ShallowJudge

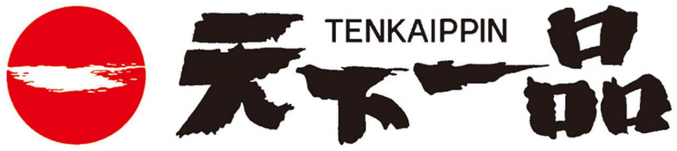
AI model debugging with reliability analysis via probabilistic method

Zhenglin Pan[1] *1781983*

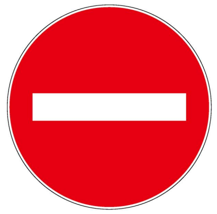
1: University of Alberta, AB, Canada

Motivation (30%)

- Mainly motivated by **AI model testing approach**.
- Traditional software testing vs. AI model testing
- AI model input can be implicit and tricky
- Regression test, Error backtracking
- Tricky sample are some time dangerous



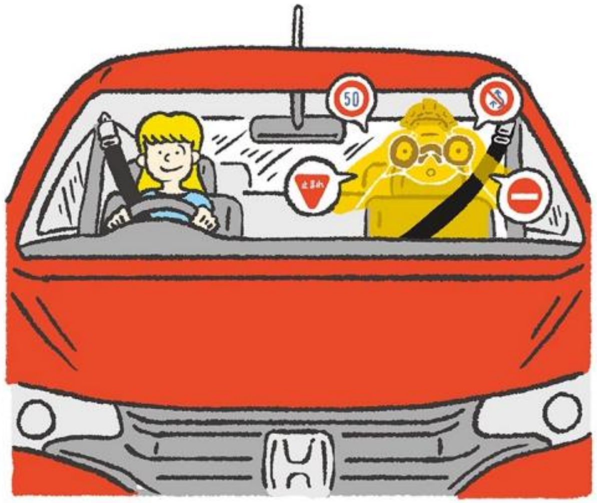
Ramen Logo

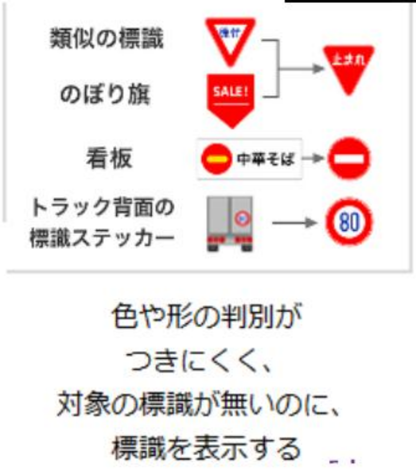


Stop sign

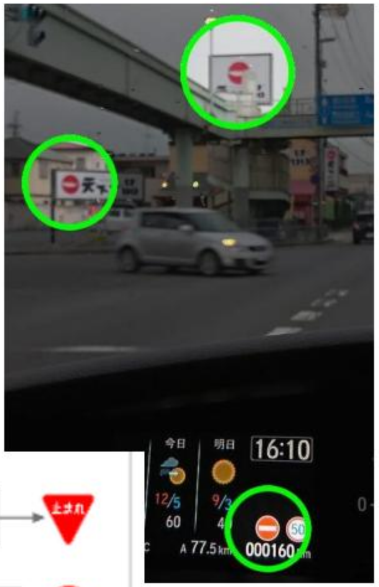
Real Cases in Japan

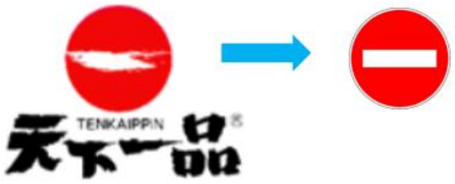
- When Honda sees ramen shop sign
  - First noticed in Dec 2017
  - Now still a caution on official website





色や形の判別が
つきにくく、
対象の標識が無いのに、
標識を表示する



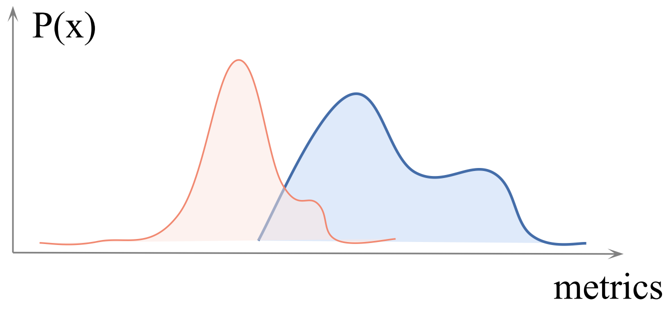


https://www.honda.co.jp/hondasensing/feature/srf/
https://soranews24.com/2018/09/17/major-japanese-ramen-chains-logo-confuses-honda-cars-ai

HONDA

Any method to find tricky cases?

