Generative Well-intentioned Networks

Justin Cosentino (justin@cosentino.io)
Jun Zhu (dcszj@mail.tsinghua.edu.cn)

Department of Computer Science, Tsinghua University

Oct. 30, 2019

Outline

- **Motivation:** Uncertainty & Classification w/ Reject
- **Framework**: Generative Well-intentioned Networks (GWIN)
- **Implementation**: Wasserstein GWIN
- **Results & Discussion**
- **Related Work**
- **Future Directions**

Motivation

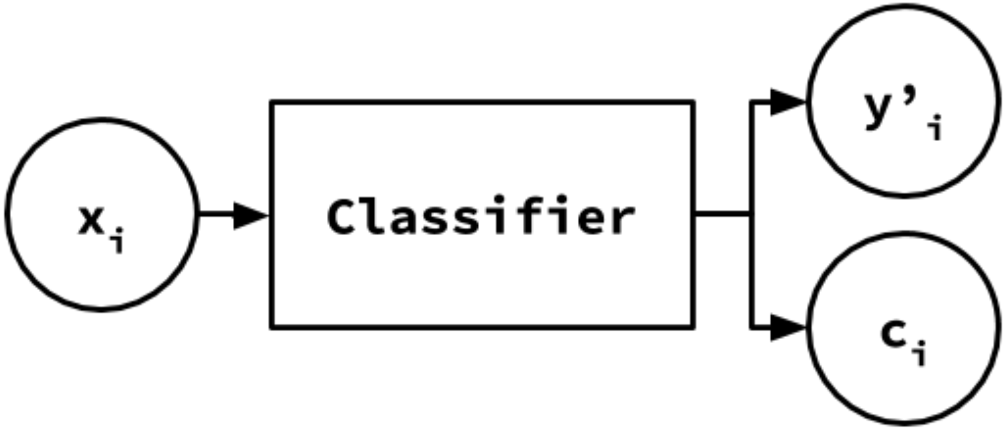
Uncertainty & Classification w/ Reject

Uncertainty in (Deep) Learning

- Understanding what a model does not know is essential
- Deep learning methodologies achieve state-of-the-art performance across a wide variety of domains, but do not capture uncertainty
  - Cannot treat softmax output as a “true” certainty (needs calibration)
  - Uncertainty is critical in many domains!
    - Machine learning for medical diagnoses
    - Autonomous vehicles
    - Critical systems infrastructure
- Traditional Bayesian approaches do not scale → Bayesian deep learning!

Uncertainty in Deep Learning; Dropout as a Bayesian Approximation; etc.

A standard classifier.

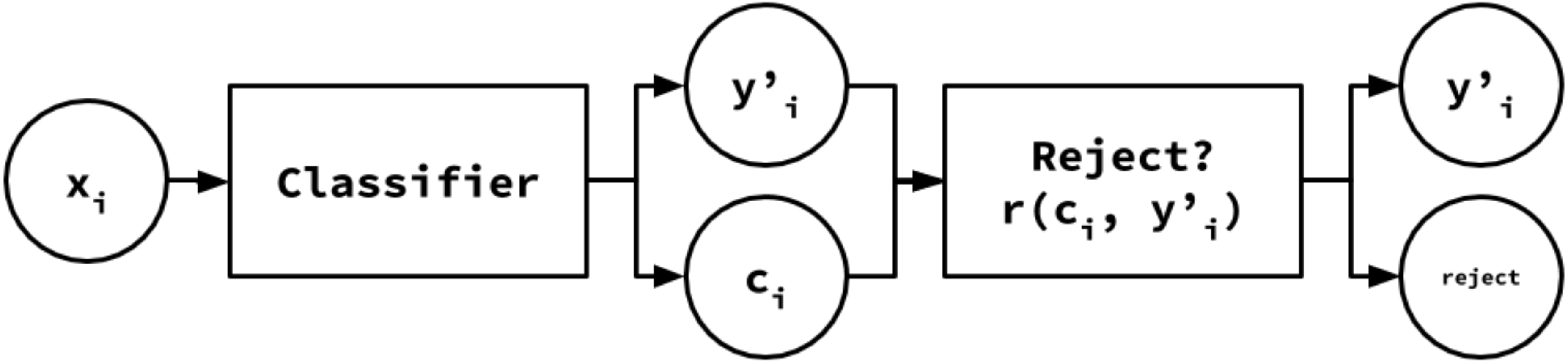


A classifier that emits a prediction and a certainty metric.

