Interpretable Learning with Distance Aware Radial Basis Function Networks

Ethan Cramer
Lone Star Analysis
Addison, TX
ecramer@lone-star.com

Randal Allen
Lone Star Analysis
Orlando, FL
rallen@lone-star.com

ABSTRACT

A rapid increase in available computing power has ushered in an era of neural network dominance in tasks such as image classification and natural language processing, yet the standard approach to training neural networks, minimizing prediction loss, leaves them ignorant of their uncertainties and vulnerable to adversarial attacks. Due to their generalization mechanism, deep neural networks often assign high confidence to regions far from the decision boundary. These models lack the ability to quantify the distance between a new test point and the training data manifold, a property known as *distance awareness*.

We show that by replacing the softmax output layer of a typical neural network classifier with distance aware radial basis function (RBF) units, the model can achieve high-quality uncertainty estimates without sacrificing classification accuracy. RBF networks are local classifiers and assign high confidence only in the region near the training data, making them inherently robust against adversarial attacks. Furthermore, we demonstrate a workflow to untangle uncertainties due to lack of data (i.e., epistemic uncertainty) from uncertainties due to conflicting data (i.e., aleatoric uncertainty), which allows the model to reject uncertain inputs, and to describe its reason for rejecting.

In experiments on toy datasets as well as difficult dataset pairs such as Fashion-MNIST and Dirty-MNIST, the RBF network shows notable improvements over traditional softmax neural networks in detecting out of distribution data, quantifying uncertainty, and separating epistemic and aleatoric uncertainty. This approach uses a single deterministic model and is therefore computationally efficient and scales well on larger datasets.

ABOUT THE AUTHORS

Ethan Cramer is a Research Engineer at Lone Star Analysis with experience in AI/ML development, particularly within the realm of uncertainty quantification. He earned both his B.S. and M.S. in Mechanical Engineering from the University of North Texas.

Randal Allen is the Chief Scientist of Lone Star Analysis. He is responsible for applied research and technology development across a wide range of M&S disciplines and manages intellectual property. He maintains a CMSP with NTSA. He has published and presented technical papers and is co-author of the textbook, “Simulation of Dynamic Systems with MATLAB and Simulink.” He holds a Ph.D. in Mechanical Engineering (University of Central Florida), an Engineer’s Degree in Aeronautical and Astronautical Engineering (Stanford University), an M.S. in Applied Mathematics and a B.S. in Engineering Physics (University of Illinois, Urbana-Champaign). He serves as an Adjunct Professor/Faculty Advisor in the MAE department at UCF where he has taught over 20 aerospace-related courses.

Interpretable Learning with Distance Aware Radial Basis Function Networks

Ethan Cramer
Lone Star Analysis
Addison, TX
ecramer@lone-star.com

Randal Allen
Lone Star Analysis
Orlando, FL
rallen@lone-star.com

INTRODUCTION & BACKGROUND

Despite the ever-improving capabilities of neural networks, they still face barriers to adoption in safety critical domains such as healthcare, defense, and aerospace due to their lack of interpretability, vulnerability to adversarial attacks, and inability to recognize out of distribution (OOD) data. In a sense, all three of these issues can be attributed to one property lacking in state-of-the-art neural networks: *distance awareness*. In other words, they lack the ability to preserve the distance between two input points in the feature mapping and output layer. Without distance awareness in the output layer, the model may confidently (and incorrectly) predict that an OOD data point belongs to one of the training classes. If the feature mapping is not distance preserving, the model may map the features of OOD data to in distribution (ID) regions of the feature-space, a problem known as *feature collapse*. To combat the first issue, we employ radial basis function (RBF) units in the output layer, which are inherently distance aware. In a paper called "Explaining and Harnessing Adversarial Examples," Goodfellow et al. writes, "RBF networks are naturally immune to adversarial examples, in the sense that they have low confidence when they are fooled." (Goodfellow et al., 2014). Preserving distance awareness in the hidden layers is a more challenging problem, which we address by requiring the hidden layers to satisfy the *bi-Lipschitz* condition. This condition enforces both smoothness in the features and sensitivity to input variations (Liu et al., 2020). The resulting distance aware network is considerably more interpretable than an equivalent unaware network, not only in that it can provide high-quality uncertainty estimates, but also because semantically similar data points are mapped close together in the feature-space.