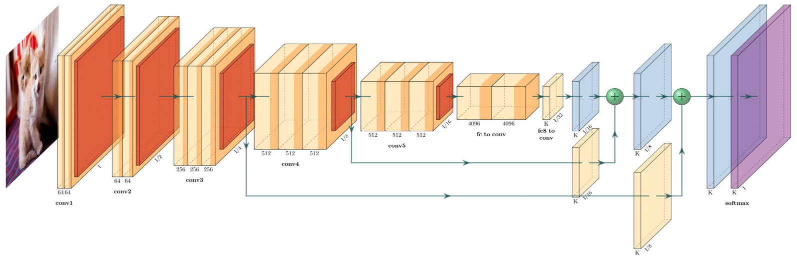
- Tricky sample diagnosis is needed
- **Assumption**: tricky samples and normal samples are subjected to different probabilistic distributions, which can be detected in inference.



for any given sample x, we have

$$P(Y=T \mid X=x) = \frac{P(X=x \mid Y=T)P(Y=T)}{\sum_{Y} P(X=x \mid Y=y)}$$

- Feature Maps are not only informative, but representative



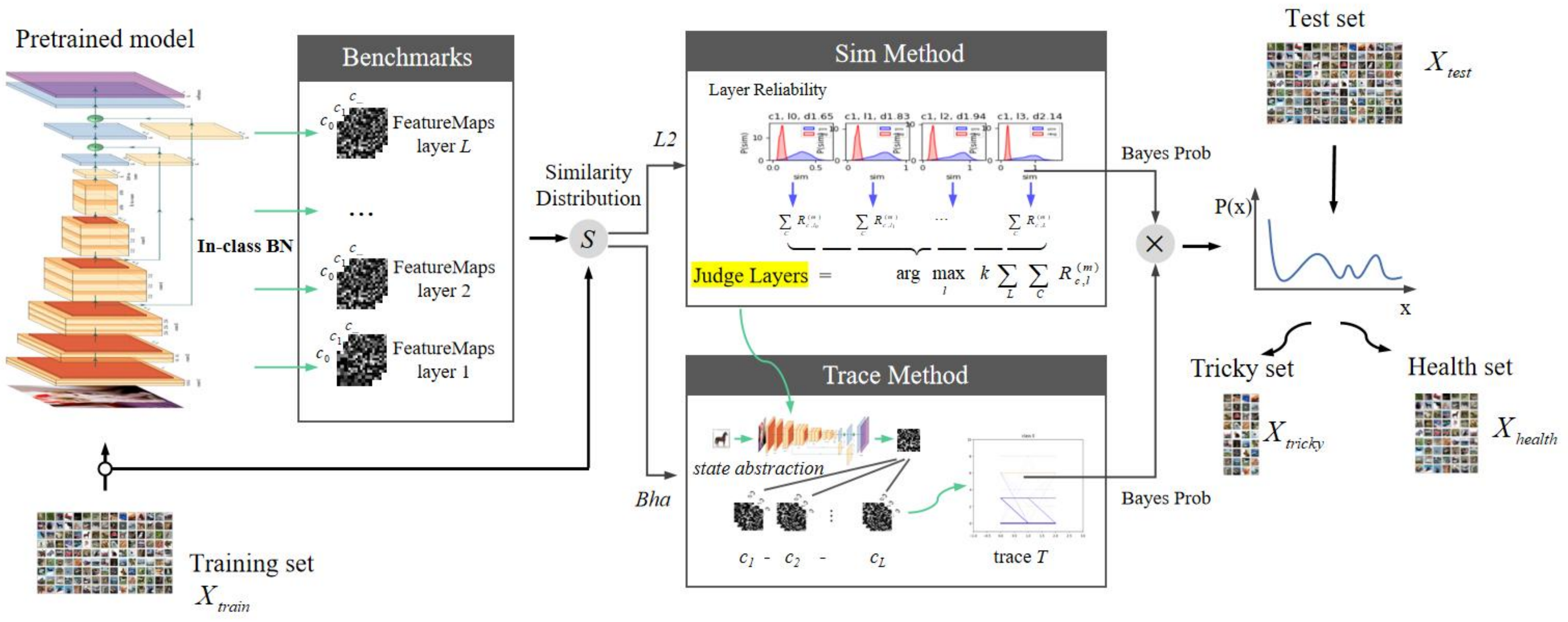
- Tricky samples and health ones subject to different distributions, these patterns are presented through feature maps.
- Is there an universal tricky sample detecting approach for all NN networks?

