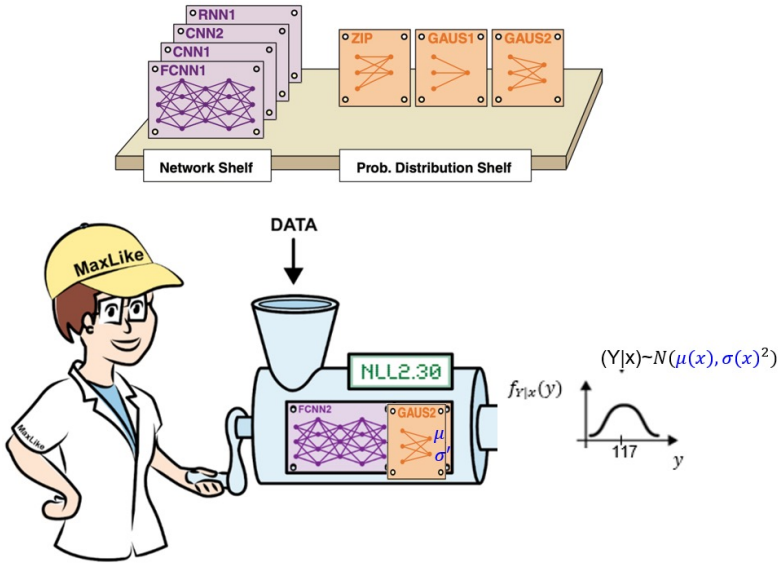
WBL Probabilistic Deep Learning:: Day 4

Beate Sick, Oliver Dürr

Day 4: Modelling Uncertainty



Topics

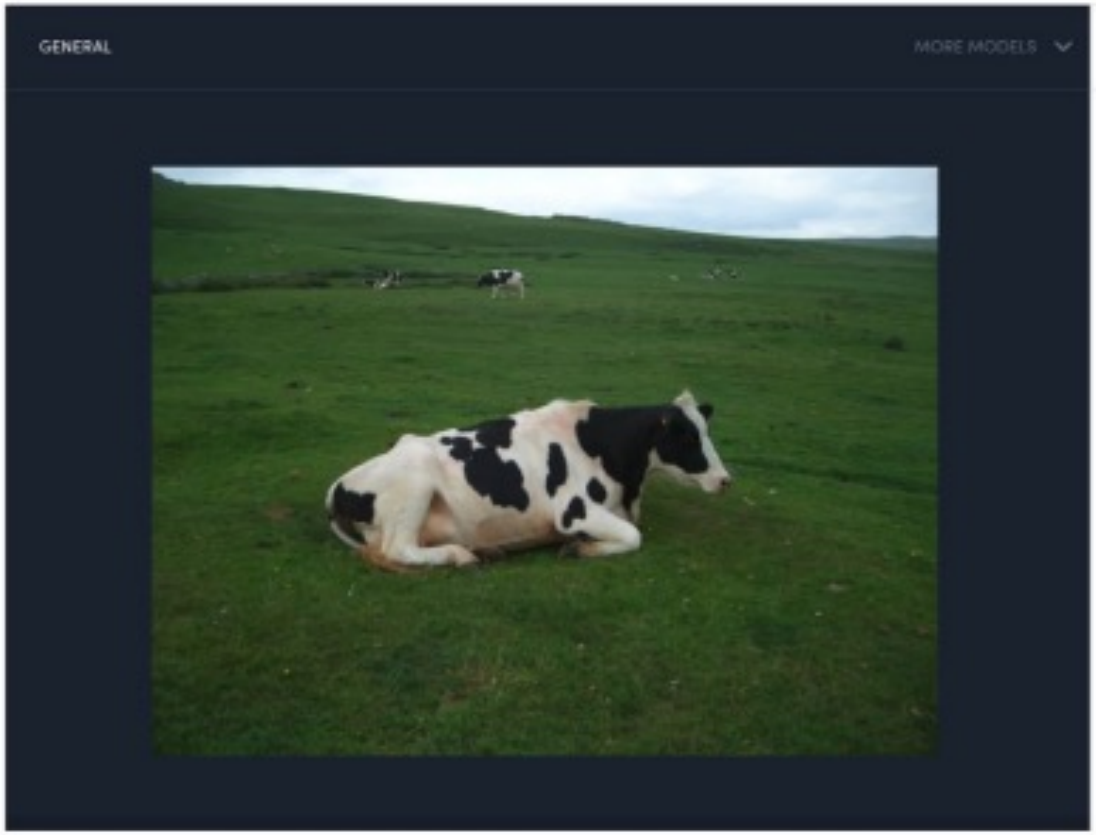
- Half day 1
  - Introduction to probabilistic DL
  - Introduction to Keras with pytorch backend
- Day 2
  - Loss function and optimization
  - Convolutional Neural Networks (CNN) for image data
  - Working with pretrained models
  - Transfer learning
  - Projects
- Day 3
  - Understand where the NN looks at
  - Model evaluation: Prediction Performance & Calibration
  - Transformer Architectures
  - Projects
- Half day 4
  - Modelling Uncertainty 13:00 – ~15:00 Lecture
  - Project presentations
    - 15:15 Spotlight Talk
    - 15:45 Poster Session

