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?

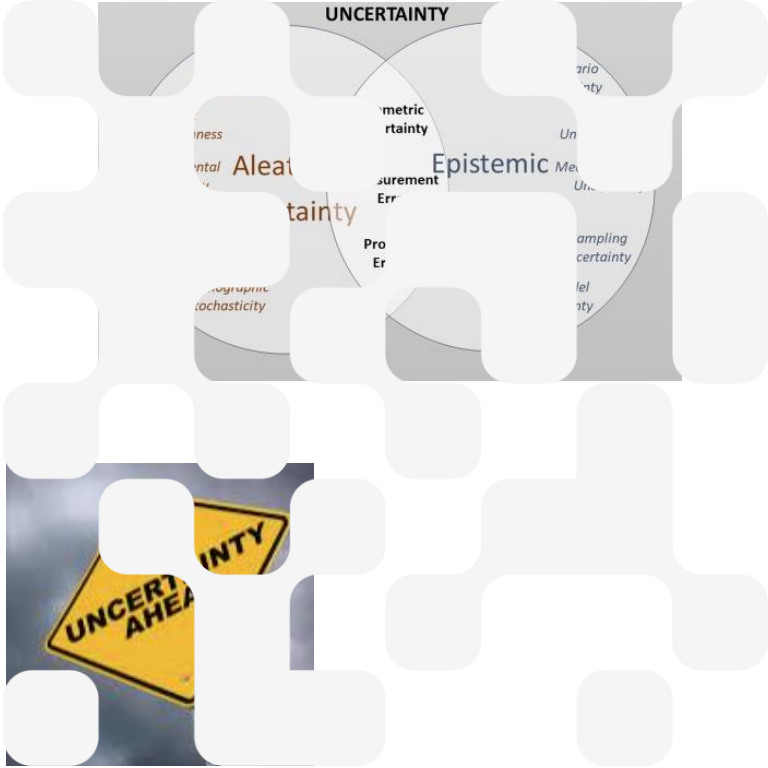
by Alexei Valerievich Kornaev, Dr. habil. in Eng. Sc.,
Researcher at the RC for AI, Assoc. Prof. of the Robotics and CV Master's Program, Innopolis University
Researcher at the RC for AI, National RC for Oncology n.a. NN Blohin
Professor at the Dept. of Mechatronics, Mechanics, and Robotics, Orel State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

Agenda

I. MAXIMUM LIKELIHOOD CRITERION (MLC)

II. UNCERTAINTY ESTIMATION USING MLC

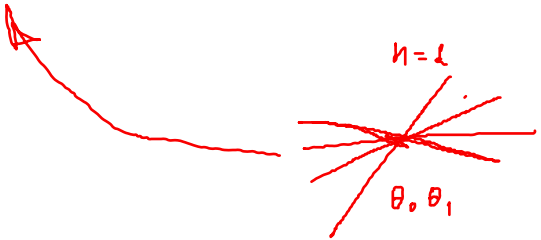
III. UNCERTAINTY ESTIMATION IN GENERAL



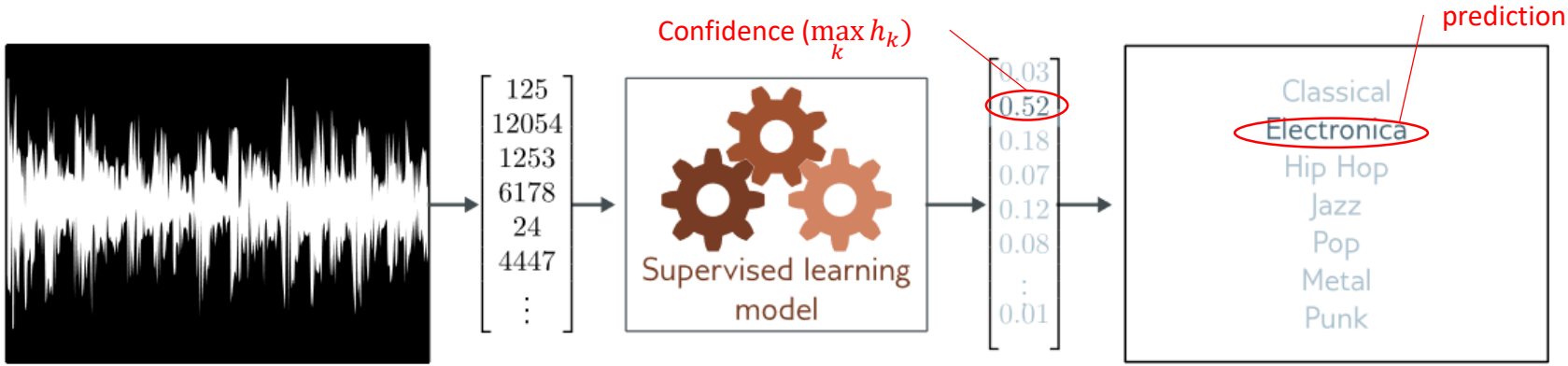
All models are wrong, but ~~some~~ models that know when they are wrong, are useful
/George Box + unknown researcher from the Google AI Brain Team/

Frequentist vs Bayesian frameworks

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

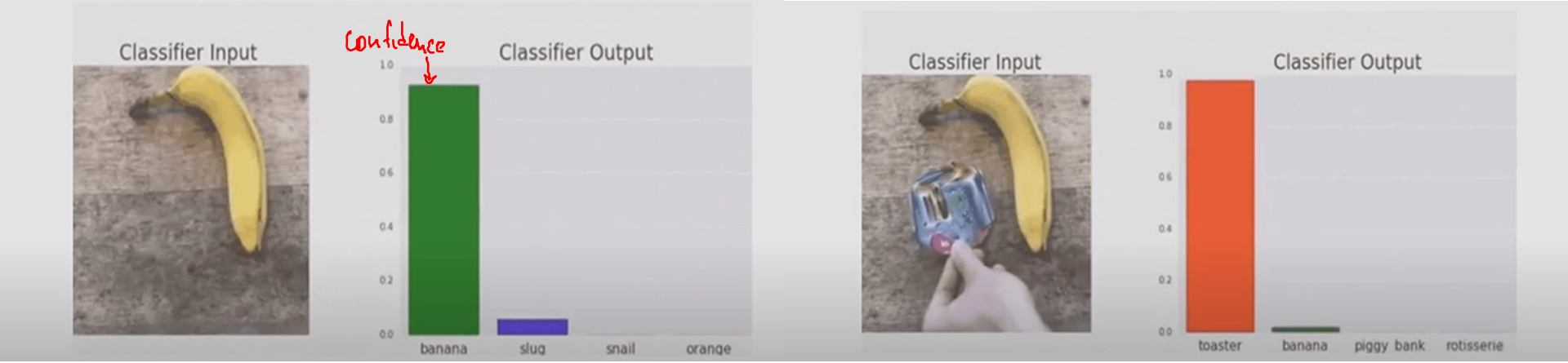


Confidence intuition

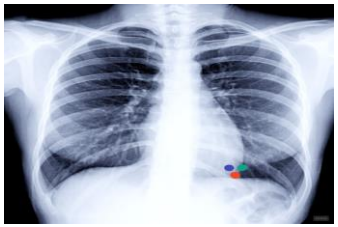


Supervised learning intuition
S. J. Prince. Understanding Deep Learning. MIT Press, 2023. URL http://udlbook.com.

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

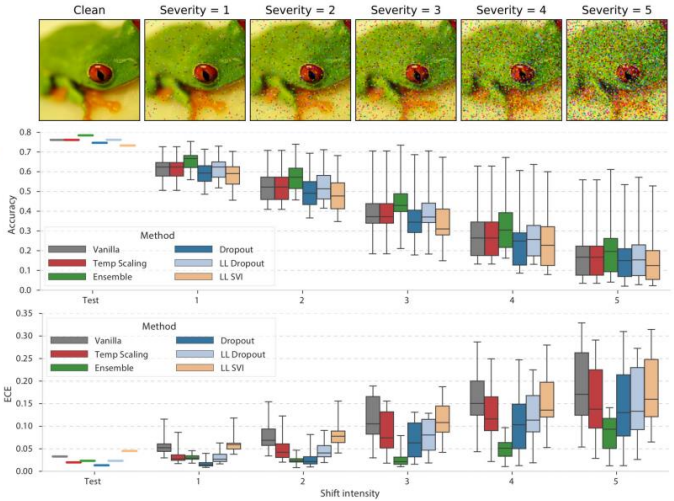


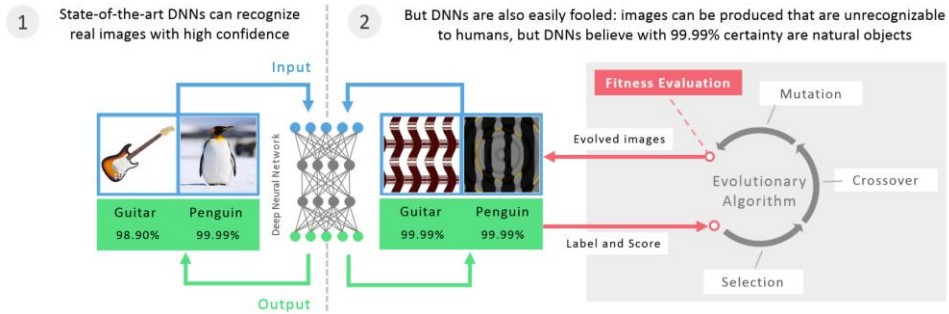
OOD Data



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”





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. Nguen et al., 2015

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

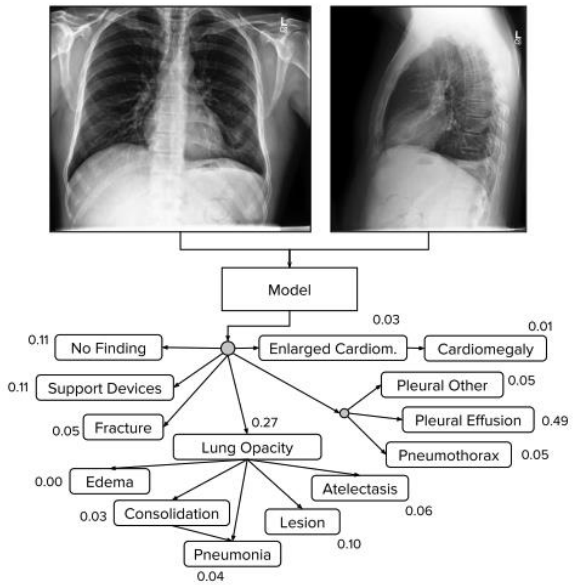


Figure 1: The CheXpert task is to predict the probability of different observations from multi-view chest radiographs.

