Careful Deep Learning: Learning to Abstain by Training on A Simple Loss Function

Ziyin Liu,
University of Tokyo

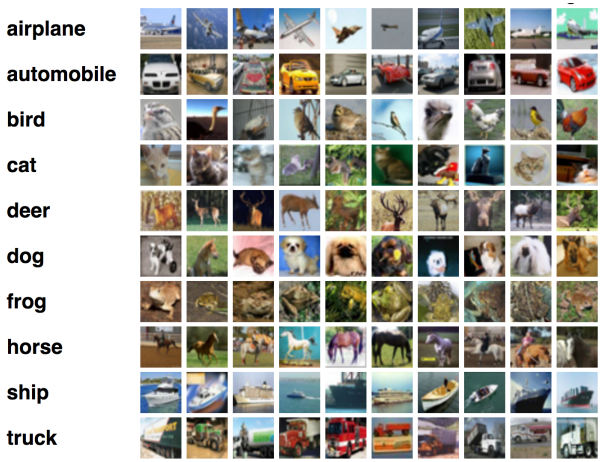
-Ziyin Liu (IPI), Zhikang T. Wang(IPI), Paul Liang(CMU), Ruslan salakhutdinov (CMU), Louis-Philippe Morency (CMU), Masahito Ueda (IPI), *Deep Gamblers: Learning to Abstain with Portfolio Theory*, Neural Information Processing Systems (NeurIPS 2019)

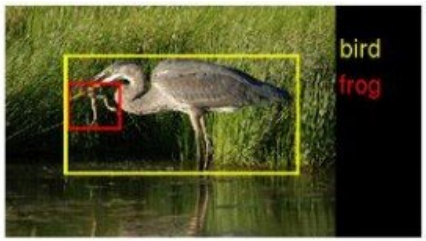
-Ziyin Liu (IPI), Ru Wang(UTokyo), Paul Liang(CMU), Ruslan salakhutdinov (CMU), Louis-Philippe Morency (CMU), Masahito Ueda (IPI), *Unpublished Work*, arxiv print: 19xx.xxxxx

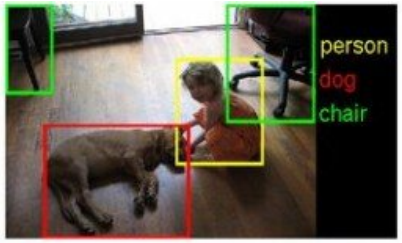
東京大学
THE UNIVERSITY OF TOKYO

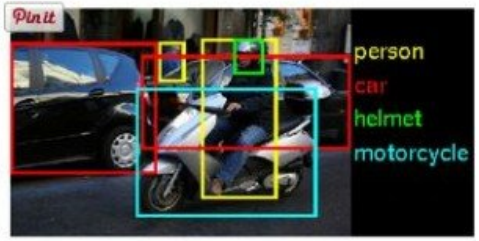
Institute for Physics of Intelligence

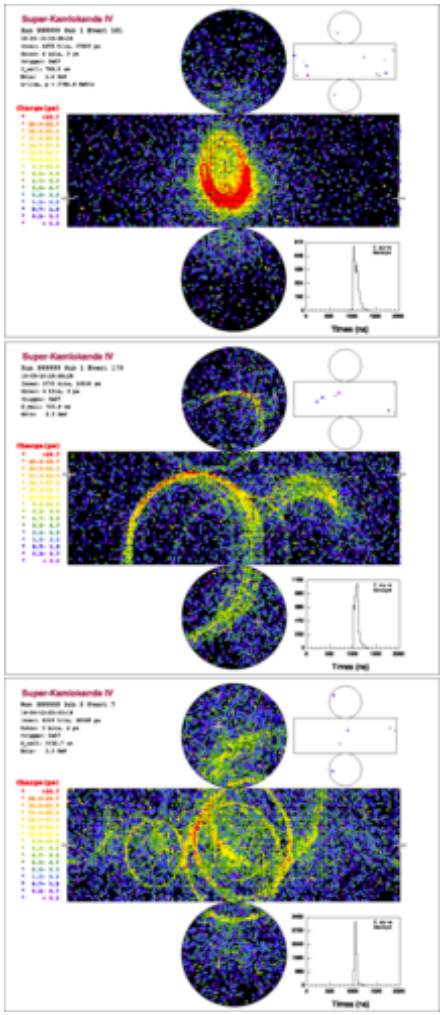
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PENNSYLVANIA 1900





