Uncertainty Quantification - Application of Uncertainty

Clément Rambour

Sorbonne Université
ISIR - équipe MLIA

Outline

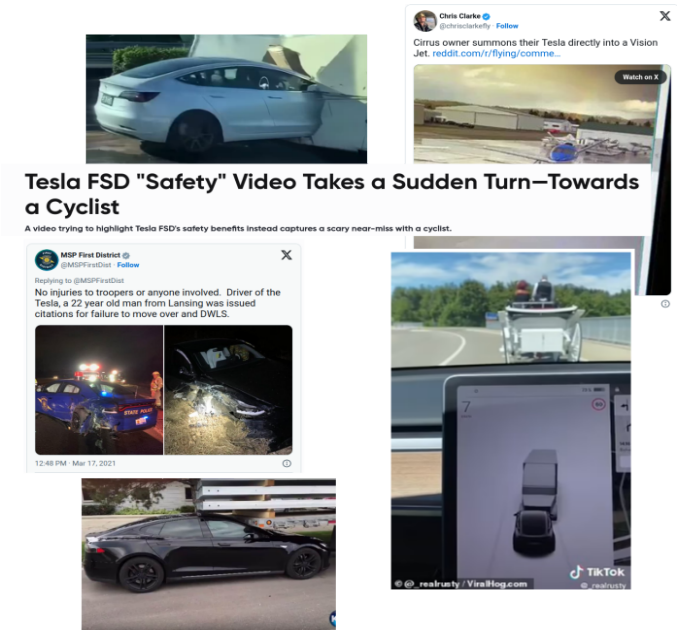
State of Play

- Within last decade, huge advances in deep neural networks (DNN) :
  - Breakthrough in computer vision and NLP
  - Ultra-realistic data synthesis
  - Superhuman AI in games
  - Real-world AI : autonomous driving, healthcare, robotics, risk management

State of Play

- Within last decade, huge advances in deep neural networks (DNN) :
  - Breakthrough in computer vision and NLP
  - Ultra-realistic data synthesis
  - Superhuman AI in games
  - Real-world AI : autonomous driving, healthcare, robotics, risk management

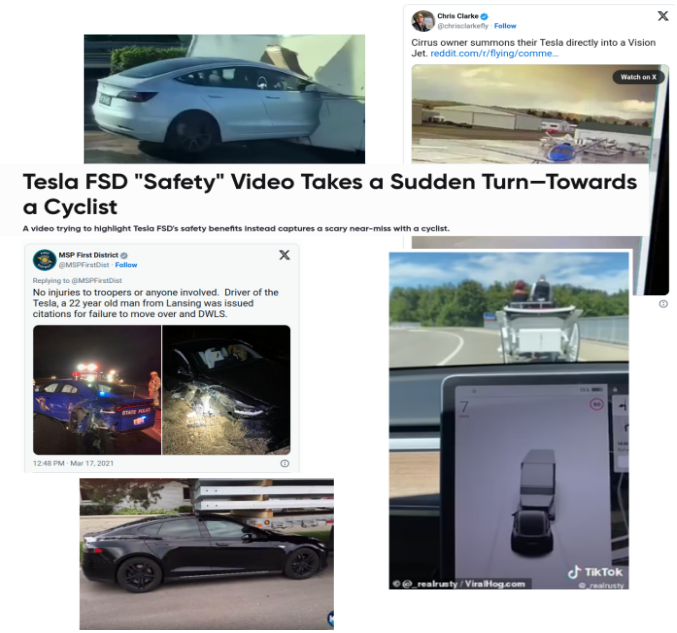
    ↪ Actually remains limited (or should be !).



State of Play

- Within last decade, huge advances in deep neural networks (DNN) :
  - Breakthrough in computer vision and NLP
  - Ultra-realistic data synthesis
  - Superhuman AI in games
  - Real-world AI : autonomous driving, healthcare, robotics, risk management

    ↪ Actually remains limited (or should be !).



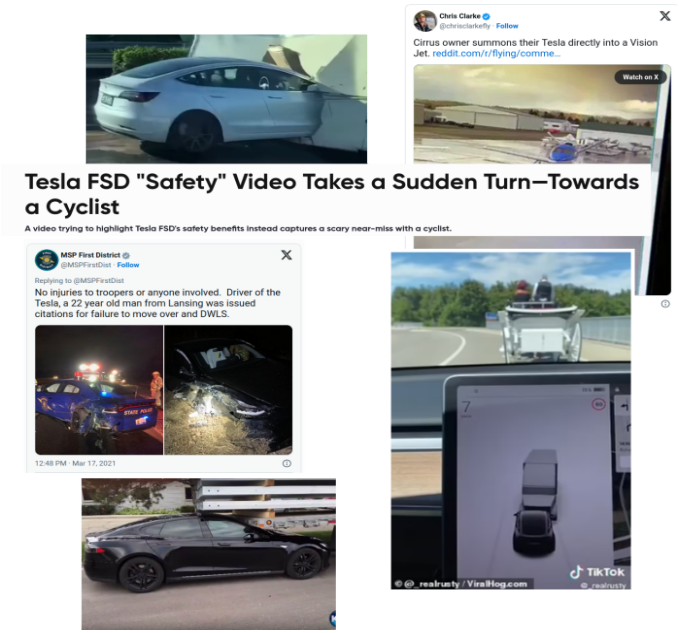
DNN weak spot

- Lack of transparency
- Inability to distinguish between in-domain and out-of-domain samples
- Inability to provide reliable uncertainty estimates
- Vulnerability to adversarial attacks

State of Play

- Within last decade, huge advances in deep neural networks (DNN) :
  - Breakthrough in computer vision and NLP
  - Ultra-realistic data synthesis
  - Superhuman AI in games
  - Real-world AI : autonomous driving, healthcare, robotics, risk management

    ↪ Actually remains limited (or should be !).



DNN weak spot

- Lack of transparency
- Inability to distinguish between in-domain and out-of-domain samples
- Inability to provide reliable uncertainty estimates
- Vulnerability to adversarial attacks

↪ Need for uncertainty estimates

↪ Uncertain prediction can be addressed to human experts

Sources of Uncertainty

Model or *epistemic* uncertainty

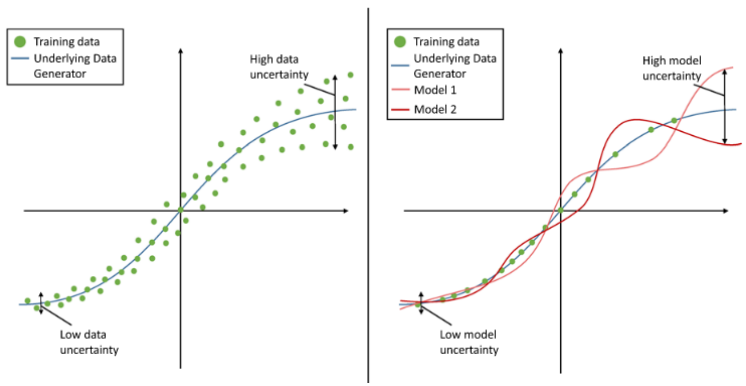
- Insufficient model structure
- Errors in the training procedure
  ↪ Lack of structure in the representations, spurious correlations
- Lack of knowledge : unknown samples, bad coverage of the training set

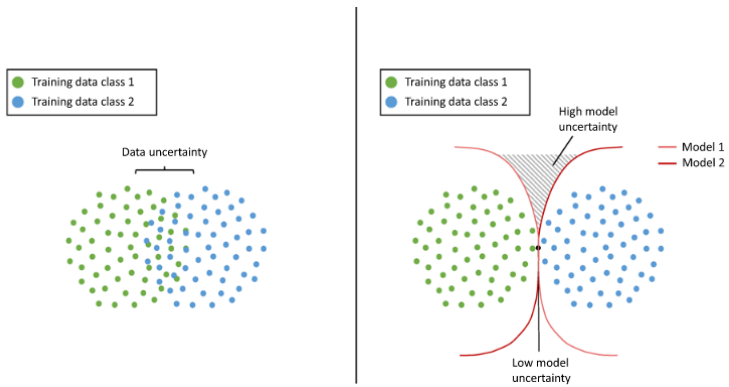
  ↪ Overconfident and unstable predictions

Data or *aleatoric* uncertainty

- Complexity of real world data
  ↪ inter-class ambiguities, intra-class dispersion
  ↪ High dimensional, high variability data
- Error in the data collection

  ↪ Noise, missing data, outliers



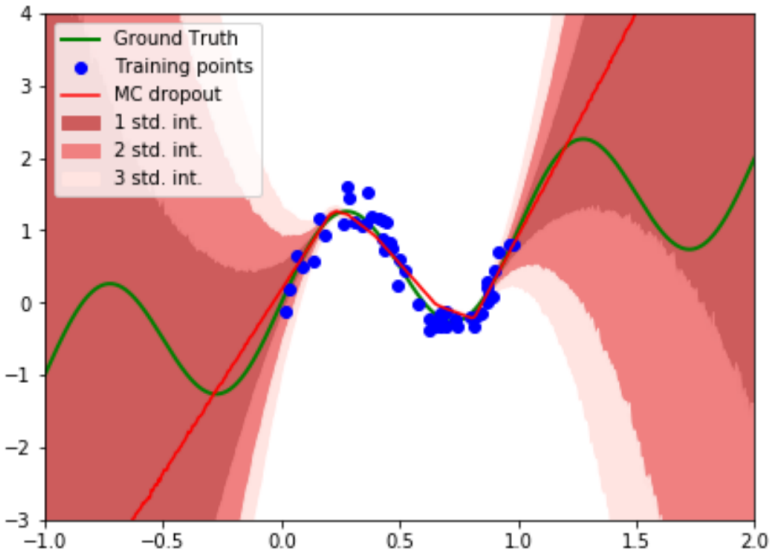


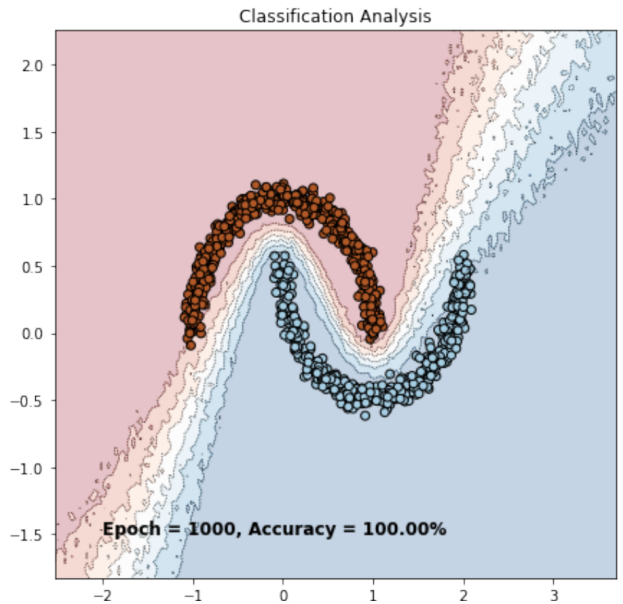
Recap : Bayesian Neural Networks (BNNs)

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

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

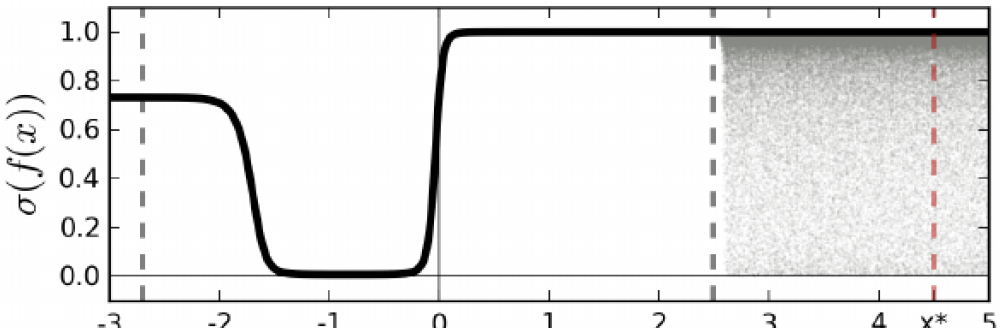
  - Sampling : discrete approximation of the distribution $p(y|\mathbf{x}^*,\mathcal{D})$
    - Regression : y continuous : mean prediction, dispersion $\Leftrightarrow$ uncertainty
    - Classification : y discrete, *e.g.* binary classification $p(y=1|\mathbf{x}^*,\mathcal{D})$

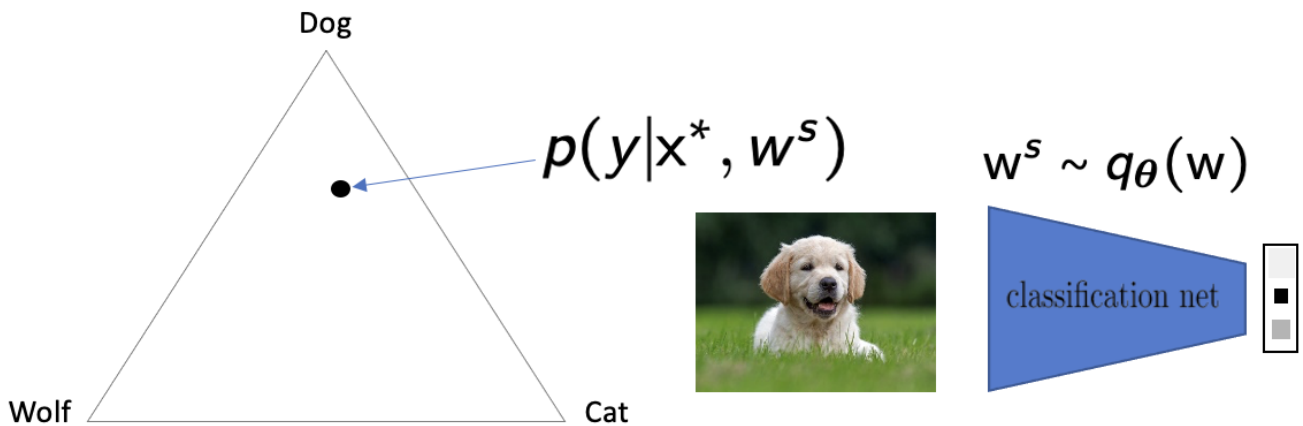




BNNs : Uncertainty in Classification

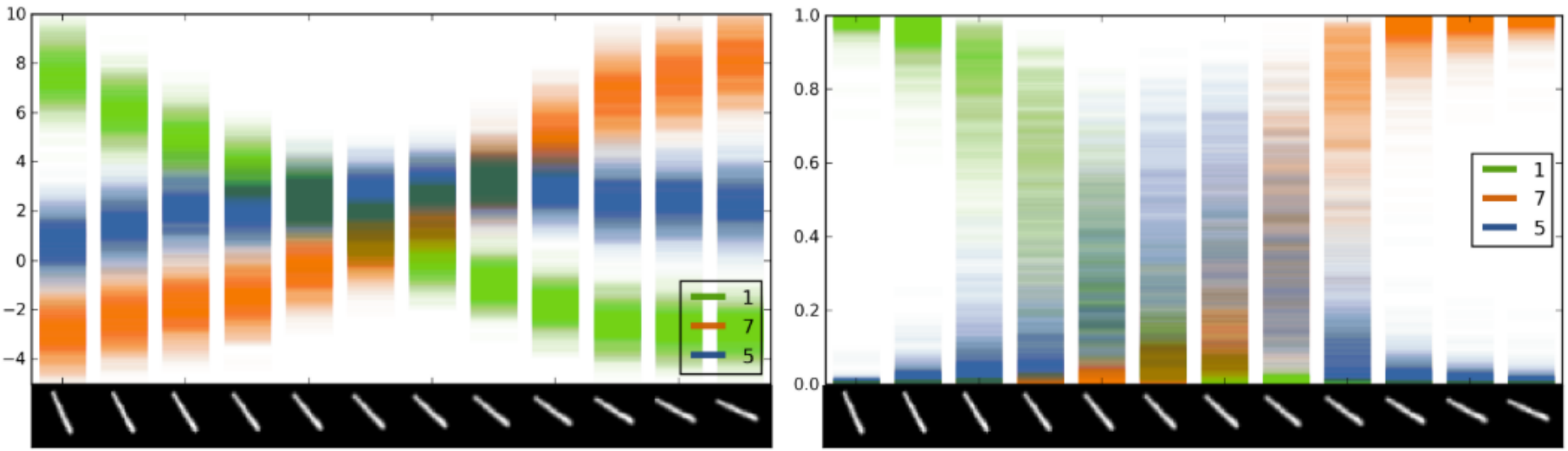
- Only using discrete output : averaging $p(y=1|\mathbf{x}^*, w^s)$ $\mathbf{w}^s \sim q_{\boldsymbol{\theta}}(\mathbf{w})$
  - Loosing the variability of $p(y=1|\mathbf{x}^*, w^s)$ among $\mathbf{w}^s$ samples
  - Why not using the continuous output of the model (after softmax) as continuous output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

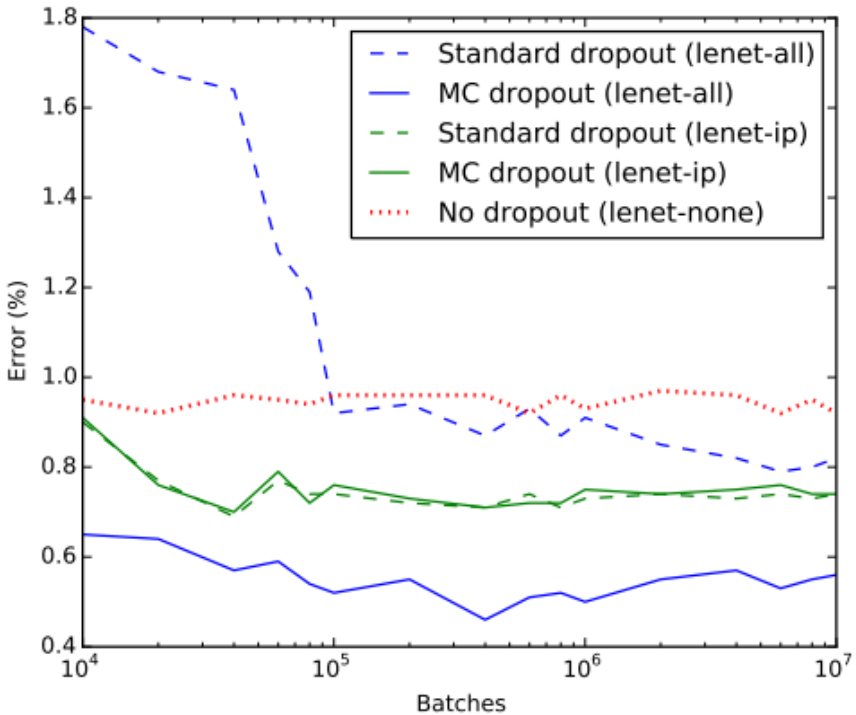


(a) Softmax *input* scatter (b) Softmax *output* scatter

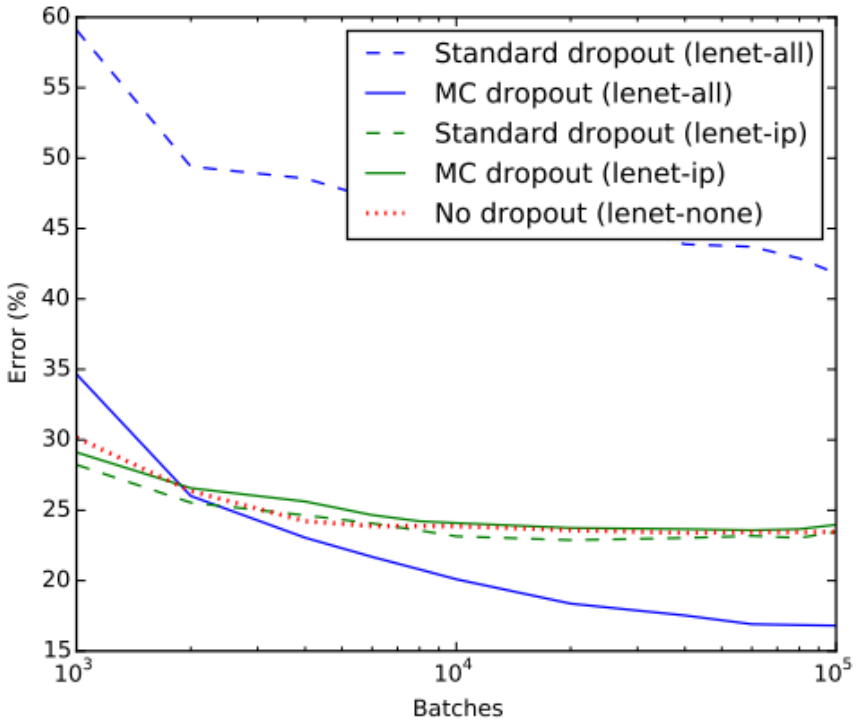
Pred : [1 1 1 1 1 5 5 7 7 7 7 7]

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

Outline

Calibration – A well-calibrated model

- Suppose we train a model to predict the probability of rain tomorrow.

