Islands of Confidence: Robust Neural Network Classification with Uncertainty Quantification

Anonymous authors
Paper under double-blind review

Abstract

We propose a Gaussian confidence measure and its optimization, for use in neural network classifiers. The measure comes with theoretical results, simultaneously resolving two pressing problems in Deep Neural Network classification: uncertainty quantification, and robustness. Existing research in uncertainty quantification mostly revolves around the confidence reflected in the input feature space. Instead, we focus on the learned representation of the network and analyze the confidence in the penultimate layer space. We formally prove that, independent of optimization-procedural effects, a set of centroids always exists such that softmax classifiers are nearest-centroid classifiers. Softmax confidence, however, does not reflect that the classification is based on nearest centroids: artificially inflated confidence is also given to out-of-distributions samples that are not near any centroid, but slightly less distant from one centroid than from the others. Our new confidence measure is centroid-based, and hence no longer suffers from the artificial confidence inflation of out-of-distribution samples. We also show that our proposed centroidal confidence measure is providing a robustness certificate against attacks. As such, it manages to reflect what the model doesn't know (as demanded by uncertainty quantification), and to resolve the issue of robustness of neural networks.

1 Introduction

The last layer of state-of-the-art neural networks computes the final classifications by approximation through the softmax function (Boltzmann, 1868). This function partitions the transformed input space into Voronoi calls, each of which encompasses a single class. Conceptually, this is equivalent to putting a number of centroids in this transformed space, and clustering the data points in the dataset by proximity to these centroids through k-means. Several recent papers posed that exploring a relation between softmax and k-means can be beneficial (Kilinc & Uysal, 2018; Peng et al., 2018; Schilling et al., 2018).

The current state of scientific knowledge on the relation between k-means and softmax is empirical. In this paper, we theoretically prove that softmax is a centroid-based classifier, and we derive a centroid-based robustness certificate. This certificate motivates the usage of a confidence measure[1], the *Gauss confidence*, which reflects the distance of observations to their assigned centroids. Gauss confidence therefore expresses the uncertainties of the model; moreover, it indicates the vulnerabilities to attacks.

We show that our Gauss networks can match (median absolute difference: 0.45 percentage points) the test accuracy of softmax networks, but at a lower confidence (as desired); both outperform the competing DUQ networks (van Amersfoort et al., 2020) when the dataset has many classes. The lower confidence also results in Gauss networks being much less susceptible to adversarial attacks. Hence, the islands of confidence as illustrated in the rightmost plot of Figure 1 reflect reality much better than the confidence landscapes of existing methods (cf. other two plots in Figure 1).

[1]previously published informally (non-peer-reviewed) in January 2020 as (NN et al., 2020)

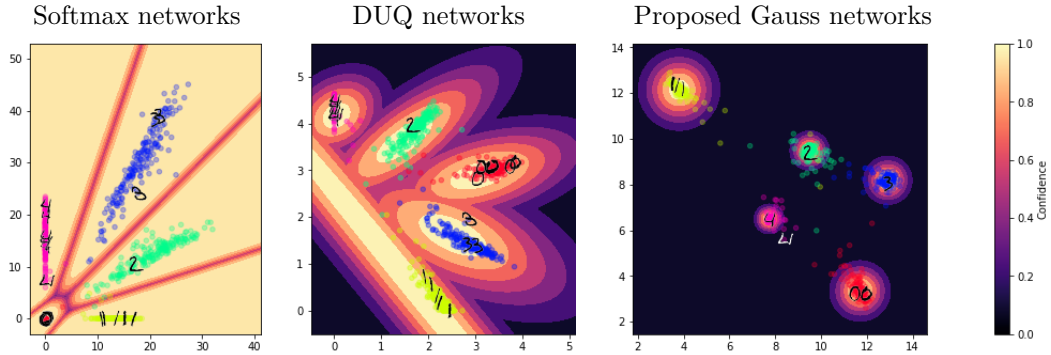


Figure 1: Classification confidence plots of actual decision boundaries/areas of a $d = 2$-dimensional penultimate layer space for three distinct Deep Neural Networks. The hand-written numbers indicate the MNIST class predictions in $\{0, \dots, 4\}$.

2 Related Work

Peng et al. (2018); Kilinc & Uysal (2018) were among the first to observe that the penultimate layer has surprising clustering properties. The phenomenon that samples gather around their centroid in the transformed feature space recently gained popularity under the term *neural collapse* (Kothapalli et al., 2022; Papyan et al., 2020). This motivates an adaptation of the confidence measure. Softmax confidence has the well-documented issue that it extrapolates with unjustified high confidence data points which are very far from the training data (Gal, 2016): softmax is overconfident. The consequences are observable in the very confident classification of noisy, entirely randomly generated images (Nguyen et al., 2015), and the effect that softmax probabilities are badly calibrated (Guo et al., 2017). Figure 1 illustrates that this effect is not surprising: in the transformed feature space of the penultimate layer, points that do not lie directly on the decision boundaries are all confidently assigned to a class. Softmax confidence has no capability to express what the model doesn't know.