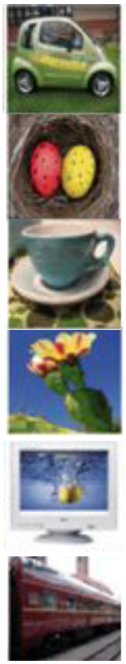






Left: Tsung-Yi Lin, arxiv.org/abs/1405.0312 Right: Z. Li, PRD 2018

Deep Learning Recap



grass

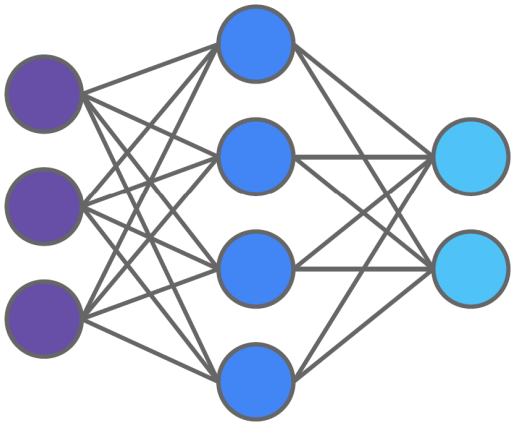
ball

table

flower

computer

train



cross-entropy loss

The nll loss for Classification

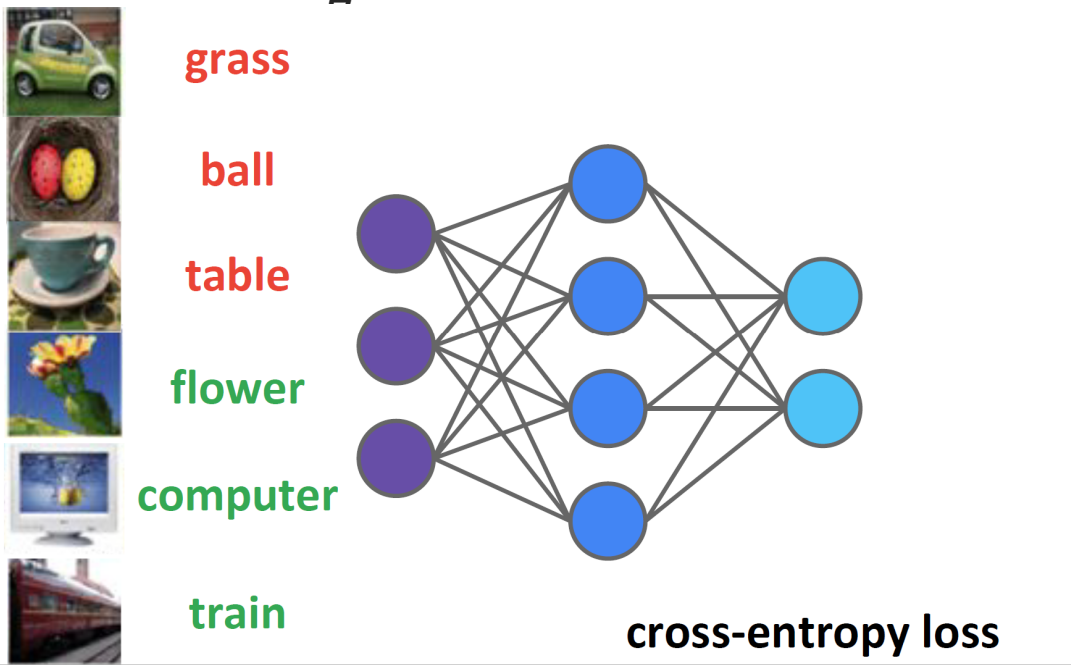
Given data x, we want to predict its label y, i. e. $\Pr(y|x)$

Maximum Likelihood Estimation:

$$\text{Want to find: } \theta = \arg\max_{\theta} \Pr(Y|\theta)$$

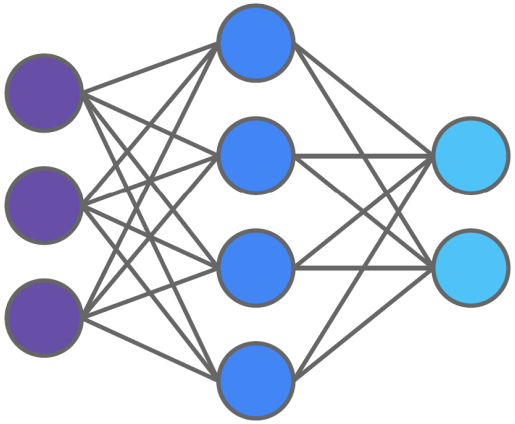
In practice, minimize $negative\ log\ loss$ (nll loss):

$$\min_{\theta} -\log p(Y|\theta)$$



Problems of the nll loss

- for example…



Problems of the nll loss

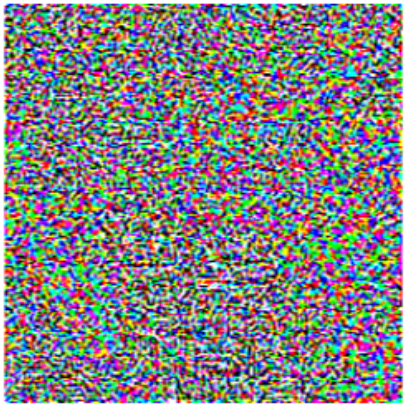
- A Simple fix?
  - observe datapoint x
  - predict probability $\hat{p}(y|x)$,
  - compute its entropy $H(\hat{y}|x)$
  - set threshold h; if $H(\hat{y}|x) > h$, mark the datapoint as uncertain
  - send uncertain datapoint to a human expert, and wait for decision

- But this does not work well

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$

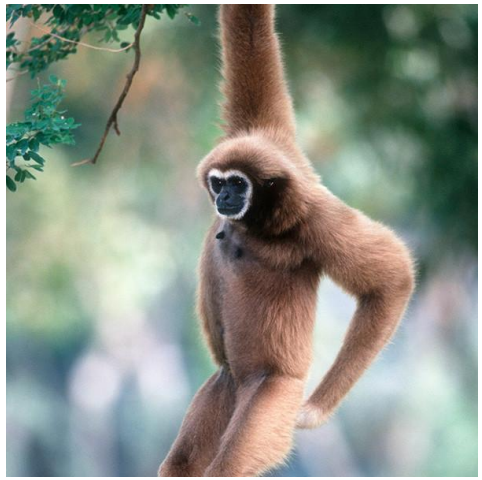


$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence



[Goodfellow et al., ICLR 2016]

Two Lessons

1. Representations learned by neural networks are not robust

   (the common lesson)

2. The output predictions (or functions thereof) ***do not reflect prediction confidence***

Problems of the nll loss

- Overfitting
- Outliers
  - Predicted probability $\hat{p}(x|y)$ is different from the true data distribution $p(x|y)$
- Model has no way of expressing uncertainty / confidence
- We need some special method to assess prediction confidence