ShallowJudge

UNIVERSITY OF ALBERTA

Approach (30%)

- System Structure Graph



All you need is a pretrained model!

Benchmarks

$$b_{c,cl} = \sigma\left(\frac{1}{M} \sum_{m=0}^{M} \overset{pos}{\underset{trn}{F}}{}^{(m)}_{c,cl} \right)$$

$$B = \left\{ b_{c,cl} \mid c \in \{0,1,\ldots,C\}, cl \in \{0,1,\ldots,CL\} \right\}$$

C: class numbers

CL: candidate layers

$\sigma(\cdot)$: probability activation function

$\overset{pos}{\underset{tnr}{F}}$: feature map of positive(healthy) samples from training set

In-class batch nomalization is applied for getting benchmark for each class c on each candidate layer cl.

Candidate layer number CL is a hyper-parameter adjusting the depth of candidates from where judge layers will be nominated.

Sim Method

Once we gain the benchmarks on training set, we can calculate the probabilistic distributions on tricky sample and postive samples.

Given the benchmark $\boldsymbol{B}$ and input samples $\boldsymbol{x}$ from training set, we have

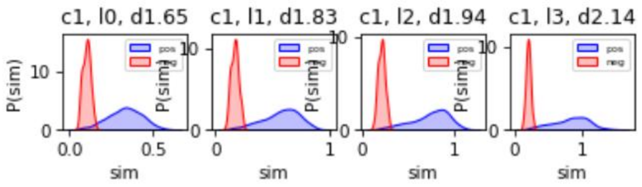
$$sim(\boldsymbol{x}, \boldsymbol{B}) = \left\{ \frac{1}{d(x_{c,l}, b_{c,l})} \mid c \in \{0,1,\ldots,C\}, l \in \{0,1,\ldots CL\} \right\}$$

where $d(\cdot)$ is a distance funtion, i.e. Euclidean distance.

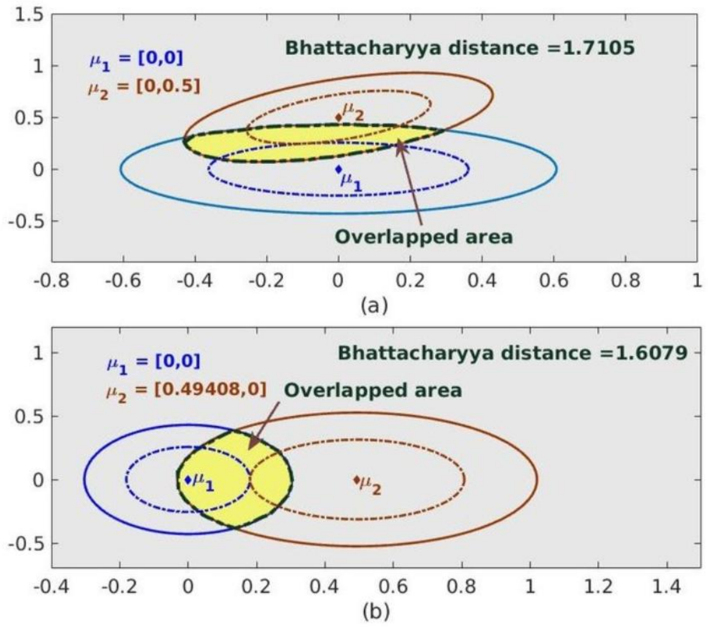
This is the similarity between one sample and its in-class benchmark.

By using Kernel Density Estimation(KDE) we can obtain the distributions.

Reliability Analysis



Bhattacharyya distance measures the similarity of two probability distributions



*

Judge Layers

$$\arg\max_{l} k \sum_{LC} \sum_{C} R_{c,l}^{(m)}$$

Within the candidate layers, the best k layers that effectively distinguish the tricky sample's distribution from the postive's.

* Bhatt, Dhaivat & Garg, Akash & Gopalakrishnan, Bharath & Krishna, Madhava. (2018). Chance Constraints Integrated MPC Navigation in Uncertainty amongst Dynamic Obstacles: An overlap of Gaussians approach.

