Hot Topics in

Machine Learning Safety

Uncertainty Quantification

WS 2023

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Topics

- Introduction and Motivation
- ML Fundamentals
- Testing
- Explainability
- **Uncertainty Estimation** 👈
- Adversarial Machine Learning
- Anomaly Detection
- Alignment

Agenda

- What is uncertainty?
- Where do we use uncertainty?
- Softmax
- Calibration
- Types & Sources of Uncertainty
- Estimating Uncertainty
- Comparing Baselines
- Open Challenges

Uncertainty

Classical ML theory

- Considers ML systems as standalone components
- Focus: Increasing the Accuracy/KPIs

However:

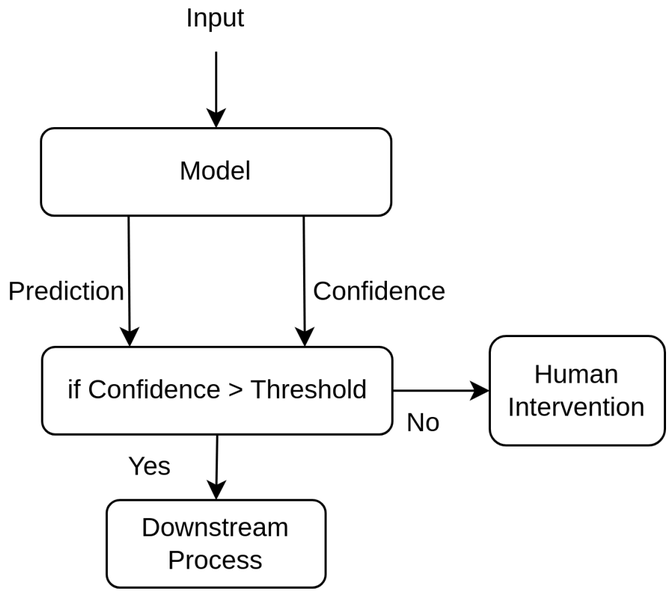
- ML components are part of a larger systems/applications
- It is desirable to make a model to quantify its own uncertainty

Uncertainty Quantification

- Making a model/system say "I am not sure" on ambiguous conditions

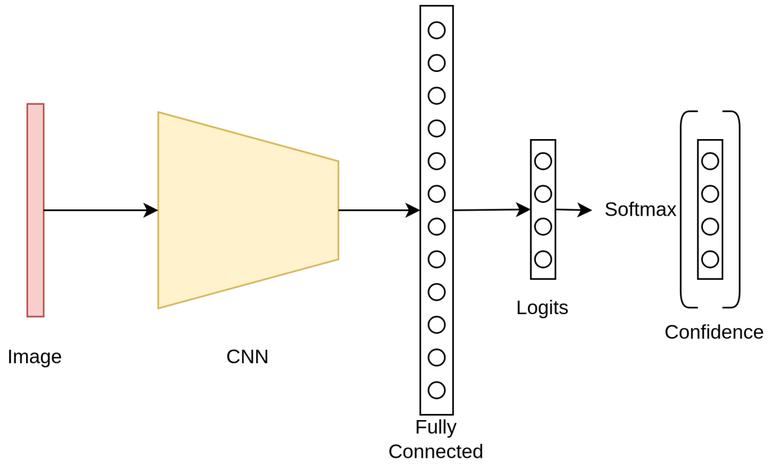
Where do we use Uncertainty?

Decide when to trust the model or when to call for human intervention



"All models are wrong, but ~~some~~ models that know when they are wrong are useful"

Softmax in Deep Neural Networks



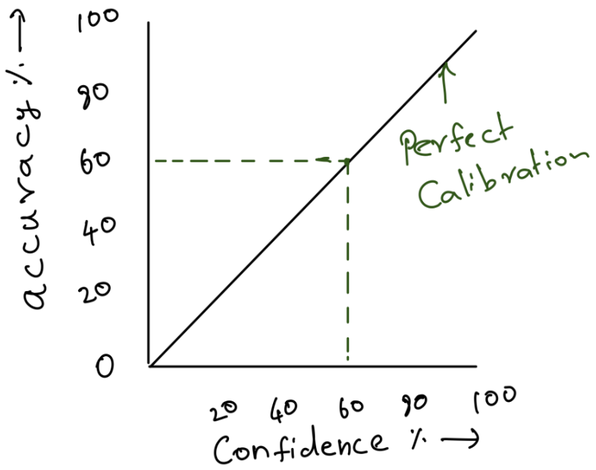
$$p(y_i \mid x) \approx \text{softmax}(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_{j=1}^{k} e^{\hat{y}_j}}$$

Isn't this enough?

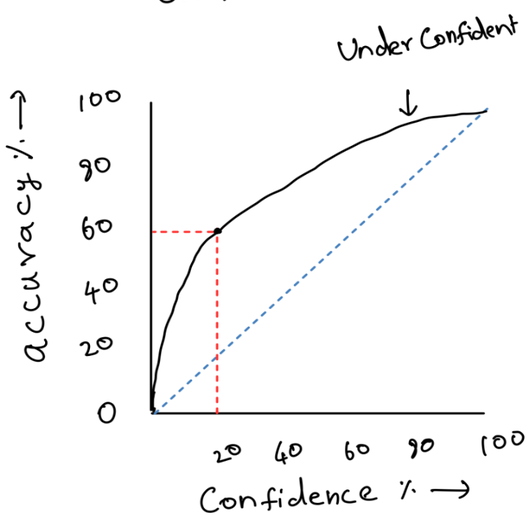
Calibration

Alignment of the models's confidence with its accuracy

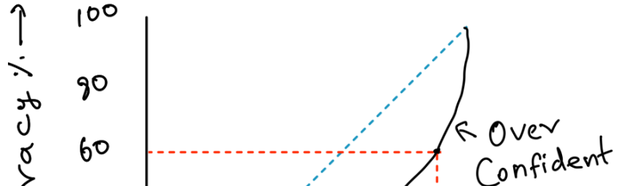
Case 1 : a model is 60% confident that it will rain, 100 days of observation ; It rained on 60 days = **Perfect Calibration**



