Sorbonne Université, Computer Science Master
Données, Apprentissage et Connaissances (DAC)
Bayesian Deep Learning

Nicolas Thome
Sorbonne Université
ISIR Lab - Machine Learning Team (MLIA)

Outline

Bayesian Neural Networks for Classification

Applications of Uncertainty in Deep Learning

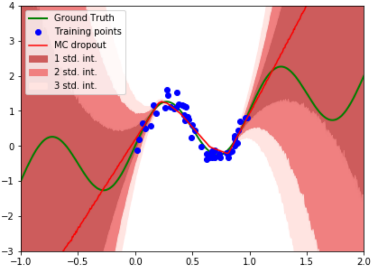
Other Robustness Issues

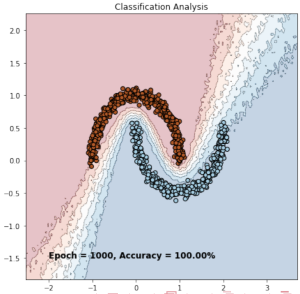
Recap: Bayesian Neural Networks (BNNs)

- Approximate posterior, *e.g.* variational $p(\mathrm{w}|\mathcal{D}) \approx q_{\theta}(\mathrm{w})$
  - $q_{\theta}(\mathrm{w})$ Gaussian or Bernoulli, *e.g.* MCDropout [Gal and Ghahramani, 2016a]
- Predictive distribution: $p(y|\mathrm{x}^*, \mathcal{D}) = \int p(y|\mathrm{x}^*, \mathrm{w})p(\mathrm{w}|\mathcal{D})dw \approx \int p(y|\mathrm{x}^*, \mathrm{w})q_{\theta}(\mathrm{w})dw$
  - $p(y|\mathrm{x}^*, \mathcal{D})$ approximated by MC sampling

$$\int p(y|\mathrm{x}^*, \mathrm{w})q_{\theta}(\mathrm{w})dw \approx \sum_{s=1}^{S} p(y|\mathrm{x}^*, w^s) \quad \mathrm{w}^s \sim q_{\theta}(\mathrm{w})$$

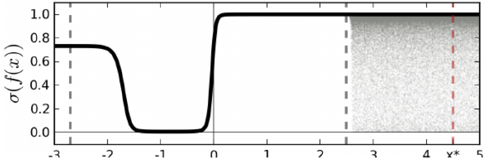
  - Sampling: discrete approximation of the distribution $p(y|\mathrm{x}^*, \mathcal{D})$
    - Regression: y continuous: mean prediction, dispersion $\Leftrightarrow$ uncertainty
    - Classification: y discrete, *e.g.* binary classification $p(y = 1|\mathrm{x}^*, \mathcal{D})$

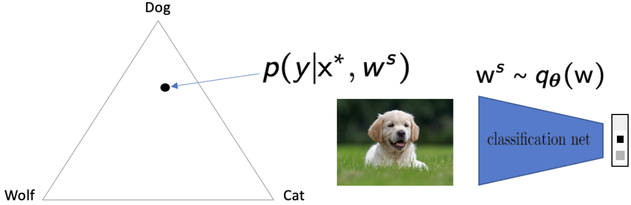




BNNs: Uncertainty in Classification

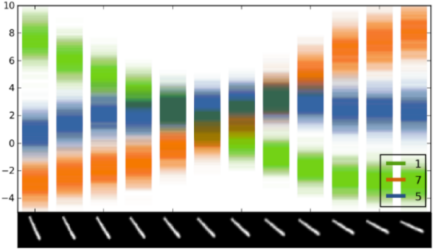
- Only using discrete output: averaging $p(y=1|\mathrm{x}^*, w^s)$ $\mathrm{w}^s \sim q_{\theta}(\mathrm{w})$
  - Loosing the variability of $p(y=1|\mathrm{x}^*, w^s)$ among w^s samples
  - Why not using the continuous output of the model (after softmax) as continuous output?
  - In binary classification: lies in the interval $[0, 1]$
  - For a general K-class classifictaion: K-dimensional simplex



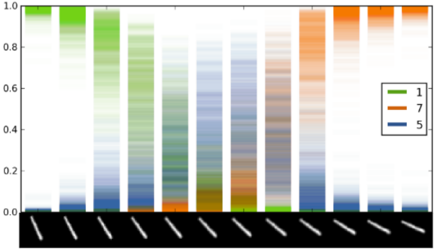


Uncertainty in classification: example with MCDropout

- Use of uncertainty in classification
  - Qualitative expe to illustrate importance of predictive distribution sampling
  - MNIST, LeNet CNN with **dropout only on last fc layer**, dropout probabilities $p = 0.5$, SGD with $LR = 0.01$ updated using momentum 0.9, weight decay $1e^{-06}$, $T = 100$ forward passes



(a) Softmax *input* scatter

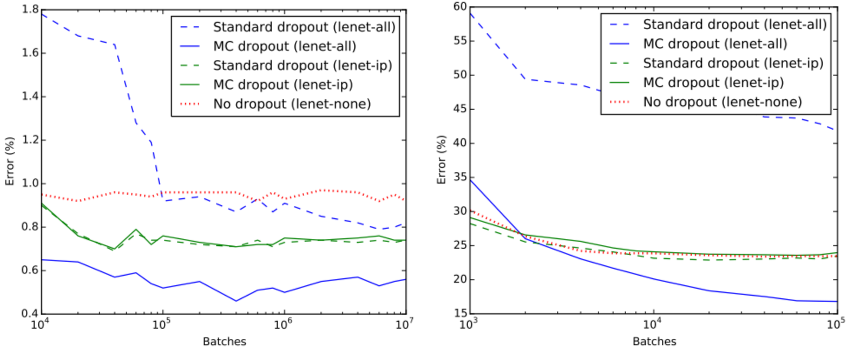


(b) Softmax *output* scatter

Pred: [1 1 1 1 1 5 5 7 7 7 7 7]

MCDropout: side note

- Performances: MC dropout *vs* dropout [Gal and Ghahramani, 2015]
- Test error with LeNet, $T = 50$ forward passes
- Take-home message: dropout only propagates the mean; MC dropout propagates whole distribution



(a) MNIST (b) CIFAR-10

Uncertainty in classification: metrics

1. **Variation ratios**
   - For each stochastic forward pass $t \in \{1; T\}$, compute label from softmax probabilities
   - c^*: most frequent label over the T passes, with frequency $f_{\boldsymbol{x}}^{c^*}$
   - Compute variation-ratio var-ratio$[\boldsymbol{x}] = 1 - \frac{f_{\boldsymbol{x}}^{c^*}}{T}$
     $\Rightarrow$ **Epistemic uncertainty**
2. **Predictive entropy**: captures the average amount of information contained in the predictive distribution.

$$\hat{\mathcal{H}}[y|x, \mathcal{D}_{train}] = -\sum_c \left(\frac{1}{T}\sum_t p(y = c|x, \hat{w}_t)\right) \log\left(\frac{1}{T}\sum_t p(y = c|x, \hat{w}_t)\right)$$

   $\Rightarrow$ **Aleatoric uncertainty**

3. **Mutual information** : maximise the mutual informations are points on which the model is uncertain on average

$$\hat{\mathcal{I}}[y, w|x, \mathcal{D}_{train}] = \hat{\mathcal{H}}[y|x, \mathcal{D}_{train}] - \frac{1}{T}\sum_{c,t} p(y = c|x, \hat{w}_t) \log p(y = c|x, \hat{w}_t)$$

   $\Rightarrow$ **Epistemic uncertainty**

Uncertainty in classification

Let's see three concrete examples:

- **all equal to 1** (*i.e.* probability vectors $\{(1,0),...,(1,0)\}$)
  → *high pred confidence, low model uncertainty*

$$\text{var-ratio} = \hat{\mathcal{H}}[y|x,\mathcal{D}_{train}] = \hat{\mathcal{I}}[y,w|x,\mathcal{D}_{train}] = 0$$