Calibration – A well-calibrated model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	20%	45%	65%	80%

Calibration – A well-calibrated model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	20%	45%	65%	80%

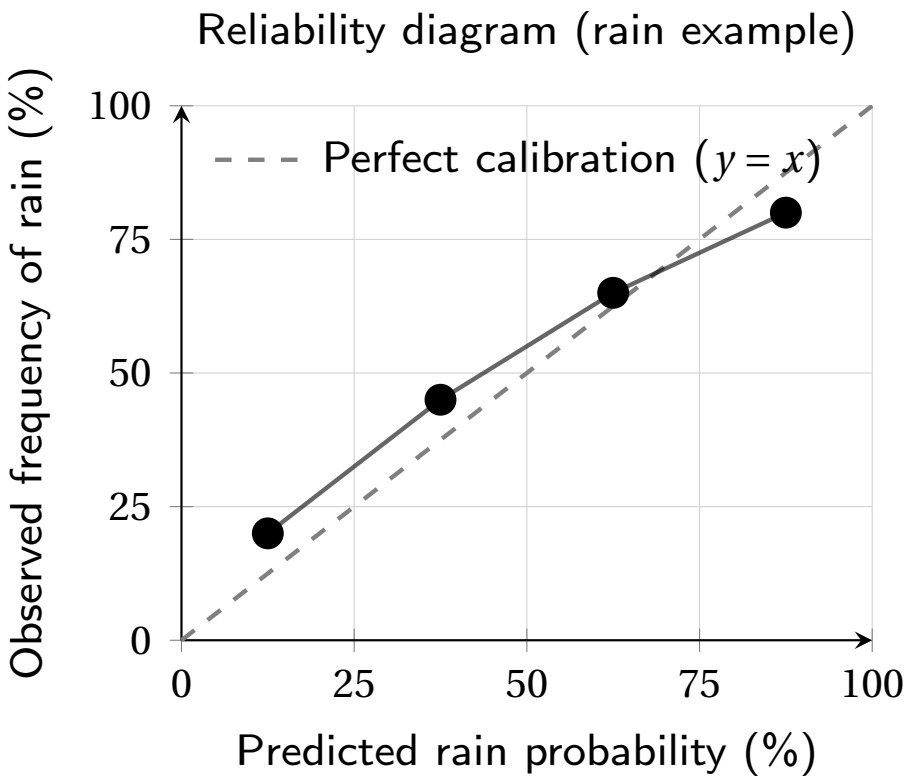
- The model is well-calibrated : the observed frequency matches the predicted probability.

Calibration – A well-calibrated model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	20%	45%	65%	80%

- The model is well-calibrated : the observed frequency matches the predicted probability.



Calibration – An overconfident model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	5%	20%	40%	70%

Calibration – An overconfident model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	5%	20%	40%	70%

- The model is over-confident : the observed frequency is lower than the predicted probability.

Calibration – An overconfident model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	5%	20%	40%	70%

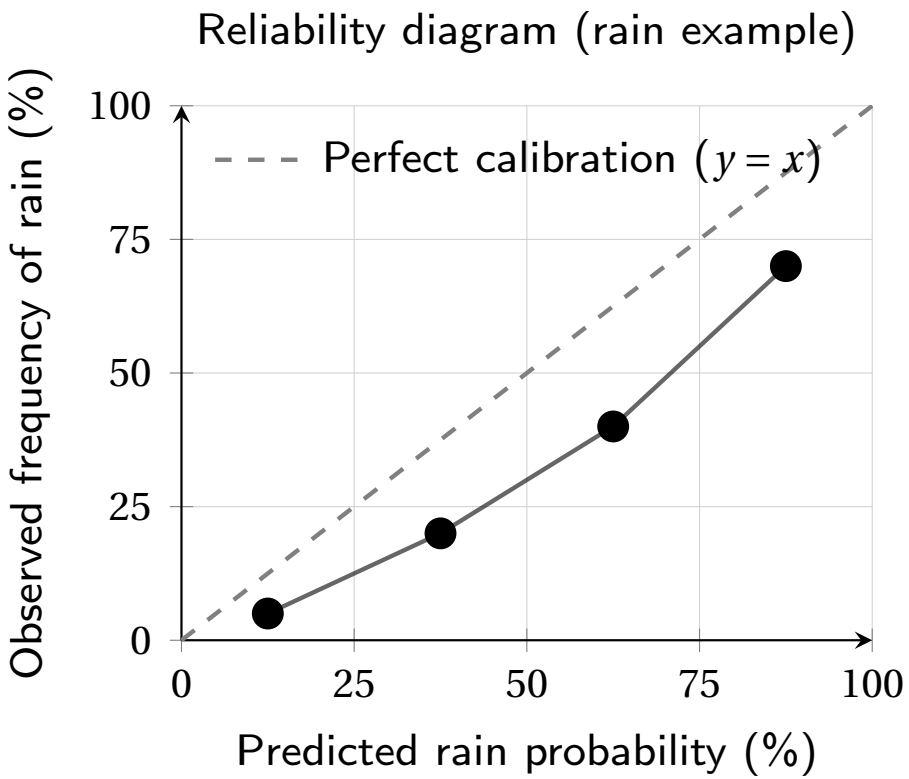
- The model is over-confident : the observed frequency is lower than the predicted probability.

Calibration – An overconfident model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	5%	20%	40%	70%

- The model is over-confident : the observed frequency is lower than the predicted probability.



Calibration – An underconfident model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	35%	60%	80%	95%

Calibration – An underconfident model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	35%	60%	80%	95%

- The model is under-confident : the observed frequency is higher than the predicted probability.

Calibration – An underconfident model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	35%	60%	80%	95%

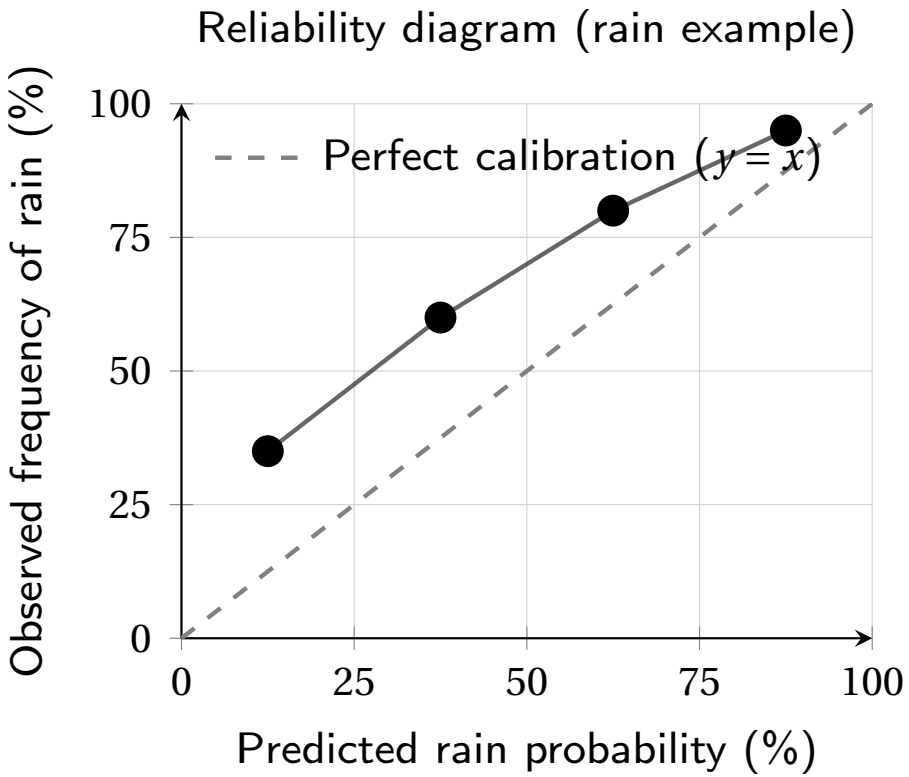
- The model is under-confident : the observed frequency is higher than the predicted probability.

Calibration – An underconfident model

- Suppose we train a model to predict the probability of rain tomorrow.
- After a year, we look back at its predictions :

Predicted rain probability	0–25%	25–50%	50–75%	75–100%
Proportion with rain	35%	60%	80%	95%

- The model is under-confident : the observed frequency is higher than the predicted probability.



Definition of Calibration

Calibration

Let (X, Y) be a random input–label pair with $Y \in \{1, \ldots, K\}$. A classifier f_w outputs a vector of predicted probabilities $f_w(X) = (\hat{p}_1, \ldots, \hat{p}_K)$, where $\hat{p}_k$ estimates $\mathbb{P}(Y = k \mid X)$.

Definition of Calibration

Calibration

Let (X, Y) be a random input–label pair with $Y \in \{1, \dots, K\}$. A classifier f_w outputs a vector of predicted probabilities $f_w(X) = \big(\hat{p}_1, \dots, \hat{p}_K\big)$, where $\hat{p}_k$ estimates $\mathbb{P}(Y = k \mid X)$.

We note the predicted class as $\hat{Y} = \arg\max_k \hat{p}_k$, and its associated confidence as $\hat{p} = \max_k \hat{p}_k$. The model is **well-calibrated** if, for every class k and every $p \in [0, 1]$,

$$\mathbb{P}\big(Y = \hat{Y} \mid \hat{p} = p\big) = p.$$

Definition of Calibration

Calibration

Let (X, Y) be a random input–label pair with $Y \in \{1, \ldots, K\}$. A classifier f_w outputs a vector of predicted probabilities $f_w(X) = \big(\hat{p}_1, \ldots, \hat{p}_K\big)$, where $\hat{p}_k$ estimates $\mathbb{P}(Y = k \mid X)$.

We note the predicted class as $\hat{Y} = \arg\max_k \hat{p}_k$, and its associated confidence as $\hat{p} = \max_k \hat{p}_k$. The model is **well-calibrated** if, for every class k and every $p \in [0, 1]$,

$$\mathbb{P}\big(Y = \hat{Y} \mid \hat{p} = p\big) = p.$$

Interpretation : Among all samples for which the model predicts label $\hat{Y}$ with confidence $\hat{p}$, the true label is Y in approximately $\hat{p} \times 100\%$ of cases.

Calibration vs. Accuracy

Accuracy

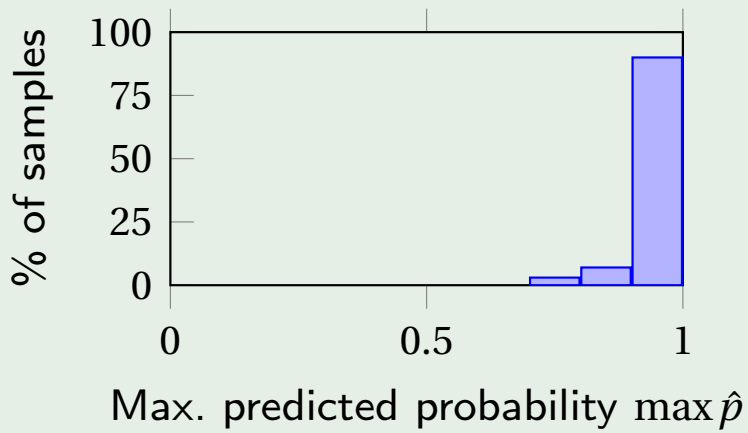
Accuracy measures the **proportion of correctly classified examples** :

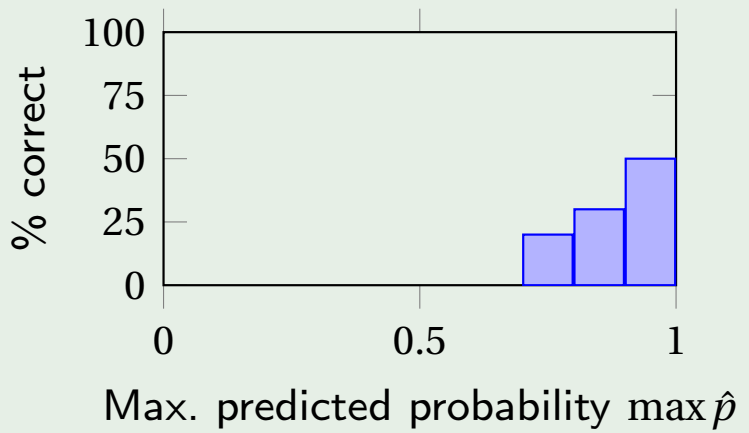
$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}.$$

Accuracy evaluates decisions, not the quality of the predicted probabilities.

Example

A model predicting class labels with high accuracy but always outputs probabilities close to 0 or 1.





Expected Calibration Error (ECE)

Definition

Predicted probabilities are grouped into M bins $\{B_m\}_{m=1}^{M}$. The Expected Calibration Error is defined as :

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

- $\text{acc}(B_m)$: empirical accuracy in bin B_m

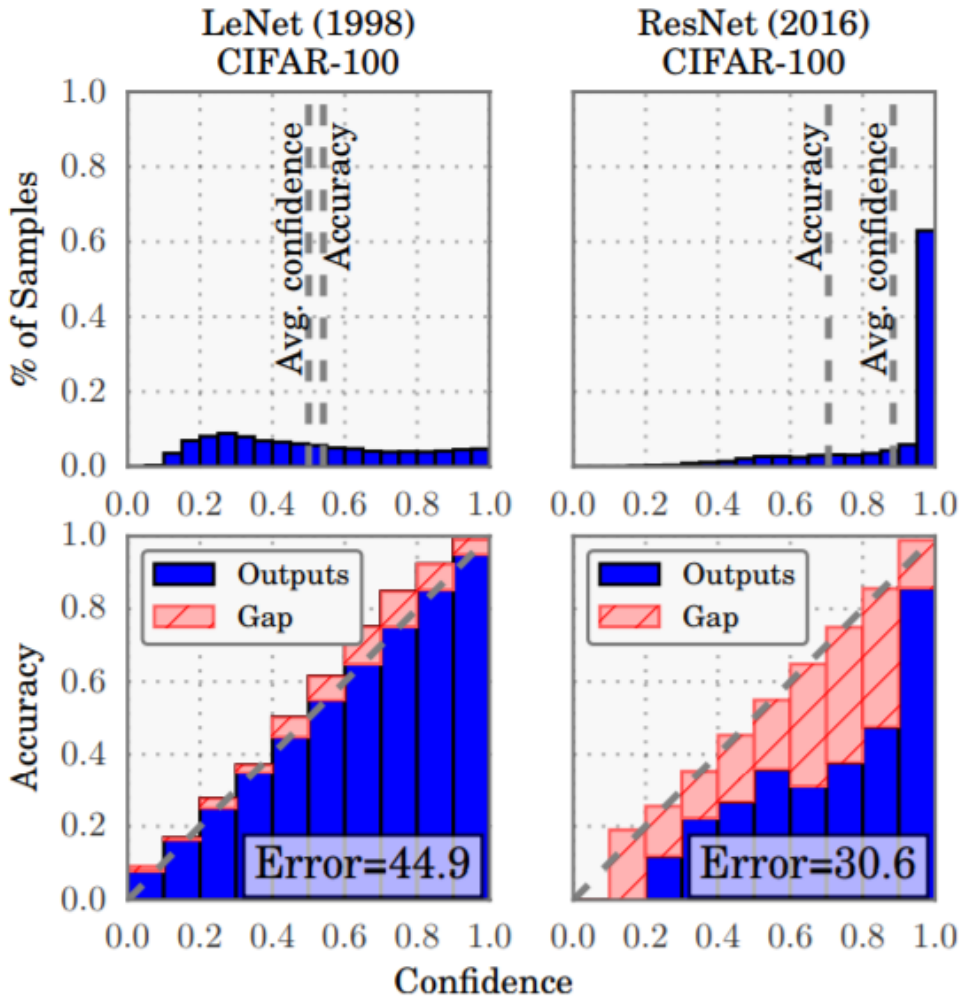
$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i)$$

- $\text{conf}(B_m)$: average predicted confidence in bin B_m

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

- perfect calibration $\iff acc(B_m) = conf(B_m) \quad \forall m$ and ECE $= 0$
- ⚠depends on binning, not a proper scoring rule

Model Calibration in deep learning



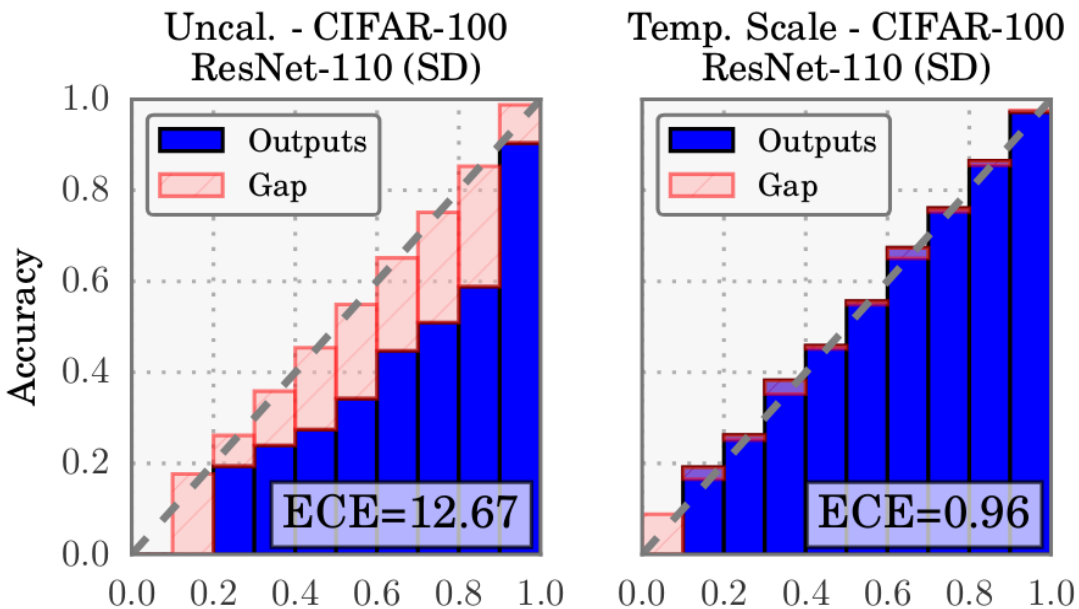
[Guo et al., 2017] **showed that modern neural networks are no longer well-calibrated !**

Post-hoc Model Calibration in deep learning

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

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

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



Calibration through regularization

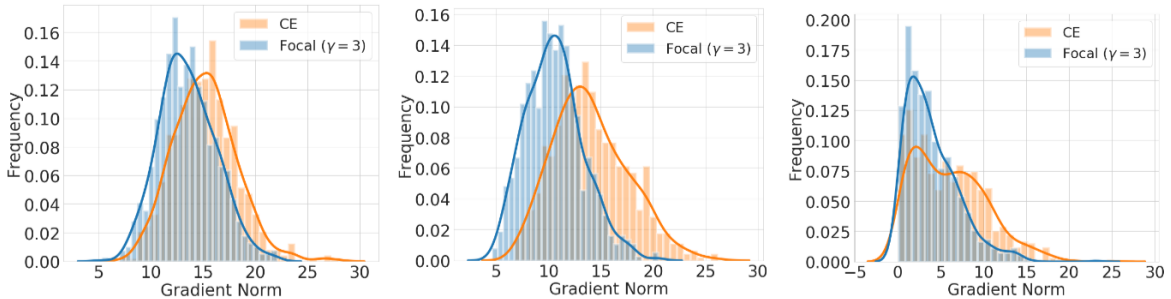


Figure – [Mukhoti et al., 2020]

- Miscalibration and NLL overfitting are linked : wrongly classified samples push NLL and ECE up.
- DNN becomes more and more confident about its predictions during training.
- Minimal cross-entropy loss $\Rightarrow \hat{p} = 1$ and thus weight norm explosion.

