



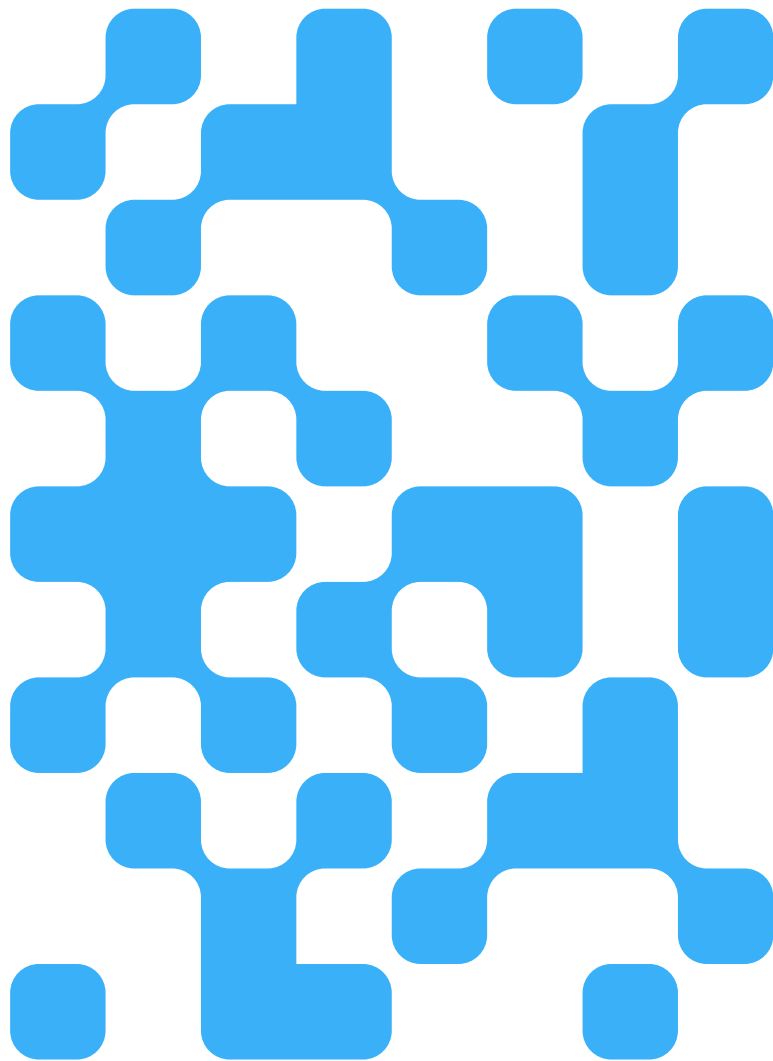
Machine Learning

2024 (ML-2024)

Lecture 6. Loss functions. Uncertainty estimation

One way to make AI more natural is to teach it to doubt. Two main questions arise. How do you train a model to estimate the uncertainties of its own predictions? What to do with the uncertain predictions if they arise?

by Alexei Valerievich Kornaev, Dr. habil. in Eng. Sc.,
Researcher at the RC for AI, Assoc. Prof. of the Robotics and CV
Master's Program, [Innopolis University](#)
Researcher at the RC for AI, [National RC for Oncology n.a. NN Blohin](#)
Professor at the Dept. of Mechatronics, Mechanics, and Robotics,
[Orel State University](#)



Agenda

- I. MAXIMUM LIKELIHOOD CRITERION (MLC)
- II. UNCERTAINTY ESTIMATION USING MLC
- III. UNCERTAINTY ESTIMATION IN GENERAL

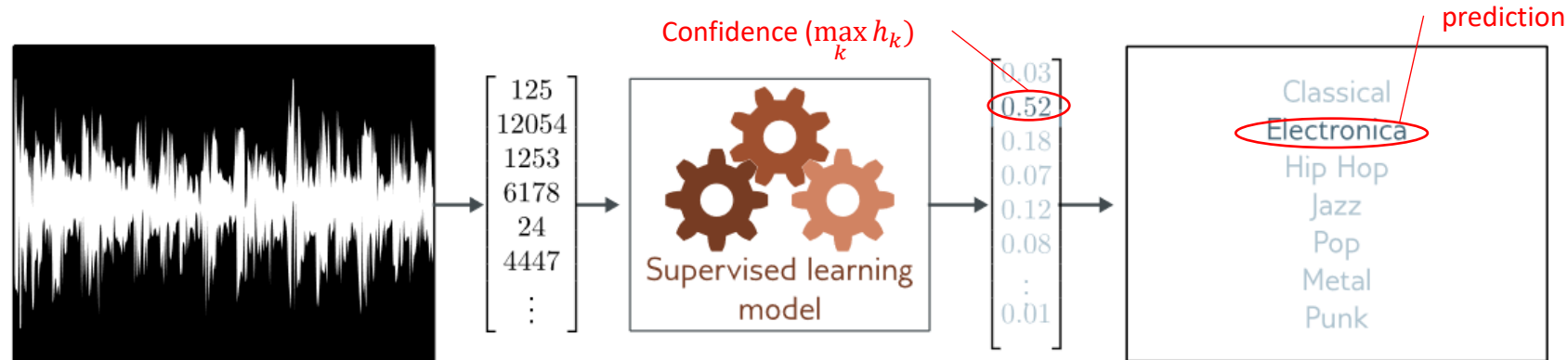
All models are wrong, but **some models that know when they are wrong**, are useful
/George Box + unknown researcher from the Google AI Brain Team/



Frequentist **vs** Bayesian frameworks

	Frequentist	Bayesian
Randomness	Objective indefiniteness	Subjective ignorance
Inference	Random and Deterministic	Everything is random
Estimates	Maximal likelihood	Bayes theorem
Applicability	$n \gg \text{size}(\theta)$	$\forall n$

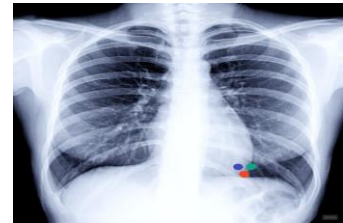
Confidence intuition



Supervised learning intuition

S. J. Prince. Understanding Deep Learning. MIT Press, 2023. URL <http://udlbook.com>.

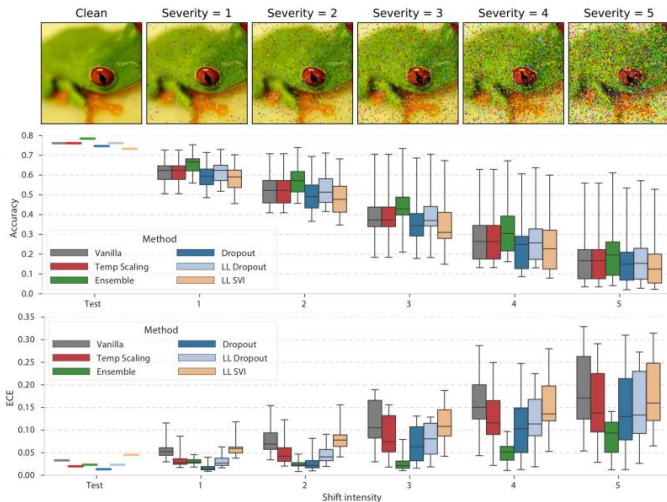
Models may give wrong predictions with a high confidence for both original, and noisy or synthetic inputs



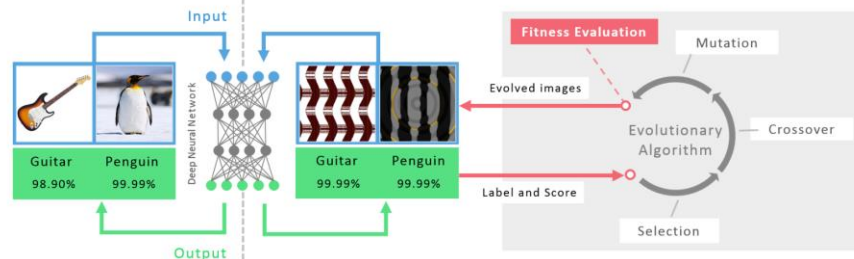
Models may give wrong predictions with a high confidence for both original, and noisy or synthetic inputs

- Accuracy drops with increasing shift on Imagenet-C

- Calibration degrades with shift -> "overconfident mistakes"



- State-of-the-art DNNs can recognize real images with high confidence
- But DNNs are also easily fooled: images can be produced that are unrecognizable to humans, but DNNs believe with 99.99% certainty are natural objects



Uncertainty and Out-of-Distribution Robustness in Deep Learning, [NeurIPS 2020 Tutorial](#)
Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift, [Ovadia et al., 2019](#)

Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images, [A. Nguyen et al., 2015](#)

A few more examples of uncertainty: SMTH wrong with the labels

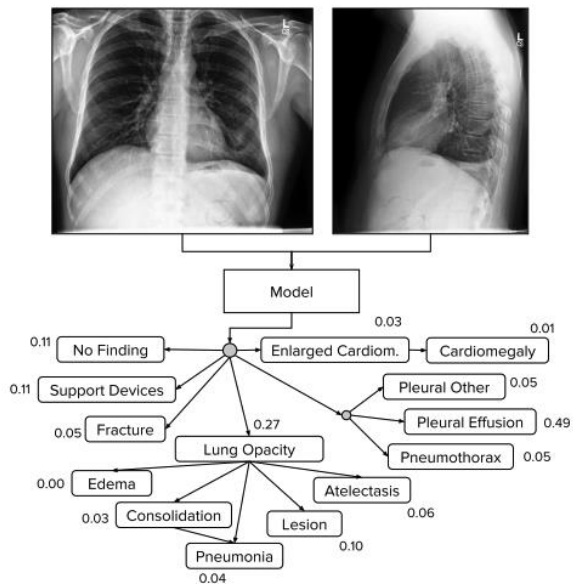


Figure 1: The CheXpert task is to predict the probability of different observations from multi-view chest radiographs.