Why do neural networks lack distance awareness? A typical feedforward neural network consists of groupings of computational units known as neurons which constitute a complex, non-linear mapping from input-space to feature-space. Each neuron has a tunable weight parameter and some non-linear activation function, traditionally a ReLU, tanh, or sigmoid function. The data points in feature-space are transformed by the network such that they are linearly separable, at which point a simple regression can fit the data (in the case of a regression model) or divide the data into classes (in the case of a classification model). To train a network for classification, we supply training examples in pairs of data, x, and true class labels, y, and minimize the prediction loss between the predicted class and the true class by an optimization algorithm such as gradient descent. Ideally, the distribution of training examples closely approximates the underlying distribution of the data source, though, in practice, it is difficult to account for all the possible variations; most models will eventually encounter novel OOD data, and since the model lacks the ability to say *I don't know*, it will have no choice but to assign one of the classes it does know to the unknown data. That ability to say *I don't know* comes from having some awareness of the distance between a test point and the training data manifold. Traditional models are designed to be highly linear, using mainly ReLU and non-saturating sigmoid activation functions, in order to be amenable to gradient-based optimization. As Goodfellow et al. points out, "their linear responses are overly confident at points that do not occur in the data distribution, and these confident predictions are often highly incorrect." (Goodfellow et al., 2014). In this reckoning, adversarial examples are a property of high-dimensional dot products, and a problem with too much linearity in the learned representations. This explanation is expanded upon by Hein, Adriushchenko, and Bitterwolf in their paper "Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem." They demonstrate mathematically that ReLU neural networks produce piecewise affine classifiers (affine meaning not distance preserving) with "infinitely many inputs which realize arbitrarily high confidence predictions in the networks." (Hein et al., 2019).

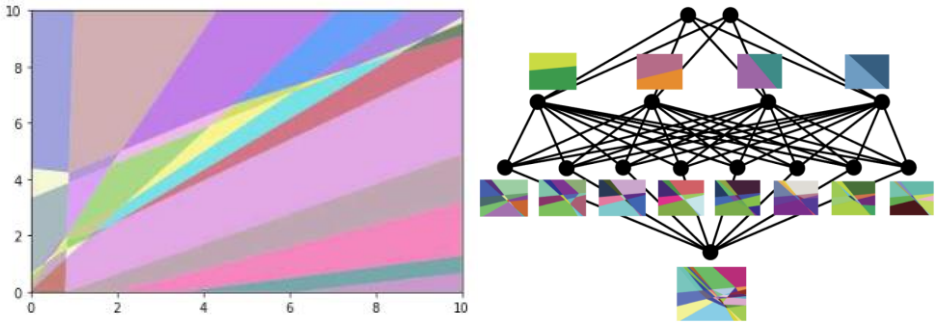

Figure 1: The linear classification regions of a two hidden layer ReLU network. The linear regions extend to infinity, which is where ReLU networks make arbitrarily high confidence predictions. Left image from (Hein et al., 2019) and right image from (Hanin & Rolnick, 2019).

There is a tradeoff to be had between high linearity of the learned representations and the tractability of training the model. Part of the reason research into RBF networks stalled out in the early 2000s was their extreme non-linearity. It took a considerable effort to stabilize training. The original appearance of RBF networks in the literature was in 1988 (Broomhead et al., 1988), but they soon disappeared in favor of ReLU and SoftMax architectures. An approach used in the original LeNet-5 architecture for handwritten digit recognition (LeCun et al., 1998) replaced only the linear output layer with RBFs to maintain gradient flow in the rest of the model, but many tricks were employed to train their convolutional neural network (CNN) for image classification. RBFs are activated based on a distance to a fixed set of centroids in the feature-space. Since the CNN is continually updating the input embeddings in feature-space, the distance to the centroids is not consistent, and this leads to instability in training. In recent years, several techniques have emerged to stabilize training of RBF networks, such as optimizing the RBF centroids using an exponential moving average of the embeddings (van den Oord et al., 2017), and minimizing the distance between the embeddings and the correct class's centroid while maximizing the distance to the other centroids using a one-vs-rest loss function (Amersfoort et al., 2020). The predicted class can be determined by finding the nearest centroid to the data embedding in feature-space, with uncertainty measured as the distance between the data embedding and the centroid.

Traditional deep neural networks with softmax output layers quantify uncertainty by measuring the entropy of the predictive categorical distribution, in which a completely uncertain answer is represented by a distribution with uniform probabilities. Unfortunately, the only type of uncertainty this approach captures is aleatoric uncertainty (Gal, 2016; Hein et al., 2019). In reality, there are two types of uncertainty we are interested in quantifying: aleatoric and epistemic. Aleatoric uncertainty, also known as irreducible uncertainty, is uncertainty inherent in the data, such as a handwritten 3 that is similar to an 8 in the MNIST dataset (Smith & Gal, 2018). In the case of aleatoric uncertainty, the true class may be impossible to determine no matter how much the model learns. Epistemic uncertainty on the other hand is reducible uncertainty; this uncertainty stems from the model's parameters and can be reduced by providing more data. When a model fails due to out of distribution data, it is a result of epistemic uncertainty (Houlsby et al., 2011). Bayesian Deep Learning (MacKay, 1992; Hinton & Van Camp, 1993) has attempted to incorporate both uncertainty sources into a holistic framework, however, it remains an open research problem to scale these types of models to a usable size. Instead, we propose a scalable and efficient approach to untangling uncertainties from a single neural network: if the data embedding is far from all RBF centroids (i.e., high epistemic uncertainty), it is considered out of distribution; if the data embedding is close to multiple centroids (i.e., high aleatoric uncertainty), it is considered in distribution but ambiguous. If both types of uncertainty are low, the model can make a correct, confident prediction.

In the following sections, we intend to:

- Describe the mathematical mechanisms involved in distance awareness and RBF-based uncertainty quantification.
- Showcase the method on a toy example to visually explore the implications of distance preservation to a model's decision regions and generalization ability.
- Apply the method to a combined Dirty-MNIST and Fashion-MNIST dataset to measure its ability to distinguish types of uncertainty.

