Simple and scalable predictive uncertainty estimation using deep ensembles

양민서

Open Seminar(2024.07.10)

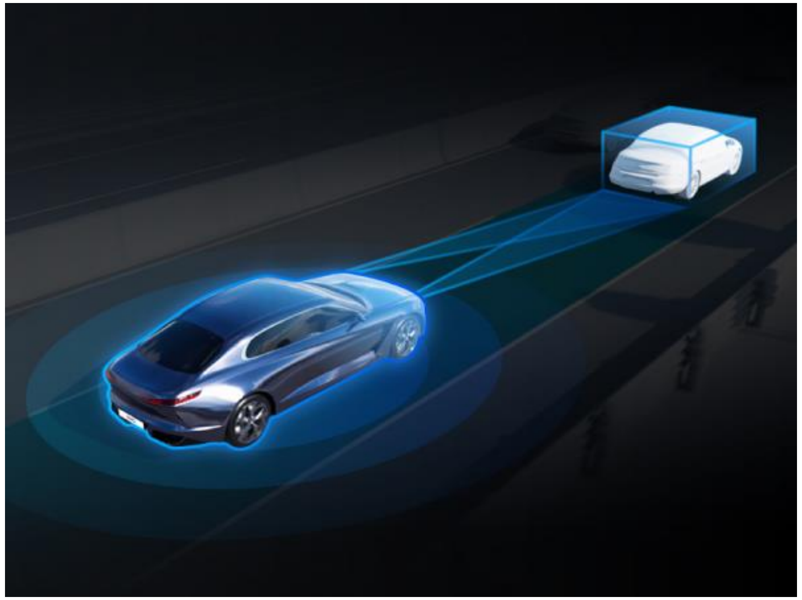
School of Intelligent Mechatronics Engineering,
Sejong University

발표자 소개

- **양민서 (Minseo Yang)**
  - 세종대학교 AI로봇학과 재학
  - Machine Intelligence and Networking Lab.
    (이현석 교수님)
- **Contact**
  - Mail : 22012002@sju.ac.kr

목차

Introduction



“overconfidence”

<출처1: news1.news>, <출처2: 현대차>

Introduction

Neural Network

Dog (p=0.95)

Cat (p=0.92)

Introduction

Neural Network

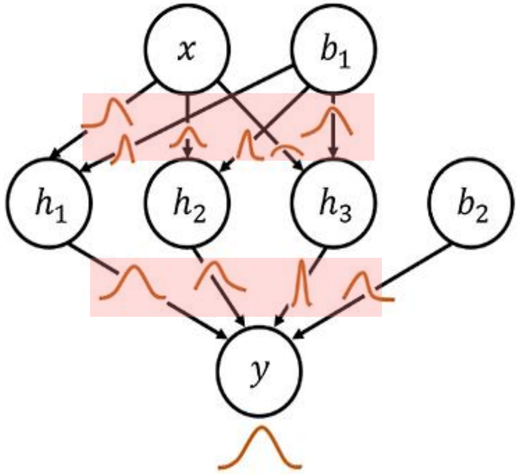
Lion (p=0.8)

Introduction

- **Bayesian NN**

  - Uncertainty를 표현하는 Bayesian 기반 모델들이 존재
  - 도출된 예측 값의 분산정보를 활용하여 불확실성을 정량화

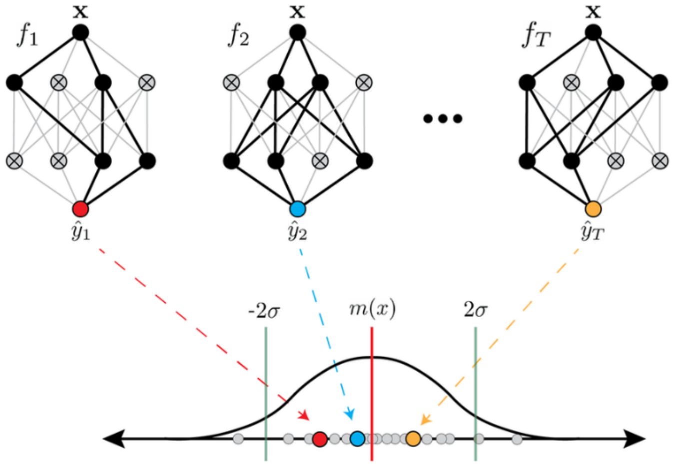
$$\underbrace{P(\theta|X,Y)}_{\text{posterior}} = \frac{\overbrace{P(Y|X,\theta)}^{\text{Likelihood}}\overbrace{P(\theta)}^{\text{prior}}}{\underbrace{P(Y|X)}_{\text{Evidence}}}$$



