A General Framework For Detecting Anomalous Inputs to DNN Classifiers

Jayaram Raghuram [*1] **Varun Chandrasekaran** [*1] **Somesh Jha** [1 2] **Suman Banerjee** [1]

Abstract

Detecting anomalous inputs, such as adversarial and out-of-distribution (OOD) inputs, is critical for classifiers (including deep neural networks or DNNs) deployed in real-world applications. While prior works have proposed various methods to detect such anomalous samples using information from the internal layer representations of a DNN, there is a lack of consensus on a principled approach for the different components of such a detection method. As a result, often heuristic and one-off methods are applied for different aspects of this problem. We propose an unsupervised anomaly detection framework based on the internal DNN layer representations in the form of a meta-algorithm with configurable components. We proceed to propose specific instantiations for each component of the meta-algorithm based on ideas grounded in statistical testing and anomaly detection. We evaluate the proposed methods on well-known image classification datasets with strong adversarial attacks and OOD inputs, including an *adaptive attack* that uses the internal layer representations of the DNN (often not considered in prior work). Comparisons with five recently-proposed competing detection methods demonstrates the effectiveness of our method in detecting adversarial and OOD inputs.

1. Introduction

Deep neural networks (DNNs) have achieved impressive performance on a variety of challenging machine learning (ML) problems such as image classification, object detection, speech recognition, and natural language processing (He et al., 2015; Krizhevsky et al., 2017). However, it is well-known that DNN classifiers can be highly inaccurate (sometimes with high confidence) on test inputs from outside the training distribution (Nguyen et al., 2015; Szegedy et al., 2014; Hendrycks & Gimpel, 2017; Hein et al., 2019). Such anomalous inputs can arise in real-world settings either unintentionally due to external factors, or due to malicious adversaries that intend to cause prediction errors in the DNN and disrupt the system (Barreno et al., 2006; Biggio & Roli, 2018). Therefore, it is critical to have a defense mechanism that can detect such anomalous inputs, and take suitable corrective action (*e.g.*, abstain from predicting (Tax & Duin, 2008) or provide a more reliable class prediction).

In this work, we propose `JTLA` (Joint statistical Testing across DNN Layers for Anomalies), a general unsupervised framework for detecting anomalous inputs (including adversarial and OOD) to a DNN classifier using its layer representations. `JTLA` utilizes the rich information at the intermediate layer representations of a DNN to obtain a better understanding of the patterns produced by anomalous inputs for detection. Our method is *unsupervised*, *i.e.*, it does not utilize any specific class(es) of known anomalous samples for learning or tuning its parameters. While a number of prior works have addressed the problem of adversarial and OOD detection (Feinman et al., 2017; Xu et al., 2018; Li & Li, 2017; Lee et al., 2018; Ma et al., 2018; Roth et al., 2019), including ones that utilize intermediate layer representations of a DNN (Li & Li, 2017; Meng & Chen, 2017; Xu et al., 2018; Lee et al., 2018; Zheng & Hong, 2018; Ma et al., 2018; Papernot & McDaniel, 2018; Miller et al., 2019; Yang et al., 2020; Sastry & Oore, 2020), some key limitations persist, that we propose to address in this work.

Limitations of prior work. First, a number of existing detection methods (Feinman et al., 2017; Lee et al., 2018; Ma et al., 2018; Yang et al., 2020) being supervised, have to be presented with a broad sampling of known anomalous samples for training (*e.g.*, different adversarial attacks of varying strength). Such methods typically also need to configure hyper-parameters based on the known anomalous samples from the training set (*e.g.*, using cross-validation). As a result, they often do not generalize well to *unknown* anomalies (*e.g.*, novel or adaptive attacks). It has been shown that a majority of the current detection methods fail to handle unseen and adaptive adversaries that are aware of the defense mechanism (Carlini & Wagner, 2017a; Tramèr et al., 2020). **Second**, detection methods that use only the input, output (pre-softmax), or a specific DNN layer (Roth et al.,

[*]Equal contribution [1]Computer Sciences, University of Wisconsin, Madison, USA. [2]XaiPient Inc., Princeton, NJ, USA. Correspondence to: Jayaram Raghuram <`jayaramr@cs.wisc.edu`>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

2019; Hendrycks & Gimpel, 2017; Feinman et al., 2017) do not jointly exploit the properties exhibited by anomalous inputs across the layers. **Third**, although methods that utilize information from multiple layers (listed earlier) propose specific test statistics or features calculated from the layer representations (*e.g.*, local intrinsic dimensionality (Ma et al., 2018)), there is a lack of a *general anomaly detection framework* where one can plug-in test statistics, aggregation, and scoring methods suitable for the detection task. **Fourth**, existing unsupervised detection methods that are based on density (generative) modeling of the DNN layer representations (Zheng & Hong, 2018; Miller et al., 2019; Feinman et al., 2017) are not well-suited to handle the (often very) high dimensional layer representations. **Finally**, we observe that existing detection methods often do not utilize the predicted class of the DNN to focus on class-conditional properties of the layer representations (Ma et al., 2018; Li & Li, 2017; Xu et al., 2018; Yang et al., 2020), which can lead to improved detection performance. While prior works such as (Roth et al., 2019; Miller et al., 2019; Zheng & Hong, 2018; Sastry & Oore, 2020) are exceptions to this, there is still need for a unified approach in this regard.

Our contributions can be summarized as follows:

- We propose a general unsupervised framework `JTLA` for detecting anomalous inputs to a DNN using its layer representations. We first present a meta-algorithm and describe its components in general terms (§ 3). We then propose specific methods for realizing the components in a principled way (§ 4). The proposed framework is modular, and a number of prior works for anomaly detection based on the layer representations can be cast into this meta-algorithm.
- The importance of designing an adaptive, defense-aware adversary has been underscored in the literature (Carlini & Wagner, 2017a; Tramèr et al., 2020). We propose and evaluate against an adversarial attack that focuses on defenses (such as ours) that use the k-nearest neighbors of the layer representations of the DNN (§ 5).
- We report extensive experimental evaluations comparing `JTLA` with five baseline methods on three image classification datasets trained with suitably-complex DNN architectures. For adversarial detection, we evaluate on three well-known whitebox attacks and our proposed defense-aware attack (§ 6 and Appendix E) [1].

2. Related Works

We provide a brief review of related works on adversarial and OOD detection, focusing on methods that use the internal layer representations of a DNN. A detailed discussion of closely-related prior works, and how they fit into the proposed anomaly detection framework is provided in Appendix A.4. Recent surveys on adversarial learning and anomaly detection for DNNs can be found in (Biggio & Roli, 2018; Miller et al., 2020; Bulusu et al., 2020).

Prior works on adversarial and OOD detection can be broadly categorized into unsupervised and supervised methods. Supervised methods such as (Lee et al., 2018), (Ma et al., 2018), and (Yang et al., 2020) use a training set of adversarial or OOD samples (*i.e.*, known anomalies) to train a binary classifier that discriminates natural inputs from anomalies. They extract specific informative test statistics from the layer representations as features for the classifier. On the other hand, unsupervised methods such as (Roth et al., 2019; Zheng & Hong, 2018; Miller et al., 2019; Sastry & Oore, 2020; Li & Li, 2017; Xu et al., 2018), rely on interesting statistical properties and generative modeling of natural inputs at specific (*e.g.*, logit) or multiple layer representations of the DNN for detection. Works such as the trust score (Jiang et al., 2018), deep kNN (Papernot & McDaniel, 2018), and by Jha et al. (2019) have explored the problem of developing a confidence metric that can independently validate the predictions of a classifier. Inputs with low confidence scores are likely to be misclassified and hence are detected as anomalies.

3. Anomaly Detection Meta-algorithm

We first introduce the notation and problem setup, followed by a description of the proposed meta-algorithm.