Augmenting the nll loss






grass

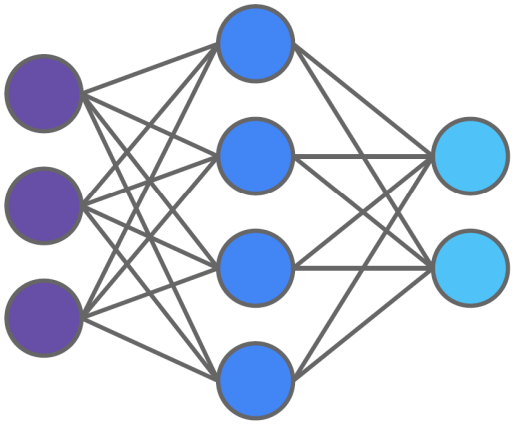
ball

table

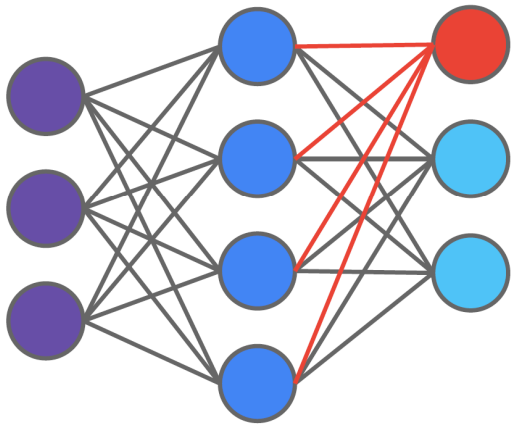
flower

computer

train



cross-entropy loss



confidence

gambler's loss

Outline

- Classification Problems in Deep Learning
  - Maximum Likelihood Estimation
  - Why Abstention?
- Related Works
  - Uncertainty Estimation in Deep Learning
- Proposed Method
  - Classification as Gambling
  - Safe Classification as Gambling with Reservation
- Some Future Work
  - The label noise problem
  - Robust Loss function against overfitting

Uncertainty in Deep Learning

- Ensembling

  Combining differently initialized networks

  [Lakshminarayanan et al., NIPS2016]

- Bayesian Deep Learning

  Uncertainty in Weight [Blundell, ICML 2015]

  Prior: $p(\theta)$

  Learning parameter θ *and its variance* σ *at the same time*

Uncertainty in Deep Learning

2 Types of Uncertainties :

Data Uncertainty (Aleatoric Uncertainty):

- Inherent uncertainty nature of the problem
- Out-of-distribution uncertainty

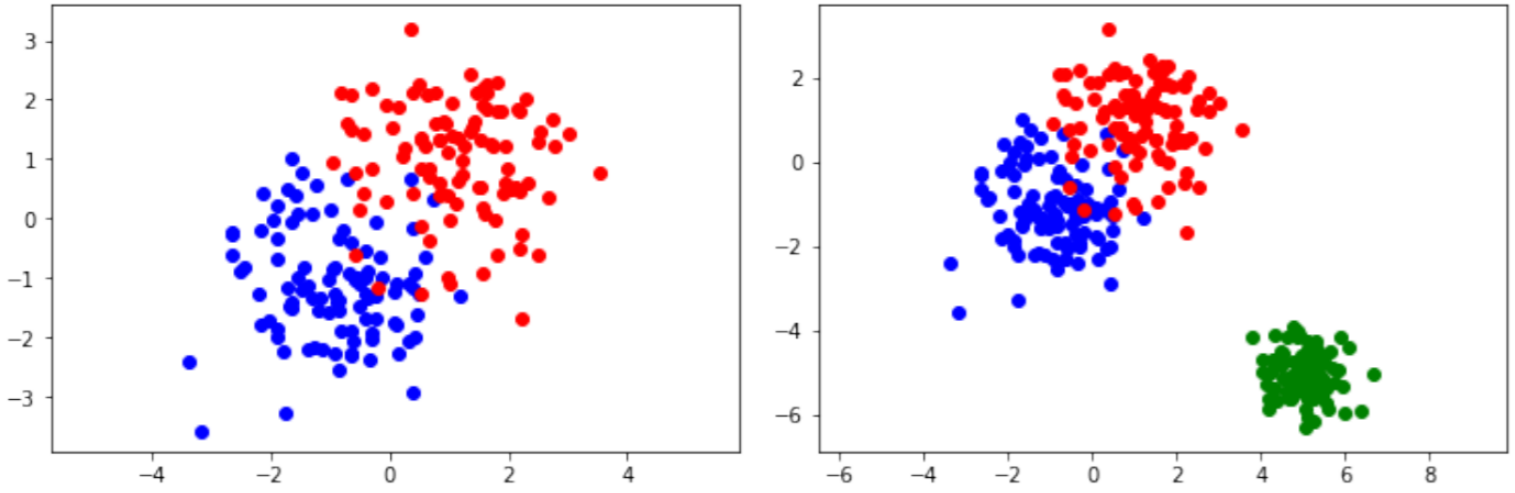
Model Uncertainty (Epistemic Uncertainty):

- Uncertainty in model parameters
- Structure uncertainty

Uncertainty in Deep Learning

Data Uncertainty:

- Inherent uncertainty nature of the problem
- Out-of-distribution samples



(a) Training Set

(b) Testing Set

Outline

- Classification Problems in Deep Learning
  - Maximum Likelihood Estimation
  - Why Abstention?
- Related Works
  - Uncertainty Estimation in Deep Learning
- Proposed Method
  - Classification as Gambling
  - Safe Classification as Gambling with Reservation
- Some Future Work
  - The label noise problem
  - Robust Loss function against overfitting

Horse Race

Horse Race

- m horses
- Wealth relative: $b_i,\ \sum_i^m b_i = 1$
- Chance of winning: p_i
- Payoff if we bet on the winning horse: o_i
- Return after winning: $S = o_i b_i$

Objective: maximize doubling rate:

$$\max \mathrm{W}(\mathrm{b}, \mathrm{p}) = \max E[\log S\,(b)]$$

Horse Race

Horse Race

Optimal Strategy: proportional betting: $b_i \propto p_i$

Horse Race with Side Information

Side information: x

$b_i \propto p(y = i|x)$

Classification Problem = Betting problem

Horse Race with Reservation

Horse Race with Reservation

m horses

Betting strategy: $\sum_{i=1}^{M} b_i \quad \rightarrow \quad \sum_{i=0}^{m} b_i$

Chance of winning: p_i

Payoff if we bet on the winning horse: o_i

Return after winning: $S = o_i b_i \quad \rightarrow \quad o_i b_i + b_0$

Objective: maximize doubling rate:

$$\max W = \max E \log(S) = \max \sum_{i=1}^{m} p_i \log(o_i b_i + b_0)$$

Classification Problem = Betting problem with Reservation with $o = 1, b_0 = 0$

Classification Problem $\leq$ Betting problem with Reservation

Gambling with Neural Networks

In short:

1. Initialize network $\mathbf{b_i} := f_i(x; \theta)$
2. Train to maximize generlized objective $W = \sum_{i=1}^{m} p_i \log(o_i b_i + b_0)$
3. Characterize uncertainty with b_0
4. Label sample as uncertain if $b_0 > h$ for some threshold h

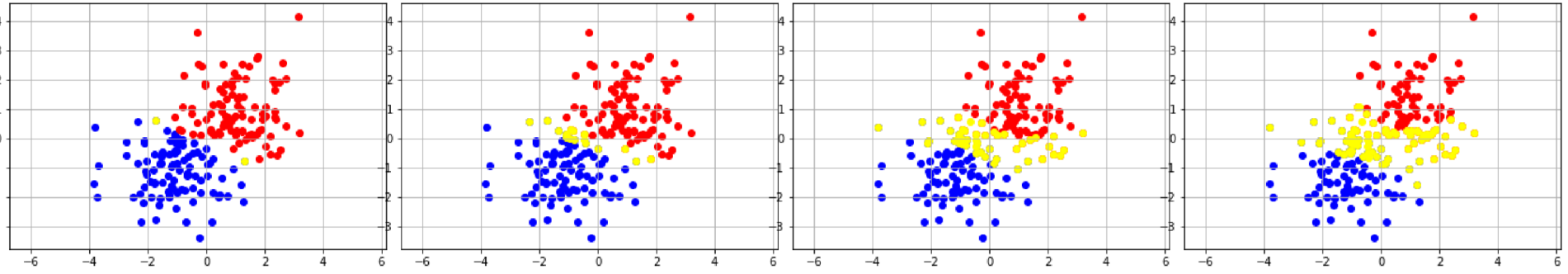
We call b_0 the “Disconfidence Score”

Experiments

Inherent Uncertainty:

Blue & Red: Confident Predictions

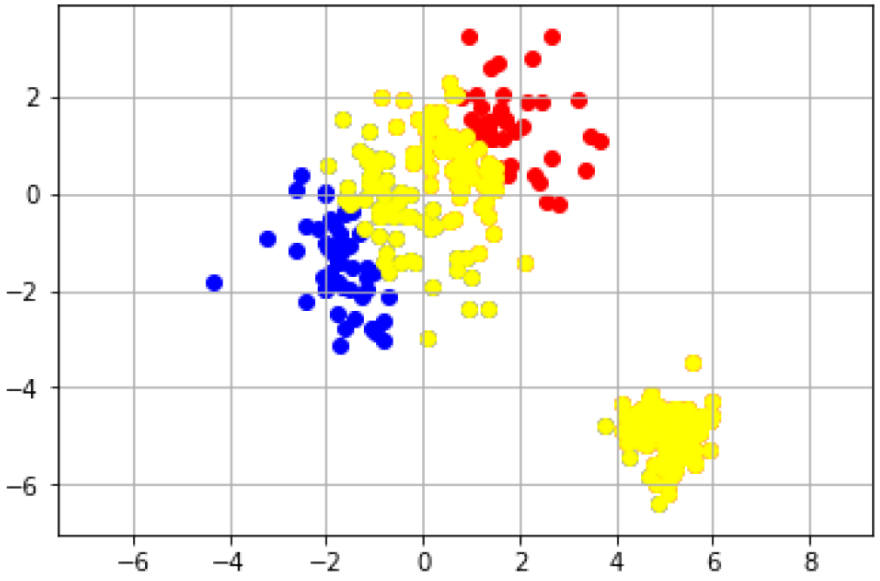
Yellow: Disconfident Inputs



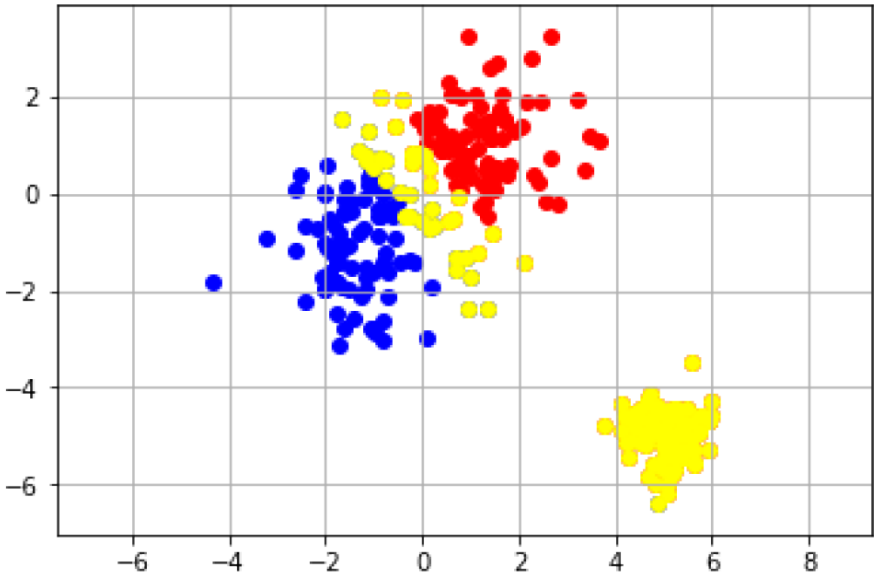
(b) $h = 0.999$ (c) $h = 0.99$ (d) $h = 0.9$ (e) $h = 0.5$

Experiments

Out of distribution problem:



(a) entropy based



(b) deep gamblers

Experiments

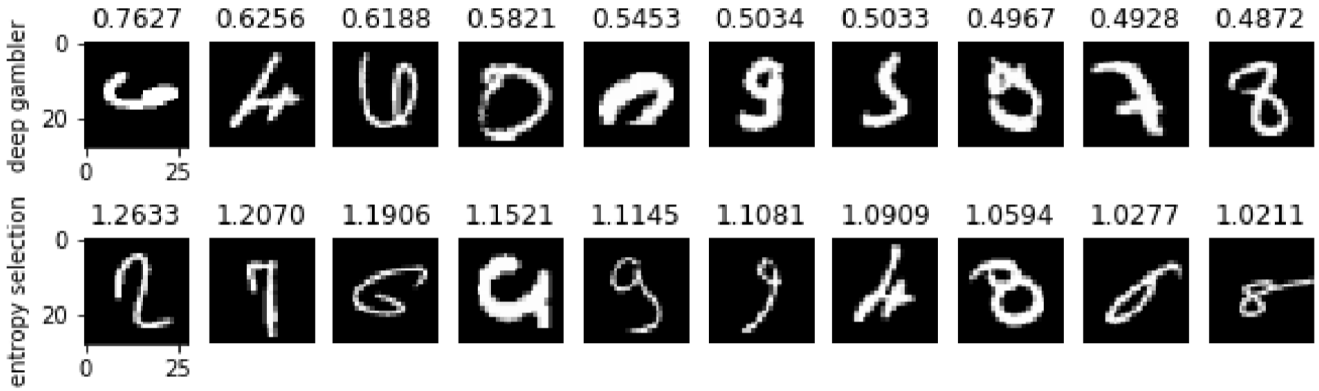


Figure 1: Top-10 rejected images in the MNIST testing set found by two methods. The number above image is the predicted uncertainty score (ours) or the entropy of the prediction (baseline). For the top-2 images, our method chooses images that are hard to recognize, while that of the baseline can be identified unambiguously by human.

Experiments

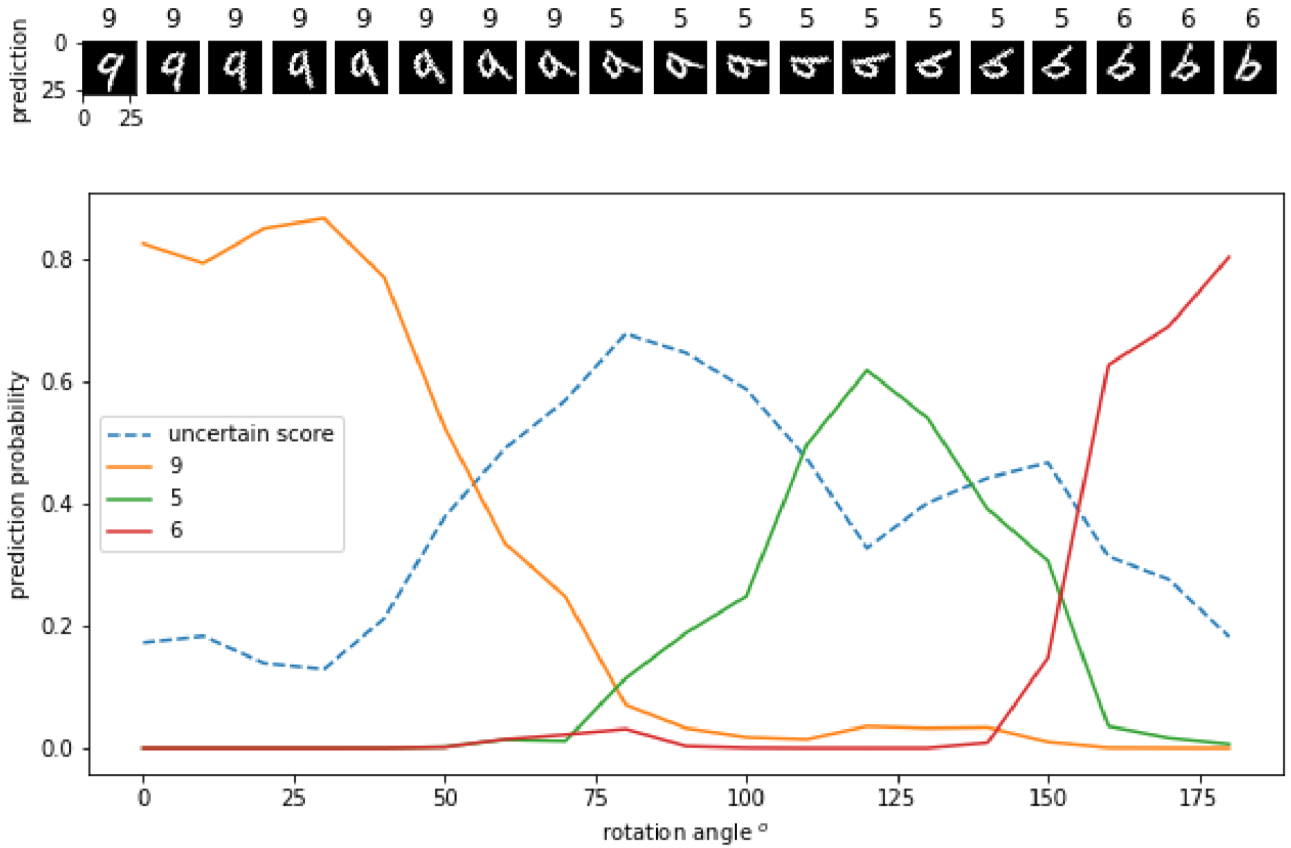


Figure 4: Rotating an image of 9 by 180 degrees. The number above the images are the prediction label of the rotated image.

Benchmarking

Very competitive results...

Coverage	Ours (Best Single Model)	Ours (Best per coverage)	SR	BD	SN
1.00	$^{o=2.6}3.24 \pm 0.09$	–	3.21	3.21	3.21
0.95	$^{o=2.6}\mathbf{1.36 \pm 0.02}$	$^{o=2.6}\mathbf{1.36 \pm 0.02}$	1.39	1.40	1.40
0.90	$^{o=2.6}\mathbf{0.76 \pm 0.05}$	$^{o=2.6}\mathbf{0.76 \pm 0.05}$	0.89	0.90	0.82 ± 0.01
0.85	$^{o=2.6}\mathbf{0.57 \pm 0.07}$	$^{o=3.6}0.66 \pm 0.01$	0.70	0.71	$\mathbf{0.60 \pm 0.01}$
0.80	$^{o=2.6}\mathbf{0.51 \pm 0.05}$	$^{o=3.6}\mathbf{0.53 \pm 0.04}$	0.61	0.61	$\mathbf{0.53 \pm 0.01}$

Table 3: SVHN. The number is error percentage on the covered dataset; the lower the better. We see that our method achieved competitive results across all coverages. It is the SOTA method at coverage $(0.85, 1.00)$.