Observation	Labeler Output
1. <i>unremarkable</i> <u>cardiomediastinal silhouette</u>	No Finding Enlarged Cardiom. Cardiomegaly
2. diffuse <u>reticular pattern</u> , which can be seen with an atypical <u>infection</u> or chronic fibrotic change. <i>no</i> focal <u>consolidation</u> .	Lung Opacity Lung Lesion Edema Consolidation Pneumonia Atelectasis
3. <i>no</i> <u>pleural effusion</u> or <u>pneumothorax</u>	Pneumothorax Pleural Effusion Pleural Other
4. mild degenerative changes in the lumbar spine and old right rib <u>fractures</u> .	Fracture Support Devices

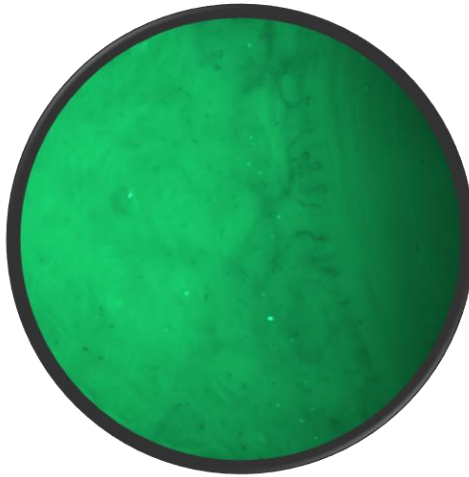
Figure 2: Output of the labeler when run on a report sampled from our dataset. In this case, the labeler correctly extracts all of the mentions in the report (underline) and classifies the uncertainties (bolded) and negations (italicized).

A few more examples of uncertainty: SMTH wrong with the inputs

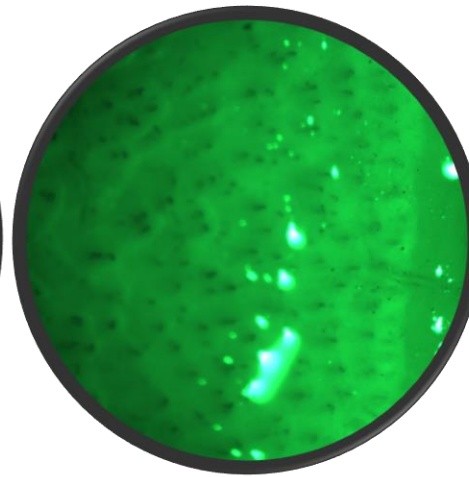
Normal



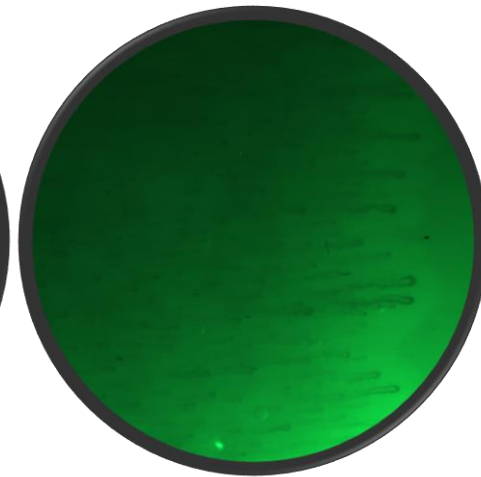
**Low capillary
density**



**Low capillary
quality**

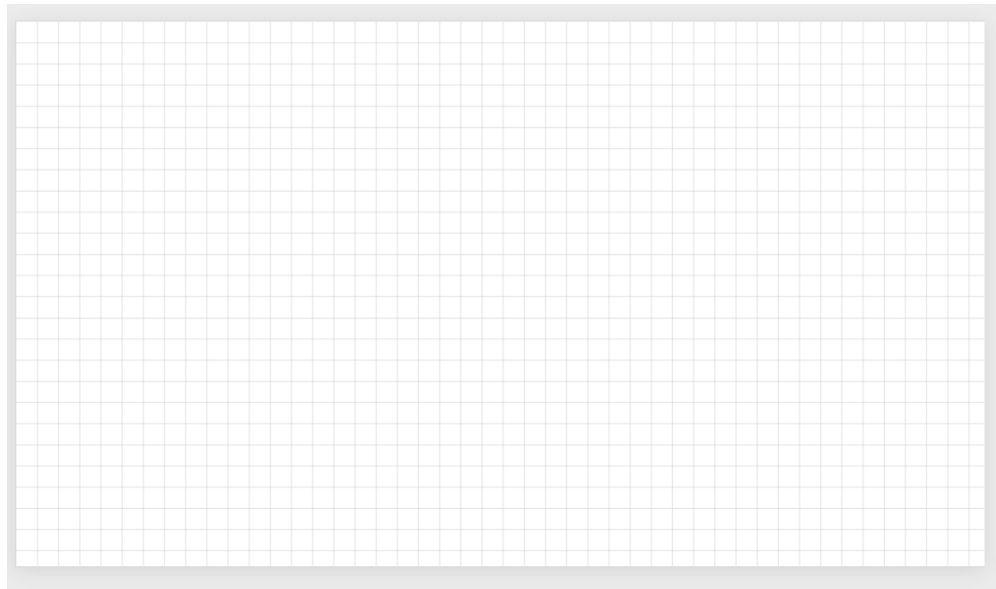
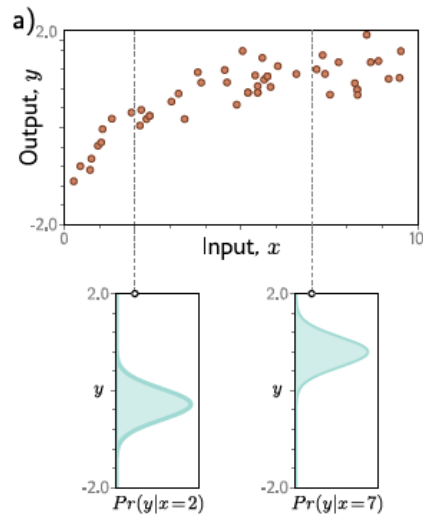


**Low image
quality**



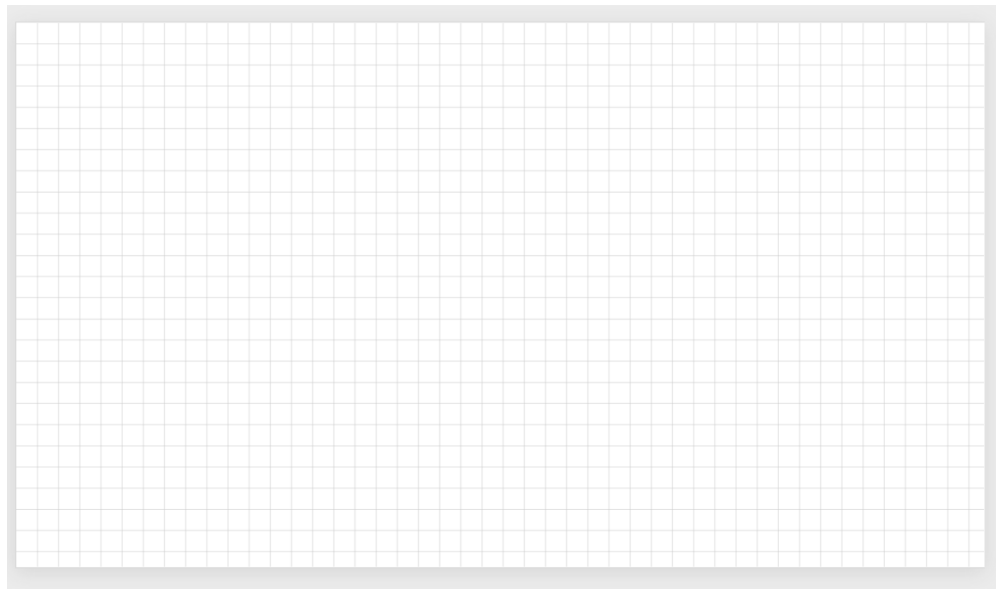
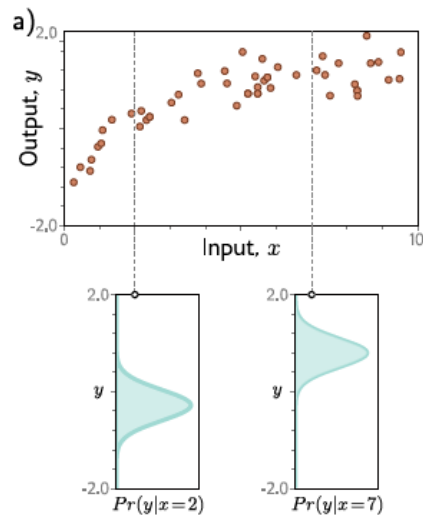
Maximum likelihood

Consider a model $f = [x^{(i)}, \phi]$ parameterized with weights ϕ that maps each i -th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$. We now shift perspective and consider the model as computing a conditional probability distribution $p(x|y)$ or $p(X|Y)$. The loss encourages each training output to have a high probability under the distribution $p(x^{(i)}|y^{(i)})$.



Maximum likelihood \rightarrow minimum negative log-likelihood criterion

Two assumptions should be met. First, we assume that the data are *identically* distributed. Second, we assume that the conditional distributions $p(x^{(i)}|y^{(i)})$ are *independent*. In other words, the data are independent and identically distributed (i.i.d.).



Algorithm for constructing loss functions

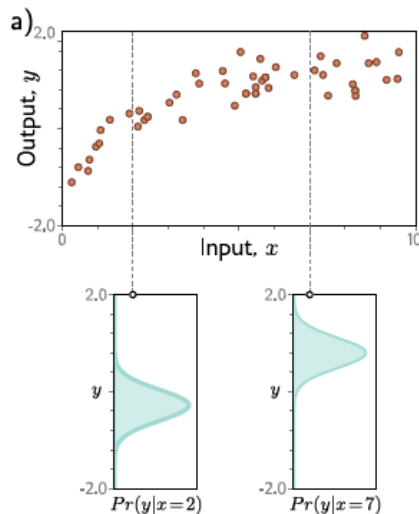
The recipe for constructing loss functions for training data $\{\mathbf{x}_i, \mathbf{y}_i\}$ using the maximum likelihood approach is hence:

1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ defined over the domain of the predictions \mathbf{y} with distribution parameters $\boldsymbol{\theta}$.
2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters, so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.
3. To train the model, find the network parameters $\hat{\boldsymbol{\phi}}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} [L[\boldsymbol{\phi}]] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[- \sum_{i=1}^I \log [Pr(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])] \right]. \quad (5.6)$$

4. To perform inference for a new test example \mathbf{x} , return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\boldsymbol{\phi}}])$ or the value where this distribution is maximized.