3.1. Notations and Setup

Consider the conventional classification problem where the goal is to accurately classify an input $\mathbf{x} \in \mathcal{X}$ into one of m classes $[m] := \{1, \cdots, m\}$. We focus on DNN classifiers that learn a function of the form $\mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), \cdots, F_m(\mathbf{x})]$, $\mathbf{F} : \mathcal{X} \mapsto \Delta_m$, where $\mathcal{X}$ is the space of inputs to the DNN and $\Delta_m = \{(p_1, \cdots, p_m) \in [0, 1]^m : \sum_i p_i = 1\}$ is the space of output class probabilities. The class prediction of the DNN based on its output class probabilities is defined as $\widehat{C}(\mathbf{x}) = \arg\max_{c \in [m]} F_c(\mathbf{x})$. The multi-layer architecture of a DNN allows the input-output mapping to be expressed as a composition of multiple functions, *i.e.*, $\mathbf{F}(\mathbf{x}) = (\mathbf{g}_L \circ \mathbf{g}_{L-1} \cdots \circ \mathbf{g}_1)(\mathbf{x})$, where L is the number of layers. The output from an intermediate layer $\ell \in \{1, \cdots, L\}$ of the DNN, $\mathbf{f}_\ell(\mathbf{x}) = (\mathbf{g}_\ell \circ \cdots \circ \mathbf{g}_1)(\mathbf{x}) \in \mathbb{R}^{d_\ell}$, is referred to as its layer representation [2]. We also use the shorthand notation $\mathbf{x}^{(\ell)} = \mathbf{f}_\ell(\mathbf{x})$, with $\mathbf{x}^{(0)} = \mathbf{f}_0(\mathbf{x})$ denoting the vectorized input. The set of layers and distinct layer pairs are denoted by $\mathcal{L} = \{0, \cdots, L\}$ and $\mathcal{L}^2 = \{(\ell_1, \ell_2) \in \mathcal{L} \times \mathcal{L} : \ell_2 > \ell_1\}$. Table 1 in the

[1]The code base associated with our work can be found at: https://github.com/jayaram-r/adversarial-detection.

[2]Layers with tensor-valued outputs (*e.g.*, convolution) are flattened into vectors. Boldface symbols are used for vectors and tensors.

Appendix provides a quick reference for the notations.

We assume access to the trained DNN classifier to defend, and a labeled data set $\mathcal{D} = \{(\mathbf{x}_n, c_n),\ n = 1, \cdots, N\}$ that is different from the one used to train the DNN and does not contain any anomalous samples. We define an augmented data set $\mathcal{D}_a = \{(\mathbf{x}_n^{(0)}, \cdots, \mathbf{x}_n^{(L)}, c_n, \hat{c}_n),\ n = 1, \cdots, N\}$ that is obtained by passing samples from $\mathcal{D}$ through the DNN and extracting their layer representations $\mathbf{x}_n^{(\ell)} = \mathbf{f}_\ell(\mathbf{x}_n),\ \ell \in \mathcal{L}$ and the class prediction $\hat{c}_n = \widehat{C}(\mathbf{x}_n)$. We also define subsets of $\mathcal{D}_a$ corresponding to each layer $\ell \in \mathcal{L}$, and each predicted class $\hat{c} \in [m]$ or true class $c \in [m]$ respectively as:

$$\widehat{\mathcal{D}}_a(\ell, \hat{c}) = \{(\mathbf{x}_n^{(\ell)}, c_n, \hat{c}_n),\ n = 1, \cdots, N \,:\, \hat{c}_n = \hat{c}\},$$
$$\mathcal{D}_a(\ell, c) = \{(\mathbf{x}_n^{(\ell)}, c_n, \hat{c}_n),\ n = 1, \cdots, N \,:\, c_n = c\}.$$

3.2. Components of the Meta-algorithm

Algorithm 1 Meta-algorithm for Anomaly Detection

```
1: Inputs: Trained DNN $\mathbf{F}(\cdot)$, Dataset $\mathcal{D}$, Test input $\mathbf{x}$,
   FPR $\alpha$ or detection threshold $\tau$.
2: Output: Detector decision – normal 0 or anomaly 1.
3: Preprocessing:
4: Calculate the detection threshold $\tau$ (if not specified).
5: Calculate the class prediction and layer representations of $\mathbf{x}$.
6: Create the data subsets corresponding to each layer, predicted
   class, and $m$ true classes.
7: I. Test statistics (TS):
8:    for each layer $\ell$:
9:       Calculate the TS at layer $\ell$ conditioned on the predicted
         class and the $m$ candidate true classes.
10:   Compile the $m + 1$ TS vectors from the layers.
11: II. Normalizing transformations:
12:   if multivariate normalization:
13:      Normalize each of the $m + 1$ TS vectors.
14:   else
15:      for each layer $\ell$:
16:         Normalize the $m + 1$ TS from layer $\ell$.
17:      for each distinct layer pair $(\ell_1, \ell_2)$: [optional]
18:         Normalize the $m + 1$ TS pairs from layers $\ell_1, \ell_2$.
19: III. Layerwise aggregation and scoring:
20:   if multivariate normalization:
21:      No need to aggregate the normalized TS.
22:   else
23:      Aggregate the normalized TS from the layers and layer
         pairs for the predicted class and each candidate true class.
24:   Calculate the final score from the $m + 1$ aggregated nor-
      malized TS.
25: IV. Detection decision:
26:   Return anomaly (1) if the final score exceeds threshold;
      Else return normal (0).
```

The proposed meta-algorithm for detecting anomalous inputs to a DNN classifier based on its layer representations is given in Algorithm 1. A more formal version of the same can be found in Algorithm 2 in the Appendix. Details of the individual components of the meta-algorithm are discussed next. For this discussion, consider a test sample $\mathbf{x}$ whose true class is unknown, predicted class is $\widehat{C}(\mathbf{x}) = \hat{c}$, and layer representations are $\mathbf{x}^{(\ell)},\ \ell \in \mathcal{L}$.

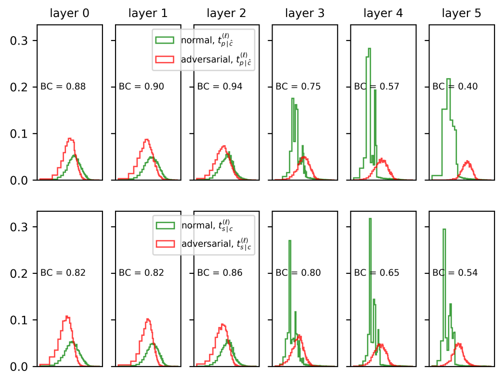


Figure 1: Distribution of test statistics corresponding to normal and adversarial samples (PGD, ℓ_∞ attack) from the layers of a DNN trained on the SVHN dataset. The top and bottom figures show the multinomial test statistic (§ 4.1) conditioned on the predicted and true class respectively. BC is the Bhattacharya coefficient, which is a measure of distribution overlap.