Distance Aware Radial Basis Function Networks

Proposing distance awareness as a necessary condition for reliable and high-quality uncertainty quantification, we then seek to define what exactly is required for the model to be considered distance aware. Simply put, the model should be able to measure the distance between a testing example **x** and the training data manifold using some distance metric $\| \, . \, \|_{\mathrm{X}}$. For a more formal definition, (Liu et al. 2020) supplies this:

Definition 1 (Input Distance Awareness)**.** *Consider a predictive distribution $p(y|\mathbf{x})$ trained on a domain $\mathcal{X}_{IND} \subset \mathcal{X}$, where $(\mathcal{X}, \| \, . \, \|_{\mathrm{X}})$ is the input data manifold equipped with a suitable metric $\| \, . \, \|_{\mathrm{X}}$. We say $p(y|\mathbf{x})$ is input distance aware if there exists $u(\mathbf{x})$ a summary statistic of $p(y|\mathbf{x})$ that quantifies model uncertainty (e.g., entropy, predictive variance, etc.) that reflects the distance between* **x** *and the training data with respect to $\| \, . \, \|_{\mathrm{X}}$, i.e.,*

$$u(\mathbf{x}) = v(d(\mathbf{x}, \mathcal{X}_{IND})) \tag{1}$$

Where v is a monotonic function and $d(\mathbf{x}, \mathcal{X}_{IND}) = E_{\mathbf{x}' \sim \mathcal{X}_{IND}} \| \mathbf{x} - \mathbf{x}' \|_{\mathrm{X}}^2$ is the distance between **x** *and the training data domain.*

In a typical deep learning model where confidence (maximum predictive probability) is given by the distance between the hidden representation $h(\mathbf{x})$ and the decision boundaries, input distance awareness is not guaranteed (i.e., The distance between $h(\mathbf{x})$ and the training data $\mathcal{X}_{IND}$ is not recoverable in the model).

Now consider a model in which the confidence depends on the distance between $h(\mathbf{x})$ and a set of feature vectors $\mathbf{e}_c$, called *centroids,* corresponding to each training class *c*. This model, which makes predictions by computing a distance function between the hidden representations and the centroids, is called an *RBF network* (LeCun et al., 1998), and uncertainty $u(\mathbf{x})$ can be expressed on a per-class basis by measuring the distance to each class's centroid:

$$d_c(h(\mathbf{x}), \mathbf{e}_c) = \exp\left[-\frac{\frac{1}{n} \| \mathbf{W}_c \, h(\mathbf{x}) - \mathbf{e}_c \|_2^2}{2\sigma^2} \right], \tag{2}$$

where $\mathbf{e}_c$ is the centroid for class c with vector-length n, σ is the length scale hyper-parameter, and $\mathbf{W}_c$ is a weight matrix of size *n* (centroid size) by *d* (hidden representation output size) (Amersfoort et al., 2020). The predictive uncertainty for each class is then $u_c(\mathbf{x}) = 1 - d_c(h(\mathbf{x}), \mathbf{e}_c)$. Measuring uncertainty on a class-by-class basis allows us to determine whether the model is uncertain because the input is far from all centroids (epistemically uncertain) or because the input is close to more than one centroid (aleatorically uncertain). To untangle aleatoric uncertainty from epistemic uncertainty, we derive a second uncertainty metric which we call the signal-to-noise (SNR) score. This score is simply the squared quotient of the model's confidence in the predicted class over the sum of all other class confidences: $SNR = \frac{d_{c,max}(h(\mathbf{x}), \mathbf{e}_{c,max})}{\sum_c d_c(h(\mathbf{x}), \mathbf{e}_c)}$. For a sample to pass uncertainty screening, it must receive a sufficiently high confidence and a sufficiently high SNR score (≥ 0.7 for both, usually).

An RBF network such as the one described above is distance aware in the sense that it produces an uncertainty metric that reflects distance in the hidden representation space $\| h(\mathbf{x}) - h(\mathbf{x}') \|_H$, but it does nothing to preserve the distance from the input space to the hidden representation space. In other words, the distance in the hidden space $\| h(\mathbf{x}) - h(\mathbf{x}') \|_H$ does not necessarily correspond to the distance $\| \mathbf{x} - \mathbf{x}' \|_X$ in the training data manifold, and therefore our model does not yet satisfy the criteria for input distance awareness defined above. This can lead to feature collapse in which OOD inputs are mapped to in distribution regions of the hidden space.

