Denoising Autoencoders for Overgeneralization in Neural Networks

Giacomo Spigler, *Member, IEEE*

Abstract—Despite recent developments that allowed neural networks to achieve impressive performance on a variety of applications, these models are intrinsically affected by the problem of overgeneralization, due to their partitioning of the full input space into the fixed set of target classes used during training. Thus it is possible for novel inputs belonging to categories unknown during training or even completely unrecognizable to humans to fool the system into classifying them as one of the known classes, even with a high degree of confidence. This problem can lead to security problems in critical applications, and is closely linked to open set recognition and 1-class recognition. This paper presents a novel way to compute a confidence score using the reconstruction error of denoising autoencoders and shows how it can correctly identify the regions of the input space close to the training distribution. The proposed solution is tested on benchmarks of 'fooling', open set recognition and 1-class recognition constructed from the MNIST and Fashion-MNIST datasets.

Index Terms—overgeneralization, fooling, autoencoder, open set recognition, open world recognition, 1-class recognition, confidence score, neural networks

1 Introduction

Discriminative models in machine learning, like neural networks, have achieved impressive performance in a variety of applications. Models in this class, however, suffer from the problem of overgeneralization, whereby the whole input space is partitioned between the set of target classes specified during training, and generally lack the possibility to reject a novel sample as not belonging to any of those.

A main issue with overgeneralization is in the context of *open set recognition* [22] and *open world recognition* [5], where only a limited number of classes is encountered during training while testing is performed on a larger set that includes a potentially very large number of unknown classes that have never been observed before. An example is shown in Figure 1 where a linear classifier is trained to discriminate between handwritten digits '0' and '6'. As digit '9' is not present in the training set, it is here wrongly classified as '6'. In general, instances of classes that are not present in the training set will fall into one of the partitions of the input space learnt by the classifier. The problem becomes worse in real world applications where it may be extremely hard to know in advance all the possible categories that can be observed.

Further, the region of meaningful samples in the input space is usually small compared to the whole space. This can be easily grasped by randomly sampling a large number of points from the input space, for example images at a certain resolution, and observing that the chance of producing a recognizable sample is negligible. Yet, discriminative models may assign a high confidence score to such random images, depending on the learnt partition of the input space. This is indeed observed with *fooling* [16], for which it was shown to be possible to generate input samples that are unrecognizable to humans but get classified as a specific target class with high confidence (see example in Figure 1). Fooling in particular may lead to security problems in critical applications.

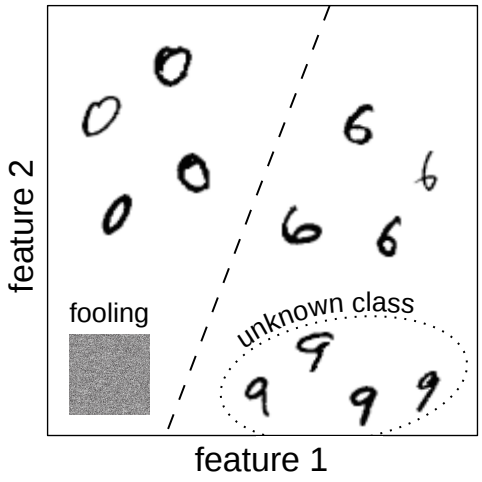

Fig. 1. A linear classifier is trained to recognize exclusively pictures of digits '0' and '6'. Digit '9' was never observed during training, but in this example it is wrongly classified as digit '6'. This is an example of overgeneralization. A similar problem is 'fooling', whereby it is possible to generate images that are unrecognizable to humans but are nonetheless classified as one of the known classes with high confidence, for example here the noise-looking picture in the bottom-left corner that is classified as digit '0'.

As suggested in [16], these problems may be mitigated or solved by using generative models, that rather than learning the posterior of the class label $P(y|X)$ directly, learn the joint distribution $P(y, X)$ from which $P(X)$ can be computed. Modeling the distribution of the data would then give a model the capability to identify input samples as belonging to known classes, and to reject those that are believed to

• *G. Spigler is with the Biorobotics Institute of Scuola Superiore Sant'Anna, Pisa, Italy.*
E-mail: giacomo.spigler@santannapisa.it

belong to unknown ones. Apart from mitigating the problem of overgeneralization, modeling the distribution of the data would also be useful for applications in novelty and outlier detection [14] and incremental learning [5], broadening the range of applications the same model could be used in.

Estimating the marginal probability $P(X)$ is, however, not trivial. Luckily, computing the full distribution may not be necessary. The results in this work suggest that identification of high-density regions close to the local maxima of the data distribution may be sufficient to correctly identify which samples belong to the distribution and which ones are to be rejected. Specifically, it is possible to identify and classify the critical points of the data distribution by exploiting recent work that has shown that in denoising [27] and contractive [21] autoencoders, the reconstruction error tends to approximate the gradient of the log-density. A measure of a confidence score can then be computed as a function of this gradient.

