A General Framework For Detecting Anomalous Inputs to DNN Classifiers

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




1 WISCONSIN UNIVERSITY OF WISCONSIN-MADISON

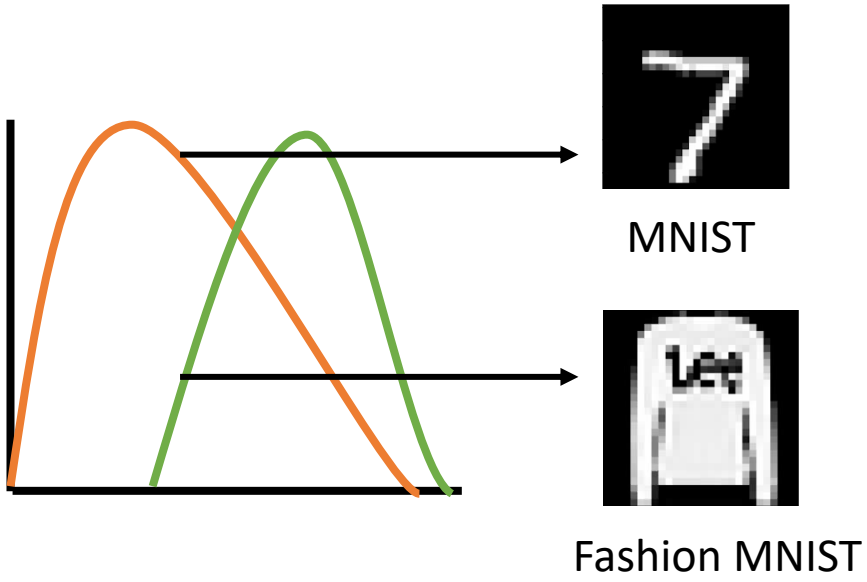
2 XAIPIENT

Motivation

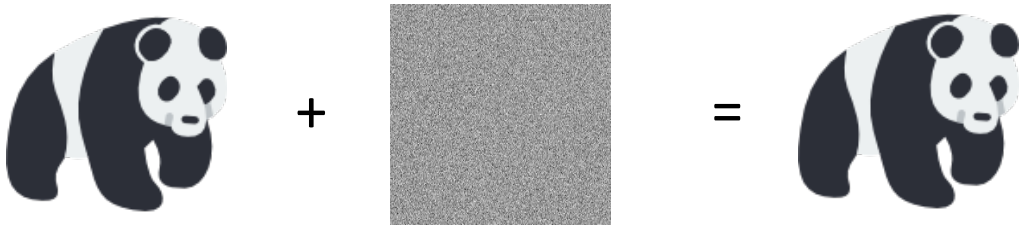
- **DNN classifiers** are widely used in many critical applications
- **Unreliable predictions** outside the training distribution, sometimes with **high confidence**
- **Anomalous inputs** are fairly common in practice
  - Novel classes, adversarial attacks
- Important to have a **defense mechanism** for DNN classifiers
  - E.g., detect anomalies and take corrective action

- We focus on test-time **detection of anomalous inputs** to a DNN classifier

Anomalous Inputs



Out-of-distribution



Adversarial attack

[Goodfellow et al., ICLR'15]

Prior Works

- **Supervised Methods:** require a broad sampling of **known anomalous** data
  - ❑ Used for configuring hyper-parameters. Does not generalize well to **unknown anomalies**
- **Specific Layers:** such methods do not jointly exploit the properties exhibited by anomalies across the DNN layers
- **Large dimensionality:** generative modeling-based methods are not well-suited
- Lack of a **general anomaly detection framework** where one can plug-in different components (e.g., test statistics)

Joint statistical Testing across Layers for Anomalies

- **JTLA** – a general framework for detecting anomalous inputs to a DNN

1. Utilizes **multiple layer representations** of a DNN for detection
2. Focuses on **class-conditional statistics** of layer representations to better isolate anomalies
3. **Unsupervised** - does not need any anomalous samples for training