I. Test Statistics: In line with prior works that use the layer representations of a DNN for detection, we define test statistics at each layer that capture a statistical property of the layer representation useful for detection (*e.g.*, Mahalanobis distances (Lee et al., 2018)). The test statistics are defined to be conditioned on the predicted class and on each candidate true class (since the true class is unknown). The latter is particularly useful for adversarial samples, since they are known to have originated from one of the m classes. For a test input $\mathbf{x}$ predicted as class $\hat{c}$, the test statistic at any layer ℓ conditioned on the predicted class is defined as $T(\mathbf{x}^{(\ell)}, \hat{c}, \widehat{\mathcal{D}}_a(\ell, \hat{c}))$. It captures how anomalous the layer representation $\mathbf{x}^{(\ell)}$ is with respect to the distribution of natural inputs predicted into class $\hat{c}$ by the DNN. Similarly, a set of m test statistics at layer ℓ, conditioned on each candidate true class c, are defined as $T(\mathbf{x}^{(\ell)}, c, \mathcal{D}_a(\ell, c)),\ c \in [m]$. They capture how anomalous $\mathbf{x}^{(\ell)}$ is with respect to the distribution of natural inputs from a true class c. A mild requirement on the definition of the test statistic function is that larger values of $T(\cdot)$ correspond to larger deviations of the layer representation from the class-conditional distribution of natural inputs. When the input is clear from the context, we denote the test statistic random variables by $T^{(\ell)}_{p\,|\,\hat{c}} := T(\mathbf{x}^{(\ell)}, \hat{c}, \widehat{\mathcal{D}}_a(\ell, \hat{c}))$ and $T^{(\ell)}_{s\,|\,c} := T(\mathbf{x}^{(\ell)}, c, \mathcal{D}_a(\ell, c))$. Specific values of the test statistic are denoted by $t^{(\ell)}_{p\,|\,\hat{c}}$ and $t^{(\ell)}_{s\,|\,c}$. The vector of test statistics across the layers is defined as $\mathbf{t}_{p\,|\,\hat{c}} := [t^{(0)}_{p\,|\,\hat{c}}, \cdots, t^{(L)}_{p\,|\,\hat{c}}]$ given the predicted class $\hat{c}$, and as $\mathbf{t}_{s\,|\,c} := [t^{(0)}_{s\,|\,c}, \cdots, t^{(L)}_{s\,|\,c}],\ \forall c \in [m]$ given each candidate true class. In § 4.1, we propose a test statistic based on the multinomial likelihood ratio test (LRT) applied to class counts from the k-nearest neighbors (kNN) of a layer representation. However, the above definitions are general and apply to test statistics proposed in prior works such as Gram matrix-based deviations (Sastry & Oore, 2020).

II. Distribution-Independent Normalization: In the absence of any prior assumptions, the class-conditional and marginal distributions of the test statistics are unknown and expected to change across the DNN layers (*e.g.*, see Fig. 1). Therefore, in order to effectively combine the test statistics from the layers for anomaly scoring, it is important to apply a normalizing transformation that (ideally) makes the transformed test statistics distribution independent. Some prior works partially address this using heuristic approaches such as z-score normalization (Roth et al., 2019) and scaling by the expected value (Sastry & Oore, 2020) in order to account for the distribution and range differences of the test statistics across the layers. We *propose two approaches for applying normalizing transformations* – the first one focuses on test statistics from the individual layers and layer pairs, and the second one focuses on the vector of test statistic across the layers. Considering test statistic pairs and the vector of test statistics allows our method to capture the joint effect of anomalous inputs on the layer representations.

Consider the first approach. In the meta-algorithm, such normalizing transformations are defined as $q(t^{(\ell)}_{p\,|\,\hat{c}})$ for the test statistic at layer ℓ conditioned on the predicted class $\hat{c}$, and $q(t^{(\ell)}_{s\,|\,c}),\ \forall c \in [m]$ for the test statistic at layer ℓ conditioned on each candidate true class. For each pair of layers $(\ell_1, \ell_2) \in \mathcal{L}^2$, $q(t^{(\ell_1)}_{p\,|\,\hat{c}}, t^{(\ell_2)}_{p\,|\,\hat{c}})$ and $q(t^{(\ell_1)}_{s\,|\,c}, t^{(\ell_2)}_{s\,|\,c}),\ \forall c \in [m]$ define the normalizing transformations for the corresponding test statistic pair conditioned on the predicted class $\hat{c}$ and on each candidate true class respectively. Since it is not efficient to include all the layer pairs beyond few tens of layers, this is specified as optional in Algorithm 1.

In the second approach, $q(\mathbf{t}_{p\,|\,\hat{c}})$ and $q(\mathbf{t}_{s\,|\,c}),\ \forall c \in [m]$ define the normalizing transformations for a vector of test statistics from the layers conditioned on the predicted class $\hat{c}$ and on each candidate true class respectively [3]. In § 4.2, we propose specific realizations for each case of the above normalizing transformations based on *class-conditional p-values*. They have the advantage of being nonparametric, and transform the test statistics into probabilities that, for natural inputs, will be approximately uniform on $[0, 1]$.

III. Layerwise Aggregation and Scoring: The normalized test statistics can be interpreted as anomaly scores that are each based on information from one or more layer representations and a specific (predicted or true) class. The goal of a scoring function is to aggregate the multiple anomaly scores in a principled way such that the *combined score is low* for inputs following the same distribution as normal inputs to the DNN, and *high for anomalies*. Prior works have taken approaches such as average or maximum of the normalized test statistics (Miller et al., 2019; Sastry & Oore, 2020), or a weighted sum of unnormalized test statistics, with the weights trained using a binary logistic classifier (Lee et al., 2018; Ma et al., 2018; Yang et al., 2020). In our meta-algorithm, we define an aggregation function $r(\cdot)$ that combines the set of all normalized test statistics from the individual layers and (optionally) layer pairs as follows: $q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}}) = r(Q_{p\,|\,\hat{c}})$ and $q_{\text{agg}}(\mathbf{t}_{s\,|\,c}) = r(Q_{s\,|\,c}),\ \forall c \in [m]$, where

$$\begin{aligned} Q_{p\,|\,\hat{c}} &= \{q(t^{(\ell)}_{p\,|\,\hat{c}}),\ \forall \ell \in \mathcal{L}\} \\ &\quad \cup\ \{q(t^{(\ell_1)}_{p\,|\,\hat{c}}, t^{(\ell_2)}_{p\,|\,\hat{c}}),\ \forall (\ell_1, \ell_2) \in \mathcal{L}^2\} \quad \text{and} \qquad (1) \\ Q_{s\,|\,c} &= \{q(t^{(\ell)}_{s\,|\,c}),\ \forall \ell \in \mathcal{L}\} \\ &\quad \cup\ \{q(t^{(\ell_1)}_{p\,|\,c}, t^{(\ell_2)}_{p\,|\,c}),\ \forall (\ell_1, \ell_2) \in \mathcal{L}^2\},\ \ \forall c \in [m] \end{aligned}$$

define the sets of normalized test statistics.

Motivated by ideas from multiple testing, *we propose specific aggregation functions* $r(\cdot)$ in § 4.3 for combining multiple p-values from the layers and layer pairs. For the normalization approach that directly transforms the test statistic vector from the layers, there is no need for an aggregation function; we simply set $q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}}) = q(\mathbf{t}_{p\,|\,\hat{c}})$ and $q_{\text{agg}}(\mathbf{t}_{s\,|\,c}) = q(\mathbf{t}_{s\,|\,c}),\ \forall c \in [m]$. The final anomaly score in the meta-algorithm is defined to be a simple function of the aggregate, normalized test statistics, *i.e.*, $S(q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}}), q_{\text{agg}}(\mathbf{t}_{s\,|\,1}), \cdots, q_{\text{agg}}(\mathbf{t}_{s\,|\,m}), \hat{c})$. We propose specific realizations of the score functions for adversarial and OOD detection in § 4.4.

IV. Detection Decision: The detection decision for a test input $\mathbf{x}$ predicted into class $\hat{c}$ is obtained by thresholding the final anomaly score as follows:

$$\begin{aligned} &\psi_\tau(\mathbf{x}^{(0)}, \cdots, \mathbf{x}^{(L)}, \hat{c}) = \\ &\mathbb{1}\left[S(q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}}), q_{\text{agg}}(\mathbf{t}_{s\,|\,1}), \cdots, q_{\text{agg}}(\mathbf{t}_{s\,|\,m}), \hat{c}) \geq \tau\right] \quad (2) \end{aligned}$$

where decisions 0 and 1 correspond to natural and anomalous inputs respectively, and $\mathbb{1}[\cdot]$ is the indicator function. The threshold τ is usually set based on a false positive rate (FPR) that is suitable for the target application. In order to operate the detector at an FPR $\alpha \in (0, 1)$ (usually a small value *e.g.*, 0.01), the threshold can be set by estimating the FPR $\widehat{P}_{\text{F}}(\tau)$ from the set of natural inputs $\mathcal{D}_a$ as follows:

$$\begin{aligned} \tau_\alpha &= \sup\{\tau \in \mathbb{R} \,:\, \widehat{P}_{\text{F}}(\tau) \leq \alpha\}, \text{ where} \\ \widehat{P}_{\text{F}}(\tau) &= \frac{1}{N} \sum_{n=1}^{N} \psi_\tau(\mathbf{x}^{(0)}_n, \cdots, \mathbf{x}^{(L)}_n, \hat{c}_n). \qquad (3) \end{aligned}$$

This threshold choice ensures that natural inputs are accepted by the detector with a probability close to $1 - \alpha$.