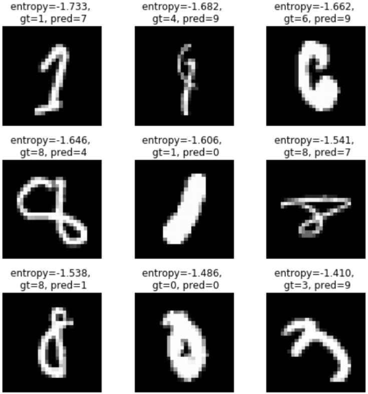
- **half-half**, *i.e.* probability vectors $\{(1,0),(0,1),(0,1),...,(1,0)\}$
  → *low pred confidence, high model uncertainty*

$$\text{var-ratio} = \hat{\mathcal{H}}[y|x,\mathcal{D}_{train}] = \hat{\mathcal{I}}[y,w|x,\mathcal{D}_{train}] = 0.5$$

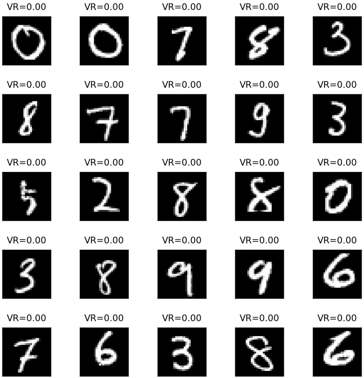
- **all close to 0.5**, *e.g.* $\{(0.51,0.49),(0.51,0.49),...,(0.51,0.49)\}$)
  → *low pred confidence, low model uncertainty*
  - $\hat{\mathcal{H}}[y|x,\mathcal{D}_{train}] = 0.5 \Rightarrow$ **Aleatoric uncertainty**
  - var-ratio $= \hat{\mathcal{I}}[y,w|x,\mathcal{D}_{train}] = 0 \Rightarrow$ **Epistemic uncertainty**

- MC dropout regularization on MNIST
  - Training a convnet with dropout ⇒ improved training performances
  - Improved Training performances with activating dropout at test time
- Use variation ratio to measure uncertainty

Most incertain test set examples



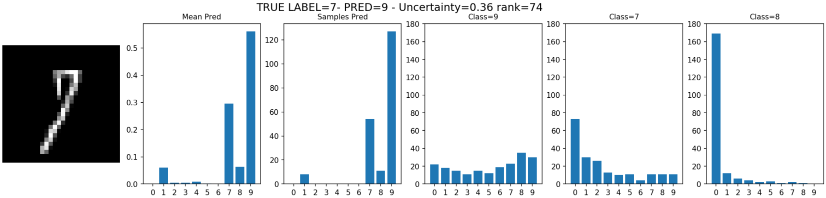
Least incertain test set examples



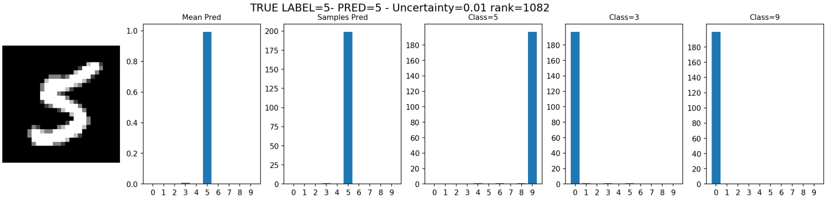
Practical Session: MC dropout on MNIST for classification

- Analyse MC sampling

Incertain example



Confident example

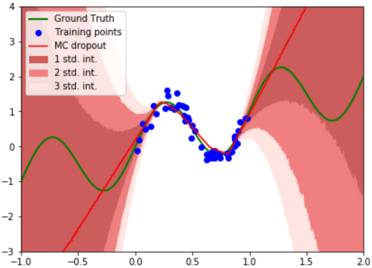


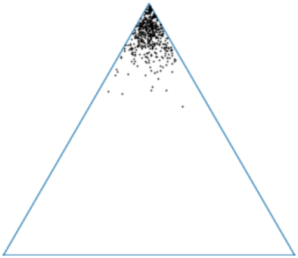
Bayesian Neural Networks & Ensembling

- Approximate posterior, *e.g.* variational $p(\mathrm{w}|\mathcal{D}) \approx q_{\theta}(\mathrm{w})$
  - $q_{\theta}(\mathrm{w})$ Gaussian or Bernoulli (dropout)
- Predictive distribution: $p(y|\mathrm{x}^*, \mathcal{D}) = \int p(y|\mathrm{x}^*, \mathrm{w})p(\mathrm{w}|\mathcal{D})dw \approx \int p(y|\mathrm{x}^*, \mathrm{w})q_{\theta}(\mathrm{w})dw$
  - $p(y|\mathrm{x}^*, \mathcal{D})$ approximated by MC sampling, *e.g.* MC dropout:

$$\int p(y|\mathrm{x}^*, \mathrm{w})q_{\theta}(\mathrm{w})dw \approx \sum_{s=1}^{S} p(y|\mathrm{x}^*, w^s) \quad \mathrm{w}^s \sim q_{\theta}(\mathrm{w})$$

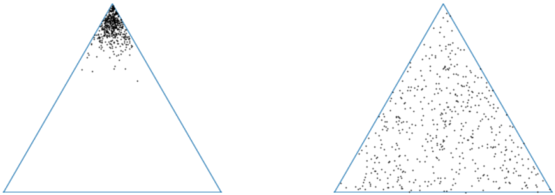
  - Sampling: mean prediction, dispersion wrt mean
  - **Interpretation: Sampling $\Leftrightarrow$ ensemble methods**





Bayesian Neural Networks & Ensembling

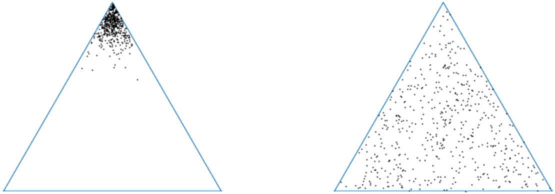
- **Ensembling** [Lakshminarayanan et al., 2017] simple method for predictive distribution approximation and uncertainty estimation
  - Train several models on the same datasets
  - Each model prediction: one sample estimate of predictive distribution
- Empirically, works well (better than MC dropout)



(a) $\{P(y|\boldsymbol{x}^*, \mathcal{M}^{(m)}\}_{m=1}^{M}$ for in-domain $\boldsymbol{x}^*$ (b) $\{P(y|\boldsymbol{x}^*, \mathcal{M}^{(m)}\}_{m=1}^{M}$ for out-of-domain $\boldsymbol{x}^*$

Bayesian Models & Ensembling

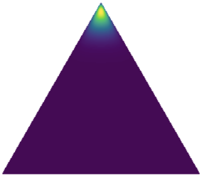
- **Ensembling** [Lakshminarayanan et al., 2017] predictive distribution approximation and uncertainty estimation
- Empirically, works well, BUT:
  - Several predictions at test time ($\sim$ MC dropout)
  - Several models training ($\neq$ MC dropout)



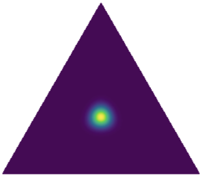
(a) $\{\mathrm{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)}\}_{m=1}^{M}$ for in-domain $\boldsymbol{x}^*$ (b) $\{\mathrm{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)}\}_{m=1}^{M}$ for out-of-domain $\boldsymbol{x}^*$

Dirichlet Prior Networks

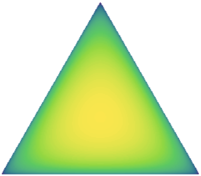
- **Ensembling** [Lakshminarayanan et al., 2017] implicit model of predictive distribution
- **Dirichlet Prior Networks (DPN) [Sensoy et al., 2018, Malinin and Gales, 2018]: explicitly modeling predictive distribution with a Dirichlet distribution**
- x input variable, y = $\{1; K\}$ output class variable
- Random variable z = $\{z_k\}_{k \in \{1;K\}}$: output probability value for classification
  - $z_k \geq 0$, $\sum_{k=1}^{K} z_k = 1 \Rightarrow$ z on the $K-1$ dim simplex
  - Models distribution over probabilities distribution



