# Inverse distance weighting attention

**Calvin McCarter**
mccarter.calvin@gmail.com

## Abstract

We report the effects of replacing the scaled dot-product (within softmax) attention with the negative-log of Euclidean distance. This form of attention simplifies to inverse distance weighting interpolation. Used in simple one hidden layer networks and trained with vanilla cross-entropy loss on classification problems, it tends to produce a "key" matrix containing prototypes and a "value" matrix with corresponding logits. We also show that the resulting interpretable networks can be augmented with manually-constructed prototypes to perform low-impact handling of special cases.

## 1 Introduction

A key question in both machine learning and computational neuroscience concerns the relationship between supervised learning (success in predictive tasks) and associative memory (forming representations for previous experiences which can be cued by similar new experiences). On the one side, models of associative memory [Hopfield, 1982, Krotov and Hopfield, 2016] have relied on energy functions which are explicitly designed to teach the network to form memories. On the other side, nearest-neighbor methods for supervised learning explicitly store the training data, which are then retrieved and weighted according to some distance metric. In contrast, standard neural networks trained with standard supervision (or self-supervision) do not tend to have network parameters with explicitly encoded memories. Nevertheless, popular deep learning models, such as attention-based Transformers and diffusion models, are implicitly trained to behave similarly to associative memories [Bricken and Pehlevan, 2021, Ambrogioni, 2023, Hoover et al., 2023].

Here, we elucidate further the connection between standard neural networks with attention and associative memory networks, by examining the learned parameters of a single-hidden-layer network trained via standard classification cross-entropy loss. (Notably, dense associative memory networks [Krotov and Hopfield, 2016] can also interpreted as single-hidden-layer networks.) After modifying the standard scaled dot-product score to the negative-log of Euclidean distance, we observe that the trained key matrix contains explicit memories of representative inputs. This negative-log distance score also leads to a weighting of prototypes that corresponds to Shepard's method [Shepard, 1968]for interpolation. We further show that adding (key, value) pairs of prototypes to trained one-hidden-layer networks can be used to perform low-impact behavior modification.

## 2 Methods

### 2.1 Inverse distance weighting attention

The widely-adopted attention mechanism is closely related to associative memory, with ($\boldsymbol{k}^{(i)}$-key, $\boldsymbol{v}^{(i)}$-value) lookups weighted by the softmax operator applied to similarity scores between $d$-dimensional $\boldsymbol{q}$ query and keys $\boldsymbol{k}^{(i)}$. Here we consider the classification setting with a single hidden layer, so that each $\boldsymbol{v}^{(i)} \in \mathbb{R}^C$ value vector encodes corresponding learned logits for $C$ classes. For each $i$th

Associative Memory & Hopfield Networks in 2023. NeurIPS 2023 workshop.

key-value pair, the attention mechanism can be written as,

$$\text{attention}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V})_i = \text{softmax}(\text{score}(\boldsymbol{q}, \boldsymbol{k}^{(i)}))\boldsymbol{v}^{(i)},$$
$$\text{softmax}(\text{score}(\boldsymbol{q}, \boldsymbol{k}^{(i)})) = \frac{\exp\left(\text{score}(\boldsymbol{q}, \boldsymbol{k}^{(i)})\right)}{\sum_j \exp\left(\text{score}(\boldsymbol{q}, \boldsymbol{k}^{(j)})\right)},$$

for $i \in \{1, \ldots, P\}$ for $P$ prototypes. Various forms of attention fall into this framework, including cosine attention with $\text{score}(\boldsymbol{q}, \boldsymbol{k}^{(i)}) = \cos(\boldsymbol{q}, \boldsymbol{k}^{(i)})$ [Graves et al., 2014], additive attention with $\text{score}(\boldsymbol{q}, \boldsymbol{k}^{(i)}; \boldsymbol{v}, \boldsymbol{W}) = \boldsymbol{v}^\top \boldsymbol{W}[\boldsymbol{q}; \boldsymbol{k}^{(i)}]$ [Bahdanau et al., 2014], and scaled dot product attention with $\text{score}(\boldsymbol{q}, \boldsymbol{k}^{(i)}) = \boldsymbol{q}^\top \boldsymbol{k}^{(i)}/\sqrt{d}$ [Vaswani et al., 2017].

