Robustness and Uncertainty Estimation for Visual Perception

Oğuzhan Fatih KAR

VILAB, EDIC, EPFL

Motivation

- Visual perception: understand the surrounding physical environment

Motivation

- Visual perception: understand the surrounding physical environment
  - Requires solving different problems

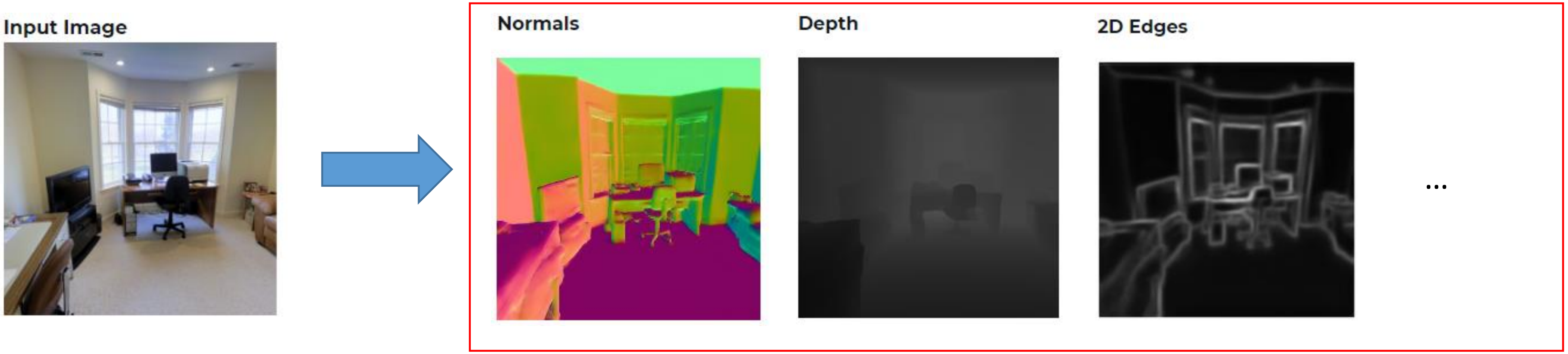
Motivation

- Visual perception: understand the surrounding physical environment
  - Requires solving different problems

Input Image

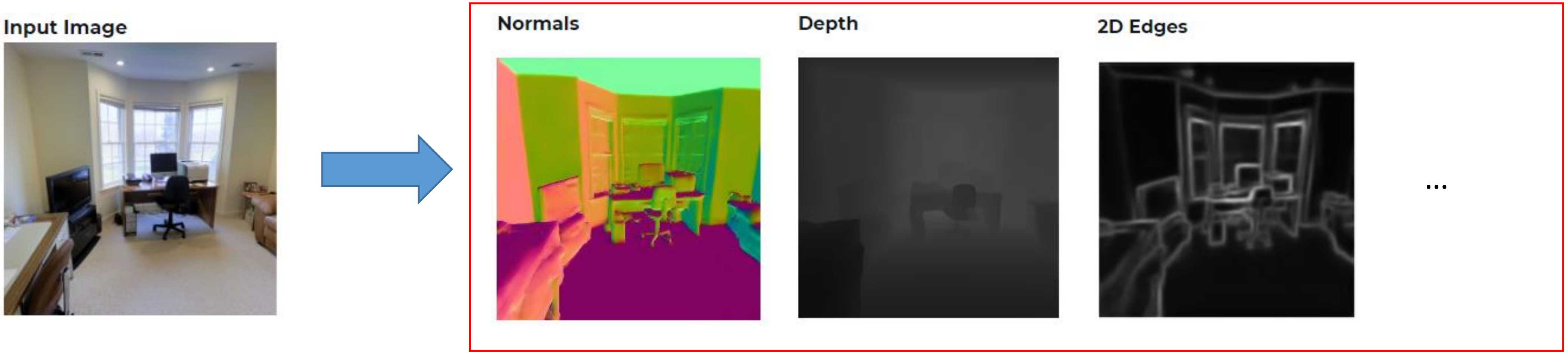
Motivation

- Visual perception: understand the surrounding physical environment
  - Requires solving different problems



Motivation

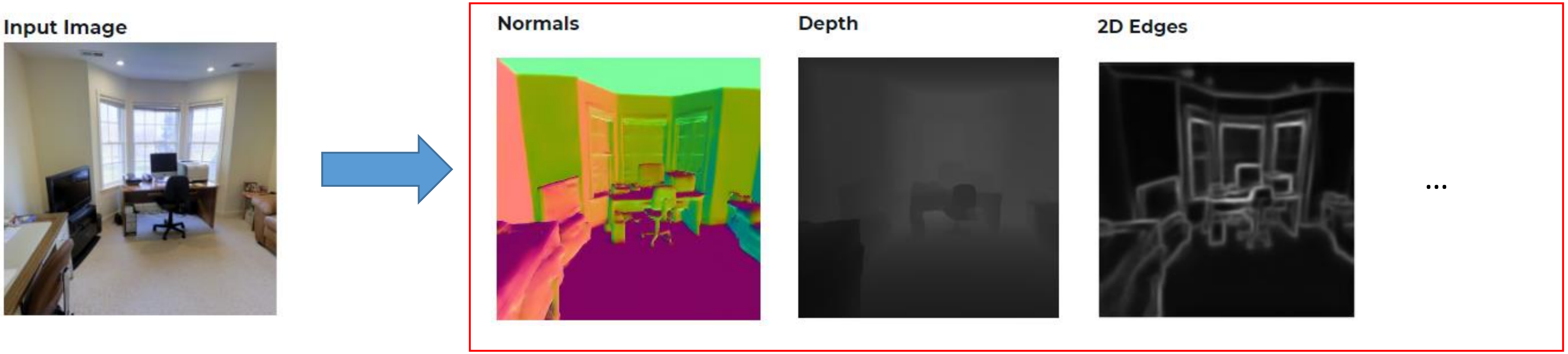
- Visual perception: understand the surrounding physical environment
    - Requires solving different problems



- Abstractions of real world

Motivation

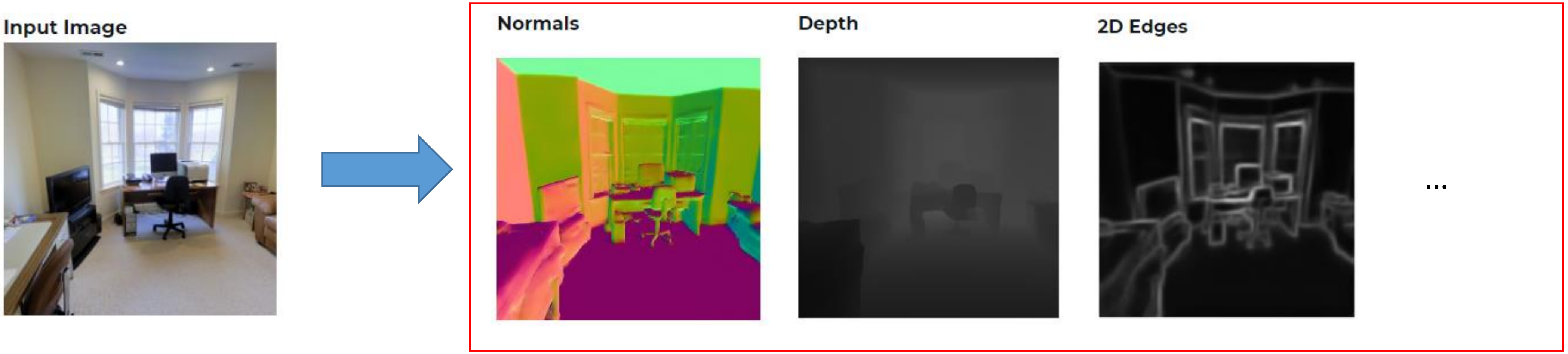
- Visual perception: understand the surrounding physical environment
  - Requires solving different problems



  - Abstractions of real world
    - Useful for many downstream tasks, e.g. navigation [1]

Motivation

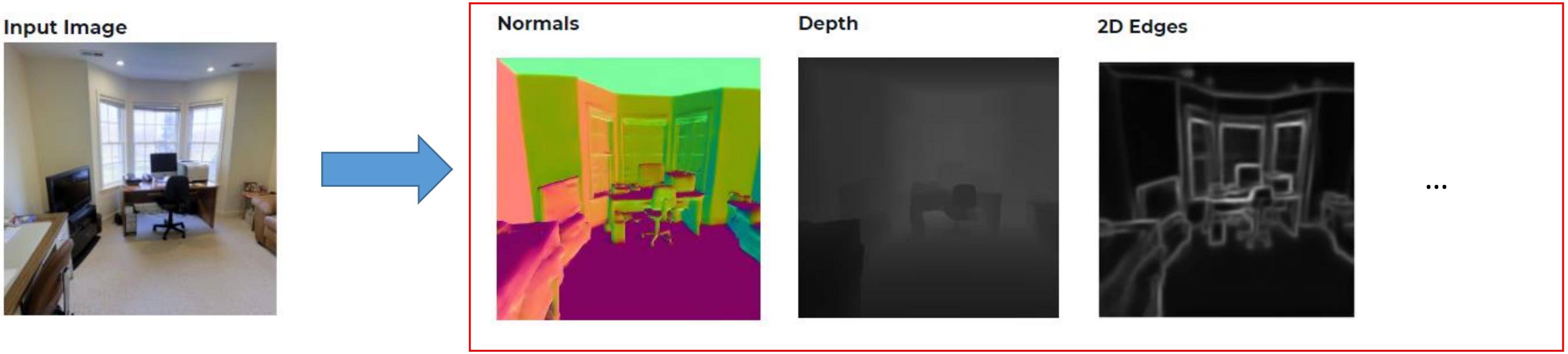
- Visual perception: understand the surrounding physical environment
  - Requires solving different problems



  - Abstractions of real world
    - Useful for many downstream tasks, e.g. navigation [1]
  - Deep learning (DL) approaches are getting better and better [1],[2],[3]

Motivation

- Visual perception: understand the surrounding physical environment
  - Requires solving different problems



- Abstractions of real world
  - Useful for many downstream tasks, e.g. navigation [1]
- Deep learning (DL) approaches are getting better and better [1],[2],[3]
  - Is it always the case?

Motivation

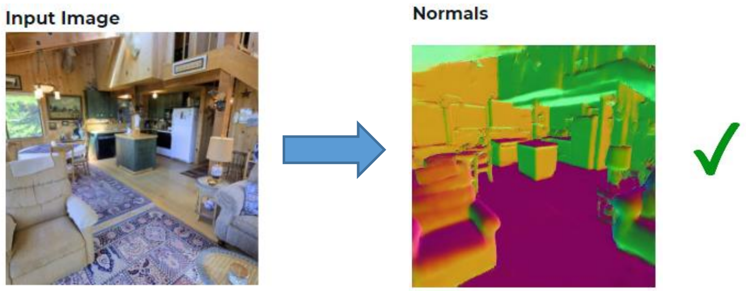
- Robustness: resilience to distribution shifts

Motivation

- Robustness: resilience to distribution shifts
  - A critical requirement for practical applications

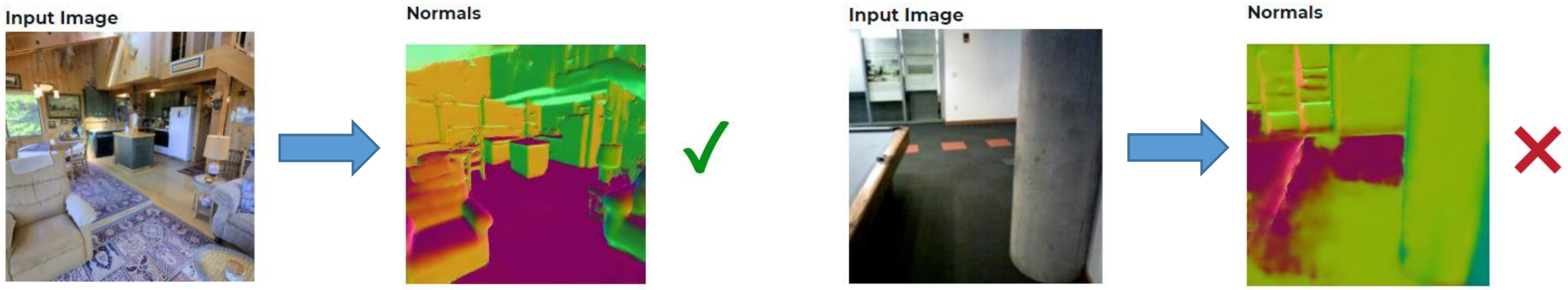
Motivation

- Robustness: resilience to distribution shifts
  - A critical requirement for practical applications



