Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

Contents

Abstract
Abstract / Contributions

Background
Bayesian and DL / Calibration / Adversarial Training

Deep Ensemble
Proper scoring rules / Adversarial Training / Ensembles / Algorithms

Experiments
Setup and Toy Example / Regression & Classification / Uncertainty Evaluation / Accuracy as a Function of Confidence

+Reference

Abstract

- 지금까지의 DNNs의 성과와 별개로 uncertainty를 구하는 것은 또 다른 challenge. 현대 딥러닝 모델은 overconfident한 경향을 보임
- 지금까지 Bayesian-NN 을 활용해 uncertainty를 구했지만 많은 parameter수정과 연산을 요구함
- 본 연구에서 병렬적으로 uncertainty estimation이 가능한 Non-Bayesian 방법론을 제시

Contributions

- Simple하면서도 scalable한 uncertainty estimation 방법론을 제시한다. 또한 ensemble과 adversarial training 을 활용한 모델 설계 및 학습을 하며 이것이 주어진 문제 해결에 도움이 되었음
- Calibration과 일반화의 관점에서 uncertainty estimation의 quality를 측정하는 방법을 제시

Background

A. Bayesian and DL

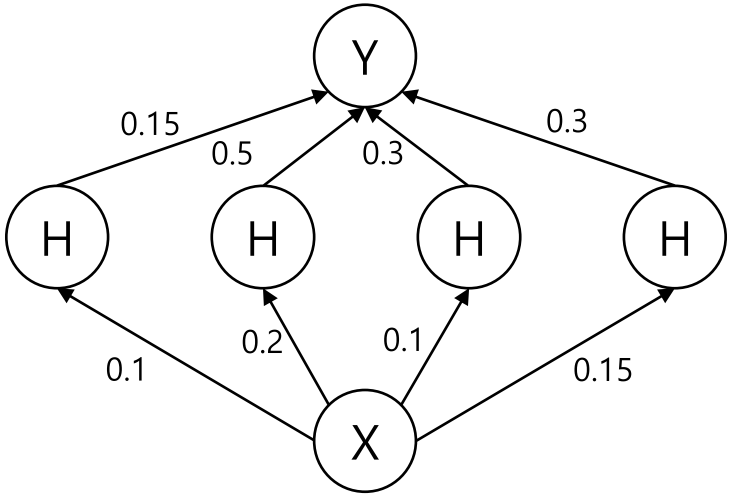
(1) $P(B|A) = \frac{P(A|B)P(B)}{P(A)} = \frac{Likelihood\ P \times Prior\ P}{Evidence} = Posterior\ P$

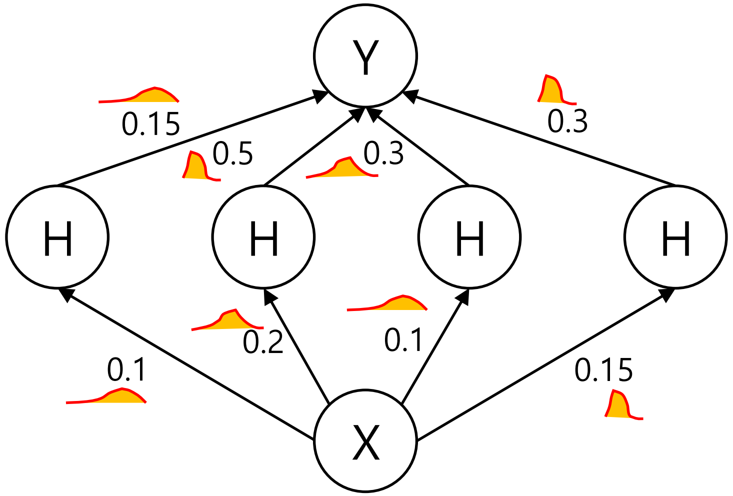
(2) $P(W|Y,X) = \frac{P(Y|W,X)P(W)}{P(Y|X)}$ (3) $P(W|D) = \frac{P(D|W)P(W)}{P(D)}$

- 암에 걸릴 확률이 0.1%이다. 어떤 테스트기는 암이 걸린 사람 99%에게 양성 반응을 보이고 병이 없는 사람에게는 1%확률로 양성반응을 보인다. 검사 결과 양성이었을 때 정말 암에 걸렸을 확률은?
- H : 암에 걸림, E : 양성반응
- $p(H|E) = \frac{0.99*0.001}{0.001*0.99+0.999*0.01} = 9\%$
- 다시 검사했더니 또 양성이었다. 이럴 경우 정말 암에 걸렸을 확률은?
- $p(H|E) = \frac{0.99*0.09}{0.09*0.99+0.91*0.01} = 91\%$
- posterior를 통해 prior를 업데이트함!