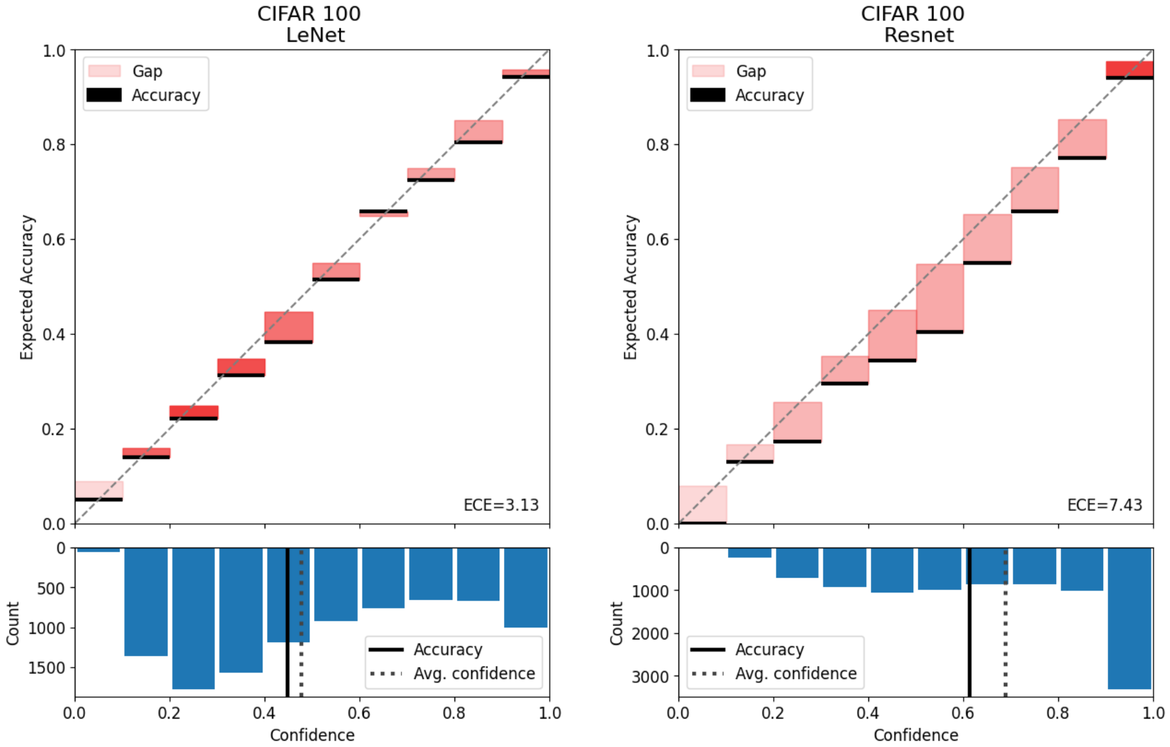
Case 2 : a model is 20% confident that it will rain, 100 days of observation ; It rained on 60 days = **Under Confident Predictions**



Case 3 : a model is 90% confident that it will rain, 100 days of



Calibration issues with modern networks



Expected Calibration Error (ECE) :

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

M = Number of Bins

B_m = predictions at m^{th} bin

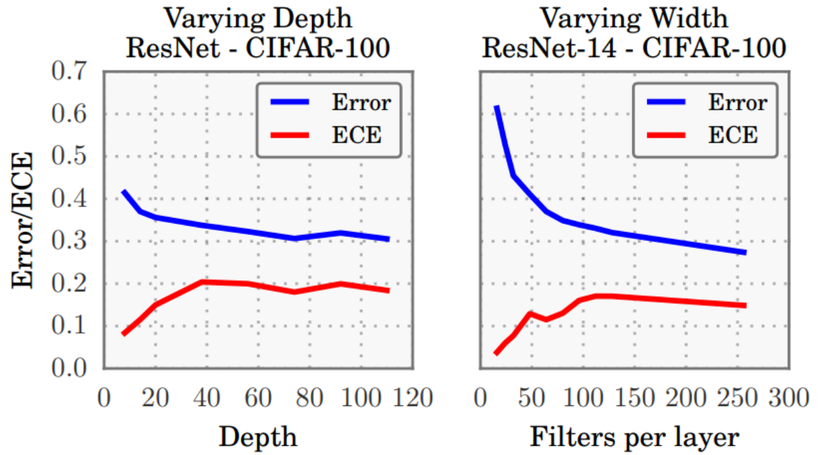
n = Number of datapoints

On Calibration of Modern Neural Networks Huo et.al

Why does this happen?

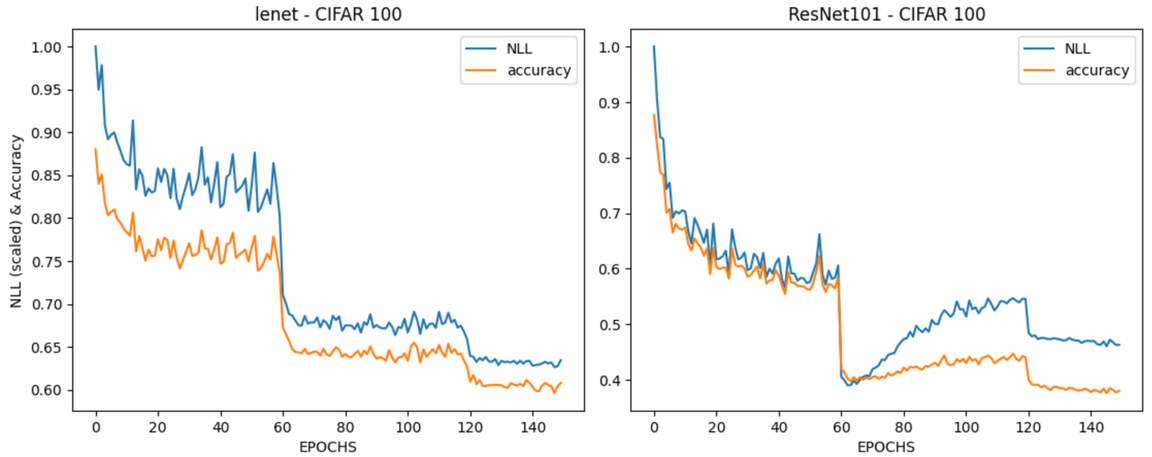
Model Capacity:

- Larger NNs: superior generalization but reduced calibration
- During training: NLL (Cross Entropy) Loss can further be reduced by increasing the confidence



NLL Optimization:

- Disconnect between NLL and accuracy
- NLL overfits during the training
- The network learns better classification accuracy at the expence of well modeled probabilities



Why compromise accuracy for calibration?

Assume a Cost sensitive - discrete decision making problem

Example cost matrix for decision making in health care:

		True Label	
		Healthy	**Diseased**
Action	Predict Healthy	0	10
	Predict Diseased	1	0
	I don't know	0.5	0.5

- We will incure heavy losses if the model predicts false negatives
- The cost of misclassifying a diseased patient as healthy is high