To remedy this issue, various methods have been proposed. Although networks with a centroid-based confidence intuitively make sense, training them is a major challenge. When using the softmax cross-entropy loss, pushing a transformed feature vector $\phi(x)$ away from one class region means that it is pushed towards other class regions. Employing a centroidal confidence, one could push a point away from all centroids. This introduces trivial global optima of common loss functions, which does not lead to a well-performing classifier. To this end, Wen et al. (2016); Pang et al. (2019) minimize the distance of transformed feature points to given (fixed) centroids. Wan et al. (2018) employ a centroidal cross entropy loss, which directly optimizes for a nearest centroid classifier, but which does not increase the density around centroids. Likewise, Lebedev et al. (2018) define a confidence measure based on the normalized RBF kernel function. Hobbhahn et al. (2022) propose a retraining of only the last layer to reflect uncertainties based on a multivaiate Gaussian representation of confidences. Closest to our approach are Deterministic Uncertainty Classifiers (DUQs) (van Amersfoort et al., 2020), which directly optimize for a multivariate Gaussian confidence measure (cf. Figure 1). The authors employ binary cross-entropy loss, which is given for one-hot encoded labels $Y^* \in \mathbb{R}^{m \times c}$ for m samples and c classes by

$$\ell_{\text{BCE}}(f_{\text{DUQ}}, Y^*) = -\frac{1}{mc} \sum_{j=1}^{m} \sum_{k=1}^{c} Y^*_{jk} \log\left(f_{\text{DUQ}}(\mathbf{x}_j)\right) + (1 - Y^*_{jk}) \log\left(1 - f_{\text{DUQ}}(\mathbf{x}_j)\right)$$

$$f_{\text{DUQ}}(\mathbf{x})_k = \exp\left(-(\phi(\mathbf{x}) - Z_{\cdot k})^\top \Sigma_k^{-1} (\phi(\mathbf{x}) - Z_{\cdot k})\right).$$

Here, $f_{\text{DUQ}} : \mathbb{R}^n \to [0, 1]^c$ returns the prediction confidences, $Z \in \mathbb{R}^{d \times c}$ gathers the centroids by its columns and $\Sigma_k \in \mathbb{R}^{d \times d}$ is a scaled covariance matrix. BCE loss works well for data with a moderate number of classes. However, we observe with increasing class complexity

that DUQ models converge to the global optimum where all training data is infinitely far away from all centroids ($\|\Sigma^{-1}\| \to 0$).

2.1 Robustness of Deep Neural Networks (DNNs)

The existence of adversarial examples demonstrates that SOTA deep learning models have warped inner representations of classes/object properties. The fact that adding humanly indistinguishable noise changes the classification from a bus to an ostrich is worrying. Consequently, a lot of research focuses on performing and preventing these attacks. The result is a circular development of attacks and defenses (Akhtar & Mian, 2018; Carlini et al., 2019; Athalye et al., 2018), which creates a need for certifiable robustness. A certificate provides, e.g., a lower bound on the magnitude of adversarial perturbation required to change the class. Existing certificates rely on bounds of the *global* Lipschitz constant, limiting the effect of any, arbitrarily large perturbation (Szegedy et al., 2014; Qian & Wegman, 2019; Gouk et al., 2018). Experimental and theoretical evaluations show that global Lipschitz bounds overly curtail the expressiveness of DNNs, yielding underfitting classifiers (Huster et al., 2018). The expressiveness of DNNs, their ability to approximate almost any function (Lu et al., 2017), is an important property which distinguishes feature transformations learned by DNNs from popular alternatives. Hence, we require a control of the *local* Lipschitz constant (restricting only the effect of small perturbations). Unfortunately, this is currently only possible for DNNs having one hidden layer (Hein & Andriushchenko, 2017).

Another way to increase robustness is the maximum margin approach. Enlarging the margin between classes in the transformed feature space, so is the idea, requires larger perturbances to push an example over the decision boundary (Liu et al., 2016; Pang et al., 2019). However, the success of this approach still depends on the Lipschitz constant of the learned feature transformation, facing again the problem to control the Lipschitz constant while maintaining expressiveness (Tsuzuku et al., 2018). The Gauss networks we introduce combine all these elements naturally yet explicitly: Lipschitz constant, margin, and confidence.

3 Why Softmax Is Actually a Centroid-based Classifier

Consider a feedforward network, mapping points in n-dimensional space to a c-dimensional probability vector (with c the number of classes), computing predictions by the function

$$f_{\text{sm}}(\mathbf{x}) = \text{softmax}\left(W^\top \phi(\mathbf{x}) + \mathbf{b}\right).$$

Here, $\phi(\mathbf{x}) \in \mathbb{R}^d$ returns the output of the penultimate layer. The last layer is linear; its weights are represented by the matrix $W \in \mathbb{R}^{d\times c}$ and bias $\mathbf{b} \in \mathbb{R}^c$.