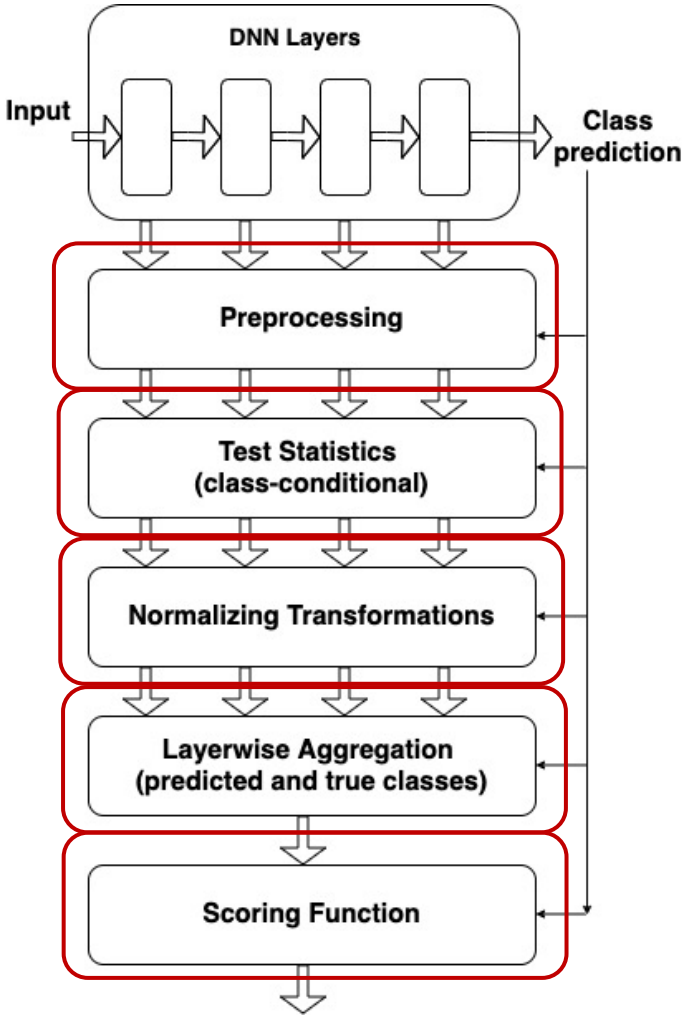
Contributions

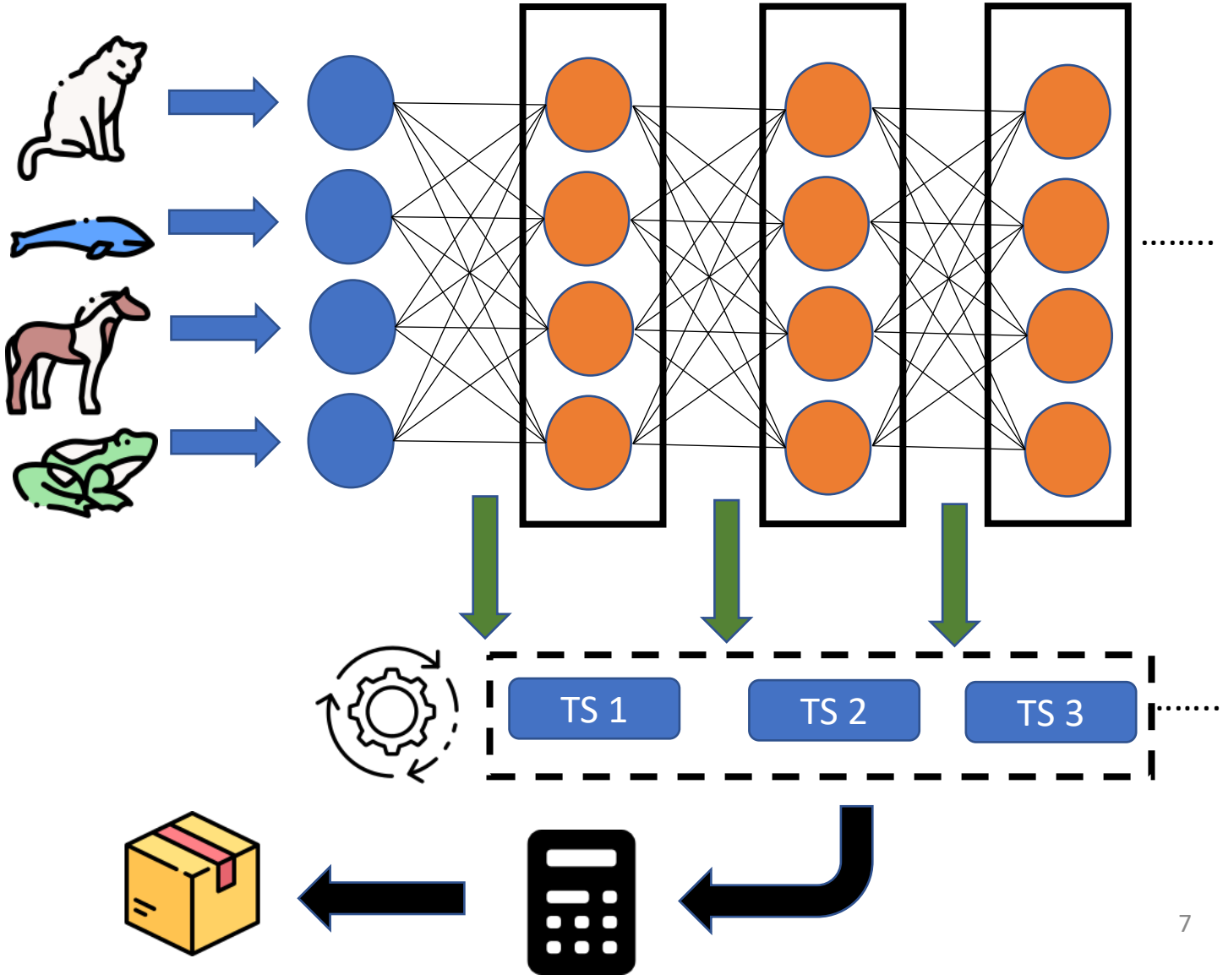
1. We present a general **meta-algorithm for anomaly detection** with formally defined components
   - Prior works on this problem can be fit into this meta-algorithm

2. We propose **novel methods for instantiating** components of the meta-algorithm

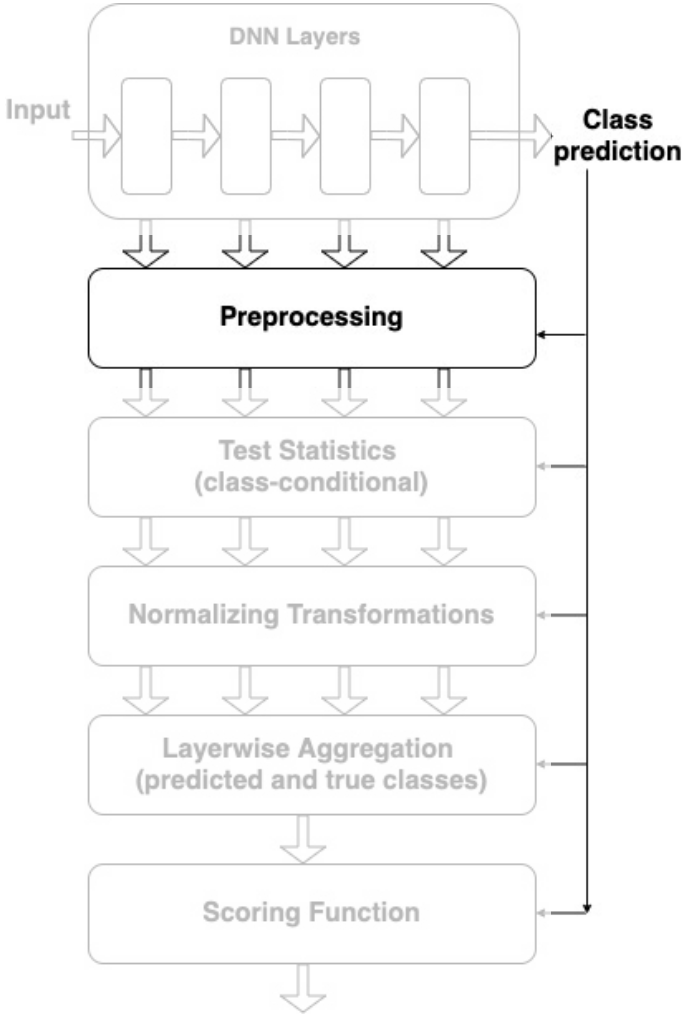
3. We formulate a **defense-aware adaptive attack** that focuses on detectors such as JTLA that use the DNN **layer representations**

Meta-algorithm For JTLA





Problem Setup & Preprocessing



- **Test input** : $\boldsymbol{x}$
  True class : ??
  DNN class prediction: $\hat{C}(\boldsymbol{x}) = \hat{c}$

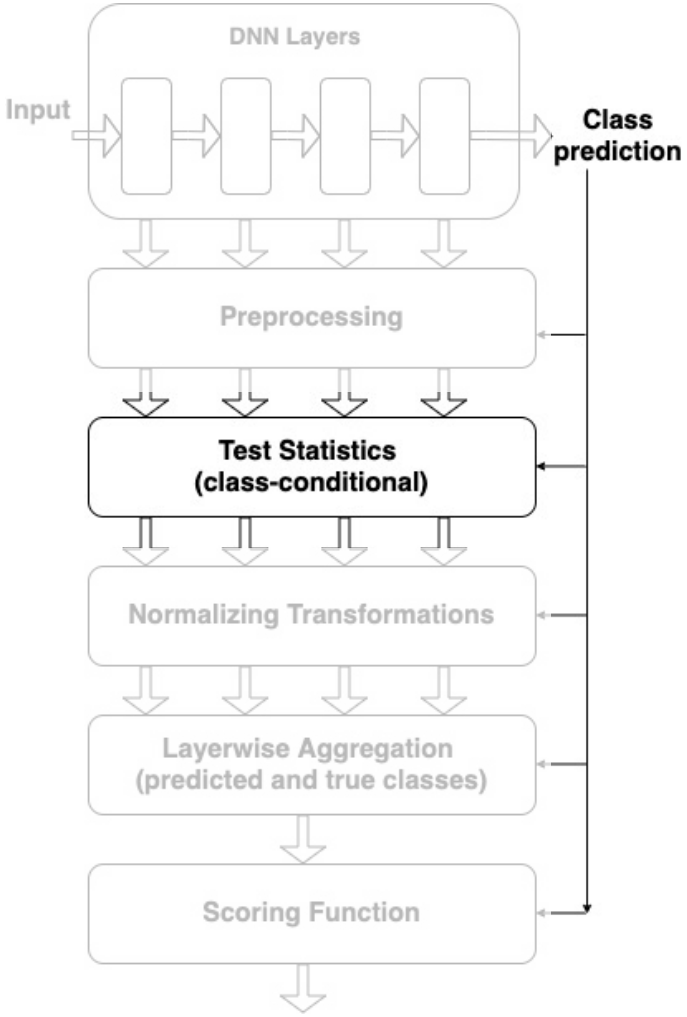
- Extract **layer representations** of $\boldsymbol{x}$:
  $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(L)}$

- **Labeled dataset** of natural inputs
  $\mathcal{D} = \{(\boldsymbol{x}_n, c_n),\ n = 1, \cdots, N\}$

- Data subsets for each **layer** ℓ and **true class** $\boldsymbol{c}$
  $\mathcal{D}_a(\ell, c) = \{(\boldsymbol{x}_n^{(\ell)}, c_n, \hat{c}_n),\ n = 1, \cdots, N : c_n = c\}$

- Data subsets for each **layer** ℓ and **predicted class** $\hat{\boldsymbol{c}}$
  $\widehat{\mathcal{D}}_a(\ell, \hat{c}) = \{(\boldsymbol{x}_n^{(\ell)}, c_n, \hat{c}_n),\ n = 1, \cdots, N : \hat{c}_n = \hat{c}\}$