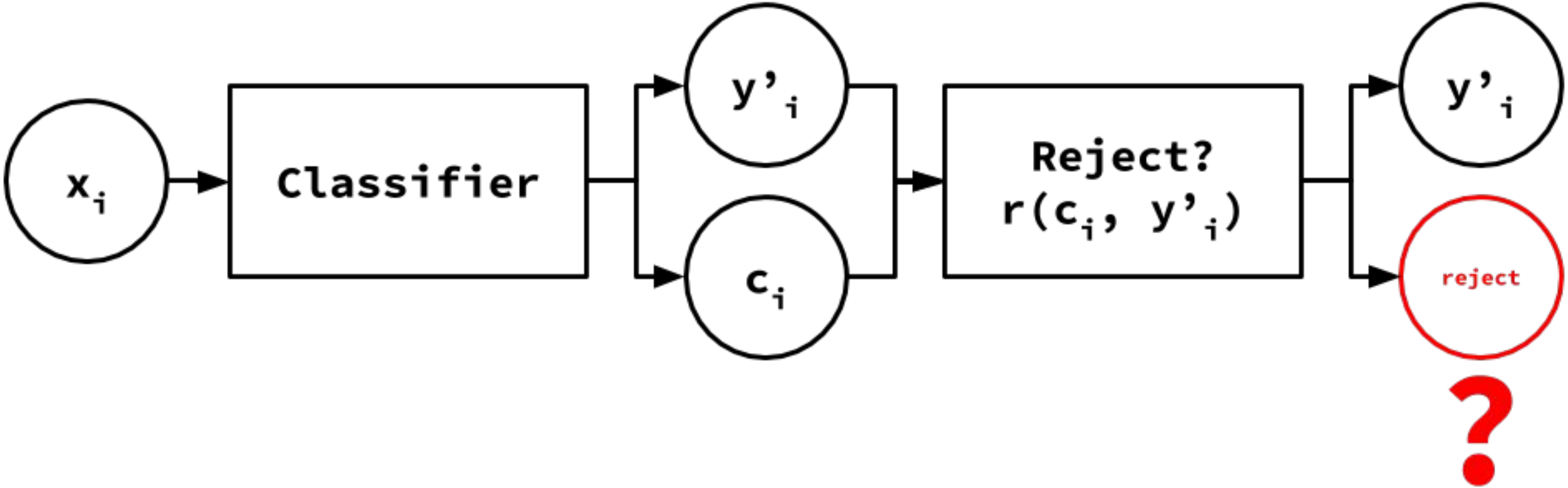
Rejection in (Deep) Learning

- How can we make use of these uncertainty estimates?
- Only label what we are certain of by introducing a rejection option
- Inherent tradeoff between error rate and rejection rate
- The problem of rejection can be formulated as
  - Given: training data $\{(x_i, y_i)\}^N_{i=1}$ and some target accuracy $1-\epsilon$
  - Goal: Learn a classifier C and a rejection rule r
  - Inference: given a sample x_k, reject if $r(x_k) < 0$, otherwise classify $C(x)$
- Majority of work focuses on binary reject in a non-deep learning setting

On optimum recognition error and reject trade-off; Learning with Rejection; Selective classification for deep neural networks



A classifier that emits a prediction and a certainty metric and that supports a reject option.



A classifier that emits a prediction and a certainty metric and that supports a reject option.

GWIN Framework

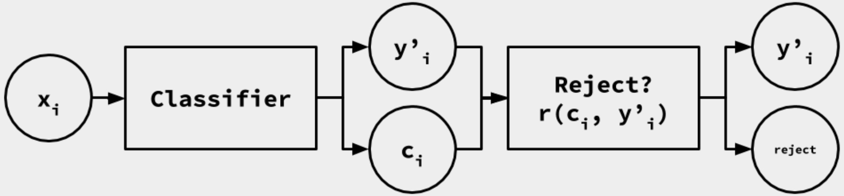
A novel method leveraging uncertainty and generative networks to handle classifier rejection.

Can we learn to map a classifier's uncertain distribution to high-confidence, correct representations?

Rather than simply rejecting input, can we treat the initial classifier as a “cheap” prediction and reformulate the observation if the classifier is uncertain?

GWIN Framework

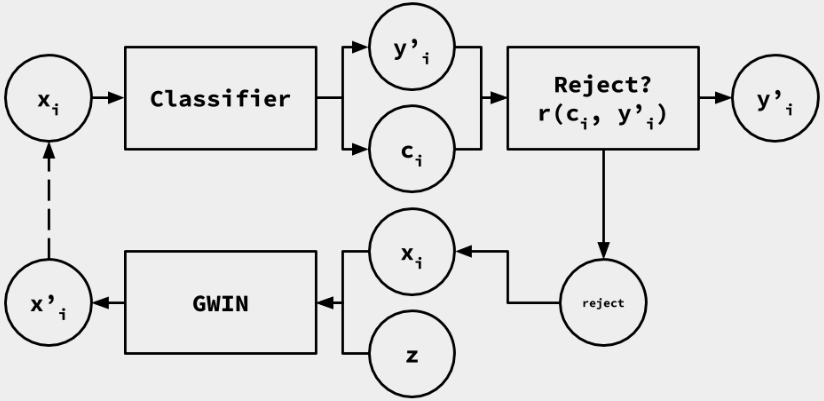
- A **pretrained**, **certainty-based classifier *C*** that emits a prediction and certainty
- A **rejection function *r*** that allows us to reject observations
- ...



A classifier that emits a prediction and a certainty metric and that supports a reject option.

GWIN Framework

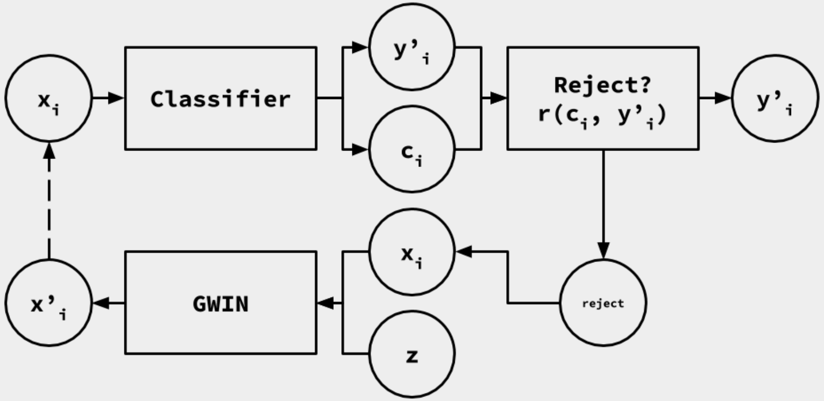
- A **pretrained**, **certainty-based classifier *C*** that emits a prediction and certainty
- A **rejection function *r*** that allows us to reject observations
- A **conditional generative network *G*** that transforms observations to new representations