3.1 Softmax and Inherent Clustering Properties

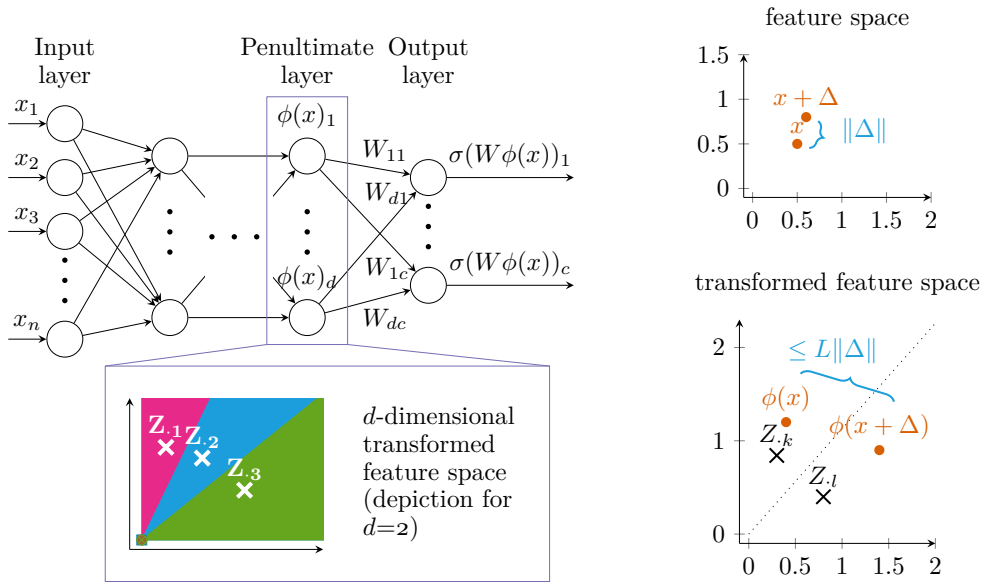
Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ denote our training data of data points $\mathbf{x}_j \in \mathbb{R}^n$ and their corresponding class labels $y_j \in \{1, \ldots, c\}$. We denote by $\|\cdot\|$ the vector l_2-norm or the matrix Frobenius norm and by $|\cdot|$ the vector l_1-norm. Figure 2a illustrates the result of Theorem 1, showing that softmax-based classifiers are nearest centroid classifiers.

Theorem 1. *Let $W \in \mathbb{R}^{d\times c}$ be the matrix of weights between the penultimate layer and the output layer. Let $\phi : \mathbb{R}^n \to \mathbb{R}^d$ be the DNN output of the penultimate layer. If the network computes predictions over the (soft)max function with bias term $\mathbf{b}$ such that $y = \arg\max_k \phi(\mathbf{x})^\top W_{\cdot k} + b_k$ and if W has at least a rank of $r \geq c$, then there exist c class centroids $Z_{\cdot k} \in \mathbb{R}^d$ such that every point $\mathbf{x}$ is assigned to the class having the nearest centroid:*

$$y = \arg\min_k \|\phi(\mathbf{x}) - Z_{\cdot k}\|^2 .$$

Proof. We show that for any dataset and network there exists a set of centroids such that the classification does not change when classifying according to the nearest centroid.

We gather the outputs of the penultimate layer in the matrix D reflecting the transformed feature vectors as rows, or conversely, as columns in $D^\top$:



(a) A feed-forward neural network, computing a feature transformation ϕ and the representation of classes in the transformed feature space by convex cones, induced by centroids $Z_{\cdot 1}, \ldots, Z_{\cdot c}$.

(b) Gauss confidence certifies robustness: a minimum distance must be bridged to change the predicted class.

Figure 2: The role of Gauss confidence for Deep Neural Networks.

$$D^\top = (\phi(\mathbf{x}_1) \quad \ldots \quad \phi(\mathbf{x}_m)) \in \mathbb{R}^{d \times m}.$$

We define $Z = W + \mathbf{v}\mathbf{1}_c^\top$, where $\mathbf{v} \in \mathbb{R}^d$ and $\mathbf{1}_c \in \{1\}^c$ is a constant one vector. The (soft)max classification of all data points in $\mathcal{D}$ is then given by the one-hot encoded matrix $Y \in \{0,1\}^{m \times c}$ that optimizes the objective

$$\arg\max_Y \operatorname{tr}(Y(W^\top D^\top + \mathbf{b}\mathbf{1}_m^\top)) = \arg\min_Y \|D - YZ^\top\|^2 + \operatorname{tr}((2\mathbf{b}\mathbf{1}_c^\top - Z^\top Z)Y^\top Y)) \tag{1}$$

Full derivation of this equivalence is given in Appendix A. The matrix $Z \in \mathbb{R}^{d \times c}$ indicates a set of c centroids by its columns. The first term of Equation (1) is minimized if Y assigns the class with the closest centroid to each data point in D. Hence, if we can show that there exists a vector $v \in \mathbb{R}^d$ such that the second term of Equation (1) is equal to zero (given D and W) then we have shown what we wanted to prove. Since $|Y_{j\cdot}| = 1$ (every point is assigned to exactly one class), the matrix $Y^\top Y$ is a diagonal matrix, having the number of data points assigned to each class on the diagonal: $Y^\top Y = \operatorname{diag}(|Y_{\cdot 1}|, \ldots, |Y_{\cdot c}|)$. Hence, the trace term on the right of Equation (1) equals

$$\sum_{k=1}^{c} (2\mathbf{b}\mathbf{1}_c^\top - Z^\top Z)_{kk} |Y_{\cdot k}| = \sum_{k=1}^{c} (2b_k - \|W_{\cdot k}\|^2 - 2v^\top W_{\cdot k})|Y_{\cdot k}| - \|\mathbf{v}\|^2 m \tag{2}$$