Here, a set of experiments is presented to compare the empirical performance of the proposed model with baselines and with the COOL (Competitive Overcomplete Output Layer) [11] model that has been recently applied to the problem of fooling.

2 Overview of Previous Work

The simplest way to limit overgeneralization in a given classifier is to set a threshold on the predicted outputs and rejecting any sample below its value (for example [10], [20]). The output of the model is thus treated as an estimate of the confidence score of the classifier. This approach, however, was shown to be sensitive to the problem of fooling [16]. Alternatively, a confidence score may be computed based on the k-Nearest-Neighbor algorithm (e.g., [8], [9], [30]) or the k-Means algorithm (e.g., [2], [15]), as a function of the distance of novel samples from the stored templates or centroids.

Another way to mitigate the problem is to use a training set of positive samples complemented with a set of negative samples that includes instances belonging to a variety of 'other' classes (e.g., [4]). This approach however does not completely solve the problem, and it is usually affected by an unbalanced training set due to the generally larger amount of negatives required [18]. As the potential amount of negatives can be arbitrarily large, a further problem consists in gathering an amount of data sufficient to approximate their actual distribution, which is made even worse by the fact that the full set of negative categories may not be known when training the system. For example, in the context of object recognition in vision, high-resolution images may represent any possible image class, the majority of which is likely not known during training. The use of negative training instances may nonetheless mitigate the effect of categories that are known to be potentially observed by the system.

The problem of overgeneralization is further present in the context of 'open set recognition', that was formally defined by Scheirer and colleagues [22]. In this framework, it is assumed that a classifier is trained on a set of 'known' classes and potentially on a set of 'known unknown' ones (e.g., negative samples). Testing, however, is performed on a larger set of samples that include 'unknown unknown' classes that are never seen during training. Models developed to address the problem of open set recognition focus on the problem of 'unknown unknown' classes [23]. The seminal paper that gave the first formal definition of the problem proposed the 1-vs-Set Machine algorithm as an extension to SVM that is designed to learn an envelope around the training data using two parallel hyperplanes, with the inner one separating the data from the origin, in feature space [22]. Scheirer and colleagues then proposed the Weibull-calibrated SVM (W-SVM) algorithm to address multi-class open set recognition [23]. Another interesting approach was recently applied to deep neural networks with the OpenMax model [6], that works by modeling the class-specific distribution of the activation vectors in the top hidden layer of a neural network, and using the information to recognize outliers.

Related to the problem of open set recognition is that of 'open world recognition', in which novel classes first have to be detected and then learnt incrementally [5]. This can be seen as an extension to open set recognition in which the 'unknown unknown' classes are discovered over time, becoming 'novel unknowns'. The new classes are then labelled, potentially in an unsupervised way, and become 'known'. The authors proposed the Nearest Non-Outlier (NNO) algorithm to address the problem.

A special case of open set recognition is 1-class recognition, in which training is performed on samples from a single class, with or without negative samples. The 1-Class SVM algorithm was proposed to address this problem [24], by fitting a hyperplane that separates all the data points from the origin, in feature space, maximizing its distance from the origin. The algorithm has been applied in novelty and outlier detection [25]. Variants of the algorithm like Support Vector Data Description (SVDD) have also been used to learn an envelope around points in the dataset [26]. Other systems have tried to estimate the boundaries of the data by computing the region of minimum volume in input space containing a certain probability mass [17].

For a more complete overview of methods proposed in the specific case of outlier detection we suggest the review by (Domingues et al., 2018) [8].

Finally, a specific sub-problem of overgeneralization is 'fooling' [16]. The "Competitive Overcomplete Output Layer" (COOL) model [11] was recently proposed to mitigate the problem of fooling. COOL works by replacing the final output layer of a neural network with a special COOL layer, constructed by replacing each output unit with ω ones (the degree of overcompleteness). The ω output units for each target class are then made to compete for activation by means of a softmax activation that forces them to learn to recognize different parts of the input space, overlapping only within the region of support of the data generating distribution. The network can then compute a confidence score as the product of the activation of all the units belonging to the same target class, that is high for inputs on which a large number of units agrees on, and low in regions far from the data distribution, where only few output units are active.

3 Proposed Solution

The solution presented here is based on a novel measure of confidence in the correct identification of data points as belonging to the training distribution, or their rejection.

Ideally, such a confidence score would be a function of the data probability $p(\mathbf{x})$. Computing the full distribution may however not be necessary. In particular, we suggest that for most applications, especially in computer vision, the problem can be simplified with the identification of points belonging to the data manifold as points that are close to local maxima of the data generating distribution, as follows.

It has been recently shown that trained denoising [27] and contractive [21] autoencoders implicitly learn features of the underlying data distribution [3], [7], specifically that their reconstruction error approximates the gradient of its log-density

$$\frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}} \propto r(\mathbf{x}) - \mathbf{x} \tag{1}$$

for small corruption noise ($\sigma \to 0$). $r(\mathbf{x}) = Dec(Enc(\mathbf{x}))$ is the reconstructed input. Larger noise is however found to work best in practice. The result has been proven to hold for any type of input (continuous or discrete), any noise process and any reconstruction loss, as long as it is compatible with a log-likelihood interpretation [7]. A similar interpretation suggested that the reconstruction error of regularized autoencoders can be used to define an energy surface that is trained to take small values on points belonging to the training distribution and higher values everywhere else [29].