Test Statistics from the Layers



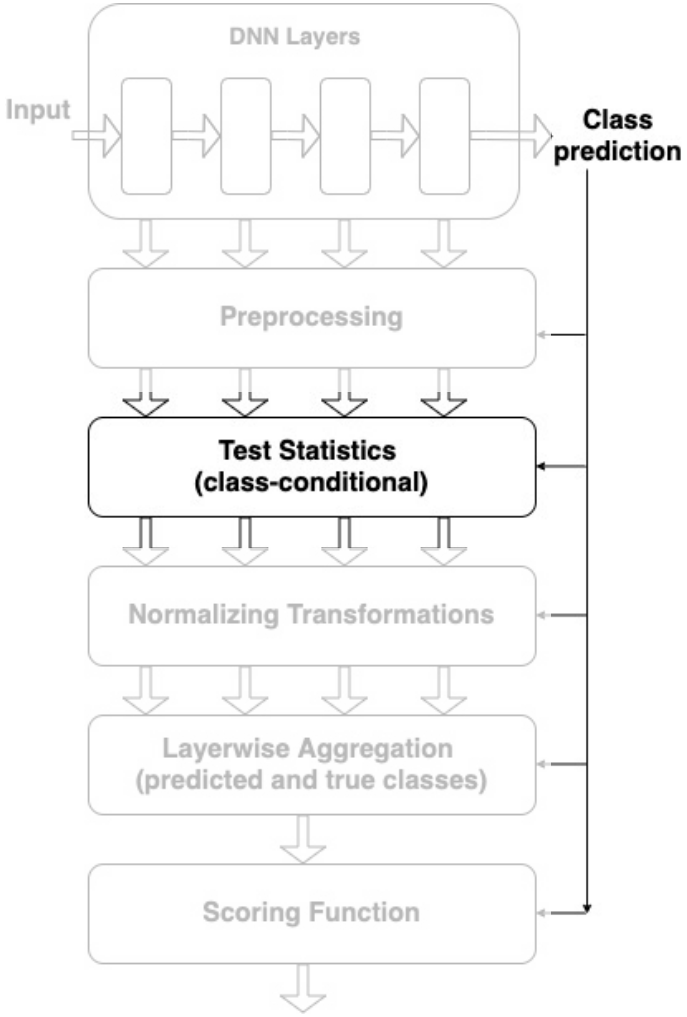
- A function of the layer representation that captures a **statistical property useful for detection**

- Test statistics proposed in **prior works**
  - ❑ Mahalanobis distance [Lee et al., NIPS'18]
  - ❑ Local intrinsic dimension [Ma et al., ICLR'18]
  - ❑ Gram matrix-based deviations [Sastry & Oore, ICML'20]

- Test statistics are defined to be **class-conditional**
- Useful for adversarial inputs that focus on specific (true class, predicted class) pairs

- **Requirement: larger test statistic** => **larger deviation** of the layer representation from natural inputs

Test Statistics from the Layers



TS conditioned on the **predicted class** $\hat{c}$

$$t_{p\,|\,\hat{c}}^{(\ell)} = T(\boldsymbol{x}^{(\ell)}, \hat{c}, \widehat{\mathcal{D}}_a(\ell, \hat{c}))$$

Layer	0	1	.	.	L
TS	$t_{p\,|\,\hat{c}}^{(0)}$	$t_{p\,|\,\hat{c}}^{(1)}$	.	.	$t_{p\,|\,\hat{c}}^{(L)}$

$$\boldsymbol{t}_{p\,|\,\hat{c}}$$

TS conditioned on each **candidate source class**

$$t_{s\,|\,c}^{(\ell)} = T\left(\boldsymbol{x}^{(\ell)}, c, \mathcal{D}_a(\ell, c)\right), \quad c = 1, \cdots, m$$

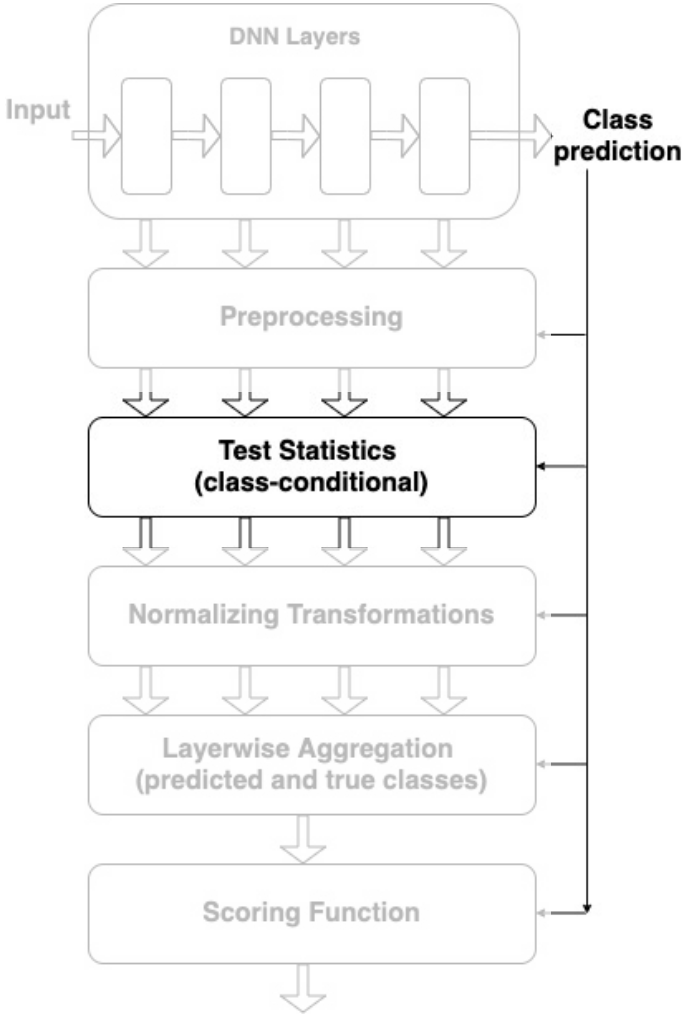
Layer	0	1	.	.	L
TS	$t_{s\,|\,c}^{(0)}$	$t_{s\,|\,c}^{(1)}$	.	.	$t_{s\,|\,c}^{(L)}$

$$\boldsymbol{t}_{s\,|\,c}$$

$m+1$ TS vectors from the layers:

$$\boldsymbol{t}_{p\,|\,\hat{c}}, \boldsymbol{t}_{s\,|\,1}, \cdots, \boldsymbol{t}_{s\,|\,m}$$

Test Statistic - Instantiation



- We propose a TS based on the **class counts of k-nearest neighbors** (KNN) of a layer representation
- We learn a **multinomial** model for the joint distribution of the **class counts from natural inputs**
- We apply the **log-likelihood ratio TS** of the well-known multinomial test [Read & Cressie, 2012]

- KNN class counts of an input at layer ℓ: $(k_1^{(\ell)}, \cdots, k_m^{(\ell)})$

