: Training results for various parameters. Note that the lower test accuracy comes from evaluating at the end of training, while we report the best achieved validation accuracy during training.

	Hidden layer dimensions	N_P	Train accuracy %	Validation accuracy %	Test accuracy %	AE error
Grayscale, elastic, 1500 epochs	32, 32, 32, 10	15	60.0	58.3	57.9	13.1
Grayscale, elastic, 3000 epochs	32, 32, 32, 10	30	58.7	59.0	57.9	13.3
Grayscale, elastic, 3000 epochs	32, 32, 32, 10	60	80.5	59.3	56.5	23.4
Grayscale, no elastic	256, 128, 64, 32	30	71.8	62.2	53.1	7.4
Color, no elastic, multiple restarts	256, 128, 64, 32	60	73.2	65.0	54.7	25.5

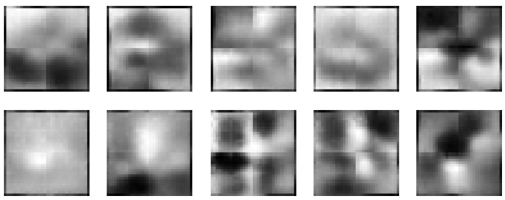

Fig. 7. Ten examples of learned prototypes for CIFAR-10 when we use grayscale conversion and $N_P = 15$. Validation accuracy of 58.6%.

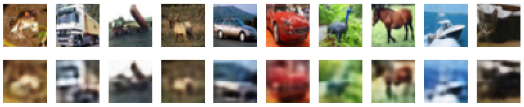

Fig. 8. Ten examples of the CIFAR-10 autoencoder reconstruction from the model that achieved the highest validation accuracy (65.0%).

As a solution, we increase the encoder hidden layer output dimensions to $(256, 128, 64, 32)$ (mirrored for the decoder) based on work showing that these sizes are suitable for natural images of this size [8]. Additionally, we also remove the elastic deformation since the natural images are already a challenge so additional data augmentation may prove harmful. As a result, the validation accuracy increases to 62.2%.

Finally, we revert to using color images and experiment with the random weight initializations to find an initialization that maximizes the validation accuracy. As part of this process, we added seeding in the code in order to reproduce our results. We trained the model with $N_P = 60$, $(256, 128, 64, 32)$ hidden dimensions, colour images and removed elastic deformation, for 60 epochs with 270 different random weight initializations. The best achieved training accuracy is 73.2% and best validation accuracy is 65.0%. Figure 8 shows the reconstruction results and Figure 9 shows the learned prototypes for the network that has the best validation accuracy. The reconstruction results are better compared to the baseline; however, the prototypes are still not interpretable. We conclude that a limitation of this approach is that it is not robust to natural images. During this project, we found that the original authors also acknowledge this limitation in their follow-up work [9].

V. Conclusions

We have used the interpretable prototype method in [1] on three different datasets, each with its particular challenges. Our main finding from MNIST and EMNIST is that the

Fig. 9. Ten examples of the learned CIFAR-10 prototypes from the model that achieved the highest validation accuracy (65.0%).

performance of classification can be retained by using a very small number of prototypes compared to the number of classes (e.g., 10 prototypes for 47 classes). Furthermore, for MNIST, there is no major impact on interpretability, since the learned prototypes still look like natural digits when decoded. That is not the case for EMNIST, where we lose 1-to-1 interpretability and the prototypes look like combinations of characters. The main finding from CIFAR-10 is that the approach is not suitable for natural images as the prototypes images are not interpretable, a fact also pointed out in the the authors' subsequent work [9].

From a data characterization viewpoint, we can attribute the favorable results for MNIST and EMNIST to their larger amenability to clustering. Thus, a good idea for future research is to come up with an interpretable, deep classifier, that provides explanations even for heterogenous data.

References

[1] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[2] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[3] NIST, "The EMNIST dataset," https://www.nist.gov/node/1298471/emnist-dataset.

[4] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[5] D. P. Kingma and M. Welling, "Autoencoding variational Bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[6] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *CoRR*, vol. abs/1702.05373, 2017.

[7] R. I.-R. BT, "Studio encoding parameters of digital television for standard 4: 3 and wide-screen 16: 9 aspect ratios," 1995.

[8] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval." in *ESANN*, 2011.

[9] C. Chen, O. Li, A. Barnett, J. Su, and C. Rudin, "This looks like that: deep learning for interpretable image recognition," *arXiv preprint arXiv:1806.10574*, 2018.