Aalto University
School of Science
Bachelor's Programme in Science and Technology

Recent developments in out-of-distribution detection for neural networks

Bachelor's Thesis

16 December 2020

Tuomas Myllymäki

Aalto-yliopisto	KANDIDAATINTYÖN
Perustieteiden korkeakoulu	TIIVISTELMÄ
Teknistieteellinen kandidaattiohjelma	

Tekijä:	Tuomas Myllymäki
Työn nimi:	Recent developments in out-of-distribution detection for neural networks
Päiväys:	16 December 2020
Sivumäärä:	21
Pääaine:	Tietotekniikka
Koodi:	SCI3027
Vastuuopettaja:	Professori Eero Hyvönen
Työn ohjaaja(t):	Tohtorikandidaatti Diego Mesquita (Tietotekniikan laitos)

Neuroverkoilla pystytään opettamaan tietokone tunnistamaan ja luokittelemaan sille annettu syöte. Neuroverkko voidaan esimerkiksi kouluttaa luokittelemaan koirien ja kissojen kuvia. Kandityöni käsittelee erästä neuroverkkojen käytännön ongelmaa: miten neuroverkko voi havaita jos sille annettu syöte ei vastaa sen koulutukseen käytettyjä syötenäytteitä? Miten kissoja ja koiria luokitteleva neuroverkko huomaisi että sille annettiin kuva männystä? Tämän ongelman ratkaiseminen on hyvin tärkeää, koska se mahdollistaisi neuroverkkojen yleisemmän käytön kriittisissä sovellusalueissa, kuten finanssialalla ja lääketeollisuudessa.

Työni on kirjallisuustutkimus, jossa käsittelen erilaisia ehdotettuja ratkaisuja tähän väärien syötteiden tunnistamisongelmaan. Työni kuvailee ja analysoi muutamia ehdotettuja ratkaisuja. Tavoitteena on luoda aiheesta kattava peruskuva, jonka perusteella voidaan helposti valita parhaiten soveltuva ratkaisu tiettyyn ongelmaan.

Tapoja havaita vääriä syötteitä on monia. Esimerkiksi eräs ehdotettu lähestymistapa on analysoida neuroverkon rakenteen käyttäytymistä oikeiden syötteiden kohdalla.

Kandidaatintyön pohjalta voidaan todeta että eri tavoissa havaita vääriä syötteitä on erilaisia rajoitteita ja vaatimuksia. Toisaalta, voidaan sanoa että lähestymistavoissa on myös paljon samaa, esimerkiksi kaikki käsittelemäni tavat havaitsevat väärät syötteet tarkastelemalla jotain muunnettua esitystä syötedatasta.

Kaikenkaikkiaan johtopäätöksenä on se että monenlaisia ratkaisuja on kehitetty. Eräs mahdollinen tulevaisuuden tutkimusalue voisi olla vastaavan ongelman ratkaiseminen muissa neuroverkkoja hyödyntävissä ongelmissa, esimerkiksi regressiossa.

Avainsanat:	neuroverkot, koneoppiminen, normaalijakauma
Kieli:	Englanti

Contents

1 Introduction **4**

2 Neural Networks **5**

3 Problem statement **6**

4 Methods **7**

4.1 Gram's matrices 7

4.2 ODIN 9

4.3 Generalized ODIN 11

4.4 Gaussian discriminant analysis & Mahalanobis 13

4.5 Energy-based approach 16

5 Conclusion **19**

References **21**

1 Introduction

Neural Networks (NNs) are deployed every day in a myriad of software systems, which might affect the lives of millions of people. They are utilized in large variety of practical problems, such as image recognition (Lecun et al., 1989), natural language processing (Bengio et al., 2003) and predicting weather forecasts (Riemer et al., 2016). Nonetheless, NNs often produce overconfident and possibly erroneous predictions for inputs outside the training dataset. For example, a neural network trained to differentiate cats and dogs from images does not expect the input to be an image of a turtle and outputs a very confident wrong prediction.

One approach for avoiding these critical failures is to detect *out-of-distribution* (OOD) inputs, i.e., inputs that differ greatly from the training data of the neural network. From a probabilistic point-of-view, OOD inputs lie in regions where the distribution of training inputs has low density.

Overall, the purpose of this thesis is to serve as an introduction on the different proposed solutions and to provide some different perspectives on the existing research. This thesis aims to give the reader an analytical but intuitive understanding of the main mechanisms utilized by some proposed methods. However, this thesis does not cover all of the proposed methods and does not discuss any other machine learning architectures than neural networks. In other words, the scope of the research includes only a small subset of the solutions proposed by academics and focuses only on the specific architecture of NNs.

OOD detection is currently an active research topic. Therefore, there is a large collection of different approaches available in the literature. This thesis discusses a small subset of the most prominent approaches and gives the reader an overview of the current state-of-the-art. The reader should be able to use the knowledge gained from this thesis to decide which approaches work best for a specific problem.