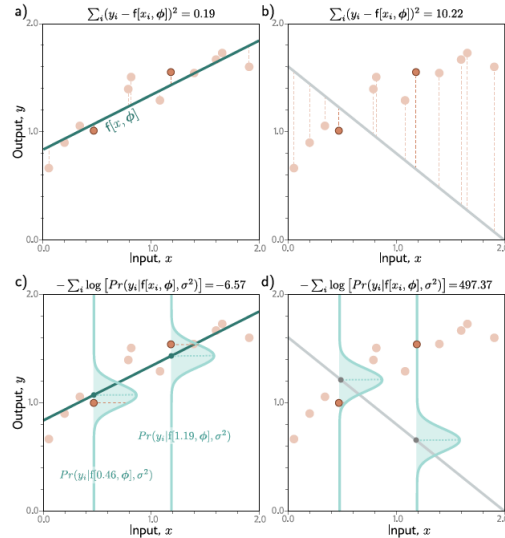
We devote most of the rest of this chapter to constructing loss functions for common prediction types using this recipe.



Example 1: Regression (univariate, homoscedastic) and MSE loss

The key idea is to use the *normal* distribution:

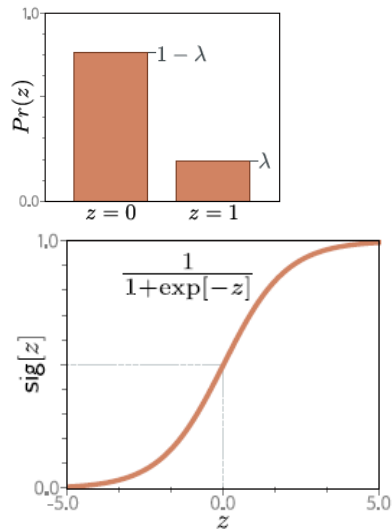
$$p(x) = N(x|\mu, \sigma^2) \\ = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right).$$



Example 2: Logistic regression and binary cross-entropy loss (BCE)

The key idea is to use *Bernoulli* distribution:

$$p(x) = B(x|\lambda) = (1 - \lambda)^{1-x} \lambda^x.$$

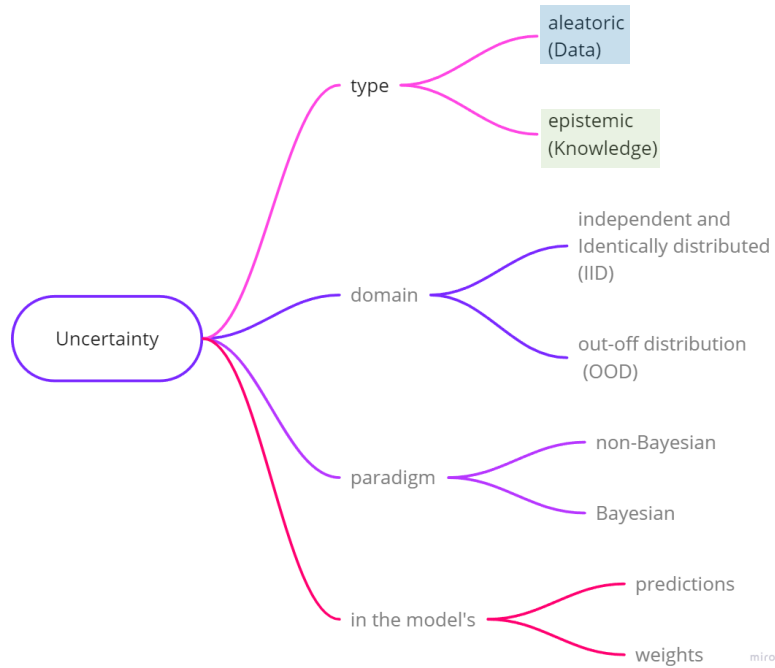


Frequentist **vs** Bayesian frameworks

	Frequentist	Bayesian
Randomness	Objective indefiniteness	Subjective ignorance
Inference	Random and Deterministic	Everything is random
Estimates	Maximal likelihood	Bayes theorem
Applicability	$n \gg \text{size}(\theta)$	$\forall n$

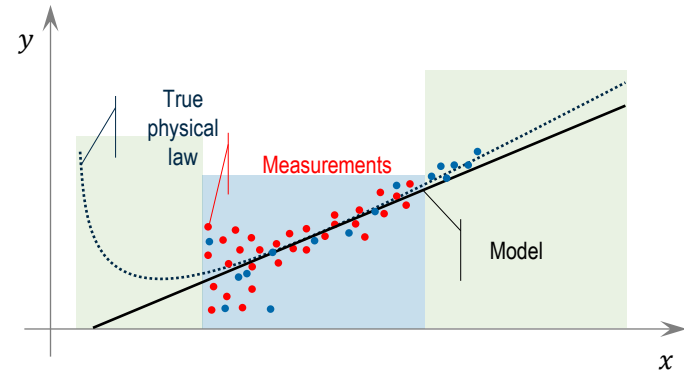
- I. MAXIMUM LIKELIHOOD CRITERION (MLC)
- II. UNCERTAINTY ESTIMATION USING MLC
- III. UNCERTAINTY ESTIMATION IN GENERAL

Terms and formalization



Would adding the **blue** data points reduce the uncertainty?

- Nope, if the uncertainty is aleatoric (U is irreducible).
- Yeah, if the uncertainty is epistemic (U is reducible).



Aleatoric and Epistemic uncertainty intuition

Agenda

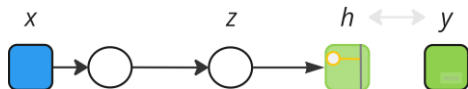
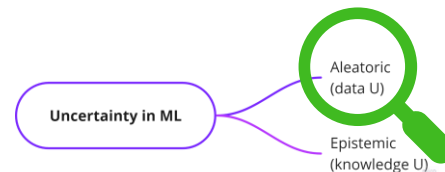
- I. MAXIMUM LIKELIHOOD CRITERION (MLC)
- II. UNCERTAINTY ESTIMATION USING MLC
- III. UNCERTAINTY ESTIMATION IN GENERAL

All models are wrong, but **some models that know when they are wrong**, are useful
/George Box + unknown researcher from the Google AI Brain Team/



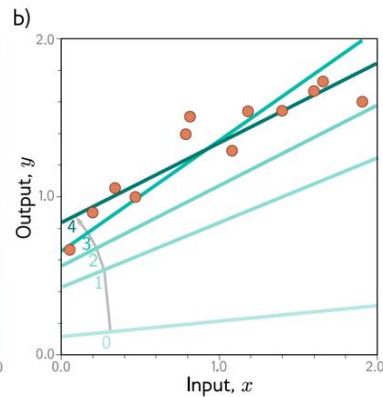
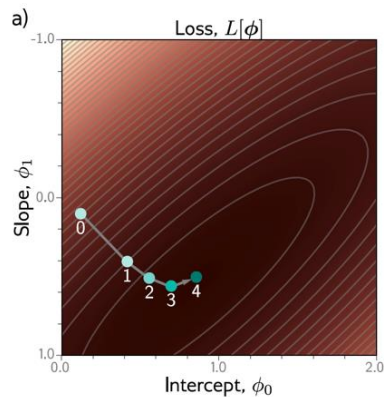
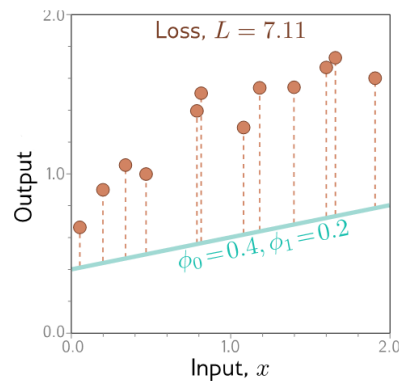
An uncertainty-aware loss intuition

Homoscedastic regression vs heteroscedastic regression

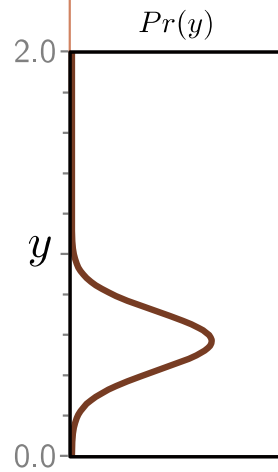
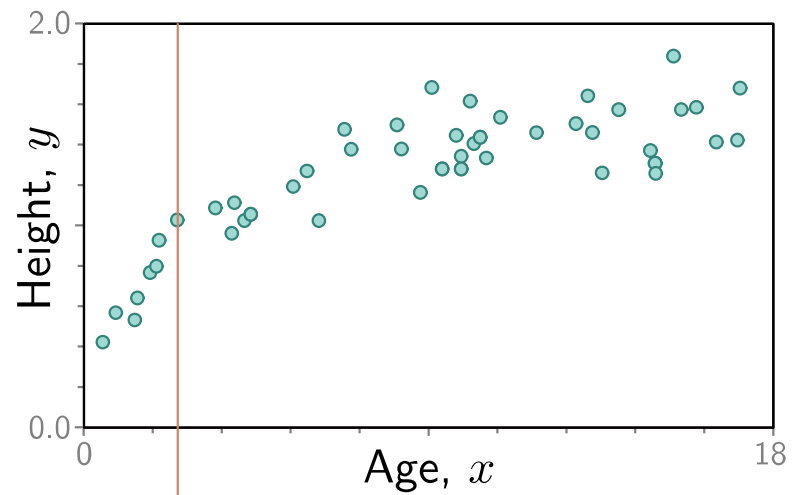