Bayes Theory

for any input data from val or test set, we calcualte its similarity with in-class benchmark and use Bayes Theory

$$P(Y=T,c \mid X=sim)=\frac{P(X=sim \mid Y=T,c)P(Y=T,c)}{\sum_{Y} P(X=sim \mid Y=y,c)}$$

However, class information c is UNKOWN!

$\downarrow$

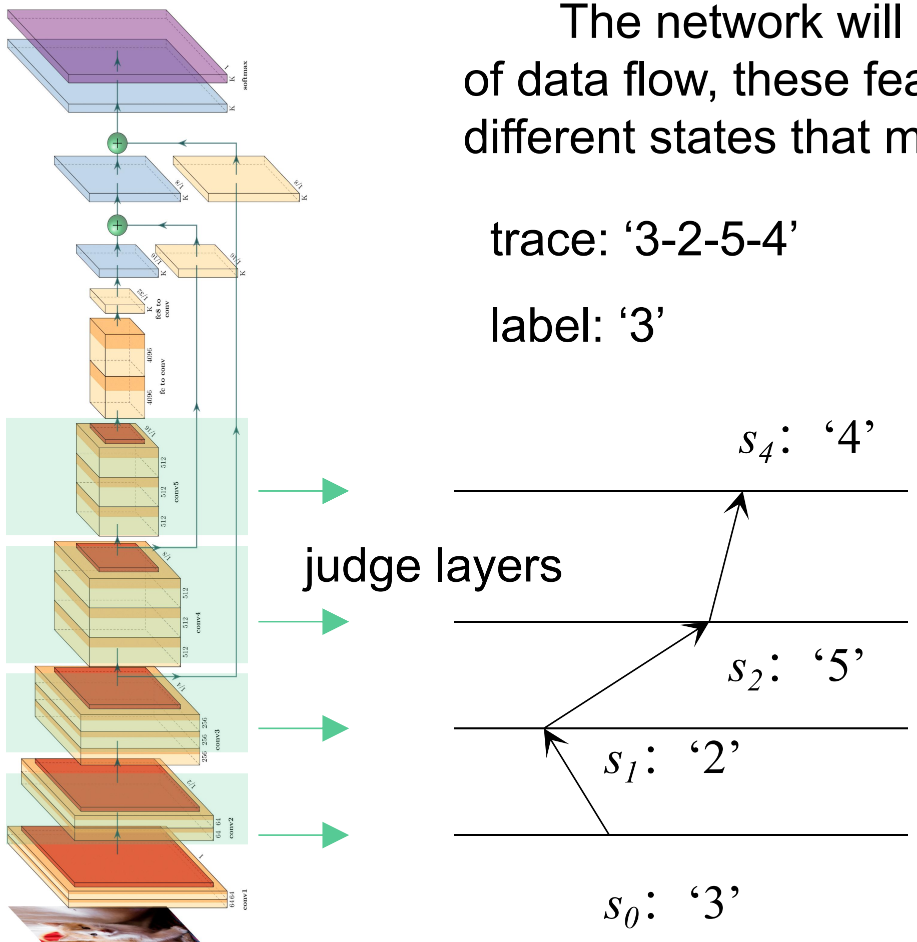
Trace Method

Trace Method

The network will yeild feature map on the direction of data flow, these feature maps can be abstracted to different states that map to benchmarks.

trace: ‘3-2-5-4’

label: ‘3’



$$P(C=c\,|\,T=t)=\frac{P(T=t\,|\,C=c)P(C=c)}{\sum_{C'}P(T=t\,|\,C=c')}$$

$P(C=c)$: Priori
$\sum P(T=t|C=c')$: Law of total probability
$P(T=t|C=c)$: Conditional probability

Given a trace on judge layers, we predict its class for test sample.

Bayes Theory

for any input data from val or test set, we calcualte its similarity with in-class benchmark and use Bayes Theory

$$P(Y=T,c \mid X=sim)=\frac{P(X=sim \mid Y=T,c)P(Y=T,c)}{\sum_{Y} P(X=sim \mid Y=y,c)}$$

However, class information is ~~UNKOWN~~!

$\downarrow$

KNOWN

c predicted by Trace Method

$$c=\arg\max_{c}\left(\frac{P(T=t \mid C=c)P(C=c)}{\sum_{C'} P(T=t \mid C=c')}\right)$$

So for any input without label, we can predict it's likelihood P() to be a tricky sample. We predict these bugs cases in inference and thus improve the accuracy of the model.

