Overview
- **Goal:** Predict \( p(y|x) \), for test data \( x \), along with **predictive uncertainty estimate**
- Bayesian neural networks are popular, but have some disadvantages
  - Non-trivial modifications to training code
  - Computationally slow & difficult to scale
  - Quality of Bayesian posterior predictions depends on prior specification (model mis-specification) and posterior approximation. Bayesian approach only translates weight uncertainty to predictive uncertainty.
- **Our contributions:**
  - An experimental protocol to measure quality of predictive uncertainty
  - Calibration measures: NLL/Brier score: Frequentist coverage of subjective forecasts
  - Robustness to dataset shift: Is predictive uncertainty higher on test examples from unknown classes (out-of-distribution)?
  - A simple probabilistic, non-Bayesian baseline that produces surprisingly good results

A Simple Recipe for Uncertainty Estimation
1. Let each neural network parametrize a distribution over the outputs, i.e. \( p(y|x) \).
2. Use a proper scoring rule as training criterion
3. Classification: cross entropy loss
4. Heteroscedastic Regression: net outputs mean \( \mu(x) \) and variance \( \sigma^2(x) \)

\[
\ell(\theta, x, y) = \frac{1}{2} \log \sigma^2(x) + \frac{(y - \mu(x))^2}{2 \sigma^2(x)} + \text{const}.
\]
5. Augment with adversarial training
6. Train an ensemble of \( M \) networks in parallel with random initialization
7. Combine predictions at test time

\[
p(y|x) = \frac{1}{M} \sum_{n=1}^{M} p_n(y|x, \theta_n)
\]

Model combination & not Bayesian Model Averaging

Adversarial Training
- Given an input \( x \) with target \( y \), create new examples \( (x', y') \) using the fast gradient sign method:

\[
x' = x + \epsilon \text{sign} (\nabla_x \ell(\theta, x, y))
\]
- Adversarial training encourages predictive distribution \( p(y|x') \) to be similar to \( p(y|x + \Delta x) \) which encourages local smoothness and improves robustness.
- Uses gradient \( \nabla_x \ell(\theta, x, y) \) instead of random direction in \( \Delta x \in [-1, 1] \)
- Can also use Virtual Adversarial training \( \Delta x = \text{arg max} \Delta x \ell(\theta, p(y|x), p(y|x + \Delta x)) \)

Regression on Toy Dataset
- Blue line: ground truth curve, red dots: observed noisy training data points and gray lines: predicted mean along with three standard deviations
- Left plot corresponds to empirical variance of 5 networks trained using MSE, middle and right plot show the effect of learning variance using a single net and 5 networks respectively
- Empirical variance significantly under-estimates predictive uncertainty

Results on Regression Benchmarks

Uncertainty Evaluation on Known and Unknown Classes (Out-of-Distribution examples)
- Train MLP on standard MNIST training set. Evaluate on standard MNIST test set (known classes) as well as NotMNIST test set (unknown classes) which contains 28 \( \times \) 28 images of alphabets
- Expect higher uncertainty on unknown classes as these inputs are far away from training data
- Measure of uncertainty: predictive entropy

Summary
- Non-Bayesian method, yet produces surprisingly good predictive uncertainty estimates
- Simple to implement. No need for hyperparameter tuning.
- Scalable & well-suited for parallel distributed computation
- Works for different output types (classification, regression) and wide variety of architectures (MLP, CNN)