➢ 훈련 절차에 상당한 수정이 필요하고 계산 비용이 많이 든다는 단점이 존재

- **MC-dropout**

  - 테스트 시점에 드롭 아웃을 사용하여 예측 불확실성을 추정



- **Deep ensemble**
  - 앙상블을 통해 불확실성을 추정하는 방법을 제안
  - 구현 단순 & 병렬 계산에 적합
  - 하이퍼파라미터 조정이 거의 필요하지 않음
  - 고품질의 예측 불확실성 추정치 산출

Deep Ensembles

- **Problem setup**
  - N개의 i.i.d 로 구성된 데이터셋 $D = \{x_n, y_n\}_{n=1}^{N}$ 가정
  - $y \in R$, $y \in \{1, \cdots, K\}$
  - Input feature x가 주어지면, 레이블에 대한 확률적 예측 분포 $p_\theta(y|x)$를 모델링

- **High-level summary**
  1. Proper scoring rule을 훈련 기준으로 사용
  2. Adversarial Training을 사용하여 예측 분포 평탄화
  3. 앙상블 학습

• Proper scoring rules

- scoring rule은 예측 불확실성 품질을 추정하는 함수
- Proper scoring rules는 잘 예측 된 확률 분포에 대해 가장 높은 기대 값을 받을 수 있도록 설계된 함수들을 정의

Scoring rule function = $S(p_\theta, (y, x))$

- 특정샘플 y|x 에 대한 예측 분포 $p_\theta(y|x)$의 품질

Proper scoring rule : $S(p_\theta, q) \leq S(q, q)$ 조건 만족 시

- Expected scoring rule = $S(p_\theta, q) = \int q(y, x)\, S(p_\theta, (y, x))dydx$
- (with equality if and only if $p_\theta(y|x) = q(y|x)$)

- $\mathcal{L}(\theta) = -S(p_\theta, q)$ 을 최소화하여 NN 학습

Deep Ensembles

- **Training criterion**

  - Maximizing likelihood : $S(p_\theta, (\mathrm{y}|\mathrm{x})) = \log q_\theta(y|x)$
  - Gibbs inequality. $D_{\mathrm{KL}}(P\|Q) \equiv \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \geq 0.$
  - $S(p_\theta, \mathrm{q}) = \mathbb{E}_{q(x)} q(y|x) log p_\theta(y|x) \leq \mathbb{E}_{q(x)} q(y|x) log q(y|x)$

$$D_{KL}(q(Y|X)||p_\theta(Y|X) = \mathbb{E}_q\left[log \frac{q(Y|X)}{p_\theta(Y|X)}\right] \geq 0$$

$$= \mathbb{E}_q[log\mathrm{q}(\mathrm{Y}|\mathrm{X})] - \mathbb{E}_q[log p_\theta(Y|X) \geq 0$$

$$\mathbb{E}_q[log\mathrm{q}(\mathrm{Y}|\mathrm{X})] \geq \mathbb{E}_q[log p_\theta(Y|X)$$

$$\mathbb{E}_q[q(Y|X)log\mathrm{q}(\mathrm{Y}|\mathrm{X})] \geq \mathbb{E}_q[q(Y|X)log p_\theta(Y|X)$$

Regression(NLL)

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

Classification(Brier score)

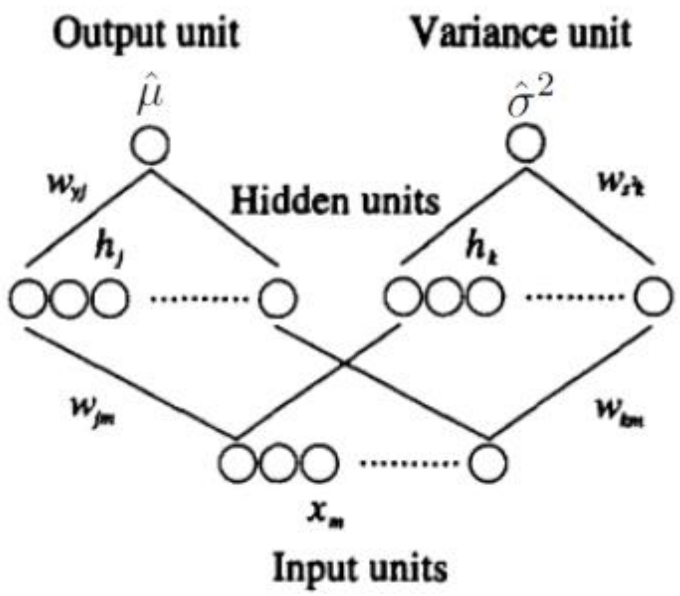
$$K^{-1}\sum_{k=1}^{K}(\delta_{k=y} - p_\theta(y = k|x))^2$$