Background

A. Bayesian and DL





- Training을 통해 weigh가 결정됨
- 학습된 weigh가 고정되고 따라서 새로운 입력에 대한 예측값도 고정됨
- 이는 training data를 신뢰하여 우도를 최대화 하였다고 봄

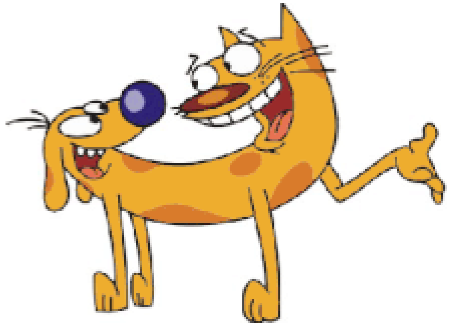
$$P(model|data) = \frac{P(data|model)P(model)}{P(data)}$$

- 학습을 통해 weight를 고정하지 않고 weight의 분포를 얻음
- dropout, 베이지안 vs 앙상블

Background

B. Calibration on ML

$\boldsymbol{x}$



DNNs that classifies
only dogs and cats

90% 개 10% 고양이

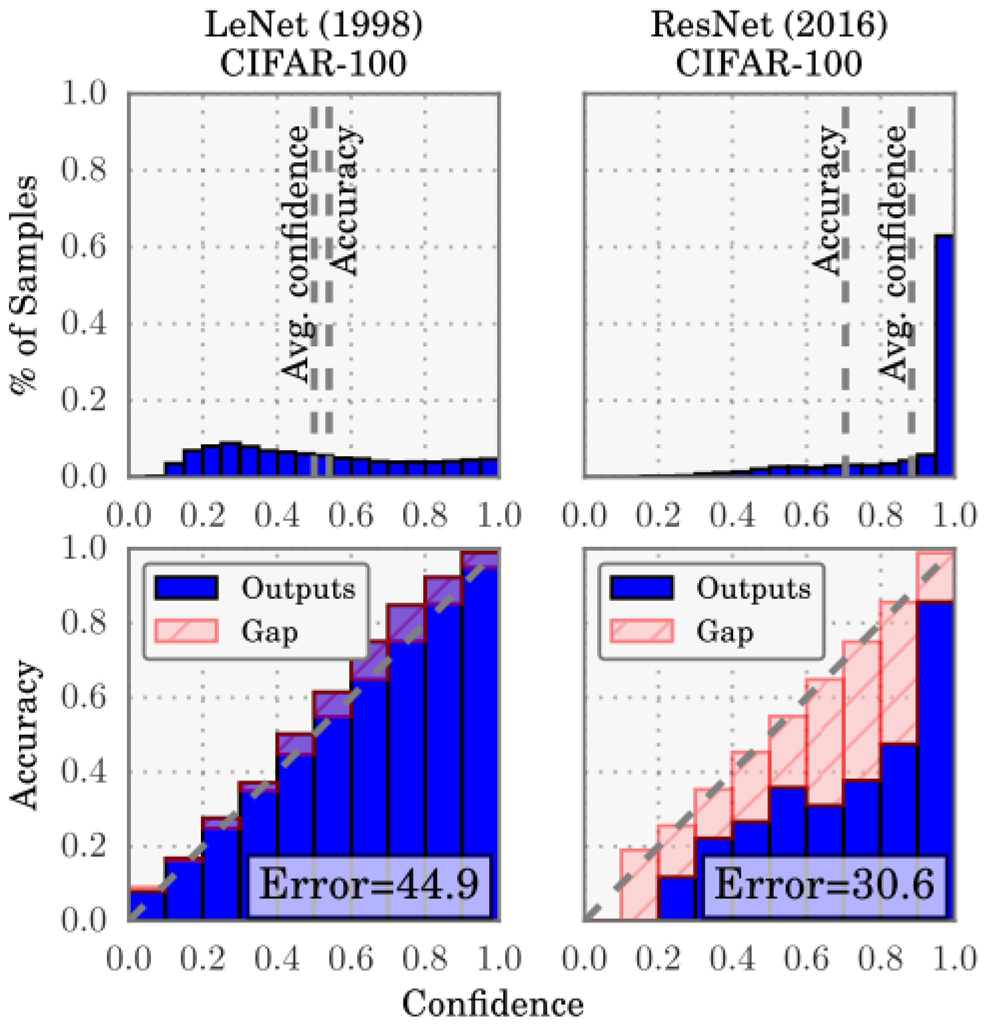
Overconfident

“모름”
or
낮은 확률로 " 개”