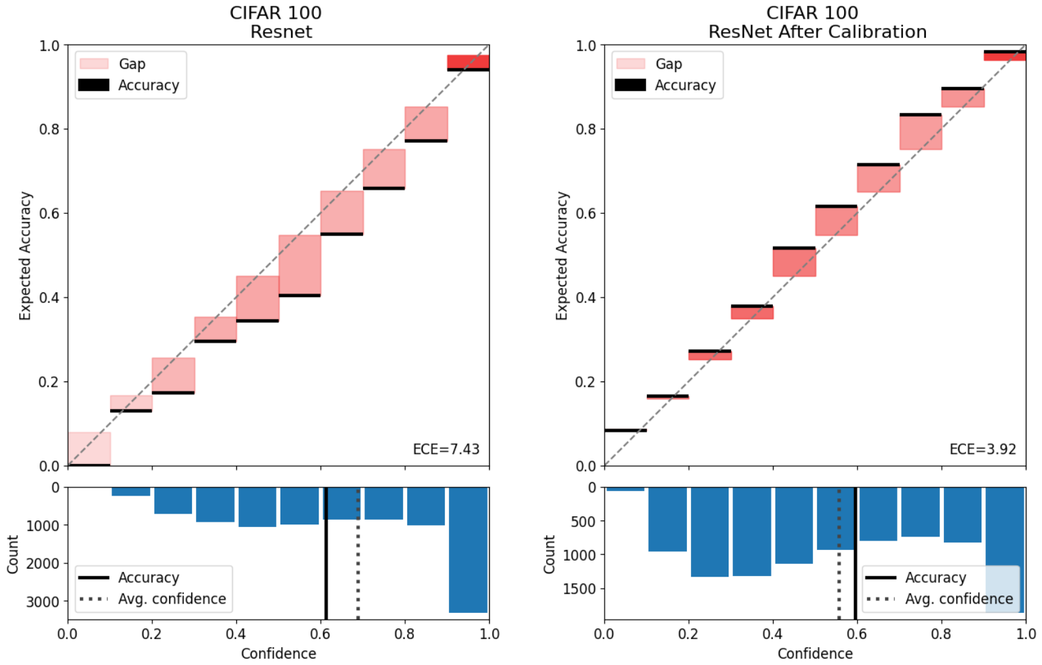
Quick Fix - Temperature Scaling

$$p(y_i \mid x) \approx \text{softmax}(\hat{y}_i) = \frac{e^{\frac{\hat{y}_i}{T}}}{\sum_{j=1}^{k} e^{\frac{\hat{y}_j}{T}}}$$

- T - Temperature > 1
- Freeze the classification model
- Parameterized by optimizing on the validation set
- Does not influence accuracy because T is class-independent

```
temperature = nn.Parameter(torch.ones(1)) # Temperature Parameter
optimizer = optim.SGD([self.temperature], lr=0.01) # Optimizer
model.eval() # Freeze Model
for data, labels in val_loader:
  with torch.nograd():
    logit = model(data)
    logits.append(logit)
    labels.append(label)
logits = torch.cat(logits, dim = 0)
labels = torch.cat(labels, dim = 0)
loss = nll_loss(log_softmax(logits/temperature), labels)
loss.backward()
optimizer.step()
confidence_scaled = softmax(logit/temperature)
```

Quick Fix - Temperature Scaling



Types and Sources of Uncertainty

Example

- Biased coin with unknown $P(1) = \frac{7}{10}$
- Dataset $\mathcal{D} = \{1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0\}$
- Maximum Likelihood guess $\hat{P} = \frac{7}{11}$

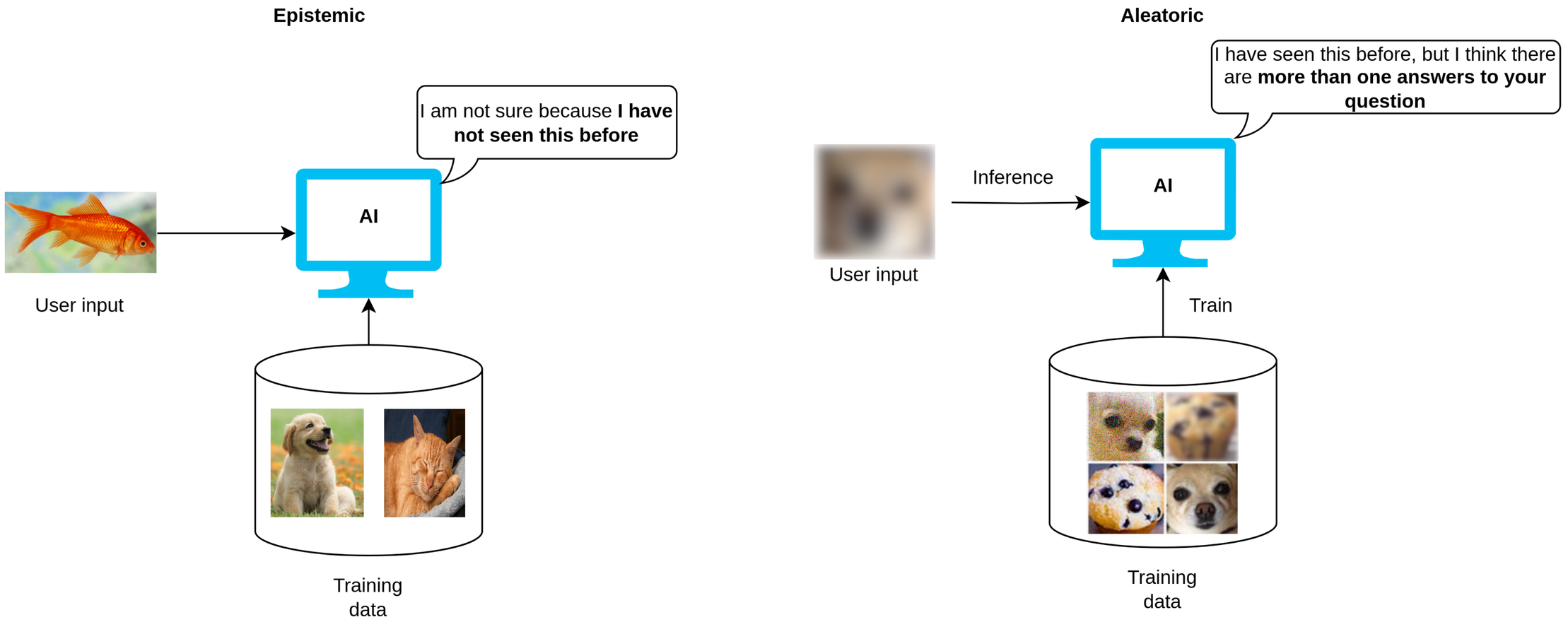
Epistemic Uncertainty/Model Uncertainty

- Uncertainty caused by lack of knowledge
- Could be reduced by including other features (like toss pattern, coin weight, etc.)
- Vanishes with infinite data

Aleatoric Uncertainty/Data Uncertainty

- Coin flip is fundamentally stochastic: impossible to precisely predict the next flip
- Inherent stochasticity present in the data (labeling noise, measurement noise, etc.)
- Irreducible, even with infinite data

Types of Uncertainties



How to estimate Uncertainty?

Learning: finding the best parameters, e.g. $\theta^\star = \arg\min -\log p(y \mid x, \theta)$

Problem:

- Single set of parameters $\longrightarrow$ one **most likely** prediction

Is there only one set of parameters?