In this work, we consider the Euclidean distance, which is negatively related to the dot product via the equality $\|\boldsymbol{q}-\boldsymbol{k}\|_2^2 = \|\boldsymbol{q}\|_2^2 + \|\boldsymbol{k}\|_2^2 - \boldsymbol{q}^\top \boldsymbol{k}$. Despite this seemingly simple and well-known relationship, we will see that using the Euclidean distance can produce substantially different parameters and different behavior. This arises from the fact that, while there exist order-preserving transformations between the Euclidean distance and the inner product, these transformations are not trivial; for each direction, the transformation involves adding one additional dimension [Bachrach et al., 2014]. For example, the Euclidean distance can be implemented in terms of the inner product by concatenating the constant 1 to the query vector; and for each key vector, the squared-norm of the original key vector.

Furthermore, because Euclidean distance measures dissimilarity, it needs to be massaged into use as a similarity score. In particular, we desire a scoring function that simultaneously (1) achieves good accuracy when trained using a standard supervised classification loss, and (2) learns keys that are prototypes of the original data. We begin by noting that these are not simultaneously achieved using the negative distance, using it within the Gaussian kernel, and using its inverse, as defined below, respectively:

$$\text{score}_{\text{neg}}(\boldsymbol{q}, \boldsymbol{k}) = -\|\boldsymbol{q} - \boldsymbol{k}\|_2^2 \qquad \text{(negative (squared) Euclidean distance)} \quad (1)$$
$$\text{score}_{\text{Gauss}}(\boldsymbol{q}, \boldsymbol{k}; \sigma) = \exp(-\|\boldsymbol{q} - \boldsymbol{k}\|_2^2/\sigma^2) \qquad \text{(Gaussian kernel Euclidean distance)} \quad (2)$$
$$\text{score}_{\text{inv}}(\boldsymbol{q}, \boldsymbol{k}; p, \epsilon) = \frac{1}{\epsilon + \|\boldsymbol{q} - \boldsymbol{k}\|_2^p}. \qquad \text{(inverse Euclidean distance)} \quad (3)$$

The $\epsilon > 0$ parameter prevents division by zero, while the power parameter $p > 0$ controls how strongly the influence of a key falls away with its distance from the query. However, we observe that, when used in concert with cross-entropy classification loss and backpropagation to compute gradients, we fail to achieve both desiderata on even simple problems. Notably, while the inverse distance score leads to better classification accuracy than the other scoring functions on the Two Moons classification problem, we witnessed "clumping" of wasted prototypes that are not pushed away from each other, so long as they are not too close to training examples of the wrong class. This appears to be caused by a vanishing gradients problem, as the inverse function flattens out for large-distance inputs.

To address the distant vanishing gradients problem with inverse distance, we replace the inverse function with the negative-log function, whose derivative vanishes more slowly as its argument goes to positive infinity:

$$\text{score}_{\text{neglog}}(\boldsymbol{q}, \boldsymbol{k}; p, \epsilon) = -\log\left(\epsilon + \|\boldsymbol{q} - \boldsymbol{k}\|_2^p\right) \qquad \text{(negative log Euclidean distance)}. \quad (4)$$

Used within the softmax operation, this simplifies to the following:

$$\text{attention}_{IDW}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V})_i = \frac{\frac{1}{\epsilon + \|\boldsymbol{q} - \boldsymbol{k}^{(i)}\|_2^p}}{\sum_j \frac{1}{\epsilon + \|\boldsymbol{q} - \boldsymbol{k}^{(j)}\|_2^p}} \boldsymbol{v}^{(i)}. \quad (5)$$

We dub this *inverse distance weighting* (IDW) attention, as it coincides with the IDW function employed by Shepard [1968] for numerical interpolation of irregularly-spaced points. When $\epsilon \to 0, p \to \infty$, the IDW weighting function approaches the Voronoi diagram [Shepard, 1968], making this equivalent to a 1-nearest-key classifier. When $p = 2, \epsilon = 1$, IDW attention has similarities that come from the Student t-distribution with one degree of freedom, the same similarity metric used for $t$-SNE [Van der Maaten and Hinton, 2008] embeddings. We will choose small $\epsilon < 1$, which has also been shown to succeed for $t$-SNE visualization [Kobak et al., 2019].

We illustrate the different distance-based attention scores in Figure 1. We depict the weight given to one of two keys as a function of its distance, when the distance of the second key is 1. Only inverse distance softmax and negative-log softmax (i.e. IDW) functions give attention approaching 1 as the key approaches the query. Meanwhile, only IDW avoids the vanishing gradient problem for both near and far distances.