The following methods are covered in this thesis:

- Gram's matrices (Shama Sastry and Oore, 2019)
- Out-of-DIstribution detector for Neural networks (ODIN, Liang et al., 2018)
- Generalized ODIN (Hsu et al., 2020)
- Gaussian discriminant analysis & Mahalanobis (Lee et al., 2018)
- Energy-based approach (Liu et al., 2020)

To answer these questions for each method, the thesis first covers the intuition behind the idea and the mechanism. Secondly, the thesis discusses the possible requirements of the method, for example does the method require training samples that are out-of-distribution. Finally, the thesis discusses the possible advantages and drawbacks of this particular method in practical use.

The remaining of this thesis is organized as follows. First, the thesis introduces the topic and provides a little bit of background information. After that, the thesis covers some of the methods utilized for detecting *out-of-distribution* input samples. For each method, thesis covers the main intuition and mechanism, requirements for training data, and strengths and weaknesses. Finally, the thesis provides a small conclusion, which draws from the discussed methods and provides an overall picture of the current methods utilized to solve this problem.

2 Neural Networks

To provide context for the reader, it is forth to cover some background information on neural networks applied to classification tasks. For a more comprehensive introduction to neural networks and how to train them, the reader should refer to the book by Goodfellow et al..

According to the book by Goodfellow et al. (2016) a neural network is a function $f : \mathbb{R}^N \to \mathbb{R}^K$ where N is the dimension of the input samples and K is the output dimension. For classification tasks, K corresponds to the number of different classes and the neural network predicts a *probability distribution* over all possible output classes $k \in \{1, 2, 3, \ldots, K\}$. Using the output, the final class prediction can be deduced by simply picking the class with the highest probability.

The book also states that, a neural network is a composition of many functions applied to the input $\vec{x}$. For example for a network with 3 layers, the layers can be described by functions f_1, f_2, f_3 so therefore the output of the function is $f_1(f_2(f_3(\vec{x})))$. The output of layer 3 then forms the final prediction. Usually, neural networks contain many layers.

In classification problems, according to the book, it is common to utilize a *soft-max* function on the last layer. This function takes a real-valued input vector and transforms it into a discrete probability distribution of same dimension. Formally, given a vector $\vec{t} \in \mathbb{R}^D$ the soft-max function is defined as

$$S_i(\vec{t}) = \frac{e^{\vec{t}_i}}{\sum_{j=1}^{D} e^{\vec{t}_j}} \quad \forall i = 1 \ldots D.$$

The output $S_i(\vec{x})$ is an estimate of the probability $p(y_i|\vec{x})$, i.e., the probability that the input $\vec{x}$ belongs to class i.

For NNs to make sensible predictions, they must first be **trained**. At each training step, we feed a set of input vectors to the network. We can then tune the network using the prediction produced by the network and the correct output. This deviation of the prediction from the correct output guides how the network should be modified. After modifying the network a certain number of times, it becomes better at predicting the class of a given input.

More specifically, this deviation is measured with the help of a loss function. The most common loss function for classification tasks is the *cross-entropy* loss function. According to the book by Goodfellow et al., given an output $f(\vec{x})$ and its respective ground-truth label $\vec{y} = [0, 0, \ldots, 0, 1, 0, \ldots, 0]$ (encoded as a one-hot vector where $\vec{y}_i = 1$ if the class i is the correct one), the cross-entropy loss function is defined as:

$$l(\vec{x}, \vec{y}) = -\sum_{i}^{K} \vec{y}_i \log f(\vec{x})_i$$

The parameters of our NN can be estimated by minimizing

$$L = \sum_{(\vec{x},\vec{y}) \in D^{\text{training}}} l(f(\vec{x}), \vec{y})$$

which is usually achieved by utilizing stochastic gradient methods, such as stochastic gradient descent (SGD).

3 Problem statement

Generally, the problem of detecting out-of-distribution input samples is self explanatory. First, let us suppose we have trained a neural network classifier on a sample dataset with inputs $\mathcal{X}_{\text{train}} = \{x_{\text{train}}^{(i)}\}_{I=1}^{N} \subset \mathbb{R}^N$ and outputs $\mathcal{Y}_{\text{train}}$. Furthermore, let p_{train} denote the density function for the distribution of $\mathcal{X}_{\text{train}}$. After training, we deploy it into the real world, where the sample distribution is p_{test}. Crucially $p_{\text{train}} \neq p_{\text{test}}$; in other words, the real-world distribution of inputs can be quite different from the one the NN has been trained for, as pointed out by Amodei et al. (2016).

As pointed out by Amodei et al., one of main problems is the overconfidence that machine learning systems have in their analysis even if the input sample is drawn from p_{test}. This causes problems when deploying systems in critical application areas, such as medicine or finance.

OOD detection methods aim to detect cases where the input sample is drawn from p_{test} and causes a neural network system to give a confident but wrong analysis. The first instinct could be to model the density p_{train} and then evaluate for new inputs drawn from p_{test}. The problem with this is that interesting input spaces (e.g., images) are often high-dimensional and have an intricate topology, therefore modelling p_{train} becomes a very challenging task, as Kirichenko et al. (2020) has pointed out. Because of this, the proposed methods leverage some latent space (e.g. the output of some layer of the neural network) to detect out-of-distribution samples.