[3] Although the function $q(\cdot)$ is different depending on the class, layer(s), and the number of test statistic inputs, we use the same overloaded notation for clarity.

4. A Realization of the Meta-algorithm

In this section, we propose concrete methods for realizing the components of the anomaly detection meta-algorithm.

4.1. Test Statistic Based on kNN Class Counts

Consider a set of natural inputs to the DNN that are predicted into a class $\hat{c} \in [m]$. The class counts from the kNN of its representations from a layer $\ell \in \mathcal{L}$ are expected to follow a certain distribution, wherein class $\hat{c}$ has a higher probability than the other classes. A similar observation can be made for natural inputs from a candidate true class $c \in [m]$. Let $(k_1^{(\ell)}, \cdots, k_m^{(\ell)})$ denote the tuple of class counts from the kNN $N_k^{(\ell)}(\mathbf{x}^{(\ell)})$ of a layer representation $\mathbf{x}^{(\ell)}$, such that $k_i^{(\ell)} \in \{0, 1, \cdots, k\}$ and $\sum_{i=1}^{m} k_i^{(\ell)} = k$. The null (natural) distribution of the kNN class counts at a layer ℓ conditioned on the predicted class $\hat{c}$ can be modeled using the following multinomial distribution

$$p(k_1^{(\ell)}, \cdots, k_m^{(\ell)} \,|\, \widehat{C} = \hat{c}) \;=\; k! \prod_{i=1}^{m} \frac{[\pi_{i\,|\,\hat{c}}^{(\ell)}]^{k_i^{(\ell)}}}{k_i^{(\ell)}!}, \qquad (4)$$

where $(\pi_{1\,|\,\hat{c}}^{(\ell)}, \cdots, \pi_{m\,|\,\hat{c}}^{(\ell)})$ are the multinomial probability parameters specific to class $\hat{c}$ and layer ℓ (they are non-negative and sum to 1). We estimate these parameters from the labeled subset $\widehat{\mathcal{D}}_a(\ell, \hat{c})$ using maximum-a-posteriori (MAP) estimation with the Dirichlet conjugate prior distribution (Barber, 2012) [4]. For a test input $\mathbf{x}$ sampled from the natural data distribution that is predicted into class $\hat{c}$ by the DNN, we expect the multinomial distribution (4) to be a good fit for the class counts observed from its kNN at layer ℓ. In order to test whether the observed class counts $(k_1^{(\ell)}, \cdots, k_m^{(\ell)})$ from layer ℓ given predicted class $\hat{c}$ are consistent with distribution (4), we apply the well-known multinomial LRT (Read & Cressie, 2012), whose log-likelihood ratio statistic is given by

$$T(\mathbf{x}^{(\ell)}, \hat{c}, \widehat{\mathcal{D}}_a(\ell, \hat{c})) \;=\; \sum_{i=1}^{m} k_i^{(\ell)} \,\log \frac{k_i^{(\ell)}}{k\, \pi_{i\,|\,\hat{c}}^{(\ell)}}. \qquad (5)$$

This test statistic is a class count deviation measure which is always non-negative, with larger values corresponding to a larger deviation from the null distribution (4). In a similar way, the test statistics conditioned on each candidate true class are defined as

$$T(\mathbf{x}^{(\ell)}, c, \mathcal{D}_a(\ell, c)) \;=\; \sum_{i=1}^{m} k_i^{(\ell)} \,\log \frac{k_i^{(\ell)}}{k\, \widetilde{\pi}_{i\,|\,c}^{(\ell)}}, \;\forall c \in [m]. \qquad (6)$$

Here, $(\widetilde{\pi}_{1\,|\,c}^{(\ell)}, \cdots, \widetilde{\pi}_{m\,|\,c}^{(\ell)})$ are the multinomial parameters specific to class c and layer ℓ, which are estimated from the corresponding data subset $\mathcal{D}_a(\ell, c)$.

[4]We set the prior counts of the Dirichlet distribution to a small non-zero value to avoid 0 estimates for the multinomial parameters.

4.2. Normalizing Transformations Based on p-values

Recall that we are interested in designing a normalizing transformation that, for natural inputs, makes the transformed test statistics across the layers and classes follow the same distribution. One such approach is to use the p-value, that calculates the probability of a test statistic taking values (as or) more extreme than the observed value. More generally, a p-value is defined as any transformation of the test statistic (possibly a vector), following the null hypothesis distribution, into a uniformly distributed probability (Root et al., 2016). This provides a simple approach for normalizing the class-conditional test statistics in both the univariate and multivariate cases, as discussed next.

A. p-values at Individual Layers and Layer Pairs

For an input predicted into class $\hat{c}$ with class-conditional test statistics at a layer ℓ given by $t_{p\,|\,\hat{c}}^{(\ell)}, t_{s\,|\,1}^{(\ell)}, \cdots, t_{s\,|\,m}^{(\ell)}$, the normalizing p-value transformations are defined as [5]:

$$\begin{aligned} q(t_{p\,|\,\hat{c}}^{(\ell)}) \;&=\; \mathbb{P}(\,T_{p\,|\,\hat{c}}^{(\ell)} \geq t_{p\,|\,\hat{c}}^{(\ell)} \,|\, \widehat{C} = \hat{c}\,) \\ q(t_{s\,|\,c}^{(\ell)}) \;&=\; \mathbb{P}(\,T_{s\,|\,c}^{(\ell)} \geq t_{s\,|\,c}^{(\ell)} \,|\, C = c\,), \;\; \forall c \in [m]. \end{aligned} \qquad (7)$$

Similarly the normalizing p-value transformations for test statistic pairs from layers (ℓ_1, ℓ_2) are defined as:

$$\begin{aligned} &q(t_{p\,|\,\hat{c}}^{(\ell_1)}, t_{p\,|\,\hat{c}}^{(\ell_2)}) \;=\; \mathbb{P}(\,T_{p\,|\,\hat{c}}^{(\ell_1)} \geq t_{p\,|\,\hat{c}}^{(\ell_1)}, T_{p\,|\,\hat{c}}^{(\ell_2)} \geq t_{p\,|\,\hat{c}}^{(\ell_2)} \,|\, \widehat{C} = \hat{c}\,) \\ &q(t_{s\,|\,c}^{(\ell_1)}, t_{s\,|\,c}^{(\ell_2)}) \;=\; \\ &\mathbb{P}(\,T_{s\,|\,c}^{(\ell_1)} \geq t_{s\,|\,c}^{(\ell_1)}, T_{s\,|\,c}^{(\ell_2)} \geq t_{s\,|\,c}^{(\ell_2)} \,|\, C = c\,), \;\; \forall c \in [m]. \end{aligned} \qquad (8)$$

Since the class-conditional distributions of the test statistics are unknown, we estimate the p-values using the empirical cumulative distribution function of the test statistics calculated from the corresponding data subsets of $\mathcal{D}_a$ [6].