Well-calibrated

Background

B. Calibration on ML



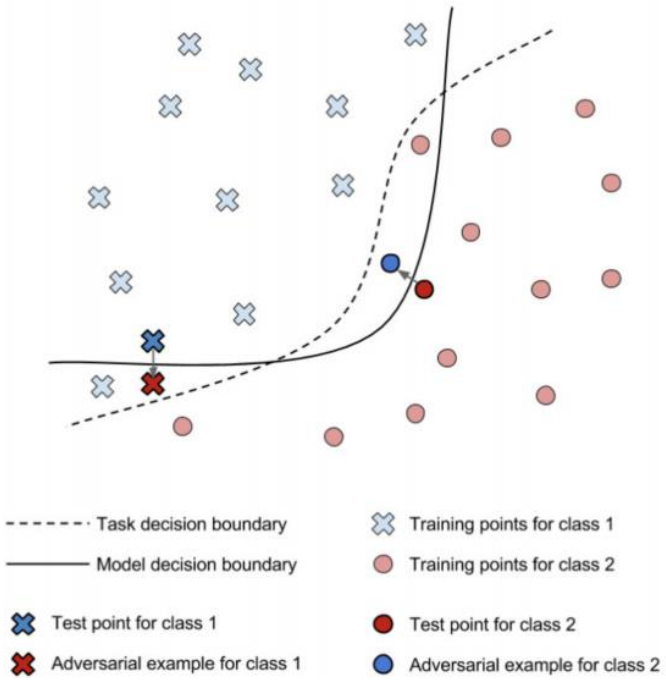
$$(1)\ P(\hat{Y} = y | \hat{P} = p) = p$$

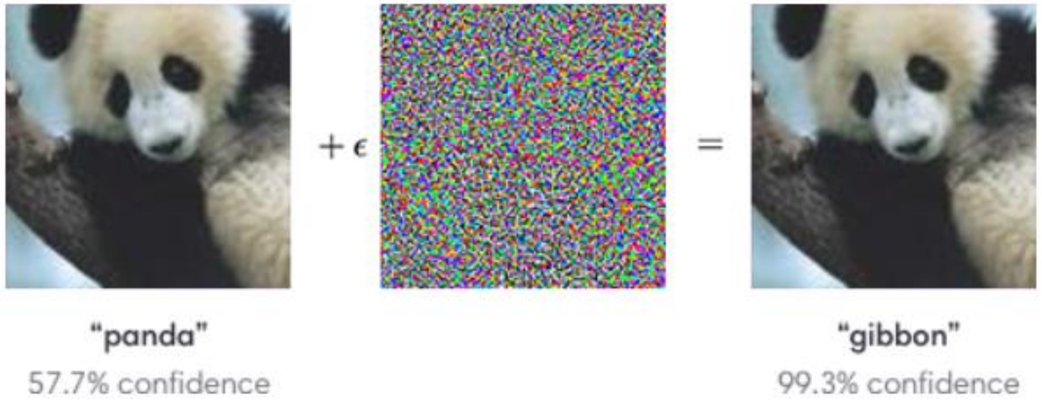
$$(2)\ \mathit{Calibration\ Error} :\ E_{\hat{P}}[P(\hat{Y} = y | \hat{P} = p) - p]$$