(a) In-domain input with low uncertainty



(b) In-domain input with high data uncertainty

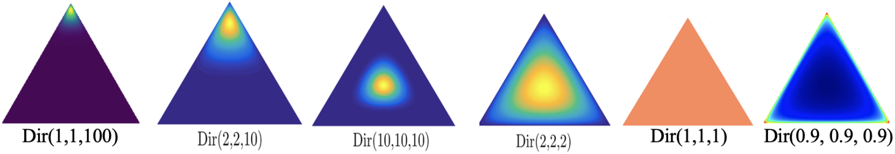


(c) Out-of-distribution input

Dirichlet distribution

$$\text{Dir}(\mathbf{z}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} z_k^{\alpha_k - 1} \quad (1)$$

- $\alpha_k > 0$ **Dirichlet concentration params**, $\alpha_0 = \sum_{k=1}^{K} \alpha_k$, Γ Gamma function
  - $\Gamma(z) = \int_0^{+\infty} x^{z-1} e^{-x} dx$
- α_k: evidence for class K, $\boldsymbol{\alpha}$ **position on simplex** $\Rightarrow$ **aleatoric uncertainty**
- α_0 **total evidence** $\sim$ **distribution spread** $\Rightarrow$ **epistemic uncertainty**



Dice roll example

- N **Dice rolls**, $\mathcal{D} = \{y_i\}_{i \in \{1;N\}}$, $y_i \in \{1, ..., K\}$, $z_k := P(y_k)$
- **Likelihood** $P(\mathcal{D}|\mathrm{z}) = Mul(\mathcal{D}|\mathrm{z}) = \frac{N!}{\prod_k N_k!} \prod_{k=1}^{K} z_k^{N_k}$
  - N_k number of times class k
- **Prior on probabilities:** $P(\mathrm{z}) \sim \mathrm{Dir}(\boldsymbol{\alpha}) \propto \prod_{k=1}^{K} \mathrm{z}_c^{\alpha_k - 1}$, *e.g.* $\boldsymbol{\alpha} = (11...1)^T$
- **Dirichlet conjugate with multinomial** $\Rightarrow$ **posterior** $P(\mathrm{z}|\mathcal{D})$ **Dirichlet**

$$P(\mathrm{z}|\mathcal{D}) \propto P(\mathcal{D}|\mathrm{z})P(\mathrm{z}) \propto \prod_{k=1}^{K} \mathrm{z}_c^{N_k + \alpha_k - 1} \sim \mathrm{Dir}(\alpha_k + N_k)$$

  - α_k pseudo count
  - Increasing N $\Rightarrow$ increasing $N_k + \alpha_k$ evidence, decreasing epistemic uncertainty

Dirichlet Prior Networks (DPN)

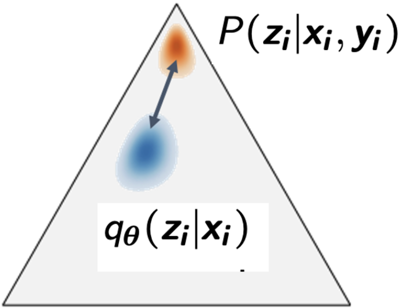
- **Training dataset** $\mathcal{D} = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{i \in \{1;N\}}$, $y_i \in \{1, ..., K\}$, $z_k := P(y_k)$
- **Likelihood** (i^{st} sample): $P(\boldsymbol{y_i}|\boldsymbol{x_i}, \boldsymbol{z_i}) \sim Cat(\boldsymbol{y_i}|\boldsymbol{x_i}, \boldsymbol{z_i}) \propto \prod_{k=1}^{K} z_{k,i}^{\tilde{N}_{k,i}}$
  - $\tilde{N}_{k,i}$ unknown, intuition large (infinite) for true class, low (0) for others
  - Ex: $\tilde{N}_{k,i} := \lambda^{-1}\delta_{k,k_i^*}$, k_i^* GT class, and *e.g.* $\lambda = 10^{-2}$
- **Prior:** Dirichlet as before: $P(\mathrm{z}_i) \sim \mathrm{Dir}(\boldsymbol{\alpha})$, *e.g.* $\boldsymbol{\alpha} = (11...1)^T$
- **Posterior:** $P(\boldsymbol{z_i}|\boldsymbol{x_i}, \boldsymbol{y_i}) \propto P(\boldsymbol{y_i}|\boldsymbol{x_i}, \boldsymbol{z_i})P(\boldsymbol{z_i}) \propto \mathrm{Dir}(\boldsymbol{\beta_i}), \beta_{k,i} = (\alpha_k + \tilde{N}_{k,i})$, *e.g.* $\beta_k = 1 + 100 \cdot \delta_{k,k_i^*}$
- **Training goal:** approximate $P(\boldsymbol{z_i}|\boldsymbol{x_i}, \boldsymbol{y_i}) \approx q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})$ (without $\boldsymbol{y_i}$)
  - $q_{\boldsymbol{\theta}}(\boldsymbol{x_i}) := \mathrm{Dir}(\boldsymbol{\gamma_i})$: outputs $\boldsymbol{\gamma_i}$ Dirichlet concentration params for $\boldsymbol{x_i}$
    - Learn $\boldsymbol{\theta}$ to make $\boldsymbol{\gamma_i}$ as close as possible to target $\boldsymbol{\beta_i}$
    - Ex: $\boldsymbol{\gamma}(\boldsymbol{x}_i, \boldsymbol{\theta}) = e^{f(\boldsymbol{x}_i, \boldsymbol{\theta})}$, $f(\boldsymbol{x}_i, \boldsymbol{\theta})$ K-dim output of a neural network (NN)
- **Inference:** $P(y = c|\boldsymbol{x}^*, \mathcal{D}) = \int p(y = c|\boldsymbol{x}^*, \boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{x}^*, \mathcal{D})d\boldsymbol{z}$ (marginalization)

$$P(y = c|\boldsymbol{x}^*, \mathrm{D}) \approx \int p(y = c|\mathrm{x}^*, \boldsymbol{z})q_\theta(\boldsymbol{z}|\mathrm{x}^*)d\boldsymbol{z}$$

$$P(y = c|\boldsymbol{x}^*, \mathcal{D}) \approx \mathbb{E}_{q_\theta(\mathrm{z}|\boldsymbol{x}^*)}[\boldsymbol{z}_c] = \frac{\gamma_c}{\gamma_0} = \frac{e^{f_c(\boldsymbol{x}^*, \boldsymbol{\theta})}}{\sum_{k=1}^{K} e^{f_k(\boldsymbol{x}^*, \boldsymbol{\theta})}}$$ $\Rightarrow$ **Soft-max prediction!**

Dirichlet Prior Networks (DPN)

- **Training goal:** approximate $P(\boldsymbol{z_i}|\boldsymbol{x_i}, \boldsymbol{y_i}) \approx q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})$
  - $q_{\boldsymbol{\theta}}(\boldsymbol{x_i}) := \text{Dir}(\boldsymbol{\gamma_i})$: outputs $\boldsymbol{\gamma_i}$ Dirichlet concentration params for $\boldsymbol{x_i}$
    - Ex: $\boldsymbol{\gamma}(\mathrm{x}_i, \boldsymbol{\theta}) = e^{f(\mathrm{x}_i, \boldsymbol{\theta})}$, $f(\mathrm{x}_i, \boldsymbol{\theta})$ K-dim output of a neural network (NN)
- Minimize $KL\left[q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})||P(\boldsymbol{z_i}|\boldsymbol{x_i}, \boldsymbol{y_i})\right] = -\int q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i}) \log \frac{P(\boldsymbol{z_i}|\boldsymbol{x_i}, \boldsymbol{y_i})}{q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})} d\boldsymbol{z}$



Dirichlet Prior Networks (DPN)