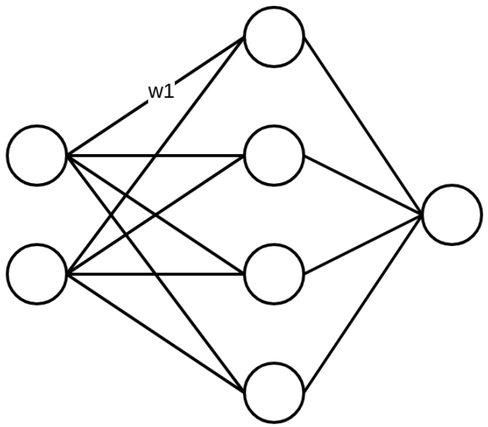
- Probabilistic Approach: Estimate a distribution over possible $\theta^\star$
- Computationaly feasible Approach: Obtain multiple good $\theta^\star$

In the following:

- Bayesian Neural Networks
- Ensemble Methods
- Test Time Augmentation
- Single Deterministic Methods

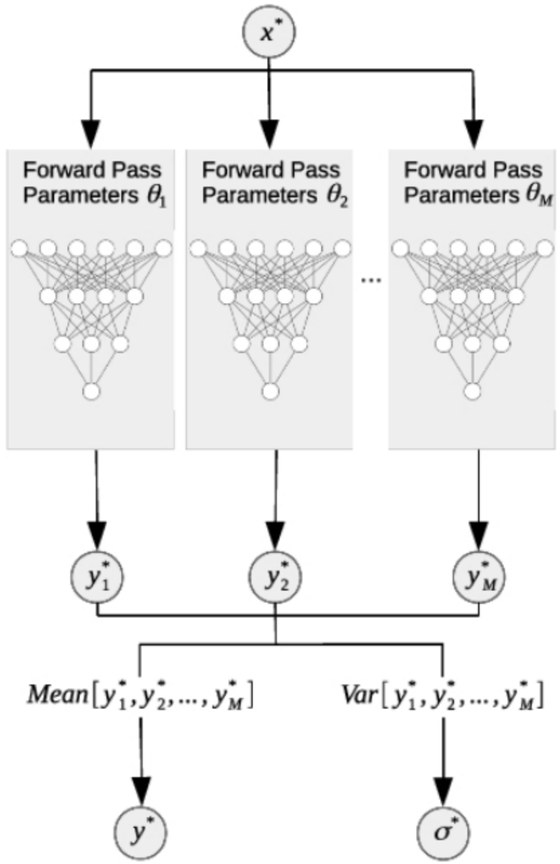
Bayesian Neural Networks

- During training: fit a distribution for each weight
- During inference: sample weights from distributions
- Propagate input forward for each set of sampled weights
- Compute the mean and the variance of predictions
- Mean $\longrightarrow$ point prediction ; Variance $\longrightarrow$ predictive variance



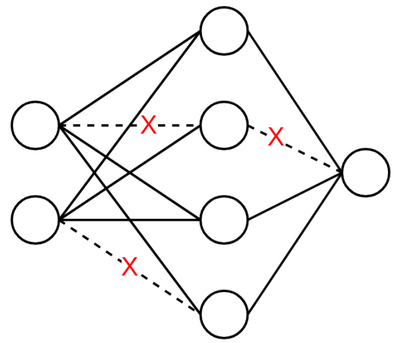
Deterministic Model
w1 = point

Probabilistic Model
w1 = Distribution N(μ, σ)



Dropouts as Bayesian Approximation

- Dropout: Regularization technique for improving generalization
- Weights are randomly turned off during training
- Stochastically dropping weights creates ensembles of architecture
- During inference: multiple forward passes with active dropout
- Mean $\longrightarrow$ point prediction ; Variance $\longrightarrow$ predictive variance



```python
model = load_model(weights)
enable_dropout(model)
results = []
samples = 50 #Number of samples

for m in range(samples):
  with torch.no_grad():
    y_hat = model(x.cuda())
    results.append(y_hat)

results = torch.cat(results) #Results from all samples
mean_logits = results.mean(dim = 0)
confidence, predictions = torch.max(mean_logits, dim=0) # Confidence and Predictions
predictive_variances = torch.var(results, dim = 0).mean(dim = 0) # Predictive Variance
```

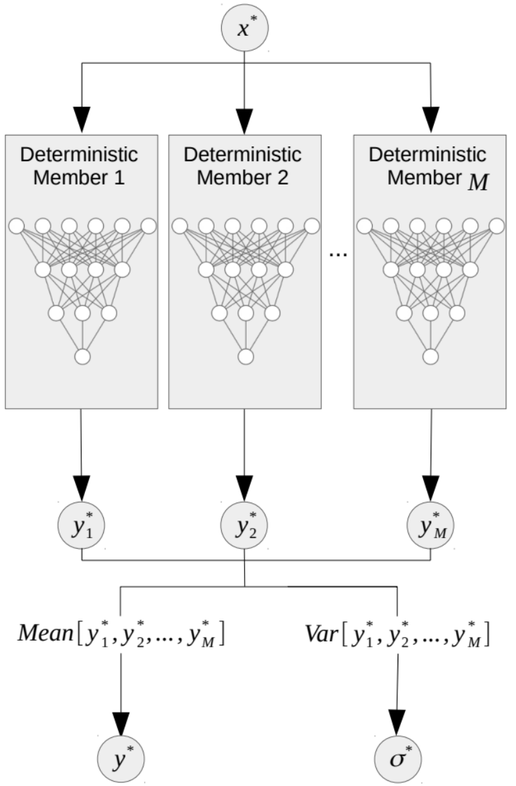
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Deep Ensembles

Ensembles

- Train multiple models
  - Multiple sets of training data (cross-validation)
  - Train models with different random initializations
  - Change the architecture structure (receptive field, number of features...)

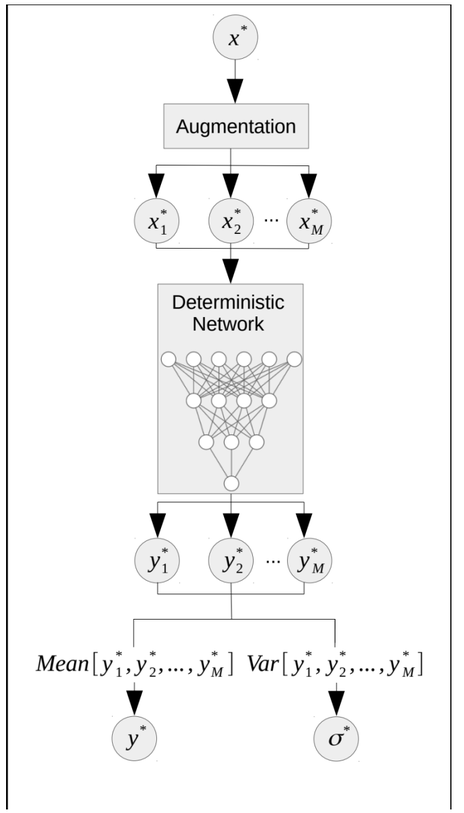
- Aggregate predictions from multiple models
- Mean $\longrightarrow$ point prediction ; Variance $\longrightarrow$ predictive variance
- Ensembles capture a broad range of patterns and uncertainty in the data