1. *unremarkable* cardiomediastinal silhouette

2. diffuse reticular pattern, which can be seen with an atypical infection **or** chronic fibrotic change. *no* focal consolidation.

3. *no* pleural effusion or pneumothorax

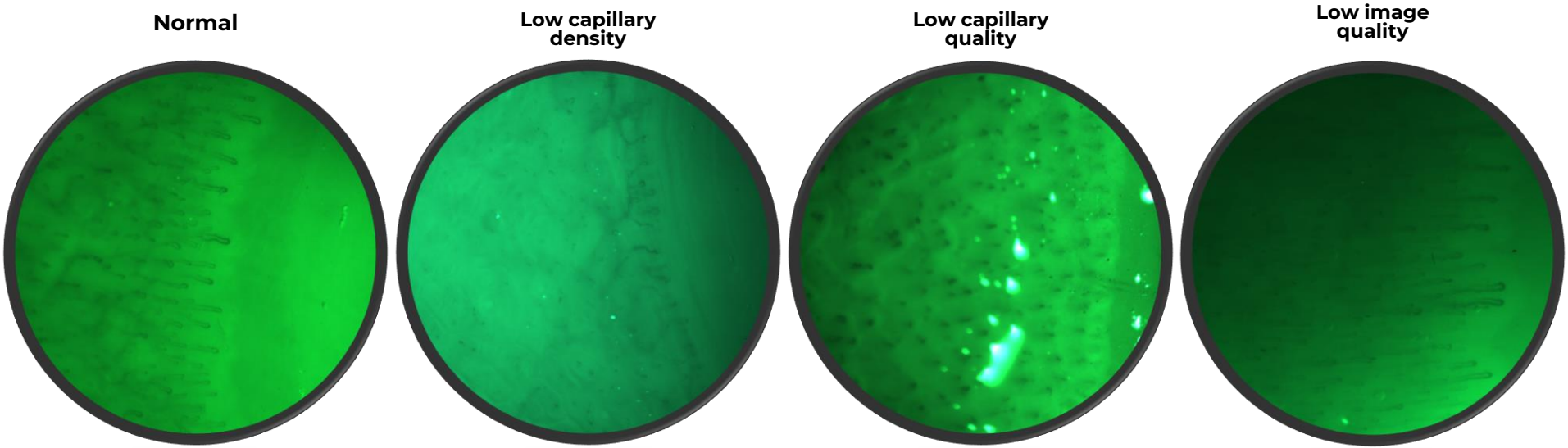
4. mild degenerative changes in the lumbar spine and old right rib fractures.

Observation	Labeler Output
No Finding	
Enlarged Cardiom.	0
Cardiomegaly	
Lung Opacity	1
Lung Lesion	
Edema	
Consolidation	0
Pneumonia	u
Atelectasis	
Pneumothorax	0
Pleural Effusion	0
Pleural Other	
Fracture	1
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).

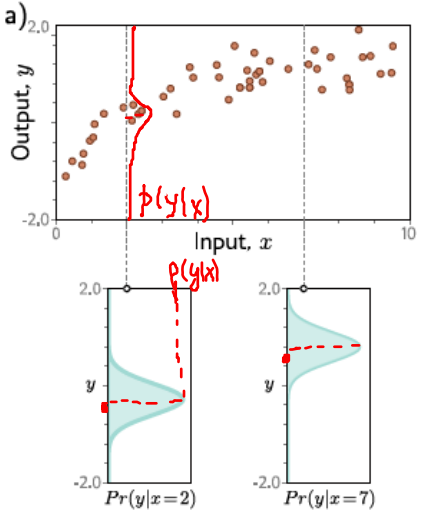
Neural Jeremy Irvin et al. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. 33rd AAAI Conference on Artificial Intelligence, AAAI 2019. doi = {10.48550/arxiv.1901.07031}

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



Maximum likelihood

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ 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 $p(\boldsymbol{y}|\boldsymbol{x})$ or $p(Y|X)$. The loss encourages each training output to have a high probability under the distribution $p\left(y^{(i)}|x^{(i)}\right)$.



$\phi \to \Phi$

$$\Phi_{ML} = \arg\max_{\Phi}\left[p\left(y^1|x^1, y^2|x^2, \ldots y^n|x^n\right)\right]$$

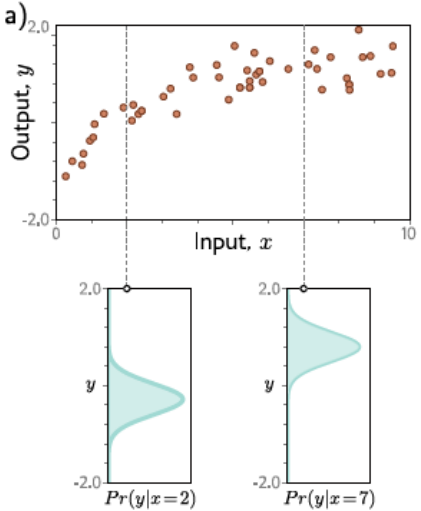
$$x^i \to f[x^i, \Phi]$$

$$\Phi_{ML} = \arg\max_{\Phi}\left[p\left(y^1|f[x^1,\Phi], \ldots y^n|f[x^n,\Phi]\right)\right]$$

S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

Maximum likelihood →minimum negative log-likelihood criterion

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



S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

$p(y|x)$ – likelihood

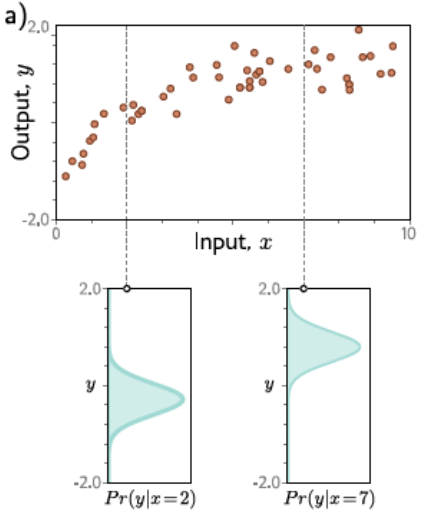
$$p(y^1|x^1, y^2|x^2, \dots y^n|x^n) = \prod_i p(y^i|x^i)$$

$$\Phi_{ML} = \arg\max_{\Phi} \left[\prod_i p(y^i|f[x^i,\Phi])\right]$$ – Maximum likelihood

$$\Phi_{NLL} = -\arg\max_{\Phi} \left[\sum_i \ln p(y^i|f[x^i,\Phi])\right]$$ – negative log-likelihood

$$\Phi_{NLL} \Rightarrow \min$$

Algorithm for constructing loss funtions



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 $\{\mathbf{x}_i, \mathbf{y}_i\}$:

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

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.

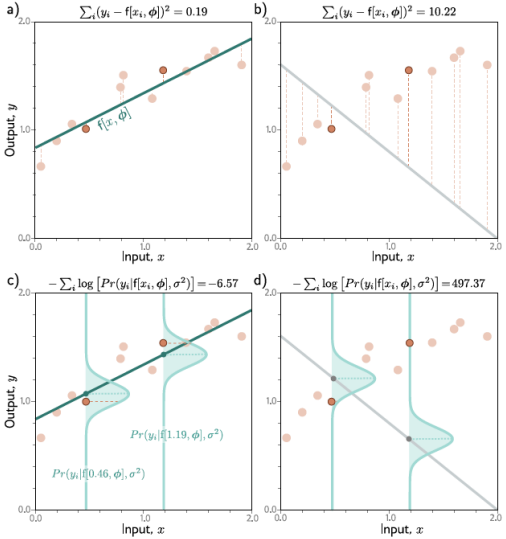
We devote most of the rest of this chapter to constructing loss functions for common prediction types using this recipe.

S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

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

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

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



$$p(y^i|x^i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y^i - h^i)^2}{2\sigma_i^2}\right)$$

$$L = -\sum_i \left[\ln \frac{1}{\sqrt{2\pi\sigma_i^2}} + \left(-\frac{(y^i - h^i)^2}{2\sigma_i^2}\right)\right] \Rightarrow \min$$

Let $\sigma_i^2 = \sigma^2$

$$\boxed{L = \sum_i (y^i - h^i)^2 \Rightarrow \min}$$

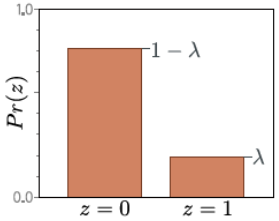
MSE loss

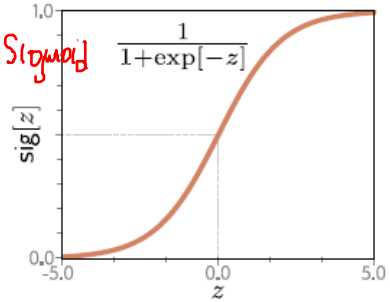
S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

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

The key idea is to use *Bernoulli* distribution:

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





$$p(y^i|x^i) = (1-h^{(i)})^{1-y^{(i)}} h^{(i)y^{(i)}}$$

$$\ln(ab) = \ln a + \ln b$$

$$\ln a^b = b \ln a$$

$$L = -\sum_i \ln\left[(1-h^{(i)})^{1-y^{(i)}} h^{(i)y^{(i)}}\right]$$

$$L = -\sum_i \left[(1-y^i)\ln(1-h^i) + y^i \ln h^i\right] \quad \text{BCE loss}$$