- **Training goal:** approximate $P(\boldsymbol{z_i}|\boldsymbol{x_i}, \boldsymbol{y_i}) \approx q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})$
  - $q_{\boldsymbol{\theta}}(\boldsymbol{x_i}) := \text{Dir}(\boldsymbol{\gamma_i})$: outputs $\boldsymbol{\gamma_i}$ Dirichlet concentration params for $\boldsymbol{x_i}$
    - Ex: $\boldsymbol{\gamma}(\mathrm{x}_i, \boldsymbol{\theta}) = e^{f(\mathrm{x}_i, \boldsymbol{\theta})}$, $f(\mathrm{x}_i, \boldsymbol{\theta})$ K-dim output of a neural network (NN)
- Minimize $KL\left[q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})||P(\boldsymbol{z_i}|\boldsymbol{x_i}, \boldsymbol{y_i})\right] = -\int q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i}) \log \frac{P(\boldsymbol{z_i}|\boldsymbol{x_i}, \boldsymbol{y_i})}{q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})} d\boldsymbol{z}$
  - Variationnal approximation (VI), deriving Evidence Lower Bound (ELBO) loss:

$$\mathcal{L}_{VI}(\boldsymbol{\theta}) = -\int q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i}) \log \frac{P(\boldsymbol{y_i}|\boldsymbol{x_i}, \boldsymbol{z_i}) P(\boldsymbol{z_i}|\boldsymbol{x_i})}{q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i}) P(\boldsymbol{y_i}|\boldsymbol{x_i})} d\boldsymbol{z}$$

$$\mathcal{L}_{VI}(\boldsymbol{\theta}) = -\int q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i}) \log P(\boldsymbol{y_i}|\boldsymbol{x_i}, \boldsymbol{z_i}) d\boldsymbol{z} + KL\left[q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})||P(\boldsymbol{z_i}|\boldsymbol{x_i})\right] + P(\boldsymbol{y_i}|\boldsymbol{x_i})$$

$$\mathcal{L}_{VI}(\boldsymbol{\theta}) \propto \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})}\Big[-\log p(y_i|\boldsymbol{x_i}, \boldsymbol{z_i})\Big] + KL\Big[q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})||P(\boldsymbol{z_i}|\boldsymbol{x_i})\Big] \tag{2}$$

  - $\mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})}\Big[-\log p(y_i|\boldsymbol{x_i}, \boldsymbol{z_i})\Big] = \lambda^{-1}\Big[\psi(\gamma_0) - \psi(\gamma_y)\Big]^a$, ψ digamma function
    - $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$, asymptotic expansion: $\psi(x) \approx ln(x) - \frac{1}{2x} + O(x^2)$
  - $P(\boldsymbol{z_i}|\boldsymbol{x_i}) = \text{Dir}(1..1) \Rightarrow KL\Big[q_{\boldsymbol{\theta}}(\boldsymbol{z_i}|\boldsymbol{x_i})||P(\boldsymbol{z_i}|\boldsymbol{x_i})\Big]$ differential entropy $\propto \frac{1}{\gamma_0}$, support distribution spread

[a] $\mathbb{E}\left[log(X_i)\right] = \psi(\gamma_i) - \psi(\gamma_0)$, $X \sim Dir(\gamma)$. $\log p(y_i|\boldsymbol{x_i}, \boldsymbol{z_i}) = \lambda^{-1} log(z_y)$, $\lambda^{-1} = \tilde{N}_{y,i}$

Outline

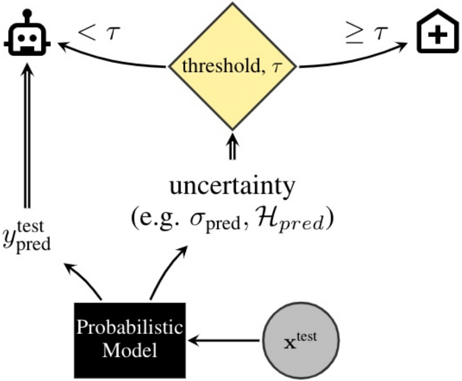
Bayesian Neural Networks for Classification

Applications of Uncertainty in Deep Learning

Other Robustness Issues

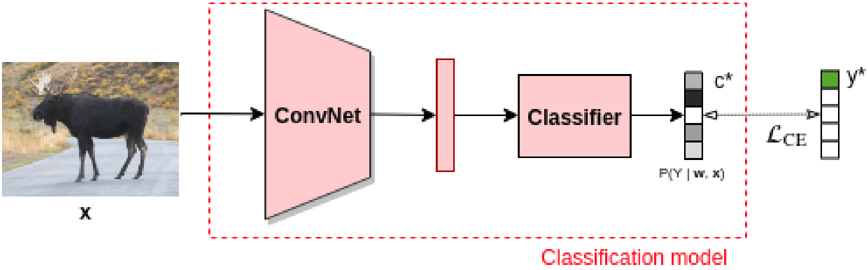
Failure Prediction

- Detecting failure of a system crucial in practice
- Use uncertainty estimate to accept / reject predictions



Failure Prediction: Model Calibration in Deep Learning

- Classification model trained on $\mathcal{D} = \{(\boldsymbol{x}_i, y_i^*)\}_{i=1}^N$



- Model prediction: $\hat{\boldsymbol{y}} = \arg\max_{k \in \mathcal{Y}} p(Y = k | \boldsymbol{w}, \boldsymbol{x})$
- Model confidence $\hat{C}(\boldsymbol{x})$:
  - Simple baseline for deep neural networks: $\boldsymbol{MCP}(\boldsymbol{x}) = \max\limits_{k \in \mathcal{Y}} p(Y = k | \boldsymbol{w}, \boldsymbol{x})$
  - More advanced methods, *e.g.* MC dropout for classification
- **Threshold confidence (uncertainty) estimate to accept / reject predictions**

Model Calibration

- Test **Data**: $\mathcal{D} = (X, Y) = \{(x_1, y_1), ..., (x_N, y_N)\}$
- $(\hat{y}_i, \hat{C}(\boldsymbol{x}_i))$ class prediction and confidence level
- Perfect calibration:

$$p\left(\hat{Y} = Y | \hat{C} = p\right) = p, \quad \forall p \in [0, 1]$$

- Predicted confidence match actual accuracy
  - *e.g. given 100 predictions, each with confidence of 0.8, we expect that 80 should be correctly classified.*
  - Link to thresholding probabilities for failure prediction: non-calibrated probabilities $\Rightarrow$ arbitrary threshold!

Model Calibration in deep learning

Reliability Diagram

M interval bins.

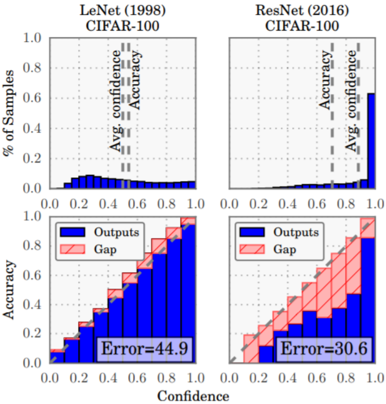
B_m set of samples whose predictions are in $I_m = (\frac{m-1}{M}, \frac{m}{M}]$

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i)$$

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i$$

perfect calibration

$\iff acc(B_m) = conf(B_m) \quad \forall m$



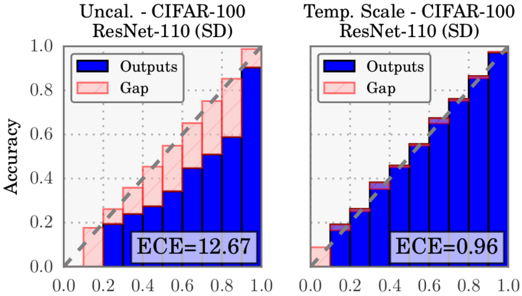
[Guo et al., 2017] **showed that modern neural networks are no longer well-calibrated!**

Model Calibration in deep learning

- Simple solution to over-confident prediction: temperature scaling [Guo et al., 2017]