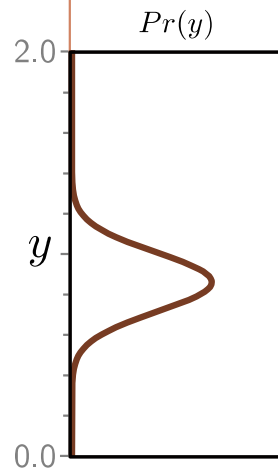
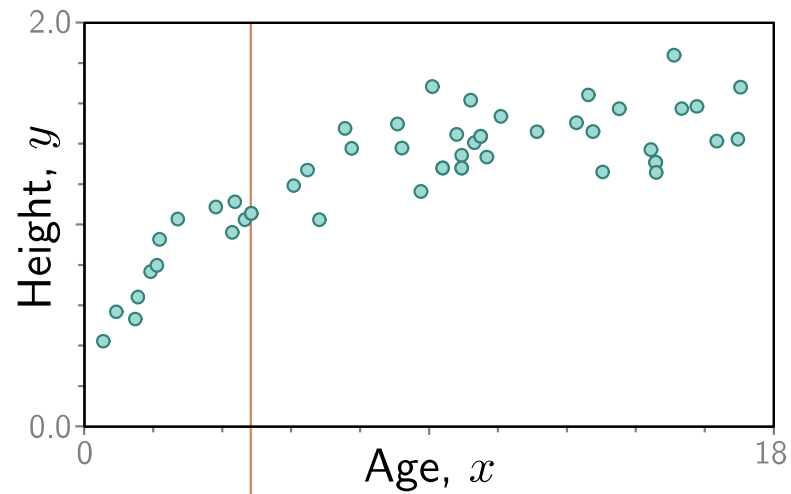
$$L_R = \frac{1}{2m} \sum_{i=1}^m \left(y^{(i)} - h^{(i)} \right)^2$$

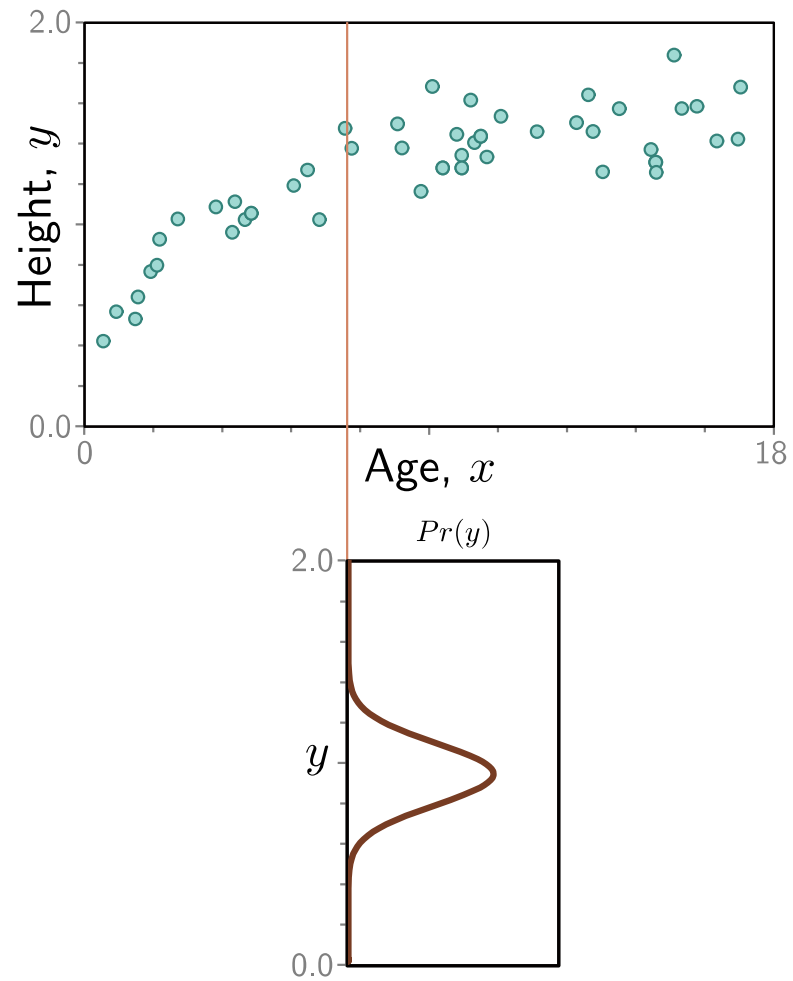
Model predicts output h given input x

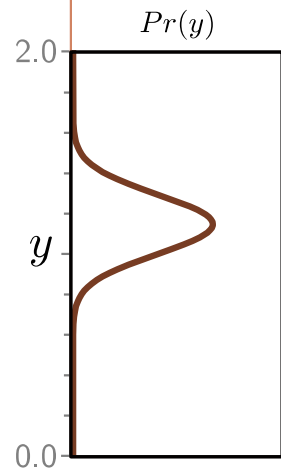
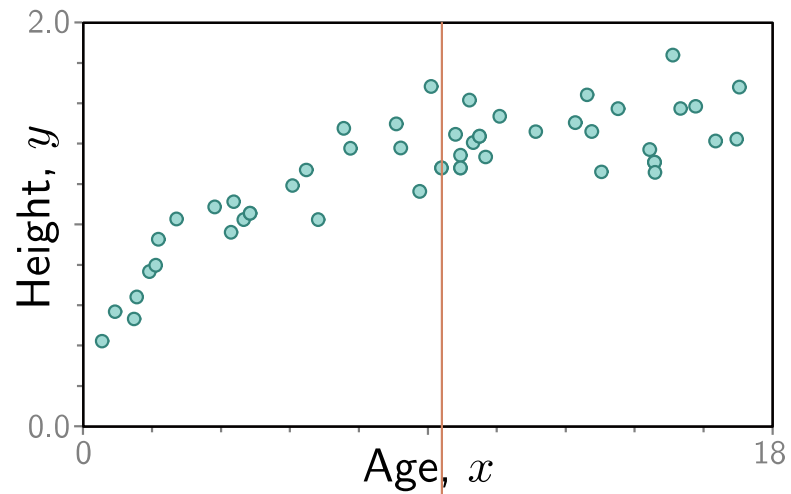


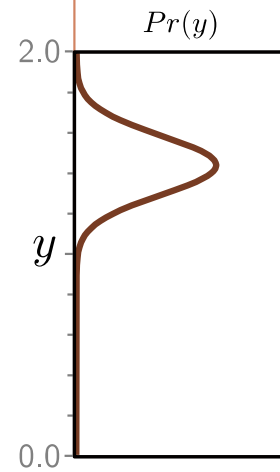
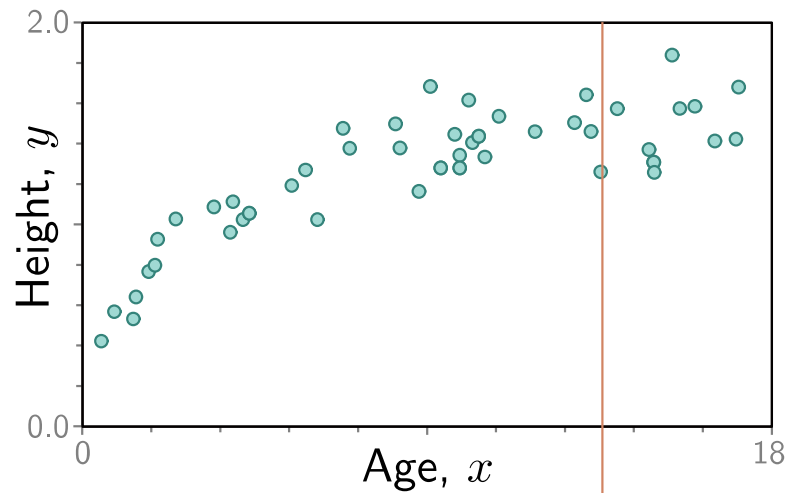
Supervised learning intuition: S. J. Prince. Understanding Deep Learning. MIT Press, 2023. URL <http://udlbook.com>.











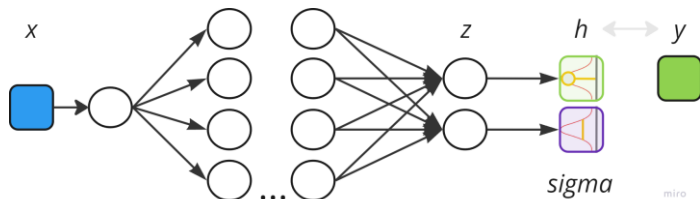
An uncertainty-aware loss intuition

Homoscedastic regression **vs** heteroscedastic regression

Uncertainty in ML

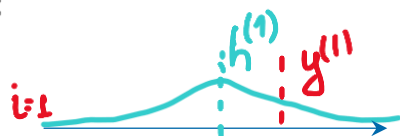
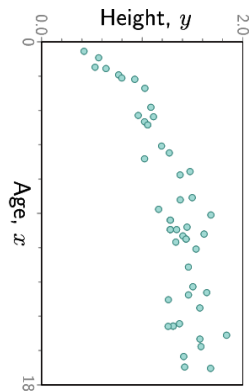
Aleatoric
(data U)

Epistemic
(knowledge U)



Model predicts output h given input x

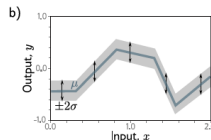
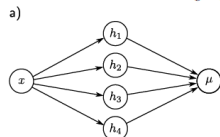
Model predicts a conditional probability distribution $p(y|x)$ over outputs h given input x