Thus, critical points of the data distribution correspond to points with small gradient of the log-density, that is small reconstruction error (Equation 1). Those are indeed points that the network can reconstruct well, and that it has thus hopefully experienced during training or has managed to generalize to well. Conversely, samples from the data distribution, that are trained to achieve small reconstruction error, are characterized by small gradient of the log-density of the data distribution, and are thus extrema points, in agreement with the initial hypothesis that data samples could be identified by their proximity to local maxima of the distribution. A confidence score can thus be designed that takes high values for points on the data manifold, that is points near the local maxima of the log-density of the data distribution, and small values everywhere else.

We note however that this approach cannot distinguish between local minima, maxima or saddle points (Figure 2 shows such an example), and may thus assign a high confidence score to a small set of points not belonging to the target distribution. Here the problem is addressed by scaling the computed confidence by a function $\Gamma(\mathbf{x})$ that favours small or negative curvature of the log-density of the data distribution, which can in turn be computed from the diagonal of the Hessian, estimated from the Jacobian of the reconstruction function as shown in [3]

$$\frac{\partial^2 \log p(\mathbf{x})}{\partial \mathbf{x}^2} \propto \frac{\partial r(\mathbf{x})}{\partial \mathbf{x}} - I \tag{2}$$

A variety of functions may be defined with the desired characteristics, exploiting Equations 1 and 2. One possible way, that we will use throughout this paper, is to compute the confidence score $\tilde{c}(\mathbf{x})$ as

$$\tilde{c}(\mathbf{x}) = \exp\left(-\frac{\alpha}{D}\|r(\mathbf{x}) - \mathbf{x}\|_2\right)\Gamma(\mathbf{x}) \tag{3}$$

$$\Gamma(\mathbf{x}) = \begin{cases} 1 & \text{if } \gamma(\mathbf{x}) \leq 0 \\ \exp(-\beta\gamma(\mathbf{x})) & \text{if } \gamma(\mathbf{x}) > 0 \end{cases} \tag{4}$$

$$\gamma(\mathbf{x}) = \frac{1}{D}\sum_i \left(\frac{\partial r_i(\mathbf{x})}{\partial x_i} - 1\right) \tag{5}$$

where D is the dimensionality of the inputs $\mathbf{x} = (x_1, x_2, \ldots, x_D)$, α a parameter that controls the sensitivity of the function to outliers and β a parameter that controls the sensitivity to $\gamma(\mathbf{x})$, which is proportional to the average of the diagonal elements of the Hessian of the log-density at x (from Equation 2). High values of α yield stricter confidence scores, at the expense of potentially discarding valid data. Low values of α, however, may be susceptible to a degree of overgeneralization.

The first component of $\tilde{c}(\mathbf{x})$ identifies the extrema points of the log-density of the data (from Equation 1), while $\Gamma(\mathbf{x})$ is used to limit high values of the confidence scores to the maxima only (i.e., to points predicted to lie near the data manifold). Note that Equations 3,4 and 5 do not require modifications to the autoencoder, but only need access to the learnt reconstruction function $r(\mathbf{x})$.

A classifier can finally be modified by scaling its predicted output probabilities y by $\tilde{c}(\mathbf{x})$ computed using a denoising autoencoder trained together with the classifier

$$\tilde{\mathbf{y}} = \tilde{c}(\mathbf{x})\mathbf{y} \tag{6}$$

If the outputs of the classifier are normalized, for example using a softmax output, this can be seen as introducing an implicit 'reject/other' class with value $1 - \tilde{c}(\mathbf{x})$.

In the experiments presented here, the classifier is constructed as a fully connected softmax layer attached on top of the top hidden layer of an autoencoder with symmetric weights (i.e., attached to the output of the encoder), in order to keep the number of weights similar (minus the bias terms of the decoder) to an equivalent feed-forward benchmark model, identical except for its lack of the decoder. In general, keeping the autoencoder separate from the classifier or connecting the two in more complex ways will work, too, as well as using a classifier that is not a neural network. In case the autoencoder and the classifier are kept separate, the autoencoder is only used to infer information about the data distribution. Pairing the systems together, however, might provide advantages outside the scope of the present work, like enabling a degree of semi-supervised learning. The autoencoder may also be further improved by replacing it with the discriminator of an EBGAN [29] to potentially learn a better model of the data.