Implementation (5%)

- This project is implemented independently by Zhenglin Pan.
- Coding Language: Python / Jupyter Notebook
- Github repo: https://github.com/ZhenglinPan/ShallowJudge

README.md

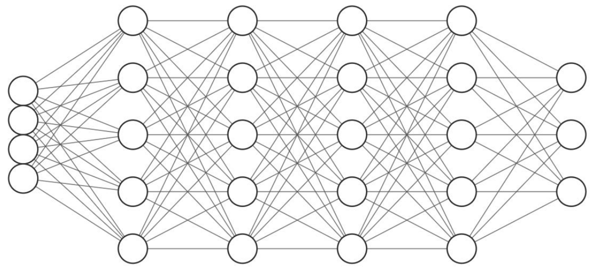
ShallowJudge

@Author: Zhenglin Pan @Date: March, 2023

This is a final course project for ECE 720 X50 Winter 2023, University of Alberta.

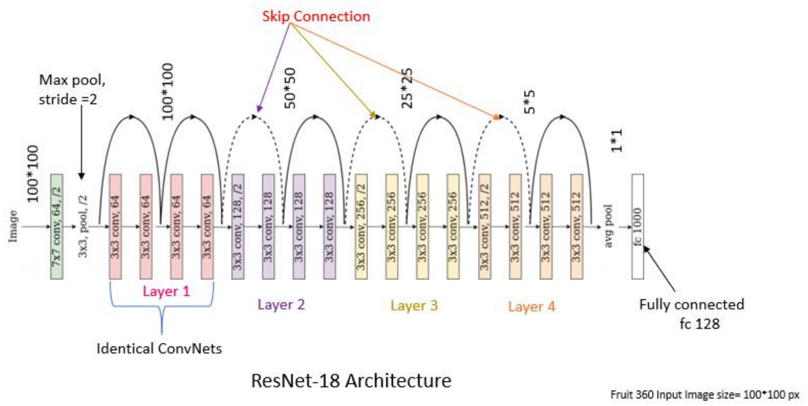
Evaluation (30%)

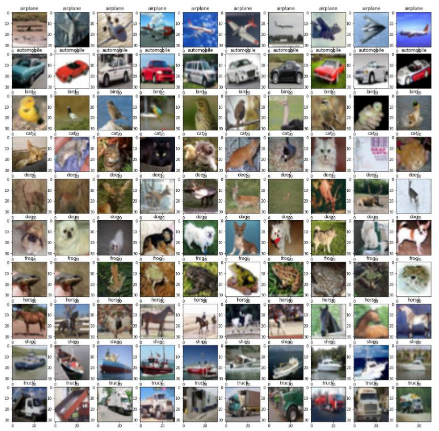
- Experiment **Group 1**:
  - 4-layer MLP + MNIST-fashion dataset



Candidates: 4
Judge layers: 3

- **Group 2**: ResNet18 + CIFAR10

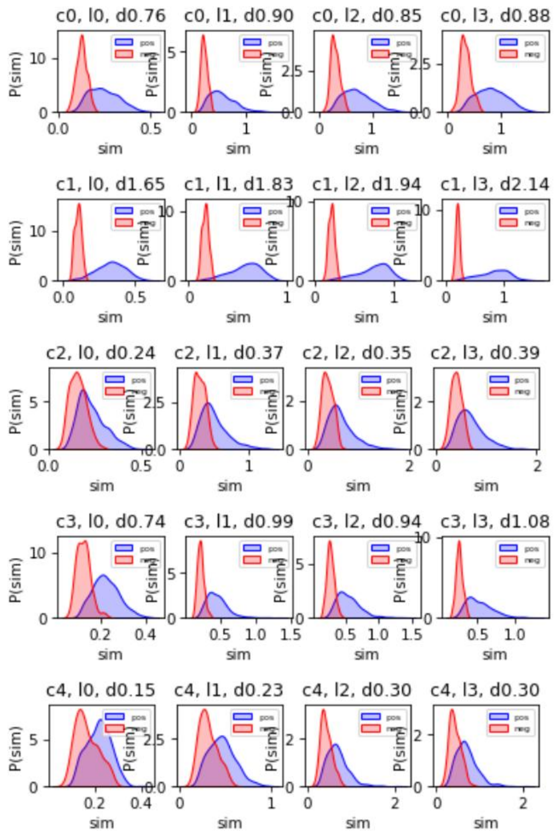




Candidates: 4
Judge layers: 3

Evaluation (30%)