B. Multivariate p-value Based Normalization

In this approach, we consider the class-conditional joint density of a test statistic vector from the layers, and propose a normalizing transformation $q : \mathbb{R}^{L+1} \mapsto [0, 1]$ based on the idea of multivariate p-values. Consider an input predicted into a class $\hat{c}$, that has a vector of test statistics $\mathbf{t}_{p\,|\,\hat{c}} = \mathbf{t}$ from the layers. Suppose $f_0(\mathbf{t}_{p\,|\,\hat{c}} \,|\, \hat{c})$ denotes the true null-hypothesis density of $\mathbf{t}_{p\,|\,\hat{c}}$ conditioned on the predicted class $\hat{c}$, then the region outside the level set of constant density equal to $f_0(\mathbf{t} \,|\, \hat{c})$ is given by $\{\mathbf{t}_{p\,|\,\hat{c}} \in \mathbb{R}^{L+1} \;:\; f_0(\mathbf{t}_{p\,|\,\hat{c}} \,|\, \hat{c}) \;<\; f_0(\mathbf{t} \,|\, \hat{c})\}$. The multivariate p-value for $\mathbf{t}$ is the probability of this region under the null hypothesis probability measure (Root et al., 2016).

We use the averaged localized p-value estimation method using kNN graphs (aK-LPE) proposed by (Qian & Saligrama,

[5]We use one-sided, right-tailed p-values since larger values of the test statistic correspond to a larger deviation.

[6]In our implementation, we averaged the p-value estimates from a hundred bootstrap samples in order to reduce the variance.

2012). The main idea is to define a score function based on nearest neighbor graphs $G(\mathbf{t})$ that captures the local relative density around $\mathbf{t}$. They show that a score function defined as the average distance from $\mathbf{t}$ to its $\frac{k}{2}$-th through $\frac{3k}{2}$-th nearest neighbors provides the following asymptotically-consistent p-value estimate:

$$q_{\text{lpe}}(\mathbf{t}) \;=\; \frac{1}{|\mathcal{D}_t|} \sum_{\mathbf{t}_n \in \mathcal{D}_t} \mathbb{1}[G(\mathbf{t}) \leq G(\mathbf{t}_n)], \qquad (9)$$

where $\mathcal{D}_t$ is a large sample of test statistic vectors corresponding to natural inputs. In our problem, we apply the above p-value transformation (using the appropriate data subsets) to normalize the $m+1$ test statistic vectors giving: $q_{\text{lpe}}(\mathbf{t}_{p\,|\,\hat{c}}), q_{\text{lpe}}(\mathbf{t}_{s\,|\,1}), \cdots, q_{\text{lpe}}(\mathbf{t}_{s\,|\,m})$.

4.3. Aggregation of p-values

The p-value based normalized test statistics capture the extent of deviation of the test statistics of an input relative to their distribution on natural inputs; smaller p-values correspond to a larger deviation. For approach A in § 4.2, we can consider each p-value to correspond to a hypothesis test involving a particular layer or layer pair. We are interested in combining the evidence from these multiple tests (Dudoit & Van Der Laan, 2007) into a single p-value for the overall problem of testing for natural versus anomalous inputs. We investigate two methods for combining p-values from multiple tests and define the corresponding aggregation functions. Note that there is no need to aggregate p-values for approach B in § 4.2, and we simply set $q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}}) = q_{\text{lpe}}(\mathbf{t}_{p\,|\,\hat{c}})$ and $q_{\text{agg}}(\mathbf{t}_{s\,|\,c}) = q_{\text{lpe}}(\mathbf{t}_{s\,|\,c}),\ \forall c \in [m]$.

Fisher's method (Fisher, 1992) provides a principled way of combining p-values from multiple independent tests based on the idea that, under the null hypothesis, the sum of the log of multiple p-values follows a χ^2-distribution. The aggregate p-value function based on this method is given by

$$\log\, q_{\text{fis}}(\mathbf{t}) \;=\; \log\, r(Q) \;=\; \sum_{q \in Q} \log q, \qquad (10)$$

where Q is one of the sets $Q_{p\,|\,\hat{c}}$ or $Q_{s\,|\,c}$ defined in Eq. (1), and $\mathbf{t}$ is the corresponding test statistic vector [7]. An apparent weakness of Fisher's method is its assumption of independent p-values. We briefly provide the aggregate p-value function for an alternate harmonic mean p-value (HMP) method for combining p-values from multiple dependent tests (Wilson, 2019), and discuss its details in Appendix B.

$$q_{\text{hmp}}(\mathbf{t})^{-1} \;=\; r(Q)^{-1} \;=\; \sum_{q \in Q} q^{-1}. \qquad (11)$$

4.4. Scoring for Adversarial and OOD Detection

We propose different score functions for detecting adversarial and general OOD inputs. An adversarial input predicted into class $\hat{c}$ by the DNN is expected to be anomalous at one or more of its layer representations relative to the distribution of natural inputs predicted into the same class. This implies that its aggregate p-value conditioned on the predicted class, $q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}})$, should have a small value. Moreover, since the adversarial input was created from a source class different from $\hat{c}$, it is expected to be a typical sample relative to the distribution of natural inputs from the unknown source class $c \neq \hat{c}$. This implies that its aggregate p-value conditioned on a candidate true class (different from $\hat{c}$) should have a relatively large value. Combining these ideas, we define the score function for adversarial inputs as

$$S(q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}}), q_{\text{agg}}(\mathbf{t}_{s\,|\,1}), \cdots, q_{\text{agg}}(\mathbf{t}_{s\,|\,m}), \hat{c}) \;=\; \log\left(\frac{\max\limits_{c \in [m] \setminus \{\hat{c}\}} q_{\text{agg}}(\mathbf{t}_{s\,|\,c})}{q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}})}\right). \qquad (12)$$

The table below provides additional insight on this score function by considering the numerator and denominator terms (inside the log) for different categories of input.

Input type & prediction	Numerator	Denominator	Score
$\mathbf{x}$ natural, $\widehat{C}(\mathbf{x}) = c$	Low	High	Low
$\mathbf{x}$ natural, $\widehat{C}(\mathbf{x}) \neq c$	High	High	Medium
$\mathbf{x}$ adversarial, $\widehat{C}(\mathbf{x}) \neq c$	High	Low	High

Similar to adversarial inputs, OOD inputs are also expected to exhibit anomalous patterns at the layers of the DNN relative to the distribution of natural inputs predicted into the same class. Since OOD inputs are not created by intentionally perturbing natural inputs from a true class different from the predicted class, we simplify score function (12) for OOD detection as follows

$$\mathrm{S}(q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}}), q_{\text{agg}}(\mathbf{t}_{s\,|\,1}), \cdots, q_{\text{agg}}(\mathbf{t}_{s\,|\,m}), \hat{c}) \;=\; -\log\, q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}}).$$

OOD inputs are expected to have a low aggregate p-value $q_{\text{agg}}(\mathbf{t}_{p\,|\,\hat{c}})$, and hence a high value for the above score.

4.5. Implementation and Computational Complexity

We briefly discuss some practical aspects of implementing `JTLA` efficiently. We apply the neighborhood preserving projection method (He et al., 2005) to perform dimensionality reduction on the DNN layer representations since they can be very high dimensional (details in Appendix D.3). In order to efficiently construct and query from kNN graphs at the layer representations, we use the fast approximate nearest neighbors method *NNDescent* (Dong et al., 2011) [8]. Together, these two techniques significantly reduce the memory utilization and running time of `JTLA`.

The computational complexity of the proposed instantiation of `JTLA` at prediction (test) time can be expressed as

[7] $q_{\text{lpe}}(\cdot)$, $q_{\text{fis}}(\cdot)$, and $q_{\text{hmp}}(\cdot)$ are specific instances of $q_{\text{agg}}(\cdot)$.