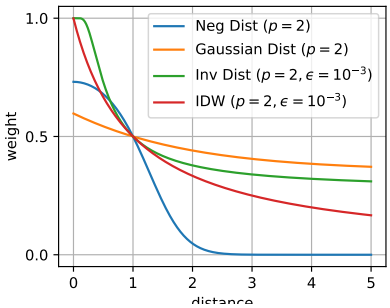


Figure 1: We depict the weight given to an example as a function of its distance. Because the weights of the two prototypes sum to 1, all scoring functions give a weight of 0.5 when the distance is 1.

### 2.2 Low-impact "special case" handling with (key, value) augmentation

In real-world settings, it is frequently the case that machine learning models need to incorporate special behavior for certain inputs. Handling such special cases would typically require explicit handling via code that is run either before or after model inference, or via modified model training / fine-tuning. However, if a model is represented in terms of prototypes that exist in the same space as inputs, behavior for special cases can be controlled transparently. This is especially easy for IDW, because the influence of a prototype decays sharply with distance, so long as $\epsilon$ is sufficiently small. Consider an input $\boldsymbol{q}$ for which we want to predict class $c \in \{1, \ldots, C\}$. If this is not already the case, then $\arg\max \sigma(\boldsymbol{d})\boldsymbol{V} \neq c$, where

$$\sigma(\boldsymbol{d})_i = \frac{\left(\epsilon + \boldsymbol{d}_i^p\right)^{-1}}{\sum_j \left[(\epsilon + \boldsymbol{d}_j^p)^{-1}\right]}, \tag{6}$$

and $\boldsymbol{d}_i$ is the Euclidean distance from $\boldsymbol{q}$ to the $i$th prototype. We change the behavior by adding new prototype with $\boldsymbol{k}' := \boldsymbol{q}$ and $\boldsymbol{v}' := \eta \boldsymbol{e}_c$. We make $\eta$ as small as possible while still fixing the model's behavior for the given input, thus minimizing behavior disruption for the rest of the input space. This is accomplished by choosing

$$\eta := \left(1 + \epsilon \sum_j \frac{1}{\epsilon + \boldsymbol{d}_j^p}\right)\left[\left(\max_{k \neq c} \sigma([\boldsymbol{d}; \epsilon])_{1:P} \boldsymbol{V}_{:,k}\right) - \sigma([\boldsymbol{d}; \epsilon])_{1:P} \boldsymbol{V}_{:,c}\right]. \tag{7}$$

## 3 Experiments

### 3.1 Two Moons synthetic data

We first train and depict single-hidden-layer networks on the standard Two Moons classification setting, shown in Figure 2. In addition to the various distance-based attention mechanisms, we also show the results for fully-connected with ReLU nonlinearity, as well as scaled-dot-product attention. For each method, we independently train 3 networks with 2, 16, and 128 prototypes (which is the same as the number of hidden activations). Only for IDW do the keys roughly recapitulate the input data distribution. Given that this is the sort of problem where nearest-neighbor perform well, it is also unsurprising that IDW has good performance, with the best test accuracy for 16 and 128 prototypes. We also show results for low-impact special case handling with IDW on Two Moons in Figure 3. We see that, for each of the IDW networks, modifying the behavior for an input barely changes its behavior, regardless of the desired label of that input (either class label 0 or 1).

### 3.2 MNIST data

We next trained networks on MNIST with 20 prototypes. The test accuracies are provided in Table 1. The IDW network had a test accuracy of 88%. While the IDW model had worse accuracy than the FC-Relu and scaled dot-product models, it had substantially better test accuracy than the other Euclidean distance-based forms of attention. Furthermore, among all the methods, only IDW has key parameters resembling digits, as depicted in Figure 4.