Deep Ensembles

- **Negative Log Likelihood**

  - 데이터가 정규 분포를 따른다고 가정
  - 출력 값으로 평균과 분산을 가지는 네트워크 사용

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



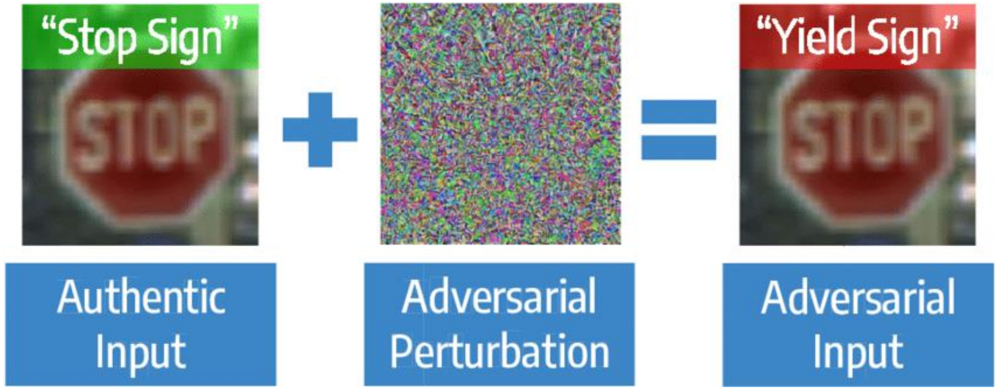
Regression(NLL)

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

- **Adversarial training**
  - Adversarial training을 통해 예측 분포를 평탄화 할 수 있음
  - Fast Gradient Sign Method(FGSM)를 제시
    - 네트워크가 손실을 증가시킬 가능성이 높은 방향으로 추가하여 새로운 훈련 예제 생성

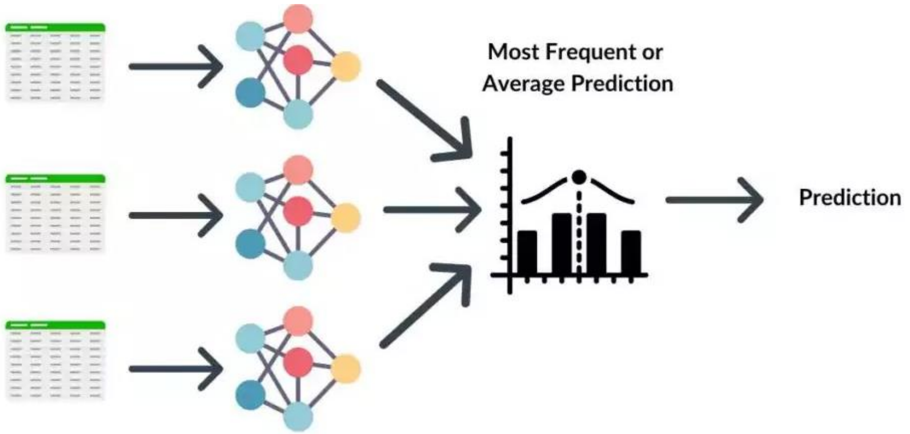
$$\mathrm{x}' = \mathrm{x} + \epsilon sign\left(\nabla_x \ell(\theta, x, y)\right)$$