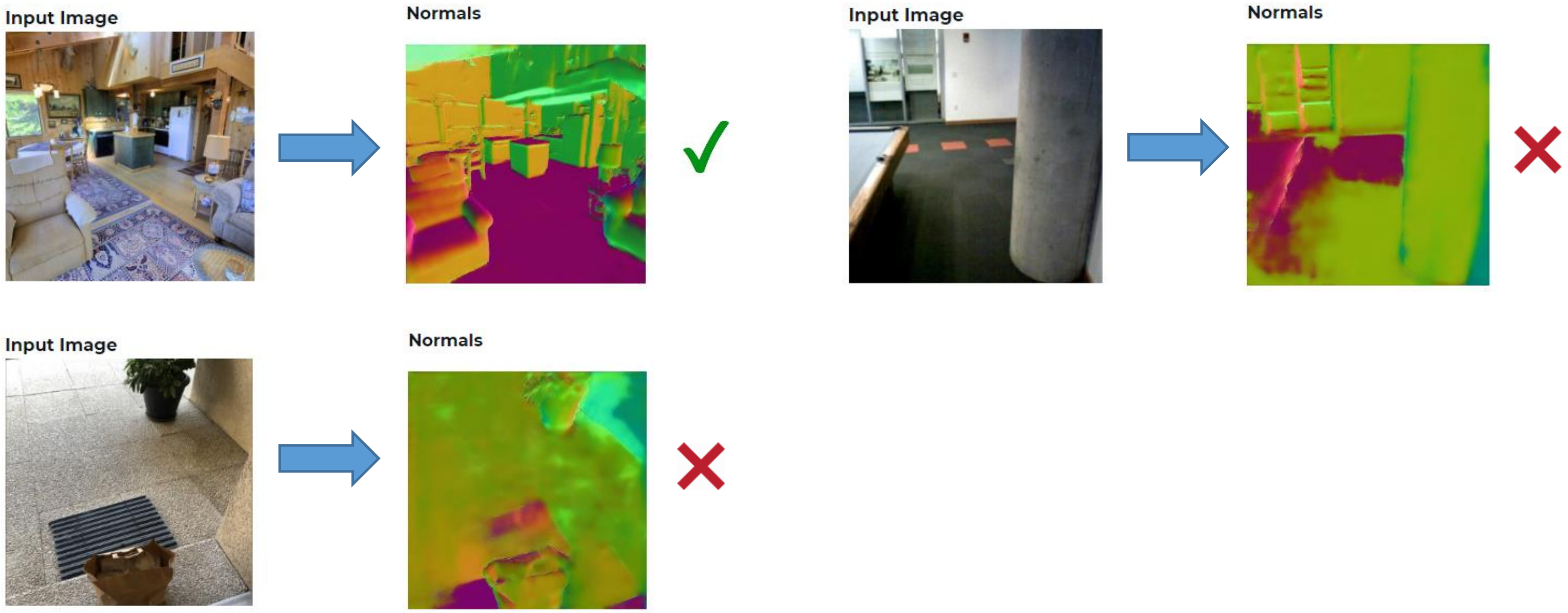
Motivation

- Robustness: resilience to distribution shifts
  - A critical requirement for practical applications



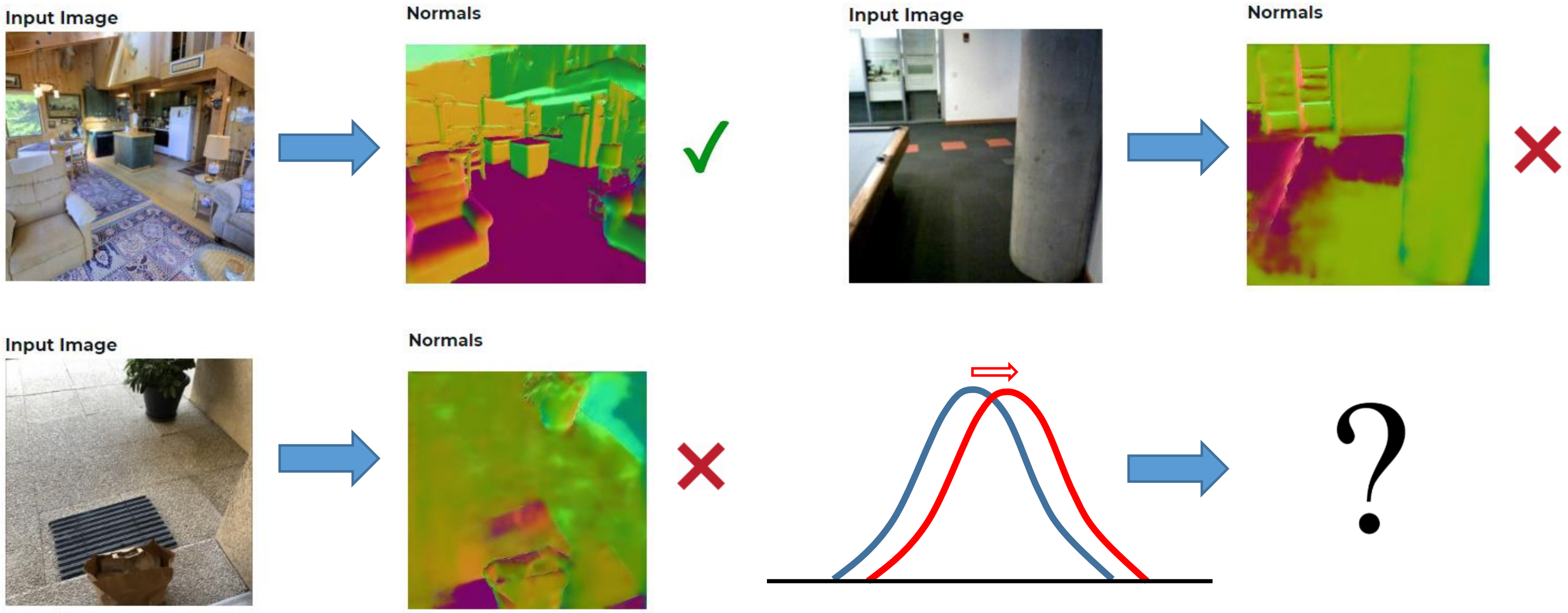
Motivation

- Robustness: resilience to distribution shifts
  - A critical requirement for practical applications



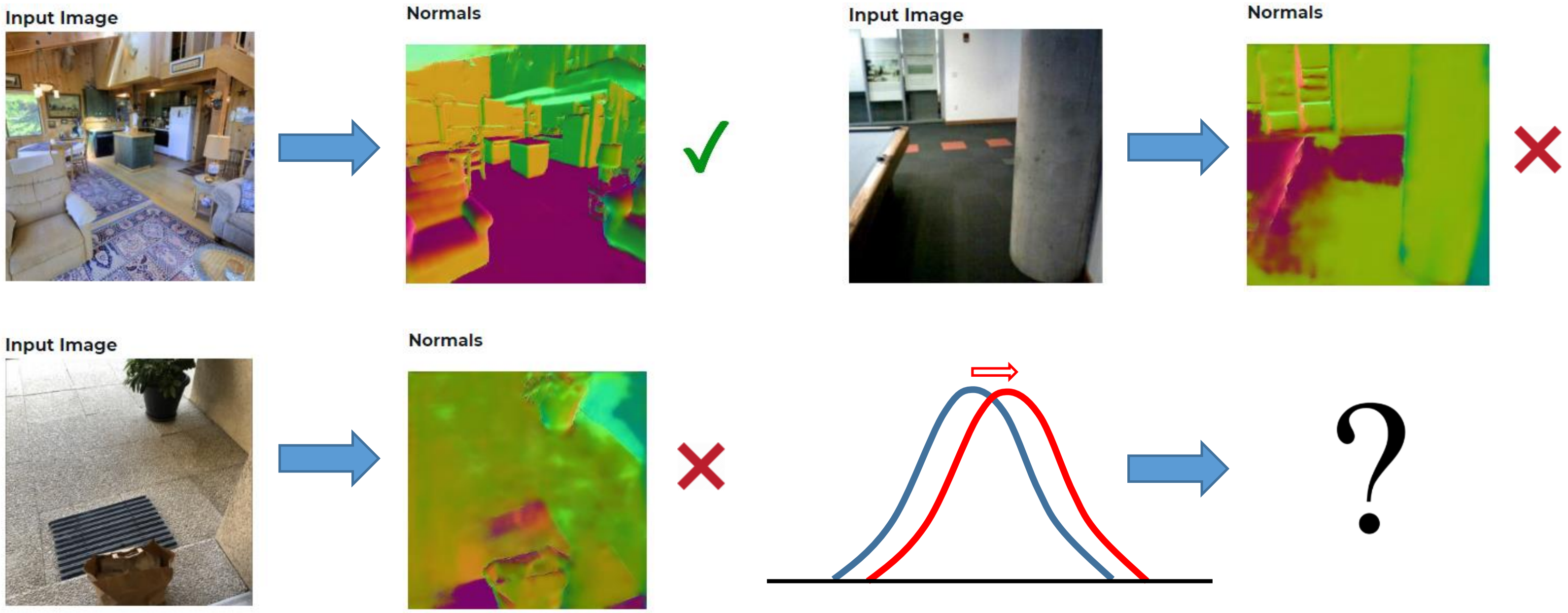
Motivation

- Robustness: resilience to distribution shifts
  - A critical requirement for practical applications



Motivation

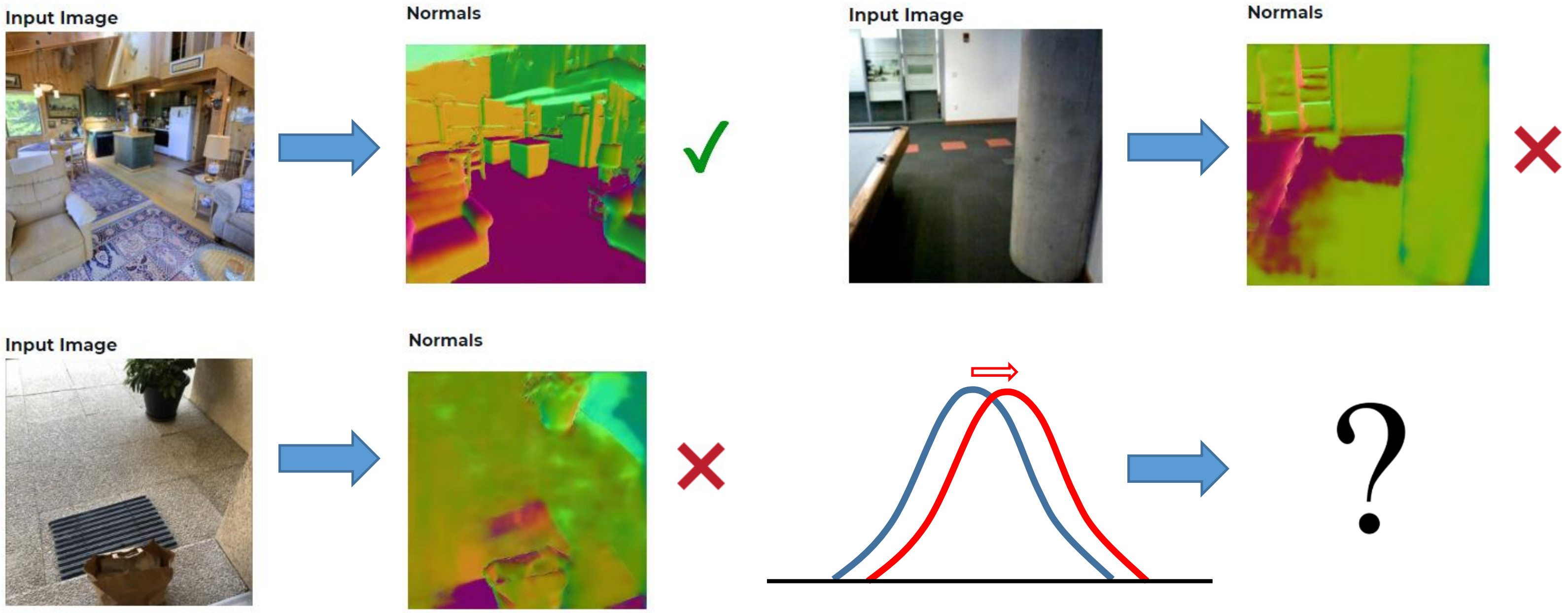
- Robustness: resilience to distribution shifts
  - A critical requirement for practical applications



- Model may be wrong sometimes

Motivation

- Robustness: resilience to distribution shifts
  - A critical requirement for practical applications



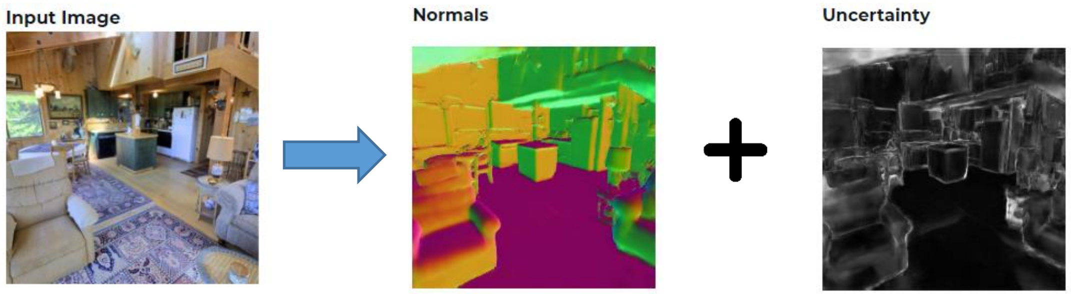
- Model may be wrong sometimes
  - Should be able to say “Hey, I’m not sure!”