Full derivation of this equivalence is given in Appendix B. We define the vector $\mathbf{u} \in \mathbb{R}^c$ such that $u_k = b_k - \frac{1}{2}\|W_{\cdot k}\|^2$. The right term of Equation (1) is constant for a vector $\mathbf{v}$ satisfying $u_k = \mathbf{v}^\top W_{\cdot k}$ for $1 \leq k \leq c$. That is, we need to solve the following equation for $\mathbf{v}$:

$$\mathbf{u} = W^\top \mathbf{v} = V\Sigma U^\top \mathbf{v}.$$

Since the rank of W is c (full column rank), this equation has a solution. It is given by the SVD of $W = U\Sigma V^\top$, where $U \in \mathbb{R}^{d \times c}$ is a left orthogonal matrix ($U^\top U = I$), $\Sigma \in \mathbb{R}_+^{c \times c}$ is a diagonal matrix having only nonnegative values, and $V \in \mathbb{R}^{c \times c}$ is an orthogonal matrix ($V^\top V = VV^\top = I$). Setting $\mathbf{v} = U\Sigma^{-1}V^\top \mathbf{u}$, this vector solves the equation. $\square$

The objective $\min_Y \|D - YZ^\top\|^2$ is the k-means cluster assignment objective in matrix factorization form. Hence, class predictions correspond to a Voronoi tesselation of $\mathbb{R}^d$. Classification accuracy is unaffected by the distance of the points $\phi(x)$ to the centroid, as long as they are in the correct Voronoi cell. Softmax confidence is high for points maximizing the inner product $\phi(x)^\top Z_{\cdot k}$, where $Z_{\cdot k}$ is the center of the predicted class, since

$$\begin{aligned}\operatorname{softmax}(W^\top \phi(\mathbf{x}) + \mathbf{b})_k &= \frac{\exp(\phi(\mathbf{x})^\top W_{\cdot k} + b_k)}{\sum_{l=1}^c \exp(\phi(\mathbf{x})^\top W_{\cdot l} + b_l)} \cdot \frac{\exp(\phi(\mathbf{x})^\top \mathbf{v})}{\exp(\phi(\mathbf{x})^\top \mathbf{v})} \\ &= \frac{\exp(\phi(\mathbf{x})^\top Z_{\cdot k} + b_k)}{\sum_{l=1}^c \exp(\phi(\mathbf{x})^\top Z_{\cdot l} + b_l)} = \operatorname{softmax}(Z^\top \phi(\mathbf{x}) + \mathbf{b})_k.\end{aligned}$$

Hence, softmax confidence is high for points $\phi(\mathbf{x})$ aligning with the direction of their class center $Z_{\cdot k}$ and having a large norm in the transformed feature space. However, the empirical observation of neural collapse indicates that networks do in fact not map the training and test data arbitrarily far away from their centroids but rather into their neighborhood. Thus, one could attach semantic meaning to the distance of transformed samples to class centroids.

3.2 A Centroid-Based Robustness Certificate

In addition to the potential reflection of learned representations of a neural network, the distance of points to their centroid also determines the robustness of a network, as the following result shows (illustrated in Figure 2b).

Theorem 2. *Let $\mathbf{x} \in \mathbb{R}^n$ be a data point with predicted class k and let $Z \in \mathbb{R}^{d \times c}$ be the centroid matrix. Assume $\phi : \mathbb{R}^n \to \mathbb{R}^d$ is Lipschitz continuous with modulus L_ϕ. Any distortion $\Delta \in \mathbb{R}^n$ changing the prediction of point $\mathbf{x} + \Delta$ to another class $l \neq k$ has then a minimum size of*

$$\|\Delta\| \geq \frac{\|Z_{\cdot l} - Z_{\cdot k}\| - \|\phi(\mathbf{x} + \Delta) - Z_{\cdot l}\| - \|\phi(\mathbf{x}) - Z_{\cdot k}\|}{L_\phi}.$$

Proof. Let $\mathbf{x}, \Delta$ and Z be as described above. We derive from the triangle inequality and from the Lipschitz continuity the following inequality:

$$\begin{aligned}\|\phi(\mathbf{x} + \Delta) - Z_{\cdot k}\| &\leq \|\phi(\mathbf{x} + \Delta) - \phi(\mathbf{x})\| + \|\phi(\mathbf{x}) - Z_{\cdot k}\| \\ &\leq L_\phi \|\Delta\| + \|\phi(\mathbf{x}) - Z_{\cdot k}\|. \end{aligned} \tag{3}$$

The triangle inequality also yields:

$$\|Z_{\cdot l} - Z_{\cdot k}\| \leq \|\phi(\mathbf{x} + \Delta) - Z_{\cdot l}\| + \|\phi(\mathbf{x} + \Delta) - Z_{\cdot k}\|.$$

Subtracting $\|\phi(\mathbf{x} + \Delta) - Z_{\cdot l}\|$ yields a lower bound on the distance $\|\phi(\mathbf{x}) - Z_{\cdot k}\|$, which we apply in Equation (3) to obtain the final bound on the distortion Δ. □

Theorem 2 provides three explanations for the phenomenon that predictions of neural networks can be flipped by small perturbations: 1) the Lipschitz modulus is large, 2) class centroids are close to each other, and 3) point $\mathbf{x}$ or $\mathbf{x} + \Delta$ is not mapped close to their class centroid. Case 1 remains an open problem: this paper will not provide a manner to suitably control the Lipschitz constant. Case 2 motivates a maximum margin approach where the centroids are supposed to be as far away from each other as possible. Case 3 motivates a novel confidence score, which reflects the distance to the centroid. In the desirable situation where the Lipschitz constant is small and centroids are far away from each other, small perturbations only result in a prediction change if at least one of the points $\mathbf{x}$ or $\mathbf{x} + \Delta$ is far from its centroid. Hence, a confidence score reflecting the distance to the centroid is in alignment with the theoretical robustness guarantee of Theorem 2. Such a confidence score expresses not only the model's uncertainties, but also its vulnerability to attacks.