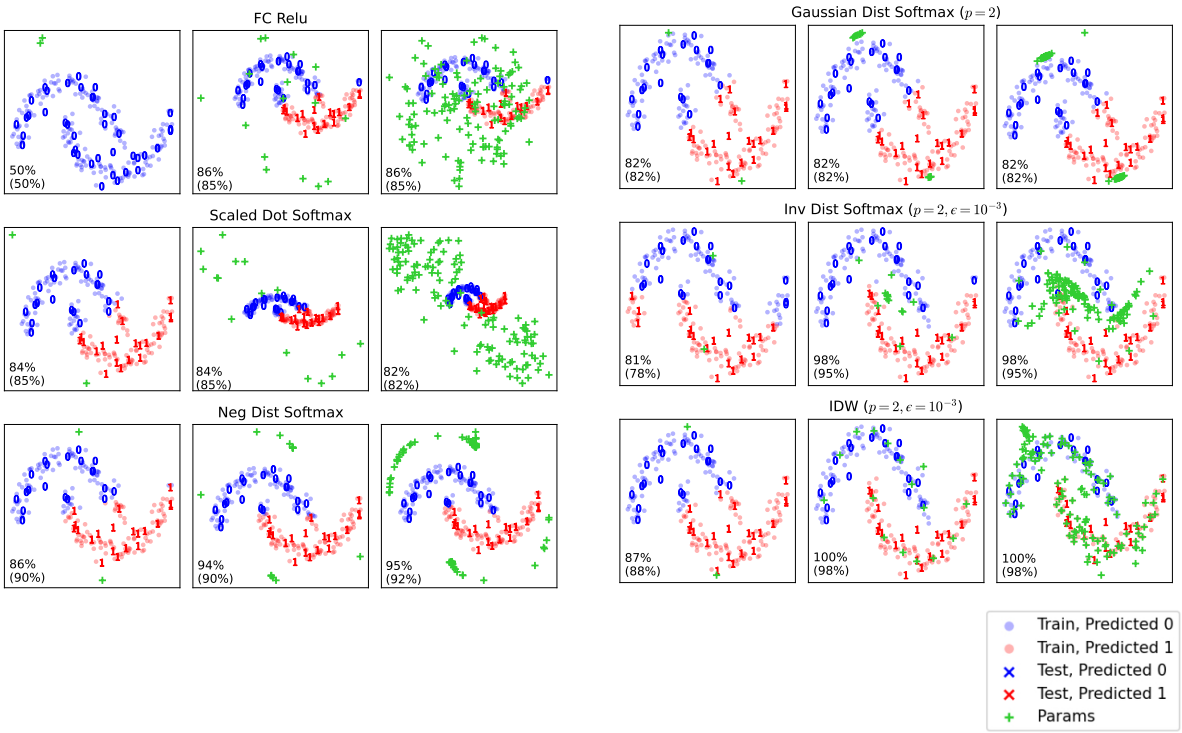


Figure 2: Results for Two Moons classification. For each method, we depict the training data, the test data, as well as the 2D parameters of the first weight matrix. We also show the train (and test) accuracy. For each method, there are 3 subplots, corresponding to 2, 16, and 128 prototypes.

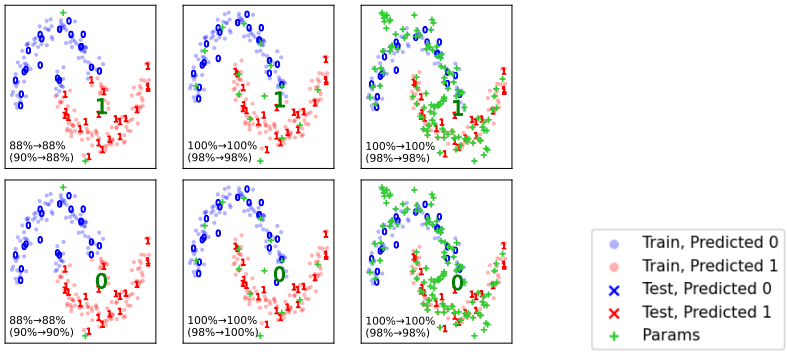


Figure 3: Low-impact behavior modification. In each subplot, we depict the desired label of the "special case" input with a large green "0" or "1". We also show the before→after train (and test) accuracy.