Motivation

- Uncertainty: A mechanism to understand model limitations

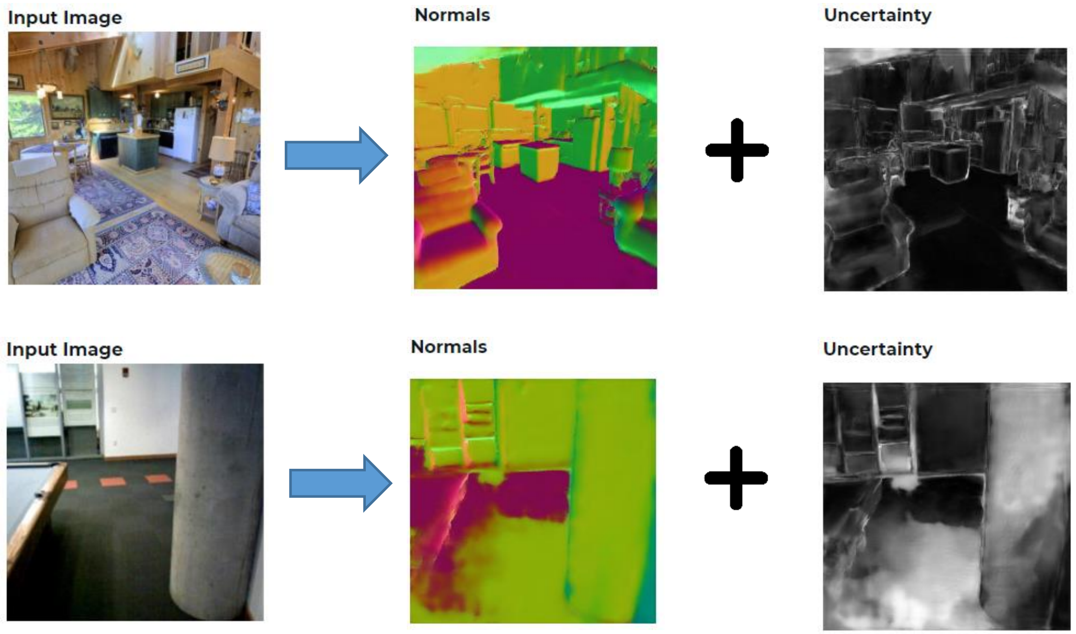
Motivation

- Uncertainty: A mechanism to understand model limitations



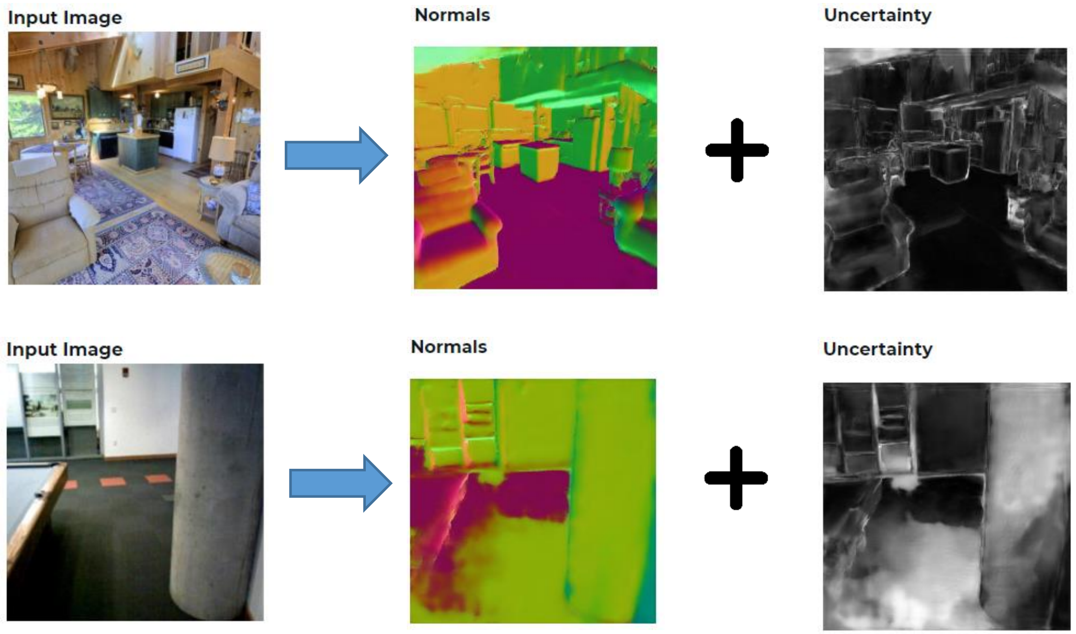
Motivation

- Uncertainty: A mechanism to understand model limitations



Motivation

- Uncertainty: A mechanism to understand model limitations



- Can be used for
  - improving robustness

Motivation

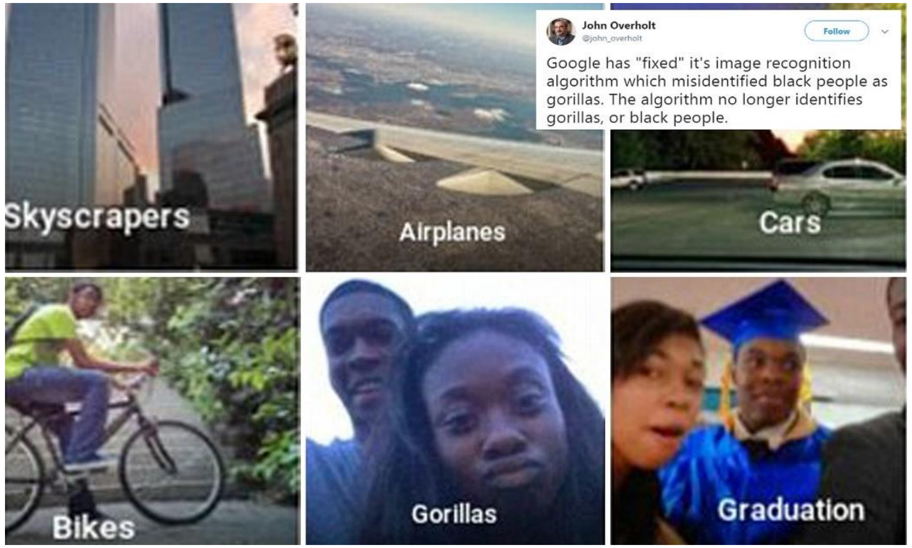
- Uncertainty: A mechanism to understand model limitations

Brad Templeton. “Tesla In Taiwan Crashes Directly Into Overturned Truck, Ignores Pedestrian, With Autopilot On”. Forbes, 2020.

- Can be used for
  - improving robustness
  - improving decision making

Motivation

- Uncertainty: A mechanism to understand model limitations



Jessica Guynn. “Google photos labeled black people ’gorillas’”. USA Today, 2015.

- Can be used for
  - improving robustness
  - improving decision making

Motivation

- 3 background papers
  - "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" (Alex Kendall, Yarin Gal) [4]
  - "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles" (Balaji Lakshminarayanan, Alexander Pritzel, Charles Blundell) [5]
  - "On Calibration of Modern Neural Networks" (Chuan Guo, Geoff Pleiss, Yu Sun, Kilian Q. Weinberger)

What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? [4]

Problem

- Deep learning has good performance, but is it always the case?
  - Tesla crash, Google photos labelling, etc.

Problem

- Deep learning has good performance, but is it always the case?
  - Tesla crash, Google photos labelling, etc.
- What happens when the model fails?

Problem

- Deep learning has good performance, but is it always the case?
  - Tesla crash, Google photos labelling, etc.
- What happens when the model fails?
  - Need for an alarm mechanism

Problem

- Deep learning has good performance, but is it always the case?
  - Tesla crash, Google photos labelling, etc.
- What happens when the model fails?
  - Need for an alarm mechanism

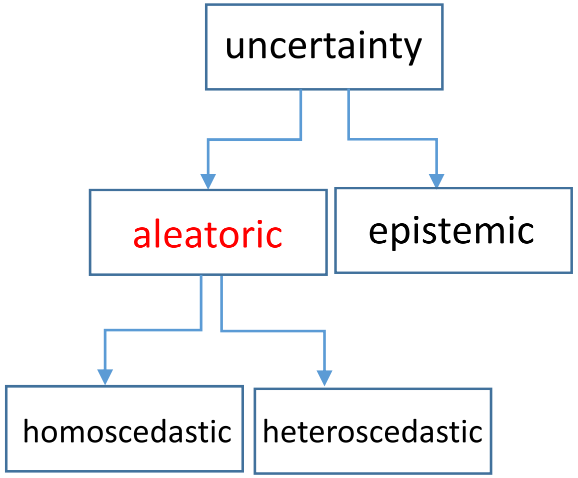
"We are mostly interested in knowing how likely certain outcomes are rather than just using the most likely one"

Solution

- Sources of uncertainty
- Modelling uncertainty

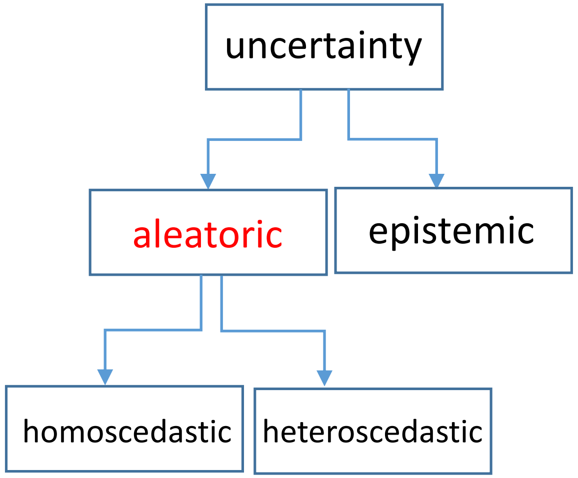
Solution

- Sources of uncertainty
  - *Aleatoric* uncertainty
    - Data uncertainty
    - Captures noise inherent in the observations
      - A function of input
    - E.g. sensor noise and blur



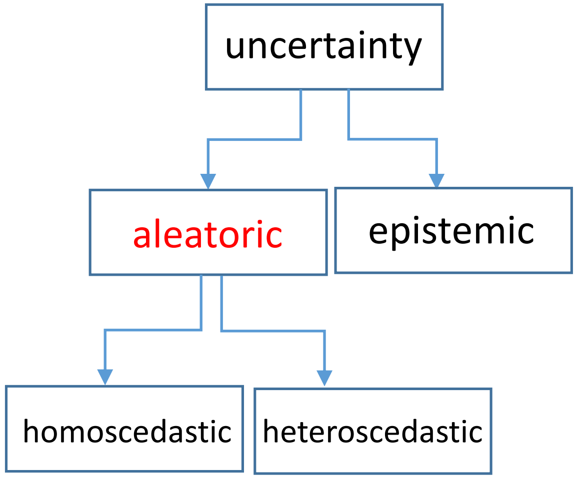
Solution

- Sources of uncertainty
  - *Aleatoric* uncertainty
    - Data uncertainty
    - Captures noise inherent in the observations
      - A function of input
    - E.g. sensor noise and blur
    - Can't decreased with more data



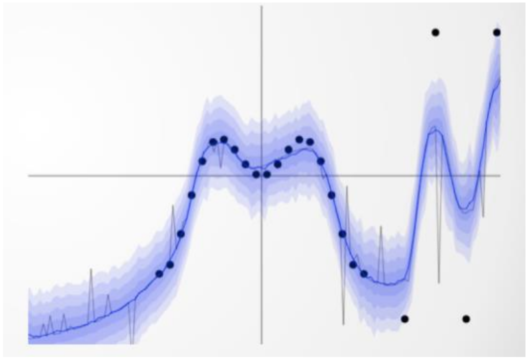
Solution

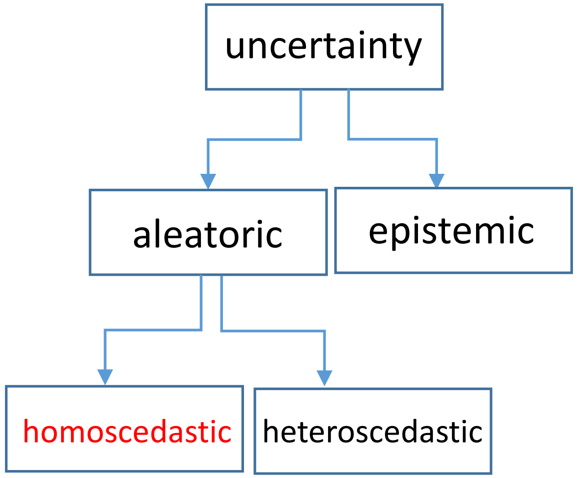
- Sources of uncertainty
  - *Aleatoric* uncertainty
    - Data uncertainty
    - Captures noise inherent in the observations
      - A function of input
    - E.g. sensor noise and blur
    - Can’t decreased with more data
    - Can decrease with increasing sensing ability