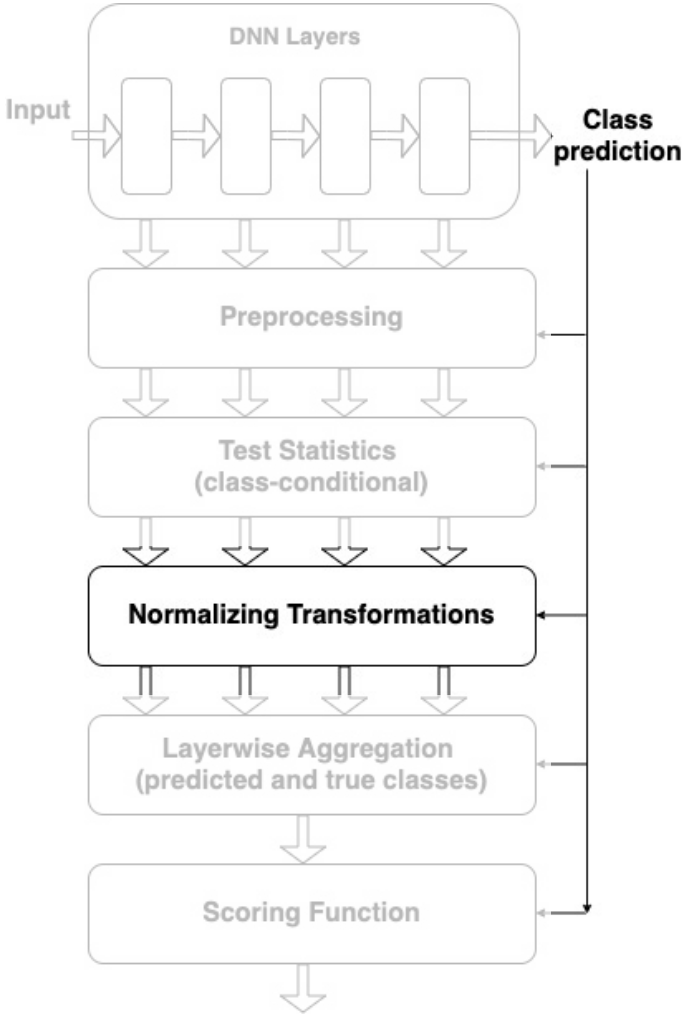
- TS conditioned on the **predicted class**:

$$T(\boldsymbol{x}^{(\ell)}, \hat{c}, \widehat{\mathcal{D}}_a(\ell, \hat{c})) = \sum_{i=1}^{m} k_i^{(\ell)} \log \frac{k_i^{(\ell)}}{k\, \pi_{i \mid \hat{c}}^{(\ell)}}$$

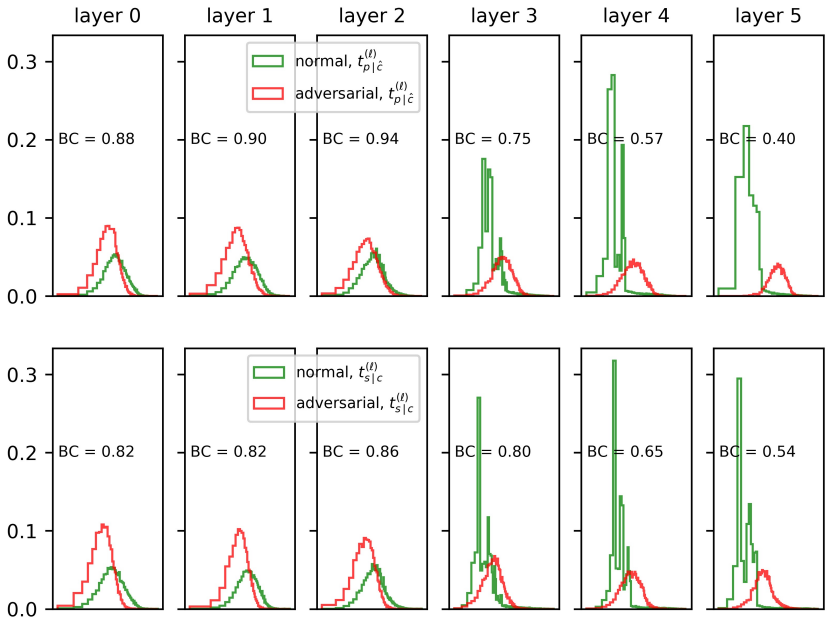
- TS conditioned on **each source class** $c = 1, \cdots, m$:

$$T\left(\boldsymbol{x}^{(\ell)}, c, \mathcal{D}_a(\ell, c)\right) = \sum_{i=1}^{m} k_i^{(\ell)} \log \frac{k_i^{(\ell)}}{k\, \tilde{\pi}_{i \mid c}^{(\ell)}}$$

Distribution-Independent Normalization

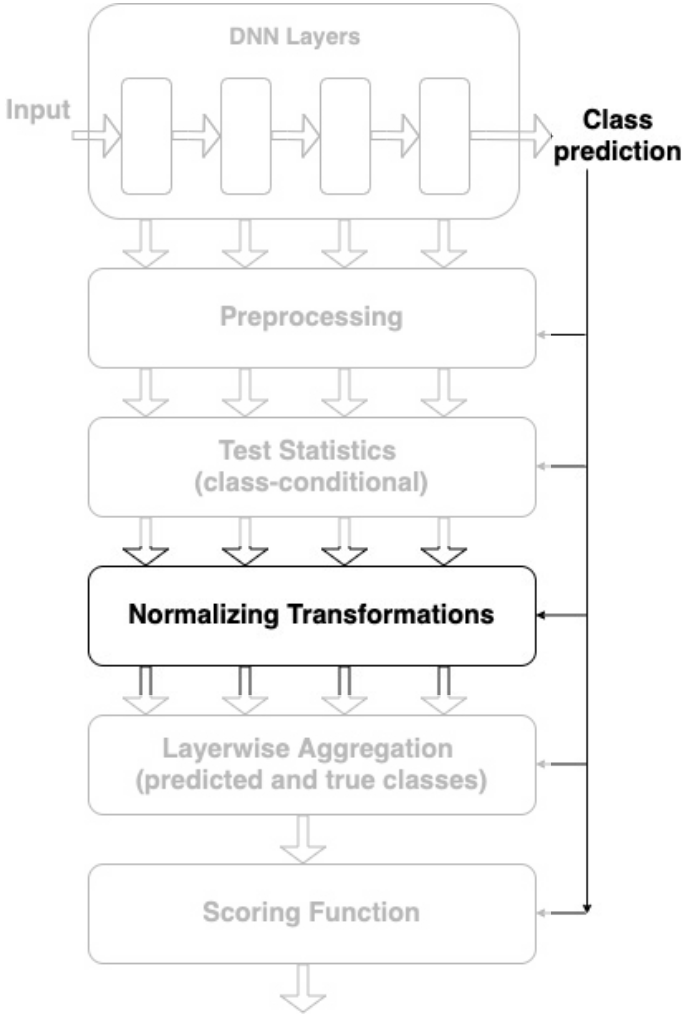


- Distribution of the test statistics are unknown and **change across the layers**



- **Goal:** transform or normalize the test statistics into a standard distribution

Distribution-Independent Normalization



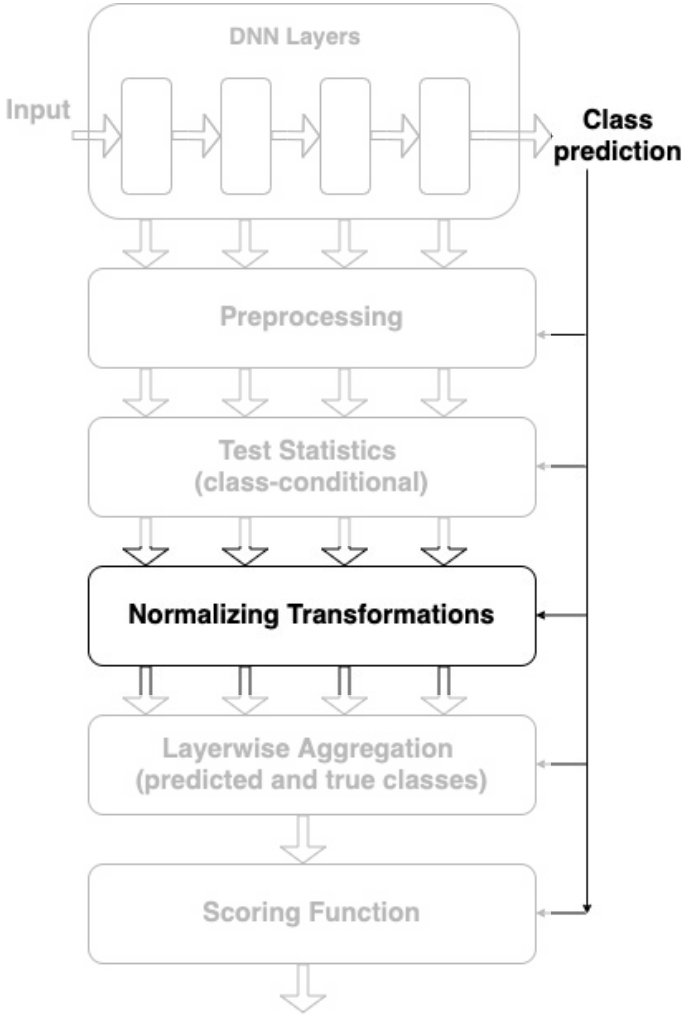
- Methods used in **prior works**
  - ❑ Z-score normalization [Roth et al., ICML'19]
  - ❑ Scaling by the expected value [Sastry & Oore, ICML'20]