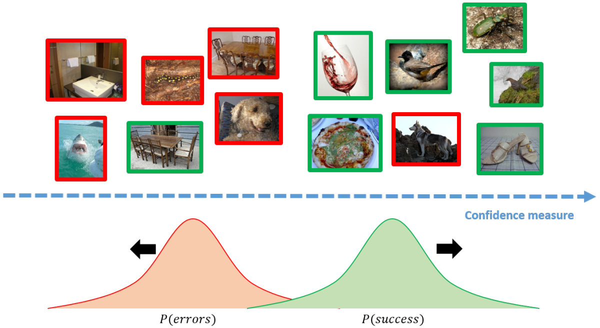
$$P(\hat{y}_k) = \frac{e^{s_k/T}}{\sum\limits_{k'=1}^{K} e^{s_{k'}/T}}$$

- temperature T optimized on val set s.t. $acc(B_m) = conf(B_m)$



Failure Prediction in deep learning

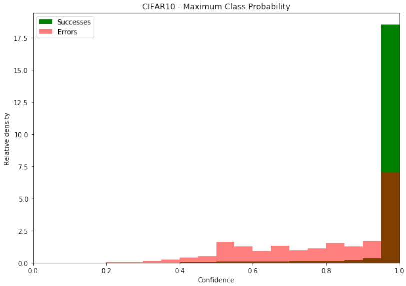
- Confidence estimate $\hat{C}(\boldsymbol{x}_i)$ goal: **distinguish correct from erroneous predictions**

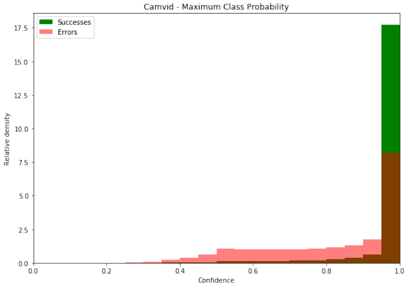


- Sort examples wrt $\hat{C}(\boldsymbol{x}_i)$
  - Evaluate capacity of $\hat{C}$ to assign larger prediction values for correct predictions than for errors

Failure Prediction

- MCP: $\boldsymbol{MCP}(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} p(Y = k | \boldsymbol{w}, \boldsymbol{x})$ unreliable confidence criterion
  - For failure prediction: by design assign largest probability



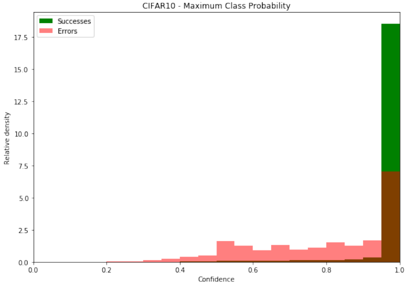


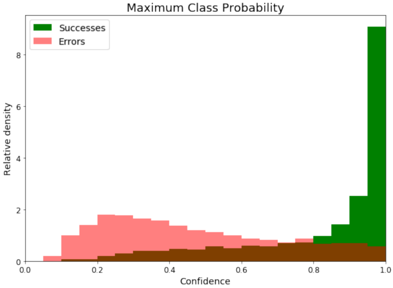
- **overlapping distributions** between successes vs. errors
  $\Rightarrow$ hard to distinguish

MCP, a sub-optimal ranking confidence measure

$$\boldsymbol{MCP}(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} p(Y = k | \boldsymbol{w}, \boldsymbol{x})$$

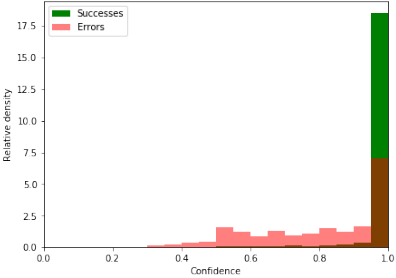
- Overconfident prediction values
  $\Rightarrow$ calibration [Guo et al., 2017]
- BUT: calibration does not solve the overlap problem



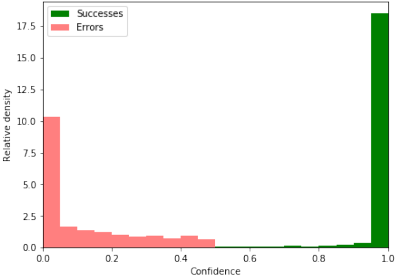


Failure Prediction with TCP [Corbière et al., 2019]

- True Class Probability (TCP): $\boldsymbol{TCP}(\boldsymbol{x}, y^*) = p(Y = y^* | \boldsymbol{w}, \boldsymbol{x})$ unreliable confidence criterion
  - For failure prediction: assign lower probability for errors



(a) Maximum Class Probability

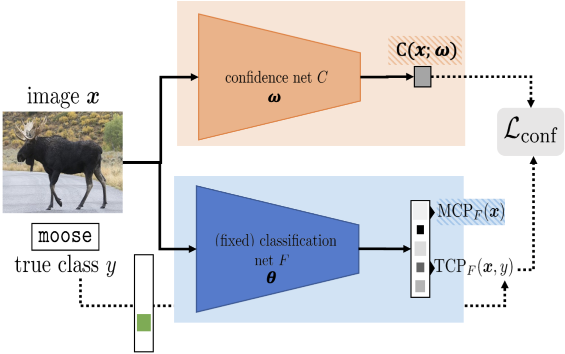


(b) Our Proposal (True Class Probability)

ConfidNet [Corbière et al., 2019]

- However, $TCP(\boldsymbol{x}, y^*)$ is **unknown** at test time.
- ConfidNet [Corbière et al., 2019]: Learning TCP Model Confidence

Given $\mathcal{D}_{train}$, **learn a confidence model** with parameters θ such that $\forall \boldsymbol{x} \in \mathcal{D}_{train}$, its scalar output $\hat{c}(\boldsymbol{x}, \theta)$ close to $TCP(\boldsymbol{x}, y^*)$

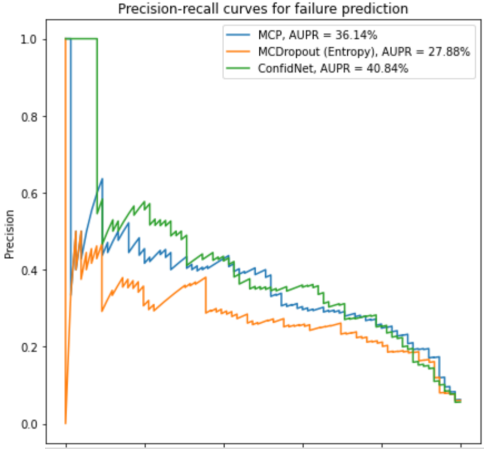


As $TCP(\boldsymbol{x}, y^*) \in [0, 1]$, we propose ℓ_2 loss to train ConfidNet:

$$\mathcal{L}_{conf}(\theta; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} (\hat{c}(\boldsymbol{x}_i, \theta) - c^*(\boldsymbol{x}_i, y_i^*))^2$$

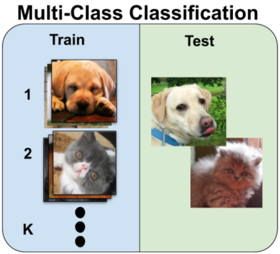
Practical Session: Failure Prediction on MNSIT

- Compute difference confidence estimate $\hat{C}$ on MNIST test data
  - MCP $\boldsymbol{MCP}(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} p(Y = k | \boldsymbol{w}, \boldsymbol{x})$, MC dropout, ConfdNet
- Compare confidence criterion quality
  - Rank test examples wrt uncertainty criterion
  - Compute Precision/Recall (PR) curve and AP_{errors}

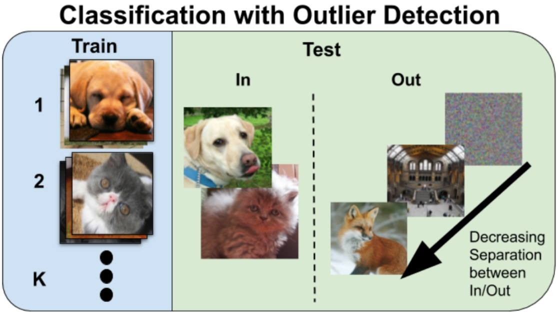


Applications: Out of Distribution Detection

- Standard classification: same classes during train and test

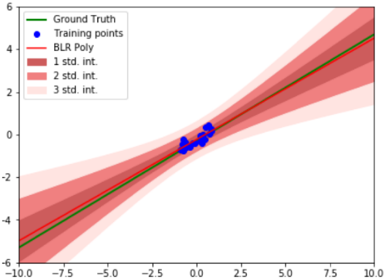


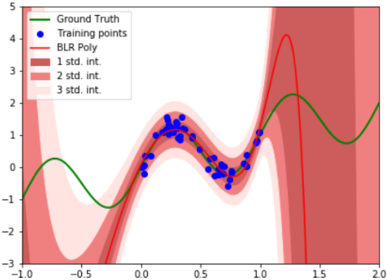
- Out of Distribution Detection: detecting unknown classes, far from training distribution