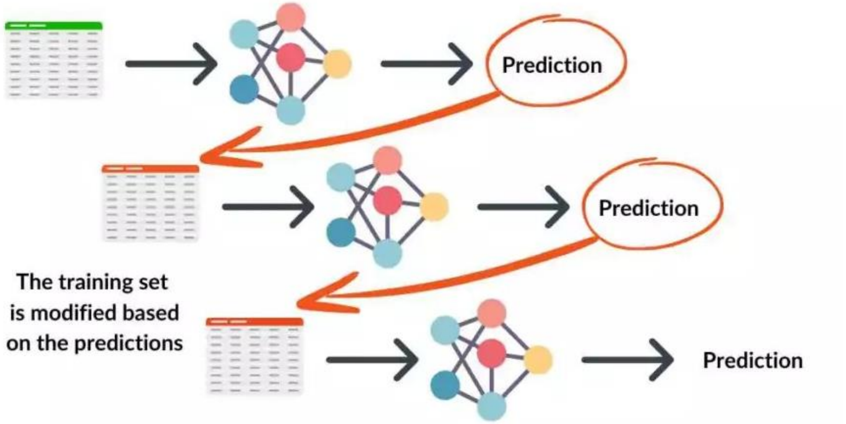
    - Adversarial training 통해 분류기의 견고성을 향상



Deep Ensembles

- **Ensemble**
  - 여러 모델의 결과 값을 활용하는 방식
    - Randomization : 앙상블을 이루는 개별 모델들이 독립적으로 학습
    - Boosting : 앙상블을 이루는 개별 모델들이 순차적으로 학습
  - 본 논문에서는 개별 모델의 정확도와 개별 모델들이 독립적인 오차를 내는 것이 중요



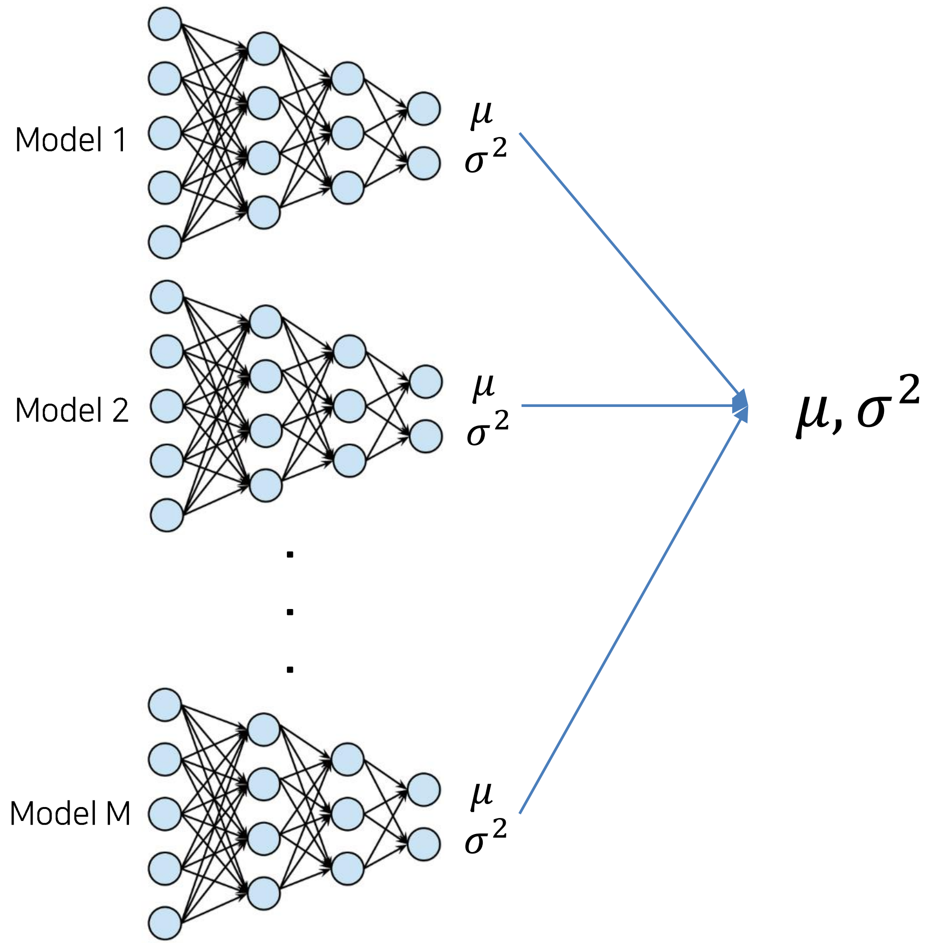


Deep Ensembles

- **Ensemble**
  - 균일 가중치로 적용된 앙상블 모델
  - 각 M개의 모델에 대해 앙상블 수행
  - M개의 예측된 정규 분포의 mixture를 통해 예측 분포 산출