4 Experiments

4.1 2D example

The model was first tested on a 2D classification task to visualize its capacity to learn the support region of the input space of each training class. Three target distributions were defined as uniform rings with thickness of 0.1, inner radius of 0.6 and centers $(-1, 1)$, $(1, 1)$ and $(1, -1)$. The training distributions are shown in Figure 2A. Training was performed with minibatches of size 64 using the Adam

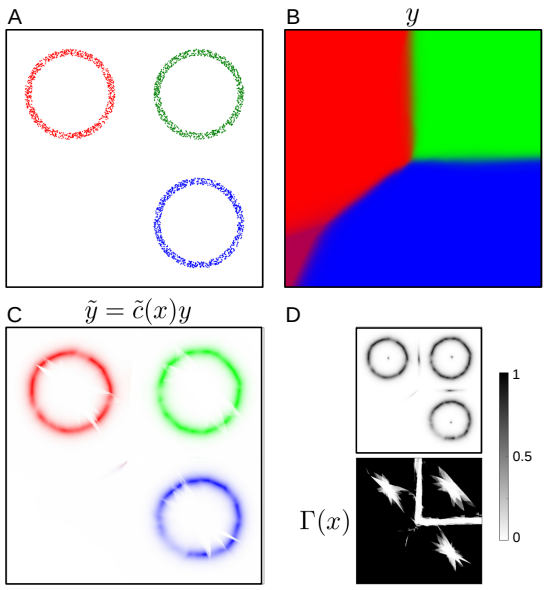

Fig. 2. The system presented here is trained to classify points sampled from three uniform ring distributions. **A.** 1000 data points are sampled from each of the target distributions. **B.** Labeling y of each point in the input space without scaling of the classifier's output by the confidence score. **C.** Labeling $\tilde{y}$ of each point in the in=put space scaled by the computed confidence score. Regions in white are assigned low confidence scores. **D.** Top: confidence score without $\Gamma(\mathbf{x})$. Bottom: estimate of the curvature of the log-distribution of the data ($\Gamma(\mathbf{x})$). The confidence score $\tilde{c}(\mathbf{x})$ is the product of the two functions. The panel in **B** is the product of the classifier's output (**C**) and the confidence score.

optimizer [12] for a total of 50000 update steps. As shown in Figure 2, the model learned to correctly identify the support region of the target distributions. On the contrary, the uncorrected classifier partitioned the whole space into three regions, incorrectly labeling most points. The confidence score computed by the model presented here helps the system to prevent overgeneralization by limiting the decisions of the classifier to points likely to belong to one of the target distributions.

4.2 Fooling

The model presented in this paper was next tested on a benchmark of fooling on the MNIST [13] and Fashion-MNIST [28] datasets similar to the one proposed in [11]. However, contrary to the previous work, the classification accuracy of the models is reported as a 'thresholded classification accuracy', which considers the classification of test samples with correctly predicted labels as correct only if the corresponding scaled output $\tilde{y}$ is above the same threshold used to consider a fooling instance as valid. This metric should indeed be reported alongside the fooling rate for each model, as otherwise a model that trivially limits the confidence scores of a network to a fixed value lower than the threshold used to consider fooling attempts to be valid would by definition never be fooled. The same model would however never classify any valid sample above that same threshold. This metric thus proves useful to compare different models with varying degrees of sensitivity to overgeneralization.

The fooling test was performed by trying to fool a target network to classify an input that is unrecognizable to humans into each target class (digits 0 to 9). The fooling

TABLE 1
MNIST fooling results

Model	Accuracy			Fooling Rate (Avg Steps)	
	0%	90%	99%	90%	99%
CNN	99.35%	99.23%	99%	100% (63.5)	99% (187.1)
COOL	99.33%	98.1%	93.54%	34.5% (238.8)	4.5% (313.4)
dAE sym	98.98%	98.11%	96.8%	**0% (-)**	**0% (-)**
dAE asym	99.14%	98.41%	97.63%	**0% (-)**	**0% (-)**

instances were generated using a Fooling Generator Network (FGN) consisting of a single layer perceptron with sigmoid activation and an equal number of input and output units (size of $(28, 28)$ here). Most importantly, the FGN produces samples with values bounded in $(0, 1)$ without requiring an explicit constraint. Fooling of each target digit was attempted by stochastic gradient descent on the parameters of the FGN to minimize the cross-entropy between the output of the network to be fooled and the specific desired target output class. Fooling of each digit was attempted for 20 trials with different random inputs to the FGN, each trial consisting of up to 10000 parameter updates, as described in [11].

In the first test we compared three models, a plain Convolutional Neural Network (CNN), the same CNN with a Competitive Overcomplete Output Layer (COOL) [11], and a network based on the system described in Section 3, built on the same CNN as the other two models with the addition of a decoder taking the activation of the top hidden layer of the CNN as input, to compute the dAE-based confidence score $\tilde{c}(\mathbf{x})$. The denoising autoencoder (dAE) was trained with corruption of the inputs by additive Gaussian noise. All the models were trained for a fixed 100 epochs. Fooling was attempted at two different thresholds, 90% and 99%, in contrast to the previous work that used only the 99% one [11]. Comparing the models at different thresholds gives more information about their robustness and may amplify their differences, thus improving the comparison. Tables 1 and 2 report the results for the three models, with the further splitting of the denoising autoencoder model in two separate cases, using either a separate decoder (*dAE asym*) or building the decoder as a symmetric transpose of the encoder (*dAE sym*). Fooling was measured as the proportion of trials (200 total, 20 repetitions of 10 digits) that produced valid fooling samples within the maximum number of updates. The average number of updates required to fool each network is reported in parentheses. The full set of parameters used in the simulations is reported in Appendix b. The model presented here outperformed the other two at both thresholds, while also retaining a high thresholded classification accuracy, even at high thresholds. As in the previous protocol [11], the cross-entropy loss used to optimize the FGN was computed using the unscaled output y of the network.