$$(3)\ ECE :\ \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$$

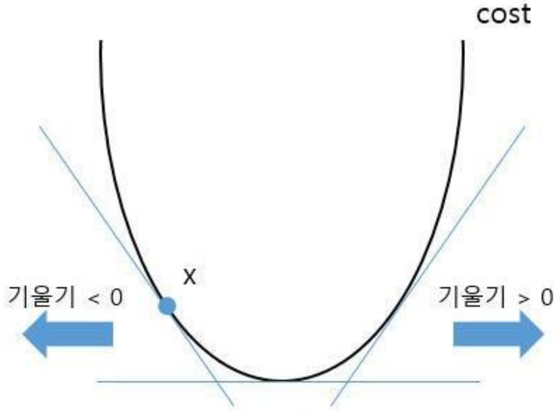
Background

C. Adversarial Training

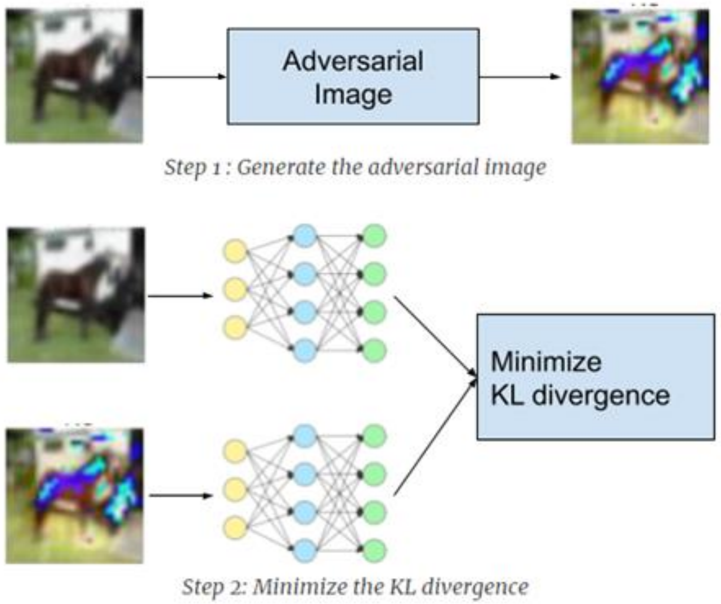




① $FGSM$: $\tilde{x} = x + \epsilon sign(\nabla_x l(\theta, x, y))$



② VAT : $\Delta x = \arg max_{\Delta x} KL(p(y|x)||p(y|x + \Delta x))$



Deep Ensemble

A. Proper scoring rules

- Uncertainty를 측정하기위한 방법으로 제시
- Well-calibrate된 경우에 더 좋은(better) 점수를 부여
- NNs의 경우 loss $L(\theta) = -S(p_\theta, q)$를 최소화하는 방법이 쓰임

Classification : Brier score

$$\mathcal{L}(\theta) = -S(p_\theta, (y, \mathbf{x})) = K^{-1} \sum_{k=1}^{K} (\delta_{k=y} - p_\theta \ (y, \mathbf{x}))^2$$

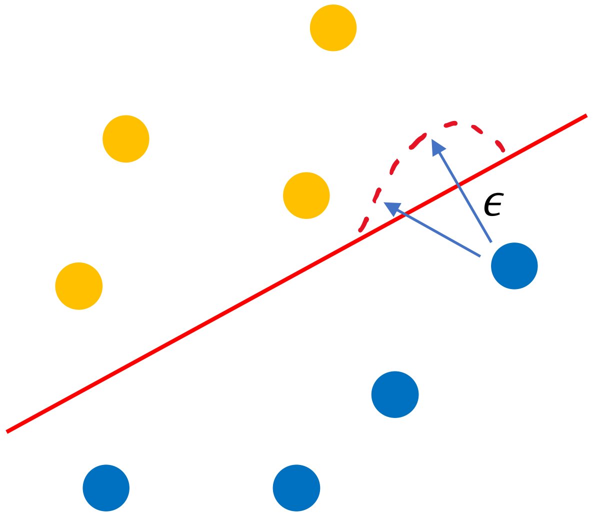
Regression : NLL

$$-\log p_\theta(y_n|\mathbf{x}_n) = \frac{\log \sigma_\theta^2(\mathbf{x})}{2} + \frac{\left(y - \mu_\theta(\mathbf{x})\right)^2}{2\sigma_\theta^2(\mathbf{x})} + \text{constant}.$$

Deep Ensemble

B. Adversarial training

- Adversarial training은 classifier의 robustness를 향상시키는 것으로 알려져있음
- FGSM과 VAT를 제시하며 실험 setting에 따라서 다양한 방법을 시도해 볼 수 있음을 제시



Deep Ensemble

C. Ensembles

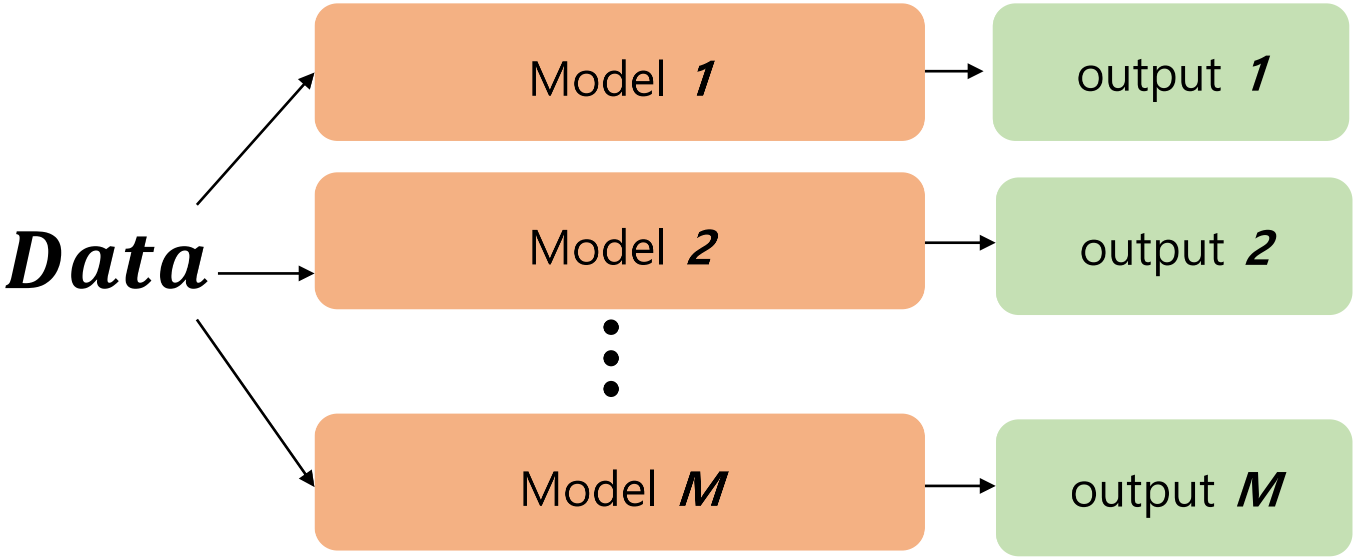
Parallel Ensemble (i.e. random forest)