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

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



Deep Ensembles

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 $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \epsilon\, \text{sign}\big(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})\big)$
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)*

1. proper scoring rule을 훈련 기준으로 사용
2. 각 네트워크 파라미터 초기화
3. M개의 네트워크만큼 반복 ▷ 각 네트워크는 독립적으로 병렬 계산 수행
   a. 랜덤하게 각 네트워크 데이터셋 구축
   b. 적대적 예제 생성
   c. Loss를 최소화 하도록 파라미터 학습

Experiment

- **Evaluation metrics and experimental setup**

NLL

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

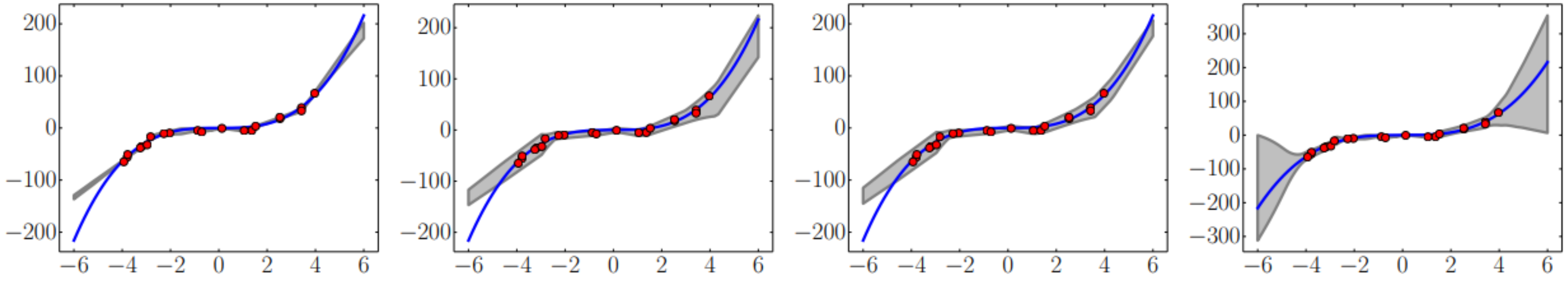
Brier score

$$K^{-1}\sum_{k=1}^{K}(\delta_{k=y} - p_\theta(y = k|x))^2$$

- Regression과 Classification모두 NLL을 사용하여 예측 불확실성을 평가
- Classification에서는 추가로 accuracy와 brier score를 계산
- Regression에서는 추가로 RMSE를 계산

