Simple & Scalable Predictive Uncertainty Estimation using Deep Ensembles

Balaji Lakshminarayanan

Joint work with Alexander Pritzel & Charles Blundell

DeepMind

Uncertainty Quantification In Deep Learning

- Predict output distribution $p(y|x)$ rather than point estimate
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance

Uncertainty Quantification In Deep Learning

- Predict output distribution $p(y|x)$ rather than point estimate
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance
- What's a "good" uncertainty estimate?
  - Calibration
  - Higher uncertainty on out-of-distribution examples

Uncertainty Quantification In Deep Learning

- Predict output distribution $p(y|x)$ rather than point estimate
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance
- What's a "good" uncertainty estimate?
  - Calibration
  - Higher uncertainty on out-of-distribution examples
- Existing Bayesian solutions: MCMC, VI, MC-Dropout

Uncertainty Quantification In Deep Learning

- Predict output distribution $p(y|x)$ rather than point estimate
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance
- What's a "good" uncertainty estimate?
  - Calibration
  - Higher uncertainty on out-of-distribution examples
- Existing Bayesian solutions: MCMC, VI, MC-Dropout

Our contribution: A simple yet powerful non-Bayesian baseline

A Simple Recipe for Uncertainty Estimation

1. Let neural network parametrize $p_{\boldsymbol{\theta}}(y|\mathbf{x})$. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$

A Simple Recipe for Uncertainty Estimation

1. Let neural network parametrize $p_{\boldsymbol{\theta}}(y|\mathbf{x})$. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Augment with **adversarial training**
   - Encourages $p(y|\mathbf{x})$ to be similar to $p(y|\mathbf{x} + \Delta\mathbf{x})$

A Simple Recipe for Uncertainty Estimation

1. Let neural network parametrize $p_{\boldsymbol{\theta}}(y|\mathbf{x})$. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Augment with **adversarial training**
   - Encourages $p(y|\mathbf{x})$ to be similar to $p(y|\mathbf{x} + \Delta\mathbf{x})$
3. Train an **ensemble of M networks** with random initialization

A Simple Recipe for Uncertainty Estimation

1. Let neural network parametrize $p_{\boldsymbol{\theta}}(y|\mathbf{x})$. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Augment with **adversarial training**
   - Encourages $p(y|\mathbf{x})$ to be similar to $p(y|\mathbf{x} + \Delta\mathbf{x})$
3. Train an **ensemble of M networks** with random initialization
4. Combine predictions at test time

$$p(y|\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} p_{\boldsymbol{\theta}_m}(y|\mathbf{x}, \boldsymbol{\theta}_m)$$

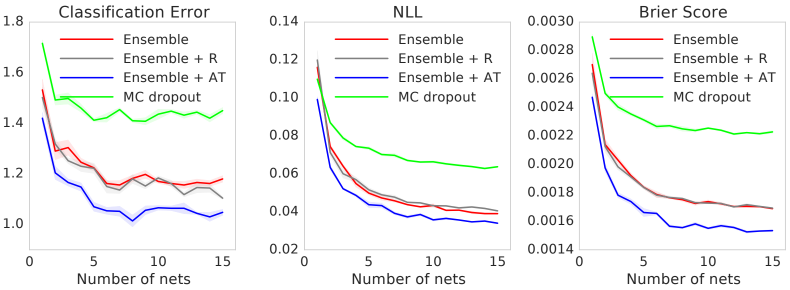
A Simple Recipe for Uncertainty Estimation

1. Let neural network parametrize $p_{\boldsymbol{\theta}}(y|\mathbf{x})$. Use a **proper scoring rule** as training criterion.
   - Classification: cross entropy loss
   - Regression: Gaussian likelihood mean $\mu_{\boldsymbol{\theta}}(\mathbf{x})$ & var $\sigma^2_{\boldsymbol{\theta}}(\mathbf{x})$
2. Augment with **adversarial training**
   - Encourages $p(y|\mathbf{x})$ to be similar to $p(y|\mathbf{x} + \Delta\mathbf{x})$
3. Train an **ensemble of M networks** with random initialization
4. Combine predictions at test time

$$p(y|\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} p_{\boldsymbol{\theta}_m}(y|\mathbf{x}, \boldsymbol{\theta}_m)$$