4 Gauss Confidence and Gauss Networks

Theorem 2 motivates the direct optimization for *good* centroids which are in current networks only indirectly learned. To do so, we employ a natural choice for a confidence measure which reflects the proximity to the cluster centroids via the Gaussian RBF-kernel function.

Definition 1. Given a function $\phi : \mathbb{R}^n \to \mathbb{R}^d$ and a centroid matrix $Z \in \mathbb{R}^{d \times c}$, we define predictions with *Gauss confidence* by the function $f_{\text{ga}}(\mathbf{x})$, where for $k \in \{1, \dots, c\}$

$$f_{\text{ga}}(\mathbf{x}; \gamma)_k = \exp(-\gamma_k \|\phi(\mathbf{x}) - Z_{\cdot k}\|^2) \in (0, 1].$$

The parameter $\gamma \in \mathbb{R}_+^c$ determines how close the transformed samples have to be to their centroid in order to get a high confidence. The Gauss confidence is a special case of the multivariate Gaussian confidence (used, e.g., by DUQ networks) where the covariance matrix for class k is equal to $\Sigma_k = {}^1\!/_{\gamma_k} I$. This might seem restrictive, but this choice has two advantages. First, the margin between classes is more easily controllable over the distance between centroids when using Gauss confidence. For example, we can see in Figure 1 that the centroids of class 2 and 3 are comparatively far away, but due to the multivariate confidence, the margin between the confidently assigned areas is small. Second, the required storage for the covariance matrix in multivariate confidences increases vastly with the dimensionality of the penultimate layer space. For example, for a ResNet-50 architecture, the covariance matrix has a dimensionality of 2048×2048. We can also not decrease the required storage by employing a low rank approximation of Σ^{-1}, since this would introduce directions in which the confidence is always equal to one. In turn, the Gauss confidence requires only to store the c-dimensional vector γ.

4.1 Training Gauss Networks

We replace softmax with the Gauss confidence. While the weights W of traditional networks indirectly determine the class centroids Z, our network learns the centroids directly, represented by the matrix $W = Z$, connecting the penultimate with the last layer.

Based on the result of Theorem 2, we aim for robust networks which map well-classifiable points close to the corresponding centroid and points which are difficult to classify further away from all centroids. In terms of clustering, we aim to minimize the within-class-scatter while maximizing the between-class-scatter. We propose a novel loss function that is based on the Gauss confidence and achieves both: it optimizes for feature transformations that map samples close to their centroid and centroids far away from each other. To this end, we introduce a parameter vector $\delta \in [0.01, 0.9]^c$.

$$\ell_{\text{ga}}(f_{\text{ga}}, Y^*) = -\frac{1}{m} \sum_{j=1}^{m} \log \left(\frac{f_{\text{ga}}(\mathbf{x}_j; \gamma)_{y_j}}{f_{\text{ga}}(\mathbf{x}_j; \gamma)_{y_j} + \sum_{k \neq y_j} f_{\text{ga}}(\mathbf{x}_j; \delta \circ \gamma)_k} \right) + \log \left(f_{\text{ga}}(\mathbf{x}_j; \gamma \circ \delta) \right)_{y_j}$$

The $\circ$ product indicates here the element-wise (Hadamard) product. The left term of the loss resembles cross-entropy loss. We introduce here the vector δ to simulate higher confidences for the wrong classes during training. This pushes centroids further away from samples that do not belong to their class, and hence increases the margin. The second term resembles negative log likelihood and increases the sample density around the centroids. The inclusion of δ forces the network to have higher sample density (and hence a smaller confidence ball around the centroid) when δ is large, and hence the margin around that class is supposedly small. To optimize for small values of δ, we further employ a weight decay. We optimize the parameters of the network, as well as the parameters γ and δ, with SGD.

While the cross-entropy term pushes towards perfect training data classification, our proposed ℓ_{ga} loss is more sensitive to the initialization than softmax models. We make use of the effect of neural collapse and our proven connection of softmax networks and nearest-centroid classifiers, and employ a warm start: we train a softmax network as usual for some epochs, then we compute the class centroids and determine the initial centroid matrix for the Gauss confidence. We then train the Gauss network for the remaining epochs with ℓ_{ga}.

5 Experimental Results

We evaluate the proposed Gauss networks with respect to robustness and a suitable reflection of classification confidences by the Gauss confidence. For this purpose we compare popular

Table 1: Test accuracy (acc) and average confidence (conf) of MNIST, Cifar-10, Cifar-100 and ImageNet for softmax, DUQ and Gauss networks, using LeNet and ResNet18 (He et al., 2015) architectures.

	MNIST LeNet		Cifar-10 ResNet18		Cifar-100 ResNet18		ImageNet ResNet50	
Network	acc	conf	acc	conf	acc	conf	acc	conf
Softmax	99.3%	0.99	94.4%	0.98	76.4%	0.86	75.7%	0.81
DUQ	98.4%	0.93	94.3%	0.99	2.6%	0.35	-	-
Gauss	99.3%	0.87	94.7%	0.91	75.8%	0.64	72.0%	0.37

network architectures with the refined Gauss variant on MNIST (Lecun et al., 1998), Cifar-10 and -100 (Krizhevsky, 2009), and ImageNet (Russakovsky et al., 2015). Our Pytorch implementation with scripts for training and attacking is publicly available[2].