Coverage	Ours (Single Best Model)	Ours (Best per Coverage)	SR	BD	SN
1.00	$^{o=2.0}2.93 \pm 0.17$	–	3.58	3.58	3.58
0.95	$^{o=2.0}1.23 \pm 0.12$	$^{o=1.4}\mathbf{0.88 \pm 0.38}$	1.91	1.92	1.62
0.90	$^{o=2.0}\mathbf{0.59 \pm 0.13}$	$^{o=2.0}\mathbf{0.59 \pm 0.13}$	1.10	1.10	0.93
0.85	$^{o=2.0}0.47 \pm 0.10$	$^{o=1.2}\mathbf{0.24 \pm 0.10}$	0.82	0.78	0.56
0.80	$^{o=2.0}\mathbf{0.46 \pm 0.08}$	$^{o=2.0}\mathbf{0.46 \pm 0.08}$	0.68	0.55	$\mathbf{0.35 \pm 0.09}$

Table 5: Cats vs. Dogs. The number is error percentage on the covered dataset; the lower the better. This dataset is a binary classification, and the input images have larger resolution.

Why does the gambler's loss work?

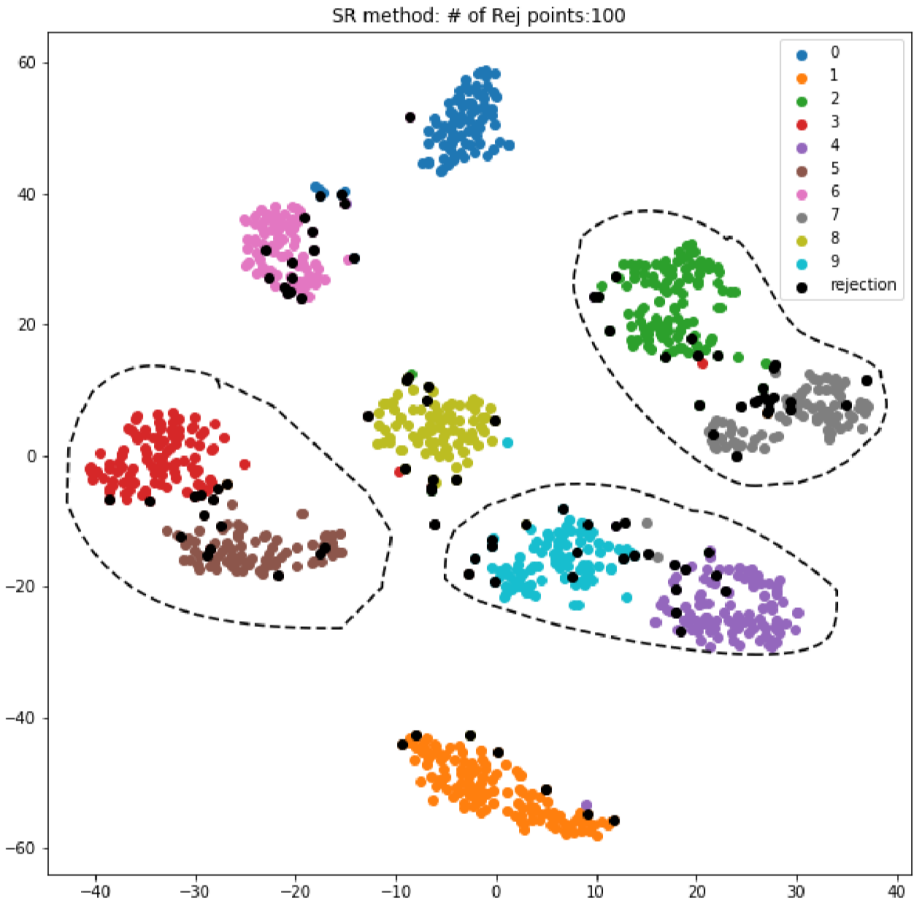
Consider: $\max W = \max \sum_{i=1}^{m} p_i \log(o_i b_i + b_0)$

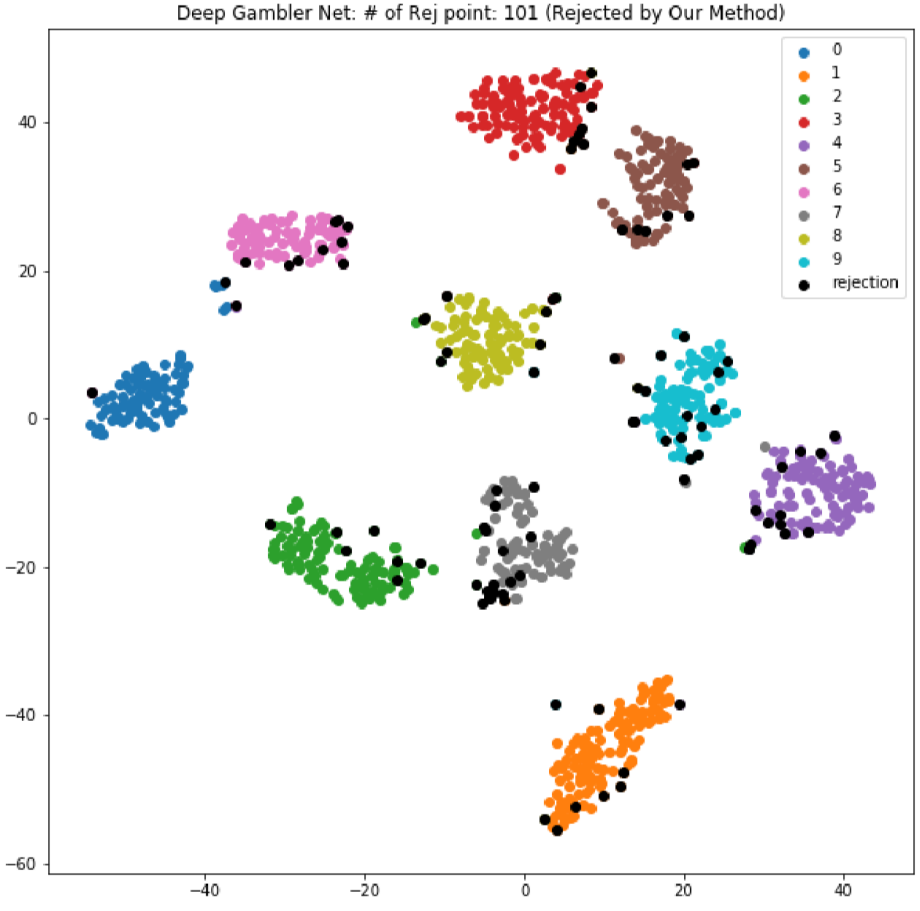
The stationary solution can be uniquely solved via KKT condition without seeing the image