Solution

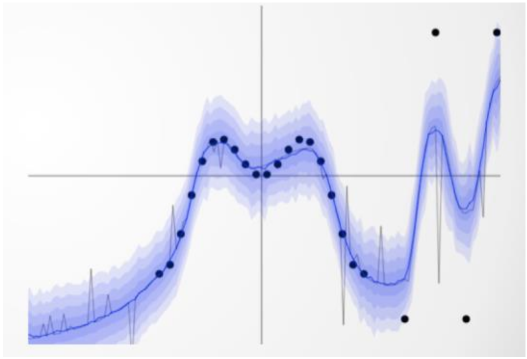
- Sources of uncertainty
  - *Aleatoric* uncertainty
    - Two variants
    - *Homoscedastic* : Constant for all inputs
      - Could change between tasks

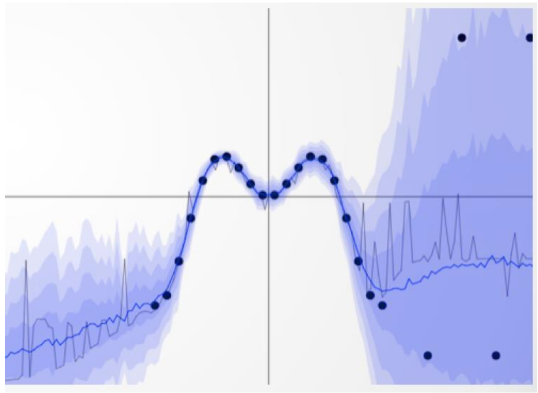


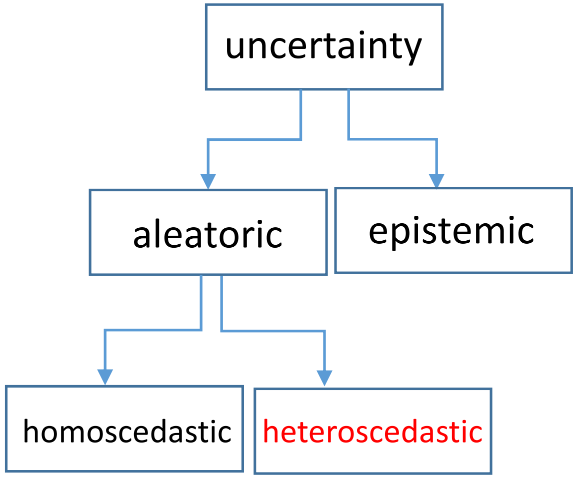


Solution

- Sources of uncertainty
  - *Aleatoric* uncertainty
    - Two variants
    - *Homoscedastic* : Constant for all inputs
      - Could change between tasks
    - *Heteroscedastic* : Changes between inputs
      - Useful for vision tasks
      - Could be learned from data



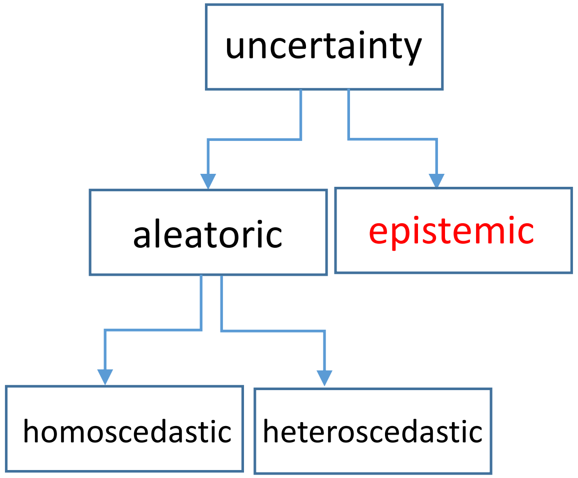


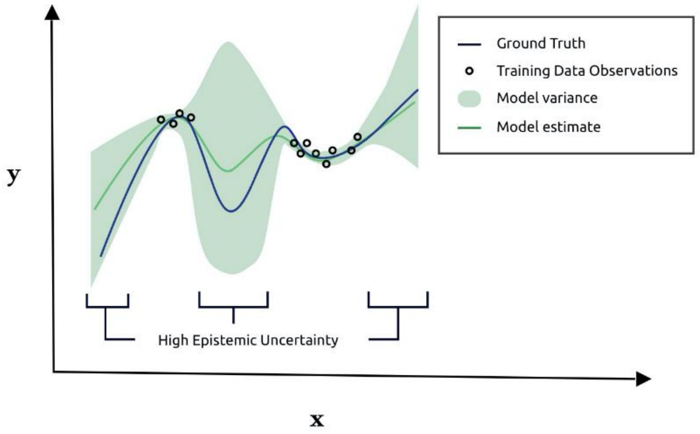


What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?

Solution

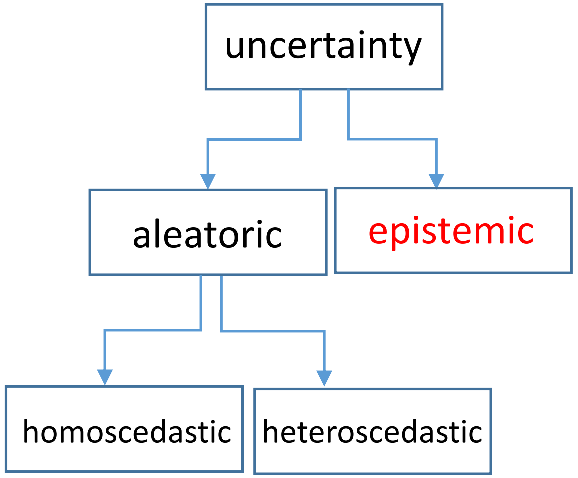
- Sources of uncertainty
  - Epistemic uncertainty
    - Model uncertainty
    - Captures uncertainty in model parameters
      - “Which model generated our data?”

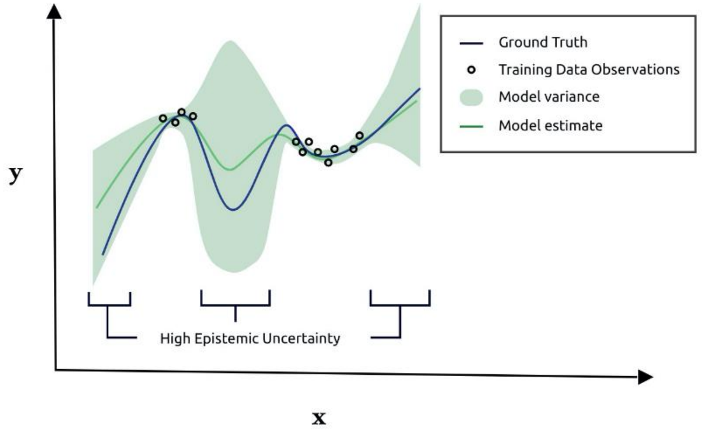




Solution

- Sources of uncertainty
  - Epistemic uncertainty
    - Model uncertainty
    - Captures uncertainty in model parameters
      - “Which model generated our data?”
    - Can be decreased with more data

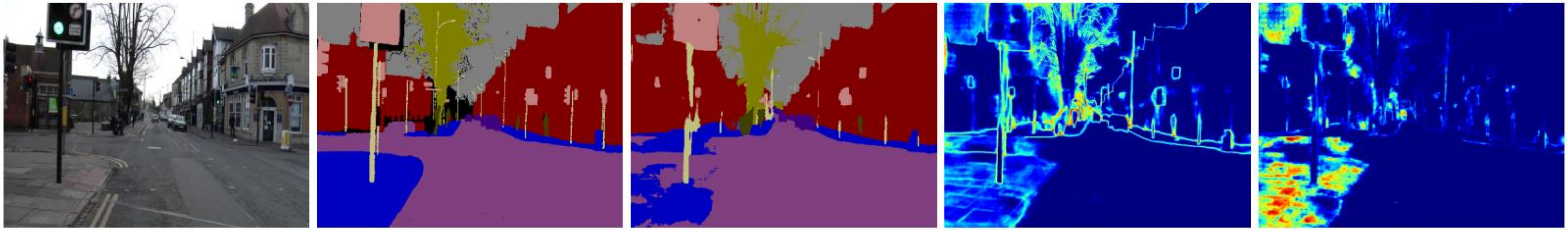




What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?

Solution

- Sources of uncertainty
  - Aleatoric vs Epistemic



Input GT Prediction Aleatoric Epistemic

Solution

- Sources of uncertainty ✓
- Modelling uncertainty
  - Aleatoric uncertainty
    - Function of input
    - Model it over outputs
  - Epistemic uncertainty:
    - Function of model
    - Model it over parameters (i.e. weights)

Solution

- Modelling aleatoric uncertainty
  - Regression model with parameters θ
  - Dataset: input $X = \{x_1, \dots, x_N\}$ and label $\mathrm{Y} = \{y_1, \dots, y_N\}$
  - Assume Gaussian likelihood

Solution

- Modelling aleatoric uncertainty
  - Regression model with parameters θ
  - Dataset: input $X = \{x_1, \ldots, x_N\}$ and label $\mathrm{Y} = \{y_1, \ldots, y_N\}$
  - Assume Gaussian likelihood
  - Negative log-likelihood loss (NLL)
    - $L_{NN}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\sigma(x_i)^2} \|y_i - f(x_i)\|_2^2 + \frac{1}{2}\log \sigma(x_i)^2$

Solution

- Modelling aleatoric uncertainty
  - Regression model with parameters θ
  - Dataset: input $X = \{x_1, \dots, x_N\}$ and label $\mathrm{Y} = \{y_1, \dots, y_N\}$
  - Assume Gaussian likelihood
  - Negative log-likelihood loss (NLL)
    - $L_{NN}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\sigma(x_i)^2} \|y_i - f(x_i)\|_2^2 + \frac{1}{2}\log \sigma(x_i)^2$
    - Predict mean $f(x_i)$ and variance $\sigma(x_i)^2$
    - Use them in the NLL
    - No label needed for $\sigma(x_i)^2$

Solution

- Modelling aleatoric uncertainty
  - Negative log-likelihood loss (NLL)
    - $L_{NN}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\sigma(x_i)^2} \|y_i - f(x_i)\|_2^2 + \frac{1}{2}\log \sigma(x_i)^2$
    - For heteroscedastic case, changes with input x_i
    - For homoscedastic case, constant free parameter

Solution

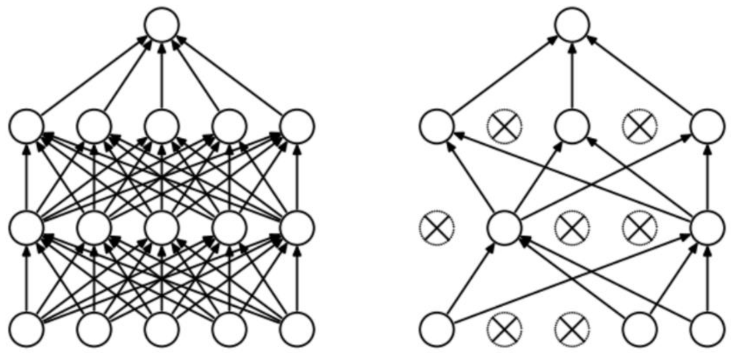
- Modelling aleatoric uncertainty
  - Negative log-likelihood loss (NLL)
    - $L_{NN}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \overbrace{\boxed{\frac{1}{2\sigma(x_i)^2}\|y_i - f(x_i)\|_2^2}}^{1} + \overbrace{\boxed{\frac{1}{2}\log \sigma(x_i)^2}}^{2}$
    - For heteroscedastic case, changes with input x_i
    - For homoscedastic case, constant free parameter
    - Balance between 1&2
      - Can't be overconfident (1⬆)
      - Can't be over-uncertain (2⬆)
      - No manual tuning

Solution

- Sources of uncertainty ✓
- Modelling aleatoric uncertainty ✓
- Modelling epistemic uncertainty
  - Assume a prior over model weights W, e.g. $W \sim N(0, I)$
  - Compute posterior $p(W|X,Y) = p(Y|X,W)p(W)/p(Y|X)$
    - Intractable, hence approximate
    - They use MC dropout [4]

Solution

- Sources of uncertainty ✓
- Modelling aleatoric uncertainty ✓
- Modelling epistemic uncertainty
  - Assume a prior over model weights W, e.g. $W \sim N(0, I)$
  - Compute posterior $p(W|X,Y) = p(Y|X,W)p(W)/p(Y|X)$
    - Intractable, hence approximate
    - They use MC dropout [5]
  - MC dropout
    - Training time: Use dropout for every weight layer
    - Test time: Use dropout to sample from posterior
    - Variance of the samples: Epistemic uncertainty



(a) Standard Neural Net (b) After applying dropout.