One way to make the hidden representation of our model *distance preserving* is by requiring *h* to satisfy the *bi-Lipschitz* condition (Searcóid, 2007):

$$L_1 * \| \mathbf{x}_1 - \mathbf{x}_2 \|_X \leq \| h(\mathbf{x}_1) - h(\mathbf{x}_2) \|_H \leq L_2 * \| \mathbf{x}_1 - \mathbf{x}_2 \|_X \tag{3}$$

where L_1 and L_2 are positive constants with $0 < L_1 < 1$ and $L_2 > 1$. A two-sided gradient penalty on the network's outputs with respect to its inputs is one way to satisfy the bi-Lipschitz condition (Gulrajani et al., 2017). The two-sided penalty restricts the norm of the gradient to 1 using a soft constraint with a penalty on the gradient norm:

$$\lambda * (\| \nabla_{\mathbf{x}} d_c(\mathbf{x}) \|_2 - 1)^2 \quad (4)$$

where λ is the penalty coefficient, which we usually set to 5 (as high as possible without destabilizing training), and the distance function $d_c(h(\mathbf{x}), \mathbf{e}_c)$ is shortened to $d_c(\mathbf{x})$. The lower bound of this condition enforces sensitivity to input variation in the hidden representation, while the upper bound enforces smoothness in the hidden representation.

In the following section, we will demonstrate the effect of this condition on uncertainty quality, but here let us also note its implications on interpretability. We can say our model, which maps semantically similar inputs together in the feature-space and outputs confident predictions only near the training data, is considerably less brittle and more readily understandable than similar Softmax + ReLU models.

EXPERIMENTS

Two Moons

Using the scitkit-learn implementation of the Two-Moons dataset (Pedregosa et al., 2011), we can visually demonstrate several key properties of a distance aware RBF network. In this demo we generate 1000 samples with a noise level of 0.1 and train on batches containing 64 samples. The model architecture consists of 3 hidden layers with 80 neurons each. For visualization purposes, we limit the centroid size to only two dimensions (a higher dimensional hidden representation would not permit plotting). The model is trained for 100 epochs using the Adam optimizer (D. P. Kingma & J. Ba., 2014).

Figure 2 makes clear the effect of the gradient penalty on preserving input distance in the hidden representation. In 2a, the model is trained without any gradient penalty and uses only binary cross entropy loss: $L^{BCE}(\mathbf{x}, \mathbf{y}) = -\sum_c y_c \log(d_c(\mathbf{x})) + (1 - y_c)\log(1 - d_c(\mathbf{x}))$. In this case, the class weights $\mathbf{W}_c$ push the Class-1 points directly onto the Class-1 centroid and the Class-2 points directly onto the Class-2 centroid. The remaining space is distorted and compressed such that the off-class and OOD points are collapsed together.

In 2b, the gradient penalty is added to the loss function such that it becomes $L = L^{BCE} + L^{GP}$ with the gradient penalty loss $L^{GP} = \lambda * (\| \nabla_{\mathbf{x}} d_c(\mathbf{x}) \|_2 - 1)^2$. Here, a negligible accuracy penalty is incurred in return for a well-preserved relationship between distance in input space and in the hidden representation spaces. The OOD points remain far away from the class centroids while avoiding collapse onto the off-class region of the space.

Two-Moons Input Transformation (No Gradient Penalty)

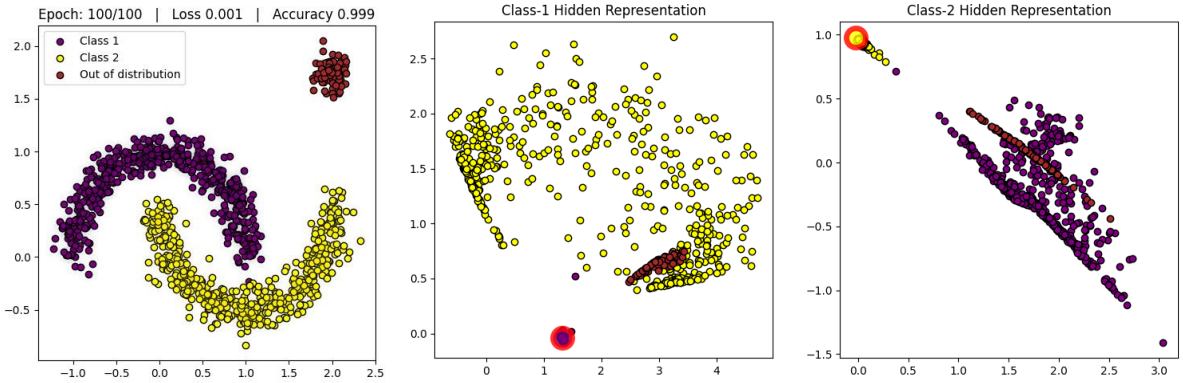

Figure 2a. The Two-Moons dataset in the input space (left), Class-1 representation space (middle), and Class-2 representation space (right) after training **without gradient penalty** applied to the loss function. The class centroids are represented by large red circles. In both cases, the true-class features are compressed down to a tiny region of space while the off-class and OOD features collapse together.