Out of Distribution Detection

- Detecting examples far from training distribution
  ⇒ natural use of Bayesian confidence estimates





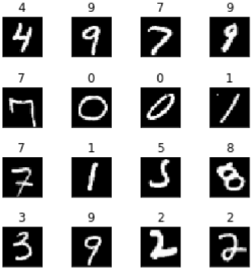
- **In classification:** estimate dispersion (*e.g.* mutual info) for Bayesian models (*e.g.* MC dropout)
- **Baseline solution:** assume that logits (before soft-max) smaller for out- than for in-distribution samples

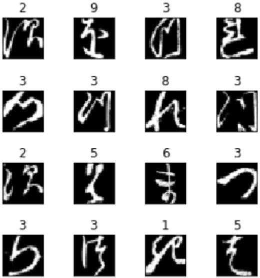
Other methods specifically learn dataset-specific OOD methods

- Ex: ODIN [Liang et al., 2018]
- Modify inputs so that maximizing differences between in and out samples
  - Temperature scaling, as in [Guo et al., 2017] : $P(\hat{y}_k) = \frac{e^{s_k/T}}{\sum\limits_{k'=1}^{K} e^{s_{k'}/T}}$
  - Apply "inverse adversarial attack": gradient of input wrt predicted class, then change input to increase prediction : $\tilde{x} = x - \epsilon \; sign\left[-\nabla_x \; log(P(\hat{y}_k))\right]$
  - Hyper-parameters (T, ϵ) optimized on val set s.t. maximizing AUC between in-distribution and OOD-samples

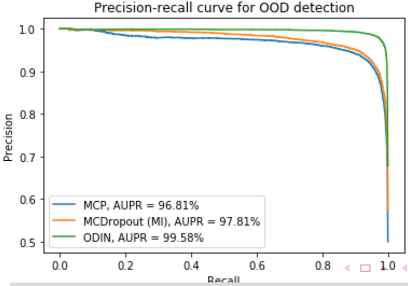
Out of Distribution Example

- Model trained on MNIST - Out: Kuzushiii-MNIST





- Comparing MCP, MC dropout, ODIN for OOD detection

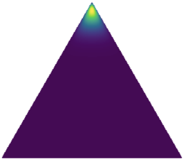


Other methods specifically learn dataset-specific OOD methods

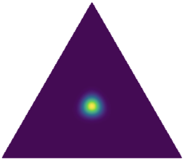
- Ex: Outlier Exposure (OE) [Hendrycks et al., 2019]
  - Training with in-distribution AND OOD samples :

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{in}}\left[\mathcal{L}(f(x),y)\right] + \lambda\mathbb{E}_{x'\sim\mathcal{D}_{out}^{OE}}\left[\mathcal{L}_{OE}(f(x'))\right]$$

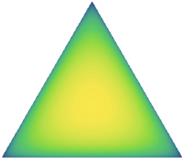
  - *e.g.* $\mathcal{L}(f(x),y)$ cross-entropy, $\mathcal{L}_{OE}(f(x')$ KL wrt uniform class distribution
- Out of Distribution with Dirichlet Prior Networks
  - **Recap for in-distribution samples:** minimize $KL\left[q_\theta(\mathbf{z_i}|\mathbf{x_i})||P(\mathbf{z_i}|\mathbf{x_i},\mathbf{y_i})\right]$, with $P(\mathbf{z_i}|\mathbf{x_i},\mathbf{y_i}) \sim Dir(\boldsymbol{\beta})$, *e.g.* $\boldsymbol{\beta} = (1,1,..101,..,1)$, *i.e.* $\beta_k = 1 + 100\cdot\delta_{k,k_i^*}$
  - **For out-of-distribution samples:** minimize $KL\left[q_\theta(\mathbf{z_i}|\mathbf{x_i})||P(\mathbf{z_i}|\mathbf{x_i},\mathbf{y_i})\right]$, with $P(\mathbf{z_i}|\mathbf{x_i},\mathbf{y_i}) \sim Dir(\boldsymbol{U})$, $\boldsymbol{U} = (1,1,...,1)$



(a) In-domain input with low uncertainty



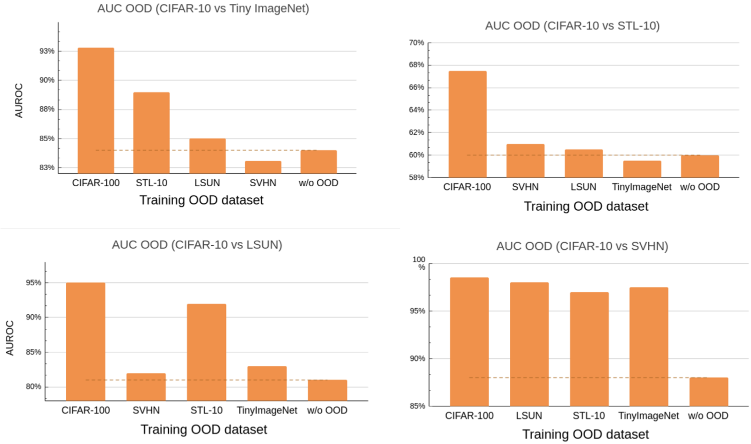
(b) In-domain input with high data uncertainty



(c) Out-of-distribution input

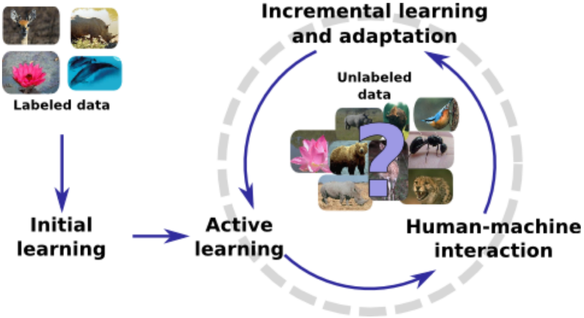
Relevance of using OOD training datasets?

- By essence, OOD are impossible to model!
- Performances way strongly depends on the training vs testing OOD datasets



Applications: Active Learning

- Active/ interactive learning: learning a model with few data annotated by users
- Challenge: determining most informative data to annotate
  - Most uncertain data: optimal convergence [Tong and Koller, 2002]



Applications: Active Learning

- Use of uncertainty in classification: active learning [Gal et al., 2017]

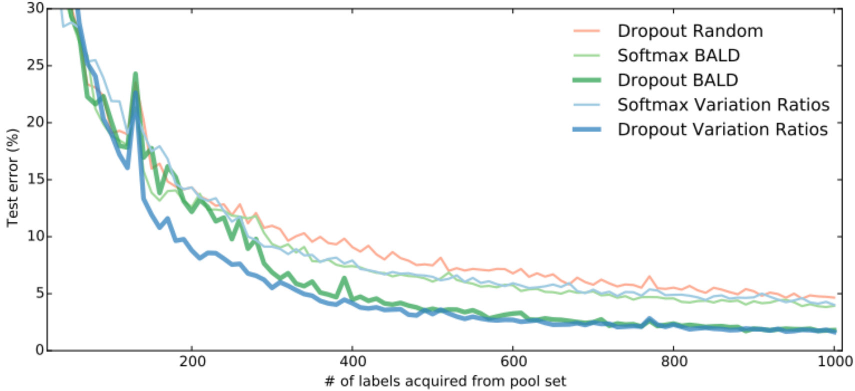
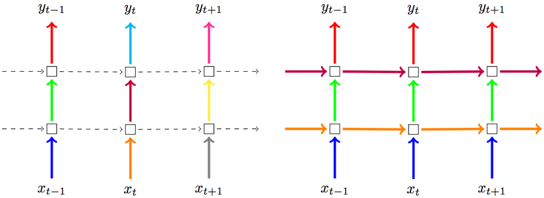


Fig. 5.1 **Test error on MNIST as a function of number of labels acquired from the pool set.** Two acquisition functions (*BALD* and *Variation Ratios*) evaluated with two approximating distributions — delta (*Softmax*) and Bernoulli (*Dropout*) — are compared to a *random* acquisition function.