The symmetric and asymmetric autoencoders were both found to achieve perfect fooling performance (0% fooling rate) and similar accuracies on MNIST. As the asymmetric autoencoder requires twice as many parameters, but does not yield major improvements, the simpler symmetric model was used for all the remaining experiments, so that the three models had a similar number of parameters ($1.31M$ for CNN

TABLE 2
Fashion-MNIST fooling results

Model	Accuracy			Fooling Rate (Avg Steps)	
	0%	90%	99%	90%	99%
CNN	91.65%	90.91%	89.27%	100% (113.0)	30.5% (902.0)
COOL	91.23%	87%	65.3%	**0% (-)**	**0% (-)**
dAE sym	91.59%	77.8%	64.87%	**0% (-)**	**0% (-)**

and dAE, $1.35M$ for COOL).

We further observed that the results in Table 1 were different from those reported in [11]. Specifically, the fooling rate of the COOL was found to be significantly lower than that reported (47%), as well as the average number of updates required to fool it (more than 5000). The major contributor to this difference was found to be the use of Rectified Linear Units (ReLUs) in the experiments reported here, compared to sigmoid units in the original study. This was shown in a separate set of simulations where all the three models used sigmoid activations instead of ReLUs and a fixed fooling threshold of 99%. In this case the thresholded classification accuracy of the models was slightly higher (98.39% for the plain CNN, 96.55% for COOL, and 96.58% for dAE), but it was matched with a significant increase in the fooling rate of the COOL model ($95.5\%(2203.9)$; plain CNN $91\%(519.2)$, dAE 0%). Other variations in the protocol that could further account for the differences found could be the different paradigm for training (100 fixed training epochs versus early stopping on a maximum of 200 epochs) and a slightly different network architecture, that in the present work used a higher number of filters at each convolutional layer.

Next, the effect of the learning rate used in the fooling update steps was investigated by increasing it from the one used in the previous study ($\eta = 0.00001$) to the one used to train the models $\eta = 0.001$, expecting a higher fooling rate. The threshold was set to 90%. Indeed, the plain CNN was found to be fooled on 100% of the trials in just 2.66 updates, while the dAE based model was still never fooled. COOL, on the other hand, significantly decreased in performance, with a fooling rate of 56.5% (878.3 average updates).

Finally, the COOL and dAE models were tested by attempting to fool their confidence scores directly, rather than their output classification scores, in contrast to [11] (i.e., using $\tilde{\mathbf{y}}$ instead of $\mathbf{y}$ for the cross-entropy loss used to update the FGN). A threshold of 99% was used. Interestingly, the COOL model was never fooled, while the model described here was fooled on 1% of the trials, although requiring a large number of updates (5470.8 on average). Also, it was found that while adding L_2 regularization to the weights of the dAE model led to a significantly higher fooling rate (100% rate in 6500.3 average updates for $\lambda_{L_2} = 10$), the generated samples actually resembled real digits closely, and could thus not be considered examples of fooling. This shows that the dAE model, when heavily regularized, is capable of learning a tight boundary around the high density regions of the data generating distribution, although at the cost of reducing its thresholded accuracy (87.84% for $\lambda_{L_2} = 10$). The set of generated samples is shown as Supplementary Figure D for $\lambda_{L_2} = \{10, 100\}$.

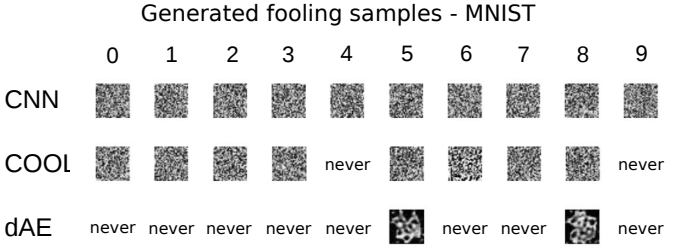

Fig. 3. Visualization of a set of generated fooling samples from the main results of Table 1 (MNIST). The samples from the plain CNN and the COOL models were computed by trying to fool each system's output classification scores above a threshold of 90%. As fooling was unsuccessful on the dAE model in this case, the results reported here were taken from the simulations in which fooling was performed directly on the output scaled by the confidence score ($\tilde{\mathbf{y}}$). Classes for which fooling was never successful within the maximum number of fooling iterations are marked as "never".

An example of the generated fooling samples is reported in Figure 3, showing instances from the main results of table 1 for the plain CNN and COOL, and for the experiment with fooling the confidence scores directly for the dAE model.