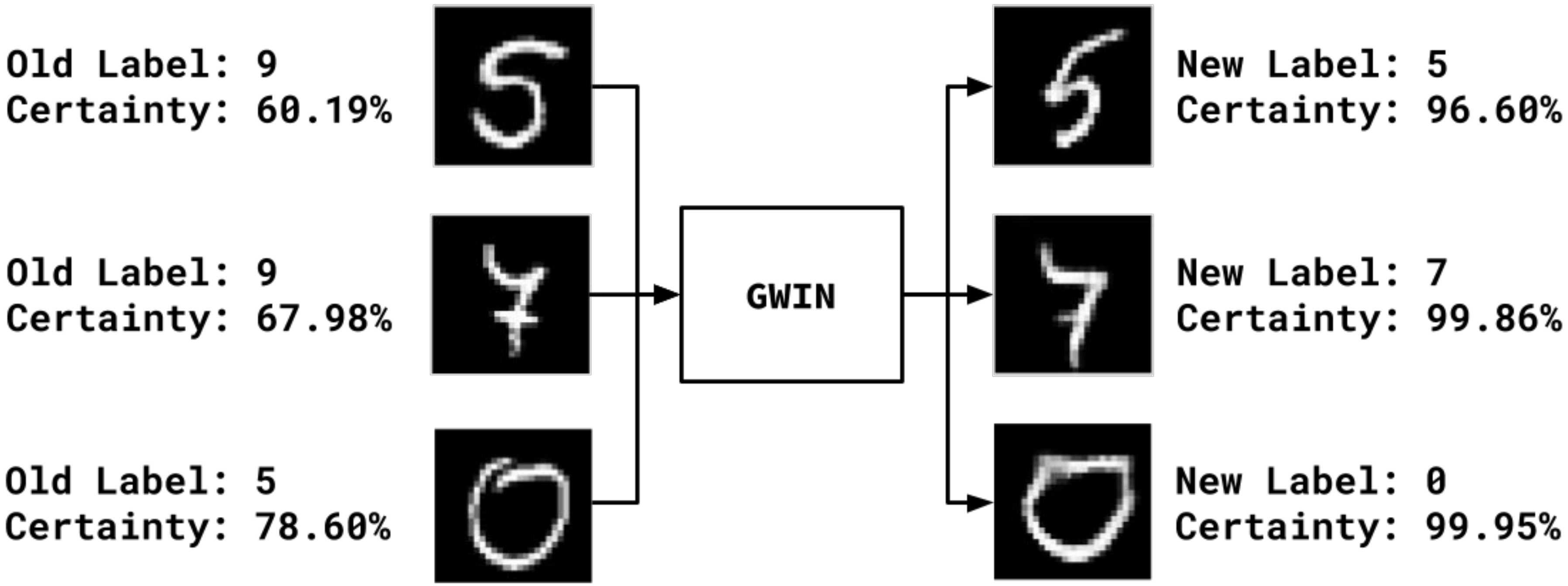
The GWIN inference process for some new observation $\mathbf{x}_i$.

GWIN Framework

- Used with *any* certainty-based classifier and does not modify the classifier structure
- Generator ***G*** learns the distribution of observations from the original data distribution that ***C*** labels correctly with high certainty
- No strong assumptions!



The GWIN inference process for some new observation $\mathbf{x_i}$.



Visualization of the GWIN transformation. Items on the left are rejected with τ=0.8 and transformed to “correct” representations.

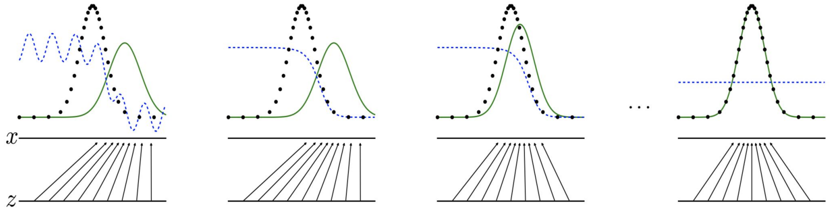
GAN Preliminaries

Quick Refresher on GANs

GANs

- Framework for estimating generative models using an adversarial network
- Contains two networks in a minimax-two player game:
  - Generative network ***G*** that captures the data distribution
  - Discriminative network ***D*** that estimates the source of a sample

$$\min_G \max_D \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\boldsymbol{x}}))]$$



[Generative Adversarial Networks]

Wasserstein GANs

- It is well known that GANs suffer from training instability:
  - mode collapse
  - non-convergence
  - diminishing gradient

- WGAN w/Earth-Mover distance:

$$\min_{G} \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}}))]$$

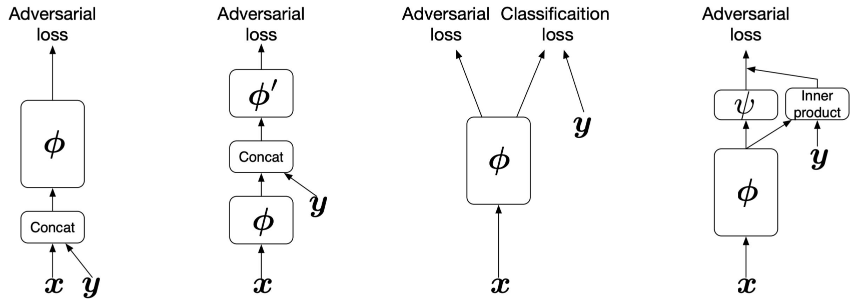
- WGAN with gradient penalty (WGAN-GP) further builds on this work, providing a final objective function with desirable properties:

$$\min_{G} \max_{D} \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]$$

Towards Principled Methods for Training Generative Adversarial Networks; Wasserstein GANs; Improved Training of Wasserstein GANs

Conditional GANs

- Extends the standard GAN to a conditional model by supplying extra information to both the critic and the generator
- Many different methods for conditioning:
    - Input concatenation
    - Hidden concatenation
    - Auxiliary classifiers
    - Projection
    - ...



Wasserstein GWIN

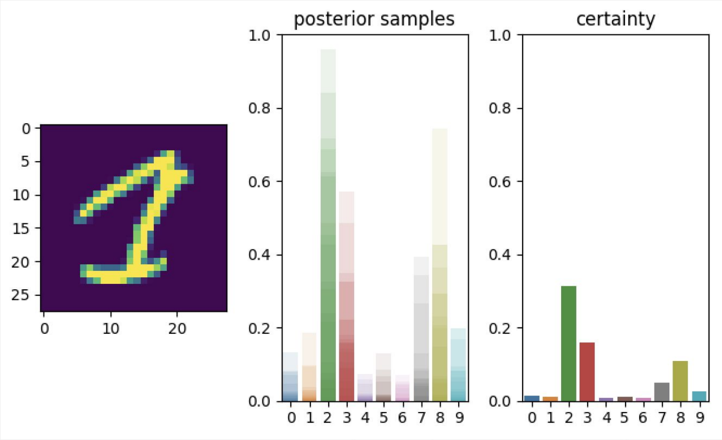
A Simple GWIN Architecture

Wasserstein GWIN (WGWIN-GP)

- **Classifier**: Bayesian Neural Network
  - Two architectures: LeNet-5 and “Improved”
  - Estimate uncertainty estimates using Monte Carlo sampling
- **Reject Function**: τ-based rejection rule
- **Generative Network**: Wasserstein GWIN (WGWIN-GP)
  - Based on Wasstein GAN with gradient penalty (WGAN-GP)
  - Modified loss function (transformation penalty)
  - Critic is trained on the “certain + correct” distribution
  - Conditional critic and generator

BNN Classifiers

- Evaluate two architectures:
  - LeNet-5 BNN
  - “Improved” BNN (BN, dropout, ...)
- Minimize ELBO loss
- Estimate model uncertainty using Monte Carlo sampling:
  - Determine the log probability of the observation given the training set by averaging draws
  - Look at mean / median of probs



Visualization of the BNN’s certainty estimation.