Experiment

- **Toy datasets 불확실성 측정**

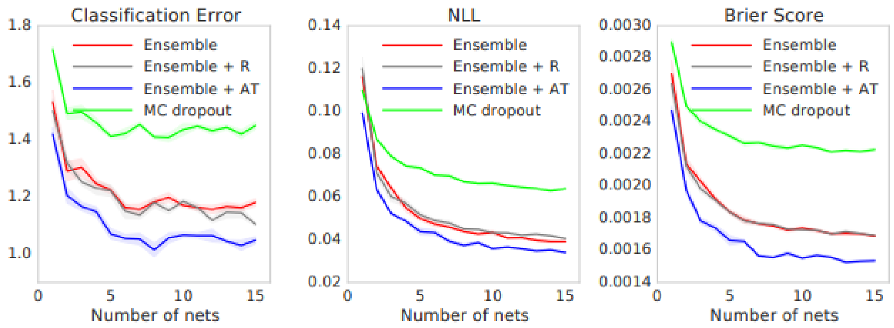


(1) : 5개 모델+MSE (2) : 단일 모델+NLL (3) : 단일 모델+NLL +AT (4) : ensemble

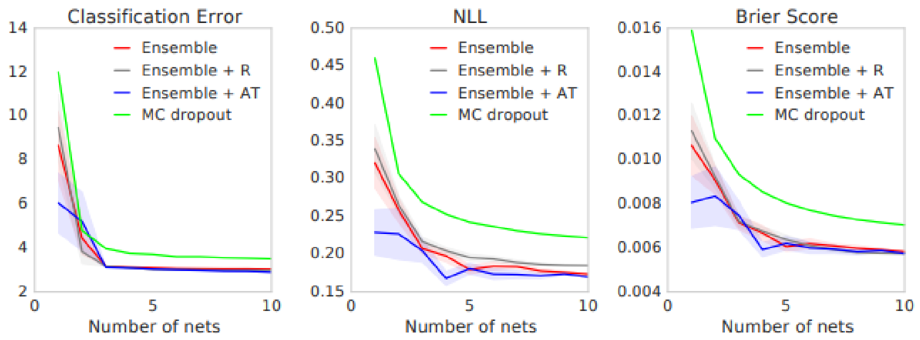
- $y = x^3 + \epsilon \; where \; \epsilon \sim \mathrm{N}(0, 3^2)$
- (2) NLL이 불확실성을 나타내는데 효과적
- (4) ensemble 적용 모델에서 학습데이터에서 멀어질수록 불확실성 증가

Experiment

- **Ensemble 및 Adversarial training 효과 검증 (MNIST, SVHN data)**
  - MNIST의 경우 MLP를 이용, SVHN의 경우 VGG 스타일의 CNN 사용
  - 앙상블 네트워크 수를 늘리면 분류 정확도, NLL의 Brier 점수에서 성능이 향상
  - 적대적 훈련도 성능이 향상하나, M이 증가하면 효과가 떨어짐
    - Class가 잘 분리되어 있으면 적대적 학습이 분류 경계면을 많이 변화시키지 않기 때문



(a) MNIST dataset using 3-layer MLP

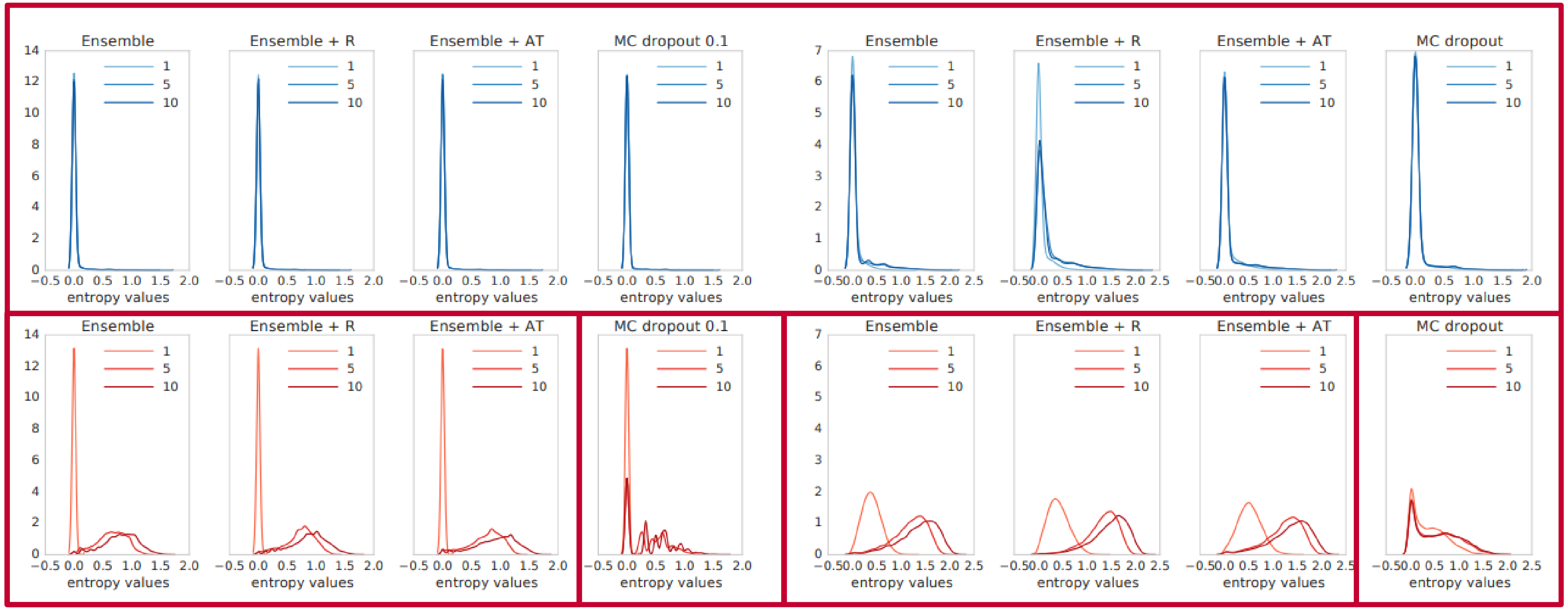


(b) SVHN using VGG-style convnet

- **Out-of-distribution데이터의 불확실성 검증**
  - 불확실한 예제에 대한 높은 신뢰도는 안정적인 예측을 불가능하게 함
  - 불확실성에 대한 지표로 entropy를 확인