- KDE distribution: stark contrasts on distribution



4layer MLP on MNIST-fashion dataset

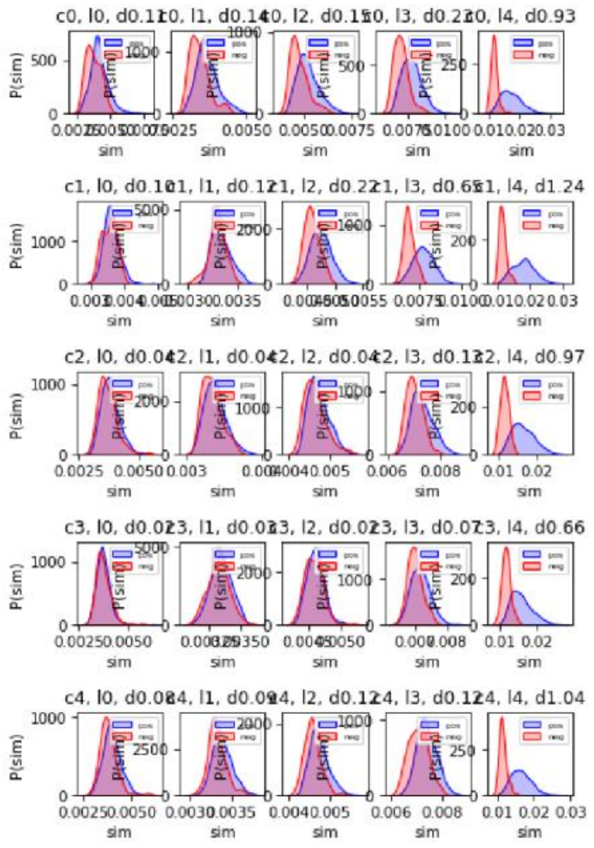
Through kernel density estimation plots, we observed a distinct difference between positive and negative samples on the distribution of similarity.

Ticky samples tend to have a smaller similarity with the class benchmark, proving our fundamental assumption is true.

Meanwhile, we noticed that the distributions are more contrastive on some layers, measured by the Bhattacharyya distance. In other words, some layers divide the positive and the negative well, and they are nominated as Judge Layers.

Evaluation (30%)

- KDE distribution: stark contrasts on distribution

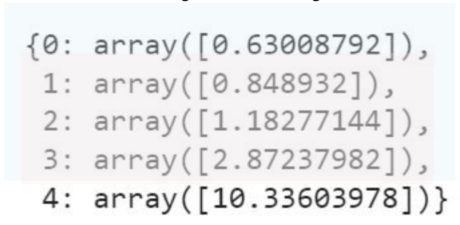


resnet18 on CIFAR10 dataset

On ResNet18 and CIFAR10:

- slower start, but an obvious progressive diversive effect.
- This is correspond to the ability of abstraction of the layers: the deep it is, the more represenative a feature map will be.
- if more candidates are allowed, more accurate results can be obtained.

Reliability Analysis:

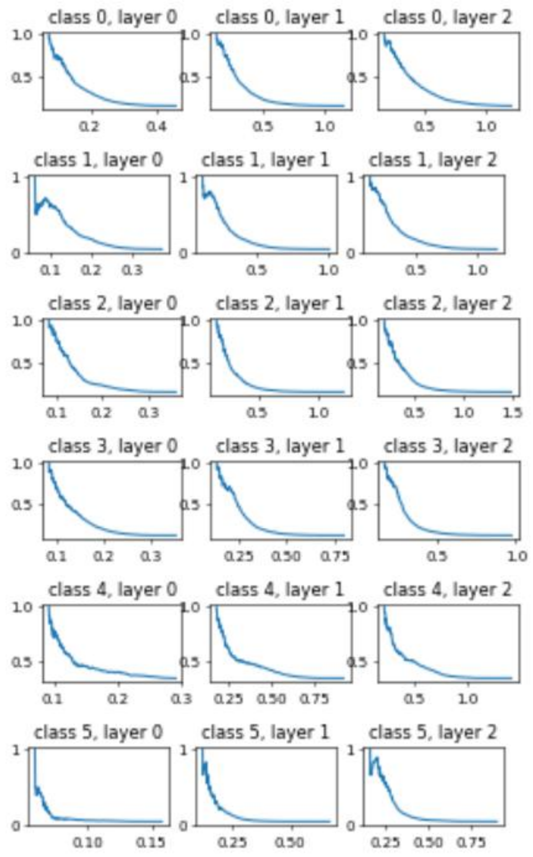


```
{0: array([0.63008792]),
 1: array([0.848932]),
 2: array([1.18277144]),
 3: array([2.87237982]),
 4: array([10.33603978])}
```