4.3 Open Set Recognition

The three models that were tested on fooling, a plain CNN, COOL [11] and the dAE model described in this paper were next compared in the context of open set recognition.

Open set recognition was tested by building a set of classification problems with varying degrees of openness based on the MNIST and Fashion-MNIST datasets. Each problem consisted in training a target model only on a limited number of 'known' training classes (digits) and then testing it on the full test set of 10 digits, requiring the model to be able to reject samples hypothesized to belong to 'unknown' classes. The degree of openness of each problem was computed similarly to [22], as

$$openness = 1 - \sqrt{\frac{num\_training\_classes}{num\_total\_classes}}$$

where $num\_training\_classes$ is the number of 'known' classes seen during training and $num\_total\_classes$ is 10 for both datasets. A high value of openness reflects a larger number of unknown classes seen during testing than that of classes experienced during training. The number of training classes was varied from 1 to 10, reflecting the full range of degrees of openness offered by the dataset.

For each fixed number of training classes used in training, the models were trained for 10 repetitions on different random subsets of the digits, to balance between easier and harder problems depending on the specific digits used. The same subsets of digits were used for all the three models. Correct classification was computed as a correct identification of the class label and a confidence score above a classification threshold of 99%, while correct rejection was measured as either assigning a low classification score (below 99%) or classifying the input sample as any of the classes not seen during training (for simplicity, the networks used a fixed number of output units for all the problems, with the target outputs corresponding to the 'unknown' classes always set

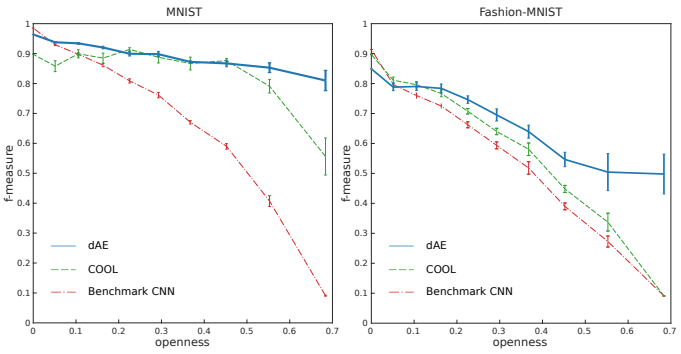

Fig. 4. Comparison of the three models on a benchmark of *open set recognition*. The F-measure was computed for each model on problems created from the MNIST (left) and Fashion-MNIST (right) datasets by only using a limited number of 'known' classes during training while testing on the full test set (e.g., training on classes 0 and 3 but testing on all classes $[0, 9]$), requiring the models to be able to reject samples belonging to 'unknown' classes. Higher values for the openness of a problem reflect a smaller number of classes used during training. The curves are averaged across 10 runs using different sub-sets of digits. Error bars denote standard deviation.

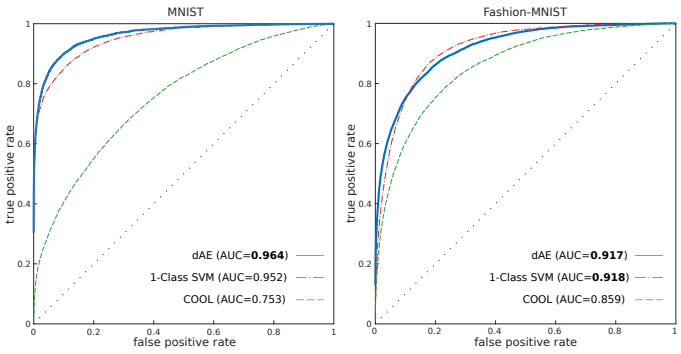

Fig. 5. ROC curves averaged over 10 *1-class recognition problems*, one for each class in MNIST (left) and Fashion-MNIST (right), for three models, the dAE model described in this paper, 1-Class SVM [24] and COOL [11].

to zero). The models were trained for a fixed 100 epochs for each task.

Figure 4 reports the results of the experiment. Like in the previous published benchmarks on open set recognition [6], [22], [23], the performance of the models for each degree of openness (indexed by i) was computed as the F-measure, the harmonic mean of the precision and recall scores, averaged across all the repetitions for the same degree of openness.

$$F_i = 2 \times \frac{precision_i \times recall_i}{precision_i + recall_i}$$

Results from a similar experiment with a lower threshold of 90% are available as Supplementary Figure F.

4.4 1-Class Recognition

The limit of open set recognition in which a single training class is observed during training, that is the problem of 1-class recognition, was next explored, comparing the model presented in this paper with COOL [11] and 1-Class SVM [24].

A separate 1-class recognition problem was created from the MNIST and Fashion-MNIST datasets for each target class. For each problem the models were trained using only samples from the selected class, while they were tested on the full test set of 10 digits. No negative samples were used during training. Each model was trained for 100 epochs on each problem.