entropy

$$H(x) = -\sum_{i=1}^{n} P(x)logP(x)$$



(a) MNIST-NotMNIST (b) SVHN-CIFAR10

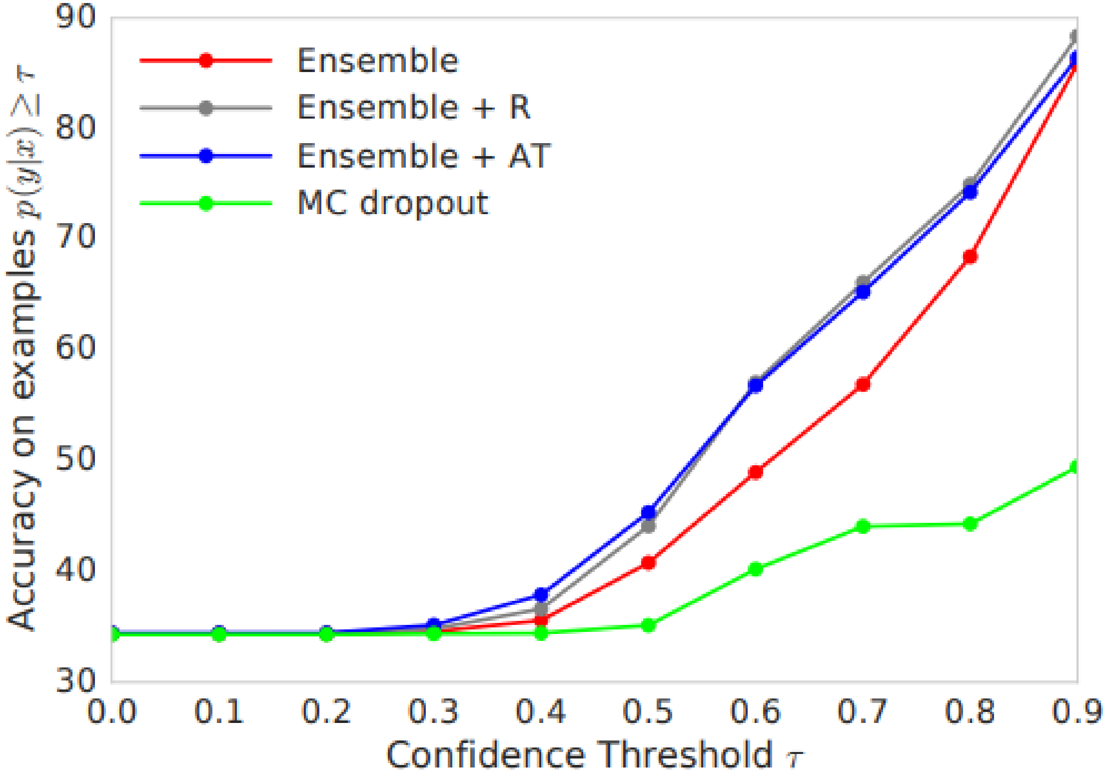
Known classes

unknown classes

Experiment

- **Accuracy as a function of confidence**

✓ p(y = k|x) : 입력 x에 대해 네트워크가 각 클래스 k에 속할 확률
✓ Confidence : $max_k p(y = k \mid x)$
✓ Accuracy : Confidence ≥ τ 인 예제만 필터링 ($0 \le \tau \le 1$)



- Unknown class와 known class를 모두 테스트
- MC-dropout은 높은 신뢰도에서 낮은 정확도를 보임
- Deep ensemble의 경우 예측의 강건성을 보여줌

Conclusion

❖ 예측 불확실성 정량화를 위한 간단하고 확장 가능한 비베이지안 솔루션 제안

❖ Proper scoring rule, Ensemble, Adversarial Training을 통해 데이터에 대한 불확실성을 포착

❖ 향후 네트워크 예측의 다양한 연구 기대

- ✓ De-correlation
- ✓ Stacking(앙상블 가중치 최적화)
- ✓ Distillation(모델 단순화)
- ✓ Implicit Ensemble

감사합니다