Solution

- Combining both uncertainties
    - Output both mean $\hat{y}$ and variance $\hat{\sigma}^2$
    - Use the NLL for training

Solution

- Combining both uncertainties
  - Output both mean $\hat{y}$ and variance $\hat{\sigma}^2$
  - Use the NLL for training
    - $s_i := \log \hat{\sigma}_i^2$ for numerical stability
    - $f^{\widehat{W}} = [\hat{y}, \hat{\sigma}^2]$
    - $L_{BNN}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \exp(-s_i) \|y_i - \hat{y}_i\|_2^2 + \frac{1}{2} s_i$

Solution

- Combining both uncertainties
  - Output both mean $\hat{y}$ and variance $\hat{\sigma}^2$
  - Use the NLL for training
    - $s_i := \log \hat{\sigma}_i^2$ for numerical stability
    - $f^{\widehat{W}} = [\hat{y}, \hat{\sigma}^2]$
    - $L_{BNN}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2} \exp(-s_i) \|y_i - \hat{y}_i\|_2^2 + \frac{1}{2} s_i$
  - Perform T passes with dropout enabled (test time)

Solution

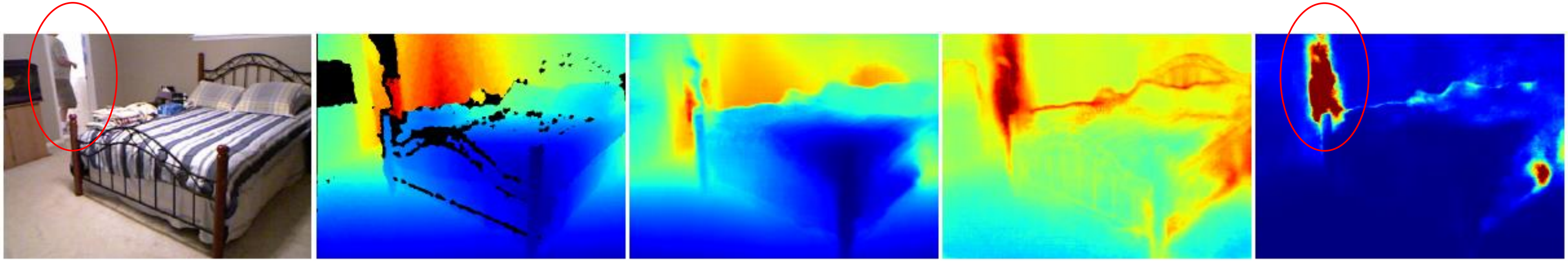
- Combining both uncertainties
  - Output both mean $\hat{y}$ and variance $\hat{\sigma}^2$
  - Use the NLL for training
    - $s_i := \log \hat{\sigma}_i^2$ for numerical stability
    - $f^{\widehat{W}} = [\hat{y}, \hat{\sigma}^2]$
    - $L_{BNN}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \exp(-s_i) \|y_i - \hat{y}_i\|_2^2 + \frac{1}{2} s_i$
  - Perform T passes with dropout enabled (test time)
  - Final predictive uncertainty is the summation of both terms
    - $Var(y) \approx \underbrace{\boxed{\frac{1}{T} \sum_{t=1}^{T} \hat{\sigma}_t^2}}_{\text{aleatoric}} + \underbrace{\boxed{\frac{1}{T} \sum_{t=1}^{T} \hat{y}_t^2 - \left(\frac{1}{T} \sum_{t=1}^{T} \hat{y}_t\right)^2}}_{\text{epistemic}}$

Results

- Pixel-wise depth regression and semantic segmentation
- Aleatoric uncertainty as loss attenuation
  - Improves accuracy
  - Modelling both uncertainties improve further

Results

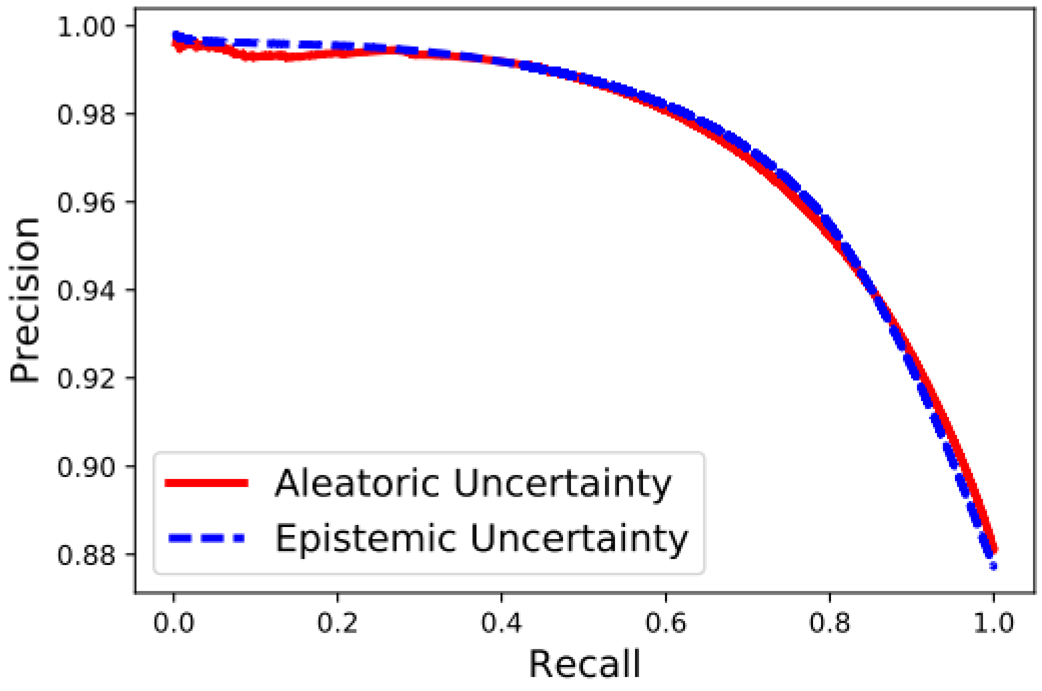
- Pixel-wise depth regression and semantic segmentation
- Aleatoric uncertainty as loss attenuation
  - Improves accuracy
  - Modelling both uncertainties improve further



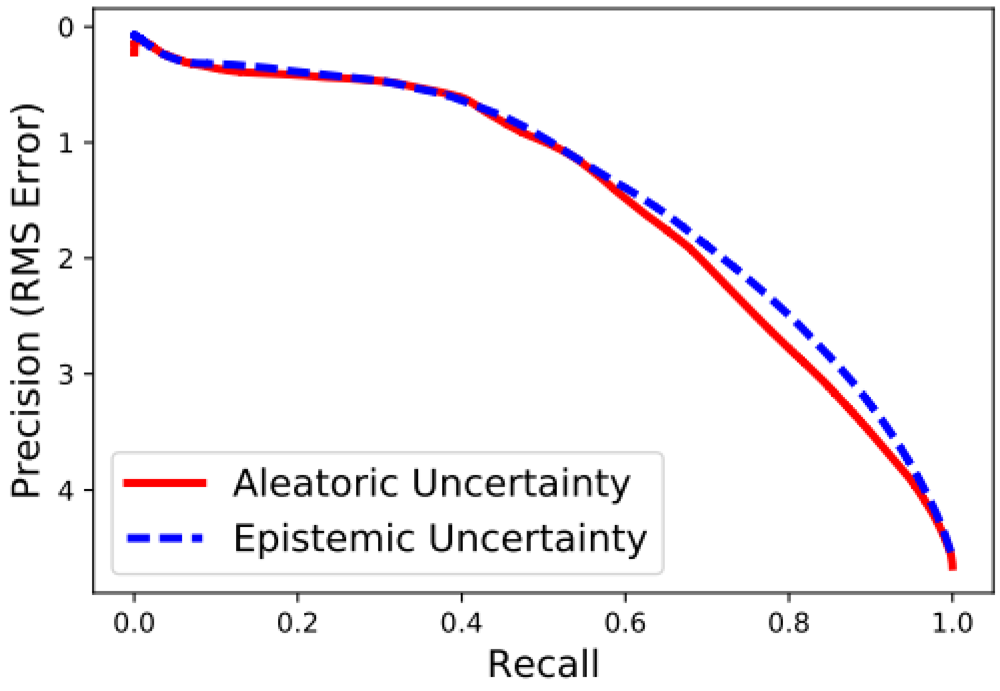
Input GT Prediction Aleatoric Epistemic

Results

- Precision decreases with increasing uncertainty



(a) Classification (CamVid)



(b) Regression (Make3D)

Results

- Epistemic uncertainty decreases with increasing training data

Train dataset	Test dataset	RMS	Aleatoric variance	Epistemic variance
Make3D / 4	Make3D	5.76	0.506	7.73
Make3D / 2	Make3D	4.62	0.521	4.38
Make3D	Make3D	3.87	0.485	2.78
Make3D / 4	NYUv2	-	0.388	15.0
Make3D	NYUv2	-	0.461	4.87

Results

- Epistemic uncertainty decreases with increasing training data
- Aleatoric uncertainty does not decrease with more data

Train dataset	Test dataset	RMS	Aleatoric variance	Epistemic variance
Make3D / 4	Make3D	5.76	0.506	7.73
Make3D / 2	Make3D	4.62	0.521	4.38
Make3D	Make3D	3.87	0.485	2.78
Make3D / 4	NYUv2	-	0.388	15.0
Make3D	NYUv2	-	0.461	4.87

Results

- Epistemic uncertainty decreases with increasing training data
- Aleatoric uncertainty does not decrease with more data
- Epistemic uncertainty increases with distribution shift

Train dataset	Test dataset	RMS	Aleatoric variance	Epistemic variance
Make3D / 4	Make3D	5.76	0.506	7.73
Make3D / 2	Make3D	4.62	0.521	4.38
Make3D	Make3D	3.87	0.485	2.78
Make3D / 4	NYUv2	-	0.388	15.0
Make3D	NYUv2	-	0.461	4.87

Discussion

- Combining different sources of uncertainty

Discussion

- Combining different sources of uncertainty
- Showcasing results for both regression and classification tasks

Discussion

- Combining different sources of uncertainty
- Showcasing results for both regression and classification tasks
- Aleatoric uncertainty adds negligible compute

Discussion

- Combining different sources of uncertainty
- Showcasing results for both regression and classification tasks
- Aleatoric uncertainty adds negligible compute
- Epistemic uncertainty needs multiple passes
    - Real-time application?

Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles [6]

Problem

- Uncertainty modelling in [4] is fine except it requires a BNN
  - Changes to standard training procedure

Problem

- Uncertainty modelling in [4] is fine except it requires a BNN
  - Changes to standard training procedure
  - Computational complexity

Problem

- Uncertainty modelling in [4] is fine except it requires a BNN
  - Changes to standard training procedure
  - Computational complexity
  - Approximation quality to posterior is critical

Problem

- Uncertainty modelling in [4] is fine except it requires a BNN
  - Changes to standard training procedure
  - Computational complexity
  - Approximation quality to posterior is critical
  - Reasonability of the assumed prior is critical

Problem

- Uncertainty modelling in [4] is fine except it requires a BNN
  - Changes to standard training procedure
  - Computational complexity
  - Approximation quality to posterior is critical
  - Reasonability of the assumed prior is critical

“We need a more general purpose solution to estimate uncertainty without changing the standard pipeline significantly”

Solution

- Step 1: using a proper loss
  - NLL is a proper loss for uncertainty estimation
  - Also utilized by [4]
  - Output both mean and variance

Solution

- Step 2: ensembling
  - Each model is trained with random initialization + shuffled data
  - The sample variance of predictions as an uncertainty representation

Solution

- Step 2: ensembling
  - Each model is trained with random initialization + shuffled data
  - The sample variance of predictions as an uncertainty representation
  - Compute final mean and variance from the ensembled M models
    - Gaussian mixture with uniform weights for mixture components
    - $Mean(y) \approx \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m$
    - $Var(y) \approx \frac{1}{M} \sum_{m=1}^{M} (\hat{\sigma}_m^2 + \hat{y}_m^2) - Mean(y)^2$

Solution

- Step 3 (optional): adversarial training
  - Proposed by [7]
  - Generates an adversarial example
    - $x' = x + \epsilon\, sign\left(\nabla_x l(\theta, x, y)\right)$ where $l(\theta, x, y)$ is loss

Solution

- Step 3 (optional): adversarial training
  - Proposed by [7]
  - Generates an adversarial example
    - $x' = x + \epsilon\, sign\left(\nabla_x l(\theta, x, y)\right)$ where $l(\theta, x, y)$ is loss
  - Training data is augmented using these examples
  - Shown to improve robustness against adv. attacks [7]