① For input

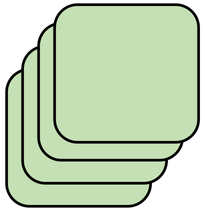
$$\lim_{N\to\infty}\left\{1-\left(1-\frac{1}{N}\right)^{N}\right\} = 1-\lim_{N\to\infty}\left(\frac{N}{N-1}\right)^{-N}$$

$$= 1-\lim_{N\to\infty}\left(1+\frac{1}{N-1}\right)^{-N} = 1-\frac{1}{e} \approx 0.632$$

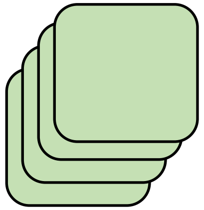
$$\because \lim_{N\to\infty}\left(1+\frac{1}{x}\right)^{x} = e$$



② For output



Classification : Majority voting



Regression : Averaging

Deep Ensemble

D. Algorithms

Algorithm 1 Pseudocode of the training procedure for our method

1: ▷ *Let each neural network parametrize a distribution over the outputs, i.e.* $p_\theta(y|\mathbf{x})$. *Use a proper scoring rule as the training criterion* $\ell(\theta, \mathbf{x}, y)$. *Recommended default values are* $M = 5$ *and* $\epsilon = 1\%$ *of the input range of the corresponding dimension (e.g 2.55 if input range is [0,255]).*
2: Initialize $\theta_1, \theta_2, \ldots, \theta_M$ randomly
3: **for** $m = 1 : M$ **do** ▷ *train networks independently in parallel*
4: Sample data point n_m randomly for each net ▷ *single* n_m *for clarity, minibatch in practice*
5: Generate adversarial example using $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \epsilon\ \mathrm{sign}\big(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})\big)$
6: Minimize $\ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m}) + \ell(\theta_m, \mathbf{x}'_{n_m}, y_{n_m})$ w.r.t. θ_m ▷ *adversarial training (optional)*

- Ensembles as a uniformly-weighted model

① For classification

$$p(y|\mathbf{x}) = M^{-1} \textstyle\sum_{m=1}^{M} p_{\theta_m}(y|\mathbf{x}, \theta_m)$$

② For Regression

$$M^{-1} \sum \mathcal{N}(\mu_{\theta_m}(\mathbf{x}), \sigma^2_{\theta_m}(\mathbf{x}))$$

$$\mu_*(x) = M^{-1} \sum_m \mu_{\theta_m}(x)$$

$$\sigma^2_* = M^{-1} \sum_m (\sigma^2_{\theta_m}(x) + \mu^2_{\theta_m}(x)) - \mu^2_*(x)$$

Experiment

A. Setup and toy example

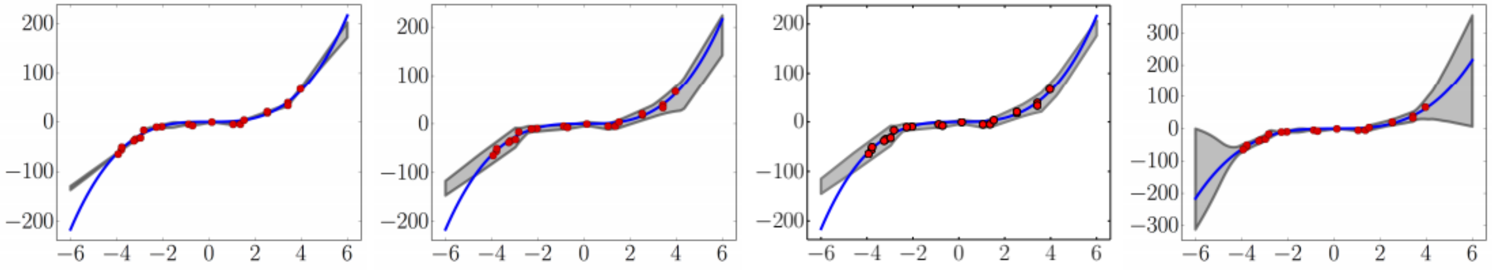
① Experimental setup

$$BS = K^{-1}\sum_{k=1}^{k}(t_k^* - p(y=k|\mathbf{x}^*))$$

(where, $t_k^* = 1 \; if \; k = y^*, and \; 0 \; o.w.$)