Calibration through regularization

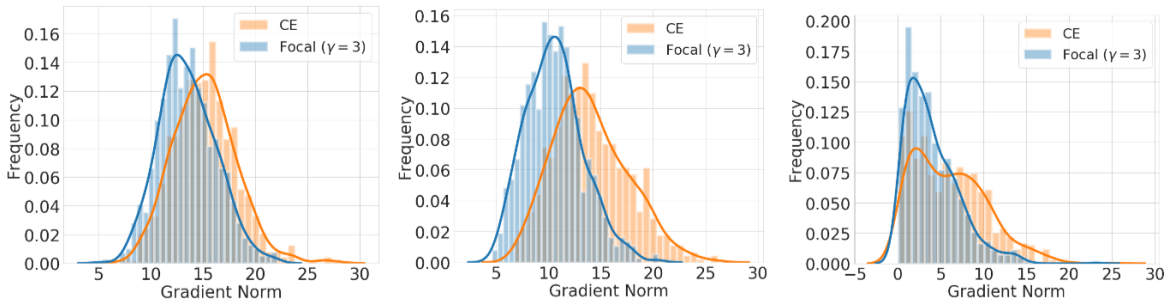


Figure – [Mukhoti et al., 2020]

Regularization

- Weight decay penalizes large weights :
  - limits the growth of logits
  - prevents over-confident predictions
- Label smoothing :
  - replaces one-hot labels with smoothed version
  - prevents the model from becoming too confident about its predictions

Calibration through regularization

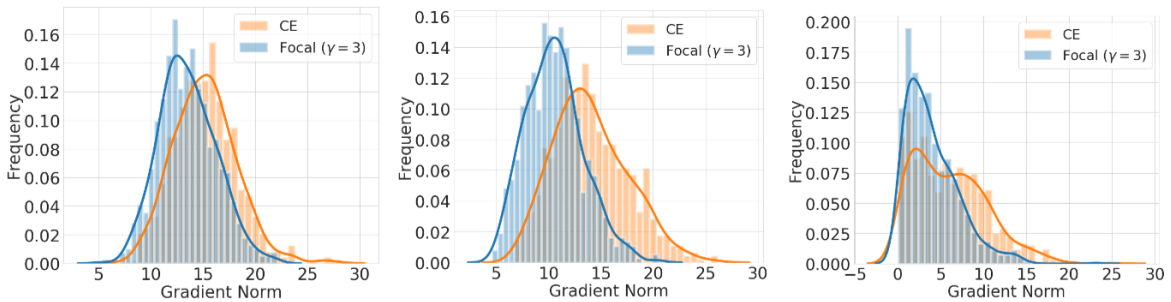


Figure – [Mukhoti et al., 2020]

Regularization

- Gradient norm regularization : penalizes large gradients of the logit w.r.t. inputs
  - *E.g.* Focal Loss [Lin et al., 2017] :

$$\mathcal{L}_{\text{FL}} = -(1-\hat{p})^{\gamma} \log \hat{p}$$

  - Focal loss down weights gradients of wrongly classified examples

$$\text{Cross-entropy :} \quad \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_y} = \hat{p}_y - 1$$

$$\text{Focal loss :} \quad \frac{\partial \mathcal{L}_{\text{FL}}}{\partial z_y} \propto (1-\hat{p}_y)^{\gamma}$$

Influence of the architecture over Calibration

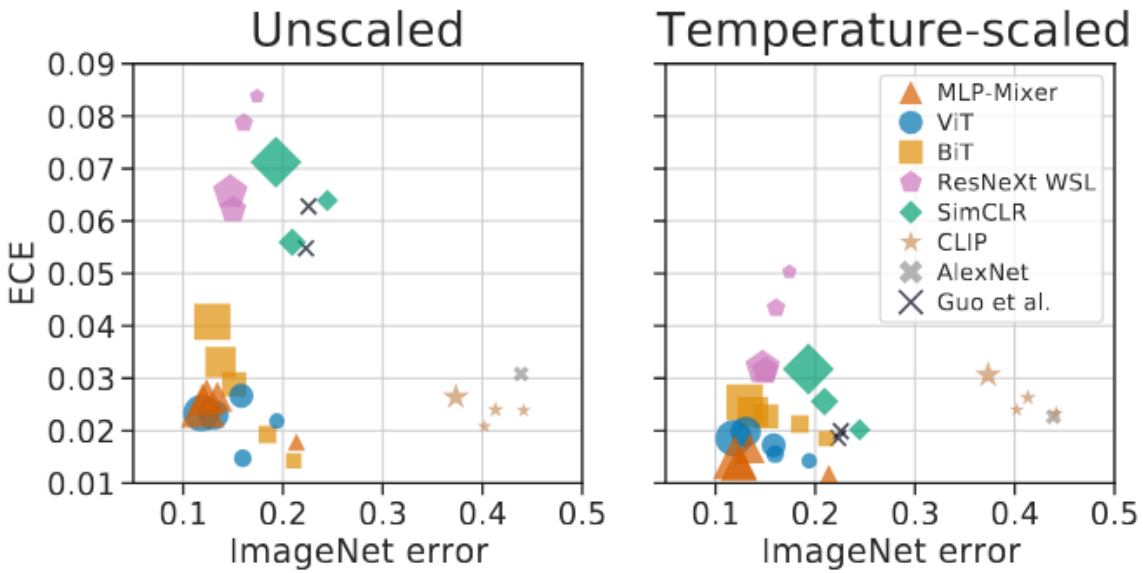


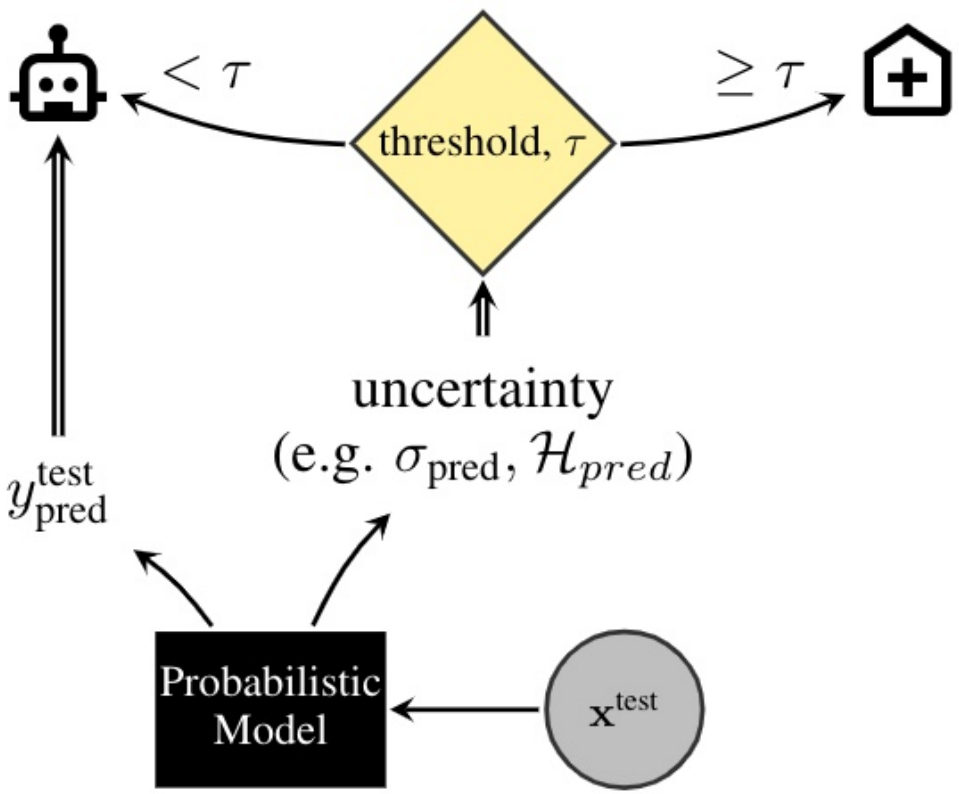
Figure – [Minderer et al., 2021]

- Overconfident deep nets : not universally true for modern architectures.
- Post-hoc calibration such as temperature scaling still helps
- Newer models (ViT, MLP Mixer) : intrinsically better calibrated than older CNNs.
- Calibration and accuracy remain positively correlated, even under distribution shift.

Outline

Failure Prediction

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



Failure Prediction

Definition

- Classification model f_w trained on $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
- Model prediction : $\hat{\boldsymbol{y}} = \arg\max_{k \in \mathcal{Y}} [f_w(\boldsymbol{x})]_k = \arg\max_{k \in \mathcal{Y}} \; p(Y = k | \boldsymbol{w}, \boldsymbol{x})$
- Model confidence $\hat{C}(\boldsymbol{x})$: score reflecting the trust in the prediction
- **Goal** : use confidence score to **accept** / **reject** model predictions based on a threshold τ :

$$\hat{C}(\boldsymbol{x}) \geq \tau \Rightarrow \text{accept prediction} \quad \text{else} \quad \text{reject prediction}$$

Failure Prediction

Definition

- Classification model f_w trained on $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$
- Model prediction : $\hat{y} = \arg\max_{k \in \mathcal{Y}} [f_w(x)]_k = \arg\max_{k \in \mathcal{Y}} \ p(Y = k | w, x)$
- Model confidence $\hat{C}(x)$: score reflecting the trust in the prediction
- **Goal** : use confidence score to **accept** / **reject** model predictions based on a threshold τ :

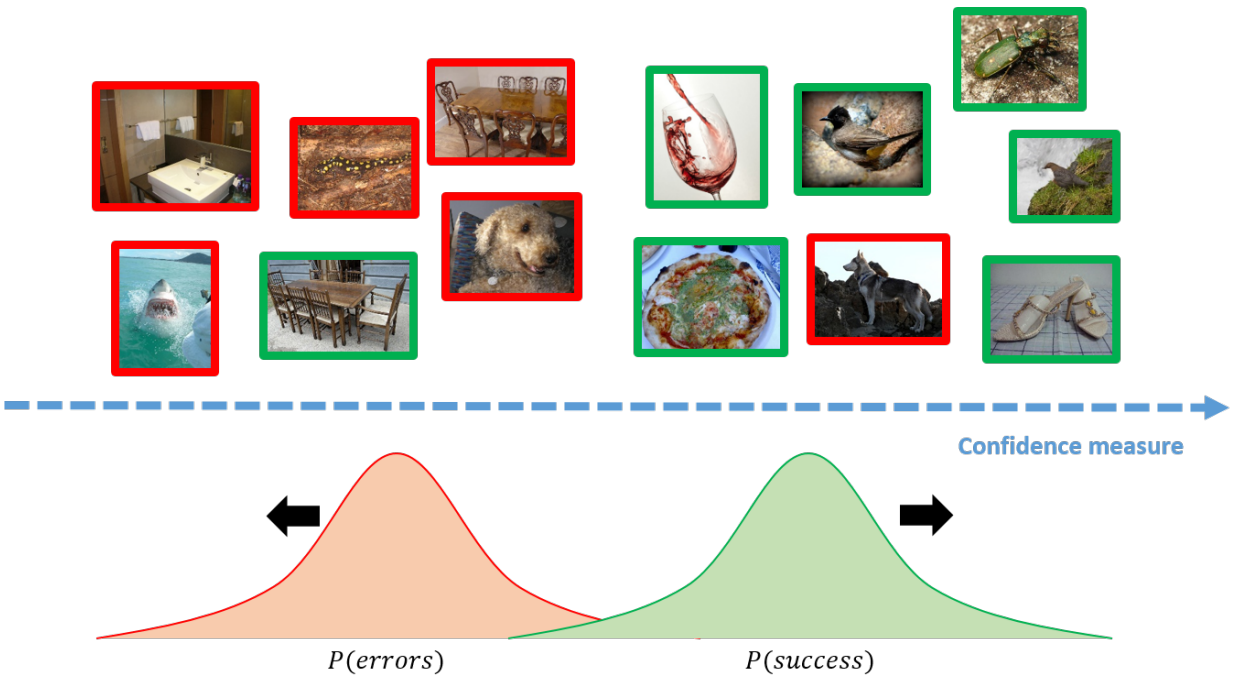
$$\hat{C}(x) \geq \tau \Rightarrow \text{accept prediction} \quad \text{else} \quad \text{reject prediction}$$

Confidence

- Simple baseline for deep neural networks :
  $\boldsymbol{MCP}(x) = \max_{k \in \mathcal{Y}} [f_w(x)]_k = \max_{k \in \mathcal{Y}} \ p(Y = k | w, x)$
- More advanced methods : *e.g.* MC dropout for classification, Deep Ensembles, etc.

Failure Prediction

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



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

Failure Prediction – A simple baseline

- MCP : $\boldsymbol{MCP}(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} p(Y = k | \boldsymbol{w}, \boldsymbol{x})$
  - Natural confidence criterion for classification
  - Measure the model's **confidence** in its prediction

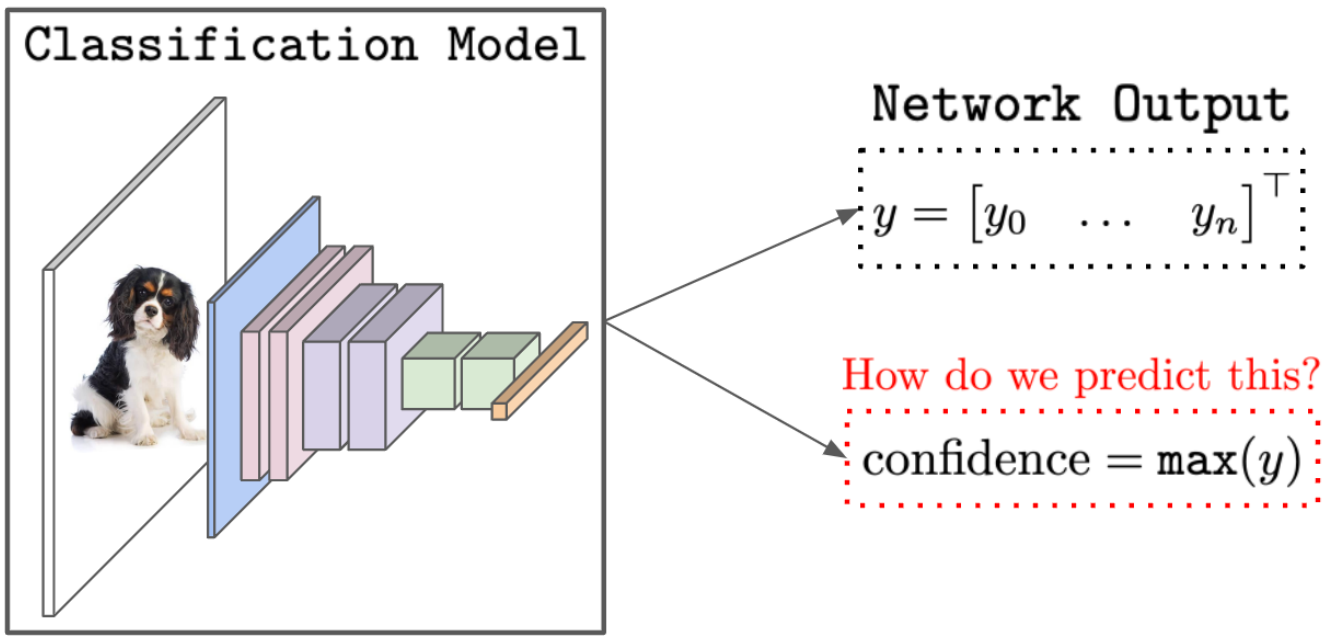
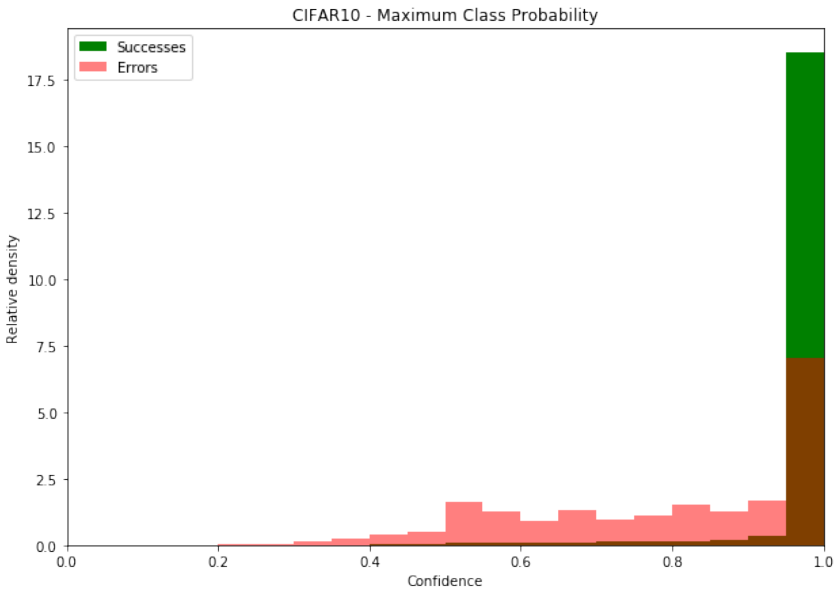
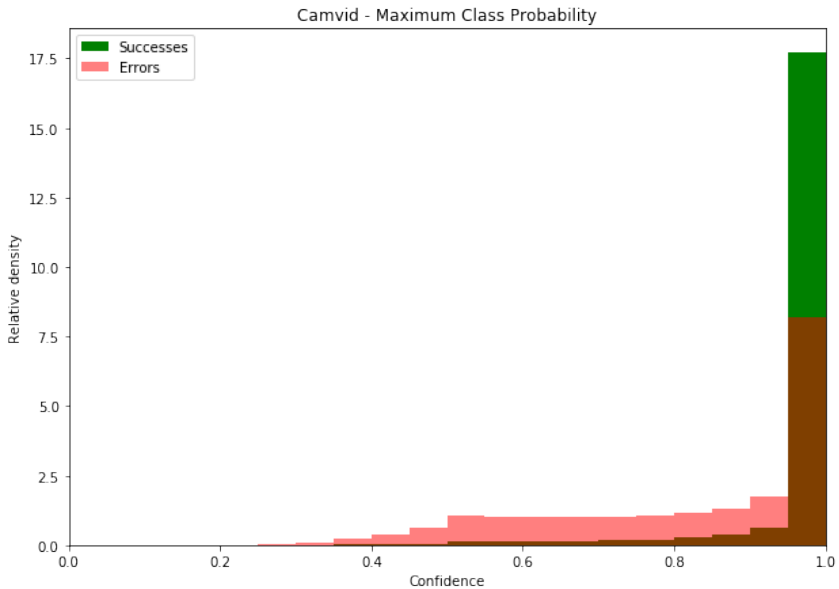