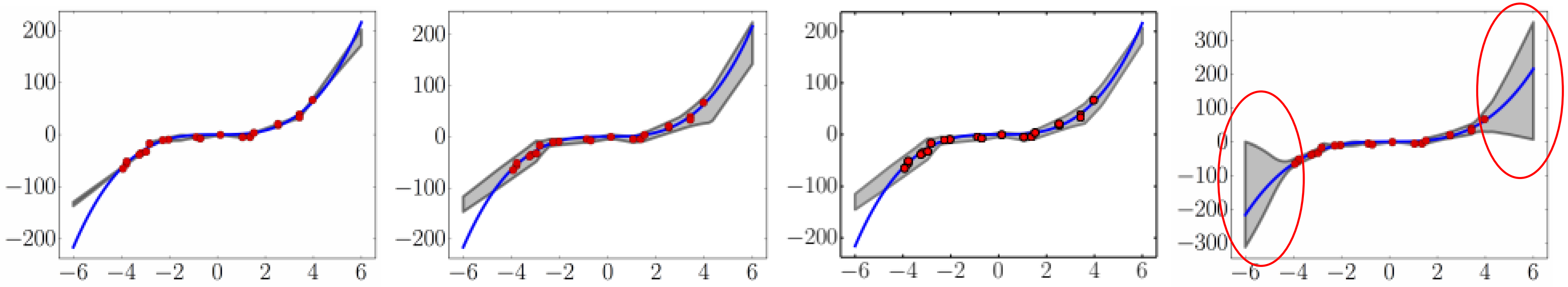
Solution

- Step 3 (optional): adversarial training
  - Proposed by [7]
  - Generates an adversarial example
    - $x' = x + \epsilon\, sign\big(\nabla_x l(\theta, x, y)\big)$ where $l(\theta, x, y)$ is loss
  - Training data is augmented using these examples
  - Shown to improve robustness against adv. attacks [7]

  - Here: use this to smooth the predicted distribution around ϵ-neighbourhood of data (thus increase the likelihood)
  - Provides additional improvement (in some cases)

Results

- Comparison with ensemble trained with MSE (instead of NLL)

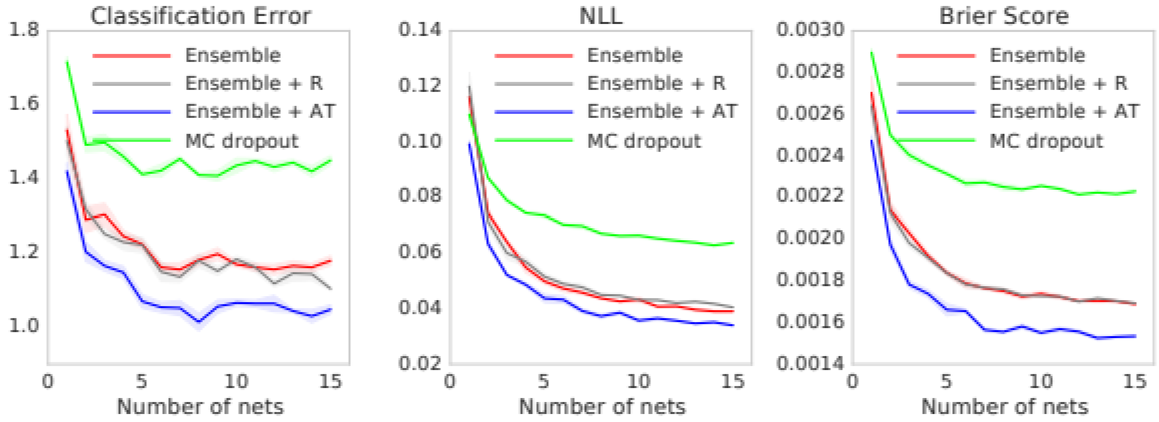


5x MSE | 1x NLL | 1x NLL (+adv) | 5x NLL (+adv)

- NLL yields better uncertainty

Results

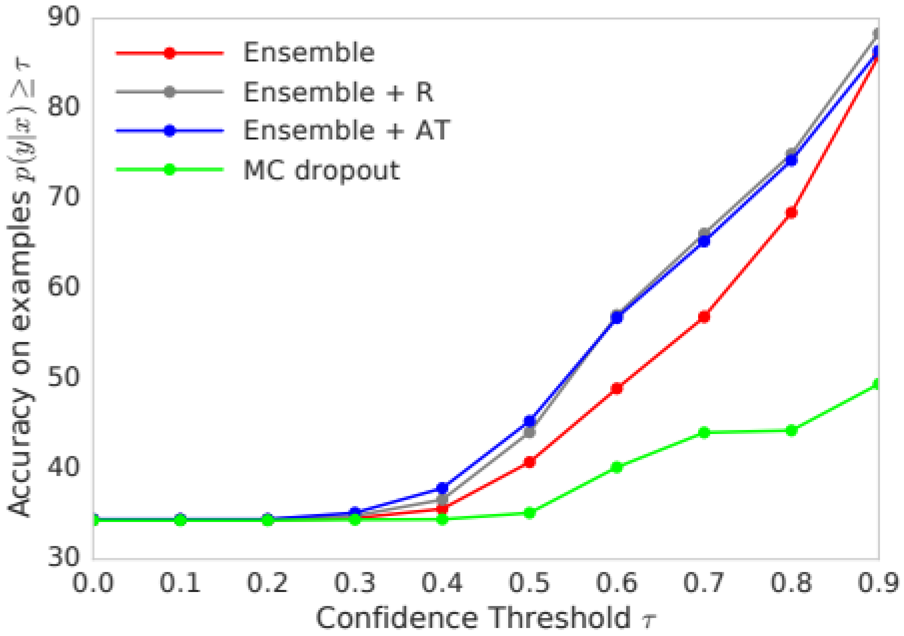
- Comparison with other baselines



- Better performance with higher # of nets

Results

- Uncertainty reliability



- Accuracy & confidence agrees well

Discussion

- Competitive empirical results with BNN based approaches

Discussion

- Competitive empirical results with BNN based approaches
- Requires multiple models for a single task
  - Memory-constrained and real-time apps?

Discussion

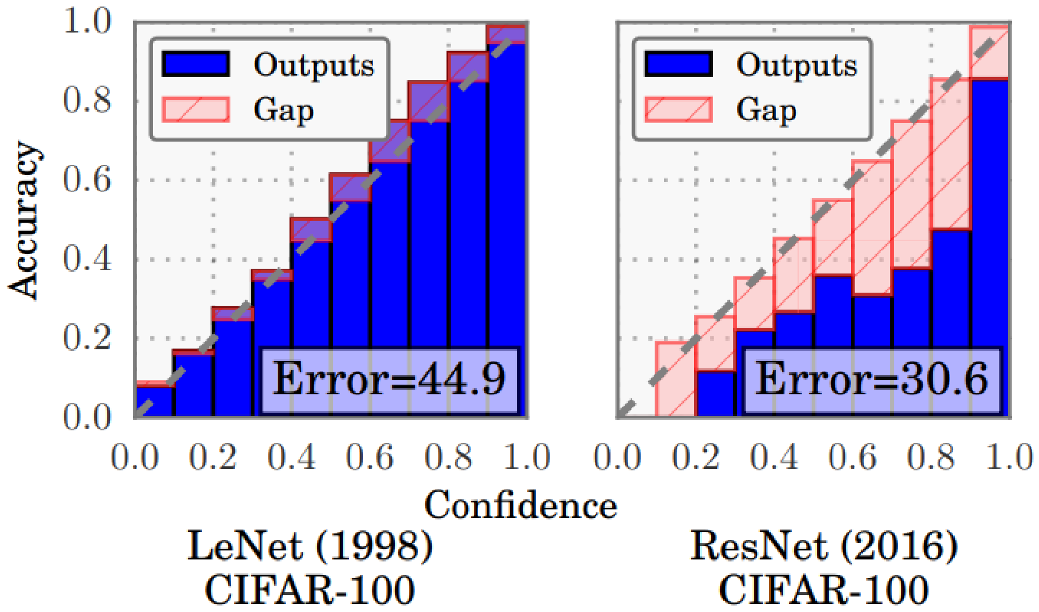
- Competitive empirical results with BNN based approaches
- Requires multiple models for a single task
  - Memory-constrained and real-time apps?
- Some potentially insightful comparisons
  - Comparison with MC dropout based on computation budget?
  - Comparison with an ensemble of MC dropout models?

Discussion

- Competitive empirical results with BNN based approaches
- Requires multiple models for a single task
  - Memory-constrained and real-time apps?
- Some potentially insightful comparisons
  - Comparison with MC dropout based on computation budget?
  - Comparison with an ensemble of MC dropout models?
- Importance of adversarial training requires further investigation
  - Comparison with standard data augmentation techniques?

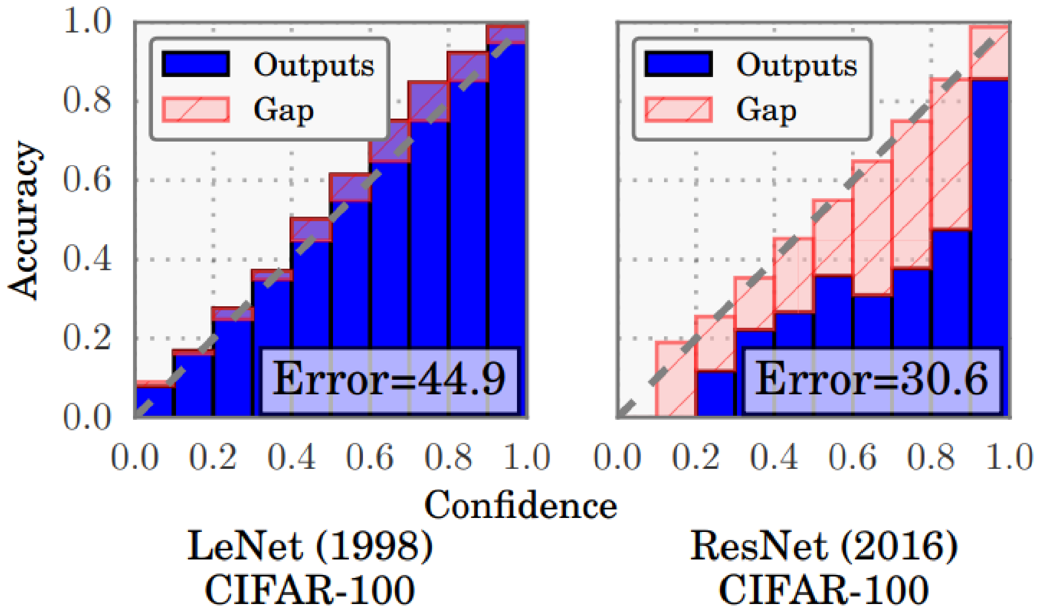
On Calibration of Modern Neural Networks [8]

Problem


Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
Outputs
Gap
Error=44.9
Outputs
Gap
Error=30.6
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0
Confidence
LeNet (1998)
CIFAR-100
ResNet (2016)
CIFAR-100

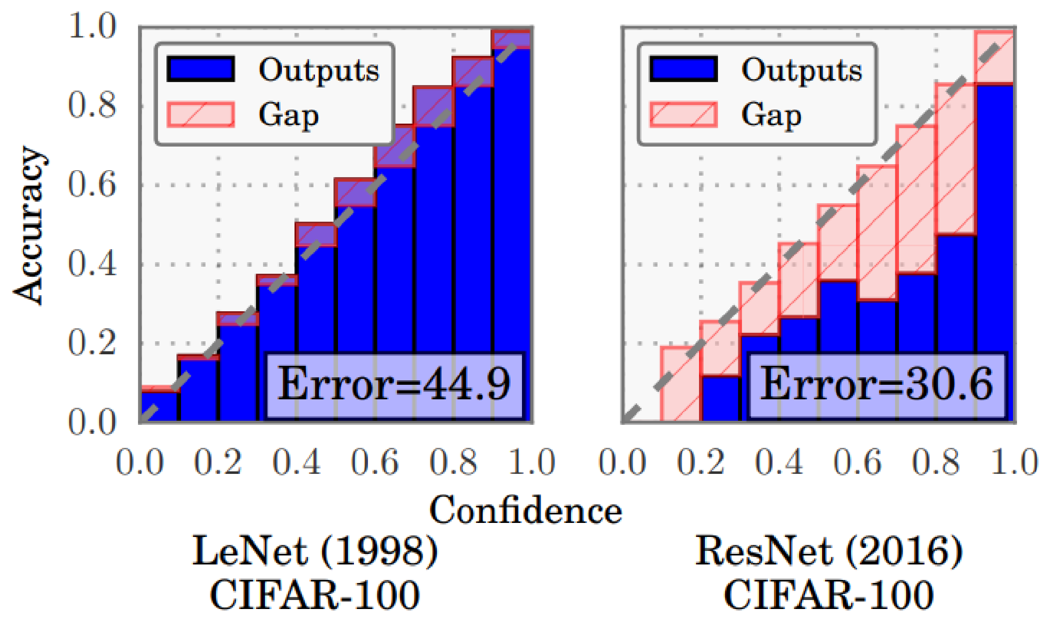
Problem

- Modern networks are overconfident in their predictions
  - Though they have better acc than their older counterparts