4 Methods

I now describe five of the most prominent methods for detecting out-of-distribution input samples in detail. The methods differ in many ways, e.g., ODIN requires access to OOD samples during training while Generalized ODIN does not. Therefore, in each subsection I also point out the advantages, limitations and requirements of each method.

4.1 Gram's matrices

Overview. Shama Sastry and Oore (2019) proposed a method for detecting out-of-distribution input samples which works by analyzing statistics of the neural network activations observed during training. The high level idea behind the method is to store the way last layers of the network are activated when considering an input sample with a certain classification class. This is achieved by applying an already trained deep convolutional neural network (DCNN) to a set of in-distribution training examples. More specifically, the idea is to store information about the activation values of each layer during training time. This information can then be used to detect out-of-distribution by comparing the stored information to the one computed for the input sample.

Method. Let $F_l(\vec{x})$ denote the feature map (activation values) of layer l given $\vec{x}$. $F_l(\vec{x})$ is defined to be an $n_l \times p_l$ matrix where p_l is the number of pixels in the output of that specific layer and n_l is the number of channels in layer l. Furthermore, let $F_l^p(\vec{x})$ denote the p-th element-wise power of $F_l(\vec{x})$. The authors define the Gram matrix G_l^p as

$$G_l^p(\vec{x}) = (F_l^p(\vec{x}) F_l^p(\vec{x})^T)^{\frac{1}{p}},$$

where the p-th root is also computed element-wise.

The authors highlight that computing $G_l^{p>1}$ instead of just G_l increases the effectiveness of this method. However, their experimental results show that powers greater than $p = 10$ do not help to achieve better test scores.

During training we store the minimum and maximum values of the elements in $\overline{G_l^p}$ for all layers $l \in \{1, 2, 3, ..., L\}$ and for all powers $p \in P = \{1, 2, 3, ..., 10\}$. The overline denotes that the matrix G_l^p is flattened into a vector. The minimum and maximum values of $\overline{G_l^p}$

are stored in the arrays Maxs$[c][l][p][i]$ and Mins$[c][l][p][i]$. The indices c, l, p, i refer to the class, layer, power and pixel/channel index, respectively.

The matrices G_l^p are utilized in calculating the deviation in each layer for a given input when compared to the training dataset. For this purpose, the paper defines deviation δ:

$$\delta(\max, \min, \text{value}) = \begin{cases} 0 & \min \leq \text{value} \leq \max \\ \frac{\min - \text{value}}{|\min|} & \text{value} < \min \\ \frac{\text{value} - \max}{|\max|} & \text{value} > \max \end{cases}$$

The authors use δ to determine how much the output value generated by an input deviates from the minimum and maximum values obtained from training.

Using δ function and the matrices G_l^p it is possible to determine the deviation of an input image D with respect to the layer l.

$$\delta_l(D) = \sum_{p=1}^{P} \sum_{i=1}^{V_l} \delta(\text{Maxs}[c][l][p][i], \text{Mins}[c][l][p][i], \overline{G_l^p(D)[i]})$$

Where V_l is the amount of correlation values in each layer.

The paper introduces the total deviation of an input image to be

$$\Delta(D) = \sum_{l=1}^{L} \frac{\delta_l(D)}{\mathbf{E}[\delta_l]}$$

Where $\mathbf{E}[\delta_l]$ serves as a normalization constant for the l-th layer. This expected deviation can be computed using validation data, according to the authors.

Finally, the authors utilize the total deviation to determine if a sample is out-of-distribution or not. They define a function isOOD the following way:

$$\text{isOOD}(D) = \begin{cases} \text{True}, & \Delta(D) > \tau \\ \text{False}, & \Delta(D) \leq \tau \end{cases},$$

where the threshold τ is set using validation data such that 95% of all testing samples give the output *False* when given as input to the isODD function.

Advantages. This method has the advantage of not requiring out-of-distribution examples. In other words, this method uses just in-distribution input samples for determining whether or not a given input is out-of-distribution.

Another advantage of this method is the fact that it does not require modifying the pre-trained neural network. Rather, this method works ”on top” of the trained neural network and only requires storing the 4-dimensional Mins and Maxs arrays.

Limitations. The method has no significant limitations. Nonetheless, it is suitable only in classification contexts, which is common limitation of most OOD methods.

Requirements. One possible drawback of this method is the requirement of keeping track of Maxs[c][l][p][i] and Mins[c][l][p][i] 4-dimensional arrays. This could become a problem when utilizing very wide DCNNs.

4.2 ODIN

Overview. The paper by Liang et al. (2018) presents ODIN, a method for detecting out-of-distribution input in images classification tasks. The proposed method is based on two components: **temperature scaling** and **input perturbation**.

The main goal of the method is to use a modified softmax function to create a difference in the softmax scores between in-distribution and out-of-distribution samples. Input perturbation is also applied to broaden this gap.

Method. Temperature scaling is a small modification in the soft-max classification function.

$$\tilde{S}_i(\vec{x};T) = \frac{\exp\left(f_i(\vec{x})/T\right)}{\sum_{j=1}^{N}\exp\left(f_j(\vec{x})/T\right)}$$

Where $f_i(\vec{x})$ is the output of the pre-trained neural network f for class $i \in \{1, 2, 3, ..., K\}$. $T \in \mathbb{R}^+$ is the temperature scaling parameter. The paper advises that during training T should be set to 1.