- We propose two types of normalizing transformations

1. Normalization of test statistics **at individual layers**
   $q : \mathbb{R} \to \mathbb{Q} \subset \mathbb{R}$
   For each layer $\ell = 0, 1, \cdots, L$:
   Predicted class: $\mathrm{q}(t^{(\ell)}_{p \mid \hat{c}})$
   Source classes: $\mathrm{q}(t^{(\ell)}_{s \mid 1}), \cdots, q(t^{(\ell)}_{s \mid m})$

2. Multivariate normalization of the **test statistic vectors**
   $q : \mathbb{R}^{L+1} \to \mathbb{Q} \subset \mathbb{R}$
   Predicted class: $q\left(\boldsymbol{t}_{p \mid \hat{c}}\right)$
   Source classes: $q\left(\boldsymbol{t}_{s \mid 1}\right), \cdots, q(\boldsymbol{t}_{s \mid m})$

Normalization methods - Instantiation



- We propose to use **p-values** for normalizing the TS
- P-value calculates the tail probability corresponding to an observed TS. How **extreme is the observed** value?

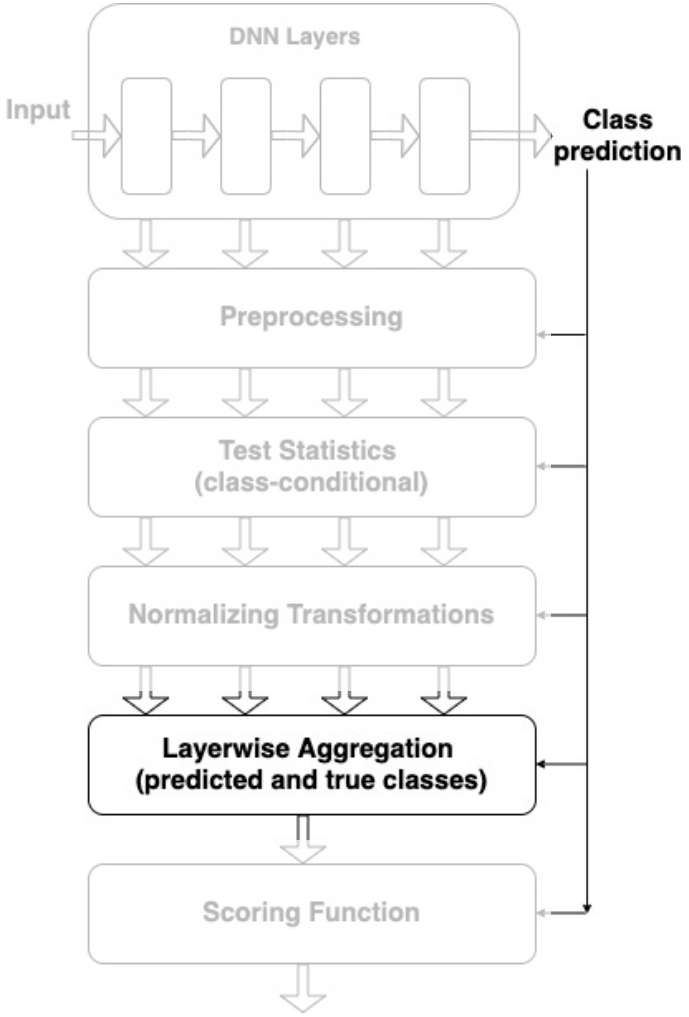
- p-values are **always uniformly distributed** on $[0, 1]$ under the null hypothesis (natural inputs)

- p-value transformation for the **predicted class**:
$q(t^{(\ell)}_{p \mid \hat{c}}) = \mathbb{P}(T^{(\ell)}_{p \mid \hat{c}} \geq t^{(\ell)}_{p \mid \hat{c}} \mid \hat{C} = \hat{c})$

- p-value transformation for the **candidate true classes**:
$q(t^{(\ell)}_{s \mid c}) = \mathbb{P}(T^{(\ell)}_{s \mid c} \geq t^{(\ell)}_{s \mid c} \mid C = c), \quad c = 1, \cdots, m$

- p-values are **estimated using the empirical CDF** based on the corresponding data subsets

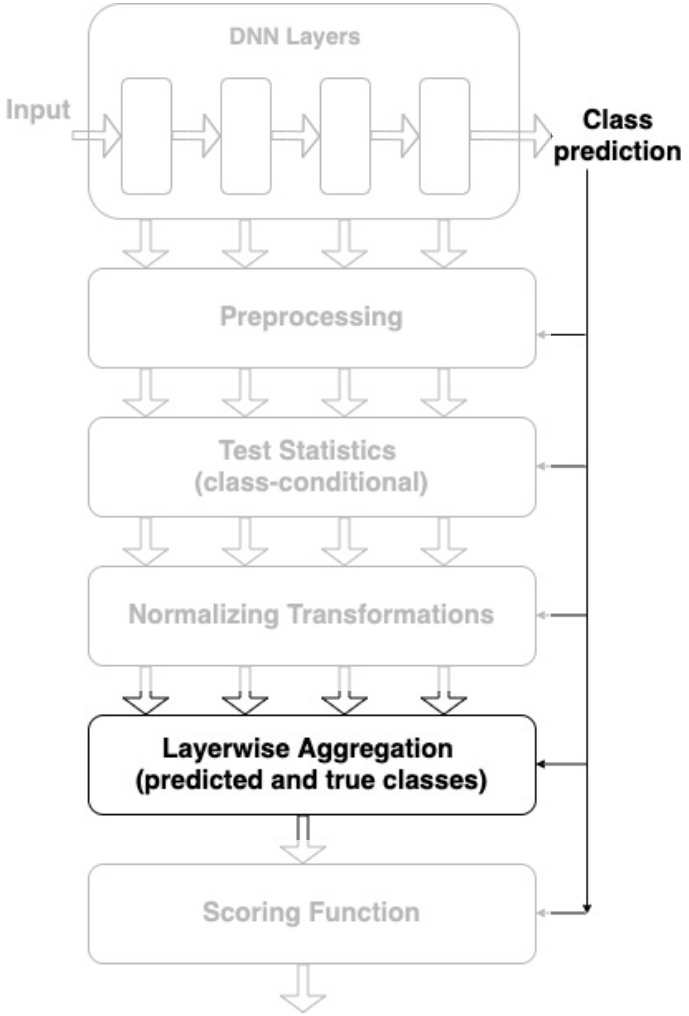
Layerwise Aggregation



- Normalized test statistics can be interpreted as local **class-conditional anomaly scores**
- An **aggregation function** combines the normalized test statistics (evidence of anomalous behavior) from the layers

- Methods used by **prior works**
  - ❑ **Weighted sum** with weights learned by a **Logistic classifier** [Lee et al., NIPS'18], [Ma et al., ICLR'18], [Yang et al., ICML'20]
  - ❑ **Maximum** or **average** of the normalized test statistics [Miller et al., NeCo'19], [Sastry & Oore, ICML'20]