– should be no "magic"

Why does the gambler's loss work?

The reason seems dynamical…





(a) Normal Model

(b) Deep Gambler

Outline

- Classification Problems in Deep Learning
  - Maximum Likelihood Estimation
  - Why Abstention?
- Related Works
  - Uncertainty Estimation in Deep Learning
- Proposed Method
  - Classification as Gambling
  - Safe Classification as Gambling with Reservation
- Some Future Work
  - The label noise problem
  - Robust Loss function against overfitting

Some Future Work

ϵ-Label noise problem:

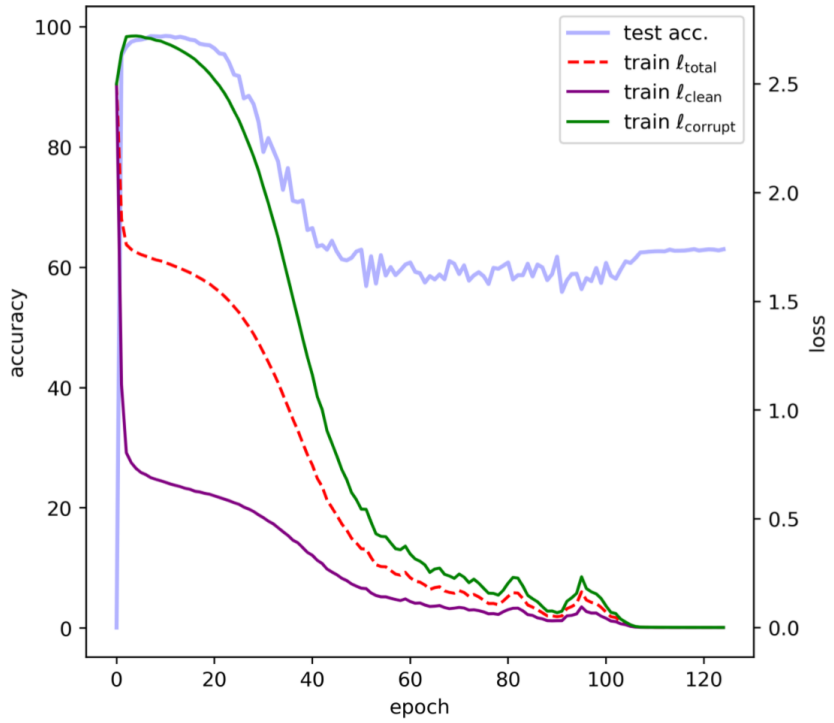
Uniform Random Corruption of probability $1 - \epsilon$ exists

Dataset $D = D_{clean} \cup D_{corrupt}$

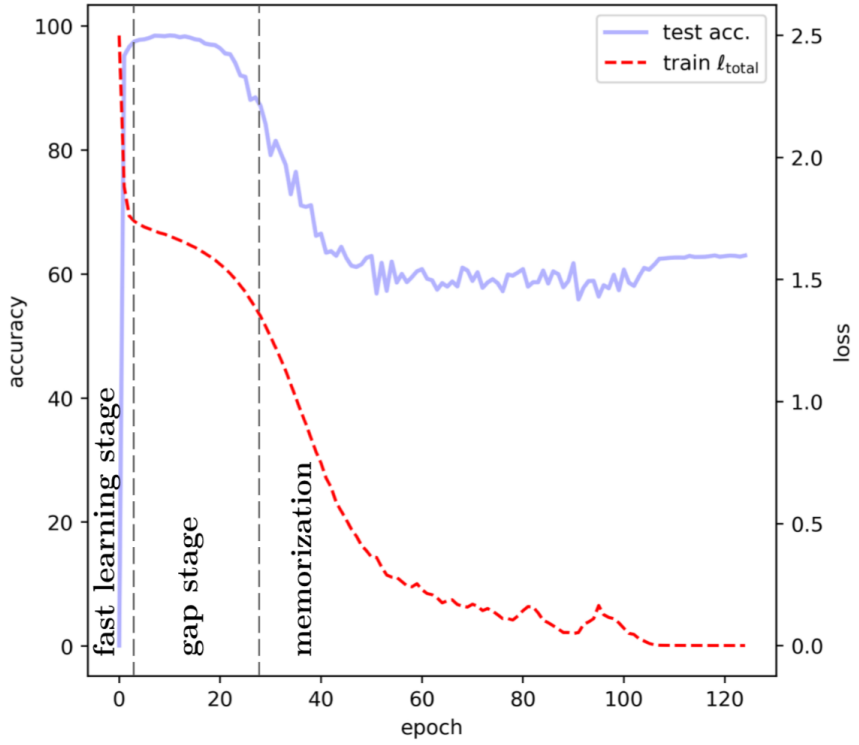
Common in real life problems

Some Future Work

Typical learning curve in the presence of label noise… [Ziyin et al., Unpublished work, 2019]



(a)



(b)

Figure 1: Different stages in the presence of label noise. (a) We plot ℓ_{total} (total loss), ℓ_{clean} (loss on $\mathcal{D}_{\text{clean}}$), and ℓ_{corrupt} (loss on $\mathcal{D}_{corrupt}$), we see that during the gap stage, there is a clear "gap" between the ℓ_{clean} and ℓ_{corrupt} where training on clean labels has completed but training on noisy labels has barely started; (b) Hypothesized qualitative division of the three stages: fast learning stage, gap stage, and the memorization stage. Experiment done on MNIST with corruption rate 0.5.