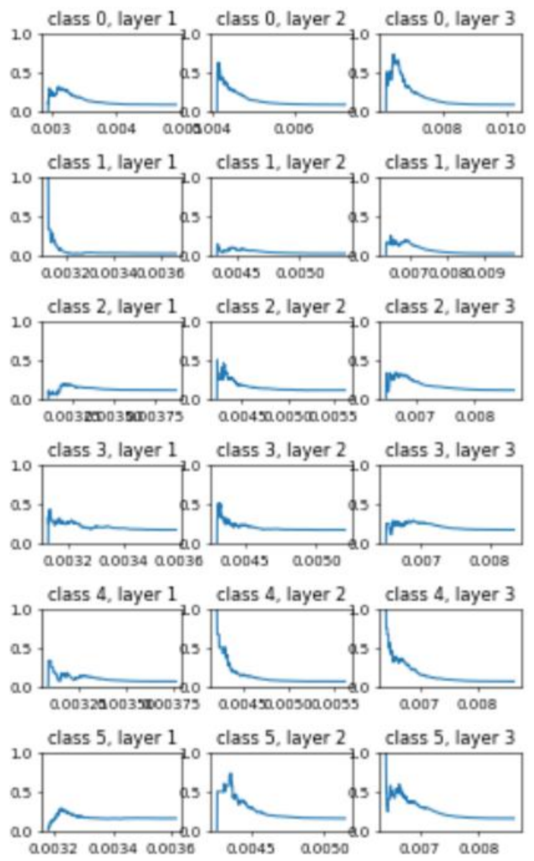
Candidates: 4
Judge layers: 3

Evaluation (30%)

- Similarity Beyas Distribution: On judge layers



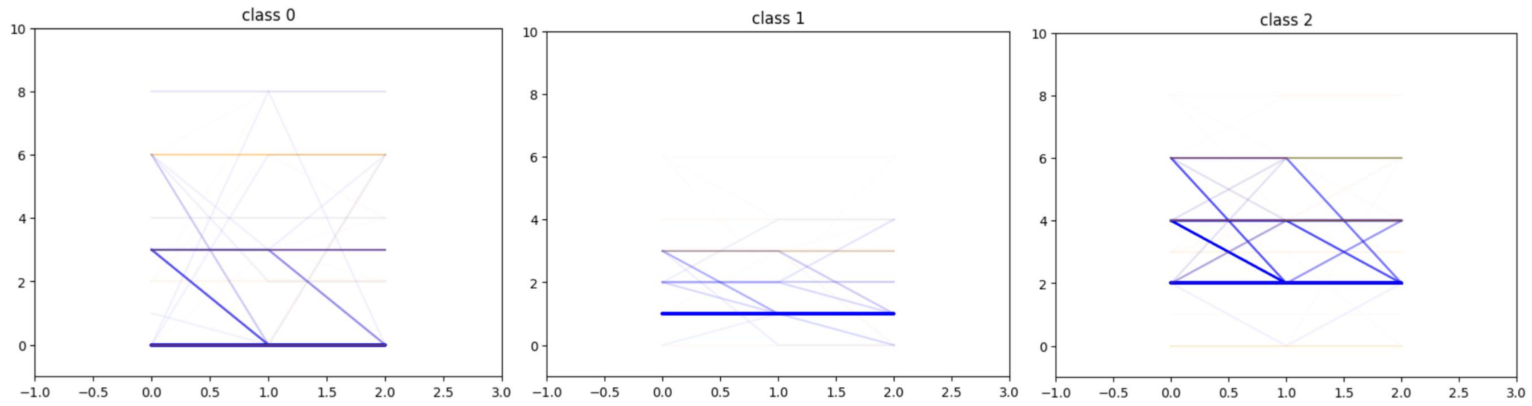
4layer MLP on MNIST-fashion dataset



resnet18 on CIFAR10 dataset

Evaluation (30%)