S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

Frequentist vs Bayesian frameworks

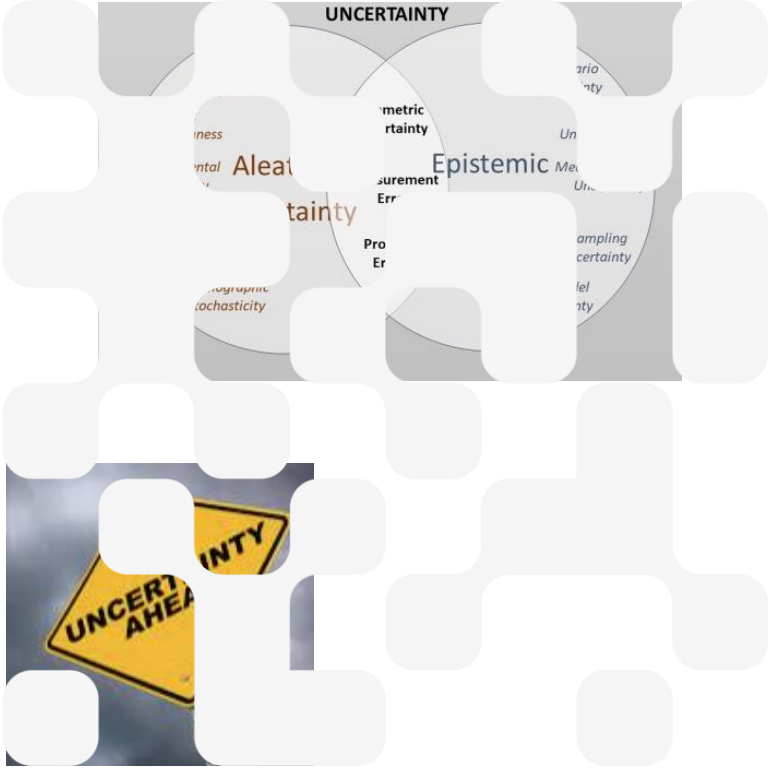
	Frequentist	Bayesian
Randomness	Objective indefiniteness	Subjective ignorance
Inference	Random and Deterministic	Everything is random
Estimates	Maximal likelihood	Bayes theorem
Applicability	$n \gg size(\boldsymbol{\theta})$	$\forall n$

Agenda

I. MAXIMUM LIKELIHOOD CRITERION (MLC)

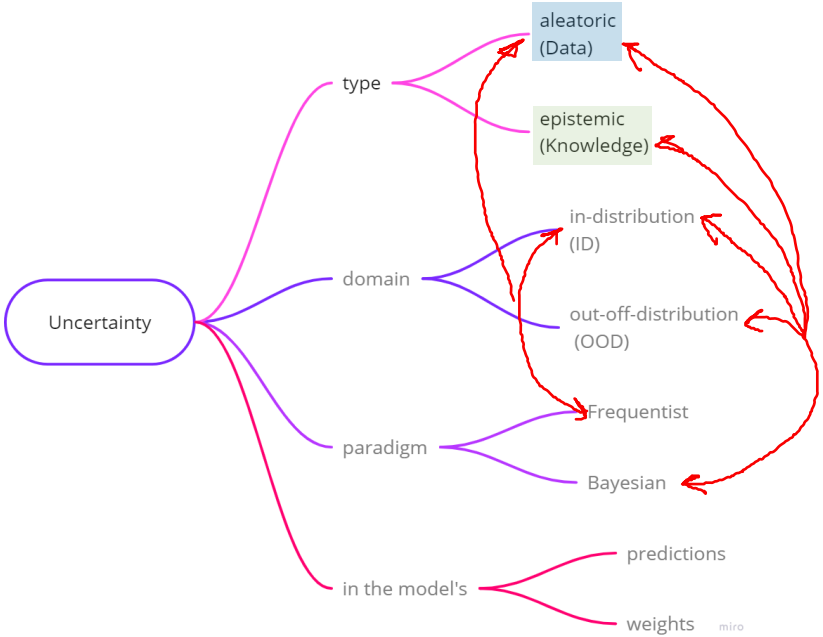
II. UNCERTAINTY ESTIMATION USING MLC

III. UNCERTAINTY ESTIMATION IN GENERAL

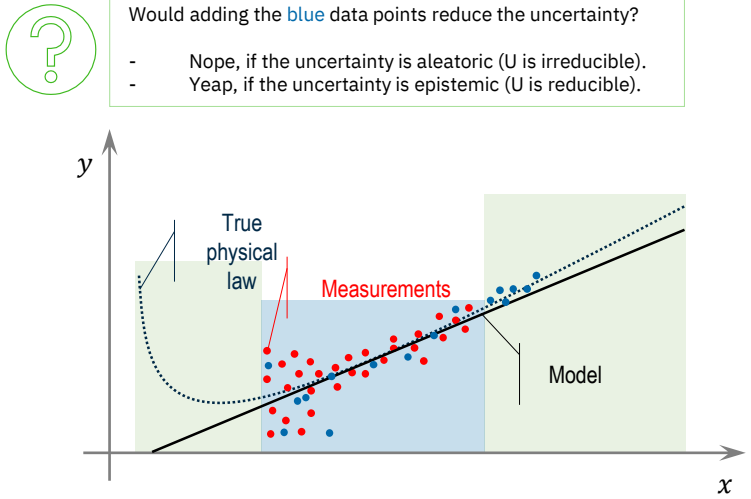


All models are wrong, but ~~some~~ models that know when they are wrong, are useful
/George Box + unknown researcher from the Google AI Brain Team/

Terms and formalization



Brain map for the uncertainty estimation



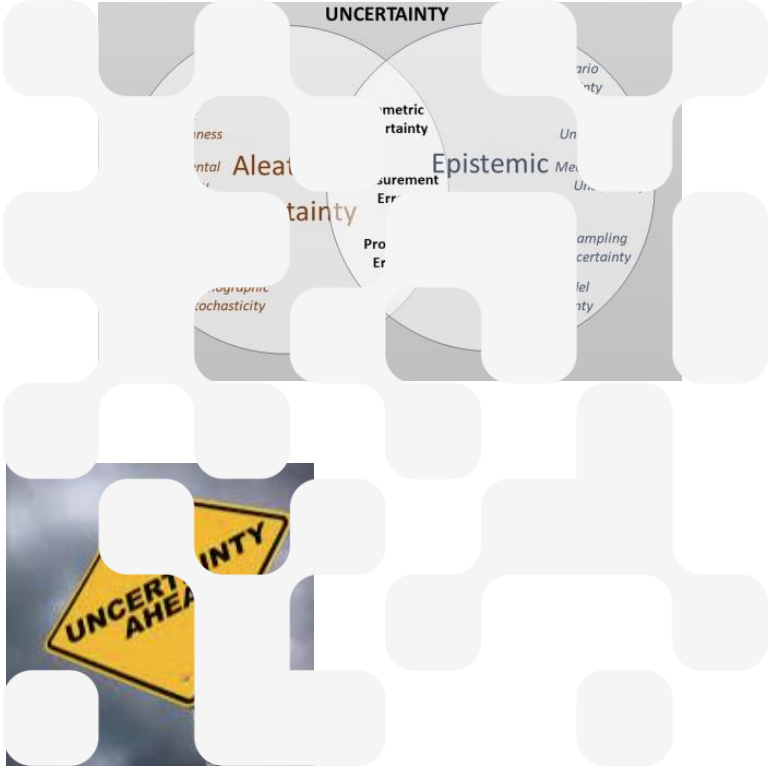
Aleatoric and Epistemic uncertainty intuition