Test Time Augmentations

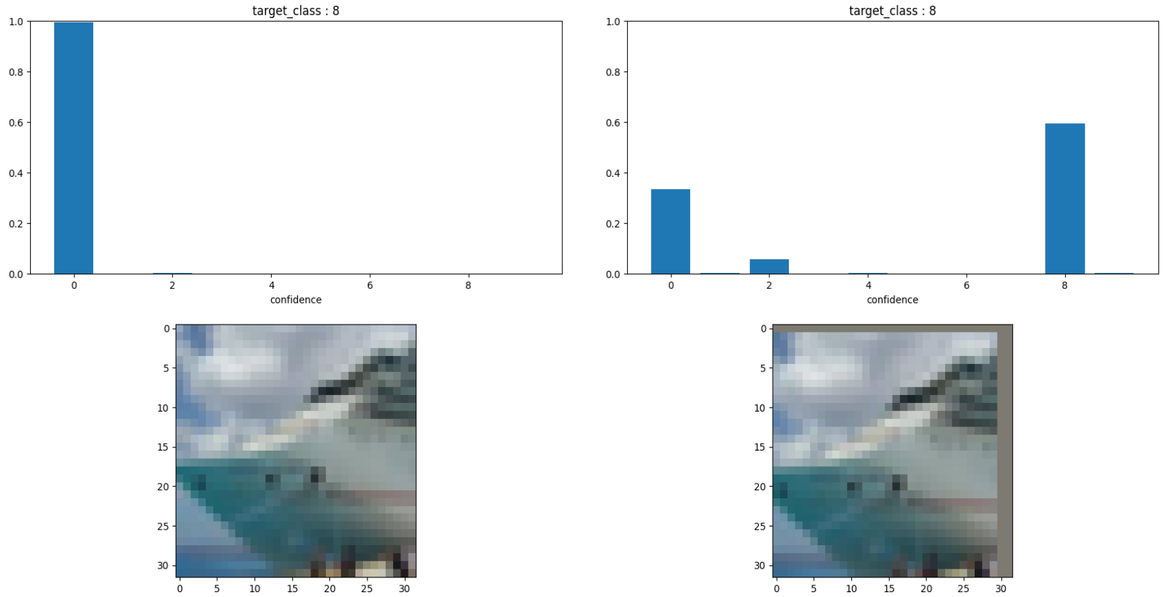
- Freeze the weights of a trained model
- Create multiple test samples from a single test sample
  - Flipping, Rotation, Scaling, Translation, etc.

- Aggregate the predictions from all the perspectives
- Intuition: exploring the same image in different perspective to estimate uncertainty

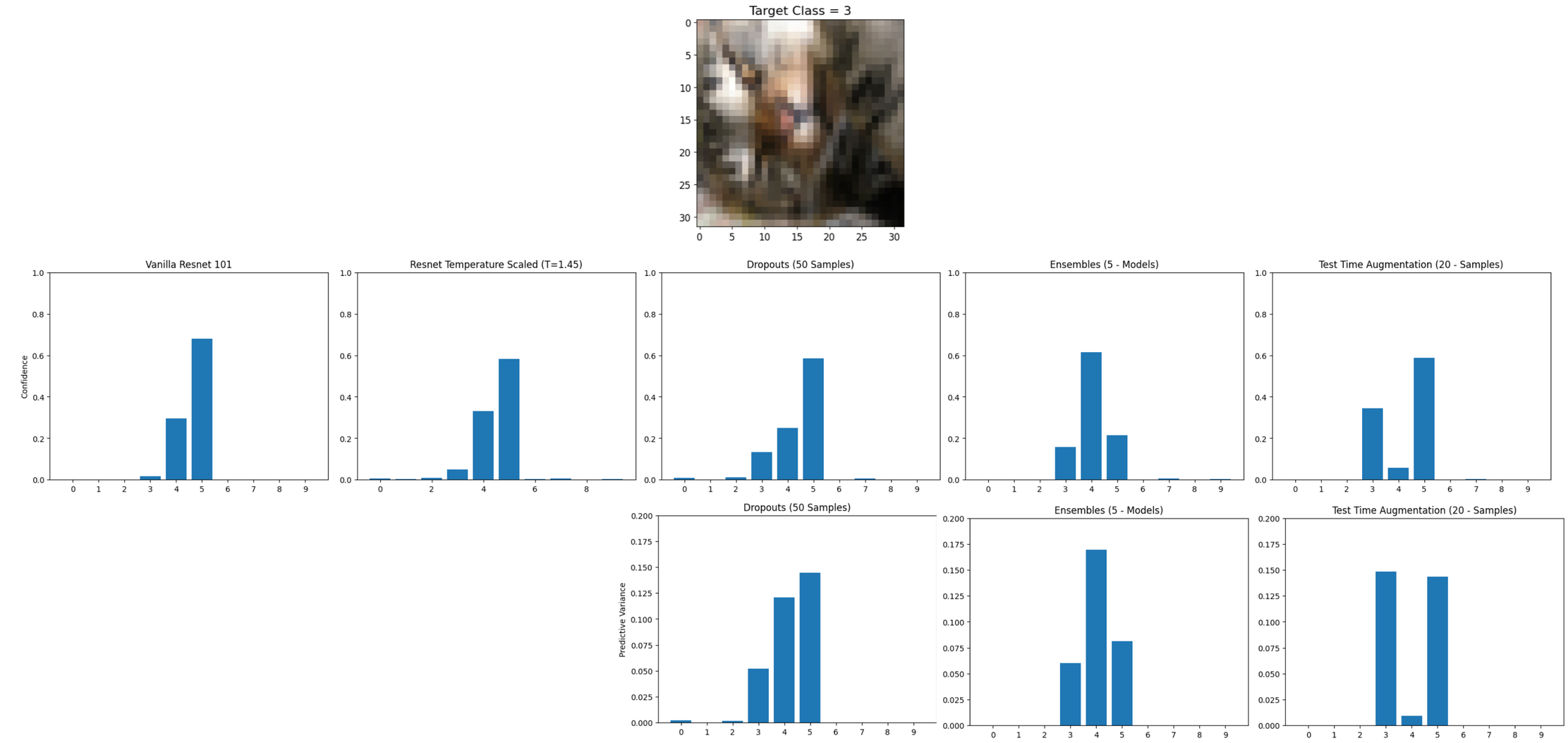


Test Time Augmentations

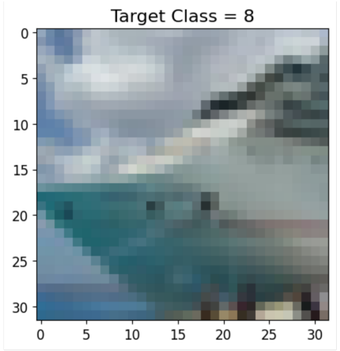
- Underlying model remains unchanged, requires no additional data necessary
- The augmentation must be valid and must not make the object unrecognizable