Two-Moons Input Transformation (Gradient Penalty)

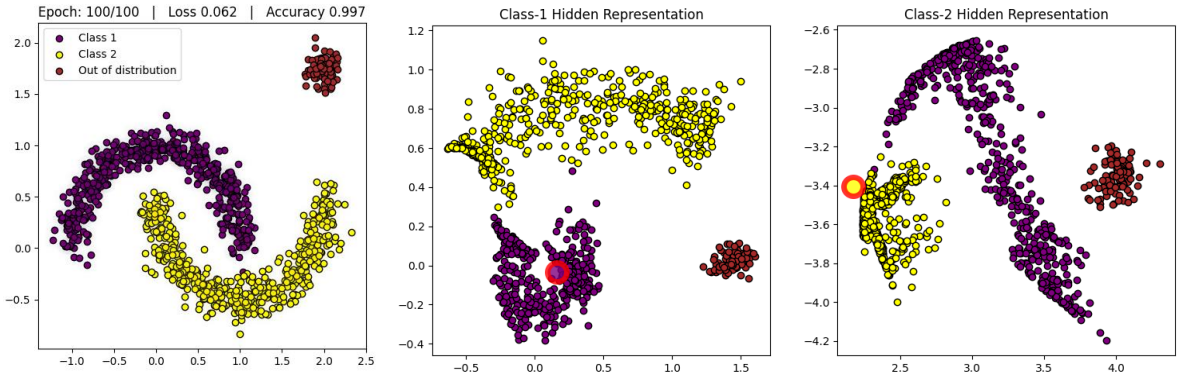

Figure 2b. The Two-Moons dataset in the input space (left), Class-1 representation space (middle), and Class-2 representation space (right) after training **with gradient penalty** applied to the loss function. The class centroids are represented by large red circles. The true-class, off-class, and OOD retain a spatial relationship that corresponds to their initial distance in input space.

In Figure 3, we see the failure of a typical ReLU + softmax neural network to separate out of distribution from in distribution data in the hidden representation space. The linear decision boundary creates two infinite regions of arbitrarily high confidence predictions.

Two-Moons Input Transformation (Typical ReLU + Softmax Network)

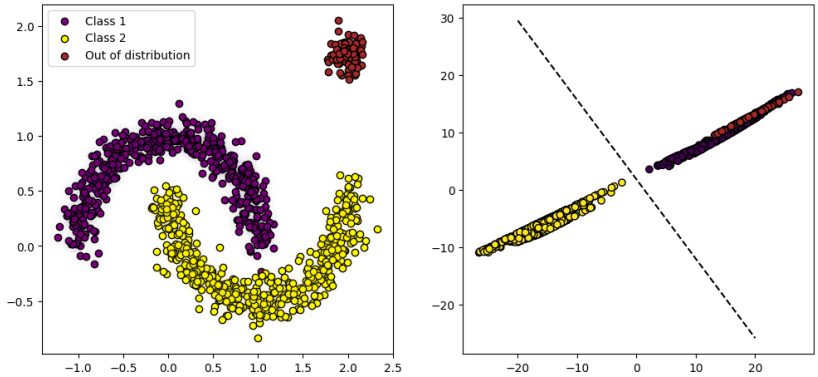

Figure 3. The Two-Moons dataset in the input space (left) and hidden representation space (right) after training a **typical ReLU + Softmax Network**. In the hidden space, the decision boundary is represented by a dashed line. The classes are well separated by the decision boundary but have no relation in hidden space to their original distance in the input space. Furthermore, the OOD data collapses onto a high confidence region of the space.

This type of decision boundary results in a very slight low confidence region between the classes and high confidence everywhere else, which is visually demonstrated by Figure 4. Compare this to the uncertainty regions produced by an RBF network, where only the regions near the training data produce high confidence predictions. Everywhere else, including the red OOD data, has low confidence.

We can project this difference in uncertainty onto a real-world scenario of identifying drones versus birds. Imagine a classification problem in which Class-1 is drones, Class-2 is birds, and the OOD data is balloons. The typical model will be successful in separating drones and birds, but due to the nature of its decision boundary will have no choice but to confidently label the balloon data as a drone. If we instead use a model with high quality uncertainty estimates, we can easily reject the balloon data as belonging to neither the drones or birds. The model was not trained to recognize anything other than these two classes; when it sees something new, it can say *I don't know*.

Uncertainty Regions (ReLU Network vs. RBF Network)

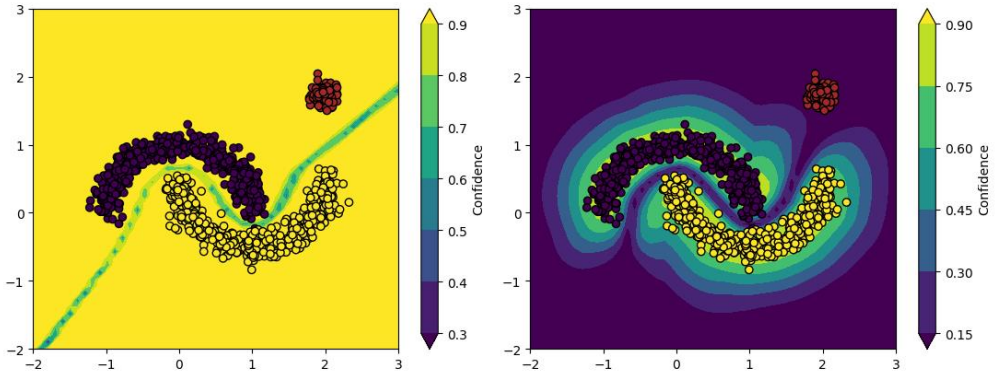

Figure 4. The uncertainty regions for a typical ReLU + softmax neural network (left) and an RBF network (right). The typical network is uncertain only along the decision boundary, allowing it to make arbitrarily high confidence predictions everywhere else, including regions completely devoid of training data. The RBF network on the other hand only has high confidence in regions where it has seen training data, and it has low confidence everywhere else.