Layerwise Aggregation



For the **predicted class,** $\hat{c}$

Layers	0	1	.	.	L
TS	$t_{p \mid \hat{c}}^{(0)}$	$t_{p \mid \hat{c}}^{(1)}$	.	.	$t_{p \mid \hat{c}}^{(L)}$
Normalized TS	$q(t_{p \mid \hat{c}}^{(0)})$	$q(t_{p \mid \hat{c}}^{(1)})$	.	.	$q(t_{p \mid \hat{c}}^{(L)})$

Aggregation function, $r(\cdot)$

$q_{\text{agg}}(\boldsymbol{t}_{p \mid \hat{c}})$

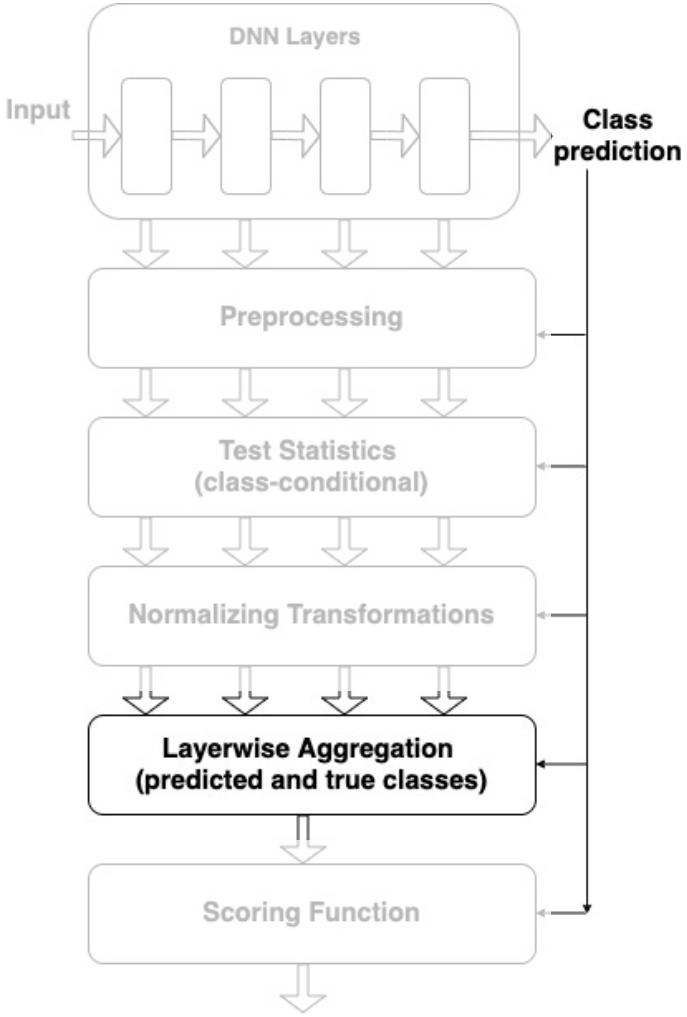
For each **candidate true class,** $c = 1, \cdots, m$

Layers	0	1	.	.	L
TS	$t_{s \mid c}^{(0)}$	$t_{s \mid c}^{(1)}$	.	.	$t_{s \mid c}^{(L)}$
Normalized TS	$q(t_{s \mid c}^{(0)})$	$q(t_{s \mid c}^{(1)})$	.	.	$q(t_{s \mid c}^{(L)})$

Aggregation function, $r(\cdot)$

$q_{\text{agg}}(\boldsymbol{t}_{s \mid c})$

Layerwise Aggregation - Instantiation



- Recall that the normalized test statistics are p-values
- **Smaller p-value** => **Larger deviation** from the natural distribution

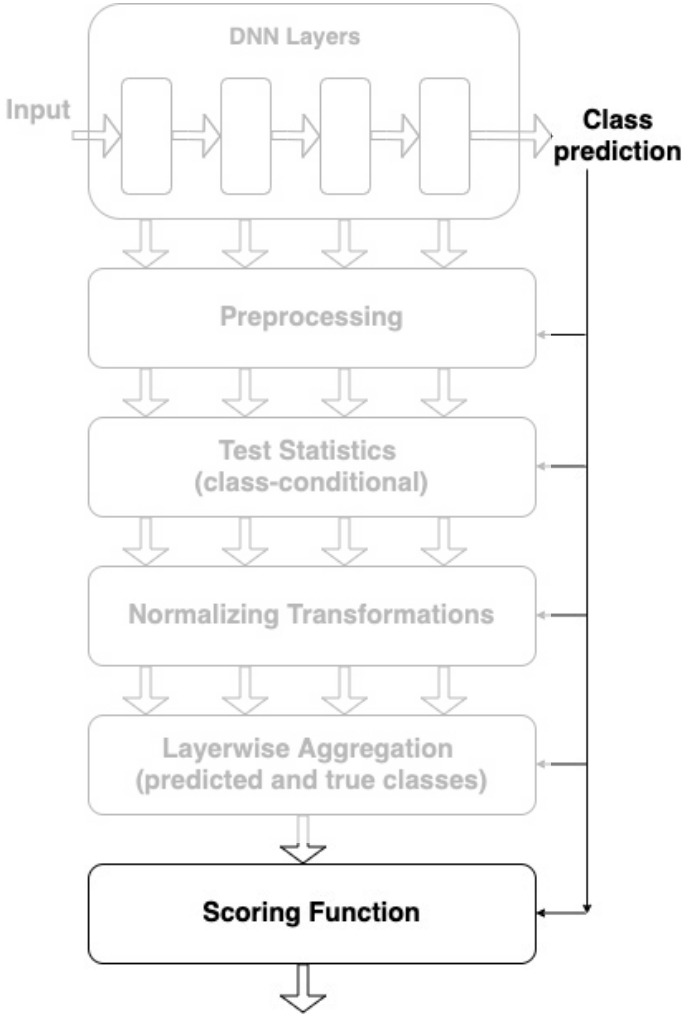
- Consider each p-value from a layer or layer pair to correspond to a **local hypothesis test**
- We apply well-known methods from **multiple testing** for **combining p-values**

- Fisher's method-based aggregation [Fisher, 1992]:
  $\log q_{\text{agg}}(\boldsymbol{t}) = \log r(Q) = \sum_{q \in Q} \log q$

- Harmonic mean p-value-based aggregation [Wilson, 2019]:
  $q_{\text{agg}}(\boldsymbol{t})^{-1} = r(Q)^{-1} = \sum_{q \in Q} q^{-1}$

- We **focus on Fisher's method** in the results, since it has better performance of the two

Proposed Scoring - Adversarial



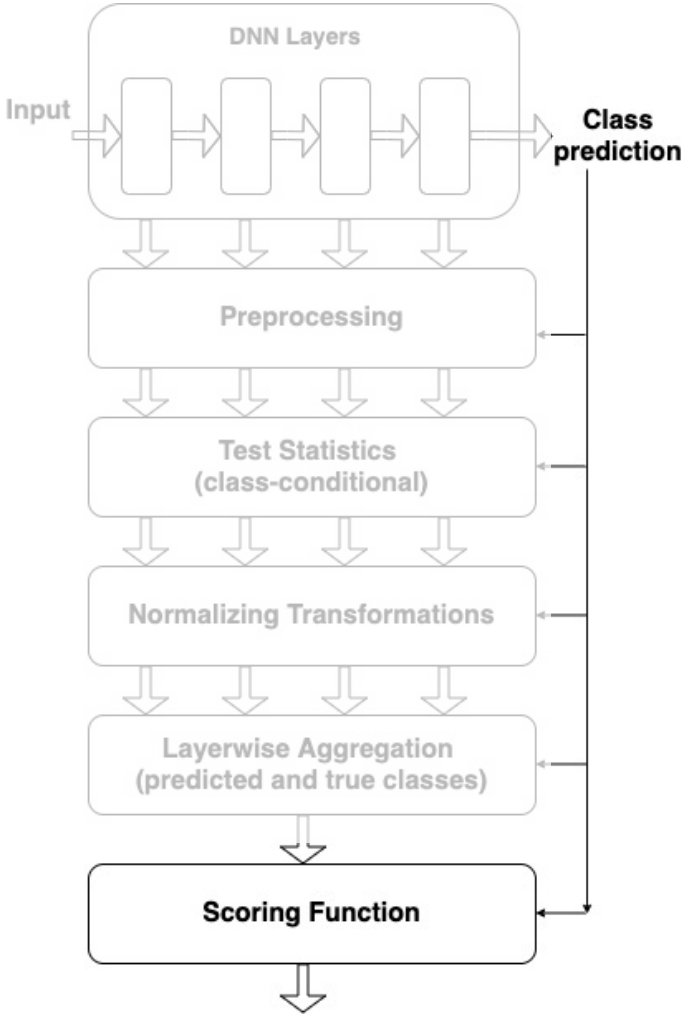
- **Key insight:** an adversarial input is expected to be
  - **Anomalous** at one or more of its layer representations **w.r.t to the predicted class**, $\hat{c}$
  - **Typical** at one or more of its layer representations **w.r.t the source class** from which it was created, $c \neq \hat{c}$