Figure – Confidence Calibration for Deep Networks : Why and How ? – medium

Failure Prediction – A simple baseline

- 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

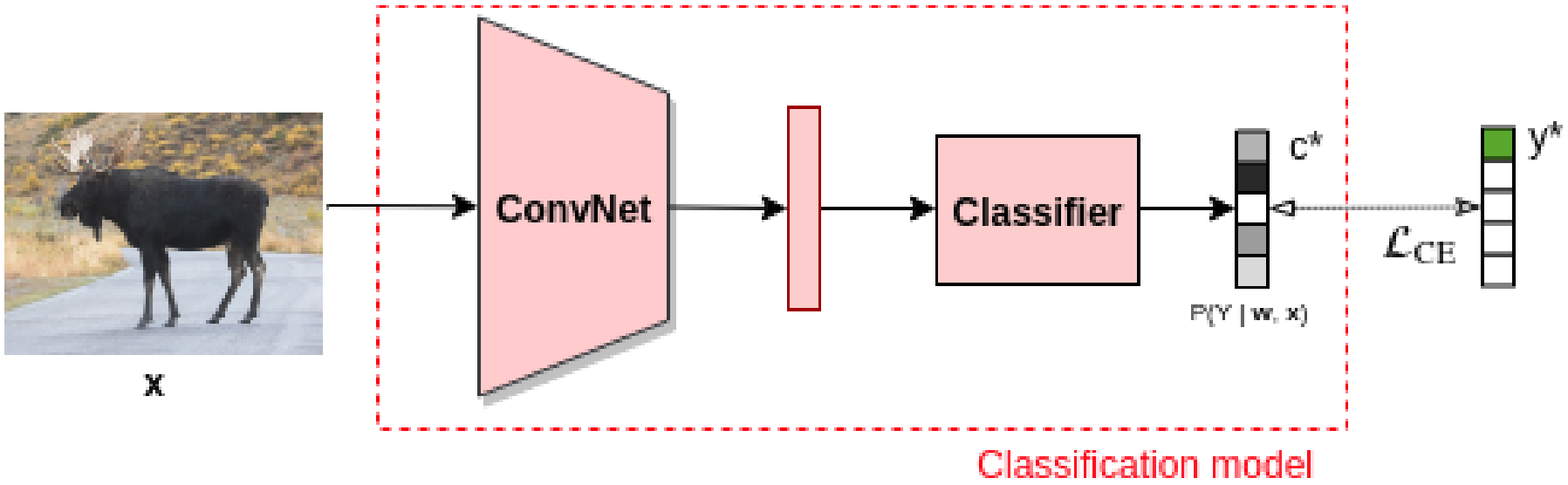




- **overlapping distributions** between successes vs. errors
  ⇒ hard to distinguish

Failure Prediction : Model Calibration in Deep Learning

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

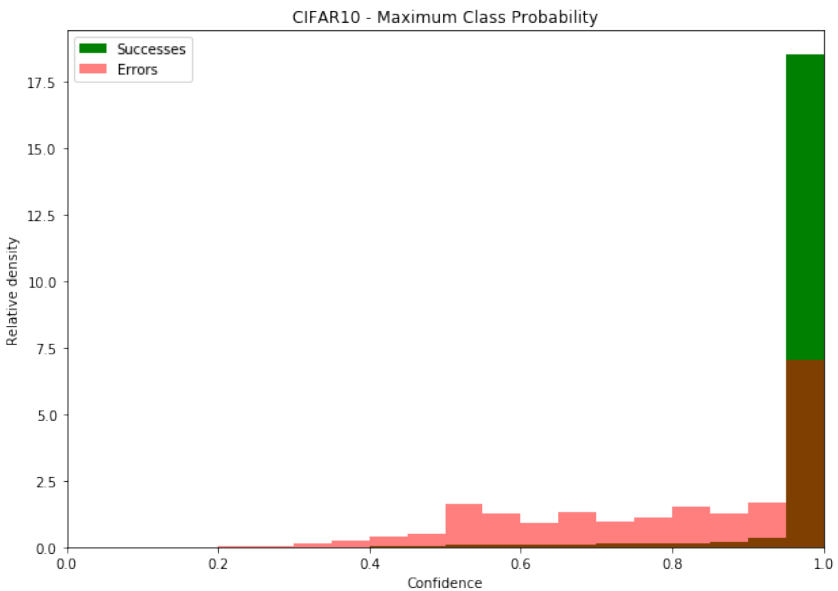


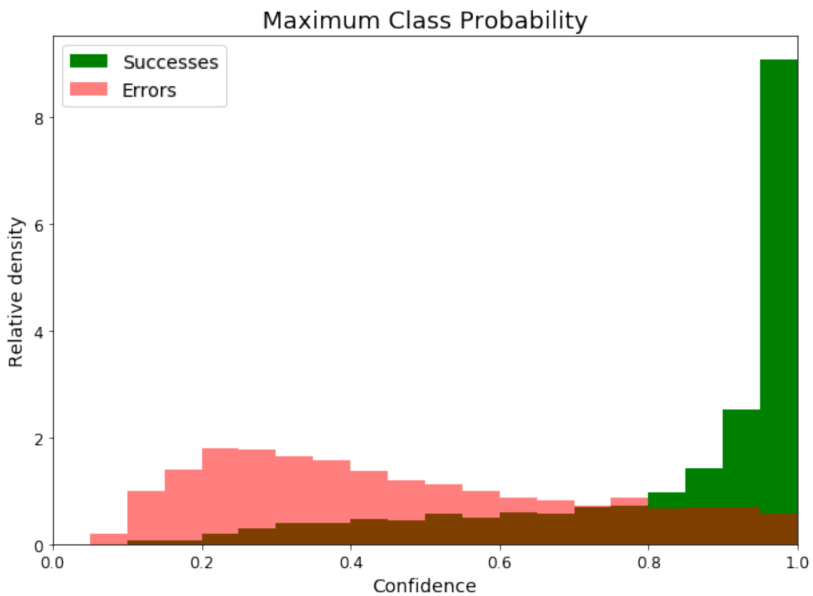
- Model prediction : $\hat{y} = \arg\max_{k \in \mathscr{Y}} p(Y = k | w, x)$
- Model confidence $\hat{C}(x)$:
  - Simple baseline for deep neural networks : $\boldsymbol{MCP}(x) = \max_{k \in \mathscr{Y}} p(Y = k | w, x)$
  - More advanced methods, *e.g.* MC dropout for classification
- **Threshold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

MCP, a sub-optimal ranking confidence measure

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

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





True Class Probability (TCP)

Definition

- True Class Probability (TCP) :

$$\boldsymbol{TCP}(\boldsymbol{x}, y^*) = \left[f_w(\boldsymbol{x})\right]_{y^*} = p(Y = y^* | \boldsymbol{w}, \boldsymbol{x})$$

- where y^* is the true label associated to $\boldsymbol{x}$

True Class Probability (TCP)

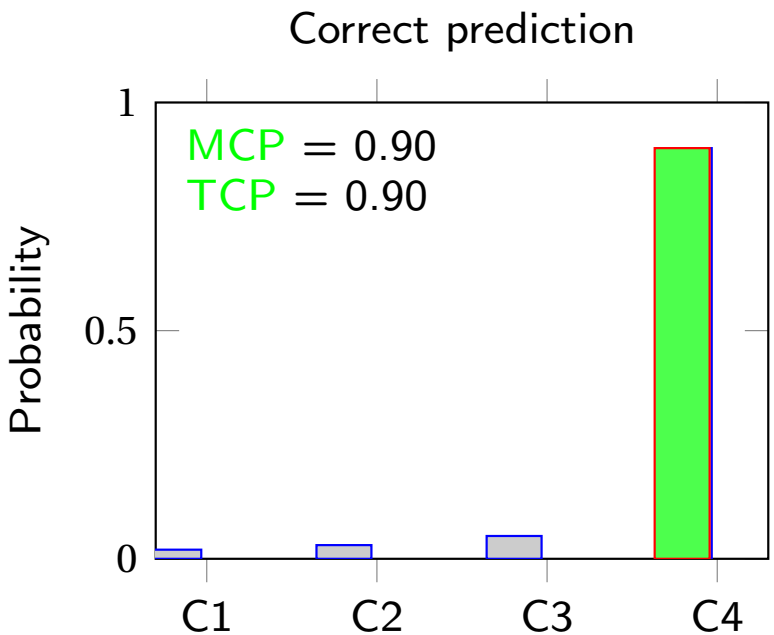
Definition

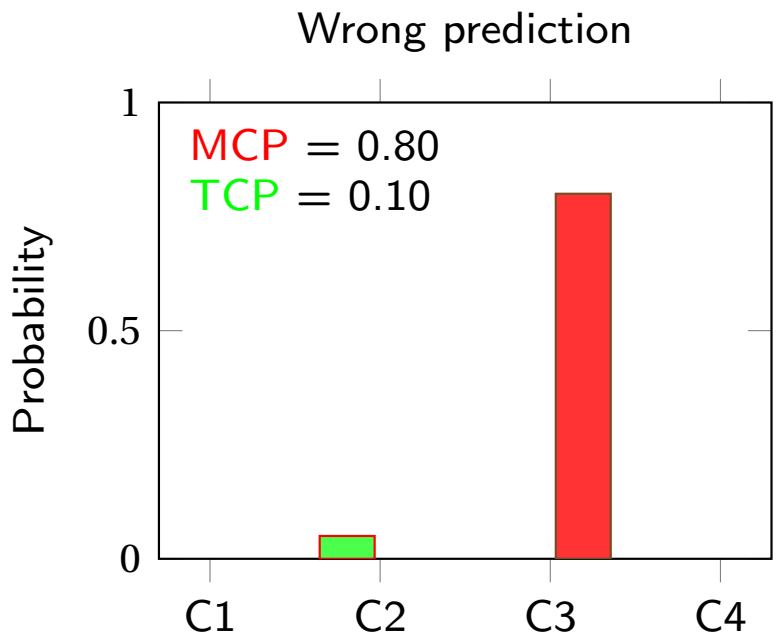
- True Class Probability (TCP) :

$$\boldsymbol{TCP}(\boldsymbol{x}, y^*) = \left[f_w(\boldsymbol{x})\right]_{y^*} = p(Y = y^* | \boldsymbol{w}, \boldsymbol{x})$$

- where y^* is the true label associated to $\boldsymbol{x}$
- Intuition :
  - If prediction is correct : $TCP(\boldsymbol{x}, y^*)$ is high
  - If prediction is incorrect : $TCP(\boldsymbol{x}, y^*)$ is low
  - $\hookrightarrow$ perfect confidence criterion for failure prediction
  - TCP measures the model's confidence in what should be the correct prediction

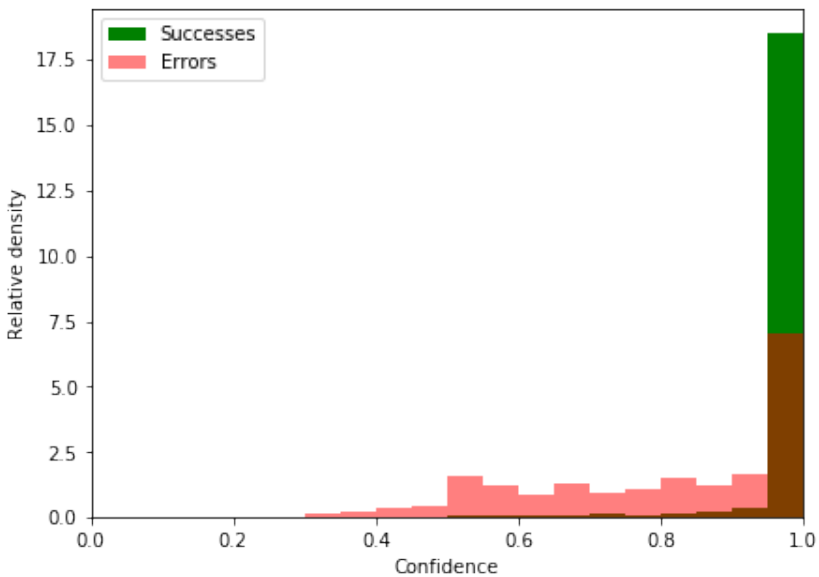
True Class Probability (TCP)



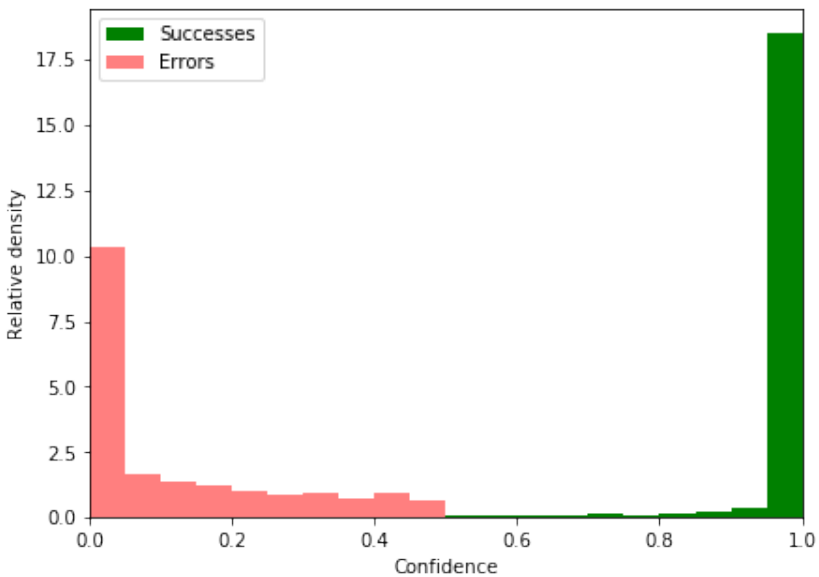


Failure Prediction with TCP [Corbière et al., 2019]

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



(a) Maximum Class Probability

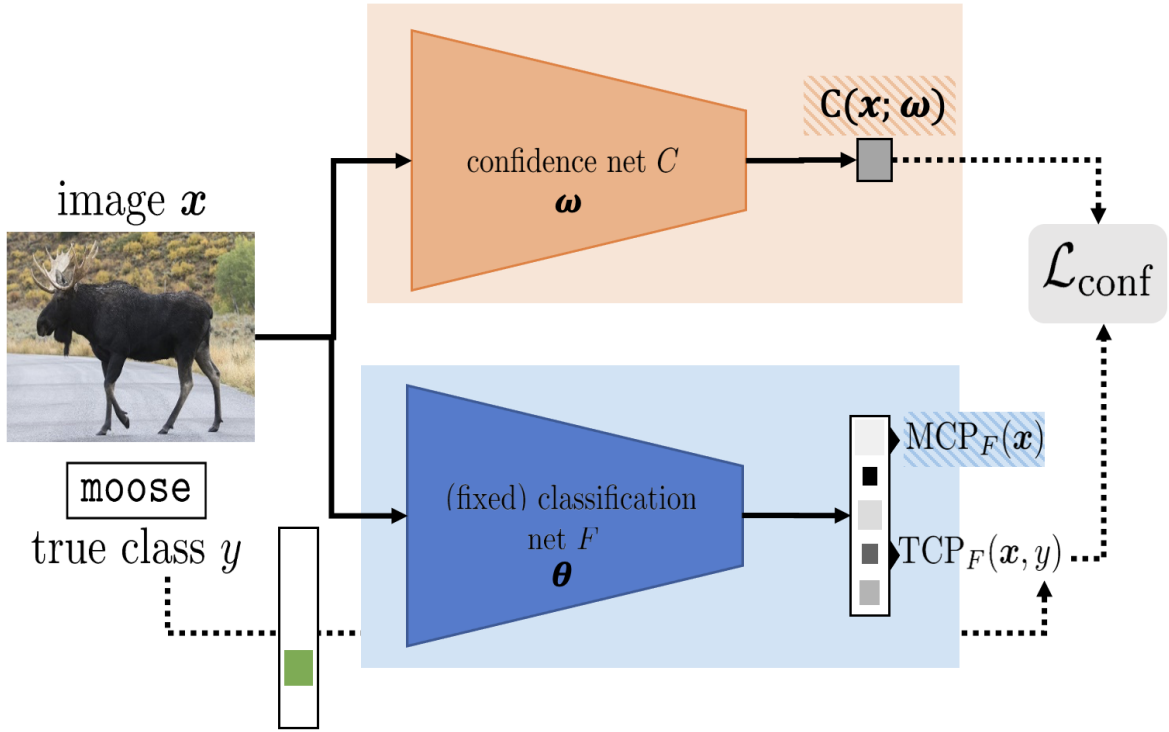


(b) Our Proposal (True Class Probability)

ConfidNet [Corbière et al., 2019]