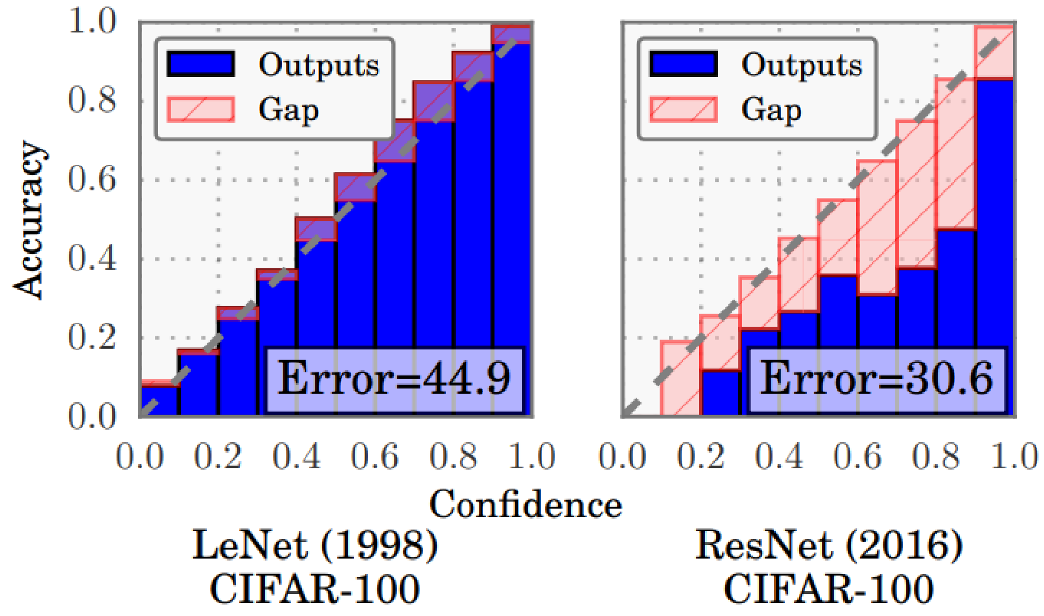
Problem

- Modern networks are overconfident in their predictions
  - Though they have better acc than their older counterparts
- Gap between the confidence and accuracy = miscalibration



Problem

- Modern networks are overconfident in their predictions
  - Though they have better acc than their older counterparts
- Gap between the confidence and accuracy = miscalibration



"We need to understand 1) why miscalibration occurs in the current models and 2) how to solve this?"

Solution

- Metrics to evaluate miscalibration
- Factors for miscalibration
- Solving miscalibration

Solution

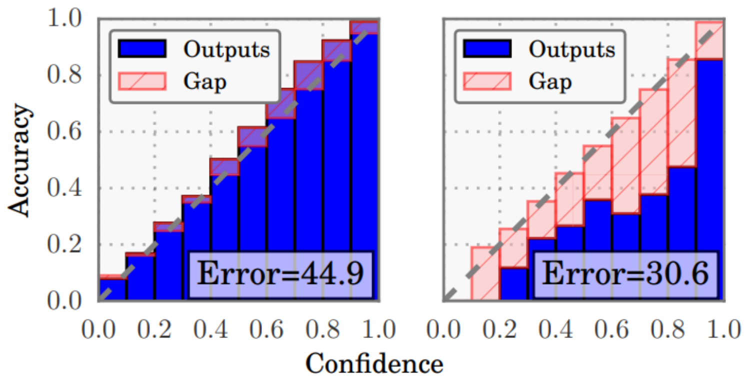
- Metrics to evaluate miscalibration
  - Perfect calibration
    - $Prob(\hat{y}_i = y \mid \hat{p}_i = p) = p, \forall p \in [0,1]$
      - $\hat{y}_i$ prediction, $\hat{p}_i$ associated confidence
    - Given 100 predictions with confidence 0.7, 70 of them should be correct
    - Impossible to achieve, but the closer the better
    - Approximate empirically

Solution

- Metrics to evaluate miscalibration
  - Perfect calibration
    - $Prob(\hat{y}_i = y \mid \hat{p}_i = p) = p, \forall p \in [0,1]$
      - $\hat{y}_i$ prediction, $\hat{p}_i$ associated confidence
    - Given 100 predictions with confidence 0.7, 70 of them should be correct
    - Impossible to achieve, but the closer the better
    - Approximate empirically
    - Group samples into M interval bins of size $1/M$
    - Let B_m is the set of sample indices in $\left(\frac{m-1}{M}, \frac{m}{M}\right]$
    - Empirical Accuracy
      - $acc(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m} \mathbf{1}(y_i = \hat{y}_i)$
    - Empirical Confidence
      - $conf(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m} \hat{p}_i$

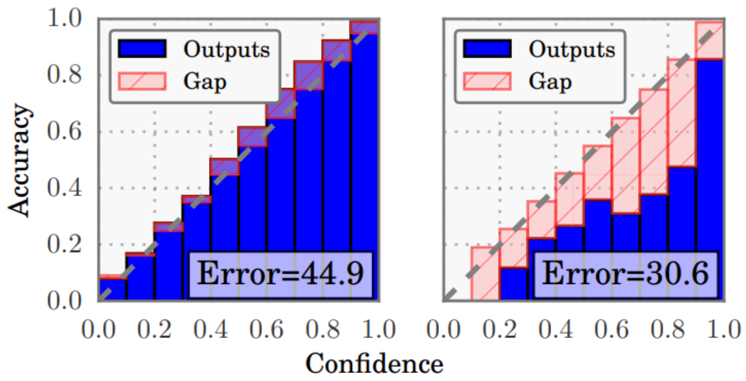
Solution

- Metrics to evaluate miscalibration
  - Reliability diagram
    - Acc vs Conf curve



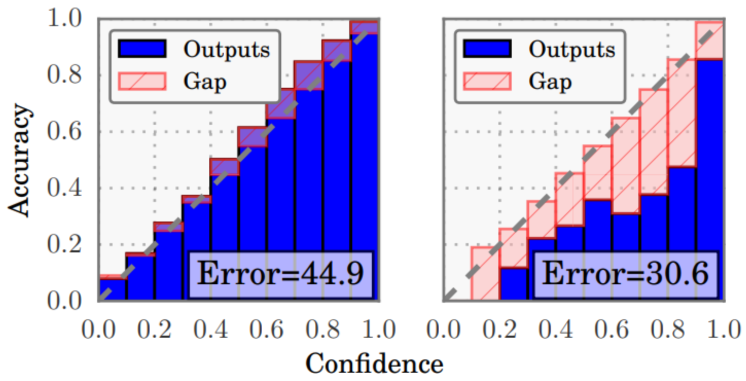
Solution

- Metrics to evaluate miscalibration
  - Reliability diagram
    - Acc vs Conf curve
  - Expected calibration error (ECE)
    - $ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$
    - Scalar summary statistics of reliability diagram



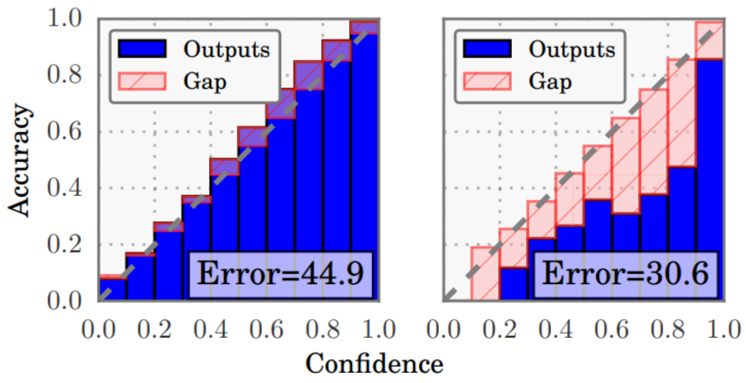
Solution

- Metrics to evaluate miscalibration
  - Reliability diagram
    - Acc vs Conf curve
  - Expected calibration error (ECE)
    - $ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$
    - Scalar summary statistics of reliability diagram
  - Maximum calibration error (MCE)
    - $MCE = \max_{m \in \{1,\dots,M\}} |acc(B_m) - conf(B_m)|$
    - Worst-case gap, critical for high-stakes apps



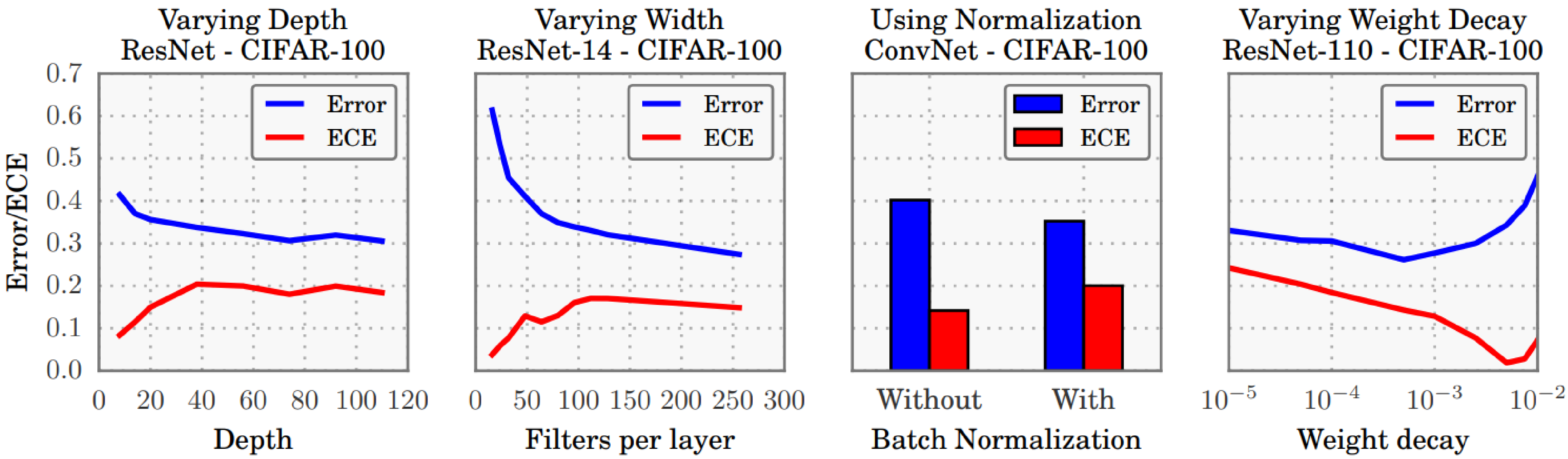
Solution

- Metrics to evaluate miscalibration
  - Reliability diagram
    - Acc vs Conf curve
  - Expected calibration error (ECE)
    - $ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$
    - Scalar summary statistics of reliability diagram
  - Maximum calibration error (MCE)
    - $MCE = \max_{m \in \{1,\ldots,M\}} |acc(B_m) - conf(B_m)|$
    - Worst-case gap, critical for high-stakes apps
  - Negative log likelihood (NLL)
    - Standard measure of quality for a probabilistic model