Applications

- Use of uncertainty in classification: beyond MC nets
  - Application to ConvNets
  - and RNNs : BPTT: Bayesian Dropout [Gal and Ghahramani, 2016b]

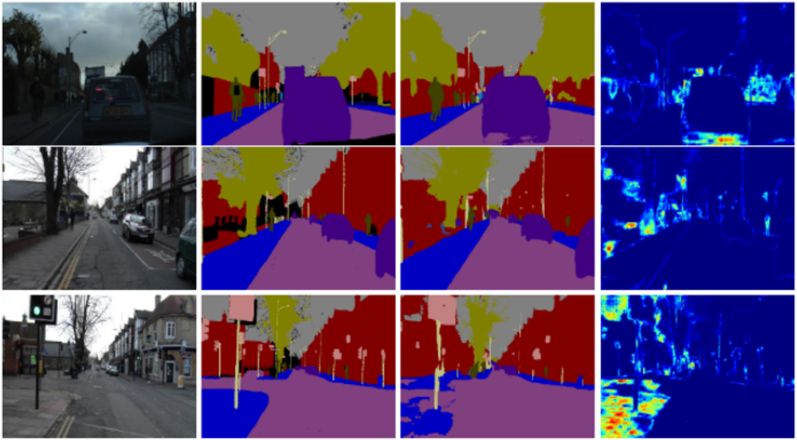


(a) Naive dropout RNN (b) Variational RNN

Figure 1: **Depiction of the dropout technique following our Bayesian interpretation (right) compared to the standard technique in the field (left).** Each square represents an RNN unit, with horizontal arrows representing time dependence (recurrent connections). Vertical arrows represent the input and output to each RNN unit. Coloured connections represent dropped-out inputs, with different colours corresponding to different dropout masks. Dashed lines correspond to standard connections with no dropout. Current techniques (naive dropout, left) use different masks at different time steps, with no dropout on the recurrent layers. The proposed technique (Variational RNN, right) uses the same dropout mask at each time step, including the recurrent layers.

Application in Semantic Segmentation

From [Kendall and Gal, 2017]



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (e) Epistemic Uncertainty

Application in Reinforcement Learning

- **Q-learning**: agent selects the best action following current Q-function estimate with some probability, and explores otherwise (ϵ-greedy)
- **Option:** use uncertainty estimates (eg dropout Q-network) to drive exploration, *e.g.* Thompson sampling (Thompson, 1933) and converge faster

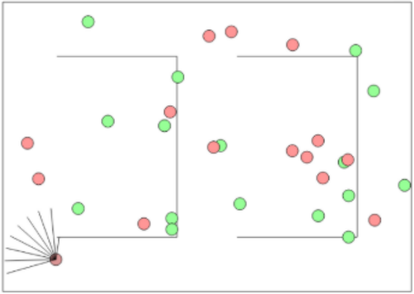


Fig. 5.3 Depiction of the reinforcement learning problem used in the experiments. The agent is in the lower left part of the maze, facing north-west.

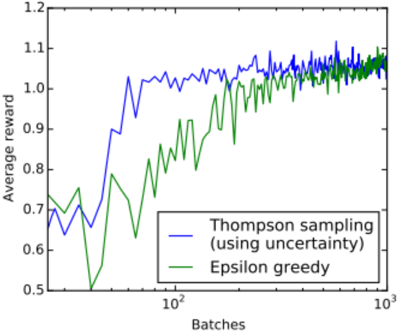


Fig. 5.4 **Log plot of average reward** obtained by both epsilon greedy (in green) and our approach (in blue), as a function of the number of batches.

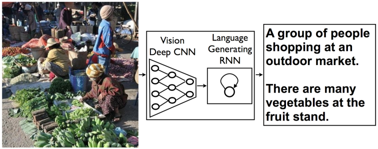
Outline

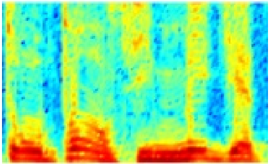
Bayesian Neural Networks for Classification

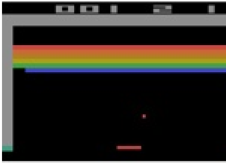
Applications of Uncertainty in Deep Learning

Other Robustness Issues

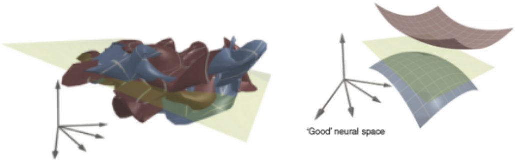
Deep Learning Theory







- Deep Learning: huge impact in terms of experimental results
- BUT: formal understanding still limited,
  - Optimization: non-convex problem
  - Generalization & over-fitting
  - Robustness

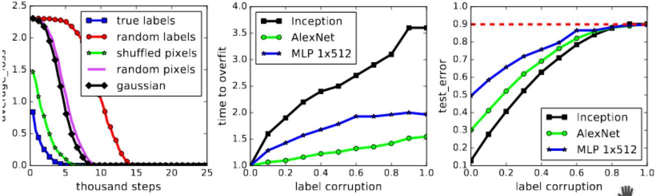


Deep Learning and generalization

- Rademacher complexity: capacity of a model to fit random label :

$$\mathcal{R}_n(\mathcal{H}) = E_\sigma \left[sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right]$$

- Rethinking generalization: Zhang *et. al.* ICLR17 [Zhang et al., 2017]



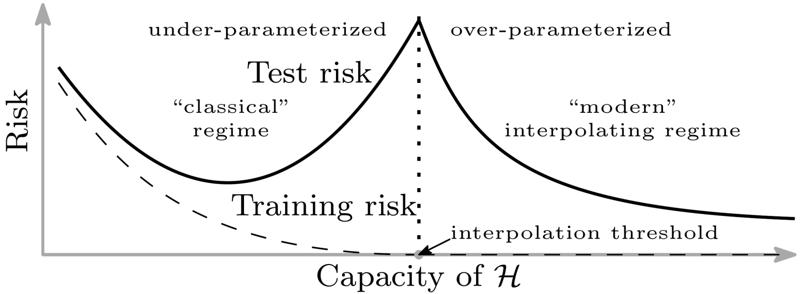
(a) learning curves (b) convergence slowdown (c) generalization error growth

  - Deep models easily fits random labels !!
  - $\mathcal{R}_n(\mathcal{H}) \approx 1 \Rightarrow$ no theoretical guarantee on generalization performances

- **Classical learning theory insufficient to explain the good generalization behavior of deep models**

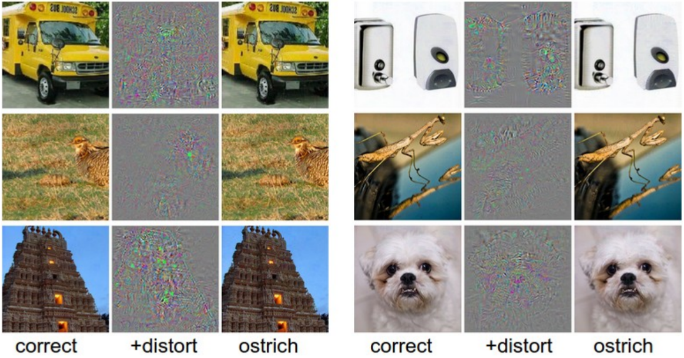
Generalization and over-parametrized models

- Double U-curve phenomena observed with deep models! [Belkin et al., 2019]