Figure 5 shows the results as a ROC curve averaged over the curves computed for each of the 10 1-class recognition problems. The dAE based model outperforms the other two, with an Area Under the Curve (AUC) of 0.964, compared to $AUC = 0.952$ of 1-Class SVM and $AUC = 0.753$ of COOL.

5 Discussion

The confidence score that was introduced in this paper was found to perform better than a set of competing models in open set recognition and 1-class recognition. The system was also found to be significantly more robust to the problem of fooling than the state of the art COOL model. Together, these results show that it is possible to use information about the data generating distribution implicitly learnt by denoising autoencoders in meaningful ways, even without explicitly modeling the full distribution.

It is to be noted that when comparing the results to the COOL model we used the same degree of overcompleteness ($\omega = 10$) as in the original paper. However, fine tuning of the parameter and in particular using higher values may achieve higher performance on the benchmarking tasks used here. Also, similarly to the original COOL paper, fooling was attempted on the output of the classifier, rather than directly on the confidence scores. This gives an advantage to systems in which the confidence score is computed in more complex ways, not directly dependent on the output of the classifier. However, further tests as presented in Section 4.2 showed that the system presented here significantly outperforms the other models even when fooling is attempted directly on the confidence scores. In this particular case, it was further found that training the denoising autoencoder with heavy regularization resulted in generated samples resembling real digits, thus showing that the model had learnt a tight boundary around the data manifold.

It is interesting that the Energy-Based GAN (EBGAN) [29] makes use of the reconstruction error of a denoising autoencoder in a way compatible with the interpretation proposed here. In particular, it uses it as an approximated energy function that is learnt by the autoencoder to take low values for points belonging to the training distribution and high values everywhere else. As we have seen in Equation 1, it has been shown that the reconstruction error of denoising autoencoders is proportional to the gradient of the log-density of the data. Thus, small absolute values of the reconstruction error correspond to extrema points of the distribution, not limited to local maxima but also including minima and saddle points. If Figure 2 were a good example of the dynamics of the system even on more complex data, then the problem of local minima and saddle points may be limited. However, if that was not the case, then EBGAN might learn to generate samples from regions of local minima of the data distribution, which may not be desirable. It would be interesting to modify the system using the $\Gamma(\mathbf{x})$ function described here (Equation 4) in order to correctly isolate only

the local maxima of the distribution.

It would also be interesting to apply the regularization function used in EBGAN to the present model, adding a Pulling-away Term (PT) that forces learnt representations to be maximally different for different data points, by attempting to orthogonalize each pair of samples in a minibatch [29]. The stronger regularization may help the denoising autoencoder to learn a better representation of the data manifold, thus improving the confidence score $\tilde{c}(\mathbf{x})$.

Further improvements in the performance of the system may be achieved by separating the classifier and the denoising autoencoder, although combining the two may have other advantages, like adding a degree of semi-supervised learning or regularization of the autoencoder. It may also be possible to train an autoencoder to reconstruct hidden representations produced by pre-trained models, thus relying on more stable feature vectors rather than high-dimensional raw inputs.

6 Conclusion

This paper presented a novel approach to address the problem of overgeneralization in neural networks by pairing a classifier with a denoising or contractive autoencoder that is used to compute a confidence score that assigns high values only for input vectors likely to belong to the training data distribution. In particular, recognition of an input as belonging to the distribution is performed by using an approximation of the gradient of the log-density and its curvature at the specific input point, and using this information to determine whether it lies close to a local maximum of the distribution.

We have further explored the application of the system in the context of open set recognition. In general, the model presented here could be used in more complex architectures to allow for incremental and continual learning, by learning to recognize the regions of input space that have already been explored and learnt and potentially provide for different training regimes in the unexplored parts, in case new samples from those regions were to be observed in the future. For example, it may be applied to a system to allow for adding novel target classes even after deployment, without requiring a full re-training that may be costly in terms of compute time required, especially for large models. Similar to open set recognition is also 1-class recognition, that has proven to be a challenging problem. Building systems capable of robust 1-class recognition has critical applications in the detection of novelty, outliers and anomalies.

In conclusion, developing discriminative models capable of capturing aspects of the data distribution, even without explicitly modeling it, can prove very useful in a large number of practical applications, and future work on the topic will be highly beneficial. Here a system was presented to address the problem and was shown to perform better than other previously proposed systems on a set of benchmarks.

APPENDIX

Details of the simulations

The models were trained on a cross-entropy loss by Stochastic Gradient Descent using the ADAM algorithm [12] with $\eta = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Tensorflow [1] was used for the experiments. The parameters α and β (Equations 3 and 4) were empirically tuned to achieve a compromise between robustness to overgeneralization and decrease in the thresholded accuracies for the different datasets and threshold levels.

6.1 2D example

The dAE model used parameters $\alpha = 40$, $\beta = 5$ and $\sigma = 0.2$, and a symmetric denoising autoencoder with inputs of size 2 and two hidden layers both of size 200. The classifier was a fully-connected layer attached to the top hidden layer of the autoencoder and had 3 output units. Training was performed for 50000 steps with minibatches of size 64. The three target distributions were defined as uniform rings with thickness of 0.1 and inner radius of 0.6, centered at the three points $(-1, 1)$, $(1, 1)$ and $(1, -1)$.