The paper notes temperature scaling to be an effective method for separating softmax scores between out-of-distribution and in-distribution samples.

Another component of the proposed method is input preprocessing, specifically the paper proposes perturbing the input before feeding it into the trained neural network.

$$\tilde{x} = \vec{x} - \epsilon \text{sign}(-\nabla_x \log \tilde{S}_{\hat{y}}(\vec{x};T))$$

Where $\tilde{S}_{\hat{y}}(\vec{x};T)$ is the maximum softmax value, $\tilde{S}_{\hat{y}}(\vec{x};T) = \max_i \tilde{S}_i(\vec{x};T)$. Intuitively, this input preprocessing sharpens the softmax score of the input $\vec{x}$. As the paper notes however, the softmax gain for in-distribution input samples is greater than for out-of-distribution samples. Therefore this allows the two to be separated more easily.

The detector can be defined now. The detector works by comparing $\max_i \tilde{S}_i(\tilde{x};T)$ to a threshold value δ. More formally

$$g(\vec{x}; \epsilon, \delta, T) = \begin{cases} 1 & \text{if} \max_i \tilde{S}_i(\tilde{x}; T) \leq \delta \\ 0 & \text{if} \max_i \tilde{S}_i(\tilde{x}; T) > \delta \end{cases}$$

Note that $\tilde{x}$ is the preprocessed input as defined above.

The authors propose searching the correct hyperparameters (δ, ϵ and T) by utilizing validation data such that the true positive rate (TPR) is 95% after training. TPR of 95% means that the amount of in-distribution samples categorized as in-distribution is 95%.

The paper also tries to give a more in depth analysis of temperature scaling. They begin by approximating $\tilde{S}_{\hat{y}}(\vec{x}; T)$ using the Taylor expansion.

$$\tilde{S}_{\hat{y}}(\vec{x}; T) \approx \frac{1}{N - \frac{1}{T}\sum_{i=1}^{N}[f_{\hat{y}}(\vec{x}) - f_i(\vec{x})] + \frac{1}{T^2}\sum_{i=1}^{N}[f_{\hat{y}}(\vec{x}) - f_i(\vec{x})]^2}$$

Next, the paper defines U_1 and U_2.

$$U_1(\vec{x}) = \frac{1}{N-1}\sum_{i \neq \hat{y}}[f_{\hat{y}}(\vec{x}) - f_i(\vec{x})]$$
$$U_2(\vec{x}) = \frac{1}{N-1}\sum_{i \neq \hat{y}}[f_{\hat{y}}(\vec{x}) - f_i(\vec{x})]^2$$

U_1 measures, by definition, the deviation of the other outputs when compared to the highest one. U_2 measures how the remaining outputs deviate from each other.

Then, the temperature-scaled soft-max classifier S can therefore be written as

$$\tilde{S} \propto \frac{U_1 - U_2/2T}{T}$$

Furthermore the paper notes that when U_1 is similar for two images, the out-of-distribution image tends to have a lower value for U_2 than the in-distribution image.

The paper also notes that for sufficiently large T the soft-max score is dominated by U_1 therefore after a certain value, increasing T is no longer effective.

For input perturbation the paper gives the following analysis. They begin by expressing the log soft-max output of a perturbed input $\tilde{x}$ using Taylor expansion.

$$\log \tilde{S}_{\hat{y}}(\tilde{x}; T) = \log \tilde{S}_{\hat{y}}(\vec{x}; T) + \epsilon||\nabla_x \log \tilde{S}_{\hat{y}}(\vec{x}; T)||_1 + o(\epsilon),$$

where $\vec{x}$ is the original input before preprocessing. Intuitively, input perturbation increases the softmax output of an input.

Moreover the paper notes that, for in-distribution images, $||\nabla_x \log \tilde{S}_{\hat{y}}(\vec{x}; T)||_1$ tends to be greater than for out-of-distribution images. Therefore, if two inputs have similar softmax scores, after preprocessing they have different softmax scores **if one of them is out-of-distribution**.

$$\tilde{S}(\vec{x}_1) \approx \tilde{S}(\vec{x}_2) \implies \tilde{S}(\tilde{x}_1) \not\approx \tilde{S}(\tilde{x}_2)$$

Overall this method is based on a simple idea of separating the softmax scores for in-distribution and out-of-distribution input samples. It does that by first preprocessing the input and then feeding it to a temperature scaled softmax classifier.

Advantages. One of the main advantages of this method is the fact that it does not require to modify an already trained neural network classifier. Also, the relative simplicity of this method is a very clear advantage. For example, when comparing this method to the one proposed by Lee et al. (2018), one can clearly see that this is much simpler approach.

Limitations ODIN, like all other methods discussed, only works on classification tasks. This is because the method works by modifying the softmax function, which is mainly suitable for classification tasks.

Requirements. This method has one substantial requirement. It requires access to out-of-distribution data for adjusting the hyperparameters T, ϵ and δ.

4.3 Generalized ODIN

Overview. The paper by Hsu et al. (2020) proposes a method based on ODIN which does not require access to out-of-distribution data. The main intuition of Generalized ODIN is to take a probabilistic viewpoint and to estimate how likely it is that a certain input sample is out-of-distribution.