Agenda

I. MAXIMUM LIKELIHOOD CRITERION (MLC)

II. UNCERTAINTY ESTIMATION USING MLC

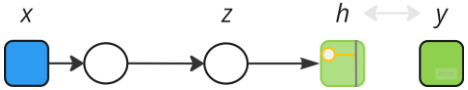
III. UNCERTAINTY ESTIMATION IN GENERAL



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/

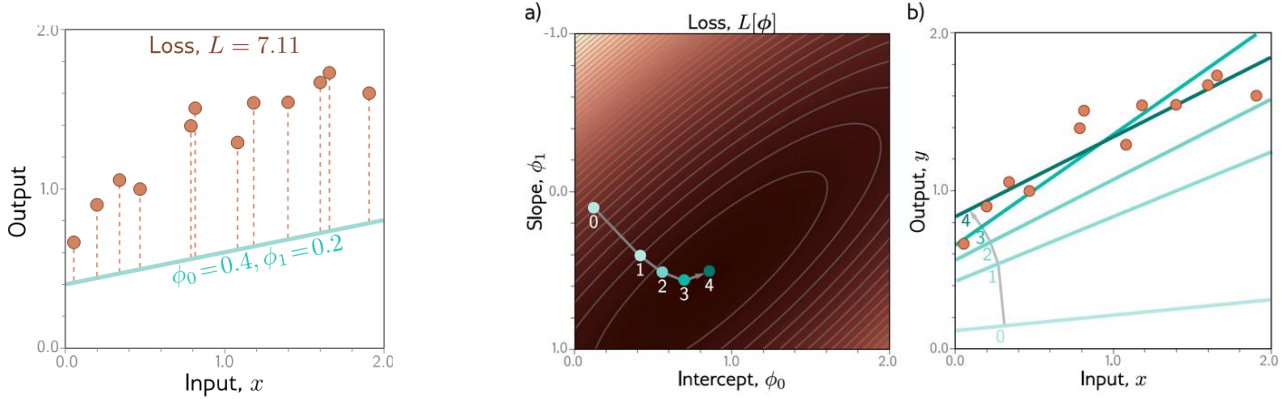
An uncertainty-aware loss intuition

Homoscedastic regression *vs* heteroscedastic regression

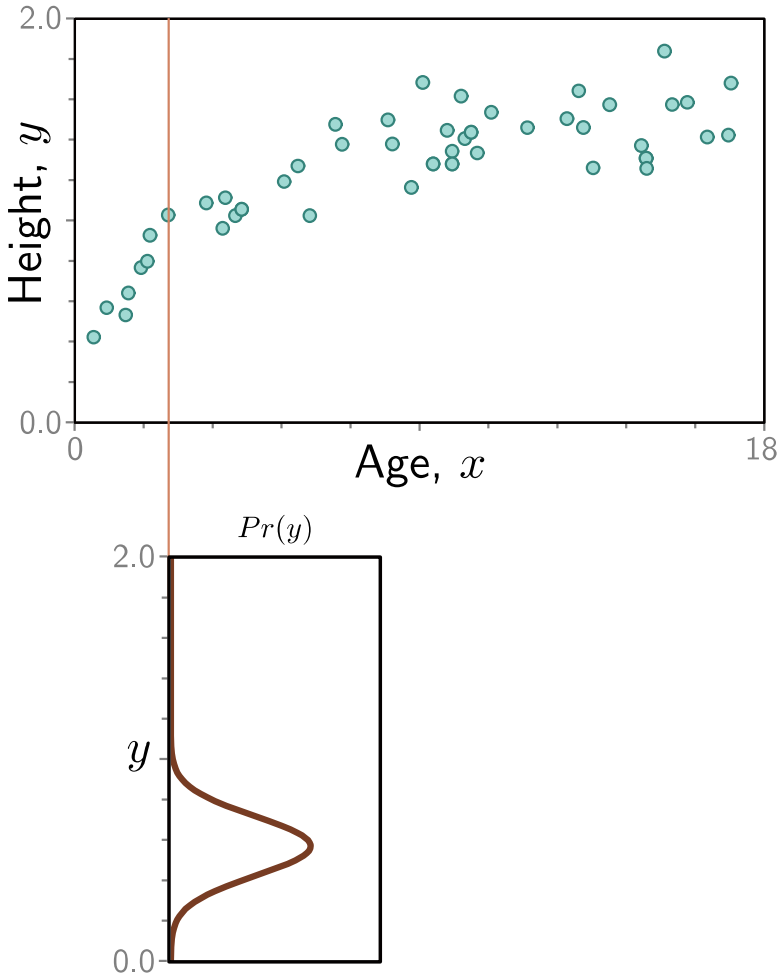


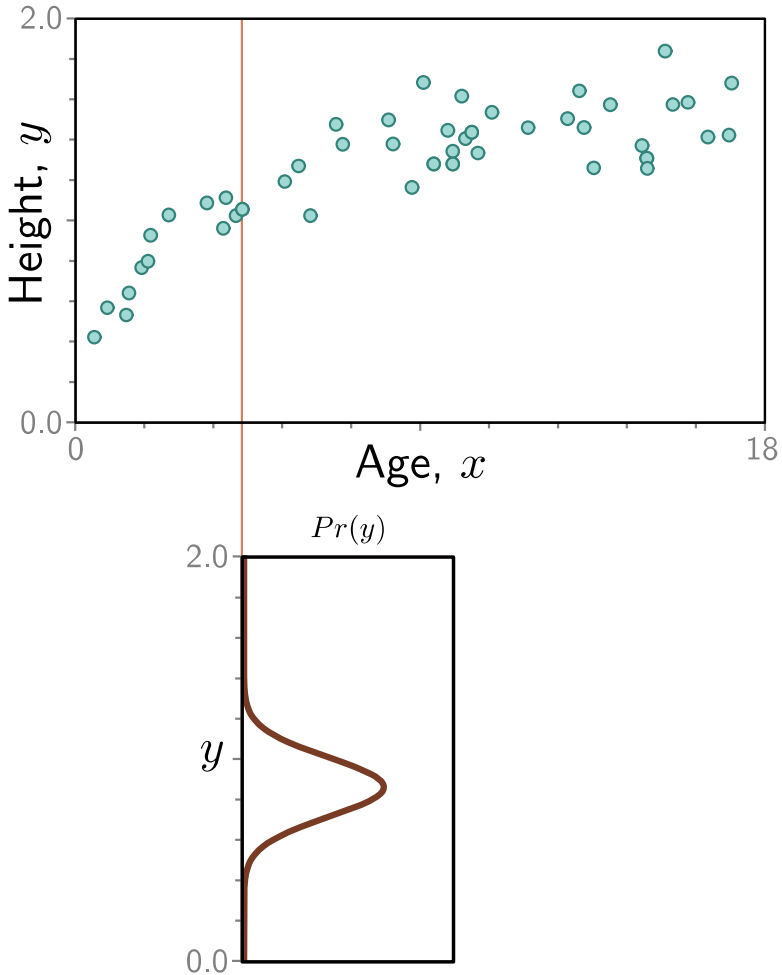
$$L_R = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - h^{(i)}\right)^2$$

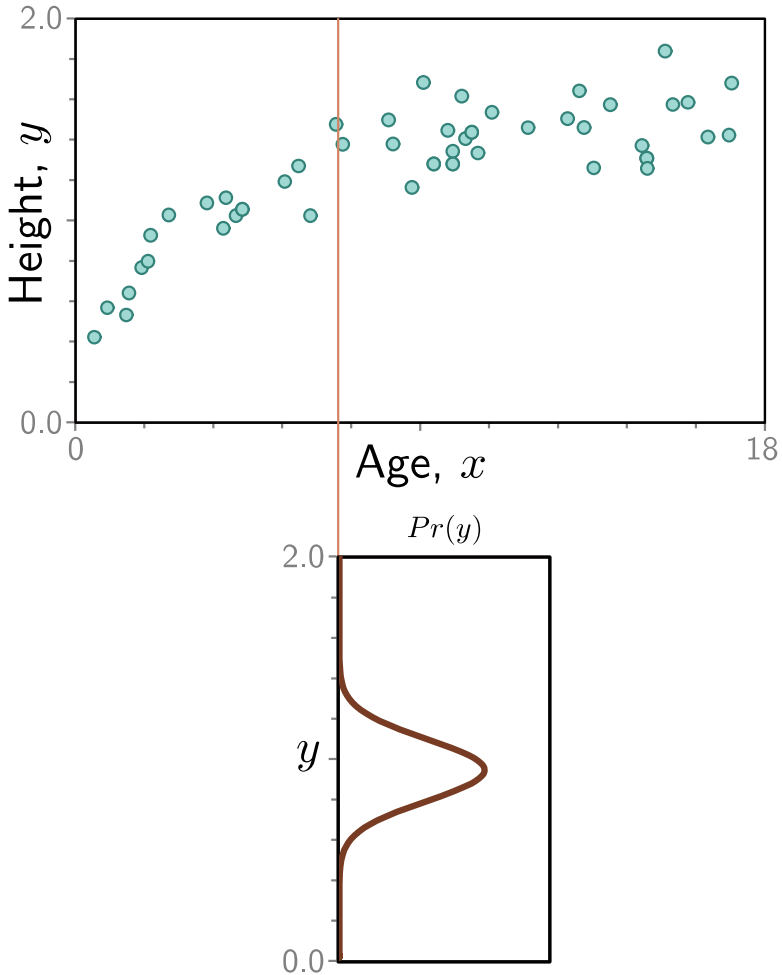
Model predicts output *h* given input *x*