Batch_size	100
Optimizer	Adam
lr	0.1
▪ Default torch weights ▪ $\epsilon = 0.01$, FGSM	

② Toy Example



$$y = x^3 + \epsilon \, , where \; \epsilon \sim \mathcal{N}(0, 3^2)$$

❶ 5 networks trained using MSE ❷ NLL using single network
❸ ②+Adversarial Training ❹ NLL+Ensemble 5 networks

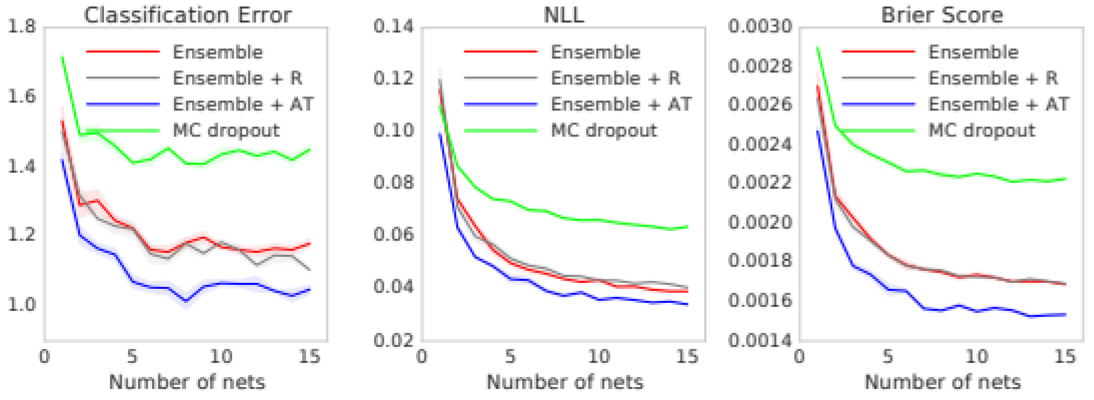
I. Scoring rule NLL 이 uncertainty prediction에 적합함