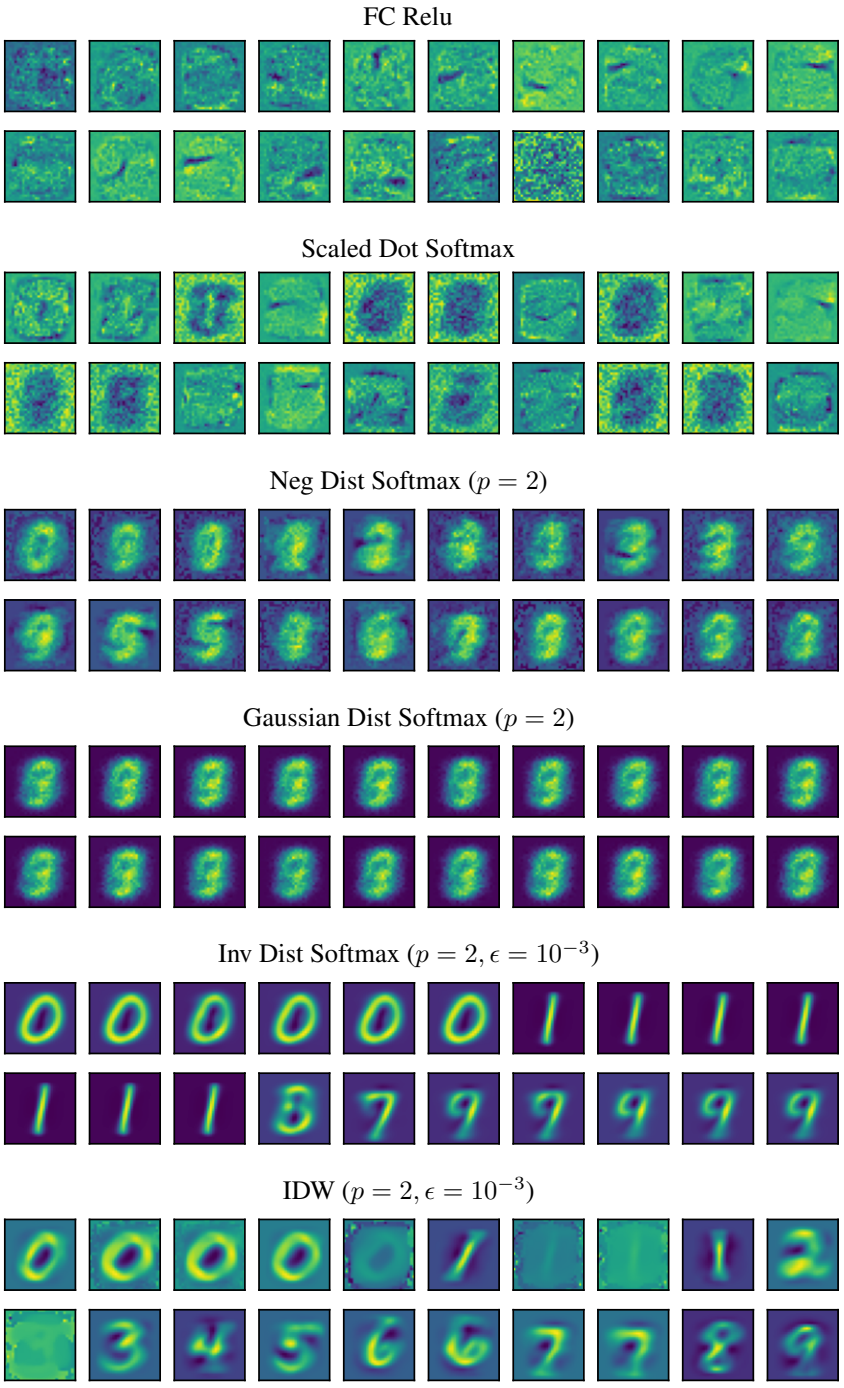


Figure 4: Learned keys after training single-hidden layer networks on MNIST. Keys are sorted by the argmax of their corresponding values.

In Table 1, we compare the accuracy of all methods on MNIST with 20 prototypes. We see that IDW performs worse than FC Relu and scaled dot-product attention, but performs better than the other distance-based forms of attention.

Table 1: Comparison of test accuracy on MNIST dataset.

| **Method** | **Test Accuracy** |
|---|---|
| FC Relu | **95.99%** |
| Scaled Dot Softmax | 93.15% |
| Neg Dist Softmax | 83.63% |
| Gaussian Dist Softmax ($p = 2$) | 11.35% |
| Inv Dist Softmax ($p = 2, \epsilon = 10^{-3}$) | 11.35% |
| IDW ($p = 2, \epsilon = 10^{-3}$) | 88.20% |

The Appendix contains further details on experimental setup and further analysis on the effects of hyperparameters. Code is available at `https://github.com/calvinmccarter/idw-attention`.