6.2 Fooling

The models compared are a regular CNN, the same CNN with the output layer replaced with a COOL layer (degree of overcompleteness $\omega = 10$, as in [11]), and the same CNN with the addition of a decoder connected to the top hidden layer of the CNN, to complete the denoising autoencoder used to compute the confidence score $\tilde{c}(\mathbf{x})$. The architecture of the CNN is $\{Conv2D(1 \rightarrow 32, 5\times5),\ MaxPool(2\times2),\ Conv2D(32 \rightarrow 64, 5 \times 5),\ MaxPool(2 \times 2),\ FullyConnected(64 \rightarrow 400),\ FullyConnected(400 \rightarrow 10)\}$. Each layer is followed by a ReLU non-linearity, except for the output layer that is followed by a softmax. Fooling was attempted for 20 times for each digit, each for up to 10000 update steps with a learning rate for updating the FGN set to $\eta = 0.00001$ as in [11]. Training was performed for 100 epochs for each model. The dAE model was trained with additive Gaussian noise with zero mean and $\sigma = 0.2$ for MNIST, $\sigma = 0.1$ for Fashion-MNIST, and parameters $\beta = 10$ and α variable depending on the threshold used ($\alpha = 20$ for the 90% classification threshold, $\alpha = 3$ for the 99% threshold on MNIST, and $\alpha = 2$ for the 99% threshold on Fashion-MNIST). All models trained on the Fashion-MNIST dataset used L_2 regularization with $\lambda_{L_2} = 10$ (i.e., CNN, COOL and dAE).

6.3 Open Set Recognition

The Open Set Recognition tests used the same models as for the MNIST fooling ones, with a single threshold of 99%.

6.4 1-Class Recognition

The COOL and dAE models used the same parameters as the other experiments, except for the MNIST experiments in which L_2 regularization of the weights was used ($\lambda_{L_2} = 10$) for the dAE model, as well as $\sigma = 0.3$. 1-Class SVM was trained using the scikit-learn library [19], and used $\nu = 0.1$ and an RBF kernel ($\gamma = 0.1$).

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[2] A. Ahmad and L. Dey, "A k-mean clustering algorithm for mixed numeric and categorical data," *Data & Knowledge Engineering*, vol. 63, no. 2, pp. 503–527, 2007.

[3] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3563–3593, 2014.

[4] H. Barakat and D. Blostein, "Training with positive and negative data samples: effects on a classifier for hand-drawn geometric shapes," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*. IEEE, 2001, pp. 1017–1021.

[5] A. Bendale and T. Boult, "Towards open world recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1893–1902.

[6] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1563–1572.

[7] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Advances in Neural Information Processing Systems*, 2013, pp. 899–907.

[8] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses," *Pattern Recognition*, vol. 74, pp. 406–421, 2018.

[9] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3. IEEE, 2004, pp. 430–433.

[10] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[11] N. Kardan and K. O. Stanley, "Mitigating fooling with competitive overcomplete output layer neural networks," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 518–525.

[12] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[14] M. Markou and S. Singh, "Novelty detection: a review - part 1: statistical approaches," *Signal processing*, vol. 83, no. 12, pp. 2481–2497, 2003.

[15] G. Münz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," in *GI/ITG Workshop MMBnet*, 2007, pp. 13–14.

[16] A. Nguyen, J. Yosinski, and J. 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*, 2015, pp. 427–436.

[17] C. Park, J. Z. Huang, and Y. Ding, "A computable plug-in estimator of minimum volume sets for novelty detection," *Operations Research*, vol. 58, no. 5, pp. 1469–1480, 2010.

[18] H. Parvin, B. Minaei-Bidgoli, and H. Alinejad-Rokny, "A new imbalanced learning and dictions tree method for breast cancer diagnosis," *Journal of Bionanoscience*, vol. 7, no. 6, pp. 673–678, 2013.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] P. J. Phillips, P. Grother, and R. Micheals, "Evaluation methods in face recognition," in *Handbook of face recognition*. Springer, 2011, pp. 551–574.

[21] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 833–840.

[22] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.

[23] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014.

[24] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[25] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.

[26] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[27] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[28] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[29] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[30] M. Zhao and V. Saligrama, "Anomaly detection with score functions based on nearest neighbor graphs," in *Advances in neural information processing systems*, 2009, pp. 2250–2258.

Giacomo Spigler was born in 1990. He received the B.Sc. (Hons.) Degree and Diploma (Hons.) in computer engineering from the University of Pisa and Scuola Superiore Sant'Anna, respectively, the M.Sc. (Hons.) in cognitive science (computational neuroscience and neuroinformatics) from the University of Edinburgh, and a Ph.D. in computational neuroscience at the University of Sheffield. He is currently a post-doctoral researcher at the Biorobotics Institute of the Scuola Superiore Sant'Anna. His current research interests include deep neural networks, meta-learning and continual learning.