II. Ensemble이 training data에서 먼 곳을 예측할 때도 성능의 향상을 보임

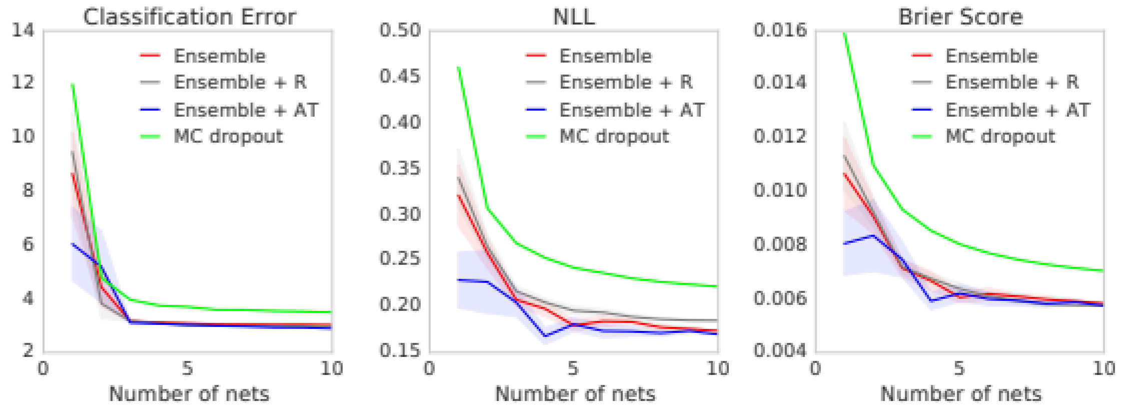
Experiment

B. Regression and Classification

Datasets	RMSE PBP	RMSE MC-dropout	RMSE Deep Ensembles	NLL PBP	NLL MC-dropout	NLL Deep Ensembles
Boston housing	**3.01 ± 0.18**	**2.97 ± 0.85**	**3.28 ± 1.00**	**2.57 ± 0.09**	**2.46 ± 0.25**	**2.41 ± 0.25**
Concrete	**5.67 ± 0.09**	**5.23 ± 0.53**	**6.03 ± 0.58**	**3.16 ± 0.02**	**3.04 ± 0.09**	**3.06 ± 0.18**
Energy	**1.80 ± 0.05**	**1.66 ± 0.19**	**2.09 ± 0.29**	2.04 ± 0.02	1.99 ± 0.09	**1.38 ± 0.22**
Kin8nm	0.10 ± 0.00	0.10 ± 0.00	**0.09 ± 0.00**	-0.90 ± 0.01	-0.95 ± 0.03	**-1.20 ± 0.02**
Naval propulsion plant	0.01 ± 0.00	0.01 ± 0.00	**0.00 ± 0.00**	-3.73 ± 0.01	-3.80 ± 0.05	**-5.63 ± 0.05**
Power plant	**4.12 ± 0.03**	**4.02 ± 0.18**	**4.11 ± 0.17**	2.84 ± 0.01	**2.80 ± 0.05**	**2.79 ± 0.04**
Protein	4.73 ± 0.01	**4.36 ± 0.04**	4.71 ± 0.06	2.97 ± 0.00	2.89 ± 0.01	**2.83 ± 0.02**
Wine	**0.64 ± 0.01**	**0.62 ± 0.04**	**0.64 ± 0.04**	0.97 ± 0.01	**0.93 ± 0.06**	**0.94 ± 0.12**
Yacht	**1.02 ± 0.05**	**1.11 ± 0.38**	**1.58 ± 0.48**	1.63 ± 0.02	1.55 ± 0.12	**1.18 ± 0.21**
Year Prediction MSD	8.88 ± NA	**8.85 ± NA**	8.89 ± NA	3.60 ± NA	3.59 ± NA	**3.35 ± NA**