The proposed approach is similar to ODIN. First, the input is preprocessed and then fed into a function which is used to classify valid samples and detect OOD samples. The difference is that the function used for classification and OOD detection is significantly more complicated than a modified softmax. This extra layer of complexity allows the method to function even without access to OOD detection.

Method. Formally, the proposed method is based on the probability equation:

$$p(y|d_{\text{in}}, \vec{x}) = \frac{p(y, d_{\text{in}}|\vec{x})}{p(d_{\text{in}}|\vec{x})},$$

where $p(y, d_{\text{in}}|\vec{x})$ is the joint probability of the input sample $\vec{x}$ being of class y and being in in-distribution. $p(y|d_{\text{in}}, \vec{x})$ is the probability that the class of the sample is y given the sample is in-distribution and the sample itself. $p(d_{\text{in}}|\vec{x})$ is the probability that the sample $\vec{x}$ is in-distribution. As the paper notes, the equation is given by the basic rule of conditional probability.

As the paper discusses, without having access to out-of-distribution data it is impossible to learn the joint probability $p(y, d_{in}|\vec{x})$ because we have no supervision on the domain d.

Therefore, the paper proposes using a logit function of the form f_i for class i:

$$f_i(\vec{x}) = \frac{h_i(\vec{x})}{g(\vec{x})}.$$

The output of f_i is then fed into a normalizing exponential function (i.e. softmax) to give the final prediction, which is subject to cross-entropy loss.

The paper discusses how using this approach, we could potentially decompose the confidence into the probabilities $p(y, d_{\text{in}}|\vec{x})$ and $p(d_{\text{in}}|\vec{x})$. In other words h_i could approximate $p(y, d_{\text{in}}|\vec{x})$ and g could approximate $p(d_{\text{in}}|\vec{x})$. As the paper points out, in low-density areas h_i has difficulties rising to a large value, therefore to minimize cross-entropy loss even more, g should be encouraged to be small. Similarly, in high-density areas h_i can reach high values more easily so g is less encouraged to be small. More specifically the article proposes using a linear layer for g.

$$g(\vec{x}) = \sigma(BN(\mathbf{w}_g f^p(\vec{x}) + b_g)),$$

where f^p is the output of the **penultimate** layer of the trained neural network, BN denotes the use of batch-normalization and σ is the nonlinear sigmoid function. We also have a bias for this layer b_g. $\vec{x}$ again denotes the input vector.

For h_i the paper proposes three forms:

$$\begin{aligned} h_i^I(\vec{x}) &= \mathbf{w}_i^T f^p(\vec{x}) + b_i \\ h_i^E(\vec{x}) &= -||f^p(\vec{x}) - \mathbf{w}_i||_2^2 \\ h_i^C(\vec{x}) &= \frac{\mathbf{w}_i^T f^p(\vec{x})}{||\mathbf{w}_i||_2 ||f^p(\vec{x})||_2} \end{aligned}$$

The *inner-product similarity* measure is denoted by $h_i^I(\vec{x})$, $h_i^E(\vec{x})$ is the *negative Euclidean distance* measure and $h_i^C(\vec{x})$ is the *cosine similarity* measure. $\mathbf{w}_i$ and b_i are learned parameters for class i.

The paper proposes that during training $f_i(\vec{x})$ is computed, followed by its softmax and then the cross-entropy loss function is applied on top of that. During testing, the class could be computed using either $\arg\max_i f_i$ or $\arg\max_i h_i$. The out-of-distribution detection can be computed using either $\max_i h_i$ or g during testing.

This work uses the same input perturbation technique employed by Liang et al. (2018). Specifically, the perturbed input $\tilde{x}$ is computed using the same form:

$$\tilde{x} = \vec{x} - \epsilon \text{sign}(-\nabla_{\vec{x}} S(\vec{x})),$$

where $S(\vec{x})$ denotes the confidence score given by $g(\vec{x})$.

Contrary to ODIN, this work does not use out-of-distribution samples for tuning the hyperparameter ϵ. Hsu et al. use only the in-distribution input samples and ϵ is chosen by iterating and choosing one which maximizes $S(\tilde{x})$:

$$\epsilon^* = \arg\max_{\epsilon} \sum_{\vec{x} \in D_{\text{val}}^{\text{in}}} S(\tilde{x}),$$

where $D_{\text{val}}^{\text{in}}$ denotes the validation dataset of in-distribution input samples.

Additionally, this work proposes using only 6 possible values for ϵ so the search is much faster than the proposed twenty-one possible values proposed by Liang et al.

Advantages. This method solves the one major limitation that the method by Liang et al. has, namely the requirement of having a out-of-distribution input sample dataset.

Limitations. One limitation of this method is the added complexity when compared to the traditional ODIN approach. Also, this method is only useful in classification contexts.

Requirements. This method requires more extensive experimentation to, for example, determine the best option for the similarity measure h_i, which depends on the specific problem.

4.4 Gaussian discriminant analysis & Mahalanobis

Overview. The paper written by Lee et al. (2018) proposes another solution for detecting out-of-distribution input samples. Broadly, the idea here is to model the last activations of the network using a class-conditional Gaussian distribution.