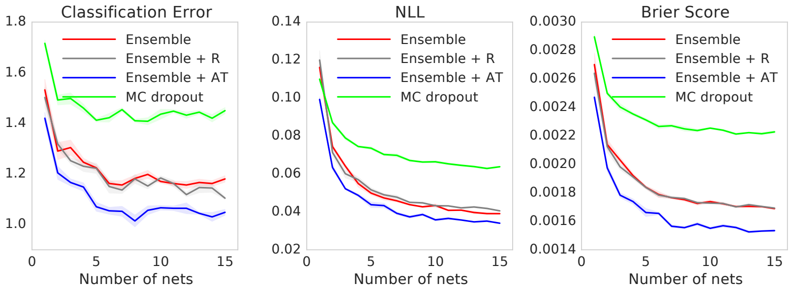
Model combination & not Bayesian Model Averaging

Classification Results on MNIST



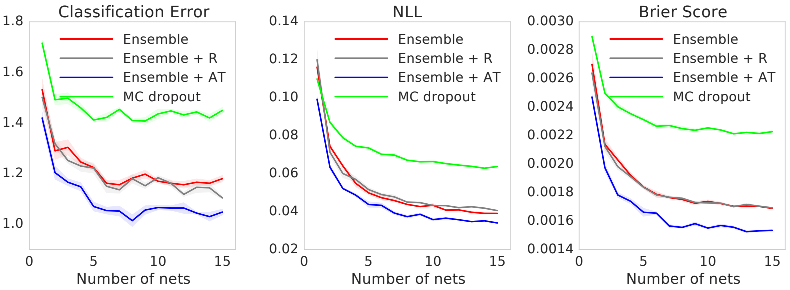
- **Ensembles lead to better predictive uncertainty**

Classification Results on MNIST



- **Ensembles lead to better predictive uncertainty**
- Adversarial training leads to further improvements

Classification Results on MNIST



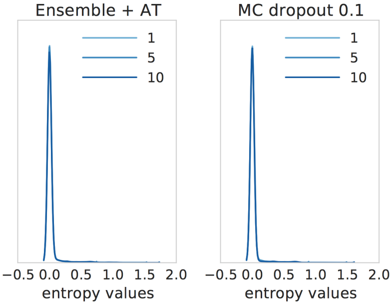
- **Ensembles lead to better predictive uncertainty**
- Adversarial training leads to further improvements
- Similar results on SVHN, ImageNet & regression

Predictive entropy on known & unknown inputs

Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)

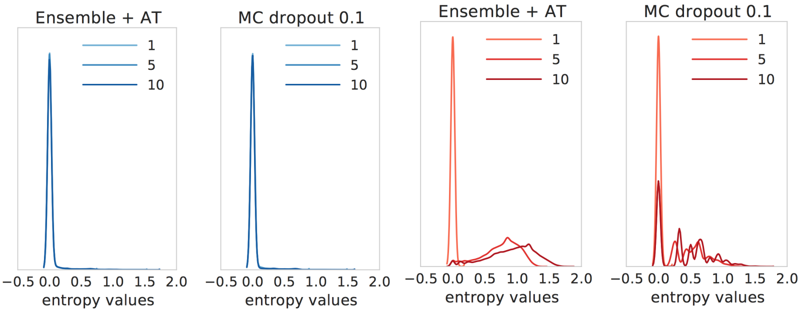
Predictive entropy on known & unknown inputs

Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



Predictive entropy on known & unknown inputs

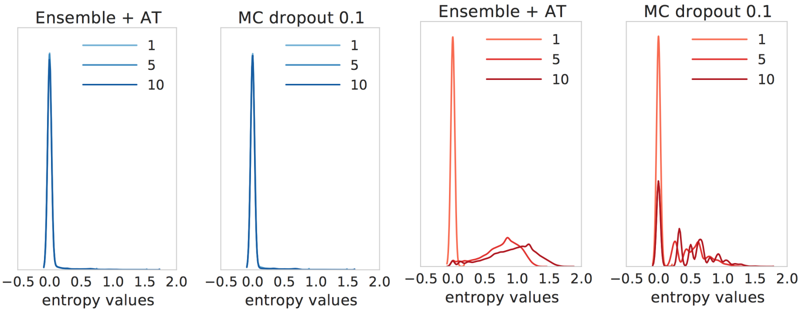
Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



Single network & MC-dropout can produce **overconfident wrong predictions**, whereas **deep ensembles are more robust.**

Predictive entropy on known & unknown inputs

Train: MNIST. **Test**: MNIST + NotMNIST (out-of-distribution)



Single network & MC-dropout can produce **overconfident wrong predictions**, whereas **deep ensembles are more robust.**
Similar results on ImageNet (dogs vs no-dogs).

Poster # 133

Thanks!