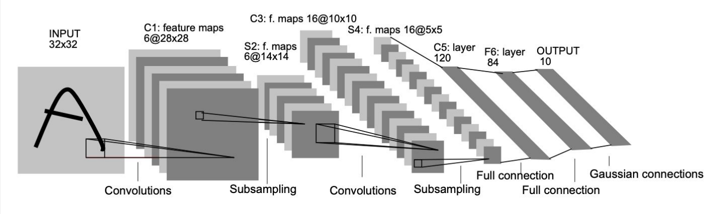


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

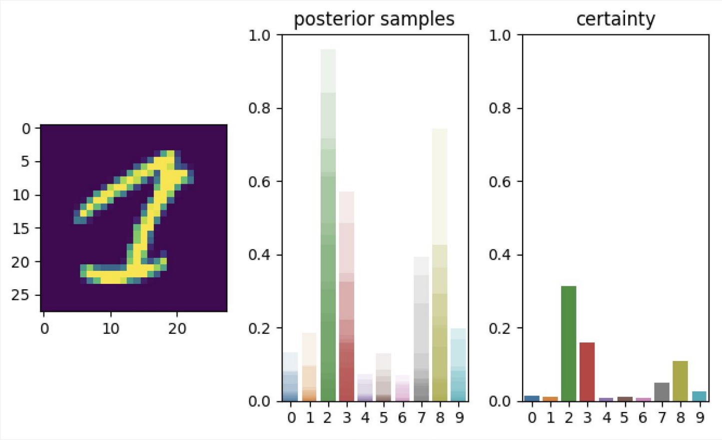
A diagram of the LeNet-5 architecture.

Rejection Function

- Simple threshold-based rejection function
- Give some rejection bound τ:

$$r(c_i, y_i') = \begin{cases} y_i', & \text{if } c_i \geq \tau \\ \text{reject}, & \text{otherwise.} \end{cases}$$

- Choice of τ is made at inference and can be tuned



Visualization of the BNN’s certainty estimation.

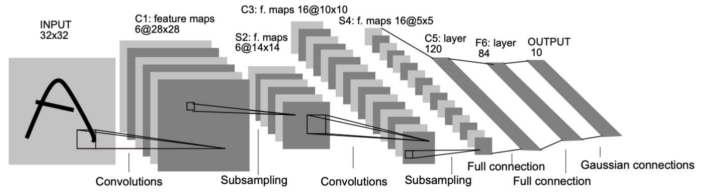
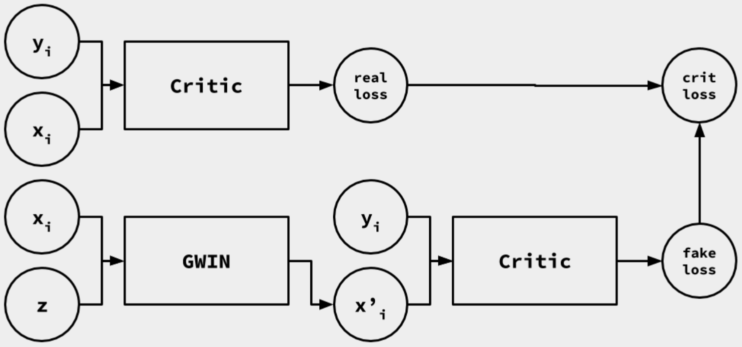


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

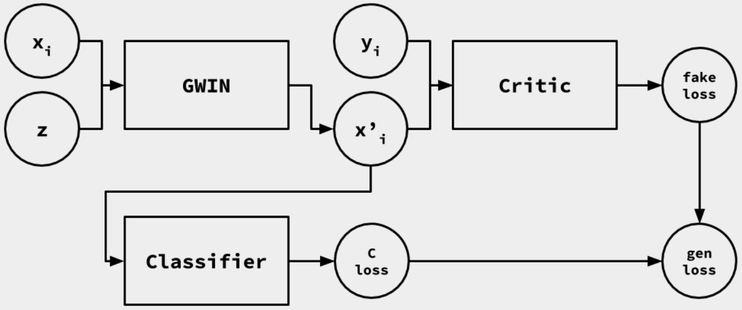
A diagram of the LeNet-5 architecture.

WGWIN-GP

- Architecture of the critic and generator follow WGAN-GP
- Add **conditioning** to both the critic and the generator:
  - The class label is depth-wise concatenated to the input and hidden layers of the critic
  - The current observation is flattened, concatenated with the noise vector, and passed to the generator
- Critic: trained on “certain” subset



The critic’s training pipeline (w/out gradient penalty).



The generator training pipeline (w/out penalty lambda).

WGWIN-GP Loss Function

- Introduces a new loss function with a **Transformation Penalty**
- This penalty penalizes the generator if it produces images that do not improve classifier performance:

$$L = \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x'} \sim \mathbb{P}_g}[D(\boldsymbol{x'}, y)] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_c}[D(\boldsymbol{x}, y)]}_{\text{WGAN Loss}} + \underbrace{\lambda_{GP} \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}[(||\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}, y)||_2 - 1)^2]}_{\text{WGAN-GP Penalty}} + \underbrace{\lambda_{Loss} \mathop{\mathbb{E}}_{\boldsymbol{x'} \sim \mathbb{P}_g}[\text{Loss}(C(\boldsymbol{x'}))]}_{\text{Transformation Penalty}}$$

- In practice, we find $\boldsymbol{\lambda_{GP} = \lambda_{LOSS} = 10}$ to work well

Require : The penalty coefficients λ_{GP} and λ_{Loss}, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2, certainty preprocessing threshold τ^*, and classifier C.

Require : initial critic parameters w_0, initial generator parameters θ_0

Build confident data distribution $\mathbb{P}_c$ from training data $\mathbb{P}_r$ using classifier C and threshold τ^*

while θ *has not converged* **do**

  for $t = 1, \dots, n_{critic}$ **do**

    for $i = 1, \dots, m$ **do**

      Sample confident data $(\boldsymbol{x}, y) \sim \mathbb{P}_c$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, and a random number $\epsilon \sim U[0, 1]$.