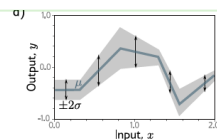
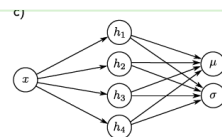
$$p(y^{(i)}|f[x^{(i)}, w]) = \frac{1}{\sqrt{2\pi\sigma_{(i)}^2}} e^{-\frac{(y^{(i)} - h^{(i)})^2}{2\sigma_{(i)}^2}}$$

$$L = -\sum_{i=1}^m \log \left[\frac{1}{\sqrt{2\pi\sigma_{(i)}^2}} e^{-\frac{(y^{(i)} - h^{(i)})^2}{2\sigma_{(i)}^2}} \right] \rightarrow \min$$

$$L_R = \frac{1}{2m} \sum_{i=1}^m (y^{(i)} - h^{(i)})^2$$



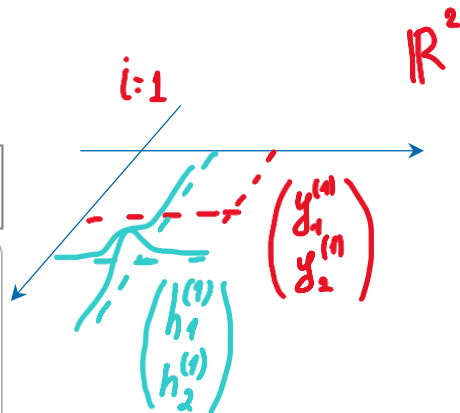
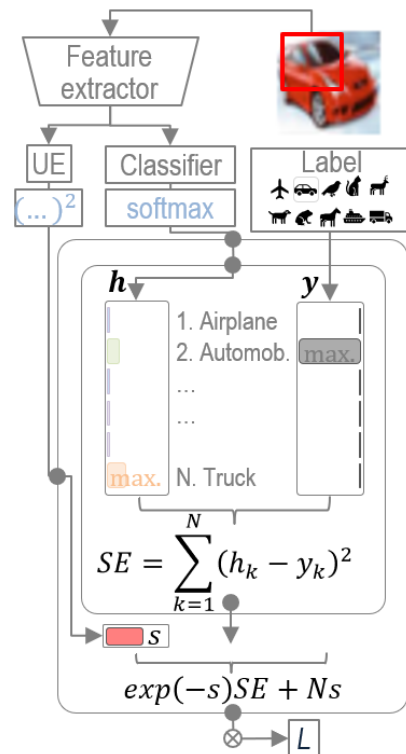
$$L_R = \frac{1}{2m} \sum_{i=1}^m \left(\frac{1}{\sigma_{(i)}^2} (y^{(i)} - h^{(i)})^2 + \log \sigma_{(i)}^2 \right)$$



Supervised learning intuition: S. J. Prince. Understanding Deep Learning. MIT Press, 2023. URL <http://ud1book.com>.

An uncertainty-aware loss intuition

Trick that allows to use a regression loss in classification



$$p(y^{(i)} | f[x^{(i)}, w]) = \frac{1}{\sqrt{(2\pi\sigma_{(i)}^2)^N}} e^{-\frac{\sum_{k=1}^N (y_k^{(i)} - h_k^{(i)})^2}{2\sigma_{(i)}^2}}$$

$$L_C = \frac{1}{2m} \sum_{i=1}^m \left(\frac{1}{\sigma_{(i)}^2} \sum_{k=1}^N (y_k^{(i)} - h_k^{(i)})^2 + N \log \sigma_{(i)}^2 \right)$$

$$L_C = \frac{1}{2m} \sum_{i=1}^m \left(e^{-s^{(i)}} \sum_{k=1}^N (y_k^{(i)} - h_k^{(i)})^2 + N s^{(i)} \right)$$

Uncertainty in ML

Aleatoric
(data U)

Epistemic
(knowledge U)

An uncertainty-aware negative log-likelihood (UANLL) loss intuition.

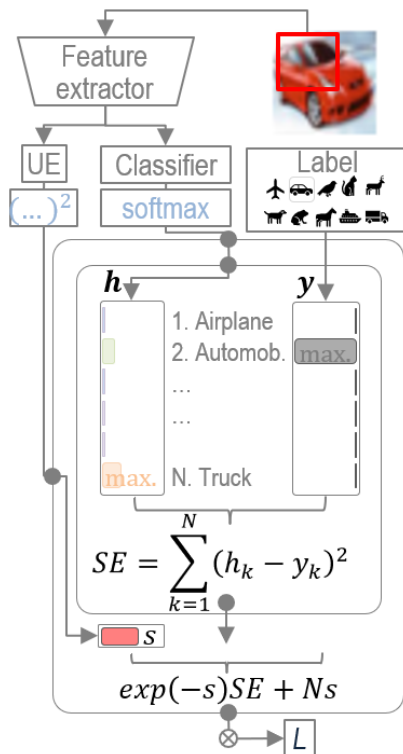
An uncertainty-aware loss intuition

Classification. Use a sample with its different test-time augmentations to calculate multiple predictions, then calculate the final prediction

Uncertainty in ML

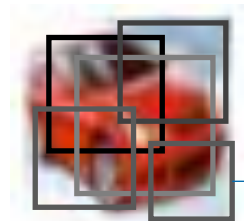
Aleatoric
(data U)

Epistemic
(knowledge U)



$$y^{(i,j)'} = \arg \max_k [h_k^{(i,j)}], \quad ce^{(i,j)} = 1 - \text{sigm}(s^{(i,j)})$$

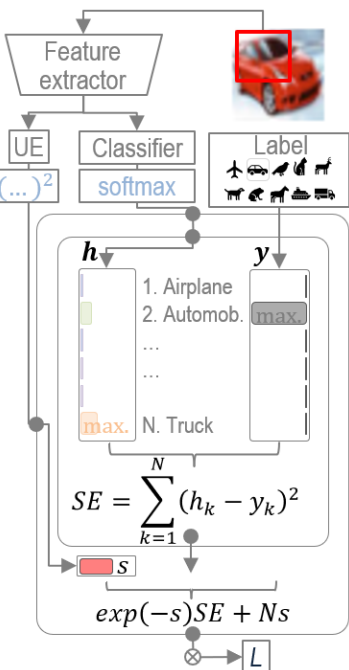
$$co^{(i,j)} = \max_k (h_k^{(i,j)})$$



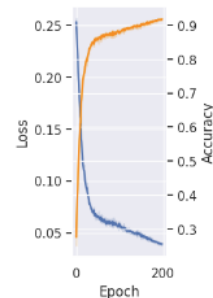
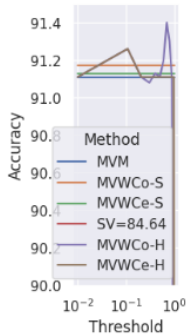
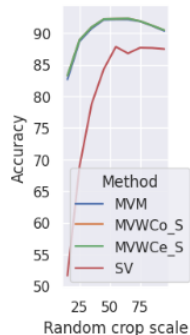
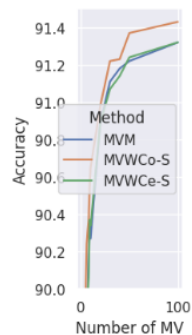
$$z_{k=y^{(i,j)'}}^{(i)} = \sum_j g^{(i,j)}, \quad y^{(i)'} = \max_k (z_k^{(i)})$$

Asymmetric noise in labels. Training of the models

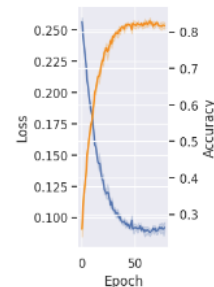
with 0%, 20%, and 40% of replaced labels: airplane ↔ bird, automobile ↔ truck, cat ↔ dog, deer ↔ horse



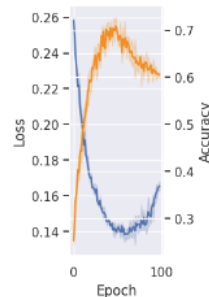
Uncertainty-aware negative log-likelihood (UANLL) loss intuition.



(a) 0%

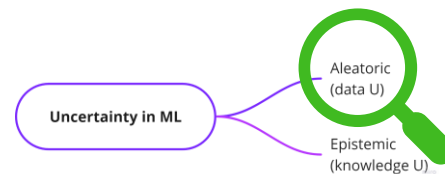


(b) 20%

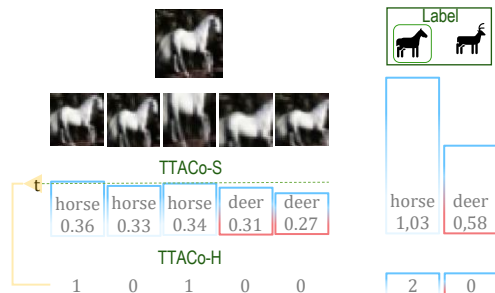
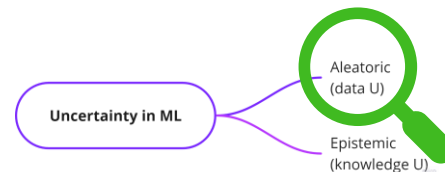


(c) 40%

Figure 3: Validation loss and accuracy of the models based on the proposed loss (7) trained with clean labels (a), and labels with asymmetric noise of 20% (b), and 40% (see Table I).



Asymmetric noise in labels. Test-time augmentation



Neptune run: Play-934

Asymptotic noise in labels of 40%:

airplane \leftrightarrow bird, automobile \leftrightarrow truck, cat \leftrightarrow dog, deer \leftrightarrow horse

Loss: UANLL

Test sample: $i = 14$

True class: 7 (horse)

test-time augmentation (TTA) times: 5

Pred. class = [7, 7, 7, 4, 4] // horse and deer

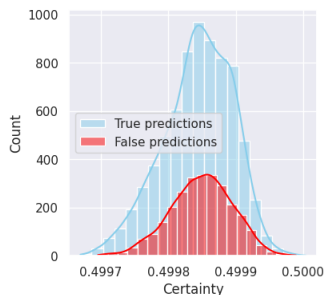
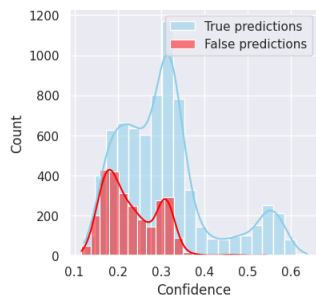
Threshold: $t = 0,34$

co_s = [0.3575, 0.3309, 0.3409, 0.3105, 0.2684] // confidences

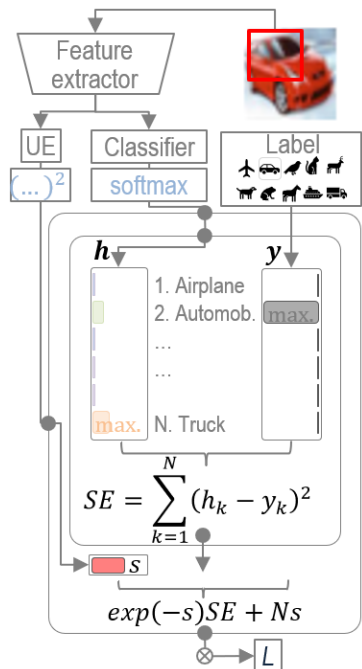
co_h = [1, 0, 1, 0, 0]

ce_s = [0.4999, 0.4999, 0.4998, 0.4999, 0.4999] // certainties

ce_h = [1, 1, 1, 1, 1]



Asymmetric noise in labels. Comparison and ablation study



Uncertainty-aware negative log-likelihood (UANLL) loss intuition.