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

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



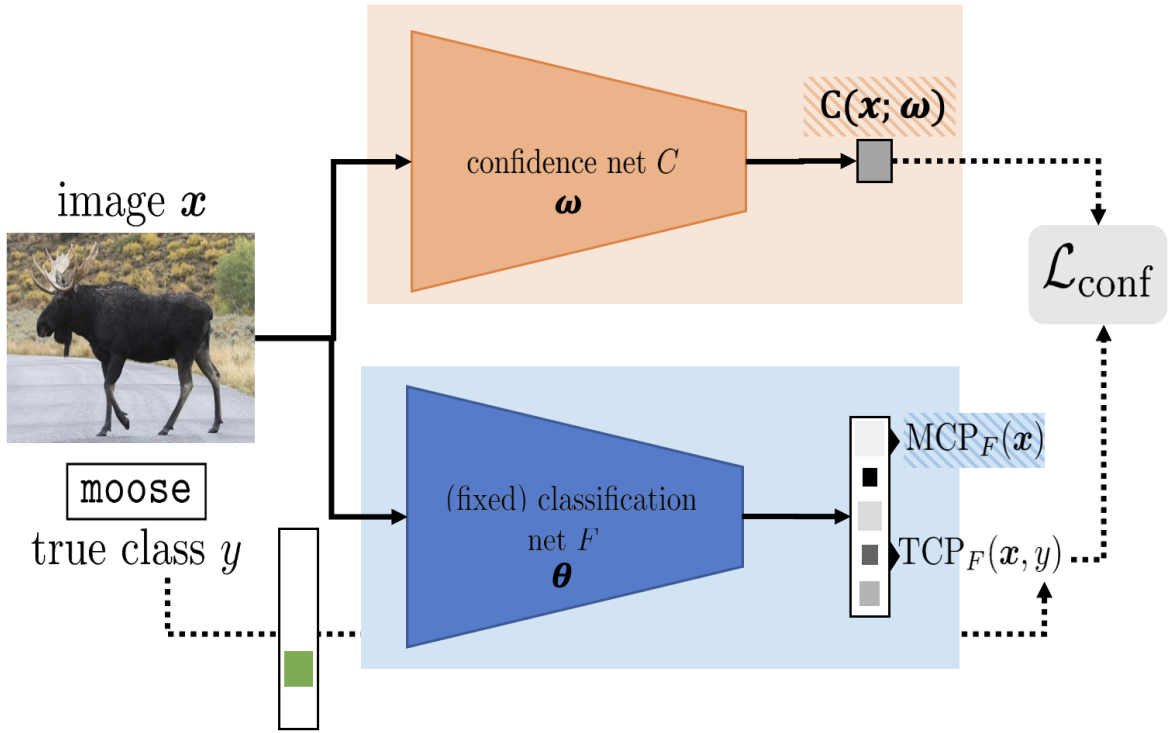
As $TCP(x, y^*) \in [0, 1]$, ℓ_2 loss to train ConfidNet :

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

ConfidNet [Corbière et al., 2019]

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

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



Could also be a Binary Cross Entropy loss :

$$\mathscr{L}_{conf}(\theta; \mathscr{D}) = \frac{1}{N} \sum_{i=1}^{N} -c^*(x_i, y_i^*) \log \hat{c}(x_i, \theta) - \left(1 - c^*(x_i, y_i^*)\right) \log(1 - \hat{c}(x_i, \theta))$$

Measuring Failure Prediction performance – AUPR

Precision–Recall

The Precision–Recall curve plots related to true labels and a confidence score :

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{vs} \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Measuring Failure Prediction performance – AUPR

Precision–Recall

The Precision–Recall curve plots related to true labels and a confidence score :

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad \text{vs} \quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

- Here, positive class : erroneous predictions
- Negative class : correct predictions
- TP : number of correctly detected errors
- FP : number of correct predictions incorrectly classified as errors
- FN : number of errors incorrectly classified as correct predictions

Measuring Failure Prediction performance – AUPR

Precision–Recall

The Precision–Recall curve plots related to true labels and a confidence score :

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad \text{vs} \quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

- Here, positive class : erroneous predictions
- Negative class : correct predictions
- TP : number of correctly detected errors
- FP : number of correct predictions incorrectly classified as errors
- FN : number of errors incorrectly classified as correct predictions

Precsision and Recall are defined for a given decision threshold on the confidence score.
Metrics are complementary :

- Precision : how many of the predicted errors are actual errors ?
- Recall : how many of the actual errors are detected ?

Measuring Failure Prediction performance – AUPR

Area under the Precision–Recall curve (AUPR)

The Precision–Recall curve plots related to true labels and a confidence score for all possible decision thresholds.
The AUPR is the **area under the Precision–Recall curve** :

$$\text{AUPR} \in [0, 1]$$

Interpretation : AUPR measures how well the model retrieves positive examples while keeping false positives low.

Measuring Failure Prediction performance – AUPR

Area under the Precision–Recall curve (AUPR)

The Precision–Recall curve plots related to true labels and a confidence score for all possible decision thresholds.
The AUPR is the **area under the Precision–Recall curve** :

$$\text{AUPR} \in [0, 1]$$

Interpretation : AUPR measures how well the model retrieves positive examples while keeping false positives low.

Remarks

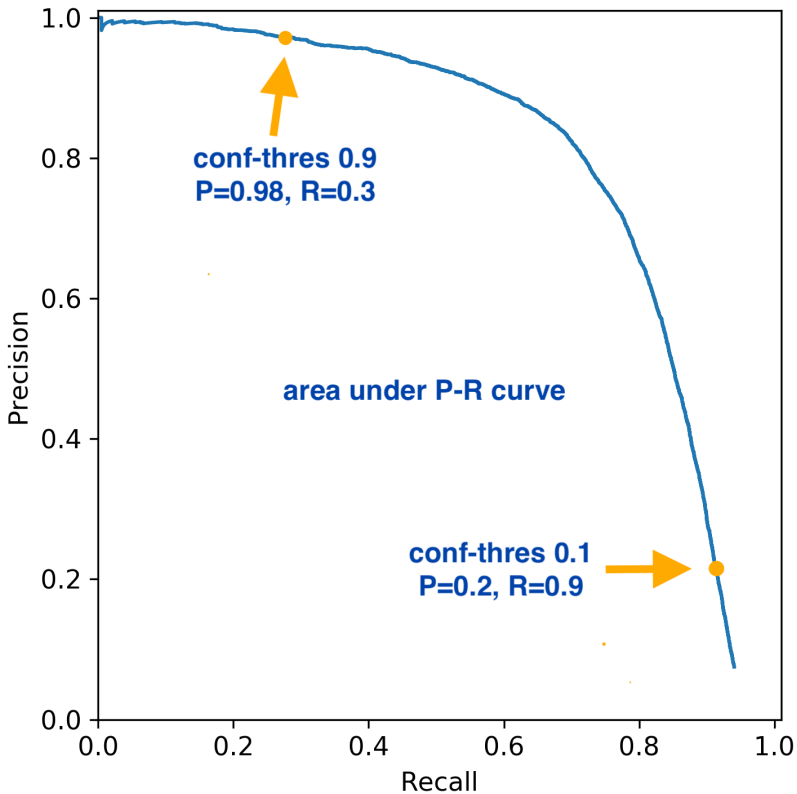
- Sensitive to class imbalance
- More informative than AUROC when positives are rare
- Depends on ranking quality, not calibration

Measuring Failure Prediction performance – AUPR

Area under the Precision–Recall curve (AUPR)

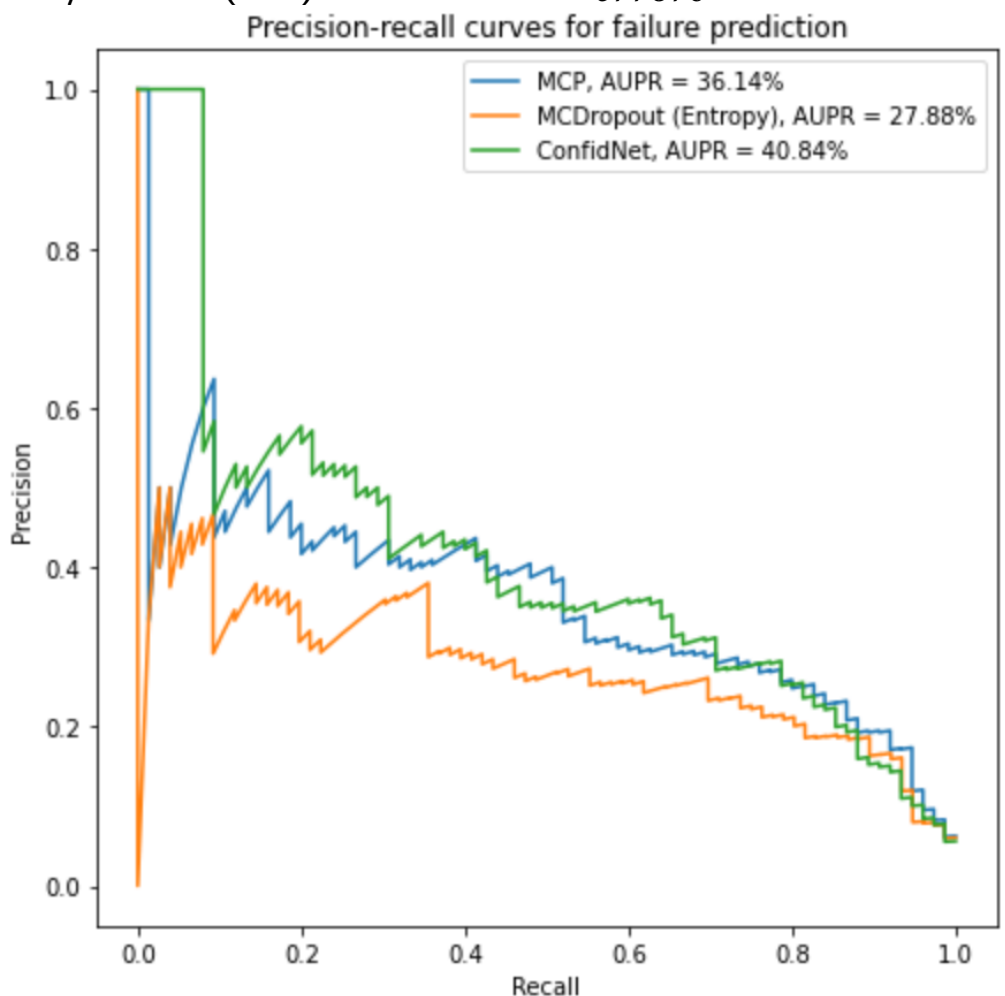
The Precision–Recall curve plots related to true labels and a confidence score for all possible decision thresholds.
The AUPR is the **area under the Precision–Recall curve** :

$$\text{AUPR} \in [0, 1]$$



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}



Measuring Failure Prediction performance – FPR95

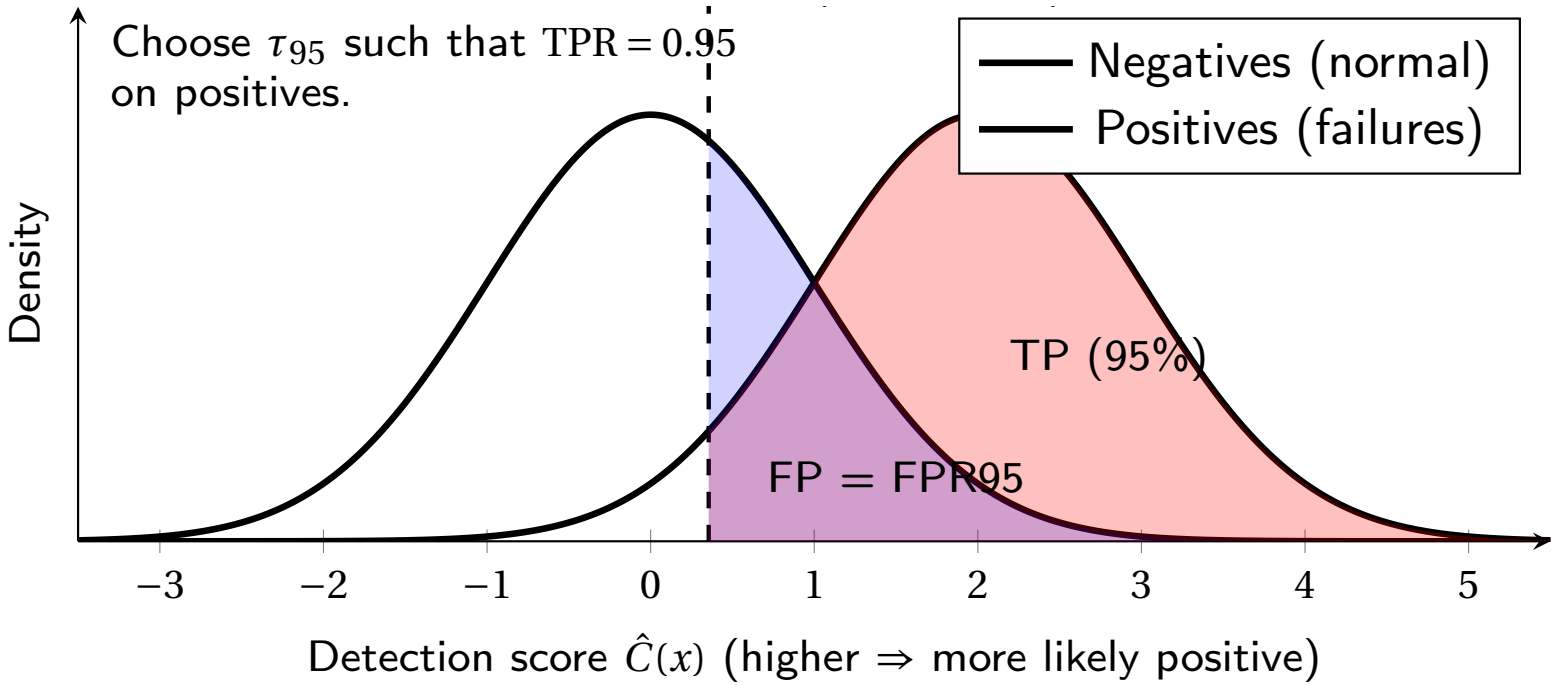
False Positive Rate at 95% True Positive Rate (FPR95)

FPR95 is the false positive rate when the true positive rate (recall) is fixed to 95% :

$$\text{FPR95} = \left.\frac{\text{FP}}{\text{FP} + \text{TN}}\right|_{\text{TPR}=0.95}$$

- Directly answers : "How many false alarms when we catch almost all positives ?"
- Lower FPR95 ⇒ better detector
- Threshold-dependent (unlike AUROC / AUPR)

Measuring Failure Prediction performance – FPR95



Extention to Segmentation

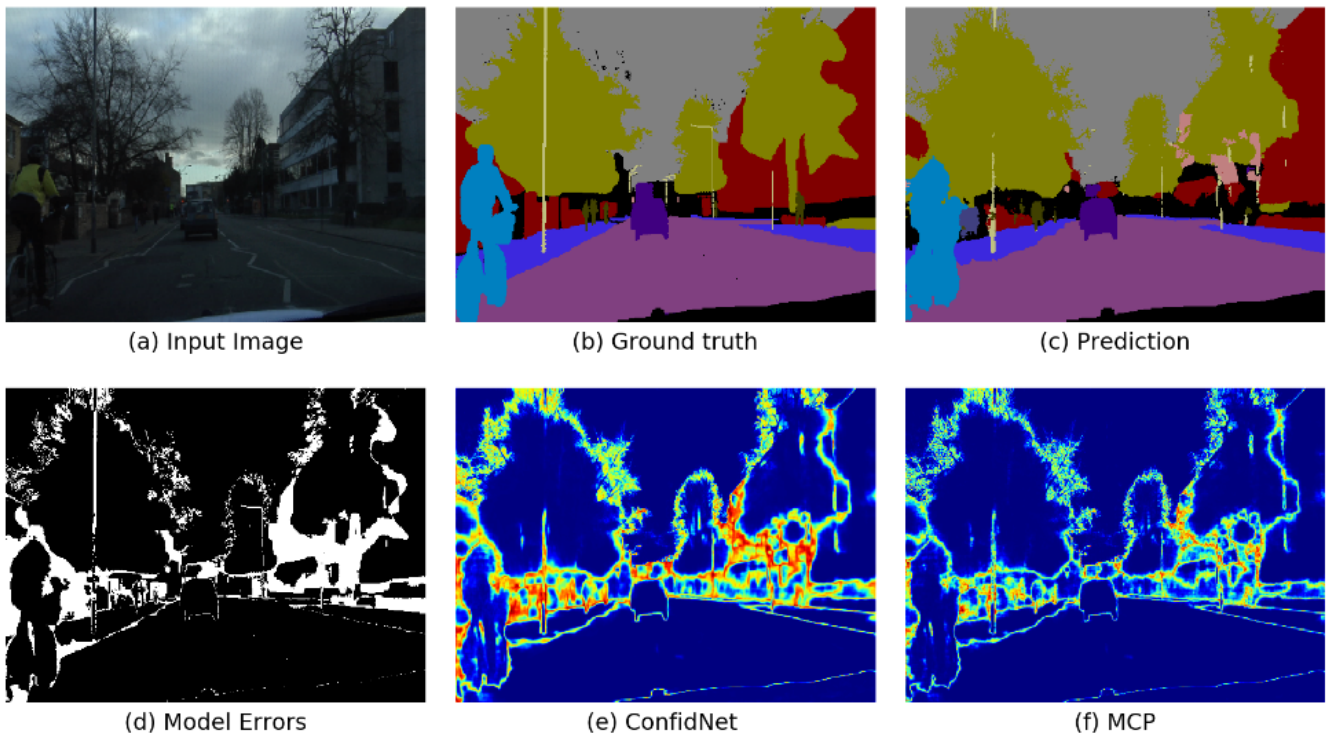


Figure – Failure Prediction for Semantic Segmentation using ConfidNet– [Corbière et al., 2019]

Failure prediction for VLM

- Vision-Language Models (VLM) are models that can process both visual and textual data.
- Examples : CLIP, SigLip, Flamingo
- High zero-shot performance on various tasks
- Uncertainty may arise from both the image and target concept

Maximum Concept Matching

Generalization of MCP to VLMs :

$$\boldsymbol{MCM}(\boldsymbol{x}, \mathcal{T}) = \max_{t \in \mathcal{T}} p(Y = t | \boldsymbol{w}, \boldsymbol{x})$$

where $\mathcal{T}$ is the set of possible concepts. For CLIP-like models, $p(Y = t|\boldsymbol{w}, \boldsymbol{x})$ is computed via cosine similarity between image and text embeddings :

$$p(Y = t|\boldsymbol{w}, \boldsymbol{x}) = \frac{\exp(\text{sim}(\phi_I(\boldsymbol{x}), \phi_T(\boldsymbol{t}))/\tau)}{\sum_{t' \in \mathcal{T}} \exp(\text{sim}(\phi_I(\boldsymbol{x}), \phi_T(\boldsymbol{t'}))/\tau)}$$