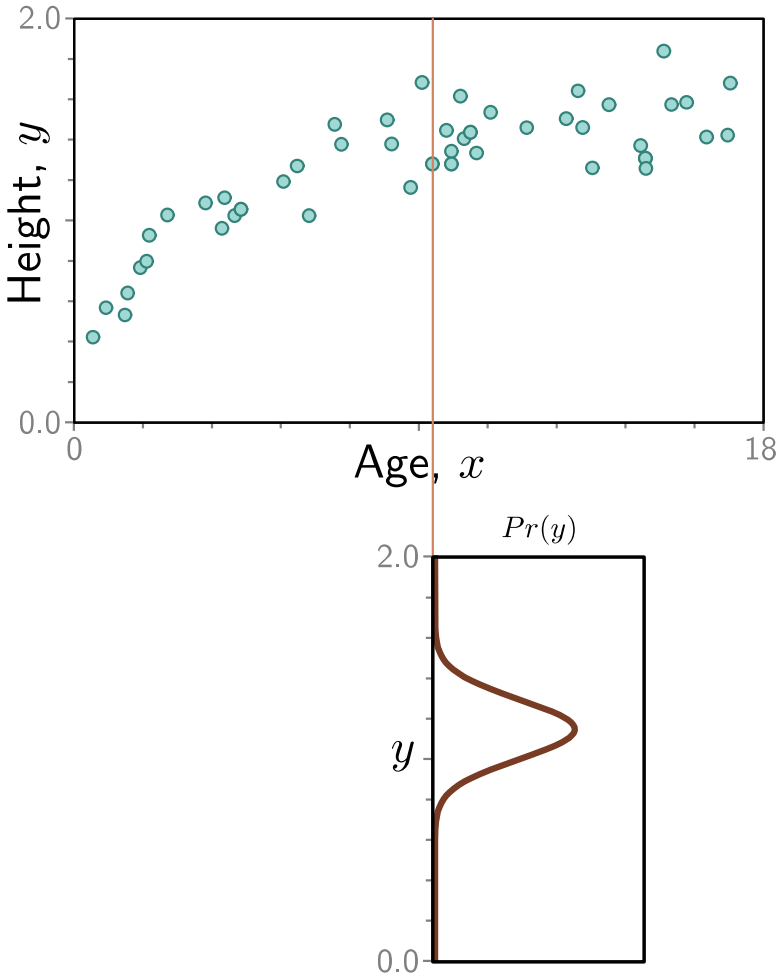


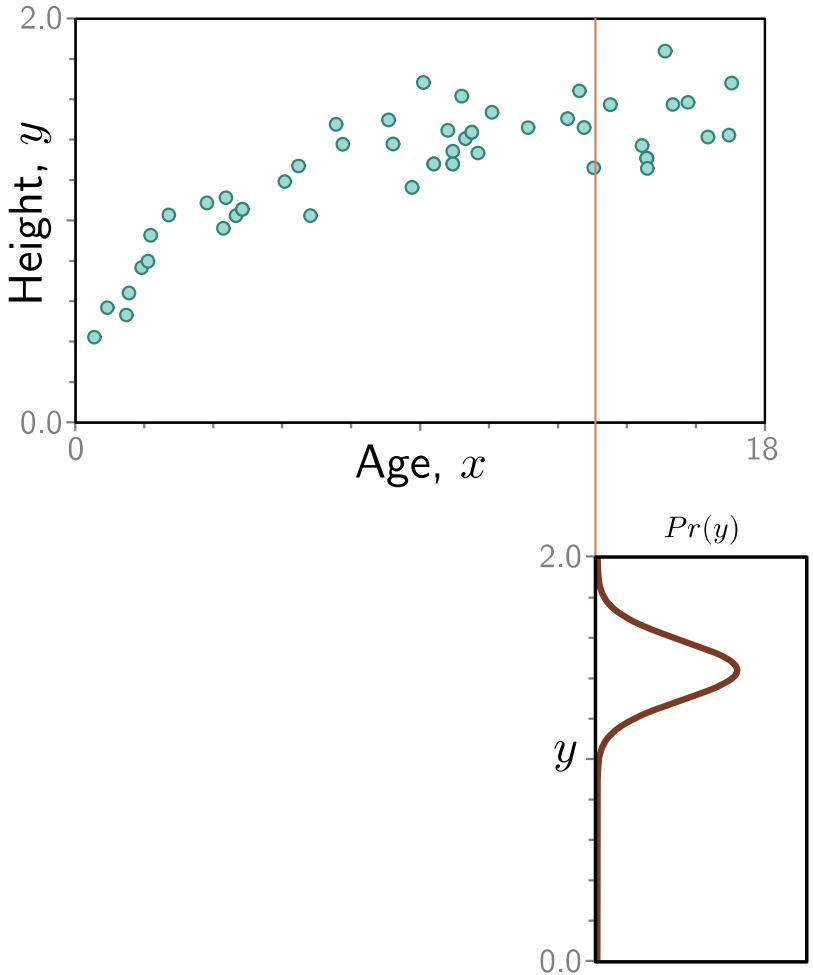
Supervised learning intuition: S. J. Prince. Understanding Deep Learning. MIT Press, 2023. URL http://udlbook.com.


2.0
0.0
Height, y
0
18
Age, x
$Pr(y)$
2.0
y
0.0


2.0
0.0
Height, y
0
18
Age, x
$Pr(y)$
2.0
y
0.0

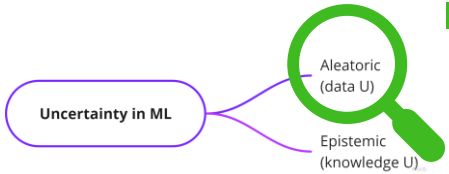

2.0
Height, y
0.0
0
18
Age, x
$Pr(y)$
2.0
y
0.0

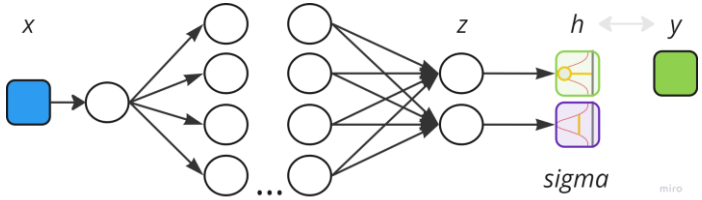

2.0
0.0
Height, y
0
18
Age, x
$Pr(y)$
2.0
y
0.0


2.0
0.0
Height, y
0
18
Age, x
$Pr(y)$
2.0
y
0.0

An uncertainty-aware loss intuition

Homoscedastic regression *vs* heteroscedastic regression





~~Model predicts output **h** given input **x**~~

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

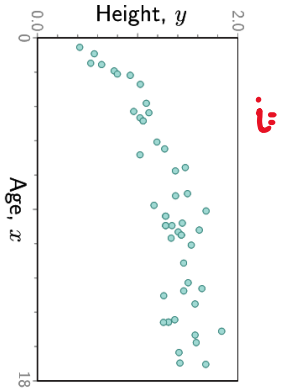
$$p\left(\mathbf{y}^{(i)}|\mathbf{f}[\mathbf{x}^{(i)},\mathbf{w}]\right)=\frac{1}{\sqrt{2\pi\sigma_{(i)}^2}}e^{-\frac{\left(y^{(i)}-h^{(i)}\right)^2}{2\sigma_{(i)}^2}}.$$

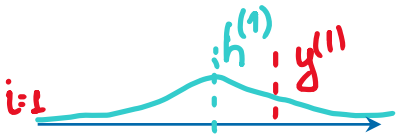
$$L=-\sum_{i=1}^{m}\log\left[\frac{1}{\sqrt{2\pi\sigma_{(i)}^2}}e^{-\frac{\left(y^{(i)}-h^{(i)}\right)^2}{2\sigma_{(i)}^2}}\right]\rightarrow min$$

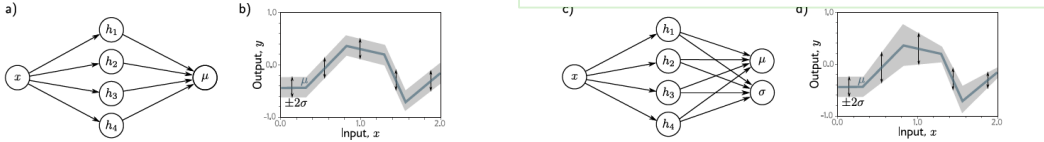
$\sigma_i^2=\sigma^2$

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

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



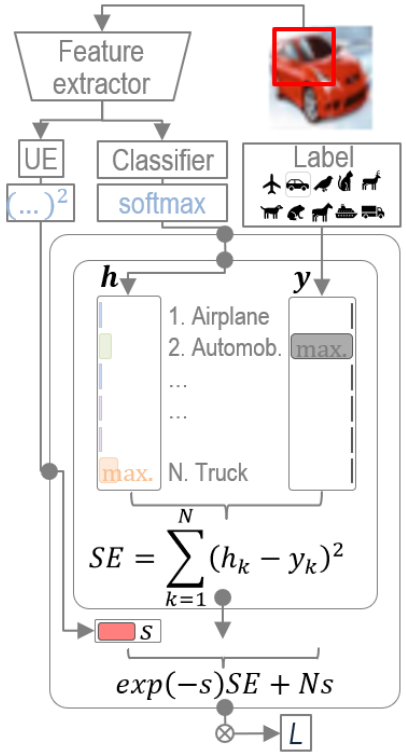


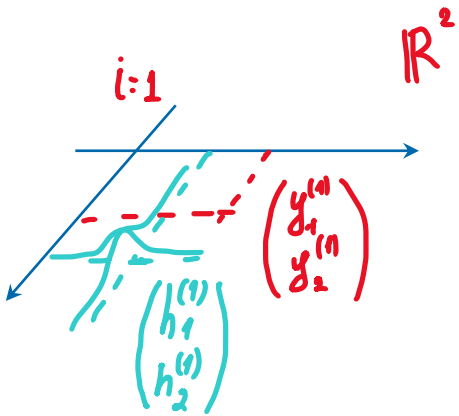


Supervised learning intuition: S. J. Prince. Understanding Deep Learning. MIT Press, 2023. URL http://udlbook.com.

An uncertainty-aware loss intuition

Trick that allows to use a regression loss in classification



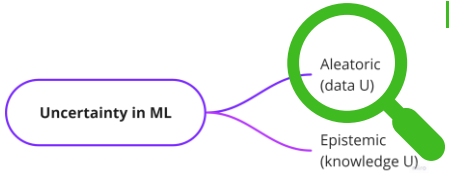


$$p\left(\mathbf{y}^{(i)}|\mathbf{f}[\mathbf{x}^{(i)},\mathbf{w}]\right) = \frac{1}{\sqrt{(2\pi\sigma_{(i)}^2)^N}} e^{-\frac{\sum_{k=1}^{N}\left(y_k^{(i)}-h_k^{(i)}\right)^2}{2\sigma_{(i)}^2}}$$

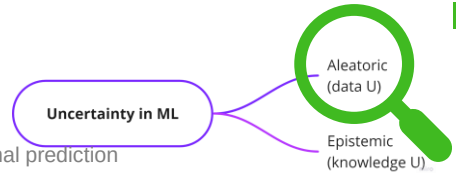
$$L_C = \frac{1}{2m}\sum_{i=1}^{m}\left(\frac{1}{\sigma_{(i)}^2}\sum_{k=1}^{N}\left(y_k^{(i)}-h_k^{(i)}\right)^2 + N\log\sigma_{(i)}^2\right)$$

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

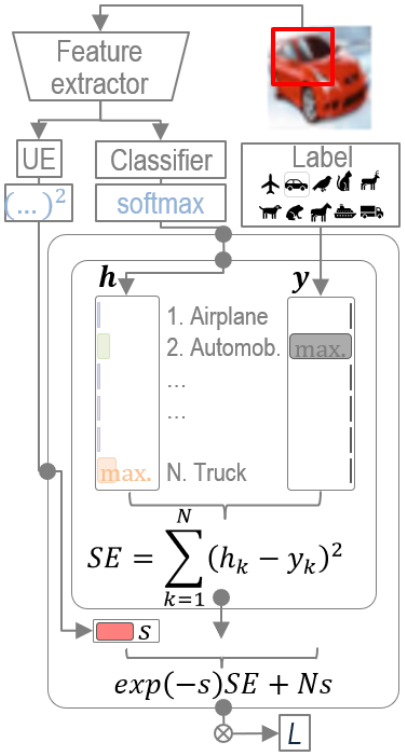
An uncertainty-aware negative log-likelihood (UANLL) loss intuition.