No	Method	Arch.	#Param. (train.par.)	LS	Accuracy, % / (ECE $\times 10^3$)		
					0	20	40
1.	Baseline CE loss	9-1CNN	4.4 M (all)	0.0	87.19 \pm 1.03/ (33.74 \pm 10.71)	80.06 \pm 0.54/ (99.22 \pm 11.69)	65.15 \pm 1.99/ (61.12 \pm 34.97)
2.	+ MVM	9-1CNN	4.4 M (all)	0.4	92.28 \pm 0.65	86.64 \pm 0.54	71.32 \pm 1.91
	+ MVWCo-S				92.34 \pm 0.66	86.69 \pm 0.56	71.23 \pm 1.91
	+ MVWCe-S				89.55 \pm 0.10/ (356.6 \pm 1.2)	79.71 \pm 0.19/ (377.5 \pm 7)	68.59 \pm 1.42/ (333.5 \pm 17.3)
	+ MVWCo-H				93.54 \pm 0.08	86.26 \pm 0.25	75.08 \pm 1.25
	+ MVWCe-H				93.61 \pm 0.12	86.32 \pm 0.31	75.15 \pm 1.24
3.	CNLCU-S [29]	9-1CNN	4.4 M (all)	—	—	85.06 \pm 0.17/—	75.34 \pm 0.32/—
4.	Co-teaching [7, 29]	9-1CNN	4.4 M (all)	—	—	86.87 \pm 0.24/—	73.43 \pm 0.62/—
5.	S2E [29, 30]	9-1CNN	4.4 M (all)	—	—	86.03 \pm 0.56/—	75.08 \pm 1.25/—
6.	LVCN[3]	9-1CNN	4.4 M (all)	—	89.2/—	—	—
7.	Deep ensembles [13]	ResNet18	—	—	95.1/(8)	—	—
8.	GMM [18]	ResNet18	—	—	93.5 \pm 0.1/—	—	—
9.	LN [5]	WideResNet-28-2	—	—	90.17 \pm 0.55/—	—	—
10.	GLS [28]	ResNet 34	—	0.4	93.05 \pm 0.04/—	—	—

$$L_C = \frac{1}{2m} \sum_{i=1}^m \left(e^{-s^{(i)}} \sum_{k=1}^N \left(y_k^{(i)} - h_k^{(i)} \right)^2 + Ns^{(i)} \right)$$

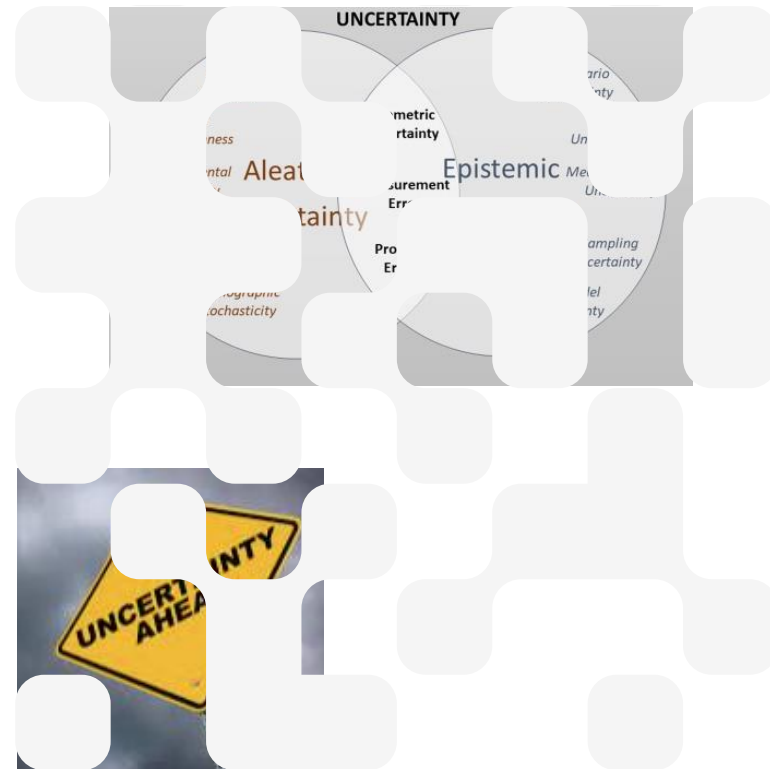
$$L_A = \frac{1}{2m} \sum_{i=1}^m \sum_{k=1}^N \left(y_k^{(i)} - h_k^{(i)} \right)^2$$

Method	Accuracy, % / (ECE $\times 10^3$)		
	0	20	40
UANLL (13)	89.41 \pm 0.18/ (355.9 \pm 1.3)	79.95 \pm 0.30/ (377.8 \pm 6.6)	66.57 \pm 1.15/ (311.3 \pm 19.6)
+ MVM	93.47 \pm 0.09	86.13 \pm 0.50	72.96 \pm 0.77
+ MVWCo-S	93.48 \pm 0.09	86.16 \pm 0.51	72.95 \pm 0.75
+ MVWCe-S	93.45 \pm 0.09	86.11 \pm 0.50	72.88 \pm 0.72
+ MVWCo-H	93.48 \pm 0.15	86.01 \pm 0.78	72.58 \pm 0.98
+ MVWCe-H	93.47 \pm 0.09	86.13 \pm 0.50	72.91 \pm 0.80

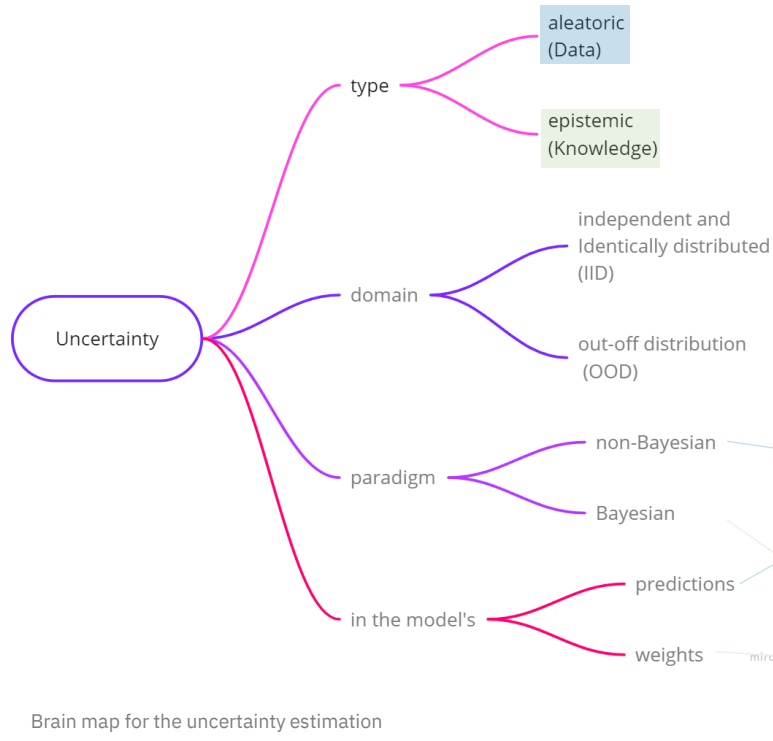
Agenda

- I. MAXIMUM LIKELIHOOD CRITERION (MLC)
- II. UNCERTAINTY ESTIMATION USING MLC
- III. UNCERTAINTY ESTIMATION IN GENERAL

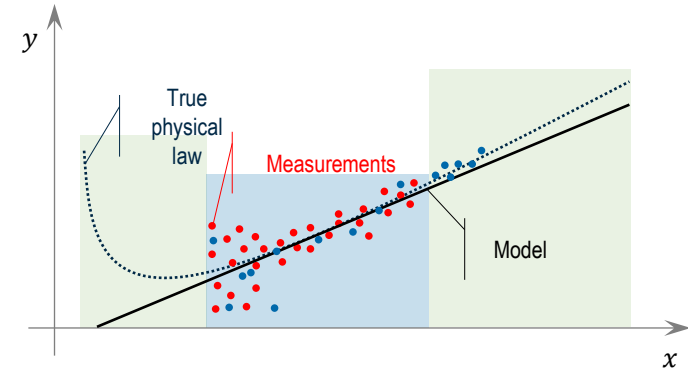
All models are wrong, but **some models that know when they are wrong**, are useful
/George Box + unknown researcher from the Google AI Brain Team/



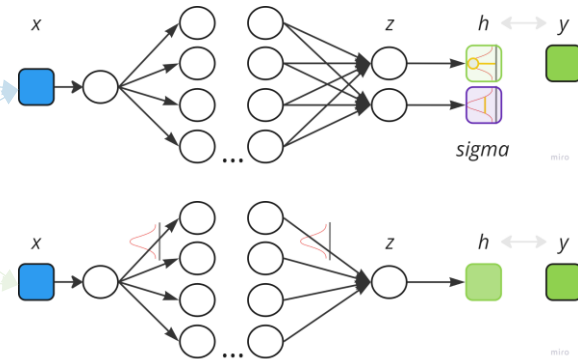
Terms and formalization



Brain map for the uncertainty estimation



Aleatoric and Epistemic uncertainty intuition



Label smoothing

Is a regularization technique used in machine learning, particularly in classification tasks, to improve the performance and generalization of models. It addresses the issue of overfitting by making the model less confident about its predictions.