- For adversarial detection

$$S\left(q_{\text{agg}}\left(\boldsymbol{t}_{p \mid \hat{c}}\right), q_{\text{agg}}\left(\boldsymbol{t}_{s \mid 1}\right), \cdots, q_{\text{agg}}\left(\boldsymbol{t}_{s \mid m}\right), \hat{c}\right)$$

$$= \log\left[\frac{\max_{c \in [m] \setminus \{\hat{c}\}} q_{\text{agg}}\left(\boldsymbol{t}_{s \mid c}\right)}{q_{\text{agg}}\left(\boldsymbol{t}_{p \mid \hat{c}}\right)}\right]$$

Proposed Scoring - OOD



- An **OOD input** is different from an adversarial input in that it is **not created from one of the true classes**
- It is anomalous w.r.t the predicted class, $\hat{c}$

- We use a simpler version of the adversarial score function

$$S\left(q_{\text{agg}}\left(\boldsymbol{t}_{p\mid\hat{c}}\right), q_{\text{agg}}\left(\boldsymbol{t}_{s\mid 1}\right), \cdots, q_{\text{agg}}\left(\boldsymbol{t}_{s\mid m}\right), \hat{c}\right)$$
$$= \log\left[\frac{1}{q_{\text{agg}}\left(\boldsymbol{t}_{p\mid\hat{c}}\right)}\right]$$

Experimental Setup

Dataset	Input Size	#Samples (train + test)	Test Accuracy (%)	DNN Architecture
MNIST	28 x 28 x 1	50,000 + 10,000	99.12	2 Conv. + 2 FC layers
Not-MNIST	28 x 28 x 1	500,000 + 18,724	N/A	N/A
SVHN	32 x 32 x 3	73,257 + 26,032	89.42	2 Conv. + 3 FC layers
CIFAR-10	32 x 32 x 3	50,000 + 10,000	95.45	ResNet-34
CIFAR-100	32 x 32 x 3	50,000 + 10,000	N/A	N/A

- **Train split:** training the DNN classifier
- **Test split:** 5-fold cross-validation stratified by class
  - **Train fold**: training the detector
  - **Test fold**: anomaly scoring and performance metrics
  - Average metrics over the test folds are reported

Adversarial Attacks & OOD datasets

Adversarial Attack	Paper	Norm	Attack Parameter
Projected Gradient Descent (PGD)	Madry et al., ICLR'18	2	Norm-ball radius $\epsilon \in \{\frac{1}{255}, \frac{3}{255}, \cdots, \frac{21}{255}\}$
Carlini-Wagner (CW)	Carlini & Wagner, S&P'17	∞	Confidence $c \in \{0, 6, 14, 22\}$
Fast Gradient Sign Method (FGSM)	Goodfellow et al., ICLR'15	∞	Maximum norm-ball radius $\epsilon_{max} = 1$
Adaptive (defense-aware) Attack	Proposed	2	Attack strength λ

Inlier dataset	Outlier dataset
MNIST	Not-MNIST
CIFAR-10	SVHN
CIFAR-10	CIFAR-100

Detection Methods Compared

Paper	Method Name	Comments	Supervised
Lee et al., NIPS'18	Deep Mahalanobis		Yes
Ma et al., ICLR'18	LID	Local intrinsic dimensionality	Yes
Roth et al., ICML'19	Odds are odd		No
Papernot & McDaniel, 2018	Deep KNN		No
Jiang et al., NIPS'18	Trust Score	Pre-logit layer	No
Proposed	JTLA, Fisher, multi	p-values from layers & layer pairs combined using Fisher's method	No
Proposed	JTLA, LPE, multi	Multivariate p-value normalization using the K-LPE method	No

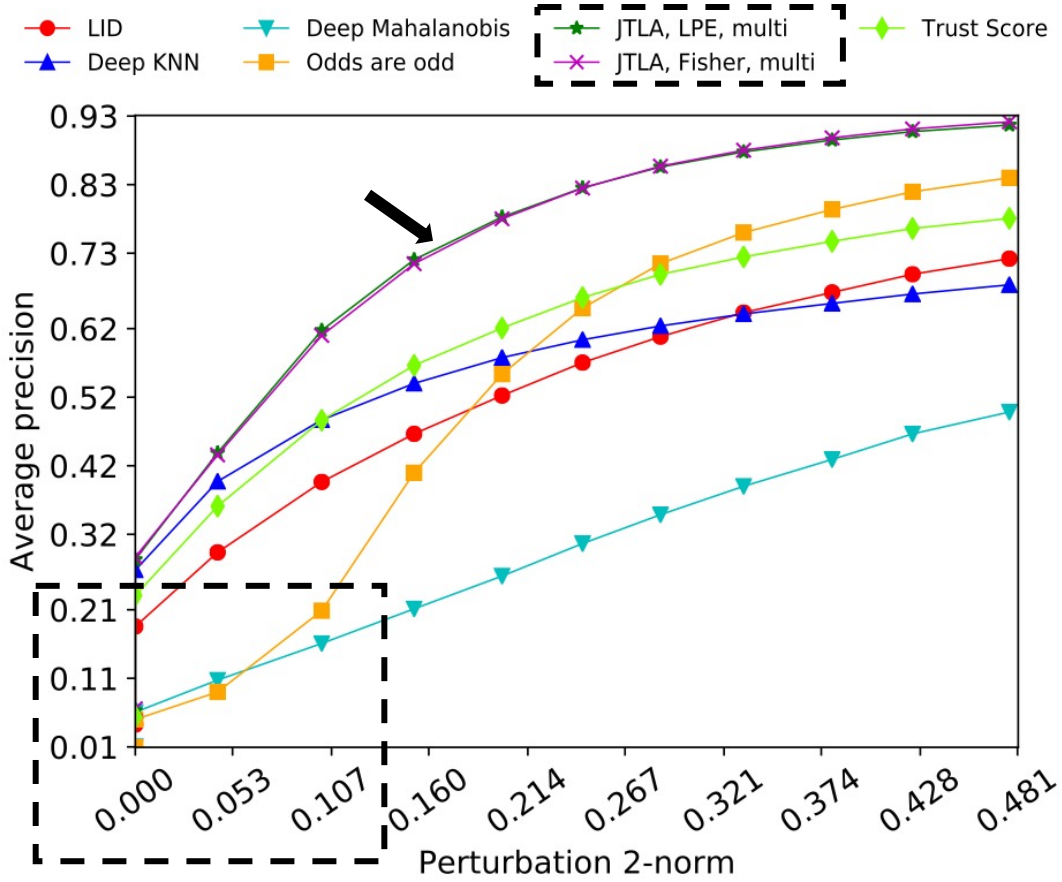
- Number of nearest neighbors k is set based on the training data size n using $k = \text{ceil}(n^{2/5})$
- Heuristic choice based on [Zhao et al., NIPS'09]

Performance Metrics

- Precision-Recall (PR) curves
  - **Average precision**: calculates approximate area under the PR curve
- ROC curve
  - **pAUC-α** : partial area under ROC curve below FPR α
- Both metrics are **threshold independent**