## 4 Conclusions

We have reported how a specific form of distance-based attention leads to formation of prototypes in a single-hidden-layer network trained with vanilla cross-entropy loss. It remains to be seen what theoretical and practical implications this phenomena has for deep networks, as well as for elucidating the set of sufficient and necessary conditions for formation of associative memories.

## References


Luca Ambrogioni. In search of dispersed memories: Generative diffusion models are associative memory networks. *arXiv preprint arXiv:2309.17290*, 2023.

Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 257–264, 2014.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Trenton Bricken and Cengiz Pehlevan. Attention approximates sparse distributed memory. *Advances in Neural Information Processing Systems*, 34:15301–15315, 2021.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Benjamin Hoover, Hendrik Strobelt, Dmitry Krotov, Judy Hoffman, Zsolt Kira, and Duen Horng Chau. Memory in plain sight: A survey of the uncanny resemblances between diffusion models and associative memories, 2023.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Dmitry Kobak, George Linderman, Stefan Steinerberger, Yuval Kluger, and Philipp Berens. Heavy-tailed kernels reveal a finer cluster structure in t-sne visualisations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 124–139. Springer, 2019.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, 1968.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

# A Appendix

## A.1 Details on experimental setup

For Two Moons problem, we generated 100 training examples and 20 test examples. We trained models with a batch size of 10, a learning rate $0.01$, 25 epochs, and the AMSGrad [Reddi et al., 2019] variant of Adam [Kingma and Ba, 2014] with cosine annealing [Loshchilov and Hutter, 2016]. We randomly initialized the keys being normally distributed with mean set to the corresponding mean of those pixels in the training data, and standard deviation as 0.1 times the observed standard deviation in the training data. We initialized the values to all-0s.

On MNIST, we used a batch size of 4, a learning rate of $0.001$, 50 epochs, and the AMSGrad [Reddi et al., 2019] variant of Adam [Kingma and Ba, 2014] with cosine annealing [Loshchilov and Hutter, 2016]. No data augmentions were used. As before, we used IDW with $p = 2, \epsilon = 1e - 3$, and initialized keys and values as described above for Two Moons.

### A.1.1 Effect of power and $\epsilon$ parameters on Two Moons

In Figure 5 we show results for a variety of settings of $p$ and $\epsilon$. Interestingly, we observe that $p = 1$ damages accuracy, while $p >>> 2$ retains accuracy but damages the formation of prototypes.

## A.2 More experiments on low-impact special case handling for Two Moons

In Figure 6, we depict the results for other special cases. We see that as long as there were 16 or more prototypes, handling special cases did not damage network performance, even when the special case had desired behavior very different from the surrounding samples.