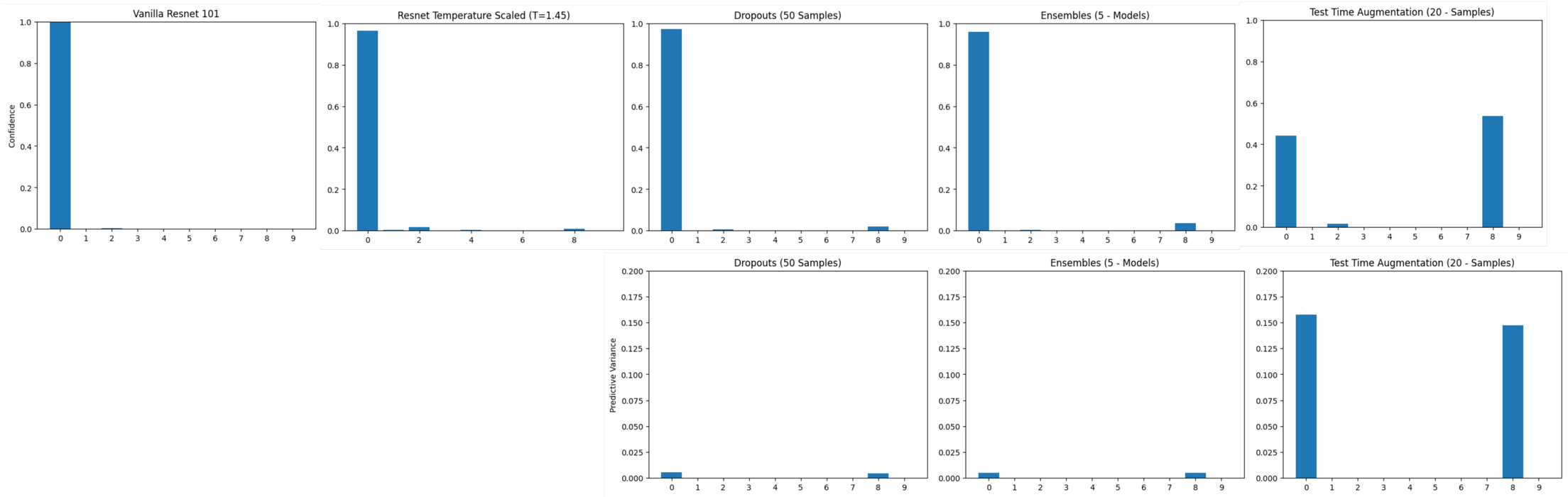


Baselines - Result Comparisons



Baselines - Result Comparisons

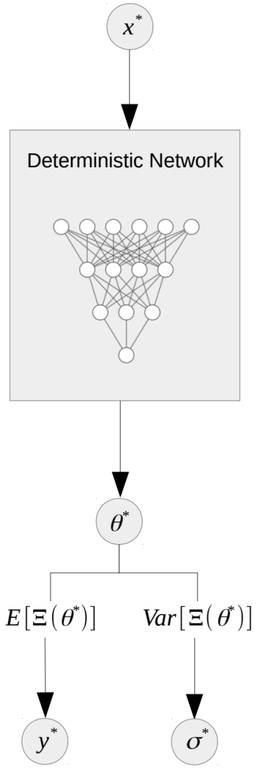




Methods - Deterministic Networks

- Previous methods:
  - Requires multiple feed forward passes
  - Not ideal for runtime applications where fast decisions are required

- Deterministic models: directly predict variances (uncertainties) and point predictions
- Active area of research



Deterministic Networks - Distance-based

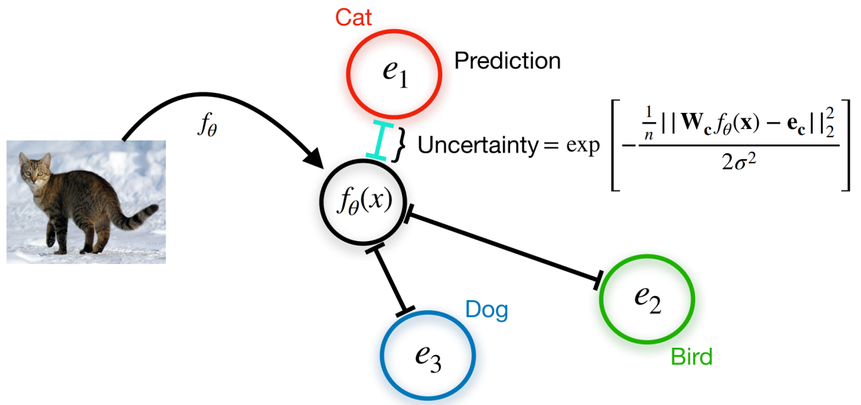


Figure 2. A depiction of the architecture of DUQ. The input is mapped to feature space, where it is assigned to the closest centroid. The distance to that centroid is the uncertainty.

Evaluating the Uncertainty Estimates

- Previously: Calibration Error ↓
- Brier Score (BS) ↑

$$BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2$$

N = Number of events

f_t = *forecast of* t^{th} *event*

o_t = *outcome (0 or 1) of* t^{th} *event*

- Gaussian Negative Log Likelihood (GNLL)

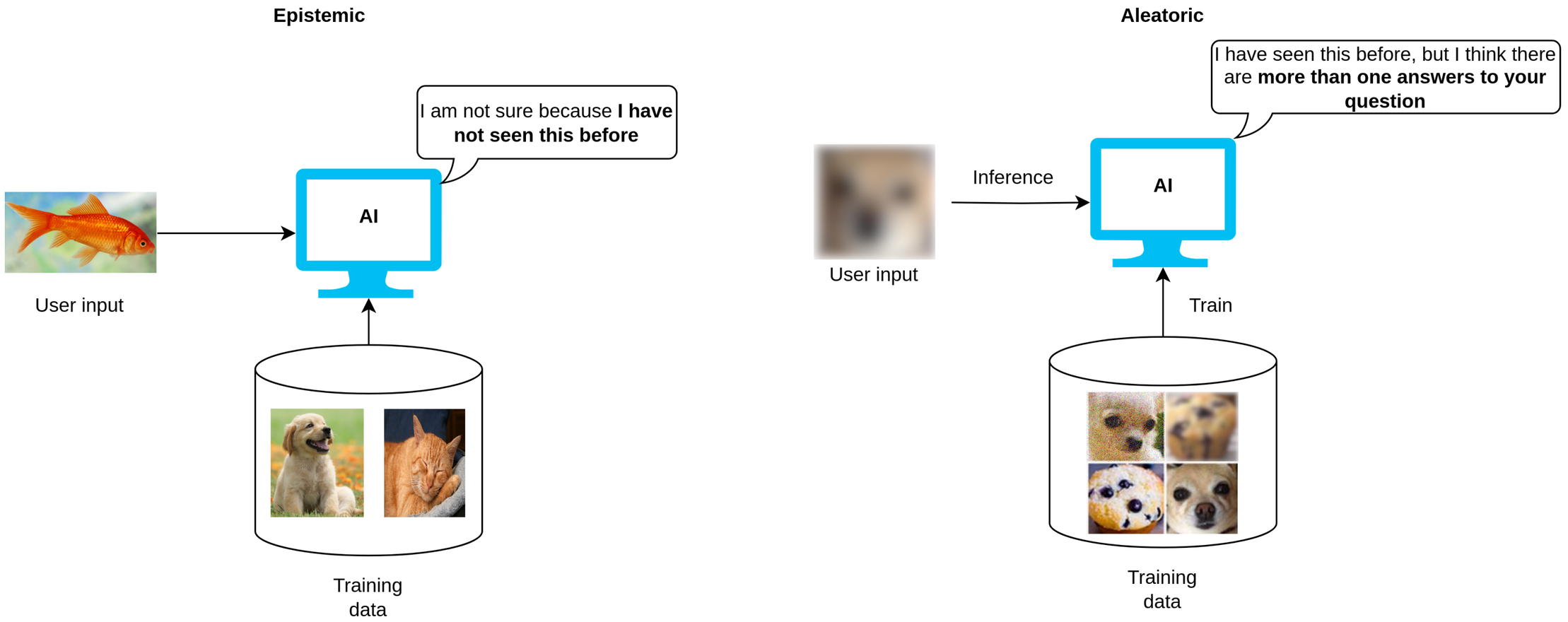
$$GNLL = \frac{1}{2}(\log(var) + \frac{(pred-target)^2}{var})$$

var = Predictive Variance

pred = point prediction

- Loss - Variance Correlation
  - Linear relation is not guaranteed, use a rank based correlation (spearman R)

Types of Uncertainties



How to estimate Uncertainty?