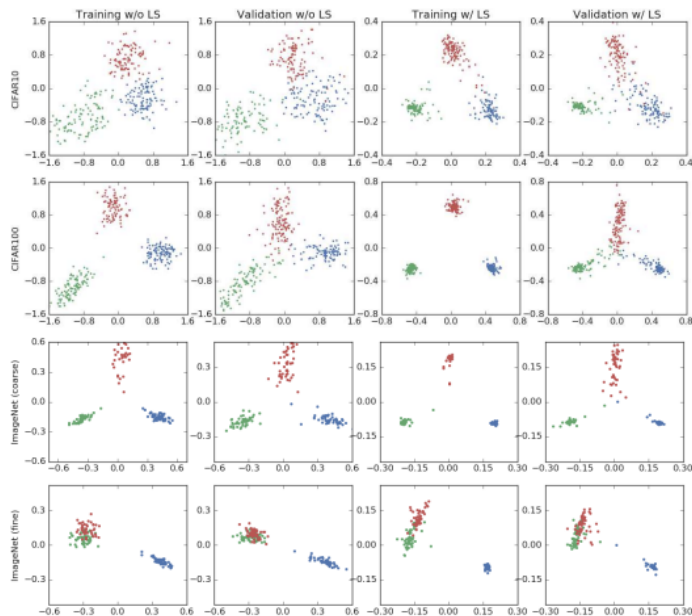


Figure 1: Visualization of penultimate layer's activations of: AlexNet/CIFAR-10 (first row), CIFAR-100/ResNet-56 (second row) and ImageNet/Inception-v4 with three semantically different classes (third row) and two semantically similar classes plus a third one (fourth row).

Traditional One-Hot Encoding

In a typical classification problem, the target labels are represented as one-hot vectors. For example, if you have a 3-class problem, the labels might be:

- Class 1: [1, 0, 0]
- Class 2: [0, 1, 0]
- Class 3: [0, 0, 1]

Label Smoothing:

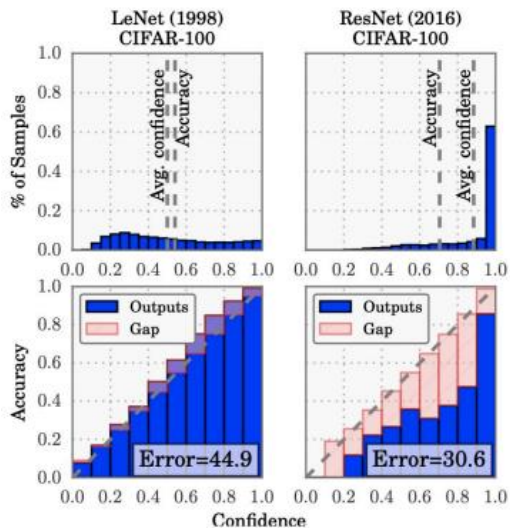
Instead of using the exact one-hot vectors, label smoothing introduces a small amount of noise to the labels. This is done by replacing the hard 0s and 1s with values slightly less than 1 and slightly greater than 0, respectively. For example, with a smoothing factor ϵ (typically a small value like 0.1), the smoothed labels for a 3-class problem might look like:

- Class 1: $[1-\epsilon, \epsilon/2, \epsilon/2]$
- Class 2: $[\epsilon/2, 1-\epsilon, \epsilon/2]$
- Class 3: $[\epsilon/2, \epsilon/2, 1-\epsilon]$

```
1 import tensorflow as tf
2 from tensorflow.keras.losses import CategoricalCrossentropy
3
4 # Define the smoothing factor
5 epsilon = 0.1
6
7 # Create a label smoothing loss function
8 loss_fn = CategoricalCrossentropy(from_logits=True, label_smoothing=epsilon)
9
10 # Example usage in model training
11 model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy'])
```


Temperature scaling

Is a post-hoc calibration technique used to improve the calibration of a trained neural network model. It is particularly useful for models that produce overconfident or underconfident predictions. The idea behind temperature scaling is to adjust the logits (raw output scores before applying the softmax function) by a scalar temperature parameter, which helps in smoothing or sharpening the predicted probabilities.



How Temperature Scaling Works:

1. Logits and Softmax:

- In a typical classification model, the final layer outputs logits, which are then passed through a softmax function to produce class probabilities.
- The softmax function is defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

where z_i are the logits.

2. Temperature Scaling:

- Temperature scaling introduces a temperature parameter T (where $T > 0$) to adjust the logits before applying the softmax function:

$$\text{softmax}(z_i/T) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

- When $T > 1$, the probabilities become smoother (less confident), and when $T < 1$, the probabilities become sharper (more confident).

3. Optimizing the Temperature Parameter:

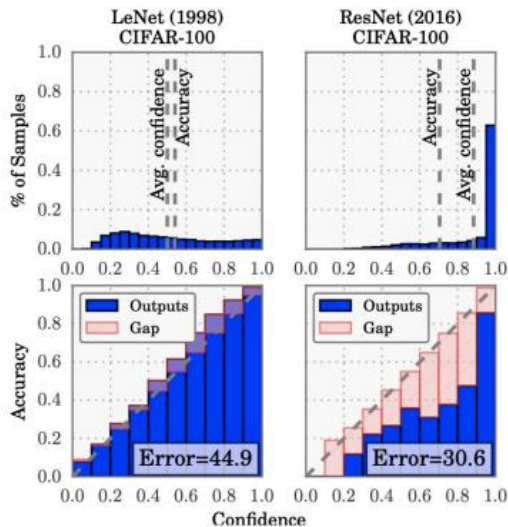
- The temperature parameter T is typically optimized to minimize a calibration error metric, such as the Expected Calibration Error (ECE), on a validation set.
- This can be done using gradient-based optimization or by searching over a range of values.

Expected calibration error (ECE)

Is a metric used to evaluate the calibration of a classification model. Calibration refers to how well the predicted probabilities of a model match the actual likelihood of the outcomes. In other words, a well-calibrated model should produce probabilities that reflect the true likelihood of the events they predicts.

$$\text{ECE} = \sum_{i=1}^B \frac{n_i}{N} |\text{conf}_i - \text{acc}_i|$$

Suppose we have a model that makes 1000 predictions and we divide the predictions into 10 bins. Here's a simplified example:

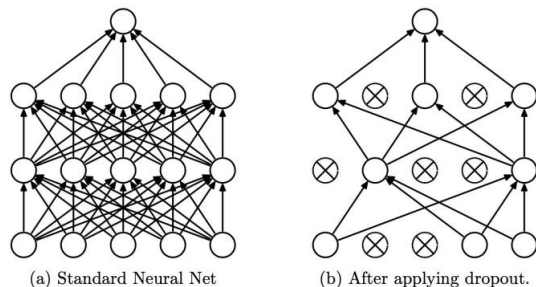


Bin	# Predictions	Avg. Pred. Prob.	Accuracy	Difference
1	100	0.1	0.08	0.02
2	150	0.2	0.18	0.02
3	200	0.3	0.27	0.03
...
10	50	0.9	0.92	0.02

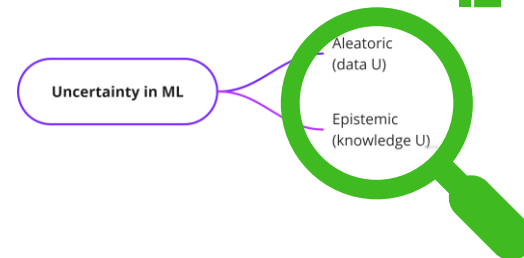
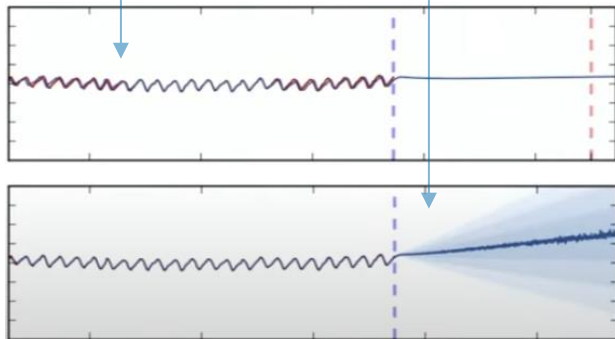
$$\text{ECE} = \frac{100}{1000} \times 0.02 + \frac{150}{1000} \times 0.02 + \frac{200}{1000} \times 0.03 + \dots + \frac{50}{1000} \times 0.02$$

Dropout training method

is identical to approximate variational inference in Bayesian modeling (proved by Yarin Gal in 2016)



[Dropout: a simple way to prevent neural networks from overfitting](#)



```
1 # Given x:
2 # - drop units at test time
3 # - repeat 10 times
4 # - and look at mean and sample variance
5
6 y = []
7 for _ in xrange(10):
8     y.append(model.output(x, dropout=True))
9 y_mean = numpy.mean(y)
10 y_var = numpy.var(y)
```

[Yarin Gal - Uncertainty in Deep Learning | MLSS Kraków 2023](#)

Ensembling and test-time augmentation

Uncertainty in ML

Aleatoric
(data U)
Epistemic
(knowledge U)

```
1 num_ensembles = 5
2 for i in xrange(num_ensembles):
3     model = create_model(...)
4     model.fit(...)
5
6 raw_predictions = [models[i].predict(x)
7                     for i in range(num_ensembles)]
8 y_mean = numpy.mean(raw_predictions)
9 y_var = numpy.var(raw_predictions)
```

What if we train the same network multiple times (an **ensemble** of networks) and compare outputs?