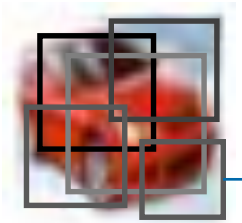


An uncertainty-aware loss intuition



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





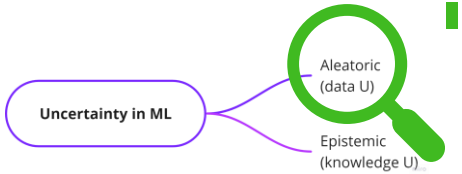
$$y^{(i,j)\prime} = \arg\max_k \left[h_k^{(i,j)}\right], \quad ce^{(i,j)} = 1 - sigm(s^{(i,j)})$$

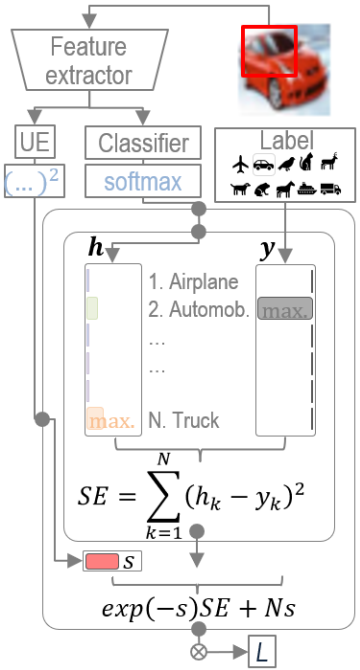
$$co^{(i,j)} = \max_k(h_k^{(i,j)})$$

$$z^{(i)}_{k=y^{(i,j)\prime}} = \sum_j g^{(i,j)}, \quad y^{(i)\prime} = \max_k(z_k^{(i)})$$

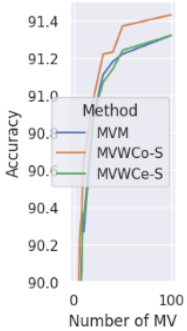
Asymmetric noise in labels. Training of the models

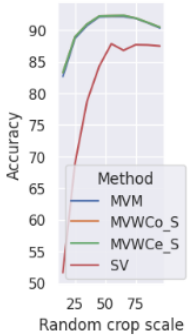
with 0%, 20%, and 40%: of replaced labels: airplane ↔ bird, automobile ↔ truck, cat ↔ dog, deer ↔ horse

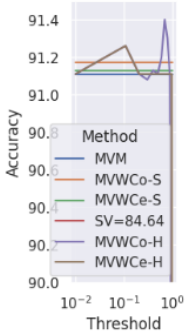


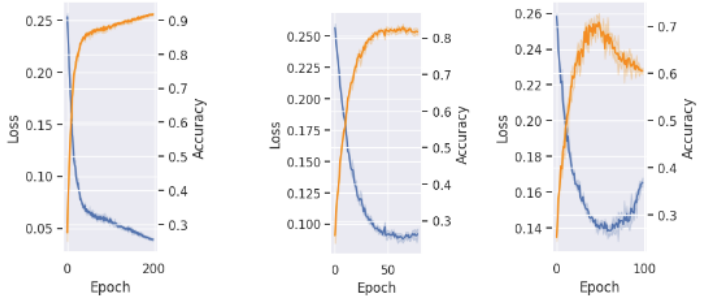


Uncertainty-aware negative log-likelihood (UANLL) loss intuition.





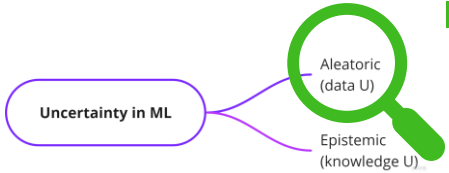


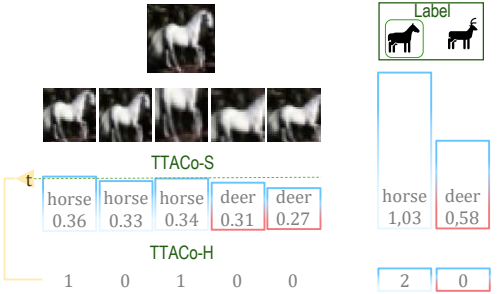


(a) 0% (b) 20% (c) 40%

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

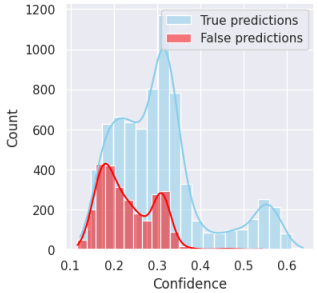


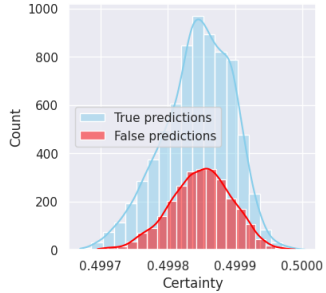


Neptune run: Play-934
Assymptotic noise in labels of 40%:
airplane ↔ bird, automobile ↔ truck, cat ↔ dog, deer ↔ horse

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]



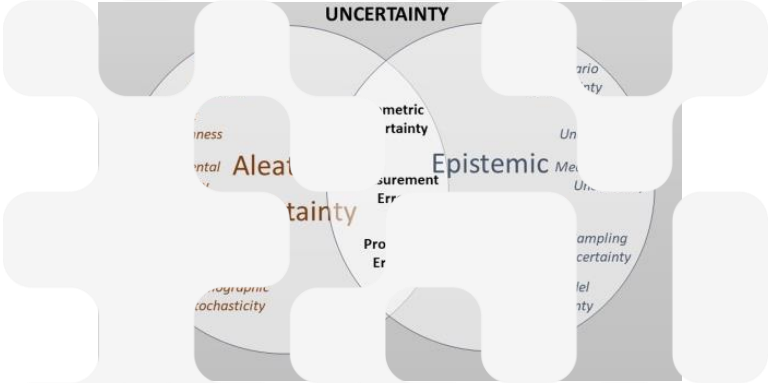


Agenda

I. MAXIMUM LIKELIHOOD CRITERION (MLC)

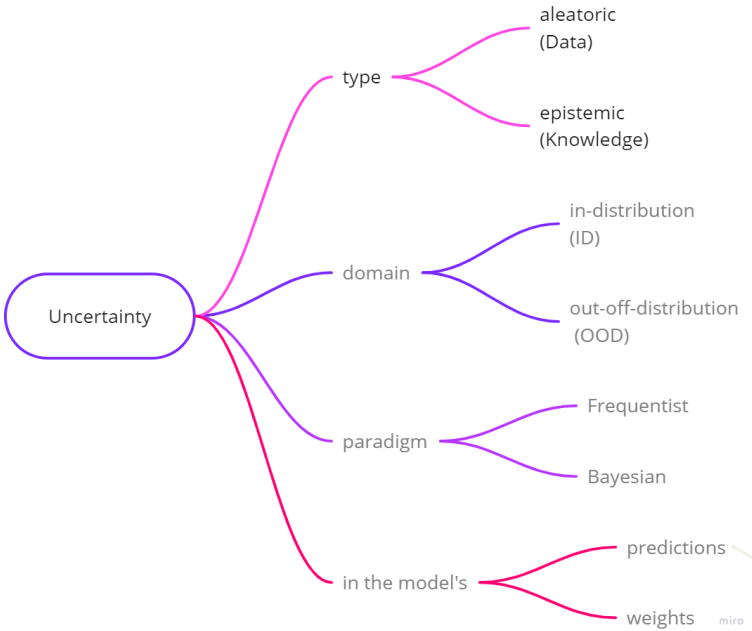
II. UNCERTAINTY ESTIMATION USING MLC

III. UNCERTAINTY ESTIMATION IN GENERAL

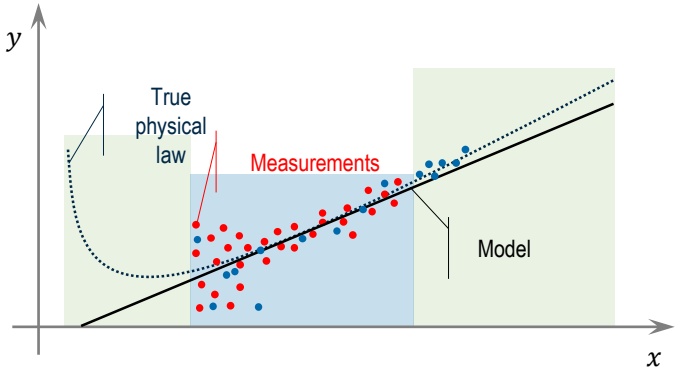


All models are wrong, but ~~some~~ models that know when they are wrong, are useful
/George Box + unknown researcher from the Google AI Brain Team/

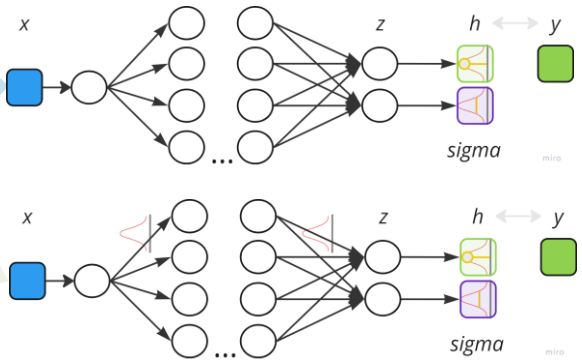
Terms and formalization



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.