[8] We use the following implementation of NNDescent: https://github.com/lmcinnes/pynndescent

$\text{O}\left(L\left(d_{\max} N^{\rho} + m^2 + B\,N\right)\right)$ when layer pairs are not used, and $\text{O}\left(L^2\left(d_{\max} N^{\rho} + m^2 + B\,N\right)\right)$ when layer pairs are used. Here $d_{\max}$ is the maximum dimension of the projected layer representations, m is the number of classes, N is the number of samples, B is the number of bootstrap replications used for estimating p-values, and $\rho \in (0, 1)$ is an unknown factor associated with the approximate nearest neighbor queries (that are sub-linear in N). The p-value calculation can be made faster and independent of N by pre-computing the empirical class-conditional CDFs. A comparison of the running time of `JTLA` with other detection methods can be found in Appendix E.6.

5. Defense-Aware Adaptive Attack

The importance of evaluating adversarial detection methods against an adaptive, defense-aware adversary has been highlighted in prior works (Carlini & Wagner, 2017a; Athalye et al., 2018; Tramèr et al., 2020). We consider a gray-box adversary that is assumed to have full knowledge of the DNN architecture and parameters, and partial knowledge of the detection method [9].

Consider a clean input sample $\mathbf{x}$ from class c that is correctly classified by the DNN. Let η_ℓ denote the distance between $\mathbf{x}^{(\ell)} = \mathbf{f}_\ell(\mathbf{x})$ and its k-th nearest neighbor from layer ℓ. The number of samples from any class i among the kNNs of $\mathbf{x}^{(\ell)}$, relative to the dataset $\mathcal{D}_a$, can be expressed as

$$k_i^{(\ell)} = \sum_{n=1\,:\,c_n=i}^{N} u(\eta_\ell - d(\mathbf{f}_\ell(\mathbf{x}), \mathbf{f}_\ell(\mathbf{x}_n))),\;\; i = 1, \cdots, m,$$

where $u(\cdot)$ is the unit step function. Consider the following probability mass function over the class labels: $p_i = k_i \,/\, \sum_{j=1}^{m} k_j,\;\; i \in [m]$, where $k_i = \sum_{\ell=0}^{L} k_i^{(\ell)}$ is the cumulative kNN count from class i across the layers. In order to fool a defense method relying on the kNN class counts from the layer representations, our attack finds an adversarial input $\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta}$ with target class $c' \neq c$ that minimizes the following log-ratio of probabilities:

$$\log \frac{p_c}{p_{c'}} = \log k_c - \log k_{c'} = \log \sum_{\ell=0}^{L} k_c^{(\ell)} - \log \sum_{\ell=0}^{L} k_{c'}^{(\ell)}, \tag{13}$$

subject to a penalty on the norm of the perturbation $\boldsymbol{\delta}$ [10]. To address the non-smoothness arising from the step function in the class counts, we use the Gaussian (RBF) kernel $h_\sigma(\mathbf{x}, \mathbf{y}) = e^{-\frac{1}{\sigma^2} d(\mathbf{x},\mathbf{y})^2}$ to obtain a smooth approximation of the class counts $k_i^{(\ell)}$. The attack objective function to minimize is a weighted sum of the ℓ_2-perturbation norm and the kernel-smoothed log-ratio of probabilities, given by

$$J(\boldsymbol{\delta}) = \|\boldsymbol{\delta}\|_2^2 + \lambda \log \sum_{\ell=0}^{L} \sum_{\substack{n=1\,: \\ c_n=c}}^{N} h_{\sigma_\ell}(\mathbf{f}_\ell(\mathbf{x}+\boldsymbol{\delta}), \mathbf{f}_\ell(\mathbf{x}_n))$$
$$- \lambda \log \sum_{\ell=0}^{L} \sum_{\substack{n=1\,; \\ c_n=c'}}^{N} h_{\sigma_\ell}(\mathbf{f}_\ell(\mathbf{x}+\boldsymbol{\delta}), \mathbf{f}_\ell(\mathbf{x}_n)). \tag{14}$$

Here $\sigma_\ell > 0$ is the kernel scale for layer ℓ and $\lambda > 0$ is a constant that sets the relative importance of the terms in the objective function. The method used for setting the kernel scale per layer and minor extensions of the proposed attack are described in Appendix C. Details of the optimization method and the choice of λ are given in Appendix D.4. In our experiments, we chose the class with the second highest probability predicted by the DNN as the target attack class.

6. Experimental Results

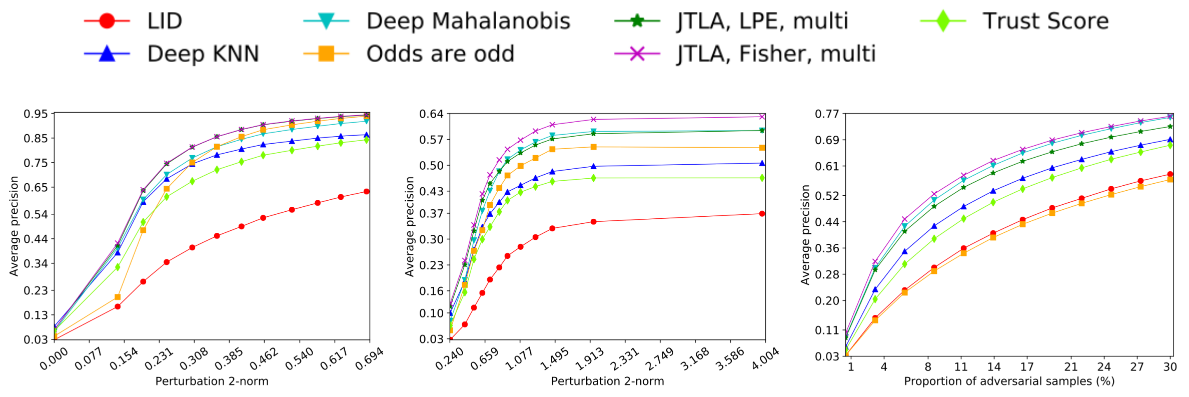
We evaluated `JTLA` on the following well-known image classification datasets: CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and MNIST (LeCun et al., 1998). We used the training partition provided by the datasets for training standard CNN architectures, including a Resnet for CIFAR-10. We performed class-stratified 5-folds cross-validation on the test partition provided by the datasets; the training folds are used for estimating the detector parameters, and the test folds are used solely for calculating performance metrics (which are then averaged across the test folds). We used the Foolbox library (Rauber et al., 2017) for generating adversarial samples from the following attack methods: (i) Projected Gradient Descent (PGD) with ℓ_∞ norm (Madry et al., 2018), (ii) Carlini-Wagner (CW) attack with ℓ_2 norm (Carlini & Wagner, 2017b), and (iii) Fast gradient sign method (FGSM) with ℓ_∞ norm (Goodfellow et al., 2015). We also implement and generate adversarial samples from the adaptive attack proposed in § 5. More details on the datasets, DNN architectures, and the attack parameters used are provided in Appendix D.

Methods Compared. We evaluated the following two variants of `JTLA` using the multinomial test statistic: 1) p-value normalization at the layers and layer pairs using Fisher's method for aggregation, 2) multivariate p-value normalization based on the aK-LPE method. The score functions from § 4.4 for adversarial and OOD detection are used for the respective tasks. The number of nearest neighbors is the **only hyperparameter** of the proposed instantiation of `JTLA`. This is set to be a function of the number of in-distribution training samples n using the heuristic $k = \lceil n^{0.4} \rceil$.