- Trace illustration on MNIST dataset

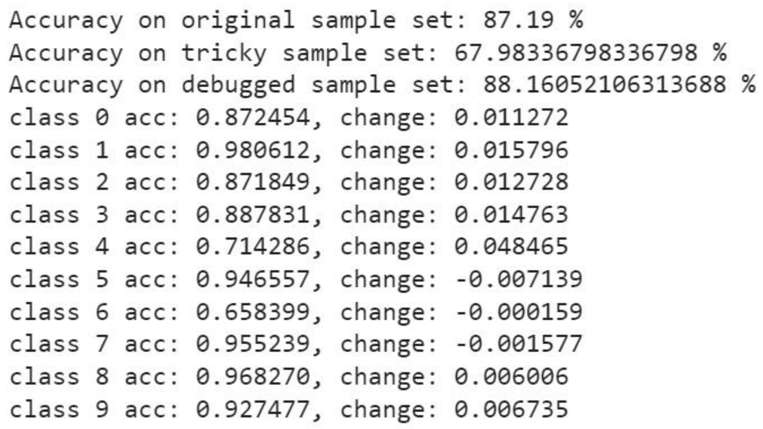


Each class has its own frequent traces. For instance, for class 0, trace 0-0-0 is the most frequent trace, there are also traces like 3-0-0, 3-3-0 or 3-3-3 and so on.

Traces for tricky samples, marked in orange, frequent 6-6-6 the most. This difference tells that they can be distinguish from each other. Note that the trace work is done only on training data.

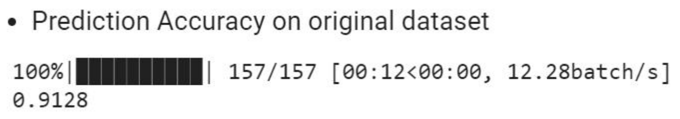
Final Result

- Group 1



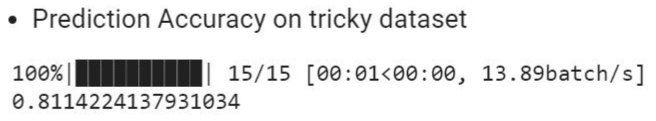
```
Accuracy on original sample set: 87.19 %
Accuracy on tricky sample set: 67.98336798336798 %
Accuracy on debugged sample set: 88.16052106313688 %
class 0 acc: 0.872454, change: 0.011272
class 1 acc: 0.980612, change: 0.015796
class 2 acc: 0.871849, change: 0.012728
class 3 acc: 0.887831, change: 0.014763
class 4 acc: 0.714286, change: 0.048465
class 5 acc: 0.946557, change: -0.007139
class 6 acc: 0.658399, change: -0.000159
class 7 acc: 0.955239, change: -0.001577
class 8 acc: 0.968270, change: 0.006006
class 9 acc: 0.927477, change: 0.006735
```

- Group 2



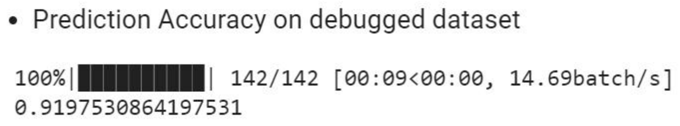
- Prediction Accuracy on original dataset

```
100%|██████████| 157/157 [00:12<00:00, 12.28batch/s]
0.9128
```



- Prediction Accuracy on tricky dataset

```
100%|██████████| 15/15 [00:01<00:00, 13.89batch/s]
0.8114224137931034
```



- Prediction Accuracy on debugged dataset

```
100%|██████████| 142/142 [00:09<00:00, 14.69batch/s]
0.9197530864197531
```

For tricky samples T predicted by judge layers, they are less likely to be accurately classified by the network, i.e. subjective to a distribution D_T that is different from that of original dataset D.

$$T \sim P(T \mid \mu, \sigma) \neq P(\mathrm{D} \mid \mu, \sigma)$$

and

$$Err(T) >> Err(\mathrm{D})$$

Dropped the tricky set, the accuracy of the network has been imporved by 1% percent. Consider that the judgement is made only by shallow judge layers at the entry of the network, instead of the whole network, this effect is promising.

UNIVERSITY OF ALBERTA

Potential Applications

- Adversarial Attack detection
- Accelerate the average inference speed by early stop
- Trace back to tricky samples and do futher weighted training
- ...

Further Improvements

- Explore alternative metrics for benchmark generation
- Use a more deterministic distance function
- Refine the God's dice approach to consider the number of judge layers
- ...

Thank you

Zhenglin Pan[1] *1781983*

1: University of Alberta, AB, Canada

Discussion (5%)