Furthermore, the goal is to measure differences between the distributions given by in-distribution samples when compared to the ones given by out-of-distribution samples. In other words when given an input, we propagate it until the last layer and measure the density of the activations under every class-conditional Gaussian distribution. If the densities for every class-conditional Gaussian are low, the input is very likely out-of-distribution.

Method. The measurement between distributions is done using a *generative* or a distance-based measure. Specifically, the paper proposes using the *Mahalanobis* distance as being this measure.

The paper first assumes that there exists a trained classifier for which the class-prediction distribution for class c is given by:

$$P(y = c|\vec{x}) = \frac{\exp\left(\mathbf{w}_c^T f(\vec{x}) + b_c\right)}{\sum_{c'} \exp\left(\mathbf{w}_{c'}^T f(\vec{x}) + b_{c'}\right)}$$

Where $\vec{x}$ denotes the input and $\mathbf{w}_c$ and b_c are the trained weights and bias for class c. $f(\cdot)$ denotes the output of the **penultimate** layer of the neural network.

More formally the paper proposes that the output of the penultimate layer of neural network $(f(\vec{x}))$ is distributed similarly to a class-conditional multivariate Gaussian distribution.

$$P(f(\vec{x})|y = c) = \mathcal{N}(f(\vec{x})|\hat{\mu}_c, \hat{\Sigma})$$

Where μ_c is the class mean of a multivariate Gaussian distribution and Σ is the tied covariance matrix. Practically these can be estimated using the training dataset.

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} f(\vec{x}_i)$$

$$\hat{\Sigma} = \frac{1}{N} \sum_c \sum_{i:y_i=c} (f(\vec{x}_i) - \hat{\mu}_c)(f(\vec{x}_i) - \hat{\mu}_c)^T$$

Where again $f(\cdot)$ denotes the output the penultimate layer of a trained neural network. c denotes the class index $c \in \{1, 2, \ldots, C\}$. N_c is the size of the training set for class c and N is the total count of training examples.

Next, the paper defines the confidence measure using Mahalanobis distance and $\hat{\mu}$ and $\hat{\Sigma}$. Intuitively, this measure is the "distance" between the closest class-conditional distribution and the output of the penultimate layer for input $\vec{x}$.

$$M(\vec{x}) = \max_c -(f(\vec{x}) - \hat{\mu}_c)^T \hat{\Sigma} (f(\vec{x}) - \hat{\mu}_c)$$

This confidence measure corresponds to the log of the probability densities of the test sample. Essentially, the Mahalanobis distance is proportional to minus the log kernel of a mutlivariate Gaussian density. Therefore high Mahalanobis distance can be associated with regions of low density.

According to the paper, by examining the output of the penultimate layer of the neural network we can get better detection results than just by analyzing the resulting class prediction distribution. This is because the class prediction distribution can give very high confidence measures even when the abnormal sample lies very far from a decision boundary of the classifier.

The authors Lee et al. also propose using a similar input preprocessing method as used in the method proposed by Liang et al..

$$\tilde{x} = \vec{x} + \epsilon \text{sign}(\nabla_{\vec{x}} M(\vec{x}))$$

Where again ϵ is a small hyperparameter. The goal of this input preprocessing is to separate the out-of-distribution and in-distribution samples from each other.

The paper also proposes a way of improving on this base form by considering **all** the layers of the neural network. They propose defining a weighted sum of all layer's confidence scores and using that as the final confidence score.

$$M(\vec{x}) = \sum_{l} \alpha_l M_l(\vec{x})$$

The weights α_l are found by training a logistic regression detector using validation data. The paper notes that using all the layers one can avoid the unfortunate case where confidence scores from some layers are not effective.

The authors also mention how using the Mahalanobis distance can be used to perform *class-incremental* learning. Essentially, the out-of-distribution detector can learn to recognize new classes. The paper notes that just by giving the neural network a new set of input samples related to a new class, it can accommodate this new class just by computing the class-conditional mean and the tied covariance.

Advantages. One advantage this method has is the fact that it does not require modifying the already trained neural network classifier. Also as discussed above, this method can be utilized in class-incremental learning tasks. Furthermore, this method does **not** require out-of-distribution input samples for training.

Limitations. This method is only applicable to classification tasks and, therefore, is not suitable for other ML tasks.

Requirements. This method has a small drawback. Namely, this method requires computing the mean $\hat{\mu}_c$ and correlation $\hat{\Sigma}$ using the in-distribution dataset. This could potentially be computationally expensive, if the size of training dataset is large.

4.5 Energy-based approach

Overview. The final method that this thesis covers is based on the concept of *energy-based models*. The paper written by Liu et al. (2020) proposes using an energy function instead of the label-overfitted output space of the neural network. The Softmax function can be written as an energy-based model and, therefore, its respective energy function can be used as a stand-in for traditional Softmax value. However, energy has significant advantages compared to softmax when detecting OOD samples.

The high-level idea of this approach is that using a certain definition of energy function, the energy gap between out-of-distribution and in-distribution samples is wide. Therefore, by placing a simple threshold value for this energy, a method for detecting out-of-distribution input samples is devised. Figure 1 illustrates the process.