      $\boldsymbol{x'} \leftarrow G_\theta(\boldsymbol{x}, \boldsymbol{z})$

      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\boldsymbol{x'}$

      $L^{(i)} \leftarrow D_w(\boldsymbol{x'}, y) - D_w(\boldsymbol{x}, y) + \lambda_{GP}(||\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}}, y)||_2 - 1)^2$

    end for

    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$

  end for

  Sample a batch of training data $\{(\boldsymbol{x}, y)^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ and latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(z)$

  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{x}, \boldsymbol{z}), y) + \lambda_{Loss}(\text{Loss}(C(G_\theta(\boldsymbol{x}, \boldsymbol{z})))), \theta, \alpha, \beta_1, \beta_2)$

end while

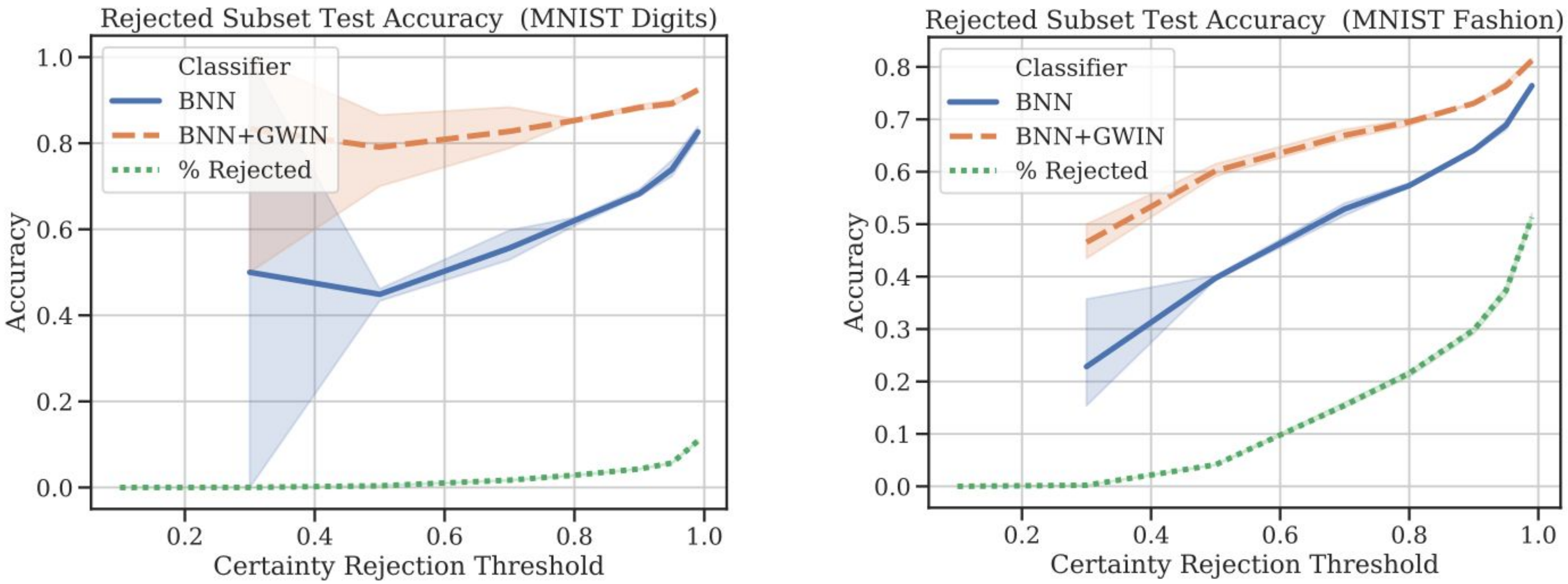
Results & Discussion

LeNet-5 and “Improved” BNN + WGWIN-GP

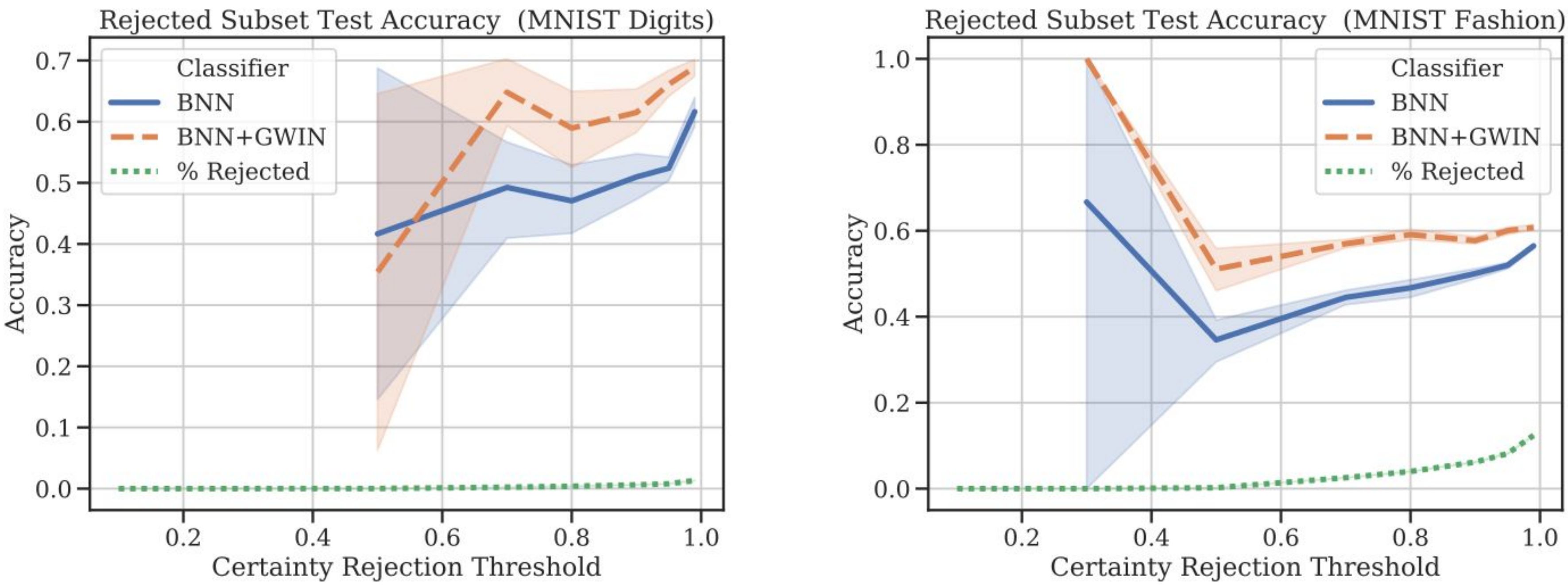
Experimental Design

- Classifiers: LeNet-5 and “Improved” BNN
- Generator: WGWIN-GP
- Rejection: τ-based rejection rule
  - $\tau \in$ { 0.1, 0.3, 0.5, 0.7, 0.8, 0.9, 0.95, 0.99 }
  - Reject inputs transformed once and then relabled
- Datasets: MNIST Digits and MNIST Fashion
  - Train: 50k
  - Eval: 10k
  - Test: 10k
  - Confident set built from train data