Dirty-MNIST & Fashion-MNIST

To quantify both epistemic and aleatoric uncertainty, we need a dataset which contains OOD data and ambiguous in distribution (ID) data. The ID data is provided by the Dirty-MNIST dataset introduced by (Mukhoti et al., 2023) which contains generated ambiguous MNIST samples with varying entropies. For OOD data, we use the Fashion-MNIST dataset (Xiao, H., Rasul, K. and Vollgraf, R., 2017). Fashion-MNIST contains 28x28 grayscale images associated with labels from 10 classes, and it serves as a drop-in replacement for standard MNIST. See Figure 5 for selection of samples from each of the datasets.

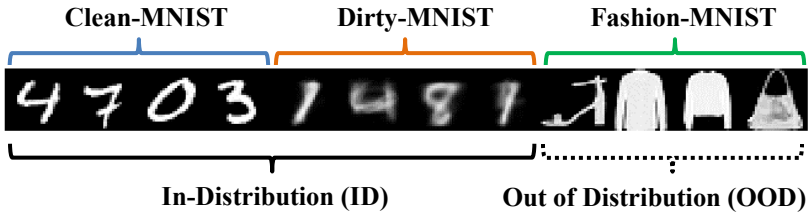

Figure 5. Clean-MNIST and Dirty-MNIST are ID, while Fashion-MNIST is OOD. Only the Clean-MNIST data is used for training. A portion of this image is taken from the dataset creators (Mukhoti et al., 2023).

We train two models on the Clean-MNIST training dataset: (1) a standard convolutional neural network with softmax output layer, and (2) an equivalent RBF network. Since neither model is trained on the Dirty-MNIST or Fashion-MNIST data, they should not be able to accurately classify the data. Results follow in Table 1.

Table 1. Softmax and RBF network performance on each of the three datasets.

Dataset	Model	Raw Accuracy (%)	Accuracy After Rejection (%)	Reject Rate (%)
Clean-MNIST	Softmax network	98.9	99.3	1.1
	RBF network	98.6	99.8	14.1
Dirty-MNIST	Softmax network	72.8	80.3	24.9
	RBF network	64.6	87.4	80.8
Fashion-MNIST	Softmax network	-	-	80.0
	RBF network	-	-	99.9

Across the results, it seems that the RBF network pays a small penalty in raw accuracy score over the equivalent Softmax network. However, after rejecting ambiguous and/or OOD samples from the test set, the accuracy of the RBF

network shows noticeable improvement over the Softmax network. For the Fashion-MNIST test set, there is no correct label (in our experimental setup), so perfect performance is equivalent to rejecting all samples. The RBF network rejected 99.9% of samples and misclassified the remaining 0.1% of samples, whereas the Softmax network only rejected 80.0% percent of samples, leaving 20% (2,000 samples) to be misclassified. Furthermore, all of these misclassifications occur with higher than 70% confidence. See Figure 6 for a breakdown of which samples were accepted versus rejected due to ambiguity and rejected due to out of distribution.