Date
03.02.2025
10.02.2025
17.02.2025
24.02.2025

Outline

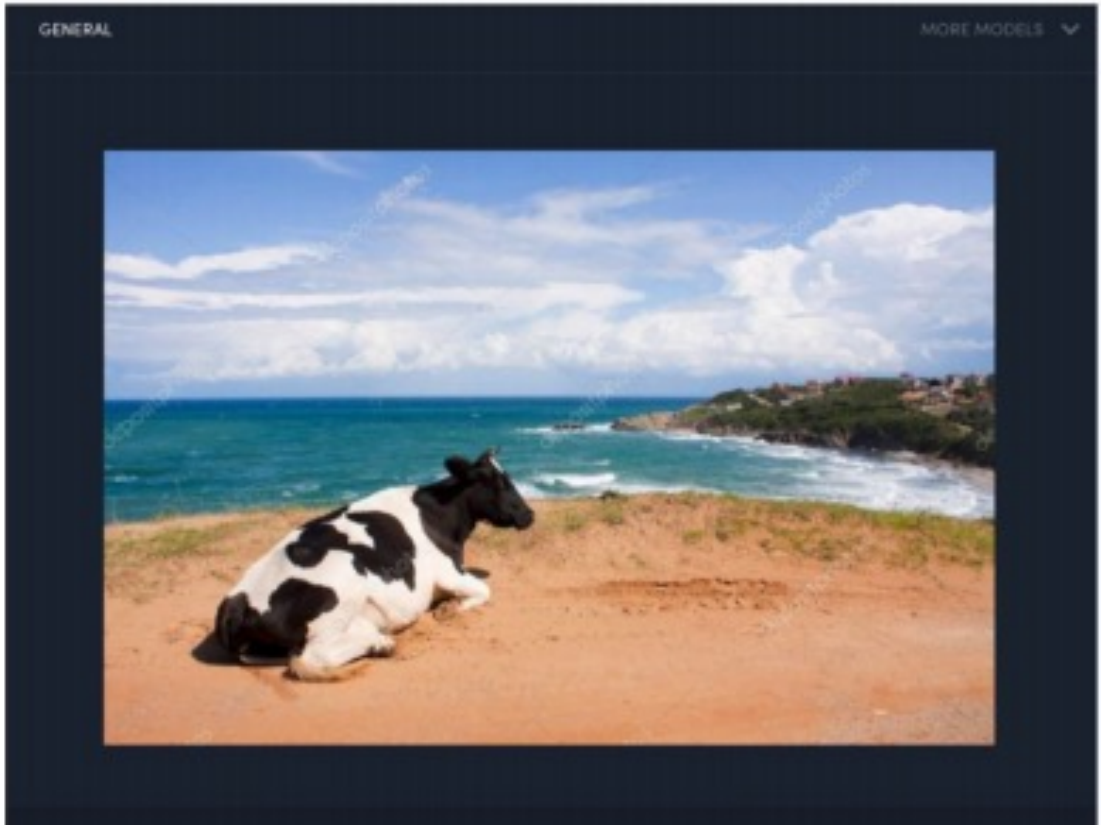
- Issues with current DL approach
  - No extrapolating / no epistemic uncertainty
- Model averaging using Ensembling
- Model averaging using Bayesian Statistics
  - A simple example (Bayes the Hacker's way)
  - Introduction to Bayesian Statistics
- Bayesian Neural Networks and Approximations
  - Variational Approximation
  - MC-Dropout