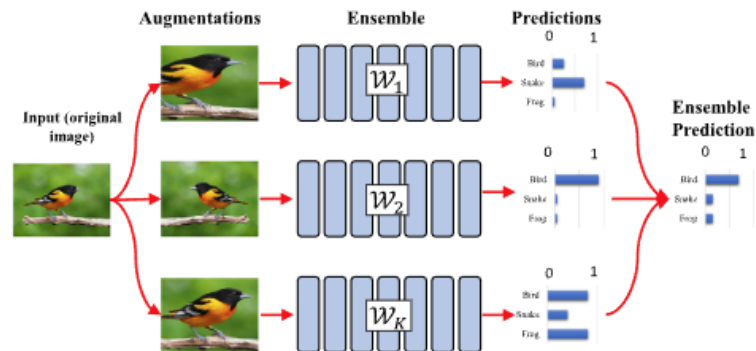
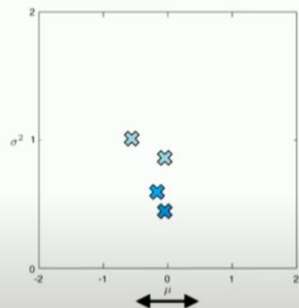
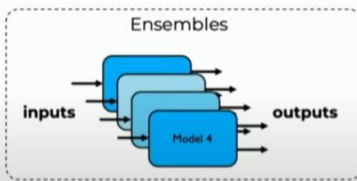


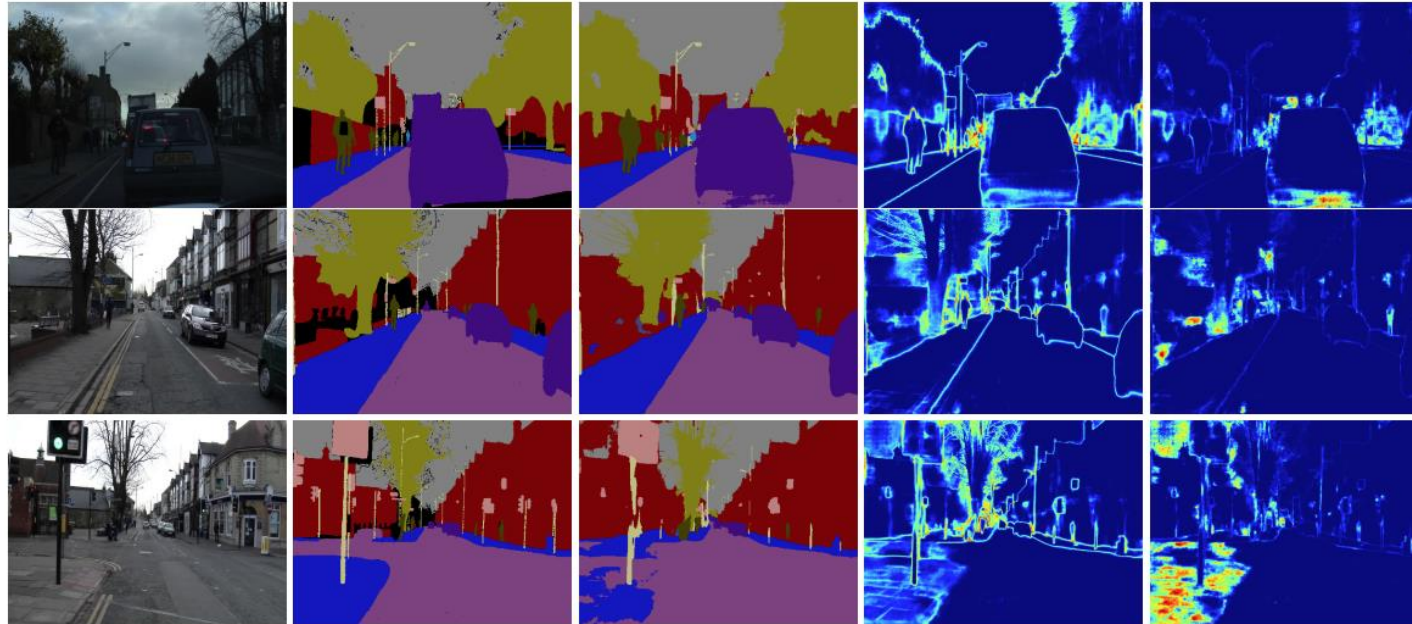
Fig. 15. A schematic view of TTA for ensembling techniques.

Source: Reproduced based on [179].

[MIT Introduction to Deep Learning 6.S191: Lecture 5 Robust and Trustworthy Deep Learning](#)
Lecturer: Sadhana Lolla

[A review of uncertainty quantification in deep learning: Techniques, applications and challenges](#)

UE in segmentation



(a) Input Image

(b) Ground Truth

(c) Semantic Segmentation

(d) Aleatoric Uncertainty

(e) Epistemic Uncertainty

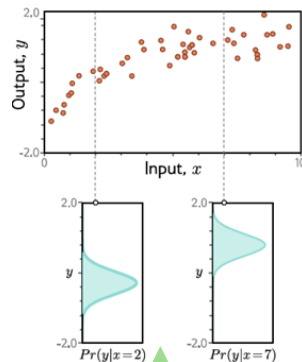
Uncertainty in ML

Aleatoric
(data U)

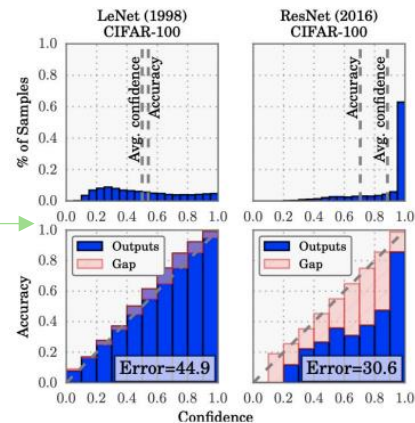
Epistemic
(knowledge U)

[Alex Kendal and Yarin Gal What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?](#)

Self-study and self-test questions



1. We've checked the *noise in labels* case supposing $p(y|x)$. Can we inverse this idea and suppose that $p(x|y)$ to study the *noise in inputs* case?
2. How does the *cross-entropy loss* for multiclass classification follows the maximum likelihood?
3. Why the *label smoothing* idea works?
4. Why the *expected calibration error* (ECE) intuition is based on the statement that in the ideal world the accuracy of the model matches the confidence of the model?

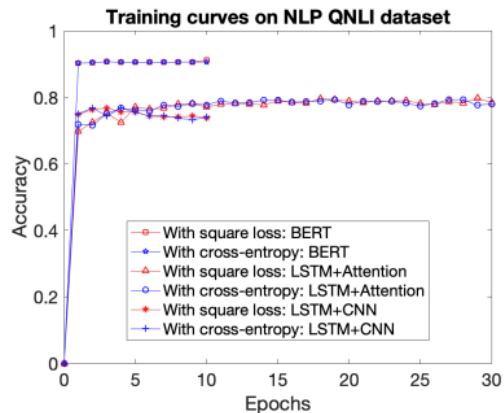


Thank you for your attention!

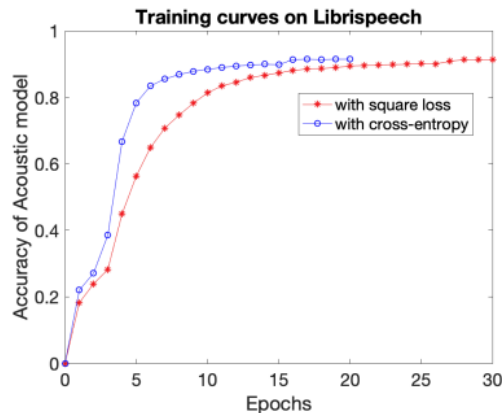
a.kornaev@innopolis.ru, [@avkornaev](https://twitter.com/avkornaev)



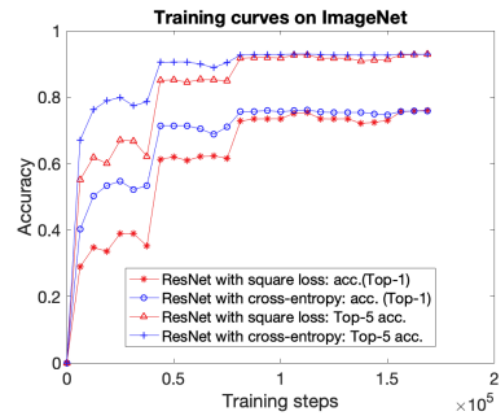
Related work: a square loss classifies well



(a) NLP tasks



(b) ASR tasks



(c) Vision tasks

Related work: ensembling and other methods

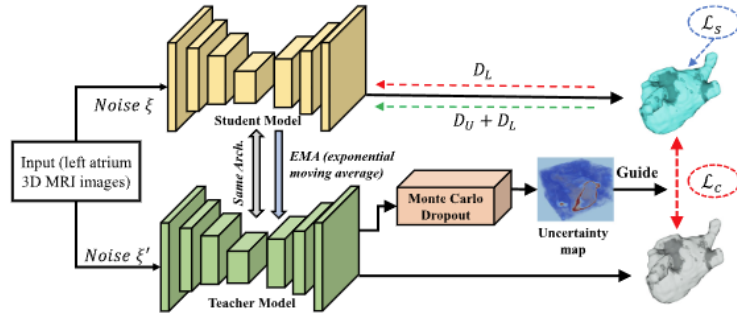
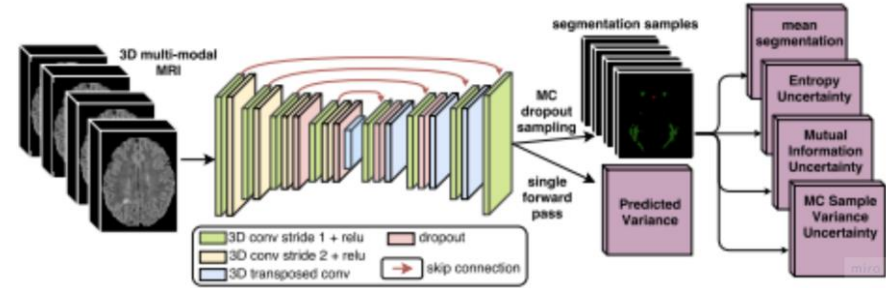


Fig. 5. A general view demonstrating the application of the semi-supervised UA-MT framework to LA segmentation. Source: Reproduced based on [43].



A review of uncertainty quantification in deep learning: Techniques, applications and challenges

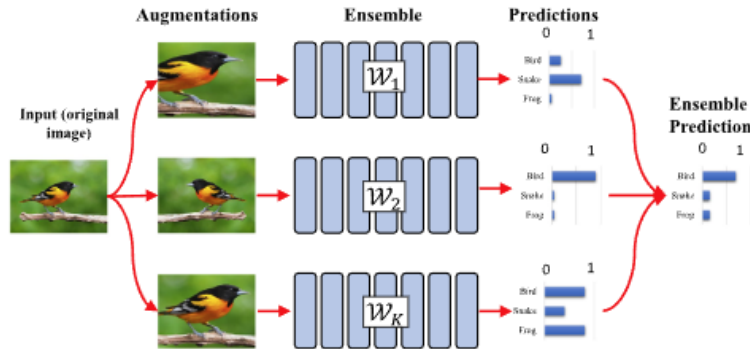


Fig. 15. A schematic view of TTA for ensembling techniques.
Source: Reproduced based on [179].