We evaluate the robustness of models by means of four attack methods: Carlini & Wagner (C&W) (Carlini & Wagner, 2017), Fast Gradient Sign Attack (FGSM) (Goodfellow et al., 2014), Projected Gradient Descent (Madry et al., 2018) with the l_∞ (l_∞PGD) and l_2 (l_2PGD) norms. For our attacks, we employ the advertorch library (Ding et al., 2019). One of the theoretical strengths of the Gauss and DUQ confidence is the possibility to identify outliers. Since the softmax confidence of the predicted class is always larger than $\frac{1}{c}$, we also specify that a point is considered as an outlier if the confidence of the predicted class is smaller than $\frac{1}{c}$: if we have ten classes, then the outlier threshold is 0.1. We train all networks with the same number of epochs.

5.1 Classification Performance

We evaluate the feasibility of our proposed loss function to attain similar test set accuracies as softmax models. We employ for Gauss networks a warmup of 10 epochs on MNIST and Cifar-10, 30 epochs on Cifar-100, and 60 epochs on ImageNet. MNIST is trained for 60 epochs, while the other datasets are trained for 100 epochs (including the warmup). For softmax and Gauss networks we employ a similar learning rate schedule, and we use the advised learning rate schedule for DUQ networks (as provided for Fashion MNIST and Cifar-10). We train the LeNet architecture with a learning rate of 0.1 for softmax nets and 0.02 for Gauss nets, which is decayed by a factor of 10 at epoch 40. We train the ResNet-18 architecture with a learning rate of 0.1 for softmax nets and 0.02 for Gauss nets, which is then decayed by a factor of 10 at epoch 60 and 80.

Table 1 summarizes the performance results. For every dataset, we report the test accuracy and the average confidence of the predictions. We observe that Gauss networks achieve equivalent accuracies and similarly high confidences as softmax networks on MNIST and Cifar-10. On Cifar-100, we observe that the accuracy drops a bit but more notably, Gauss networks are less confident on these classes. While softmax networks are rather overconfident (the confidence is higher than the accuracy), Gauss networks tend to be less confident. This effect recurs on ImageNet: Gauss networks lose a few percentage points of accuracy w.r.t. softmax networks, but the latter have a higher confidence than accuracy. DUQ networks can compete on the MNIST and Cifar-10 datasets, but its training of Cifar-100 failed. The accuracy can possibly be improved by a tailored initialization. However, the low accuracy in comparison to the low loss value at the end of the training indicates the flaw of the BCE loss. For Cifar10, the BCE loss sums for each data point 9 terms that push the data point away from 9 classes. Yet, for Cifar100, the BCE loss sums for each data point 99 terms that push the point away from 99 classes – these terms overwhelm the one term that pushes towards the class centroid. In addition, the employed gradient penalty of DUQ networks quadruples the runtime. Consequently, we didn't try to optimize DUQ for ImageNet. Note that the runtime for one epoch is the same for Gauss and softmax nets.

[2] https://anonymous.4open.science/r/GaussNetworks-A7E7

Table 2: Results of Attacks on MNIST and Cifar-10. We report the rate of successful attacks (rate) and the average confidence of the adversarial samples (conf) for each attack. The displayed attacks are Carlini & Wagner (C&W) (Carlini & Wagner, 2017), Fast Gradient Sign Attack (FGSM) (Goodfellow et al., 2014), Projected Gradient Descent (Madry et al., 2018) with l_∞- (l_∞PGD) and with l_2-norm (l_2PGD). For all attacks we set $\epsilon = 0.1$.

		C&W		FGSM(0.1)		l_∞PGD(0.1)		l_2 PGD(0.1)	
Dataset	Network	rate	conf	rate	conf	rate	conf	rate	conf
MNIST	Softmax	2%	0.76	12%	0.75	23%	0.80	1%	0.54
	DUQ	0%	0.00	3%	0.33	44%	0.59	0%	0.00
	Gauss	0%	0.00	4%	0.44	20%	0.52	0%	0.00
Cifar-10	Softmax	95%	0.93	72%	0.85	95%	1.00	45%	0.93
	DUQ	96%	0.71	42%	0.31	97%	0.94	41%	0.96
	Gauss	67%	0.46	34%	0.68	58%	0.43	18%	0.80

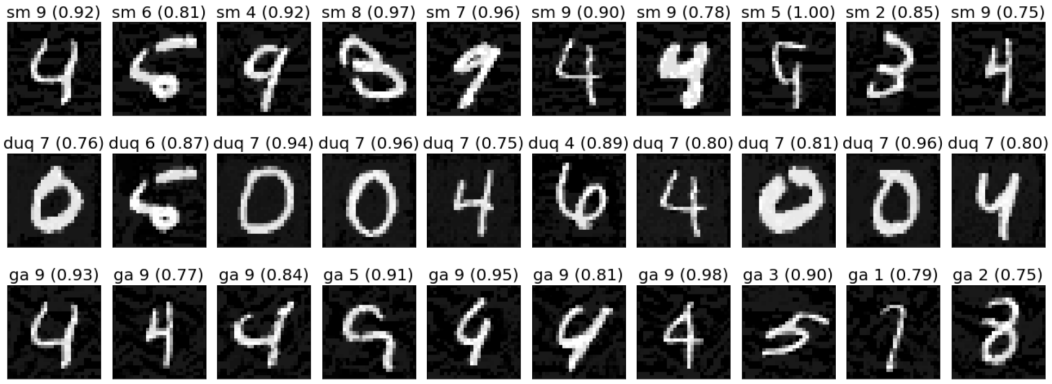


Figure 3: Adversarial examples achieved with the l_∞PGD attack on MNIST with confidence ≥ 0.7. Above each image you find the network (sm: softmax, duq: DUQ, ga: Gauss), the erroneously predicted class, and the confidence of the prediction in parentheses.