We compared against the following recently-proposed methods: (i) Deep Mahalanobis detector (`Mahalanobis`) (Lee et al., 2018), (ii) Local Intrinsic Dimensionality detector (`LID`) (Ma et al., 2018), (iii) The odds are odd detector (`Odds`) (Roth et al., 2019), (iv) Deep kNN (`DKNN`) (Paper-

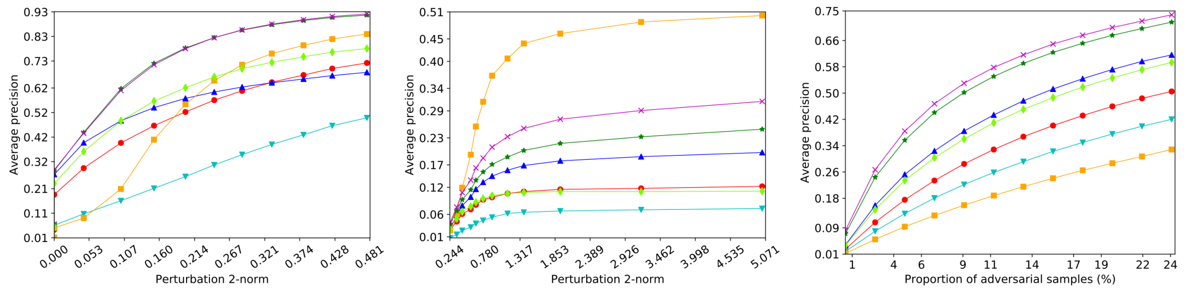
[9]For example, the detection threshold and specific layers of the DNN used may be unknown.

[10]A similar type of attack on kNN-based models has been recently proposed in (Sitawarin & Wagner, 2020).



(a) CW, confidence = 0 (b) Adaptive attack (c) PGD, $\epsilon = 1/255$

Figure 2: Adversarial detection performance on CIFAR-10 under different attacks.



(a) CW, confidence = 0 (b) Adaptive attack (c) PGD, $\epsilon = 1/255$

Figure 3: Adversarial detection performance on SVHN under different attacks.

not & McDaniel, 2018), and (v) Trust Score (`Trust`) (Jiang et al., 2018). `Mahalanobis` and `LID` are supervised (they utilize adversarial or outlier data from the training folds), while the remaining methods are unsupervised. `LID` and `Odds` are excluded from the OOD detection experiment because they specifically address adversarial samples. Details on the implementation, hyperparameters, and layer representations used by the methods can be found in Appendix D.3.

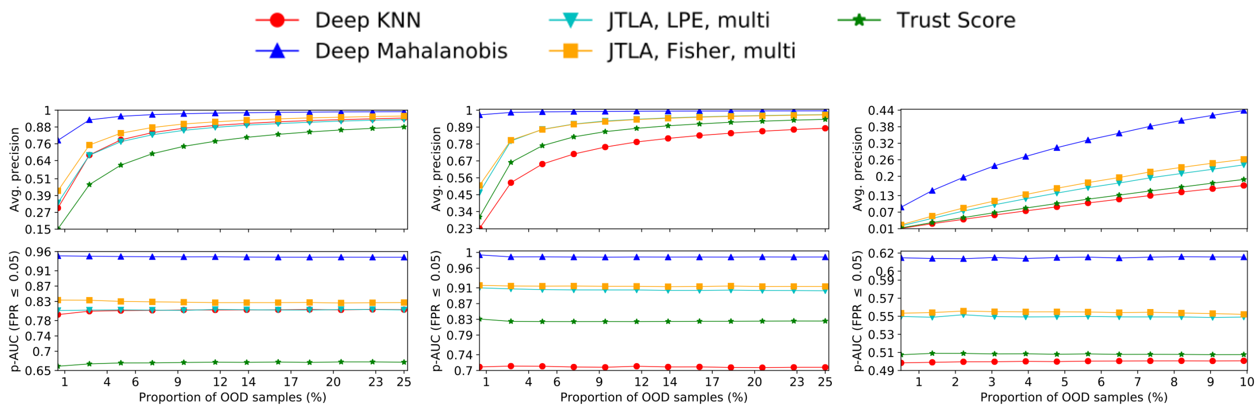
Performance metrics. We evaluate detection performance using the precision-recall (PR) curve (Davis & Goadrich, 2006; Flach & Kull, 2015) and the receiver operating characteristic (ROC) curve (Fawcett, 2006). We use *average precision* as a threshold-independent metric to summarize the PR curve, and partial area under the ROC curve below FPR α (*pAUC*-α) as the metric to summarize low-FPR region of the ROC curve. Note that both the metrics do not require the selection of a threshold. We do not report the area under the entire ROC curve because it is skew-insensitive and tends to have optimistic values when the fraction of anomalies is very small (Ahmed & Courville, 2020).

6.1. Detecting Adversarial Samples

Figures 2 and 3 show the average precision of the detection methods as a function of the perturbation ℓ_2 norm of the adversarial samples generated by the CW (confidence = 0) and adaptive attack methods. For the PGD attack ($\epsilon = 1\,/\,255$), the proportion of adversarial samples is shown on the x-axis instead of the perturbation norm because most of the samples from this attack have the same norm value. We observe that in almost all cases, `JTLA` outperforms the other baselines. Methods such as `Mahalanobis`, `Odds`, and `DKNN` perform well in some cases but fail on others, while `LID` performs poorly in nearly all scenarios. We observe an outlying trend in Figure 3b, where `Odds` outperforms `JTLA` on the adaptive attack applied to SVHN. However, a comparison of the pAUC-0.2 metric for this scenario (Figure 10 in Appendix E.4) reveals that `JTLA` has higher pAUC-0.2 for low perturbation norm (where adversarial samples are likely to be more realistic and harder to detect). We provide additional results in Appendix E that include: (i) attack transfer and attacks of varying strength, (ii) evaluation of the pAUC-0.2 metric, (iii) results on the MNIST dataset, and (iv) results on the FGSM attack.

6.2. Detecting Out-Of-Distribution Samples

We evaluated OOD detection using the following image dataset pairs, with the first dataset used as in-distribution (inliers) and the second dataset used as out-distribution (outliers): 1) MNIST vs. Not-MNIST (Bulatov, 2011), 2) CIFAR-10 vs. SVHN. While a majority of papers on OOD



(a) CIFAR-10 vs. SVHN (b) MNIST vs. not-MNIST (c) CIFAR-10 vs. CIFAR-100

Figure 4: Comparison of OOD detection performance.

detection evaluate using such dataset pairs, the importance of evaluating against outliers that are semantically meaningful has been emphasized by Ahmed & Courville (2020). Therefore, we performed an experiment where object classes from the CIFAR-100 dataset (not in CIFAR-10) are treated as outliers relative to CIFAR-10. This is a more realistic and challenging task since novel object categories can be considered semantically-meaningful anomalies. Using the same 5-fold cross-validation setup, we compared the performance of `JTLA` with `Mahalanobis`, `DKNN`, and `Trust` (`Odds` and `LID` are excluded because they focus on adversarial examples). Since `Mahalanobis` is a supervised method, it uses both inlier and outlier data from the training folds, while the remaining methods (all unsupervised) use only the inlier data from the training folds. To promote fairness in the comparison, we excluded outlier data corresponding to one half of the classes from the training folds, and included outlier data from *only* the excluded classes in the test folds. Additionally, in the test folds we included image samples with random pixel values uniformly selected from the same range as valid images. The number of random samples is set equal to the average number of test-fold samples from a single class. Figure 4 shows the average precision and pAUC-0.05 as a function of the proportion of OOD samples on the OOD detection tasks [11]. We observe that `JTLA` outperforms the unsupervised methods `DKNN` and `Trust` in all cases, but does not achieve the very good performance of `Mahalanobis`. This should be considered in light of the fact that `Mahalanobis` uses outlier samples from the training folds to train a classifier and tune a noise parameter (Lee et al., 2018). However, in real-world settings, one is unlikely to have the prior knowledge and sufficient number (and variety) of outlier samples for training.

[11]We report pAUC below 5% FPR because the methods achieve high detection rates at very low FPR.

6.3. Ablation Studies

We performed ablation studies to gain a better understanding of the different components of the proposed method. Specifically, we evaluated 1) the relative performance of the proposed p-value based normalization and aggregation methods, 2) the performance improvement from testing at layer pairs in addition to the individual layers, 3) the relative performance of using only the last few layers compared to using all the layer representations, and 4) the relative performance of the two scoring methods in § 4.4. These results are discussed in Appendix E.2.