Deep Learning (DL) & Stability

- **Stability:** decision function with "controlled" variations
  - Small input variations ⇔ reasonably small output variations on decision, *e.g.* Lipschitz property
  - **Decision function of deep Models not always stable**
    - **Ex: Adversarial Examples**



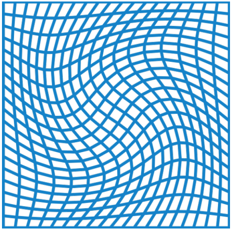
Deep Learning (DL) & Stability

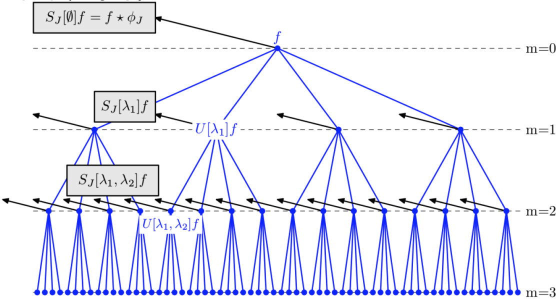
- Adversarial attacks in real-world

[Evtimov et al., 2017]

Formal stability analysis of deep models

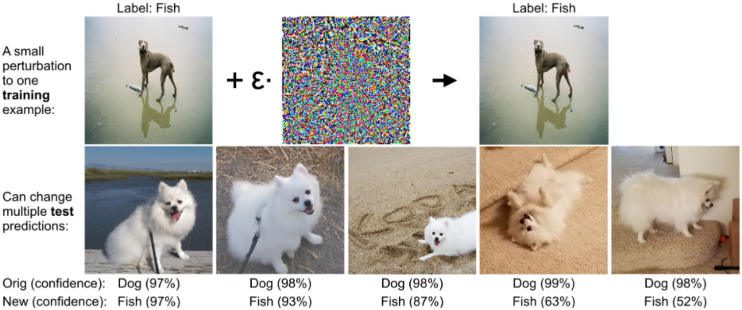
- Harmonic analysis in scattering operators [Mallat, 2012, Bruna and Mallat, 2013], *i.e.* "deep wavelets"
  - Show stability / invariance to diffeomorphisms
  - Stability bounds
- Generalized to deep kernel machines, closer to SoTA deep ConvNet architectures [Bietti and Mairal, 2017]





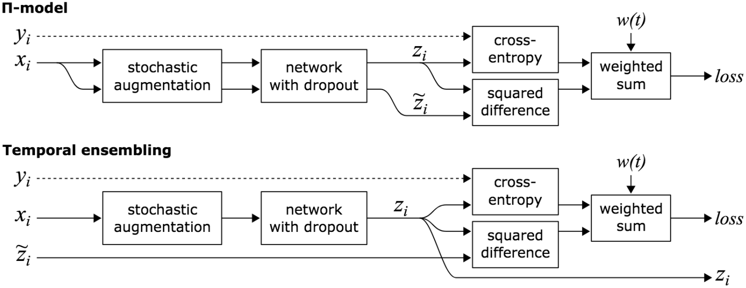
Formal stability analysis of deep models

- Influence Functions [Cook and Weisberg, 1980]
  - Characterize decision function influence on training examples
    - Removing a training point: $\mathcal{I}_{up,loss}(z, z_{test}) = -\nabla_\theta L(z_{test}, \hat{\theta})^T H_\theta^{-1} \nabla_\theta L(z, \hat{\theta})$
    - Perturbing it: $\mathcal{I}_{pert,loss}(z, z_{test})^T = -\nabla_\theta L(z_{test}, \hat{\theta})^T H_\theta^{-1} \nabla_x \nabla_\theta L(z, \hat{\theta})$
  - Adapted / applied to deep networks [Koh and Liang, 2017]
- Data poisoning



Ad hoc stability training

- Regularization criterion supporting learning stable decision function
  - Underlying model might not be stable, but helps to focus on a subet of stable functions of the family
- Robustness of the decision to transformations [Sajjadi et al., 2016], stability accross iterations [Laine and Aila, 2017, Tarvainen and Valpola, 2017]



References I

[Belkin et al., 2019] Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019).
Reconciling modern machine-learning practice and the classical bias–variance trade-off.
Proceedings of the National Academy of Sciences, 116(32):15849–15854.

[Bietti and Mairal, 2017] Bietti, A. and Mairal, J. (2017).
Invariance and stability of deep convolutional representations.
In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6210–6220. Curran Associates, Inc.

[Bruna and Mallat, 2013] Bruna, J. and Mallat, S. (2013).
Invariant scattering convolution networks.
IEEE Trans. Pattern Anal. Mach. Intell., 35(8):1872–1886.

[Cook and Weisberg, 1980] Cook, R. and Weisberg, S. (1980).
Characterizations of an empirical influence function for detecting influential cases in regression.
Technometrics, 22(4):495–508.

[Corbière et al., 2019] Corbière, C., Thome, N., Bar-Hen, A., Cord, M., and Pérez, P. (2019).
Addressing Failure Detection by Learning Model Confidence.
In *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada.

[Evtimov et al., 2017] Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D. (2017).
Robust physical-world attacks on machine learning models.
CoRR, abs/1707.08945.

[Gal, 2016] Gal, Y. (2016).
Uncertainty in Deep Learning.
PhD thesis, University of Cambridge.

[Gal and Ghahramani, 2015] Gal, Y. and Ghahramani, Z. (2015).
Bayesian convolutional neural networks with bernoulli approximate variational inference.
CoRR, abs/1506.02158.

References II

[Gal and Ghahramani, 2016a] Gal, Y. and Ghahramani, Z. (2016a).
Dropout as a bayesian approximation: Representing model uncertainty in deep learning.
In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org.

[Gal and Ghahramani, 2016b] Gal, Y. and Ghahramani, Z. (2016b).
A theoretically grounded application of dropout in recurrent neural networks.
In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1027–1035, USA. Curran Associates Inc.

[Gal et al., 2017] Gal, Y., Islam, R., and Ghahramani, Z. (2017).
Deep Bayesian Active Learning with Image Data.
In *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*.

[Guo et al., 2017] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017).
On calibration of modern neural networks.
CoRR, abs/1706.04599.

[Hendrycks et al., 2019] Hendrycks, D., Mazeika, M., and Dietterich, T. G. (2019).
Deep anomaly detection with outlier exposure.
In *ICLR*.

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017).
What uncertainties do we need in bayesian deep learning for computer vision?
In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc.

[Koh and Liang, 2017] Koh, P. W. and Liang, P. (2017).
Understanding black-box predictions via influence functions.
In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia. PMLR.

References III

[Laine and Aila, 2017] Laine, S. and Aila, T. (2017).
Temporal ensembling for semi-supervised learning.
In *International Conference on Learning Representations (ICLR)*.

[Lakshminarayanan et al., 2017] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017).
Simple and scalable predictive uncertainty estimation using deep ensembles.
In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Liang et al., 2018] Liang, S., Li, Y., and Srikant, R. (2018).
Enhancing the reliability of out-of-distribution image detection in neural networks.
In *International Conference on Learning Representations*.

[Malinin and Gales, 2018] Malinin, A. and Gales, M. (2018).
Predictive uncertainty estimation via prior networks.
In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Mallat, 2012] Mallat, S. (2012).
Group invariant scattering.
Communications in Pure and Applied Mathematics, 10:1331–1398.

[Sajjadi et al., 2016] Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016).
Regularization with stochastic transformations and perturbations for deep semi-supervised learning.
In *Advances in Neural Information Processing Systems (NIPS)*.

[Sensoy et al., 2018] Sensoy, M., Kaplan, L., and Kandemir, M. (2018).
Evidential deep learning to quantify classification uncertainty.
In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Tarvainen and Valpola, 2017] Tarvainen, A. and Valpola, H. (2017).
Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.
In *Advances in Neural Information Processing Systems (NIPS)*.

References IV

[Tong and Koller, 2002] Tong, S. and Koller, D. (2002).
Support vector machine active learning with applications to text classification.
J. Mach. Learn. Res., 2:45–66.

[Zhang et al., 2017] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017).
Understanding deep learning requires rethinking generalization.