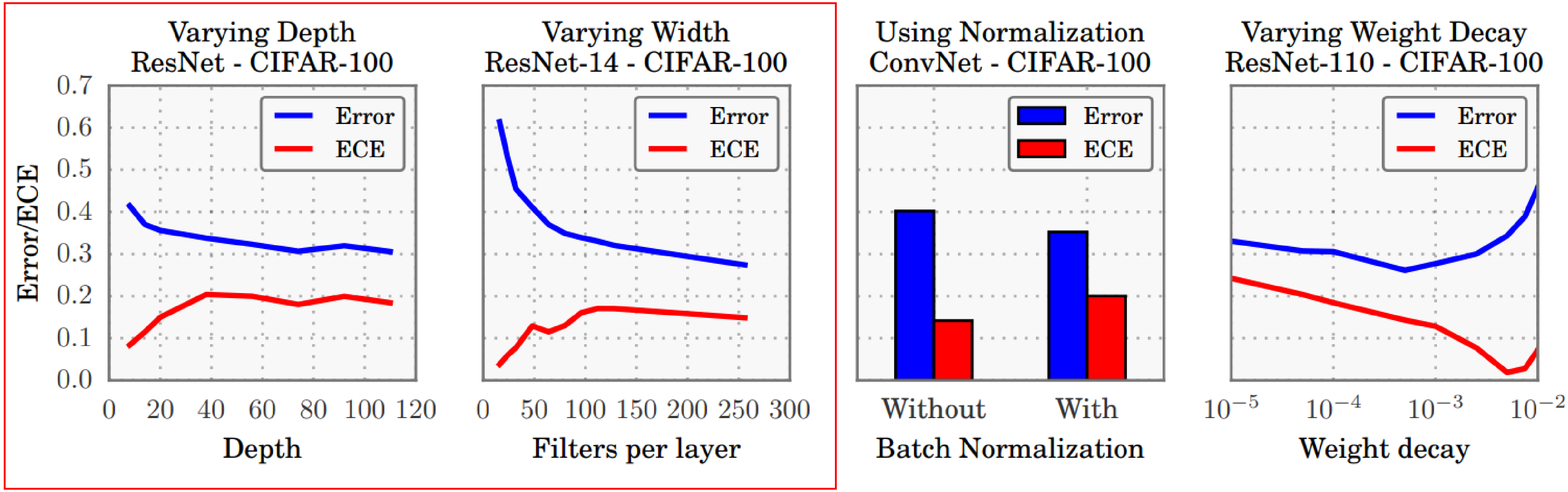
Solution

- Metrics to evaluate miscalibration ✓
- Factors for miscalibration



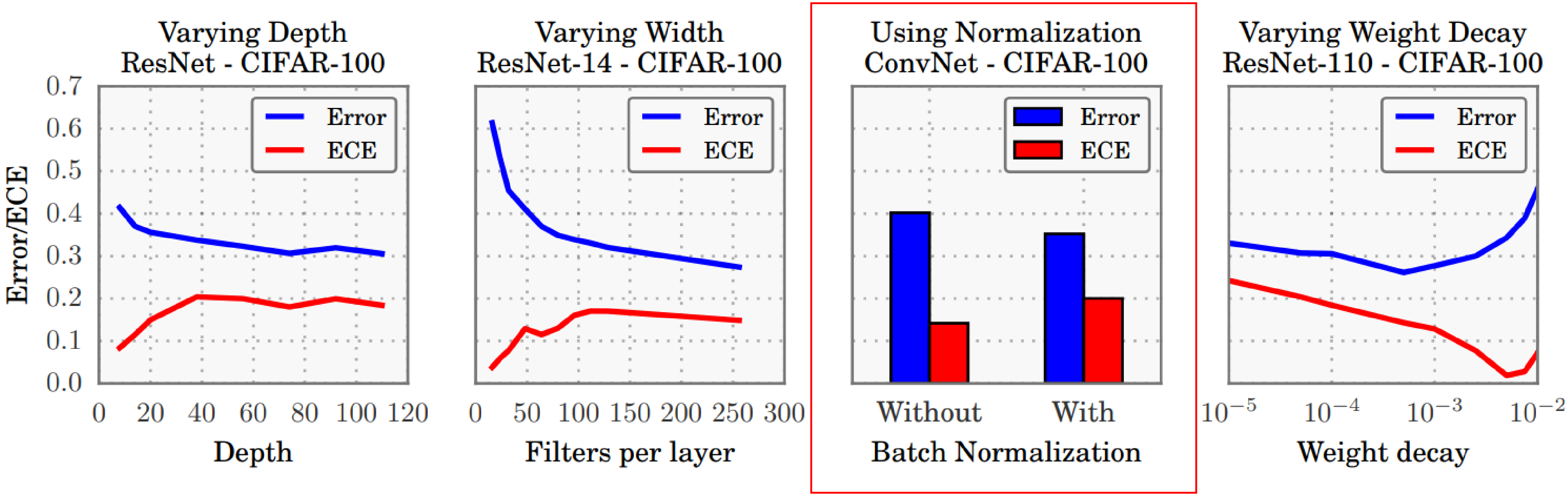
Solution

- Factors for miscalibration
  - Deeper & wider models => poor calibration



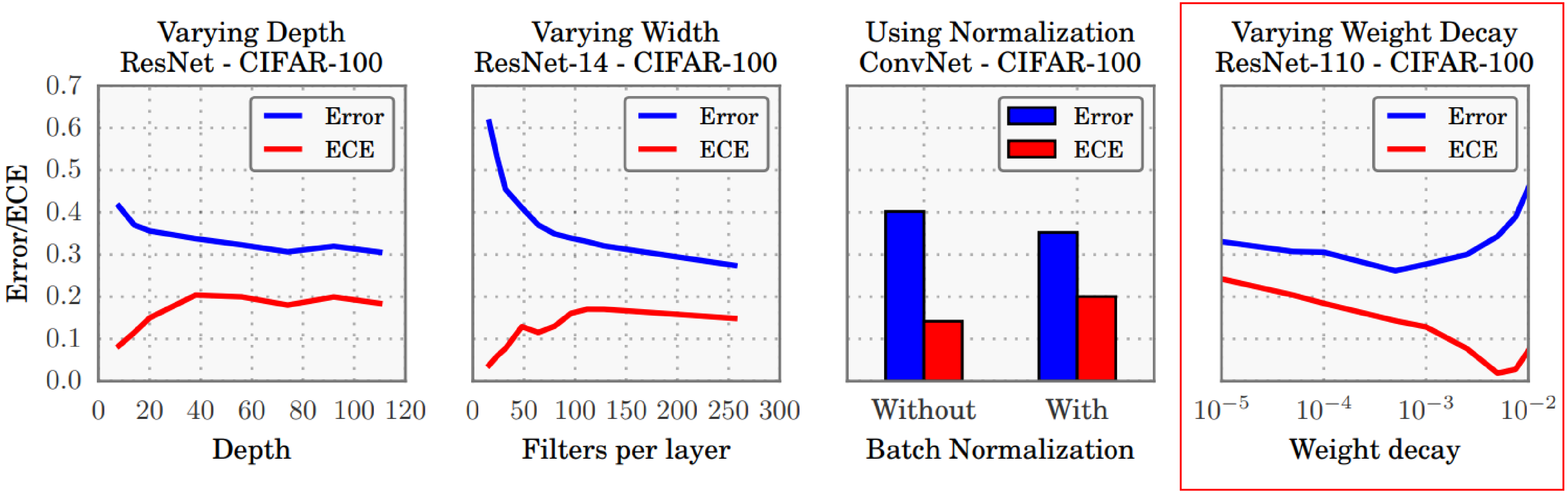
Solution

- Factors for miscalibration
  - Batch normalization => poor calibration



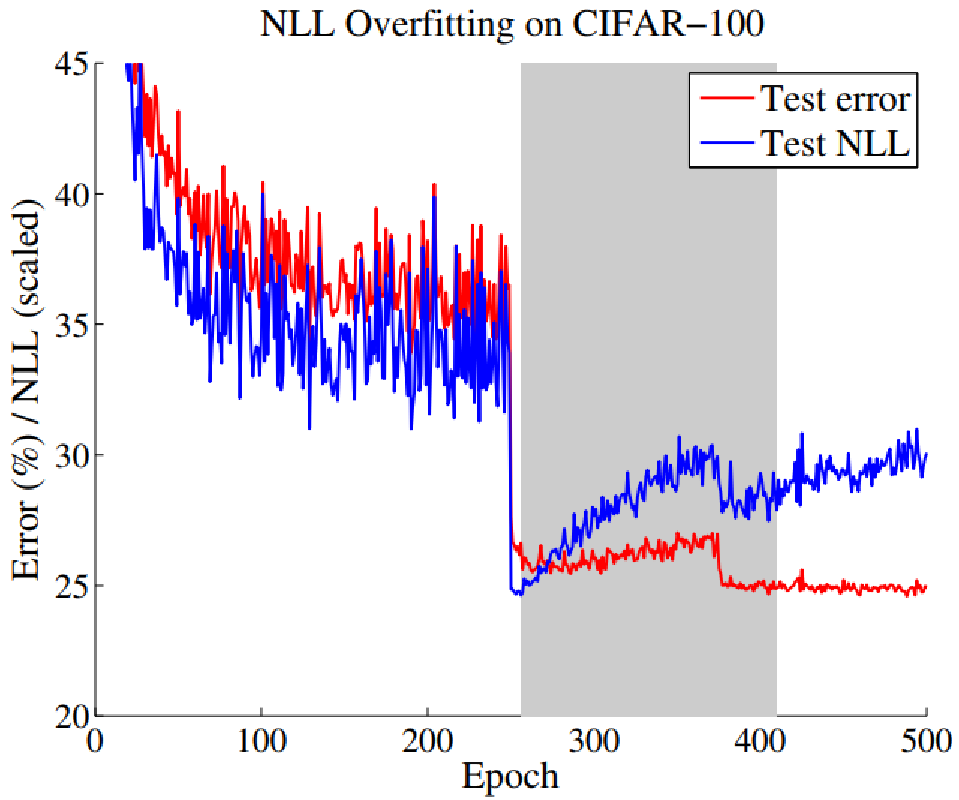
Solution

- Factors for miscalibration
  - Lack of regularization => poor calibration



Solution

- Factors for miscalibration
  - Disconnect between NLL and 0/1 loss
  - Better accuracy at the expense of well-calibrated model?



Solution

- Metrics to evaluate miscalibration ✓
- Factors for miscalibration ✓
- Solving miscalibration
  - Many approaches in literature
  - They used a single parameter variant of Platt scaling, “temperature scaling”

Solution

- Solving miscalibration
  - Temperature scaling
    - $q_i = \max_{k \in 1,\ldots,K} \sigma_{SM}\left(\frac{z_i}{T}\right)^{(k)}$
      - K classes, z_i original logit vector, $\sigma_{SM}(.)^{(k)}$ softmax function, q_i confidence

Solution

- Solving miscalibration
  - Temperature scaling
    - $q_i = \max_{k \in 1,\ldots,K} \sigma_{SM}\left(\frac{z_i}{T}\right)^{(k)}$
      - K classes, z_i original logit vector, $\sigma_{SM}(.)^{(k)}$ softmax function, q_i confidence
    - Decrease confidence of softmax output when $T > 1$

Solution

- Solving miscalibration
  - Temperature scaling
    - $q_i = \max_{k \in 1,\ldots,K} \sigma_{SM}\left(\frac{z_i}{T}\right)^{(k)}$
      - K classes, z_i original logit vector, $\sigma_{SM}(.)^{(k)}$ softmax function, q_i confidence
    - Decrease confidence of softmax output when $T > 1$
    - Optimize T over validation set (freeze model)

Solution

- Solving miscalibration
  - Temperature scaling
    - $q_i = \max_{k \in 1,\ldots,K} \sigma_{SM} \left(\frac{z_i}{T}\right)^{(k)}$
      - K classes, z_i original logit vector, $\sigma_{SM}(.)^{(k)}$ softmax function, q_i confidence
    - Decrease confidence of softmax output when $T > 1$
    - Optimize T over validation set (freeze model)
    - Performs best among other calibration choices

Solution

- Solving miscalibration
  - Temperature scaling
    - $q_i = \max_{k \in 1,\ldots,K} \sigma_{SM}\left(\frac{z_i}{T}\right)^{(k)}$
      - K classes, z_i original logit vector, $\sigma_{SM}(.)^{(k)}$ softmax function, q_i confidence
    - Decrease confidence of softmax output when $T > 1$
    - Optimize T over validation set (freeze model)
    - Performs best among other calibration choices
    - Does not change maximum of softmax function
      - Better calibration without decreasing accuracy

Discussion

- Addresses overconfidence problem in classification NNs

Discussion

- Addresses overconfidence problem in classification NNs
- Investigated possible reasons

Discussion

- Addresses overconfidence problem in classification NNs
- Investigated possible reasons
- Provided a solution with empirical success
  - Outperforms more complex approaches

Discussion

- Addresses overconfidence problem in classification NNs
- Investigated possible reasons
- Provided a solution with empirical success
  - Outperforms more complex approaches
- Analysis done for ID
  - OOD performance can be critical for practical apps

References

[1] A. Sax, J. O. Zhang, B. Emi, A. Zamir, S. Savarese, L. Guibas, and J. Malik, “Learning to navigate using mid-level visual priors,” in Conference on Robot Learning, 2020, pp. 791–812.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang et al., “End to end learning for self-driving cars,” arXiv preprint arXiv:1604.07316, 2016.

[3] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, “Dermatologist-level classification of skin cancer with deep neural networks,” nature, vol. 542, no. 7639, pp. 115–118, 2017.

[4] A. Kendall and Y. Gal, “What uncertainties do we need in Bayesian deep learning for computer vision?” in Advances in Neural Information Processing Systems, 2017, pp. 5574–5584.

[5] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in Advances in Neural Information Processing Systems, 2017, pp. 6402–6413.

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in International Conference on Machine Learning, 2017, pp. 1321–1330.

[7] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in International Conference on Machine Learning, 2016, pp. 1050–1059.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv preprint arXiv:1412.6572, 2014.

[9] J. Jo and Y. Bengio, “Measuring the tendency of cnns to learn surface statistical regularities,” arXiv preprint arXiv:1711.11561, 2017.

[10] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, “A fourier perspective on model robustness in computer vision,” in Advances in Neural Information Processing Systems, 2019, pp. 13 276–13 286.

Thank You