Recap : Zero-shot prediction with CLIP

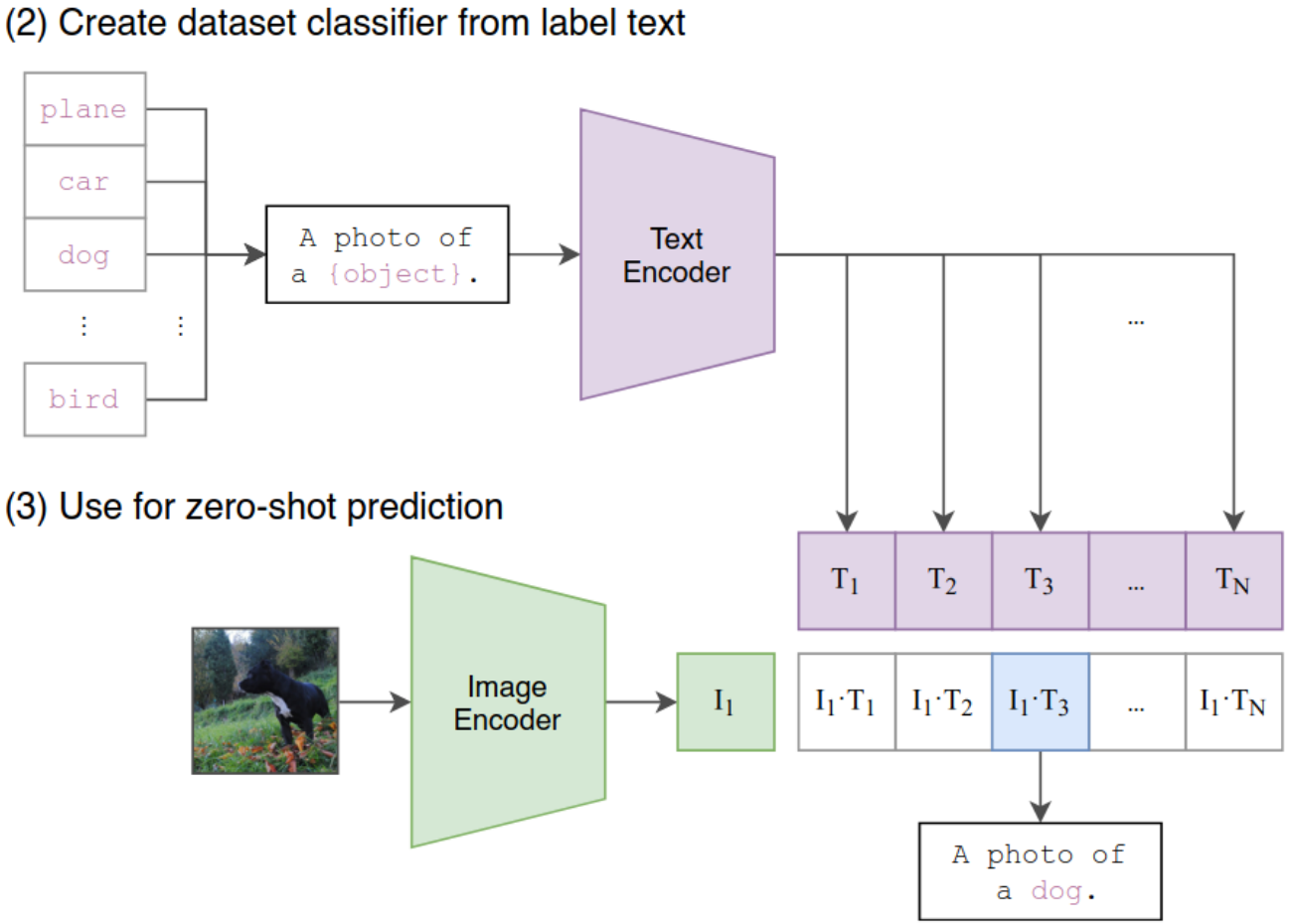


Figure – Zero-shot classification with CLIP – [Radford et al., 2021]

Failure Prediction with CLIP

- MCM an assign a high confidence to a wrong prediction
- New source of uncertainty : ambiguity between concepts
- The concept granularity impacts the failure prediction performance
- Previous method only consider image uncertainty and fail to capture concept uncertainty

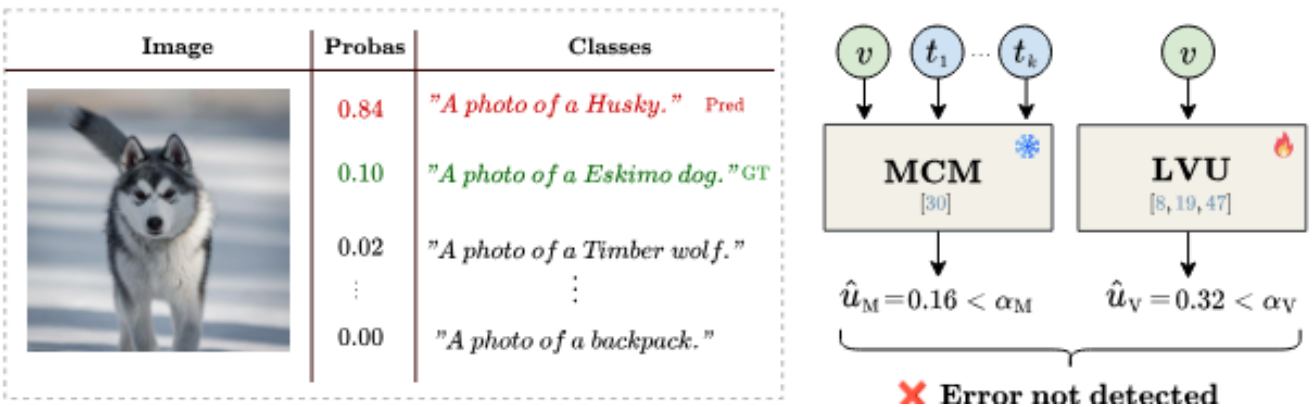


Figure – Failure Prediction with CLIP – [Lafon et al., 2025]

Learning uncertainty on CLIP

- Learning a failure prediction model on top of CLIP
- Confidence model trained to predict classify between correct and incorrect predictions
- Use both image and concept features

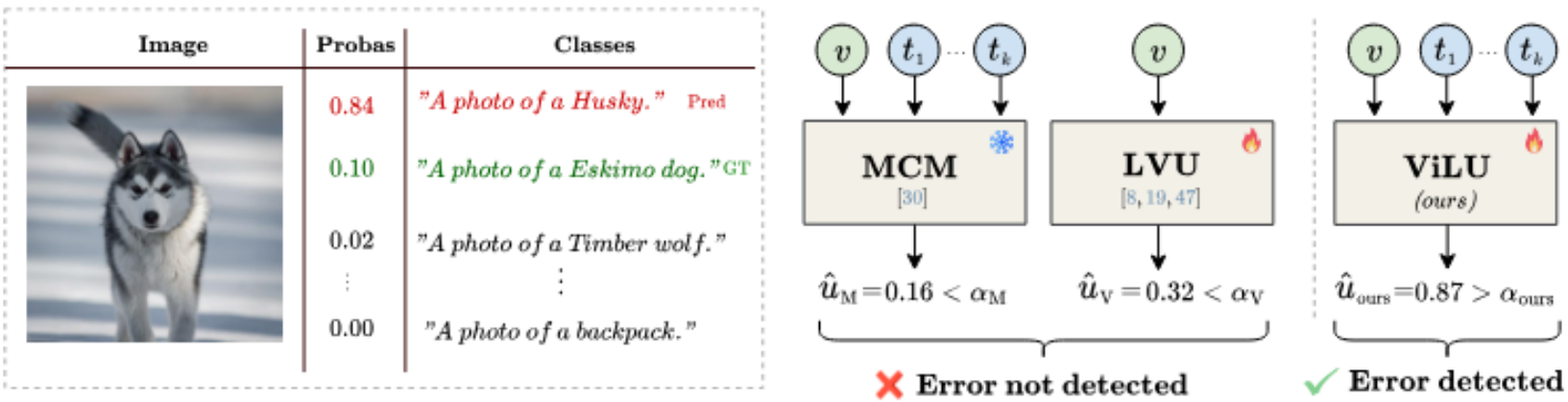


Figure – Failure Prediction with CLIP – [Lafon et al., 2025]

Learning uncertainty on CLIP

Recent Vision-Language Uncertainty framework (ViLu) [Lafon et al., 2025] :

- Contextualize uncertainty estimates by leveraging all textual representations
- Uncertainty-aware representation by combining
  - Visual embeddings $\phi_I(x) = z_v$
  - Predicted textual embedding $\phi_T(\hat{t}) = z_{\hat{t}}$ with $\hat{t} = \underset{t \in \mathcal{T}}{\arg\max}\, p(Y = t | w, x)$
  - Learned image conditioned textual embedding z_t^α

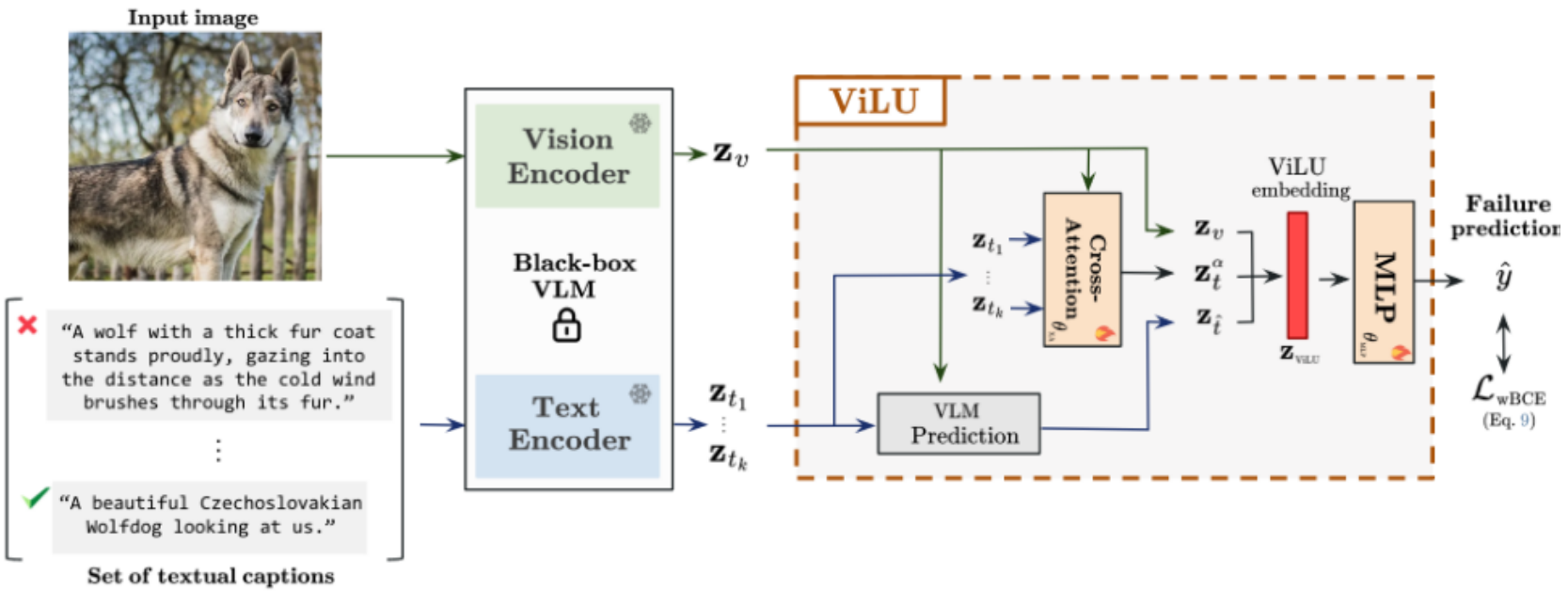


Figure – Vision-Language Uncertainty – [Lafon et al., 2025]

ViLu [Lafon et al., 2025] in a closed-set scenario

- Evaluate failure prediction performance on dataset with finite set of concepts
- ViLu Trained on training set of each dataset
- ViLu outperforms MCM and other baselines

	CIFAR-10 88.3%		CIFAR-100 68.6%		Caltech101 91.4%		Flowers102 64.0%		OxfordPets 85.1%		Food101 78.9%		ImageNet-1k 62.0%	
	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓
MCM [30]	89.9	52.1	82.7	67.3	88.1	68.7	86.6	68.0	87.2	59.9	86.4	63.3	80.8	71.3
TS [15] + MCM [30]	89.9	51.5	83.9	68.4	90.4	55.7	86.9	66.0	89.1	55.5	86.9	62.8	80.7	71.5
Entropy [38]	88.7	59.9	79.8	71.9	86.1	78.8	85.5	65.0	88.0	60.0	86.1	65.0	78.3	76.8
Doctor [14]	89.5	56.5	82.3	69.7	88.7	66.5	86.7	63.9	88.9	56.6	86.8	63.4	80.3	72.9
Rel-U [13]	86.2	54.4	81.0	68.2	90.2	58.5	90.0	47.3	83.5	59.3	81.8	73.4	75.1	85.0
LVU [8, 19, 47]	96.6	21.2	80.3	68.5	89.8	50.9	90.5	38.3	84.1	55.7	82.7	69.9	78.7	77.0
BayesVLM [2]	92.6	44.9	87.0	60.3	94.0	37.4	87.3	62.4	89.5	60.3	87.8	60.3	81.5	70.3
ViLU (*Ours*)	**98.3**	**7.7**	**91.5**	**35.4**	**96.7**	**18.2**	**98.7**	**5.1**	**94.4**	**24.5**	**94.8**	**28.5**	**89.5**	**47.4**

Figure – Vision-Language Uncertainty – [Lafon et al., 2025]

ViLu [Lafon et al., 2025] in a open-set scenario

- Evaluate failure prediction performance on free concepts such as captions at training time
- Failure : wrong concept showing greater similarity than the correct caption
- The more large and diverse the training captions, the better the failure prediction performance

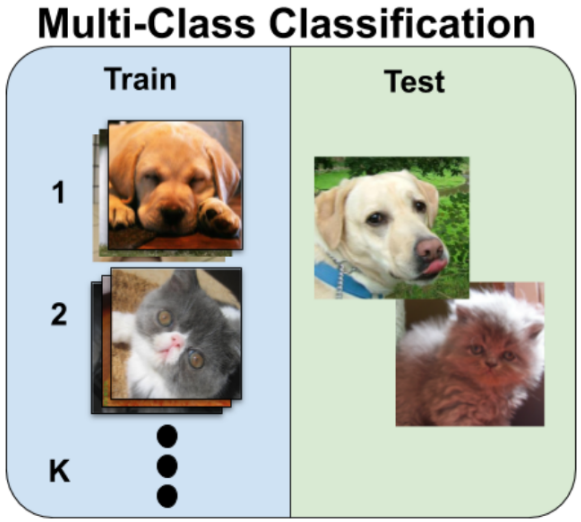
	CC3M 58.8%		CC12M 73.5%		LAION-400M 90.5%	
	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓
MCM [30]	83.9	69.0	88.8	58.8	91.7	50.2
Entropy [38]	82.5	73.3	87.7	63.0	89.4	62.5
Doctor [14]	83.7	70.1	88.6	59.9	91.2	54.5
LVU [8, 19, 47]	69.3	82.5	74.4	76.5	80.2	72.3
BayesVLM [2]	87.1	62.6	90.9	53.3	95.1	26.4
ViLU (*Ours*)	**91.4**	**41.1**	**95.2**	**25.2**	**97.3**	**21.2**

Figure – Vision-Language Uncertainty – [Lafon et al., 2025]

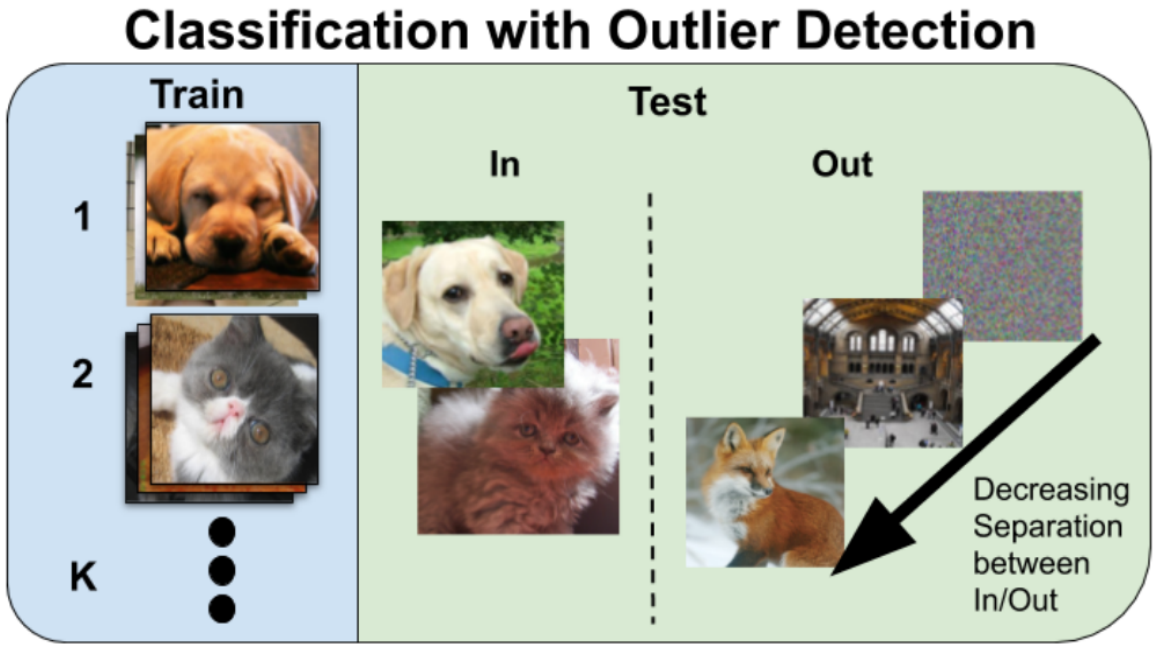
Outline

Out of Distribution Detection

- Standard classification : same classes during train and test



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



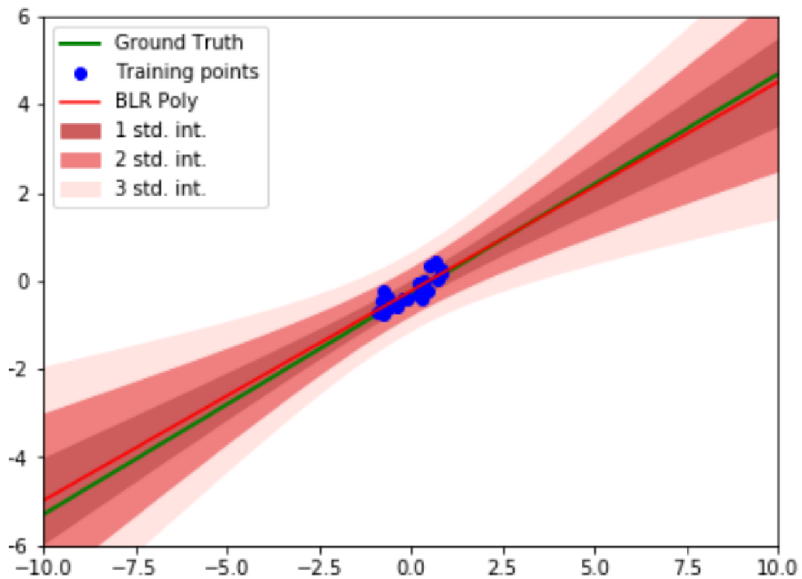
OOD Detection

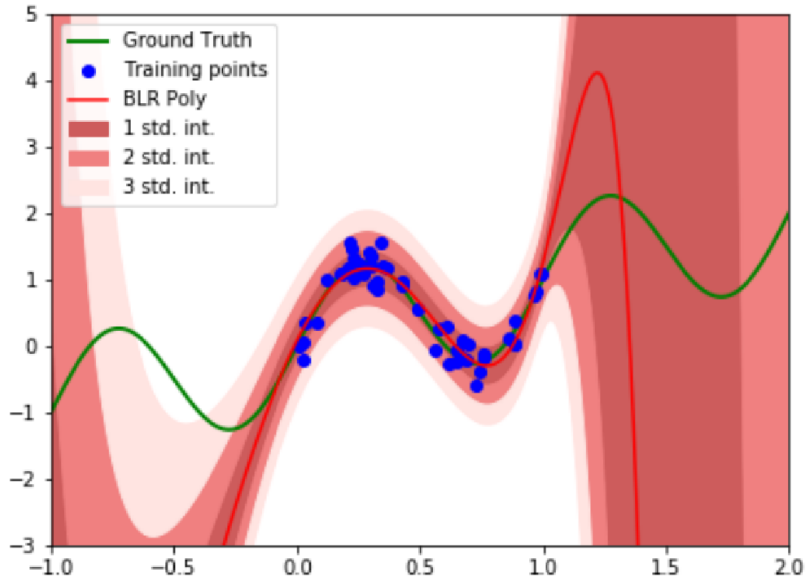
Problem statement

- Given a trained classifier $f(x)$ on in-distribution dataset $\mathscr{D}_{in}$, detect if a test sample x^* is from $\mathscr{D}_{in}$ or not
- Use a confidence score $\hat{C}(x^*)$ to decide if x^* is OOD or not :
  - If $\hat{C}(x^*) < \tau$, then x^* is OOD
  - Else, x^* is in-distribution
- Threshold τ can be chosen to reach a desired True Positive Rate (TPR) on a validation set