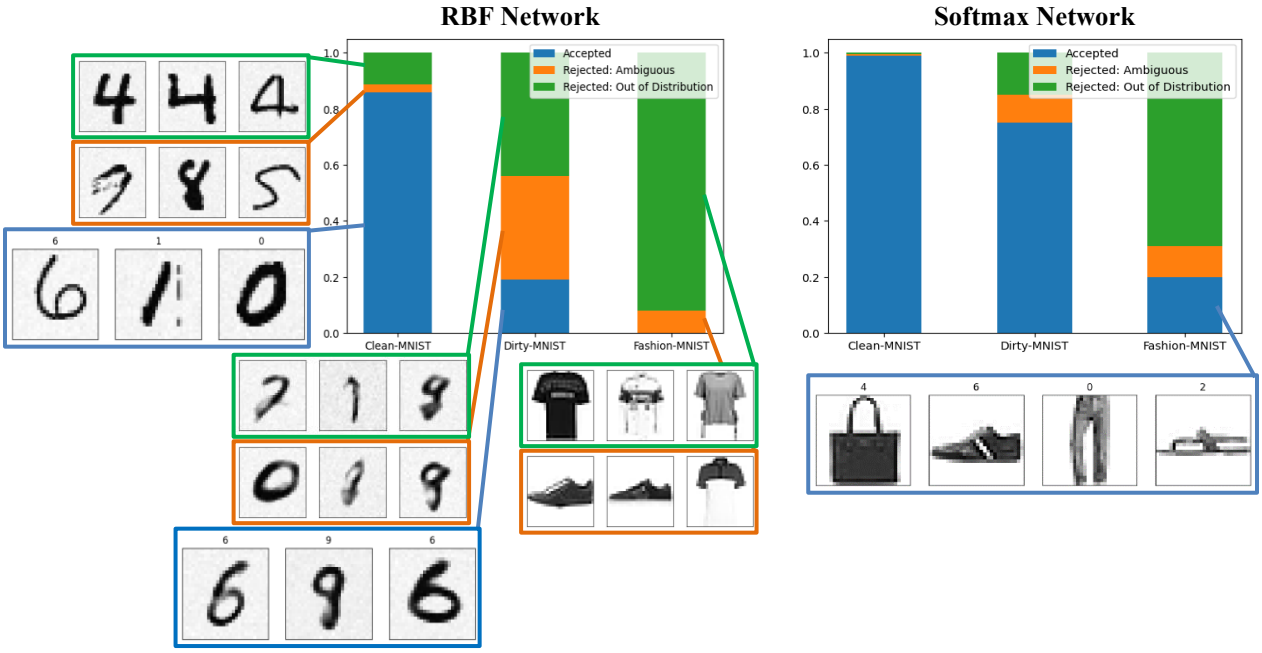

Figure 6. Breakdown of classification versus rejection for the RBF network and Softmax network on each of the three datasets. The Softmax network attempts to classify 20% of the Fashion-MNIST samples, resulting in purses, pants, and shoes confidently receiving labels of 4, 6, 0, and 2.

It's interesting to speculate on why the models decided to reject certain inputs and accept others. Why did so many 4's get labeled OOD by the RBF network? One hypothesis, the *manifold hypothesis*, is that "MNIST is a low dimensional manifold, sweeping and curving through its high-dimensional embedding space" (Olah, 2014). In other words, all of the variation in the dataset is contained in a very small region, which makes it easy to model with a typical neural network, and difficult to find the RBF centroids which separate ID from OOD data. It's worth noting that our confidence threshold, 0.7, automatically rejects inputs which receive lower than 70% confidence—so in practice the RBF network could have made correct, somewhat uncertain predictions for the rejected Clean-MNIST samples. Ultimately, the confidence and SNR thresholds indicate the level of risk tolerance desired in the model. Some tasks may have severe consequences for false-positives (weapons targeting, medical diagnosis), in which cases one may want to turn up the thresholds and pass anything with middling certainty over to a human reviewer. Low thresholds might be preferred in cases where the goal is high accuracy and the dataset is unlikely to contain much ambiguity or OOD data.

In Figure 7 and 8 we see that the RBF network is able to (1) separate OOD (Fashion-MNIST) from ID (Dirty-MNIST and Clean-MNIST) using confidence as an uncertainty metric and (2) separate ambiguous samples (Dirty-MNIST) from clean samples (Clean-MNIST) using the SNR score as an uncertainty metric. Compared to the RBF network, the Softmax network struggles to separate any uncertainty. All Clean-MNIST samples and most of the Dirty-MNIST samples are predicted with 100% confidence. Even for some Fashion-MNIST samples, the model is 100% confident in its predictions. Although the Softmax network succeeded in rejecting some of the OOD samples, it does not have a robust ability to holistically quantify uncertainty.

Kernel Density Estimate of an RBF Network's Confidence and SNR

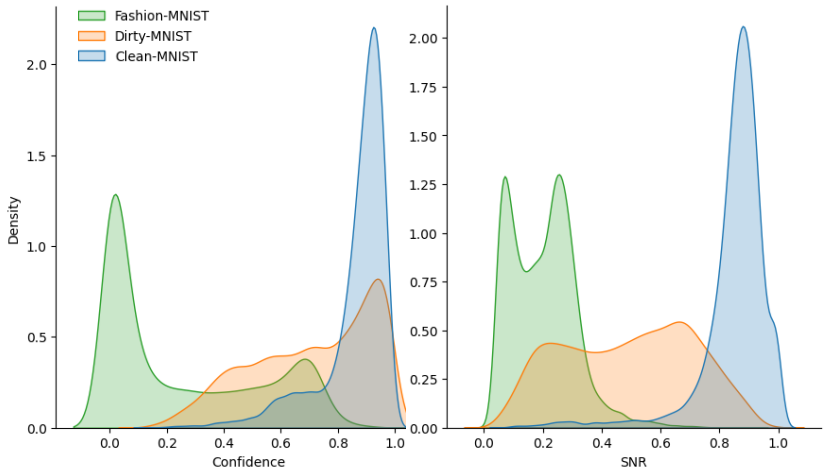

Figure 7. The RBF network can separate OOD (Fashion-MNIST) from ID (Dirty-MNIST and Clean-MNIST) using confidence as an uncertainty metric. Using the SNR score the model can separate ambiguous samples from clean samples. With these two scores, we can reject samples due to high aleatoric uncertainty (high confidence + low SNR) or due to high epistemic uncertainty (low confidence).