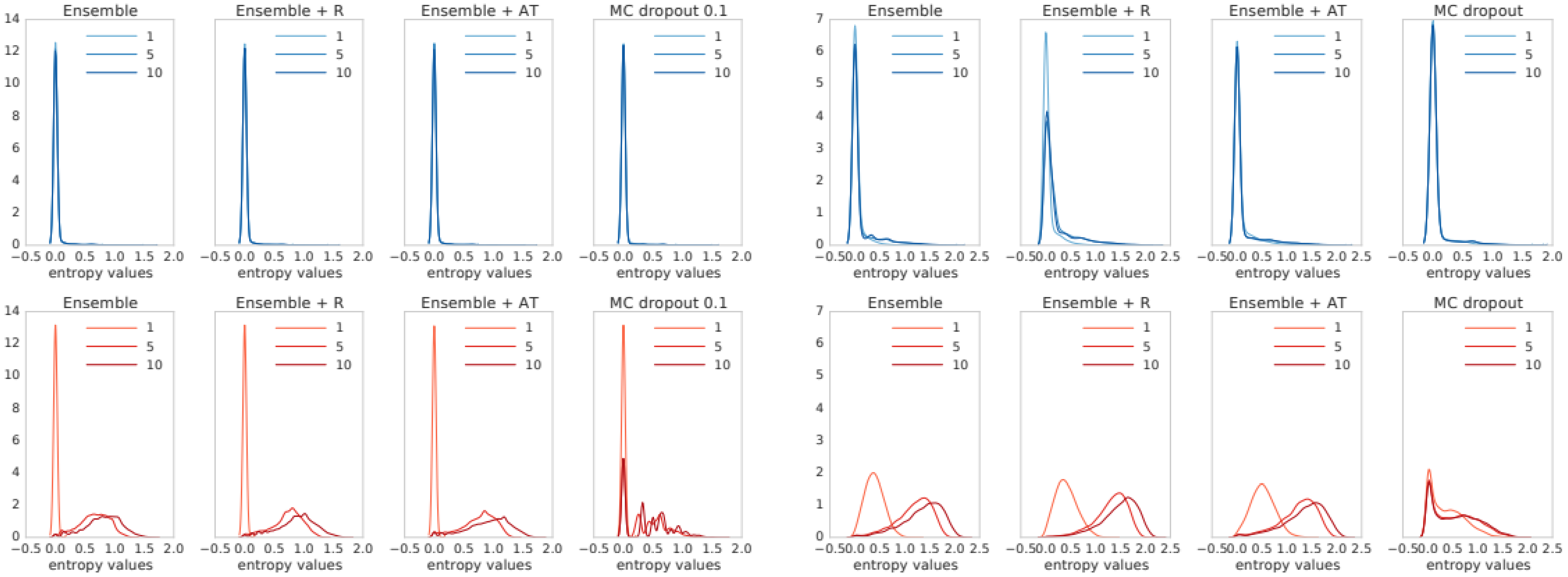
(a) MNIST dataset using 3-layer MLP



(b) SVHN using VGG-style convnet

Experiment

C. Uncertainty evaluation



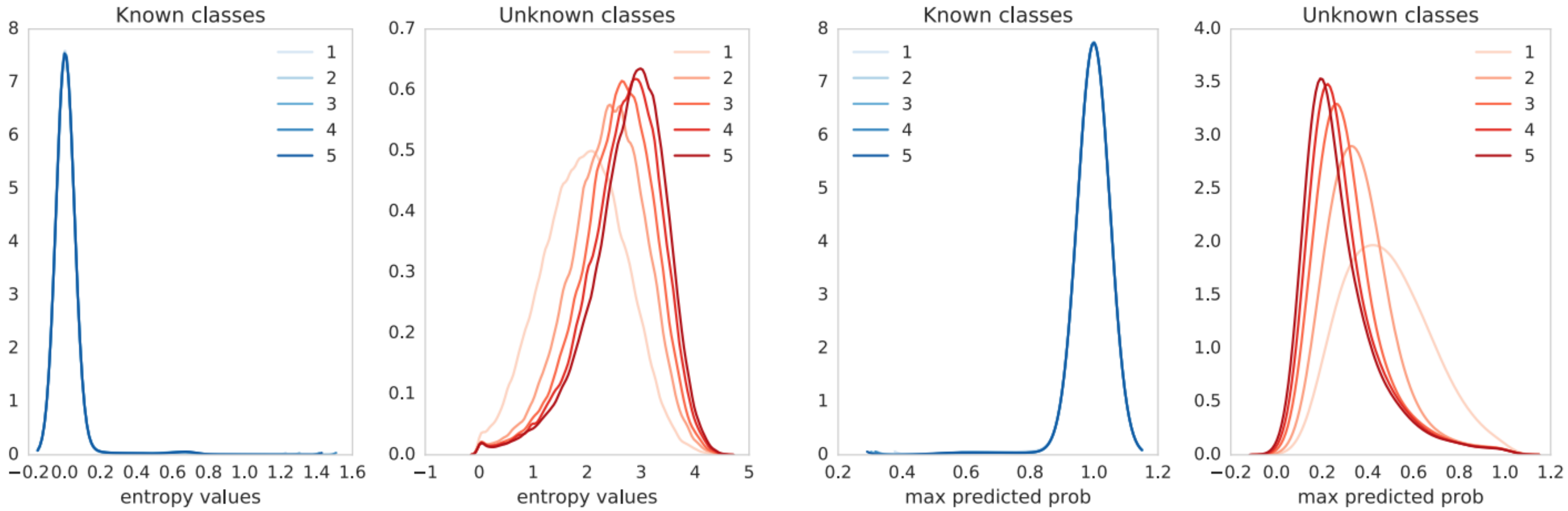
(a) MNIST-NotMNIST

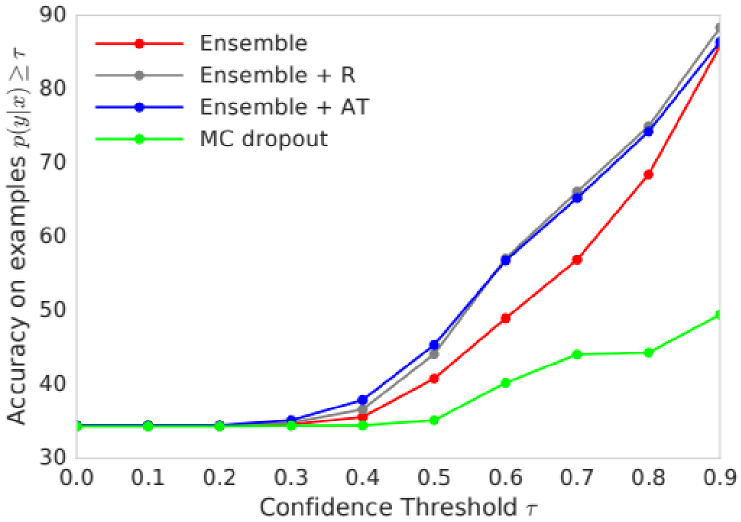
(b) SVHN-CIFAR10

Experiment

D. Accuracy as a function of confidence

M	Top-1 error %	Top-5 error %	NLL	Brier Score $\times 10^{-3}$
1	22.166	6.129	0.959	0.317
2	20.462	5.274	0.867	0.294
3	19.709	4.955	0.836	0.286
4	19.334	4.723	0.818	0.282
5	19.104	4.637	0.809	0.280
6	18.986	4.532	0.803	0.278
7	18.860	4.485	0.797	0.277
8	18.771	4.430	0.794	0.276
9	18.728	4.373	0.791	0.276
10	18.675	4.364	0.789	0.275





Reference

[1] On Calibration of Modern Neural Networks

[2] Explaining and Harnessing Adversarial Examples

[3] Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

THANK

YOU