Label Noise Problem

Typical Strategies:

Use a special learning algorithm $A(f, \epsilon)$ to alleviate the negative influence from the noises

F-matrix: proposed a surrogate loss function [Patrini, CVPR 2017]

Co-teaching: train 2 networks, and make them teach each other [Han bo et al., NeuRIPS 2018]

Problems:

Requires knowing ϵ

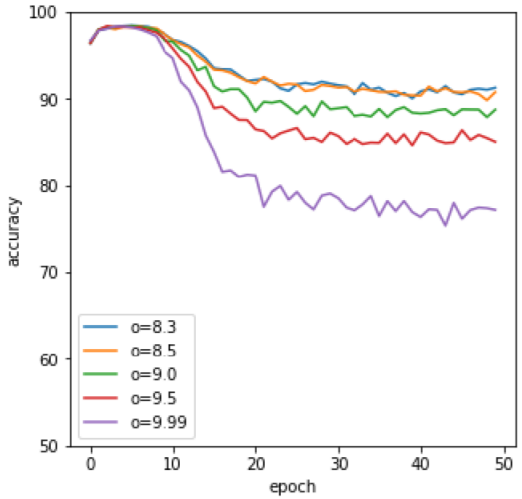
Separation of normal learning and noisy learning

Label Noise Problem

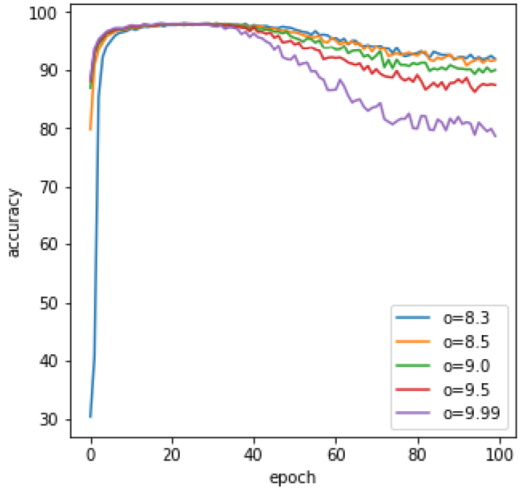
Gambler's loss:

Automatic robustness to label noise

Dataset	Performance	
	nll loss	Gblers
MN $r = 0.2$	84.7 ± 0.5	96.7 ± 0.2
MN $r = 0.5$	55.1 ± 3.1	91.2 ± 0.7
MN $r = 0.65$	39.7 ± 2.5	85.9 ± 1.1
MN $r = 0.8$	19.1 ± 3.0	76.1 ± 0.3
MN $r = 0.85$	14.5 ± 0.7	71.0 ± 1.4



(a) Training with Adam



(b) Training with SGD

Label Noise Problem

Gambler's loss:

Reduced overfitting

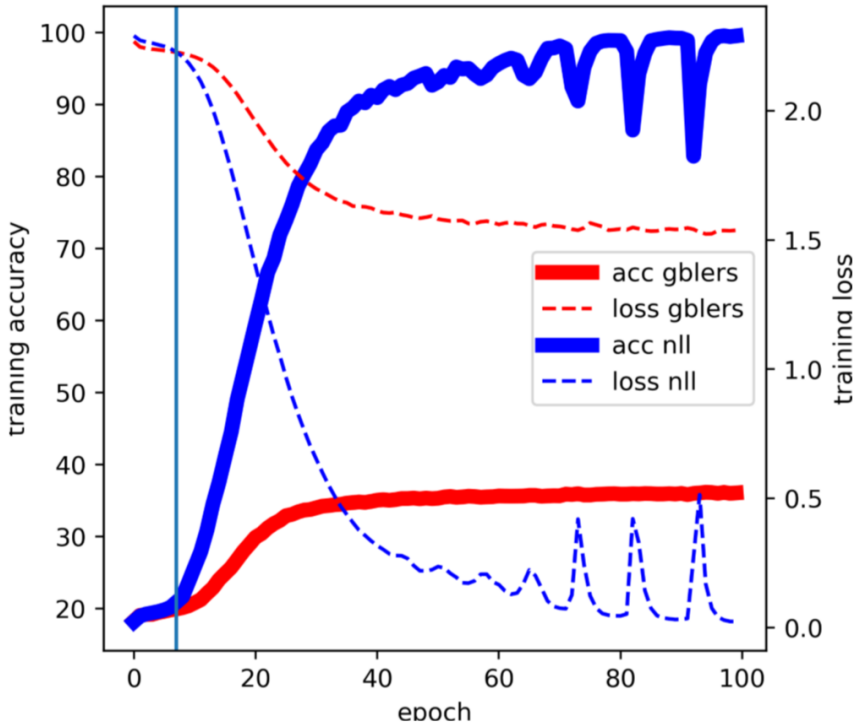


Figure 6: Training accuracy and training loss On MNIST with corruption rate 0.8 with Adam. $o = 9.7$. Training with gambler's loss prevents memorization of noisy labels. At convergence, *nll* loss reaches 19% testing accuracy, while the gambler's loss stays around 76%.

Key Messages

- classification is a special case of gambling
- the $gambler's\ loss$ is a natural generalization of the $nll\ loss$
- $gambler's\ loss$ reduces overfitting

Future Work

- extension of the gambler's loss to regression problems
- adversarial attacks
- application to real life / large scale datasets

Contact

Email: zliu@cat.phys.s.u-Tokyo.ac.jp
Website: http://cat.phys.s.u-tokyo.ac.jp/~zliu/