Kernel Density Estimate of a Softmax Network's Confidence and SNR

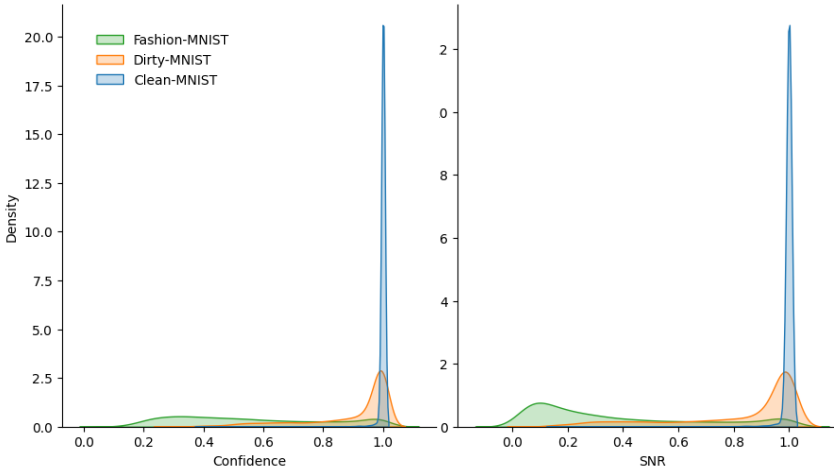

Figure 8. Compared to the RBF network above, the Softmax network struggles to separate any uncertainty. Nearly all Clean-MNIST samples and most Dirty-MNIST samples are predicted with 100% confidence. Even for some Fashion-MNIST samples, the model is 100% confident in its predictions. Although the Softmax network succeeded in rejecting some of the OOD samples, it does not have a robust ability to truly quantify uncertainty.

While the RBF network approach demonstrates noticeable improvements over the traditional network, there is still work to be done in formalizing the distinction between aleatoric and epistemic uncertainty in the model's output. The current method of using a combination of confidence and a "signal-to-noise" score succeeded in rejecting out of distribution data and a large portion of ambiguous data, but it also rejected some of the clean data which the Softmax network successfully classified. This approach also requires setting thresholds for confidence and SNR which adds an additional level of complexity when creating architectures for new problems.

Still, we believe this approach provides a promising alternative to traditional neural networks, especially in high-risk domains where a high-confidence misclassification can result damaged equipment, injury, or worse. If the model was tasked with classifying drones and birds, in some cases it might not even be clear to a human observer if a particular image (or radar PDW) is a bird, drone, or something else. Not only do we need to trust the model to tell us "I don't know" when it's not sure if the data is a bird or a drone, we also need the model to tell us if the data is neither a bird nor a drone, but something else entirely, like a kite or a balloon.

CONCLUSIONS

In this paper we have motivated the need for better uncertainty quantification methods, namely distance awareness, in neural networks, and shown that RBF networks are a viable approach in this regard. Distance aware RBF networks are more robust than equivalent Softmax networks in rejecting adversarial attacks, and they are more interpretable in the sense that they place semantically similar data points close together in feature-space.

We have seen that enforcing sensitivity and smoothness conditions on the model's hidden representation allows the RBF network to avoid feature collapse and make high-quality uncertainty estimates while maintaining accuracy. Because RBF networks are local classifiers, they assign high confidence only in the regions near the training data, making them naturally immune to adversarial attacks. We have also demonstrated a workflow for separating epistemic uncertainty—uncertainty due to lack of data—from aleatoric uncertainty—uncertainty due to conflicting data. This was demonstrated on the difficult dataset pair of Dirty-MNIST and Fashion-MNIST, which contains clean, ambiguous, and out of distribution samples. We saw notable improvements over a more traditional ReLU and Softmax neural network in quantifying uncertainty.

REFERENCES

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Liu, J., Lin, Z., Padhy, S., Tran, D., Bedrax Weiss, T., & Lakshminarayanan, B. (2020). Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in neural information processing systems*, *33*, 7498-7512.

Hein, M., Andriushchenko, M., & Bitterwolf, J. (2019). Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 41-50).

Hanin, B., & Rolnick, D. (2019, May). Complexity of linear regions in deep networks. In *International Conference on Machine Learning* (pp. 2596-2604). PMLR.

Lowe, D., & Broomhead, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex systems*, *2*(3), 321-355.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Van Den Oord, A., & Vinyals, O. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, *30*.

Van Amersfoort, J., Smith, L., Teh, Y. W., & Gal, Y. (2020, November). Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning* (pp. 9690-9700). PMLR.

Gal, Y., & Ghahramani, Z. (2016, June). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050-1059). PMLR.

Smith, L., & Gal, Y. (2018). Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*.

Houlsby, N., Huszár, F., Ghahramani, Z., & Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Mackay, D. J. C. (1992). *Bayesian methods for adaptive models*. California Institute of Technology.

Hinton, G. E., & Van Camp, D. (1993, August). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory* (pp. 5-13).

O'Searcoid, M. (2006). *Metric spaces*. Springer Science & Business Media.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, *30*.

Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Mukhoti, J., Kirsch, A., van Amersfoort, J., Torr, P. H., & Gal, Y. (2023). Deep deterministic uncertainty: A new simple baseline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 24384-24394).

Olah, C. (2014, October 9). *Visualizing Mnist: An exploration of dimensionality reduction*. Visualizing MNIST: An Exploration of Dimensionality Reduction - colah's blog. https://colah.github.io/posts/2014-10-Visualizing-MNIST/