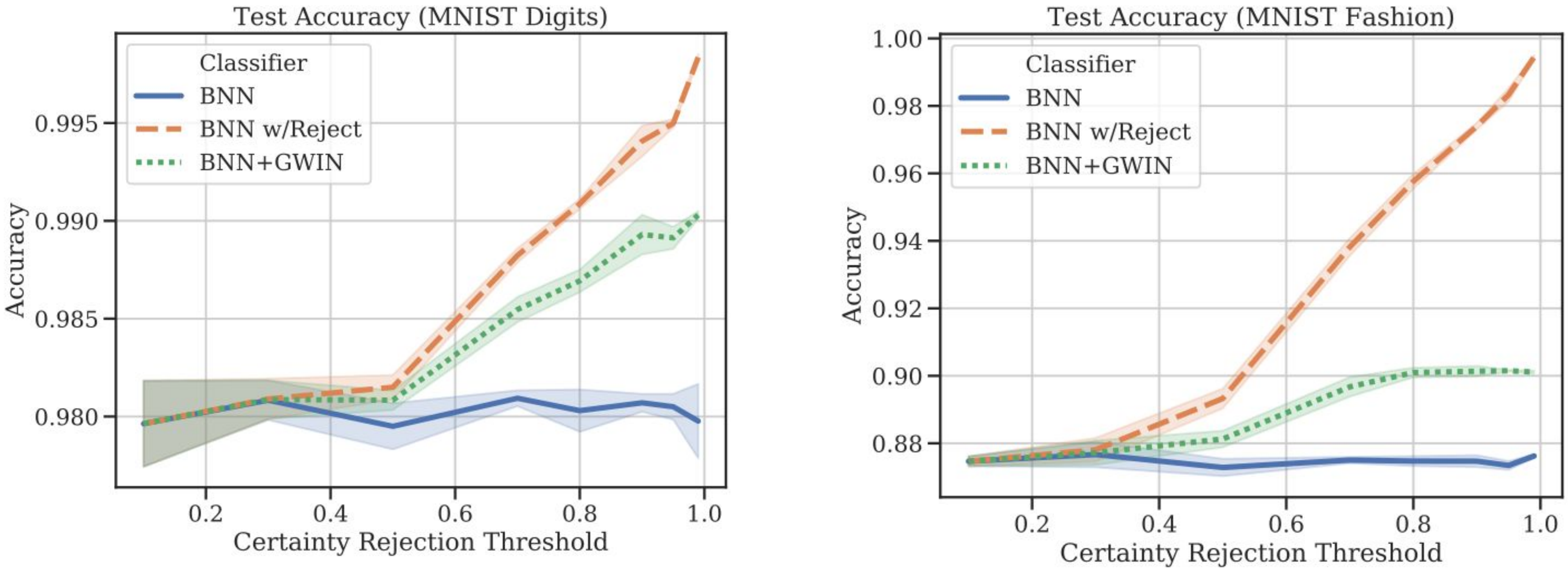
MNIST Digits; Fashion MNIST



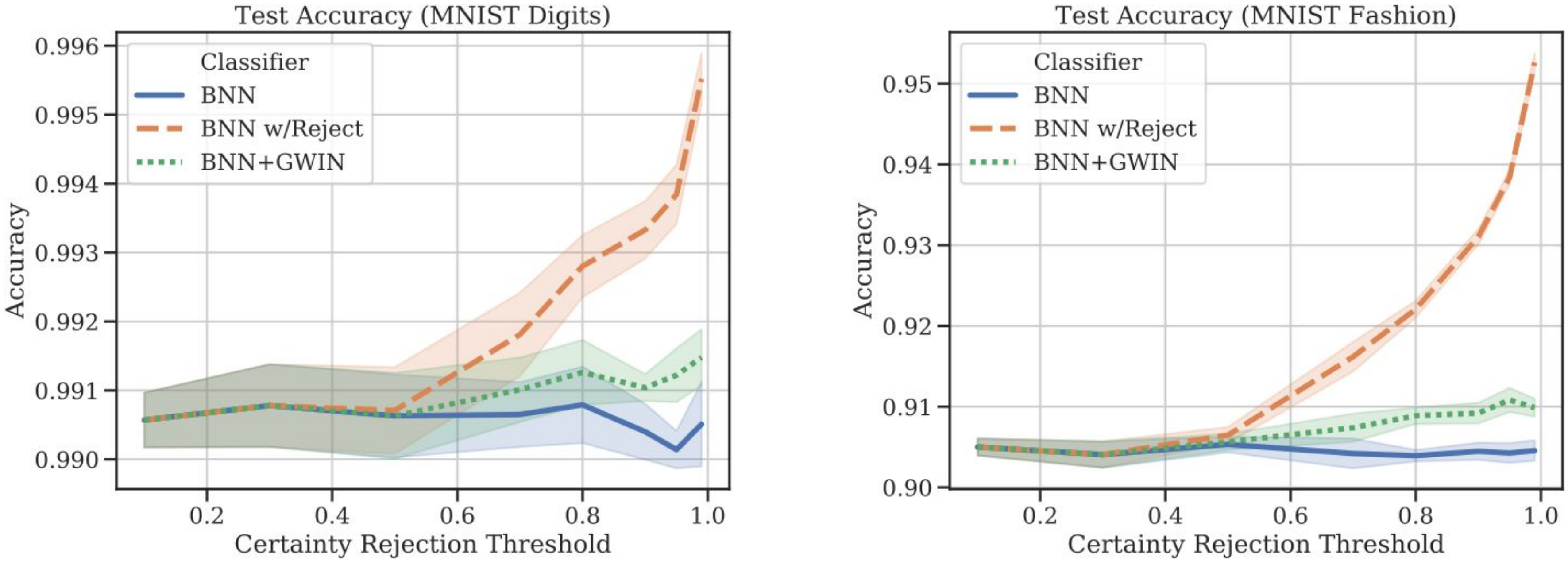
Change in **LeNet-5 accuracy** on the **rejected subset** for **varying rejection rates** $\boldsymbol{\tau}$. *BNN* denotes standard BNN performance while *BNN+GWN* denotes the classifier's performance on transformed images. *% Rejected* denotes the % of observations rejected by the classifier.