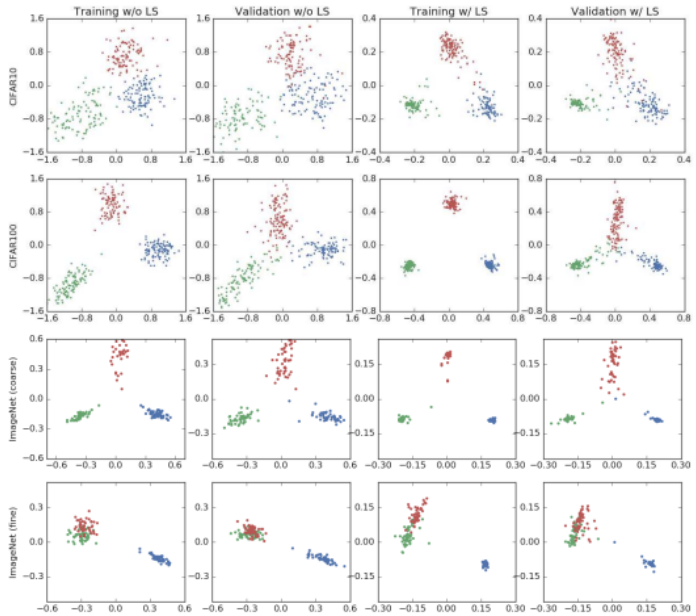


Figure 1: Visualization of penultimate layer's activations of: AlexNet/CIFAR-10 (first row), CIFAR-100/ResNet-56 (second row) and ImageNet/Inception-v4 with three semantically different classes (third row) and two semantically similar classes plus a third one (fourth row).

R Muller et al. When does label smoothing help? NIPS, 2019

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:

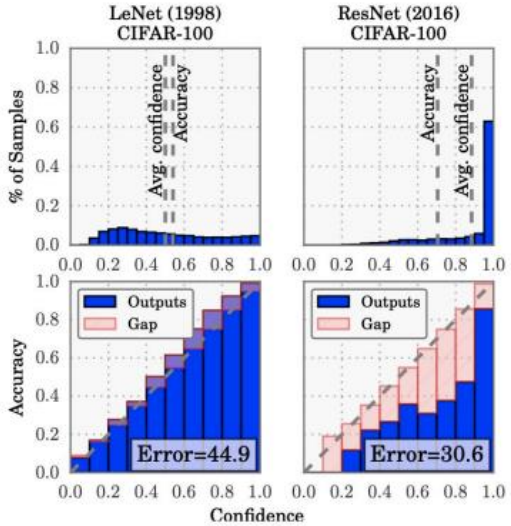
Instead of using the exact one-hot vectors, label smoothing introduces a small amount of noise to the labels. This is done by replacing the hard 0s and 1s with values slightly less than 1 and slightly greater than 0, respectively. For example, with a smoothing factor ϵ (typically a small value like 0.1),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

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

```python
import tensorflow as tf
from tensorflow.keras.losses import CategoricalCrossentropy

# Define the smoothing factor
epsilon = 0.1

# Create a label smoothing loss function
loss_fn = CategoricalCrossentropy(from_logits=True, label_smoothing=epsilon)

# Example usage in model training
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.



Ch Guo et al. On Calibration of Modern Neural Networks. ICML, 2017

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:

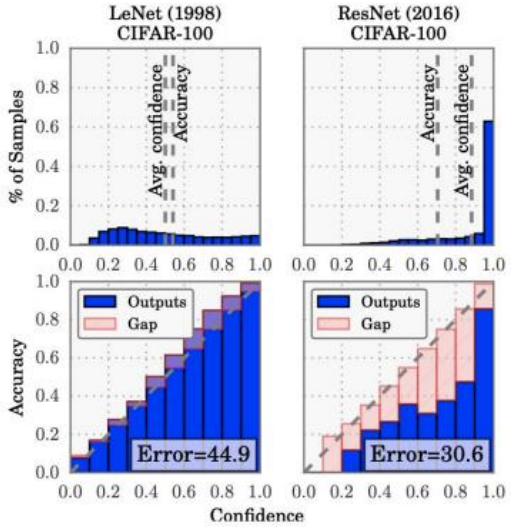
$$\text{softmax}(z_i/T) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

   - 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.

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



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

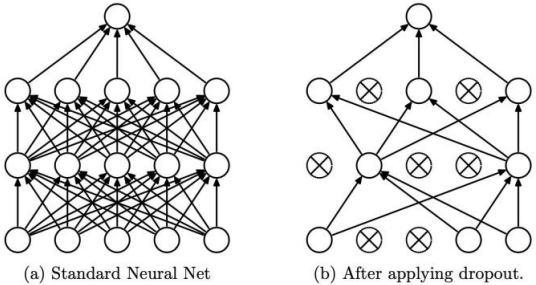
Bin	# Predictions	Avg. Pred. Prob.	Accuracy	Difference
1	100	0.1	0.08	0.02
2	150	0.2	0.18	0.02
3	200	0.3	0.27	0.03
…	…	…	…	…
10	50	0.9	0.92	0.02

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

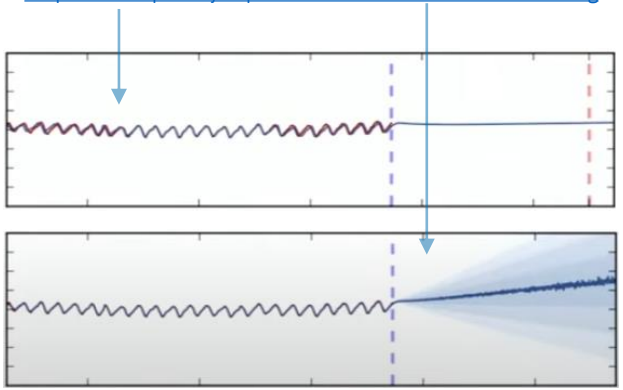
Ch Guo et al. On Calibration of Modern Neural Networks. ICML, 2017

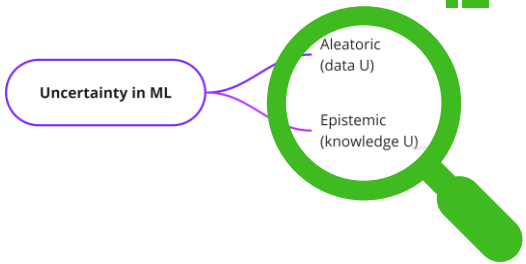
Dropout training method

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

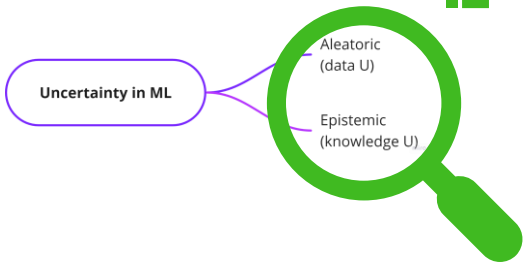




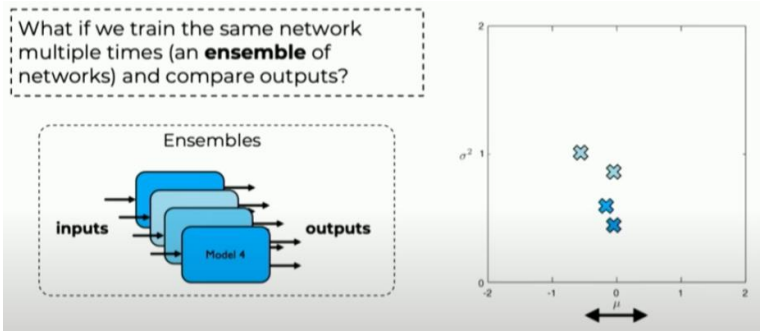

```
# Given x:
# - drop units at test time
# - repeat 10 times
# - and look at mean and sample variance

y = []
for _ in xrange(10):
    y.append(model.output(x,dropout=True))
y_mean = numpy.mean(y)
y_var = numpy.var(y)
```

Yarin Gal - Uncertainty in Deep Learning | MLSS Kraków 2023

Ensembling and test-time augmentation




```python
num_ensembles = 5
for i in xrange(num_ensembles):
    model = create_model(...)
    model.fit(...)