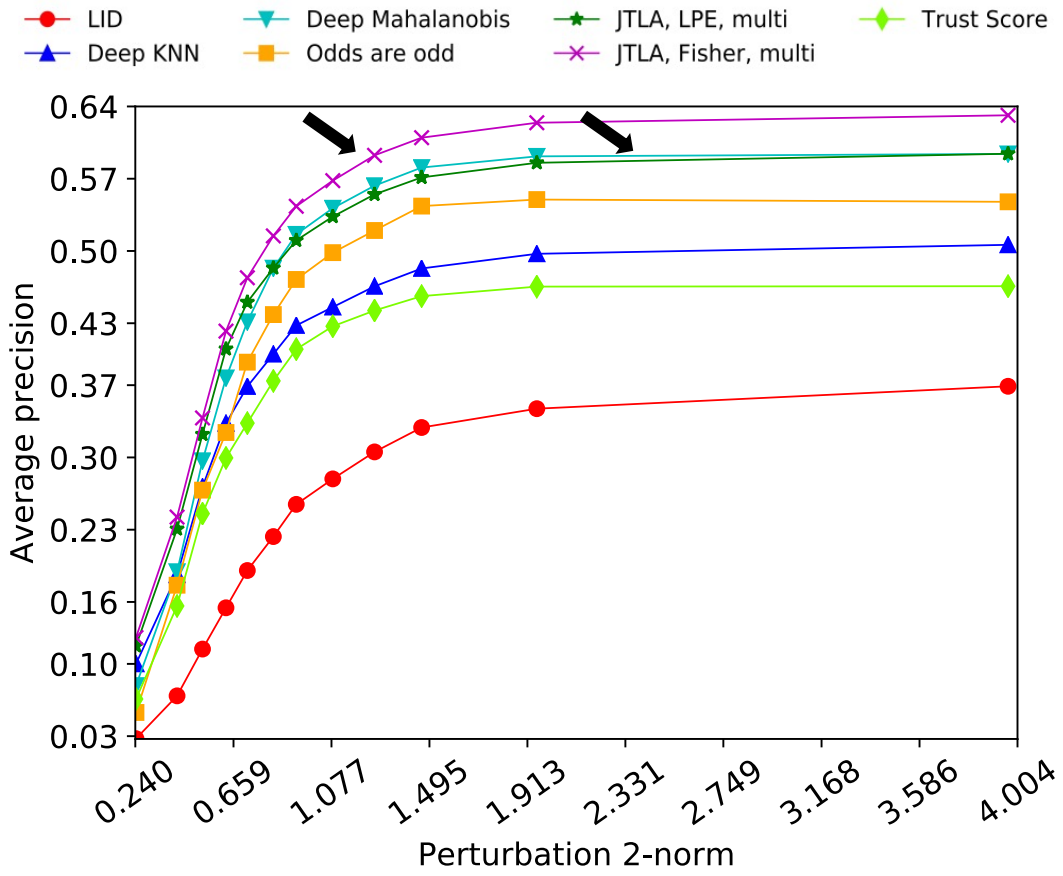
- **AUROC** is often used, but is **not a good metric** [Ahmed & Courville, AAAI'20]
  - Insensitive to the proportion of anomalies
  - Optimistic bias and insufficient contrast in values

Adversarial Detection: SVHN, CW attack



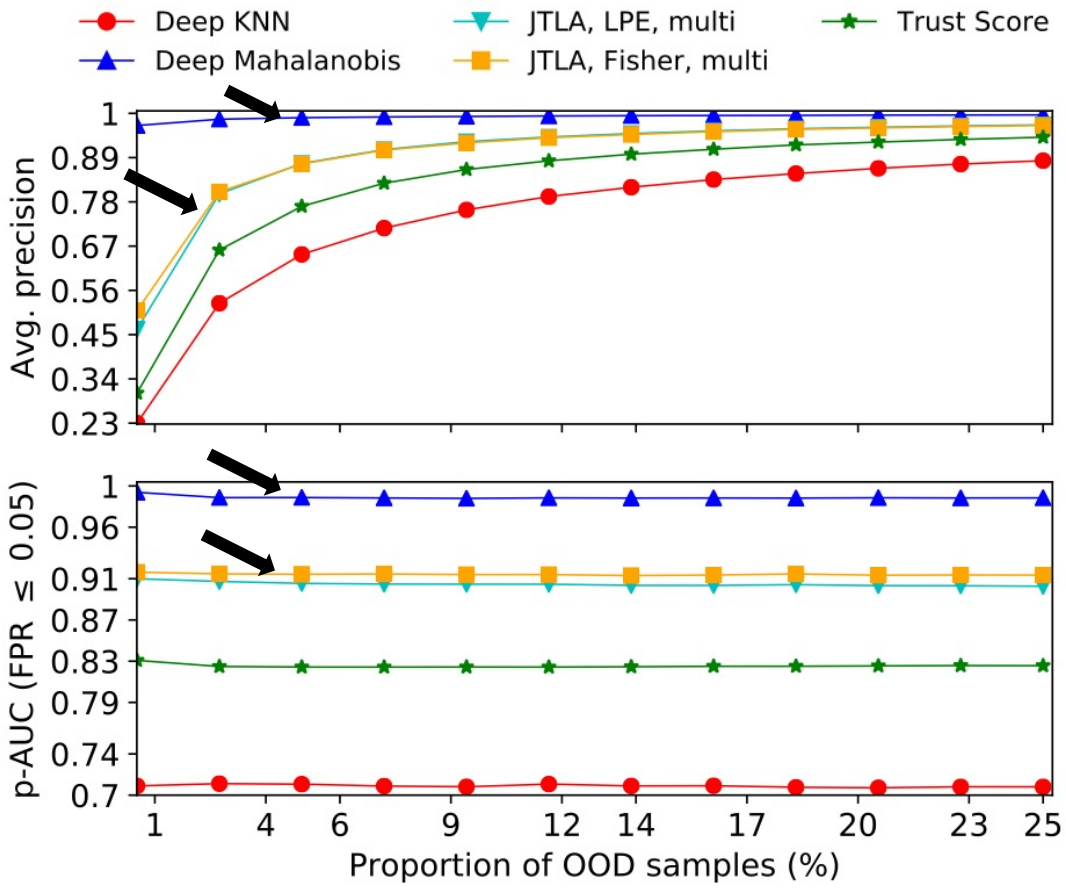
- **Both variants of JTLA** outperform other methods
- **Deep Mahalanobis** and **Odds are odd** (for low perturbation norm) have **bad performance**
- Comparison of **p-AUC** has **similar trends**

Adversarial Detection: CIFAR-10, Adaptive attack



- **JTLA, Fisher** has the best performance although the adaptive attack targets this particular method
- **Deep Mahalanobis** and **JTLA, LPE** also have similar good performance

OOD Detection: MNIST vs. Not-MNIST



- **Deep Mahalanobis** has the best performance
  - Deep Mahalanobis is the **only supervised method**.
  - It uses outlier samples from the training folds
- The two variants of JTLA outperform the other unsupervised methods

Takeaways from Experiments

Adversarial detection

- The **baseline methods** perform well on some datasets/attacks, but fail on others
- JTLA has more **consistent performance** across datasets and attacks

OOD detection

- **Deep Mahalanobis** has the best performance
  - ❑ Has the advantage of being **supervised**, using outlier samples for training
  - ❑ In real-world settings, it is **hard to collect sufficient number and variety** of outlier samples
- **JTLA** outperforms the other unsupervised baselines

Code Availability

- **Github repo**: https://github.com/jayaram-r/adversarial-detection
  - Implementation is modular. Easy to add new techniques

- Important libraries/packages used
  - PyTorch (deep learning and autograd)
  - Numpy, Scipy, Numba, and Scikit-Learn
  - Foolbox (adversarial attacks)
  - PyNNDescent (approximate nearest neighbors)

Thank you!
Questions?