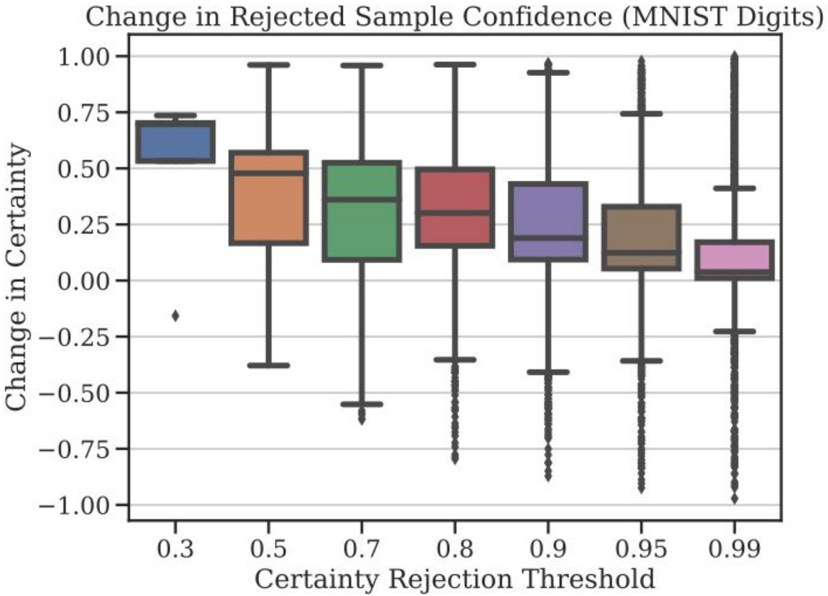
Change in **Improved BNN accuracy** on the **rejected subset** for **varying rejection rates τ**. *BNN* denotes standard BNN performance while *BNN+GWN* denotes the classifier's performance on transformed images. *% Rejected* denotes the % of observations rejected by the classifier.

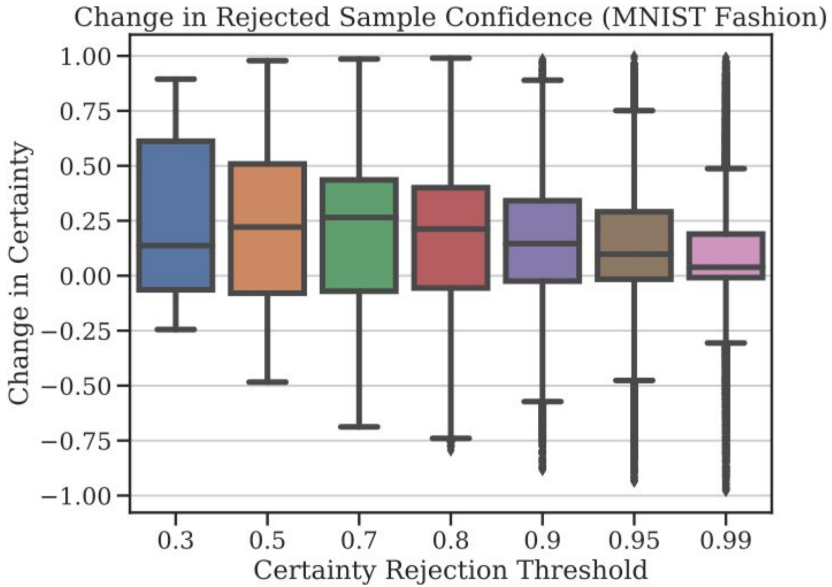


Change in **LeNet-5 accuracy** on the **test set** for **varying rejection rates τ**. *BNN* denotes standard BNN performance, *BNN+GWN* denotes the classifier's performance on transformed, rejected images, and *BNN w/Reject* denotes the classifier's performance with a "reject" option (not required to label).