Learning: finding the best parameters, e.g. $\theta^\star = \arg\min -\log p(y \mid x, \theta)$

Problem:

- Single set of parameters $\longrightarrow$ one **most likely** prediction

Is there only one set of parameters?

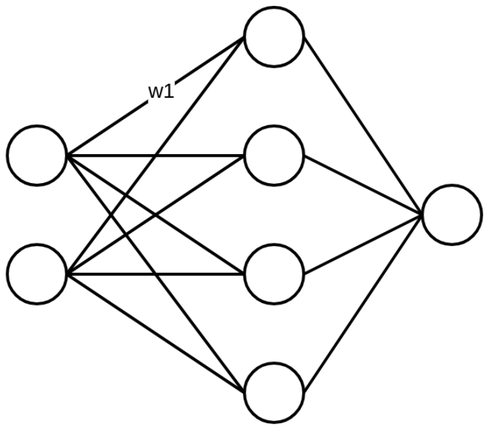
- Probabilistic Approach: Estimate a distribution over possible $\theta^\star$
- Computationaly feasible Approach: Obtain multiple good $\theta^\star$

In the following:

- Bayesian Neural Networks
- Ensemble Methods
- Test Time Augmentation
- Single Deterministic Methods

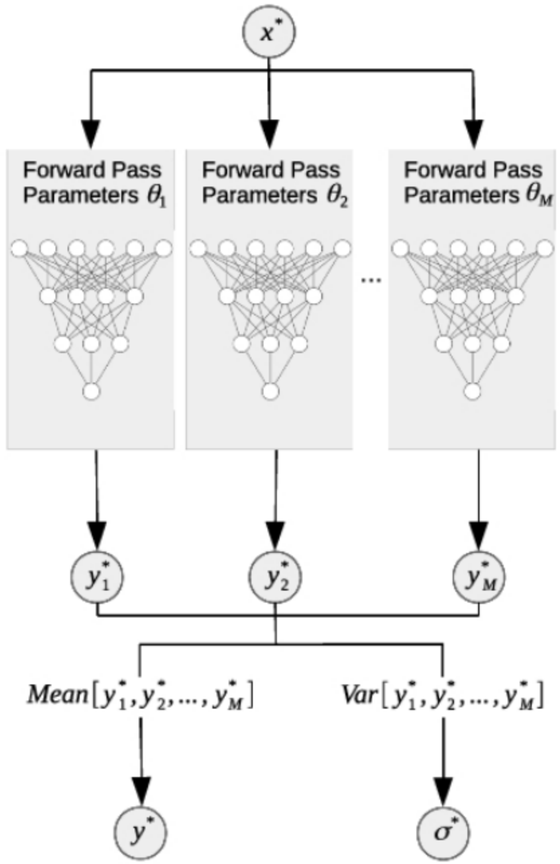
Bayesian Neural Networks

- During training: fit a distribution for each weight
- During inference: sample weights from distributions
- Propagate input forward for each set of sampled weights
- Compute the mean and the variance of predictions
- Mean $\longrightarrow$ point prediction ; Variance $\longrightarrow$ predictive variance



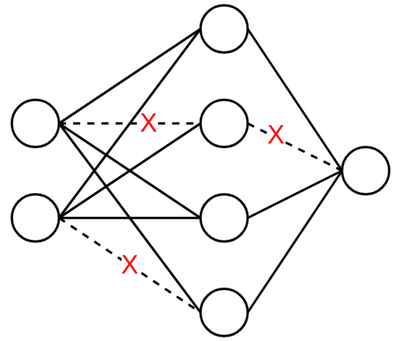
Deterministic Model
w1 = point

Probabilistic Model
w1 = Distribution N(μ, σ)



Dropouts as Bayesian Approximation

- Dropout: Regularization technique for improving generalization
- Weights are randomly turned off during training
- Stochastically dropping weights creates ensembles of architecture
- During inference: multiple forward passes with active dropout
- Mean $\longrightarrow$ point prediction ; Variance $\longrightarrow$ predictive variance



```python
model = load_model(weights)
enable_dropout(model)
results = []
samples = 50 #Number of samples

for m in range(samples):
  with torch.no_grad():
    y_hat = model(x.cuda())
    results.append(y_hat)

results = torch.cat(results) #Results from all samples
mean_logits = results.mean(dim = 0)
confidence, predictions = torch.max(mean_logits, dim=0) # Confidence and Predictions
predictive_variances = torch.var(results, dim = 0).mean(dim = 0) # Predictive Variance
```

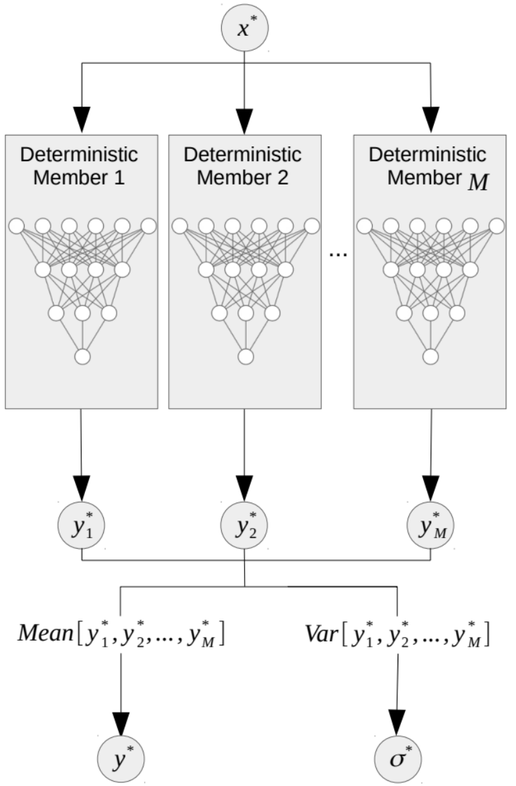
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Deep Ensembles

Ensembles

- Train multiple models
  - Multiple sets of training data (cross-validation)
  - Train models with different random initializations
  - Change the architecture structure (receptive field, number of features...)