5.2 Robustness to Attacks

We investigate to which extent Gauss networks provide an inbuilt robustness (cf. Theorem 2) to four attacks on the MNIST and Cifar-10 datasets. Table 2 reports the fraction of images for which the attacks lead to misclassification, and the average confidence of the network for these misclassifications. For both measures, lower is better.

For all four attacks on both datasets in Table 2, the Gauss network outperforms the softmax network in terms of both attack success rate and confidence in the adversarial examples. On MNIST, the Gauss and DUQ networks have equivalent attack success rates for three out of four attacks; the only substantial difference is under the l_∞PGD(0.1) attack, where the Gauss network outperforms the DUQ network. On Cifar-10, the Gauss network outperforms the DUQ network in terms of attack success rate for all four attacks; in terms of confidence, the Gauss network outperforms the DUQ network on three attacks, while the DUQ network outperforms the Gauss network under the FGSM(0.1) attack.

The most successful Gauss network attacks are the l_∞PGD attack on MNIST and the C&W attack on Cifar-10. Confidently assigned adversarial examples are displayed in Figures 3 and 4. Adversarial examples on MNIST indicate that Gauss networks often confuse a four with a nine; these also are the classes with the nearest centroids (distance of 9.5; average centroid distance is 12.1 ± 1.4). The adversarial examples on Cifar-10 are less conclusive. While some of the adversarial examples for Gauss net are justifiable, other adversarial examples hardly make sense. This indicates that our proposed confidence measure does

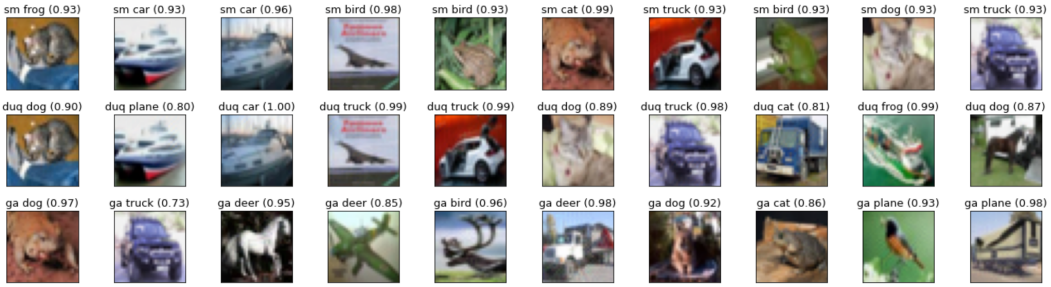


Figure 4: Adversarial examples achieved with the C&W attack on Cifar-10 with confidence ≥ 0.7. Above each image you find the network (sm: softmax, duq: DUQ, ga: Gauss), the erroneously predicted class, and the confidence of the prediction in parentheses.

not guarantee human-like conceptualization of images, but it provides a first step towards indications of model representations and knowledge.

6 Conclusions

We introduce a loss function to optimize for neural networks with Gauss confidence, a centroidal confidence measure for Deep Neural Networks, instead of the standard softmax classification and the recently proposed alternative of DUQ networks (van Amersfoort et al., 2020). We prove that softmax-based classifiers are nearest centroid classifiers (cf. Theorem 1). Softmax classification accuracy is typically insensitive to the distance of the points to its centroid: as long as a point lies in the correct Voronoi cell, softmax is confident of its decision. We prove that one can certify the robustness of the network through these distances (cf. Theorem 2). We build on this centroid-based robustness certificate to define our Gauss confidence (cf. Definition 1). Its islands of confidence are illustrated in the rightmost plot of Figure 1, in contrast to the wider confidence regions generated by the competitors.

Experiments on the four datasets MNIST, Cifar-10, Cifar-100, and ImageNet show that Gauss networks can match the confidence achieved by softmax networks (cf. Table 1): the median absolute difference between the two is 0.45 percentage points. Gauss networks do so while expressing a lower confidence than softmax, which occasionally displays a higher confidence than its accuracy. DUQ networks can compete on the datasets with ten classes (MNIST and Cifar-10), but not on the others (with substantially more classes).

Experiments with four forms of adversarial attacks on two datasets show that Gauss networks outperform softmax networks in terms of both attack success rate and confidence in the adversarial examples (cf. Table 2). DUQ networks perform equivalently to Gauss networks for three out of four attacks on the MNIST datasets; Gauss networks beat DUQ networks under the l_∞PGD(0.1) attack. On the Cifar-10 dataset, Gauss networks outperform DUQ networks for all four attacks in terms of attack success rate, and on three out of four attacks in terms of confidence in the adversarial examples. The high-confidence adversarial examples in Figure 3 illustrate that Gauss networks have the most trouble distinguishing 4s from 9s (which are the classes with the nearest centroids), DUQ networks confuse 0s with 7s, and softmax networks vary wildly in their misclassifications.

Gauss networks can achieve comparable accuracies to softmax networks, but are more robust against adversarial attacks. In some cases, DUQ networks match Gauss networks in their robustness (though Gauss networks are more robust on the Cifar-10 dataset), but DUQ networks cannot achieve the same accuracy as Gauss or softmax networks when the number of classes in the dataset is substantially larger than ten. In future work, we intend to improve Gauss networks by allowing for multiple centroids per class. This would allow for classes that are not one monolithic whole: they could consist of several parallel subsets, or even hierarchical concepts.