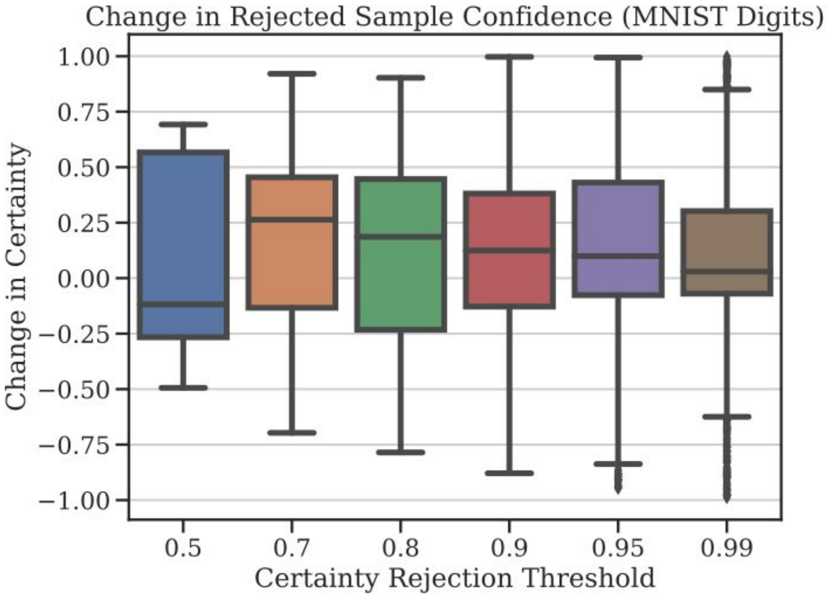


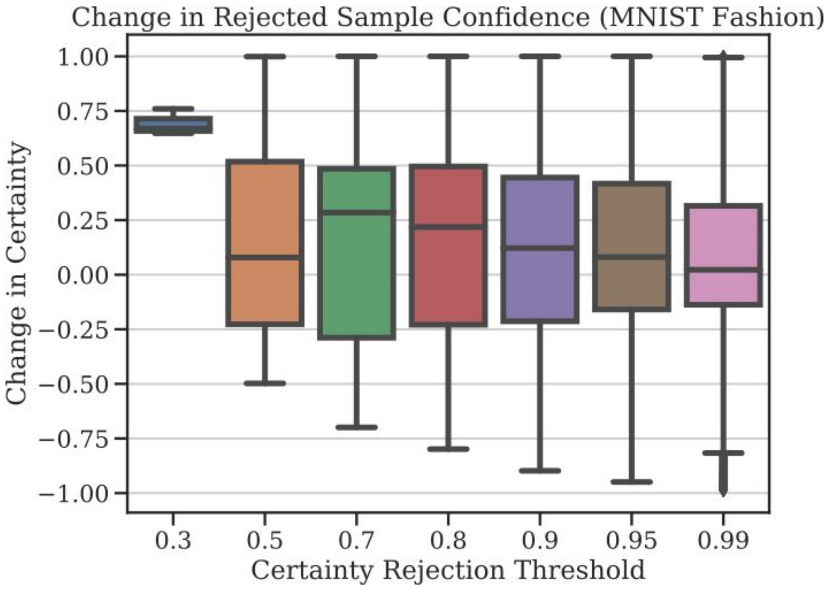
Change in **Improved BNN accuracy** on the **test set** for **varying rejection rates τ**. *BNN* denotes standard BNN performance, *BNN+GWN* denotes the classifier's performance on transformed, rejected images, and *BNN w/Reject* denotes the classifier's performance with a "reject" option (not required to label).





Change in **LeNet-5 certainty for the ground-truth class** in the **rejected subset** for **varying rejection rates τ**.
Outliers are those values that fall outside of 1.5IQR and are denoted with diamonds.





Change in **Improved BNN certainty for the ground-truth class** in the **rejected subset** for **varying rejection rates** τ. Outliers are those values that fall outside of 1.5IQR and are denoted with diamonds.

Discussion

- BNN+GWIN performance is consistently better than the BNN at most certainty thresholds; addition of transformation, without modifying the base classifier, improves performance on uncertain observations.
- The GWIN transformation increases certainty in the correct class in the majority of classes; tradeoff between rejection threshold and accuracy.
- We see gains in <u>rejected subset</u> accuracy, but these gains do not have a large impact on <u>overall</u> accuracy if the rejected subset is small

Related Work

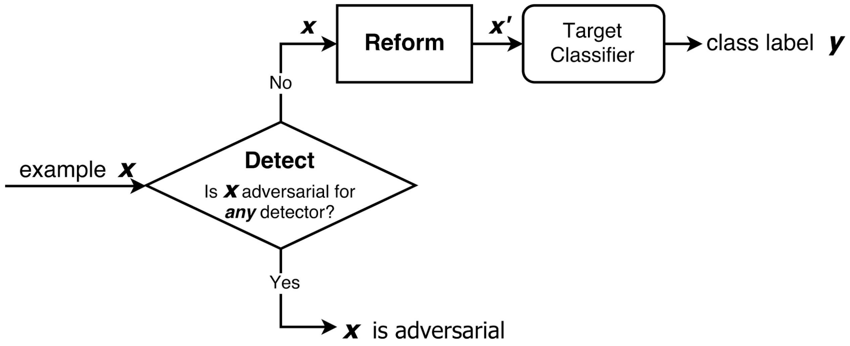
A comparison with denoising and robustness methods

Denoising and Robustness Methods

- Network distillation: trains a classifier such that it is nearly impossible to generate adversarial examples using gradient-based attacks.
- Data augmentation
  - Adversarial training
  - Hallucination methods
  - …
- Defense using generative models:
  - MagNet: a Two-Pronged Defense against Adversarial Examples
  - Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models

MagNet

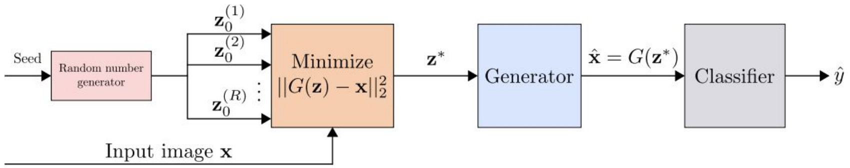
- Does not modify protected classifier
- MagNet consists of two core components:
  - a **detector** that rejects examples that are far from the manifold boundary
  - a **reformer** that, given an example x, strives to find an example x' on or close to the manifold where x' is a close approximation to x, and then gives x' to the target classifier
- Uses **autoencoders** rather than GANs
- Use a series of detectors; select one at random to increase robustness of model



MagNet workflow in test phase. MagNet includes one or more detectors. It considers a test example x adversarial if any detector considers x adversarial. If x is not considered adversarial, MagNet reforms it before feeding it to the target classifier

Defense-GAN

- Does not modify protected classifier and makes weaker assumptions about the classifier than GWINs
- Defense-GAN aims to **denoise** adversarial examples by projecting images back to the real data set while **minimizing reconstruction loss**
- Defense-GAN preprocesses all input to the classifier, incurring a larger transformation cost
- Only used in the context of defense from adversarial attacks



Overview of the Defense-GAN algorithm.

Conclusions and Future Work

- Proposed a new framework for leveraging uncertainty and generative networks to handle classifier rejection
- Showed that this works with a very simple proof of concept (WGWIN-GP)
- Next steps:
  - Encourage mode collapse for high-certainty representations?
  - Iterative transformation process
  - Explore other, more powerful GWIN architectures
    - Principled classification with reject?
    - Variational autoencoders?
    - Larger networks, different conditioning methods?

References

- Uncertainty in Deep Learning
- Dropout as a Bayesian Approximation
- On optimum recognition error and reject trade-off
- Learning with Rejection
- Selective classification for deep neural networks
- Generative Adversarial Networks
- Towards Principled Methods for Training Generative Adversarial Networks
- Wasserstein GANs
- Improved Training of Wasserstein GANs
- Conditional Generative Adversarial Nets
- cGANs with Projection Discriminator
- Generative Adversarial Text to Image Synthesis
- Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks
- MagNet: a Two-Pronged Defense against Adversarial Examples
- Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models

Please see our paper for a full list of references.

Thanks!

Justin Cosentino (justin@cosentino.io)
Jun Zhu (dcszj@mail.tsinghua.edu.cn)

Department of Computer Science, Tsinghua University

Oct. 30, 2019