Out of Distribution Detection Metrics

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

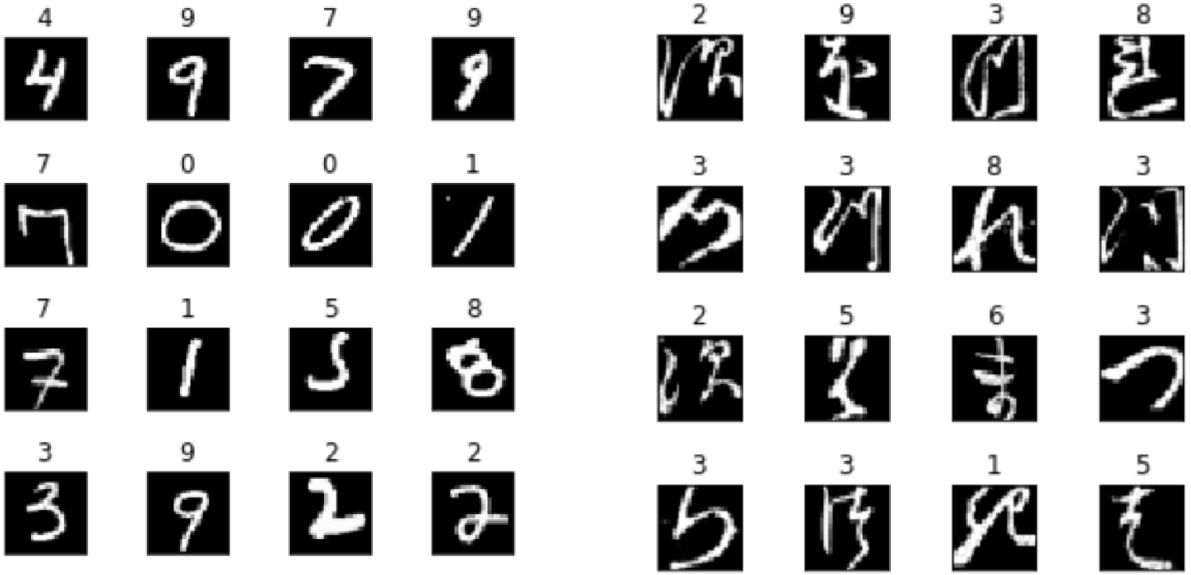




- **In classification :** estimate dispersion (*e.g.* mutual info) for Bayesian models (*e.g.* MC dropout)
- **Baseline solution :** assume that logits (before soft-max) smaller for out- than for in-distribution samples

Out of Distribution Example

- Model trained on MNIST - Out : Kuzushiii-MNIST



- Other typical benchmark :
  - ID : CIFAR-10 – OOD : CIFAR-100, SVHN, LSUN, TinyImageNet
  - ID : CIFAR-100 – OOD : CIFAR-10, SVHN, LSUN, TinyImageNet
  - ID : ImageNet – OOD : iNaturalist, SUN, Places, Textures

Out of Distribution Detection

Evaluation of OOD detection methods

- Standard metrics for OOD detection [Hendrycks and Gimpel, 2016] :
  - AUPR : Area Under Precision-Recall curve
  - FPR at 95% TPR : False Positive Rate when True Positive Rate is at 95%

Learning OOD

Outlier Exposure (OE) [Hendrycks et al., 2019]

- Training with in-distribution AND OOD samples :

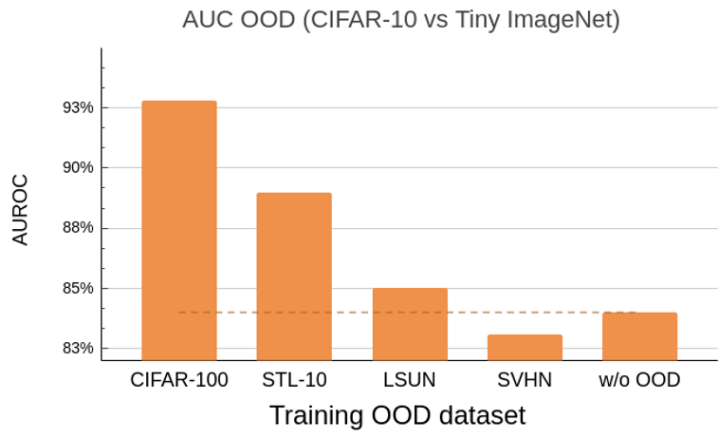
$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{in}}\left[\mathcal{L}(f(x),y)\right]+\lambda\mathbb{E}_{x'\sim\mathcal{D}_{out}^{OE}}\left[\mathcal{L}_{OE}(f(x'))\right]$$

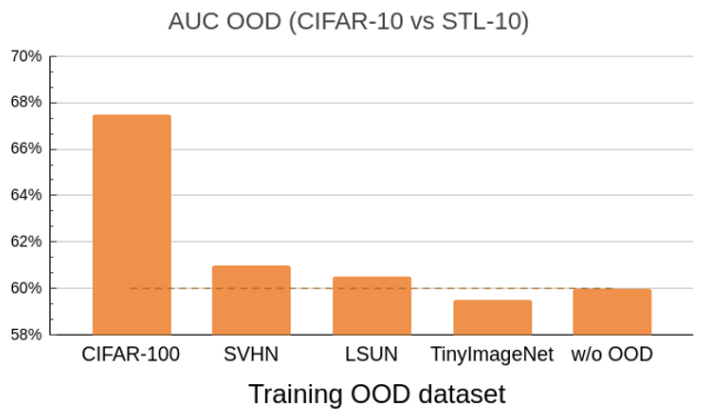
- *e.g.* $\mathcal{L}(f(x),y)$ cross-entropy,
  $\hookrightarrow$ encourages high confidence on in-distribution samples
- $\mathcal{L}_{OE}(f(x')$ KL wrt uniform class distribution
  $\hookrightarrow$ encourages low / uniform confidence on OOD samples

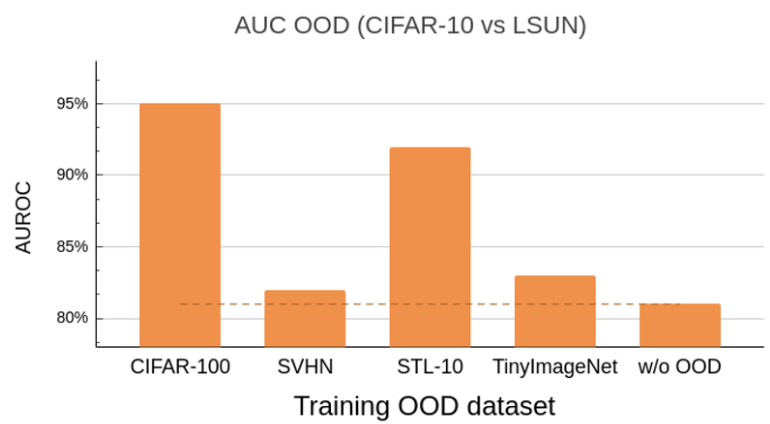
Out of Distribution Detection

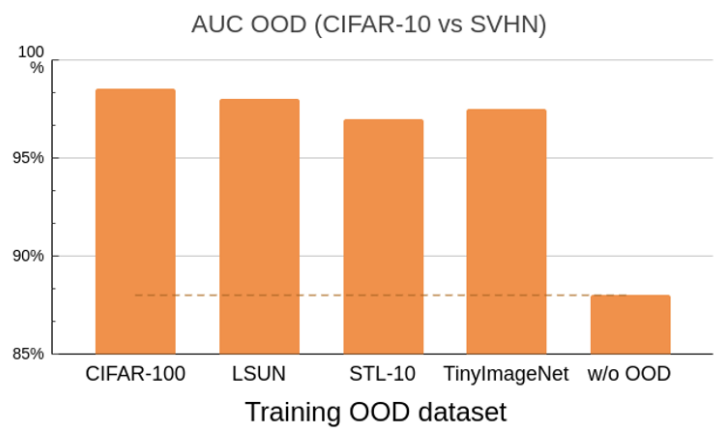
Relevance of using OOD training datasets ?

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







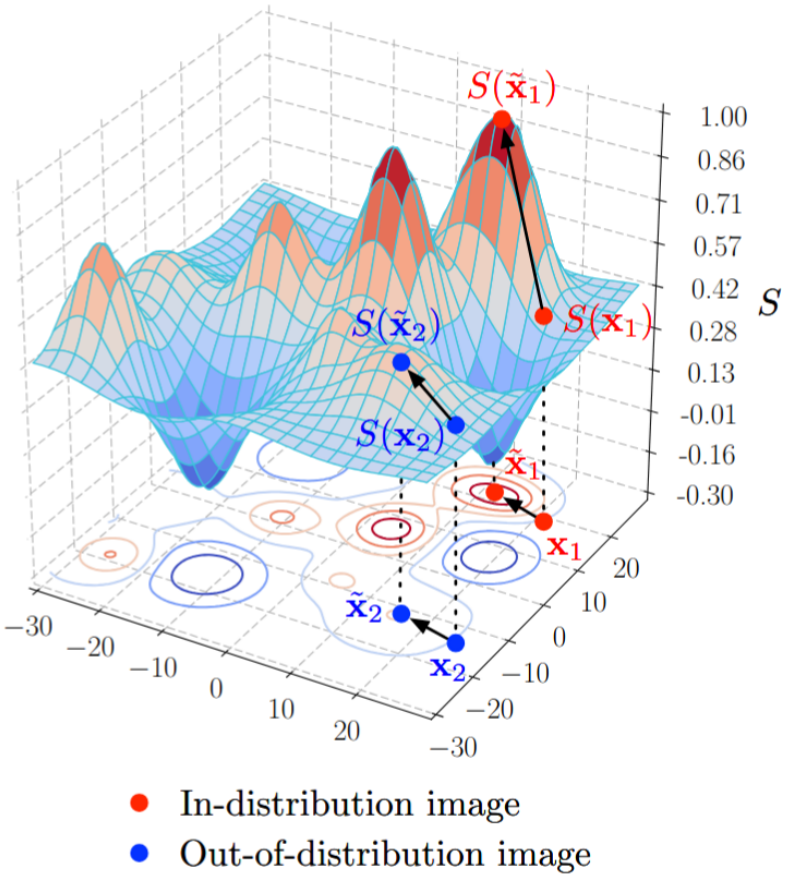


Post-hoc OOD detection methods

- Classifier based methods :
  - Maximum Class Probability (MCP) [Hendrycks and Gimpel, 2016].
  - ODIN [Liang et al., 2018].
  - REACT [Sun et al., 2021], ASH.
- Bayesian methods :
  - MC Dropout.
  - Deep Ensembles [Fort et al.,].
- Density based methods :
  - Gaussian Mixture Models [Lee et al., 2018].
  - Normalizing Flows.
- Distance based methods :
  - KNN in feature space.
  - GMM in feature space.
- Subspace based methods :
  - ViM [Wang et al., 2022].

Out-of-DIstribution detector for Neural networks – ODIN

- ODIN [Liang et al., 2018]
- Modify inputs so that maximizing differences between in and out samples
  - Temperature scaling, as in [Guo et al., 2017] :
    $$P(\hat{y}_k) = \frac{e^{s_k/T}}{\sum_{k'=1}^{K} e^{s_{k'}/T}}$$
  - 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



Mahalanobis-based OOD detector

Mahalabonis-based OOD detector [Lee et al., 2018]

- Fit a class-conditional Gaussian distribution $\mathcal{N}(\mu_c, \mathbf{\Sigma}_c)$ to the features of each class c at different layers of the network
- Compute Mahalanobis distance between test point embedding z and each class-conditional Gaussian :

$$D_c(z) = \sqrt{(z - \mu_c)^T \mathbf{\Sigma}_c^{-1} (z - \mu_c)}$$

- Use the minimum Mahalanobis distance as confidence score for OOD detection :

$$\hat{C}(z) = -\min_c D_c(z)$$

Mahalanobis-based OOD detector

Mahalabonis-based OOD detector [Lee et al., 2018]

- Fit a class-conditional Gaussian distribution $\mathcal{N}(\mu_c, \mathbf{\Sigma}_c)$ to the features of each class c at different layers of the network
- Compute Mahalanobis distance between test point embedding z and each class-conditional Gaussian :

$$D_c(z) = \sqrt{(z-\mu_c)^T \mathbf{\Sigma}_c^{-1} (z-\mu_c)}$$

- Use the minimum Mahalanobis distance as confidence score for OOD detection :

$$\hat{C}(z) = -\min_c D_c(z)$$

- Extension : Self-supervised learning when no labels available [**?**]
  - SimCLR [**?**] or InfoNCE [**?**] losses to learn clustered representations
  - Generate pseudo-labels using clustering (*e.g.* k-means)
  - Fit class-conditional Gaussian distributions to the features of each class at different layers of the network

KNN-based OOD detector

KNN-based OOD detector [Sun et al., 2022]

- For a test point x^*, compute its embedding z^* at a given layer of the network
- Compute the Euclidean distance between z^* and all training embeddings z_i
- Use the distance to the K-th nearest neighbor as confidence score for OOD detection :

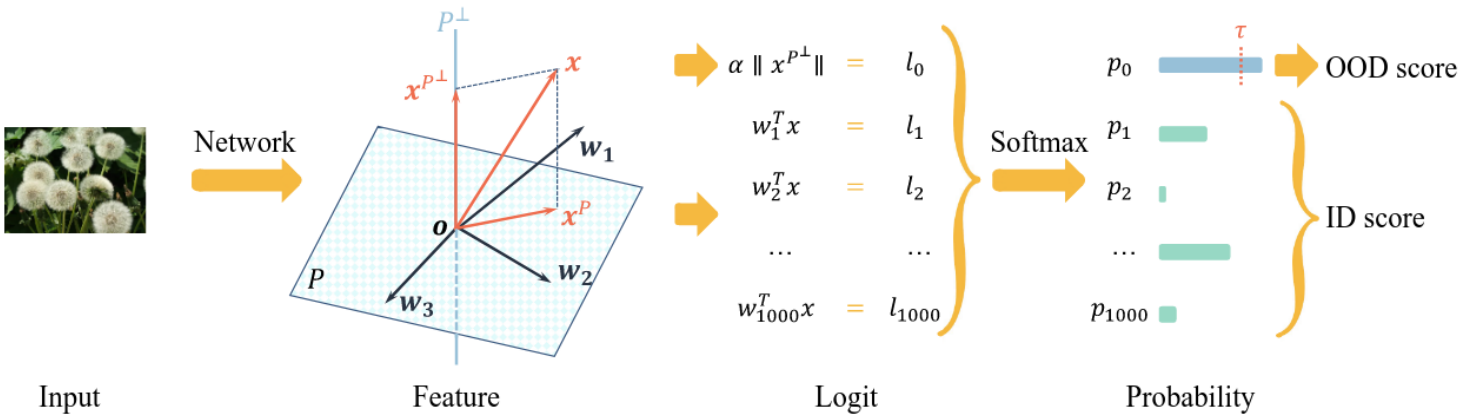
$$\hat{C}(z^*) = -\text{dist}(z^*, z_{(K)})$$

- Extension : Contrastive learning
  - Contrastive loss (SimCLR [?], SupCon loss [?]) to cluster representations
  - Use KNN-based OOD detection on learned representations

Virtual-logit Matching (ViM) OOD detector

ViM OOD detector [Wang et al., 2022]

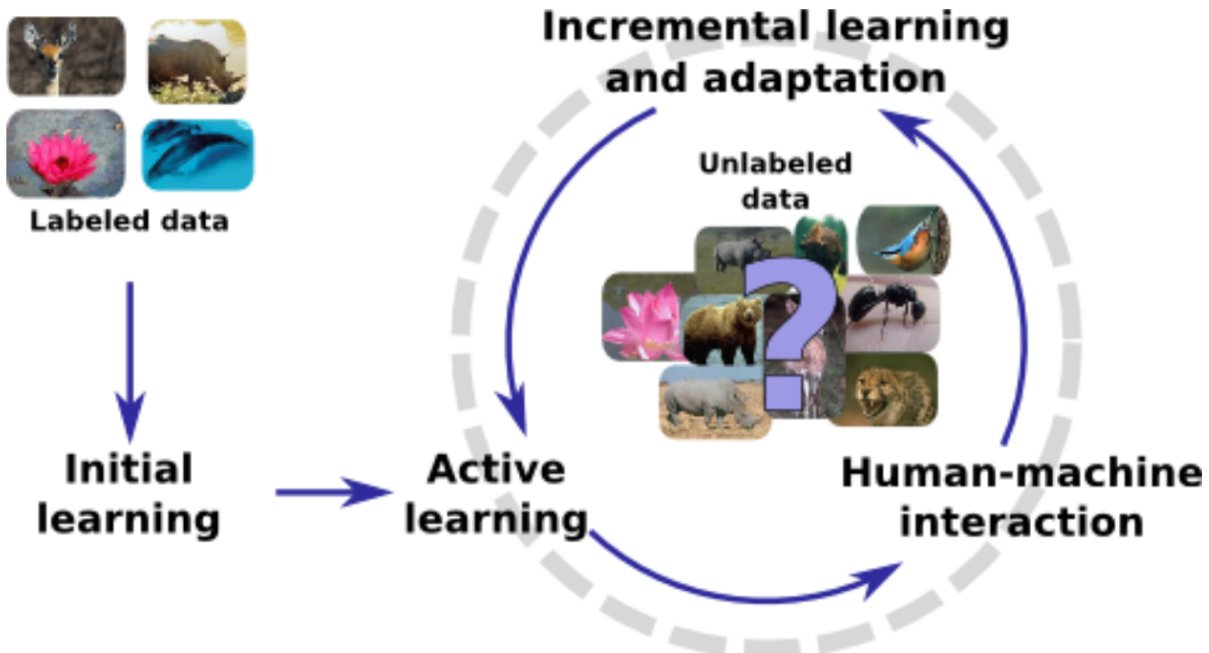
- Decompose the feature space $\mathbf{Z}$ into two subspaces :
  - $\mathbf{R}$: spanned by the smallest principal components of $\mathbf{Z}^T\mathbf{Z}$
- ViM score for new test point x^* with embedding $\mathbf{z}^*$:
  - Projection on the residual subspace : $\mathbf{z}^*\mathbf{R}\mathbf{R}^T\mathbf{z}^*$
  - Normalization of the score with respect to the maximum logits
- Use the ViM score as confidence score for OOD detection