References

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2807385.

Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9690–9700. PMLR, 2020. URL http://proceedings.mlr.press/v119/van-amersfoort20a.html.

Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 274–283, 2018.

Ludwig Boltzmann. Studien über das gleichgewicht der lebendigen kraft zwischen bewegten materiellen punkten. *Wiener Berichte*, 58:517–560, 1868.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J. Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *CoRR*, abs/1902.06705, 2019. URL http://arxiv.org/abs/1902.06705.

Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

Yarin Gal. Uncertainty in deep learning - section 1.5. *University of Cambridge*, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *CoRR*, abs/1804.04368, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1321–1330, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv e-prints*, art. arXiv:1512.03385, Dec 2015.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pp. 2266–2276, 2017.

Marius Hobbhahn, Agustinus Kristiadi, and Philipp Hennig. Fast predictive uncertainty for classification with bayesian deep networks. In *Uncertainty in Artificial Intelligence*, pp. 822–832. PMLR, 2022.

Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the lipschitz constant as a defense against adversarial examples. In *Joint European conference on machine learning and knowledge discovery in databases (ECML PKDD)*, pp. 16–29. Springer, 2018.

Ozsel Kilinc and Ismail Uysal. Learning latent representations in neural networks for clustering through pseudo supervision and graph-based activity regularization. In *Proc. ICLR*, 2018.

Vignesh Kothapalli, Ebrahim Rasromani, and Vasudev Awatramani. Neural collapse: A review on modelling principles and generalization. *ArXiv*, abs/2206.04041, 2022.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report 1 (4), 7, University of Toronto, 2009.

Vadim Lebedev, Artem Babenko, and Victor Lempitsky. Impostor networks for fast fine-grained recognition. *arXiv preprint arXiv:1806.05217*, 2018.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, pp. 7, 2016.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pp. 6231–6239, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

NN, NN, and NN. Reference scrubbed for double-blind reviewing purposes. 2020.

Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen, and Jun Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. In *International Conference on Learning Representations*, 2019.

Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences of the United States of America*, 117:24652 – 24663, 2020.

Xi Peng, Joey Tianyi Zhou, and Hongyuan Zhu. k-meansnet: When k-means meets differentiable programming. *CoRR*, abs/1808.07292, 2018. URL http://arxiv.org/abs/1808.07292.

Haifeng Qian and Mark N. Wegman. L2-nonexpansive neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Achim Schilling, Claus Metzner, Jonas Rietsch, Richard Gerum, Holger Schulze, and Patrick Krauss. How deep is deep enough?–quantifying class separability in the hidden layers of deep neural networks. *arXiv preprint arXiv:1811.01753*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. URL http://arxiv.org/abs/1312.6199.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 6541–6550, 2018.

Weitao Wan, Yuanyi Zhong, Tianpeng Li, and Jiansheng Chen. Rethinking feature distribution for loss functions in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9117–9126, 2018.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pp. 499–515. Springer, 2016.

A Derivation of Equation (1)

$$
\begin{aligned}
&\arg\max_{Y} \operatorname{tr}(Y(W^\top D^\top + b\mathbf{1}_m^\top)) \\
&= \arg\max_{Y} \operatorname{tr}(YW^\top D^\top) + \operatorname{tr}(Yb\mathbf{1}_c^\top Y^\top)) \\
&= \arg\min_{Y} \|D\|^2 - 2\operatorname{tr}(YW^\top D^\top) - 2\operatorname{tr}(\mathbf{1}_m v^\top D^\top) + \operatorname{tr}(2b\mathbf{1}_c^\top Y^\top Y)) \\
&= \arg\min_{Y} \|D\|^2 - 2\operatorname{tr}(Y(W^\top + \mathbf{1}_c v^\top)D^\top) + \operatorname{tr}(2b\mathbf{1}_c^\top Y^\top Y)) \\
&= \arg\min_{Y} \|D\|^2 - 2\operatorname{tr}(YZ^\top D^\top) + \operatorname{tr}(YZ^\top ZY^\top) - \operatorname{tr}(YZ^\top ZY^\top) + \operatorname{tr}(2b\mathbf{1}_c^\top Y^\top Y)) \\
&= \arg\min_{Y} \|D - YZ^\top\|^2 + \operatorname{tr}((2b\mathbf{1}_c^\top - Z^\top Z)Y^\top Y))
\end{aligned}
$$

B Derivation of Equation (2)

$$
\begin{aligned}
\sum_{k=1}^{c}(2b\mathbf{1}_c^\top - Z^\top Z)_{kk}|Y_{\cdot k}| &= \sum_{k=1}^{c}(2b_k - Z_{\cdot k}^\top Z_{\cdot k})|Y_{\cdot k}| \\
&= \sum_{k=1}^{c}(2b_k - (W_{\cdot k} + v)^\top (W_{\cdot k} + v))|Y_{\cdot k}| \\
&= \sum_{k=1}^{c}(2b_k - \|W_{\cdot k}\|^2 - 2v^\top W_{\cdot k} - \|v\|^2)|Y_{\cdot k}| \\
&= \sum_{k=1}^{c}(2b_k - \|W_{\cdot k}\|^2 - 2v^\top W_{\cdot k})|Y_{\cdot k}| - \|v\|^2 m
\end{aligned}
$$