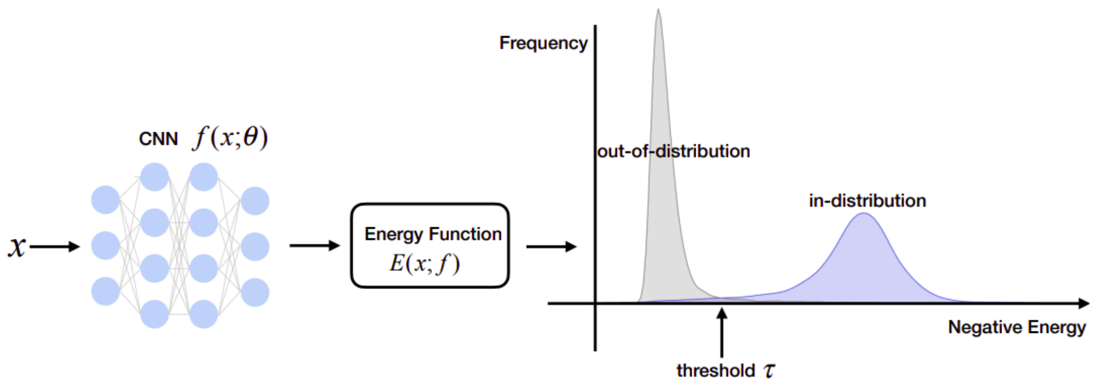

Figure 1: CNN refers to a trained convolutional neural network with input $\vec{x}$ and parameters θ. $E(\vec{x}; f)$ is the energy for input $\vec{x}$. τ is the threshold for classifying $\vec{x}$ as out-of-distribution.

Method. More formally, according to the paper, energy-based models work by defining energy to be some scalar defined by a function $E : \mathbf{R}^D \to \mathbf{R}$ for the input vector $\vec{x}$. For a collection of energy points, a probability density $p(\vec{x})$ can then be defined using the *Gibbs distribution*, formally:

$$p(y|\vec{x}) = \frac{e^{-E(\vec{x},y)/T}}{\int_{y'} e^{-E(\vec{x},y')/T}} = \frac{e^{-E(\vec{x},y)/T}}{e^{-E(\vec{x})/T}}$$

Where T is a temperature scalar. The paper also notes that the log of the denominator is the *Helmholtz free energy* $E(\vec{x})$ of a point $\vec{x}$.

$$E(\vec{x}) = -T\log(\int_{y'} e^{-E(\vec{x},y')/T})$$

The main observation that the authors have made is that this notion of energy is connected to a softmax classification function with a temperature parameter T.

$$p(y|\vec{x}) = \frac{e^{f_i(\vec{x})/T}}{\sum_j^K e^{f_j(\vec{x})/T}}$$

Where f_i is the output of the neural network before the softmax classifier for class-label with index i. f_i is also known as the *logit* of index i. K is the amount of classes.

By combining these equations we can define the energy function $E(\vec{x}, y) = -f_y(\vec{x})$ for input $\vec{x}$. Therefore, the Helmholtz free energy of input $\vec{x}$ and neural network f is defined to be:

$$E(\vec{x}; f) = -T\log(\sum_i^K e^{f_i(\vec{x})/T})$$

The authors then propose a simple threshold based approach for detecting out-of-distribution samples.

$$G(\vec{x}; f; \tau) = \begin{cases} 1, & \text{if} - E(\vec{x}; f) > \tau \\ 0, & \text{if} - E(\vec{x}; f) \leq \tau \end{cases}$$

$G(\vec{x}; f; \tau)$ is 0 for out-of-distribution samples and 1 for valid in-distribution ones. The parameter τ is chosen using in-distribution training data so that a high proportion of the valid samples are classified as in-distribution.

The authors justify using the energy $E(\vec{x}; f)$ for out-of-distribution detection by examining the density function of data $p(\vec{x})$ using an energy-based model.

$$p(\vec{x}) = \frac{e^{-E(\vec{x};f)/T}}{\int_{\vec{x}} e^{-E(\vec{x};f)/T}}$$

The problem with this is that the integral $\int_{\vec{x}} e^{-E(\vec{x};f)/T}$ is often not tractable and often cannot even be estimated. To solve this problem, the authors propose taking a logarithm:

$$\log p(\vec{x}) = -E(\vec{x}; f)/T - \log Z$$

Where $\log Z$ is a **constant** for all $\vec{x}$. Therefore, the quantity $E(\vec{x}; f)$ is proportional to the log likelyhood function $\log p(\vec{x})$.