raw_predictions = [models[i].predict(x)
    for i in range(num_ensembles)]
y_mean = numpy.mean(raw_predictions)
y_var = numpy.var(raw_predictions)
```



MIT Introduction to Deep Learning 6.S191: Lecture 5 Robust and Trustworthy Deep Learning Lecturer: Sadhana Lolla

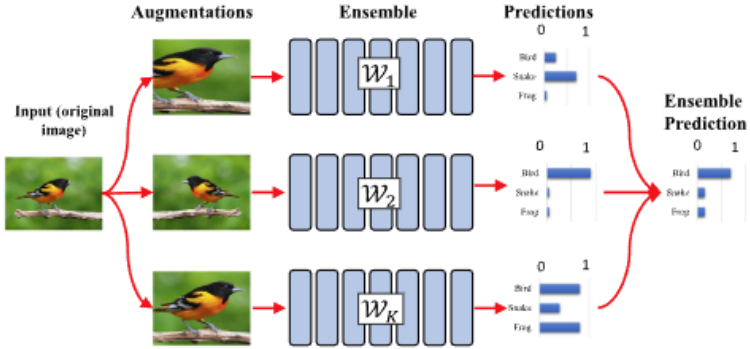
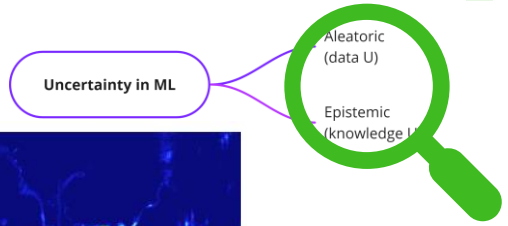
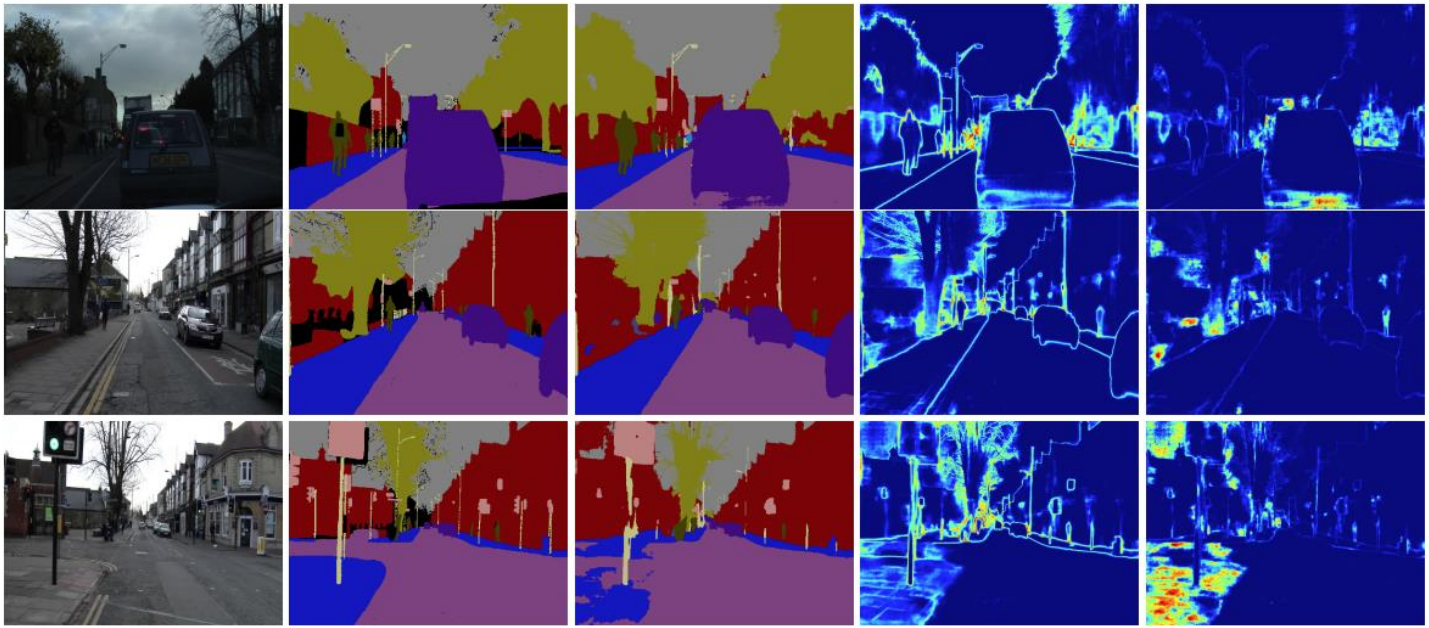


Fig. 15. A schematic view of TTA for ensembling techniques.
Source: Reproduced based on [179].

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

UE in segmentation

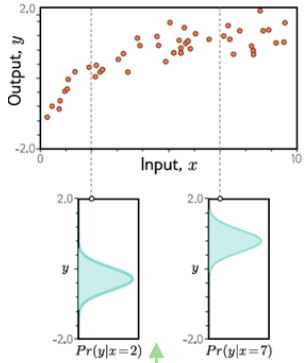




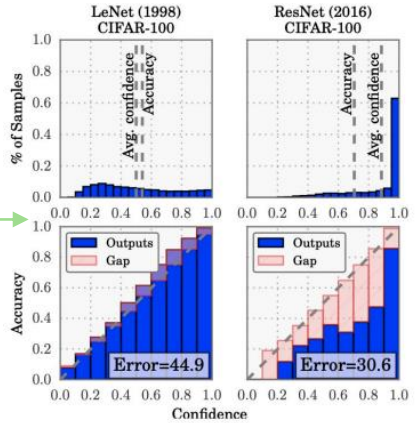
(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

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

Recap

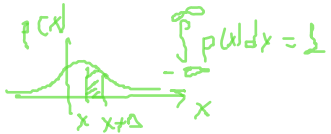
$h^i \leftrightarrow y^i$

1. BCE $L = -\sum_i \left[(1 - y^{(i)}) \ln(1 - h^i) + y^i \ln h^i \right]$

① ②

$y^i = 1$, $h^i \to 0$ ① 0 ② $-\infty$

1 балл



2. $p(x, y) = p(y|x)\, p(x)$

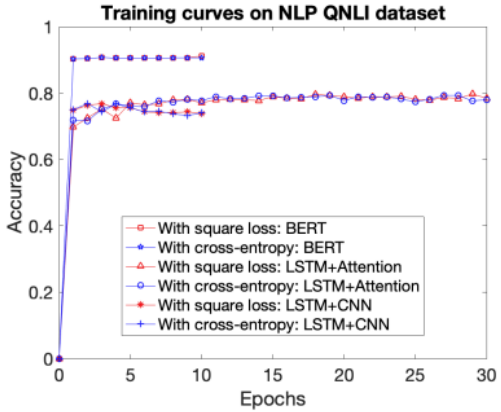
1 балл

Предп. x и y независ. соб., тогда $p(x, y) = p(y)\, p(x)$

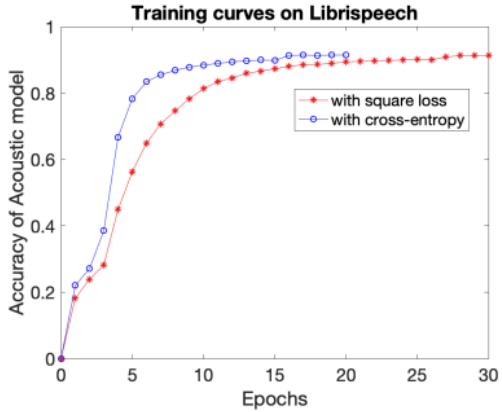
3. $p(y|x) = \frac{p(x, y)}{p(x)}$ $\Big| \int dy \to$?

2 балла

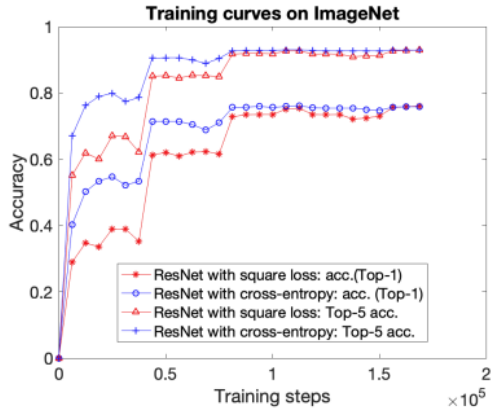
Related work: a square loss classifies well



(a) NLP tasks



(b) ASR tasks



(c) Vision tasks

Like Hui, Mikhail Belkin What Evaluation of Neural Architectures Trained with Square Loss vs Cross-Entropy in Classification Tasks

Related work: ensembling and other methods

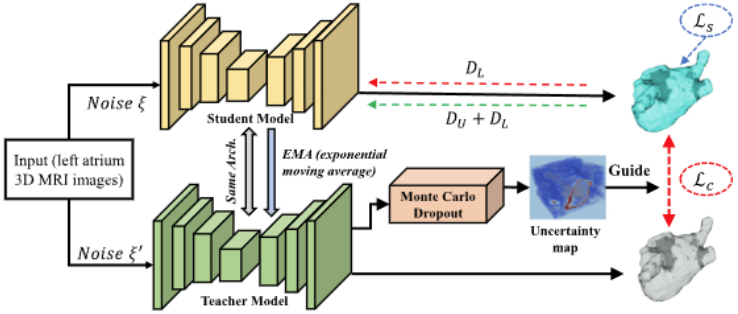
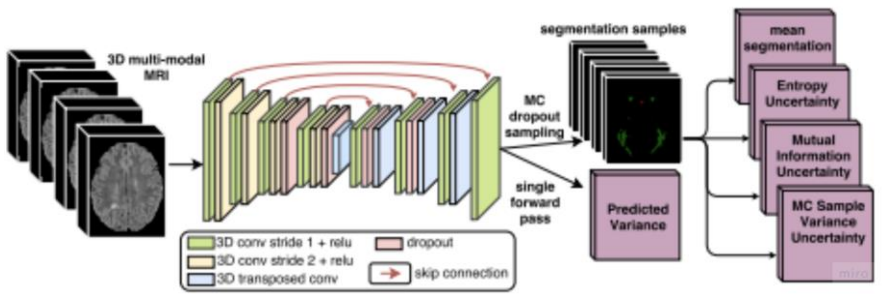


Fig. 5. A general view demonstrating the application of the semi-supervised UA-MT framework to LA segmentation.
Source: Reproduced based on [43].



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

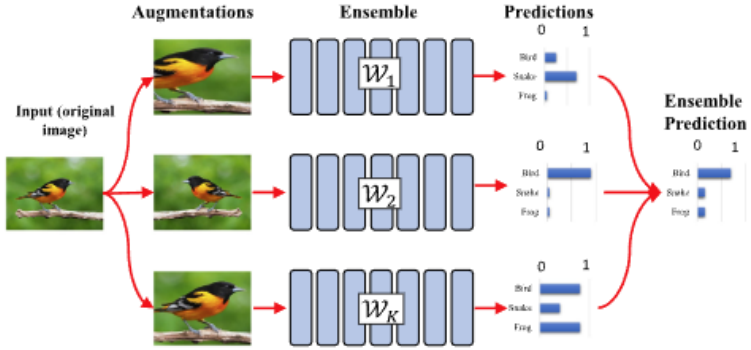
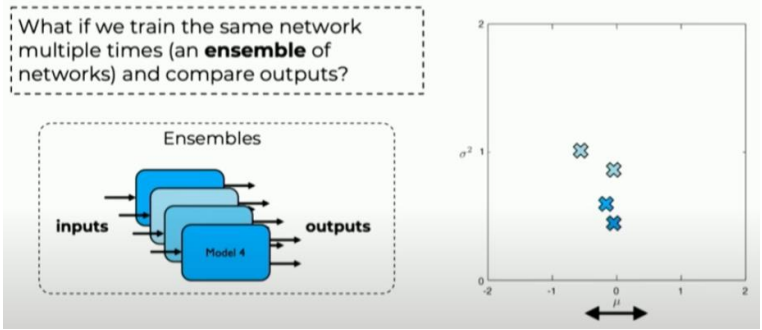


Fig. 15. A schematic view of TTA for ensembling techniques.
Source: Reproduced based on [179].



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