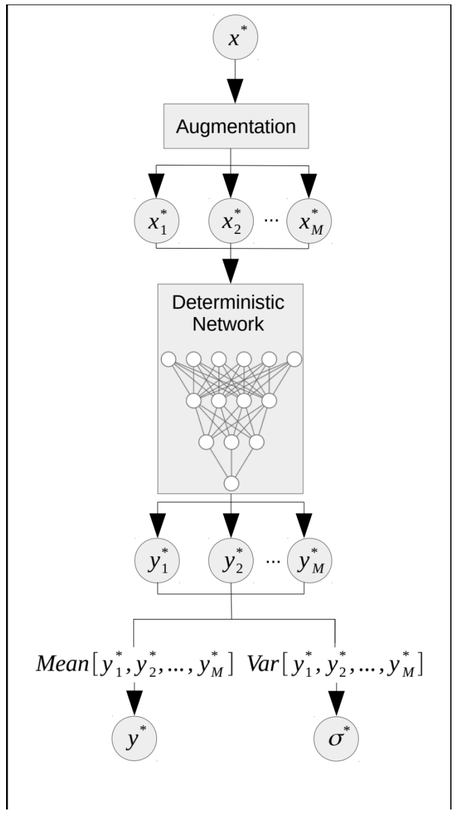
- Aggregate predictions from multiple models
- Mean $\longrightarrow$ point prediction ; Variance $\longrightarrow$ predictive variance
- Ensembles capture a broad range of patterns and uncertainty in the data



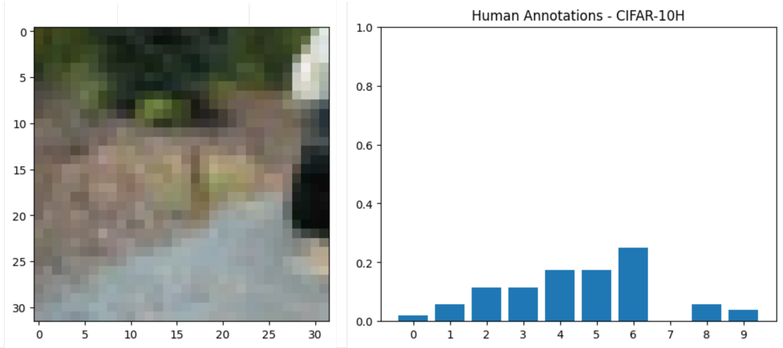
Test Time Augmentations

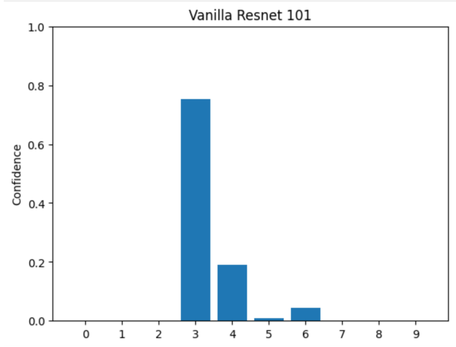
- Freeze the weights of a trained model
- Create multiple test samples from a single test sample
  - Flipping, Rotation, Scaling, Translation, etc.

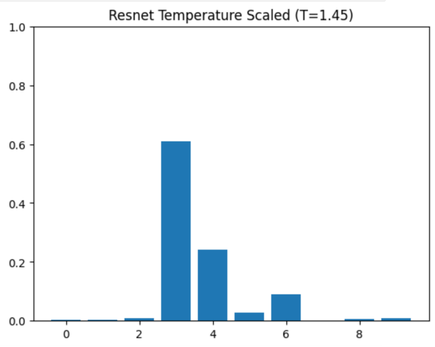
- Aggregate the predictions from all the perspectives
- Intuition: exploring the same image in different perspective to estimate uncertainty

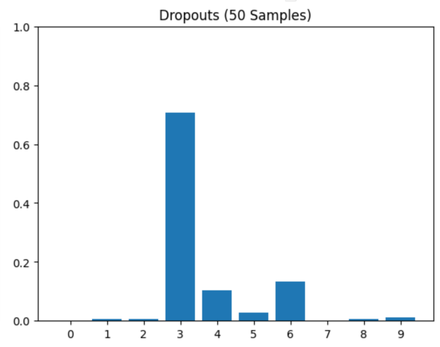


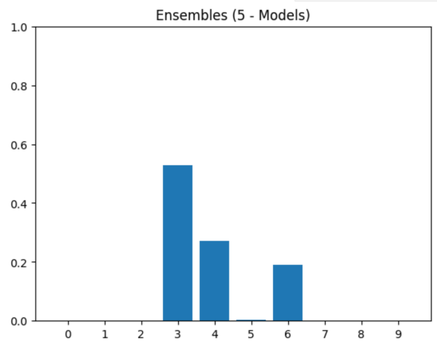
Comparing with the human annotations

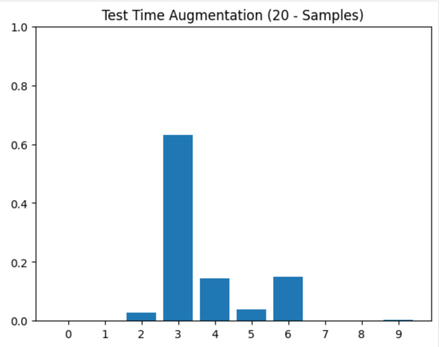


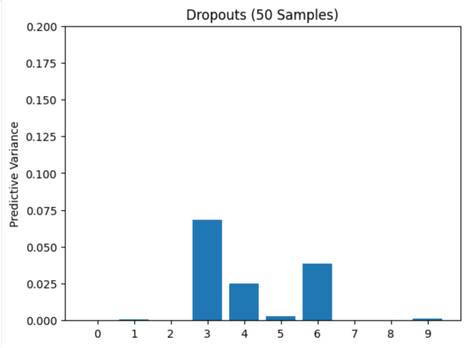


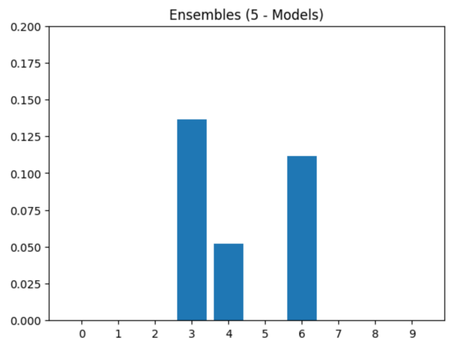


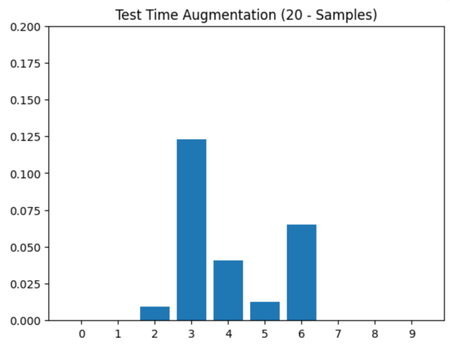












Challenges and Open Question

- Validation of existing methods over real world problems
- Lack of ground truth uncertainties
- Explainability
  - What (feature) caused a highly uncertain prediction?

- Quality of uncertainty depends on the underlying methods
  - What kind of architectural variations in ensembles?
  - What kind of data augmentations in TTA?

- Disentangling Epistemic and Aleatoric uncertainty

Reading Suggestions

- On Calibration of Modern Neural Networks
  - Foundational Paper
  - Arxiv: https://arxiv.org/pdf/1706.04599.pdf

- Introduction to Uncertainty in Deep Learning
  - https://www.gatsby.ucl.ac.uk/~balaji/balaji-uncertainty-talk-cifar-dlrl.pdf