Outline

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]

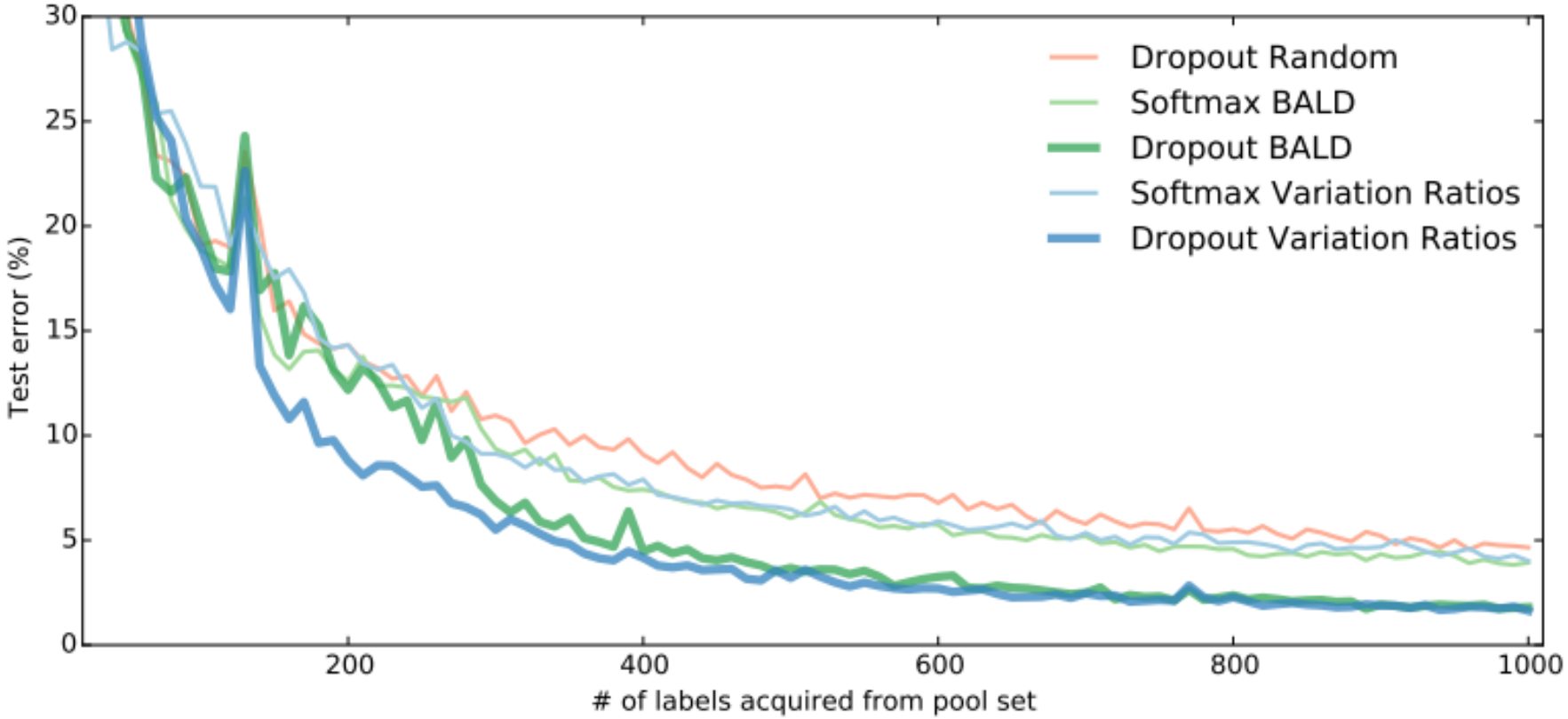
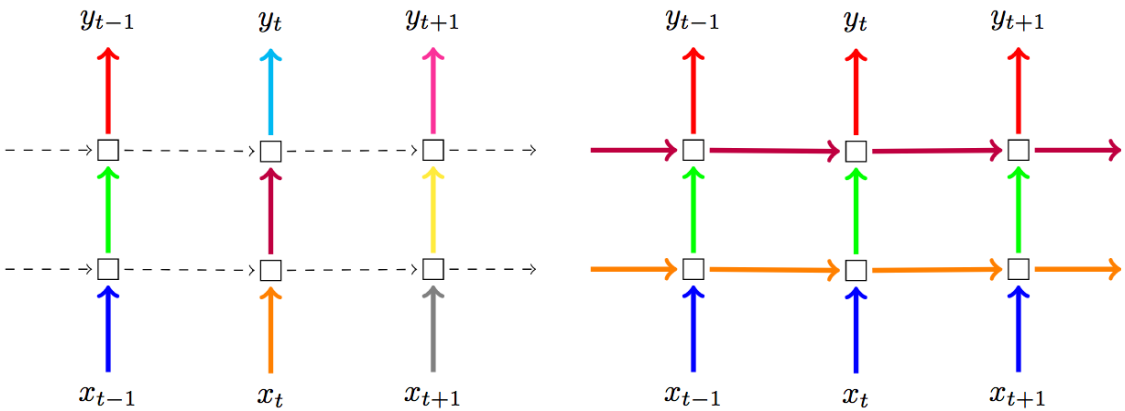


Fig. 5.1 **Test error on MNIST as a function of number of labels acquired from the pool set.** Two acquisition functions (*BALD* and *Variation Ratios*) evaluated with two approximating distributions — delta (*Softmax*) and Bernoulli (*Dropout*) — are compared to a *random* acquisition function.

Applications

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

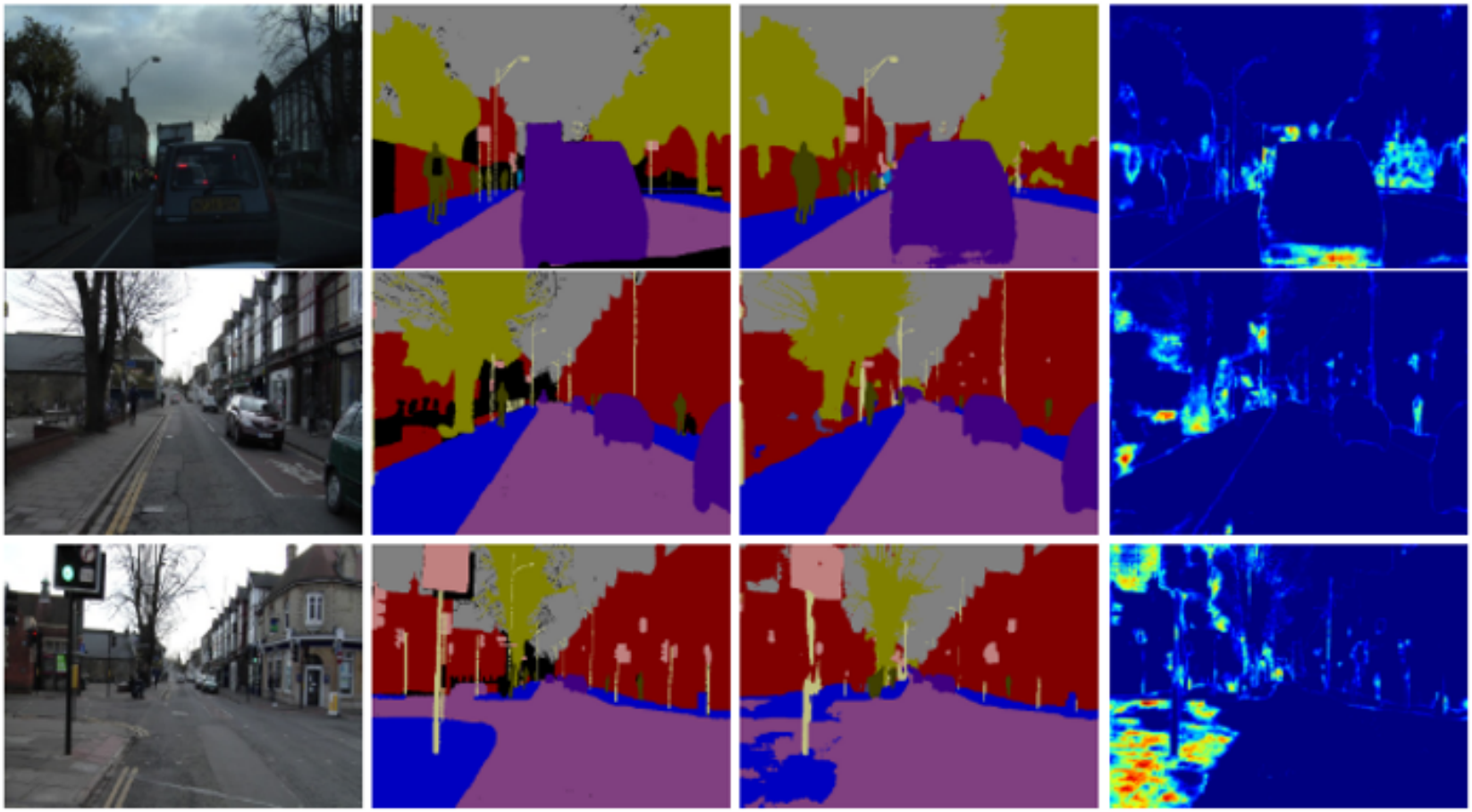


(a) Naive dropout RNN (b) Variational RNN

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

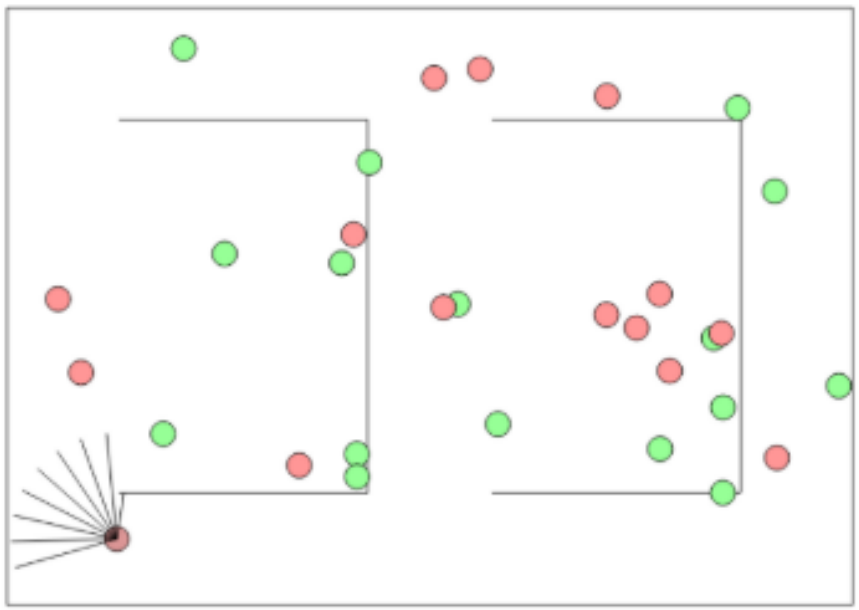


Fig. 5.3 Depiction of the reinforcement learning problem used in the experiments. The agent is in the lower left part of the maze, facing north-west.

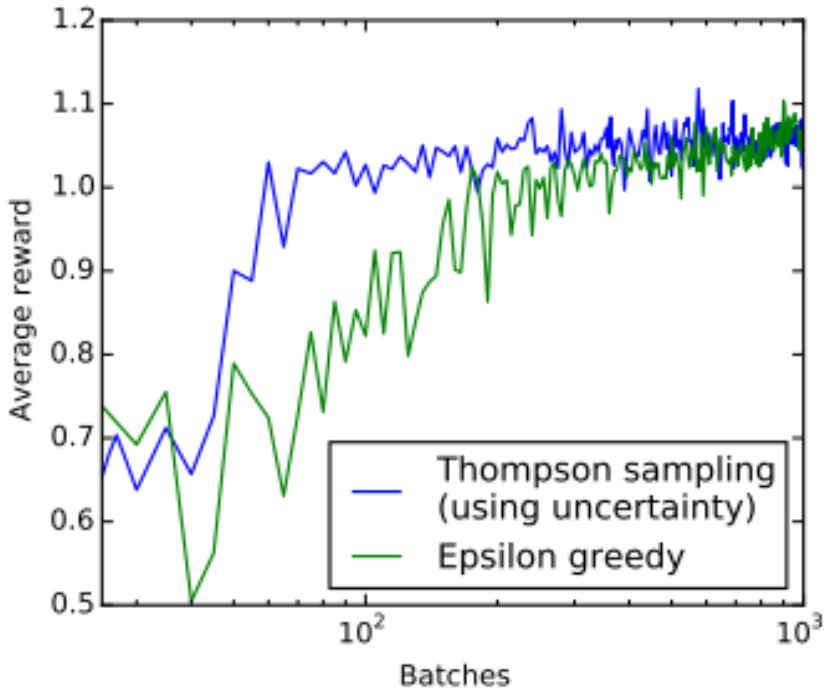
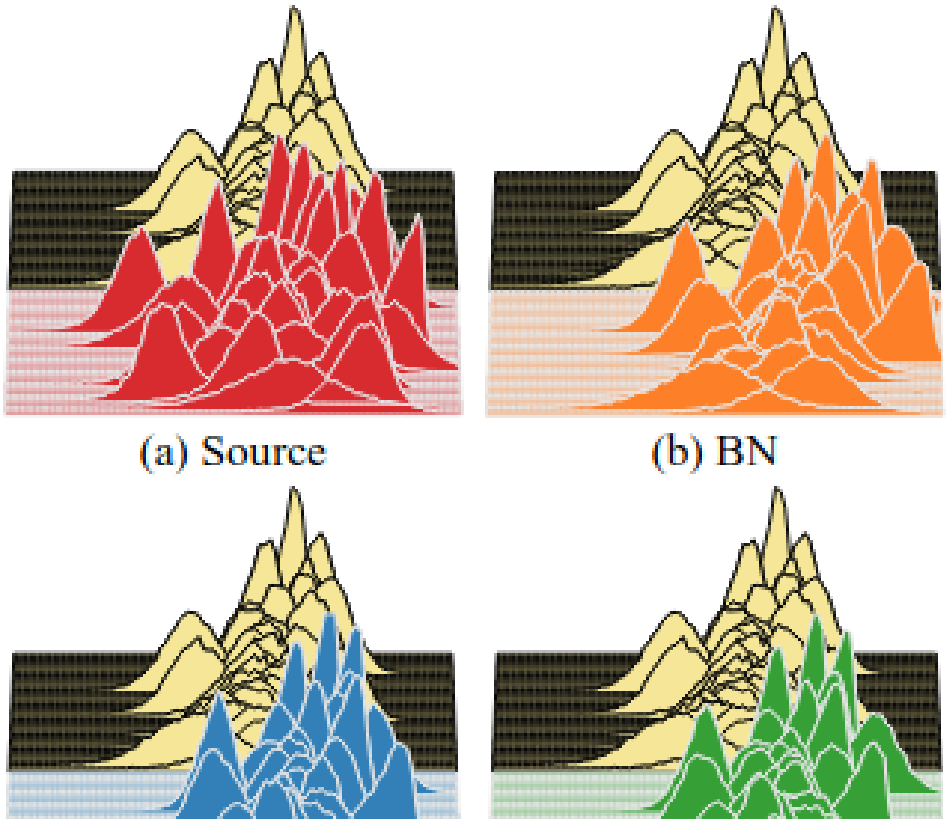


Fig. 5.4 **Log plot of average reward** obtained by both epsilon greedy (in green) and our approach (in blue), as a function of the number of batches.

Test Time Adaptation

- Use uncertainty estimates to adapt a pre-trained model at test time to distribution shifts [Wang et al., 2020]
- Example : Tent method [Wang et al., 2020]
  - Minimize the entropy of the model predictions on test data
  - Update only the affine parameters of batch-norm layers
- Extensions : use uncertainty estimates to weight the loss contributions of each test sample



References I

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.

Fort, S., Hu, C. H., and Lakshminarayanan, B.

Deep ensembles : A loss landscape perspective.

Gal, Y. and Ghahramani, Z. (2015).

Bayesian convolutional neural networks with bernoulli approximate variational inference.
CoRR, abs/1506.02158.

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, 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, 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, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017).

On calibration of modern neural networks.
CoRR, abs/1706.04599.

References II

Hendrycks, D. and Gimpel, K. (2016).

A baseline for detecting misclassified and out-of-distribution examples in neural networks.
arXiv preprint arXiv :1610.02136.

Hendrycks, D., Mazeika, M., and Dietterich, T. G. (2019).

Deep anomaly detection with outlier exposure.
In ICLR.

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.

Lafon, M., Karmim, Y., Silva-Rodríguez, J., Couairon, P., Rambour, C., Fournier-Sniehotta, R., Ben Ayed, I., Dolz, J., and Thome, N. (2025).

Vilu : Learning vision-language uncertainties for failure prediction.
In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 17807–17817.

Lee, K., Lee, K., Lee, H., and Shin, J. (2018).

A simple unified framework for detecting out-of-distribution samples and adversarial attacks.
Advances in neural information processing systems, 31.

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.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017).

Focal loss for dense object detection.
In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

References III

Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., Tran, D., and Lucic, M. (2021).
Revisiting the calibration of modern neural networks.
Advances in neural information processing systems, 34 :15682–15694.

Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P., and Dokania, P. (2020).
Calibrating deep neural networks using focal loss.
Advances in neural information processing systems, 33 :15288–15299.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021).
Learning transferable visual models from natural language supervision.
arXiv preprint arXiv :2103.00020.

Sun, Y., Guo, C., and Li, Y. (2021).
React : Out-of-distribution detection with rectified activations.
Advances in neural information processing systems, 34 :144–157.

Sun, Y., Ming, Y., Zhu, X., and Li, Y. (2022).
Out-of-distribution detection with deep nearest neighbors.
In International conference on machine learning, pages 20827–20840. PMLR.

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

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. (2020).
Tent : Fully test-time adaptation by entropy minimization.
arXiv preprint arXiv :2006.10726.

References IV

Wang, H., Li, Z., Feng, L., and Zhang, W. (2022).

Vim : Out-of-distribution with virtual-logit matching.

In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4921–4930.