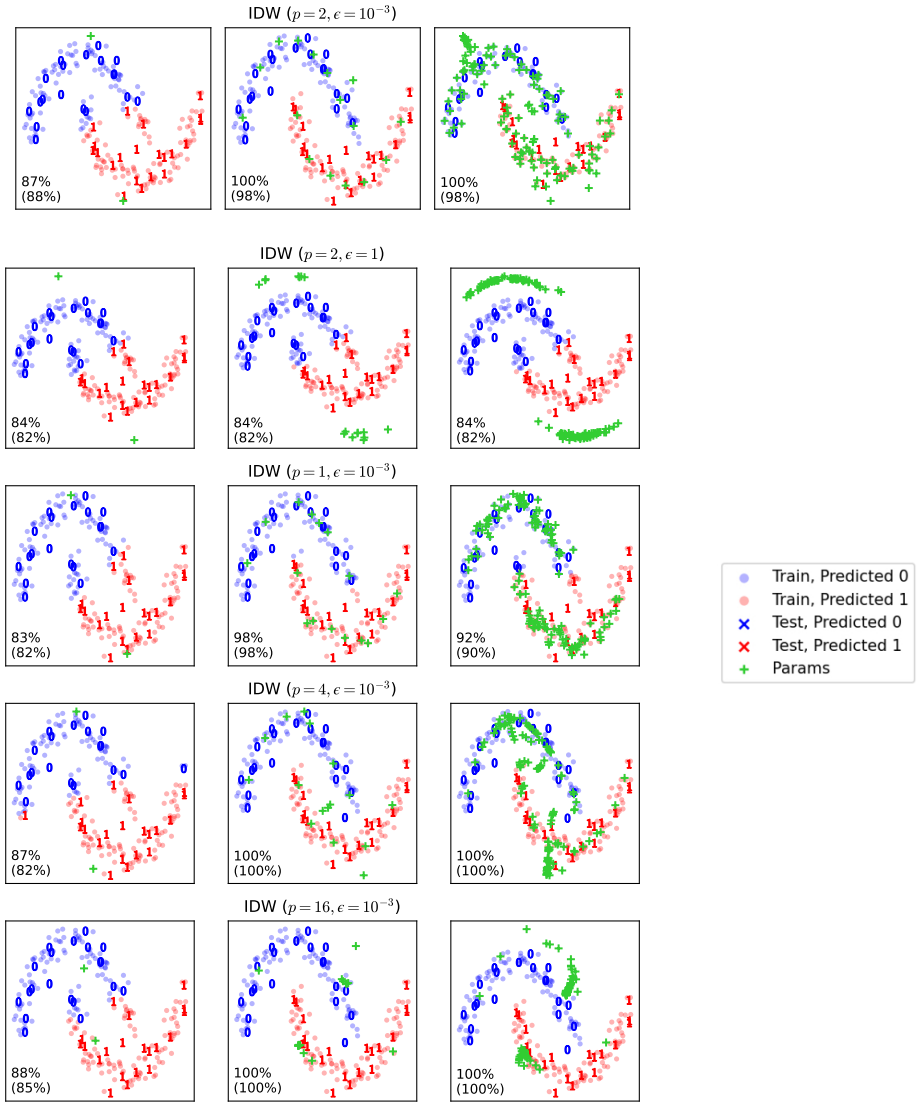


Figure 5: Results for various choices of IDW parameter settings on the Two Moons dataset.

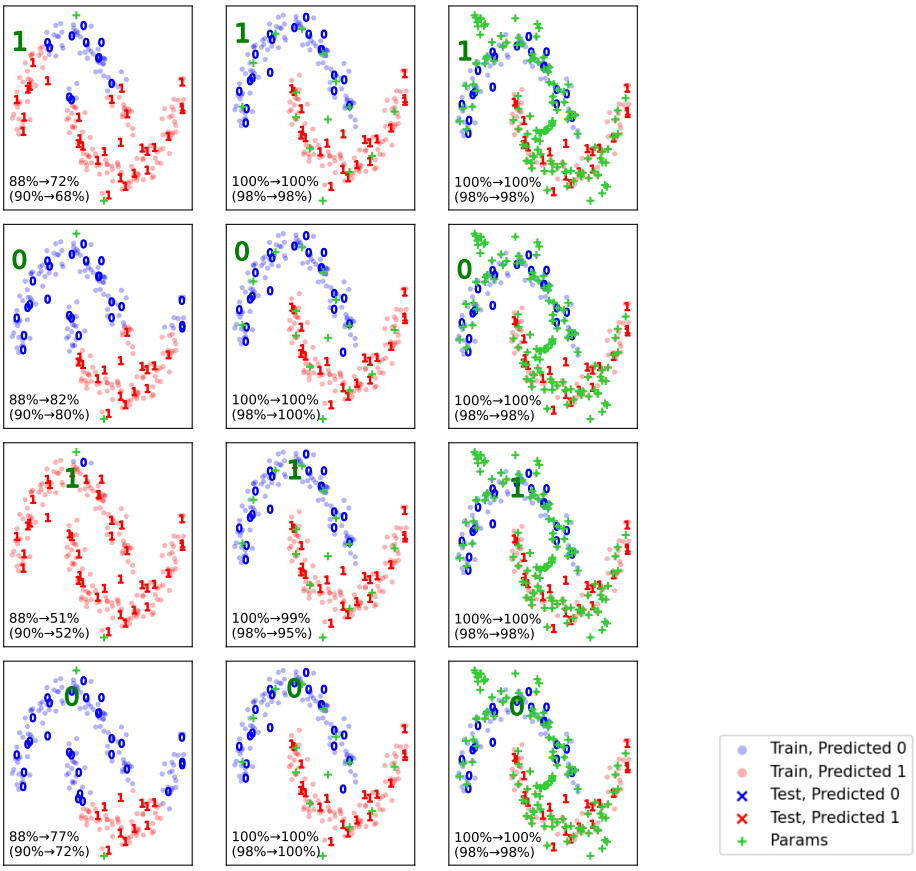


Figure 6: Results for other special cases, using IDW special case behavior modification.