The authors further assert that the energy function $E(\vec{x}; f)$ is a better estimator of $p(\vec{x})$ than the softmax confidence score. The claim is justified with this mathematical deduction.

$$\begin{aligned}
\max_y p(y|\vec{x}) &= \max_y \frac{e^{f_y(\vec{x})}}{\sum_i^K e^{f_i(\vec{x})}} \\
&= \frac{1}{\sum_i^K e^{f_i(\vec{x})} - e^{f_{\max}(\vec{x})}} \\
\implies \log \max_y p(y|\vec{x}) &\overset{T=1}{=} E(\vec{x}; f(\vec{x}) - f^{\max}(\vec{x})) \\
&= E(\vec{x}; f) + f^{\max}(\vec{x}) \\
&= -\log p(\vec{x}) + f^{\max}(\vec{x}) - \log Z \qquad (1)
\end{aligned}$$

The term $\frac{e^{f_y(\vec{x})}}{\sum_i^K e^{f_i(\vec{x})}}$ is the traditional softmax confidence score that is used for out-of-distribution detection. The authors note that equation (1) implies that the traditional confidence score is not directly proportional to the confidence probability and is therefore ill-suited for out-of-distribution detection.

While the energy difference can be used to differentiate out-of-distribution input samples from in-distribution samples, for even better performance a fine-tuning should be applied. The paper discusses a method for carrying-out this tuning of the network to produce an even wider energy gap between out-of-distribution and in-distribution input samples. This is done simply by adding a term L_{energy} to the loss function.

$$L_{\text{energy}} = \mathbb{E}_{(\vec{x}_{\text{in}}, y)}(\max(E(\vec{x}_{\text{in}}) - m_{\text{in}}, 0))^2 + \mathbb{E}_{(\vec{x}_{\text{out}})}(\max(E(m_{\text{out}} - \vec{x}_{\text{out}}), 0))^2$$

Where m_{in} and m_{out} are hyperparameters searched using validation data. Also, $\vec{x}_{\text{out}}$ is sampled from an auxiliary dataset containing OOD data. Intuitively, this loss penalizes input samples whose energy is in the range $[m_{\text{in}}, m_{\text{out}}]$. This creates a wider gap between the energy distributions.

L_{energy} is then included in the standard cross-entropy loss of the network. The training objective is therefore:

$$\min_\theta \mathbb{E}_{(\vec{x}_{\text{in}}, y)}(-\log F_y(\vec{x})) + \lambda L_{\text{energy}}$$

Where $F(\vec{x})$ is the output of the softmax classifier for input $\vec{x}$. λ is a chosen hyperparameter, it is not chosen using validation data. θ denotes **all** the parameters of the network, i.e. the weights and biases.

Advantages. One of the advantages of this method is that it does not require out-of-distribution training data. However, the fine-tuning of this method does require it. In other words, this method can be used regardless of whether out-of-distribution training data is available, however the approach performs better with training data.

I would also argue that implementing this method is not very complex and it does not require large amounts of memory, since the main variables are the threshold τ and hyperparameter T. Also it is not very computationally intensive to compute the energy $E(\vec{x})$ for the given input.

Limitations. It it worth mentioning that this approach has not been tested on any other datasets than CIFAR-100 and CIFAR-10. This means that further testing is possibly required for making sure this method can perform in other environments.

This work is also limited to classification tasks and does not discuss using the on other machine learning problems.

Requirements. This method has the common one of needing to set hyperparameter τ using in-distribution training data. Also, the optional fine-tuning requires setting of hyperparameters m_{in} and m_{out} using OOD validation data.

5 Conclusion

We have explored state-of-the-art methods to detect out-of-distribution samples and highlighted their specific advantages, limitations and requirements. By examining these different approaches we have noticed that all of them are restricted to classification tasks. Therefore, there is still a demand for out-of-distribution methods for other ML tasks, such as regression. Additionally, another possible research area could be to develop methods which could be used in unsupervised tasks, such as clustering.

Moreover, the methods covered here were focused on out-of-distribution detection for image domains. For example the method based on Gram's matrices depends on the existence of channels in the network, which is a feature of convolutional neural networks. Naturally, these methods were tested only on image datasets. Therefore, another research direction could be to evaluate whether existing methods can be used with other input data structures, e.g., graph data or time series.

In summary, the OOD literature has already seen important advances, both in terms of understanding the OOD phenomena and of developing effective strategies to handle it. However, I would argue that there is still room for improvement, especially with trying to solve OOD in other task contexts. Solving this problem would permit neural networks to be utilized more in critical application areas and it would decrease the risk companies have from utilizing neural networks. In turn, this would allow for more innovative products and services to be created.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman and Dan Mané. Concrete problems in AI safety. *ArXiv:1606.06565*, 2016.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

Y. Hsu, Y. Shen, H. Jin and Z. Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Polina Kirichenko, Pavel Izmailov and Andrew Gordon Wilson. Why normalizing flows fail to detect out-of-distribution data, 2020.

Yann Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L.D. Jackel. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems (NeurIPS)*, 1989.

Kimin Lee, Kibok Lee, Honglak Lee and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Shiyu Liang, Yixuan Li and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. 2018. International Conference on Learning Representations, (ICLR).

W. Liu, X. Wang, J. D. Owens and Y. Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Matthew Riemer, Aditya Vempaty, Flavio Calmon, Fenno Heath, Richard Hull and Elham Khabiri. Correcting forecasts with multifactor neural attention. *International Conference on Machine Learning (ICML)*, 2016.

Chandramouli Shama Sastry and Sageev Oore. Detecting Out-of-Distribution Examples with In-distribution Examples and Gram Matrices. *ArXiv:1912.12510*, 2019.