7. Conclusions

We presented `JTLA`, a general framework for detecting anomalous inputs to a DNN classifier based on joint statistical testing of its layer representations. We presented a general meta-algorithm for this problem, and proposed specific methods for realizing the components of this algorithm in a principled way. The construction of `JTLA` is modular, allowing it to be used with a variety of test statistics proposed in prior works. Extensive experiments with strong adversarial attacks (including an adaptive defense-aware attack we proposed) and anomalous inputs to DNN image classifiers demonstrate the effectiveness of our method.

Acknowledgements

We thank the anonymous reviewers for their useful feedback that helped improve the paper. VC, JR, and SB were supported in part through the following US NSF grants: CNS-1838733, CNS-1719336, CNS-1647152, CNS-1629833, CNS-1942014, CNS-2003129, and an award from the US Department of Commerce with award number 70NANB21H043. SJ was partially supported by Air Force Grant FA9550-18-1-0166, the NSF Grants CCF-FMitF-1836978, SaTC-Frontiers-1804648 and CCF-1652140, and ARO grant number W911NF-17-1-0405.

References

Ahmed, F. and Courville, A. C. Detecting semantic anomalies. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, NY, USA, February 7-12, 2020*, pp. 3154–3162. AAAI Press, 2020.

Akhtar, N. and Mian, A. S. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M. E., Kawarabayashi, K.-i., and Nett, M. Estimating local intrinsic dimensionality. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 29–38. ACM, 2015.

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 274–283. PMLR, 2018.

Barber, D. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012. ISBN 0521518148.

Barreno, M., Nelson, B., Sears, R., Joseph, A. D., and Tygar, J. D. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pp. 16–25. ACM, 2006. doi: 10.1145/1128817.1128824.

Biggio, B. and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018. doi: 10.1016/j.patcog.2018.07.023.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML*. icml.cc / Omnipress, 2012.

Bulatov, Y. NotMNIST dataset. http://yaroslavvb.com/upload/notMNIST/, 2011.

Bulusu, S., Kailkhura, B., Li, B., Varshney, P. K., and Song, D. Anomalous instance detection in deep learning: A survey. *CoRR*, abs/2003.06979, 2020. URL https://arxiv.org/abs/2003.06979.

Carlini, N. and Wagner, D. A. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS*, pp. 3–14. ACM, 2017a. doi: 10.1145/3128572.3140444.

Carlini, N. and Wagner, D. A. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 39–57. IEEE Computer Society, 2017b. doi: 10.1109/SP.2017.49.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3): 1–58, 2009.

Davis, J. and Goadrich, M. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, 2006.

Dong, W., Moses, C., and Li, K. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pp. 577–586, 2011.

Dudoit, S. and Van Der Laan, M. J. *Multiple testing procedures with applications to genomics*. Springer Science & Business Media, 2007.

Fawcett, T. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Fawzi, A., Moosavi-Dezfooli, S., and Frossard, P. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pp. 1624–1632, 2016.

Fawzi, A., Fawzi, O., and Frossard, P. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, 2018.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *CoRR*, abs/1703.00410, 2017. URL http://arxiv.org/abs/1703.00410.

Fisher, R. A. Statistical methods for research workers. In *Breakthroughs in statistics*, pp. 66–70. Springer, 1992.

Flach, P. and Kull, M. Precision-recall-gain curves: PR analysis done right. In *Advances in neural information processing systems*, pp. 838–846, 2015.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, Conference Track Proceedings*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, X., Cai, D., Yan, S., and Zhang, H.-J. Neighborhood preserving embedding. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pp. 1208–1213. IEEE, 2005.

Hein, M., Andriushchenko, M., and Bitterwolf, J. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 41–50. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00013.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, Conference Track Proceedings*. OpenReview.net, 2017.

Jha, S., Raj, S., Fernandes, S. L., Jha, S. K., Jha, S., Jalaian, B., Verma, G., and Swami, A. Attribution-based confidence metric for deep neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, pp. 11826–11837, 2019.

Jiang, H., Kim, B., Guan, M. Y., and Gupta, M. R. To trust or not to trust a classifier. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pp. 5546–5557, 2018.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, Conference Track Proceedings*. OpenReview.net, 2017.

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

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pp. 7167–7177, 2018.

Li, X. and Li, F. Adversarial examples detection in deep networks with convolutional filter statistics. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 5775–5783. IEEE Computer Society, 2017.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S. N. R., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. In *6th International Conference on Learning Representations, Conference Track Proceedings*. OpenReview.net, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, Conference Track Proceedings*. OpenReview.net, 2018.

Meng, D. and Chen, H. MagNet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 135–147. ACM, 2017.

Miller, D. J., Wang, Y., and Kesidis, G. When not to classify: Anomaly detection of attacks (ADA) on DNN classifiers at test time. *Neural Computation*, 31(8):1624–1670, 2019.

Miller, D. J., Xiang, Z., and Kesidis, G. Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, 108(3):402–433, 2020.

Moosavi-Dezfooli, S., Fawzi, A., and Frossard, P. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.282.

Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 86–94. IEEE Computer Society, 2017.

Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS*, pp. 27–38. ACM, 2017. doi: 10.1145/3128572.3140451.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Nguyen, A. M., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 427–436. IEEE Computer Society, 2015.

Papernot, N. and McDaniel, P. D. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *CoRR*, abs/1803.04765, 2018. URL `http://arxiv.org/abs/1803.04765`.

Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P*, pp. 372–387. IEEE, 2016. doi: 10.1109/EuroSP.2016.36.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Qian, J. and Saligrama, V. New statistic in p-value estimation for anomaly detection. In *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 393–396. IEEE, 2012. doi: 10.1109/SSP.2012.6319713.

Rauber, J., Brendel, W., and Bethge, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

Read, T. R. and Cressie, N. A. *Goodness-of-fit statistics for discrete multivariate data*. Springer Science & Business Media, 2012.

Root, J., Saligrama, V., and Qian, J. Learning minimum volume sets and anomaly detectors from KNN graphs. *CoRR*, abs/1601.06105, 2016. URL `http://arxiv.org/abs/1601.06105`.

Roth, K., Kilcher, Y., and Hofmann, T. The odds are odd: A statistical test for detecting adversarial examples. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5498–5507. PMLR, 2019.

Ruder, S. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL `http://arxiv.org/abs/1609.04747`.

Sastry, C. S. and Oore, S. Detecting out-of-distribution examples with gram matrices. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8491–8501. PMLR, 2020.

Sitawarin, C. and Wagner, D. A. Minimum-norm adversarial examples on KNN and KNN-based models. In *IEEE Security and Privacy Workshops, SP Workshops*, pp. 34–40. IEEE, 2020. doi: 10.1109/SPW50608.2020.00023.

Steinhardt, J., Koh, P. W., and Liang, P. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pp. 3517–3529, 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, Conference Track Proceedings*, 2014.

Tax, D. M. J. and Duin, R. P. W. Growing a multi-class classifier with a reject option. *Pattern Recognition Letters*, 29(10):1565–1570, 2008.

Tramèr, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*, 2020.

Wilson, D. J. The harmonic mean p-value for combining dependent tests. *Proceedings of the National Academy of Sciences*, 116(4):1195–1200, 2019.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2018.

Yang, P., Chen, J., Hsieh, C., Wang, J., and Jordan, M. I. ML-LOO: Detecting adversarial examples with feature attribution. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 6639–6647. AAAI Press, 2020.

Yuan, X., He, P., Zhu, Q., and Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9): 2805–2824, 2019. doi: 10.1109/TNNLS.2018.2886017.

Zhao, M. and Saligrama, V. Anomaly detection with score functions based on nearest neighbor graphs. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems*, pp. 2250–2258, 2009.

Zheng, Z. and Hong, P. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pp. 7924–7933, 2018.