Typical example of a cow



General	VIEW DOCS
cow	0.992
cattle	0.983
mammal	0.979
grass	0.978
livestock	0.966
farm	0.964
landscape	0.963
pasture	0.954
grassland	0.949
agriculture	0.948
no person	0.945

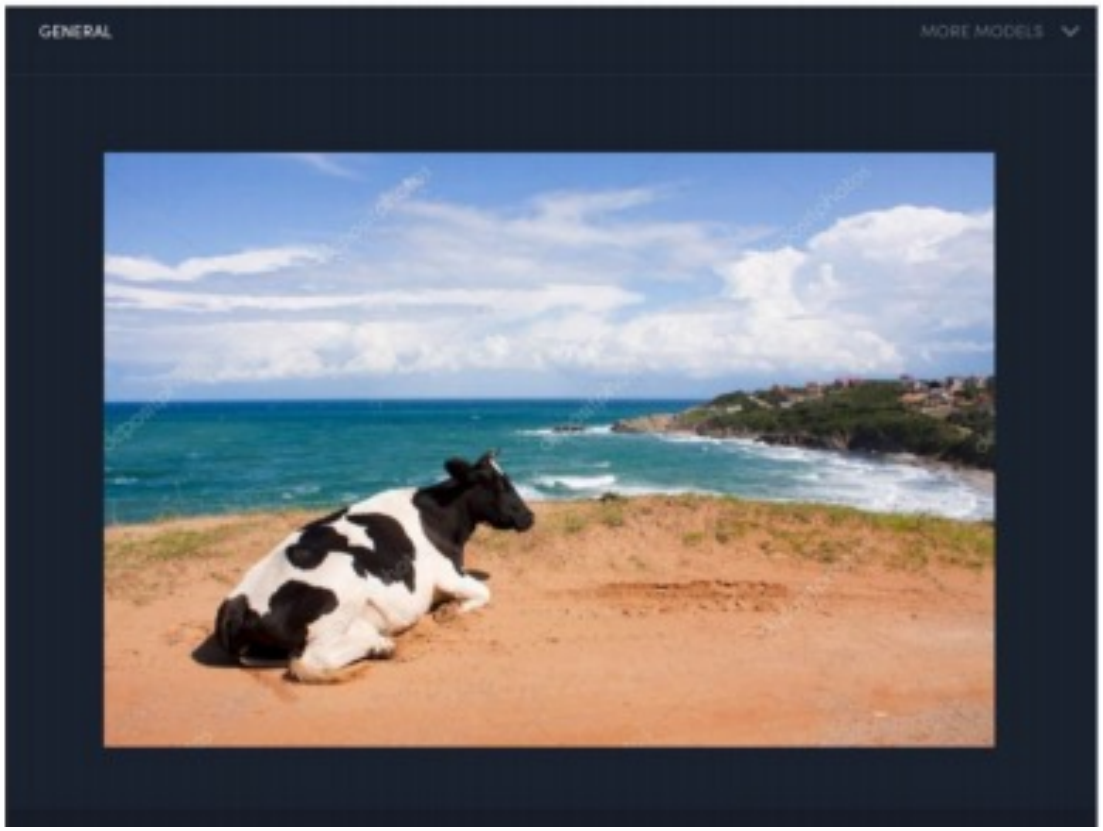
Non-Typical Example of a cow



Non-Typical Example of a cow

Image is too far away from training data

Does not detect cow

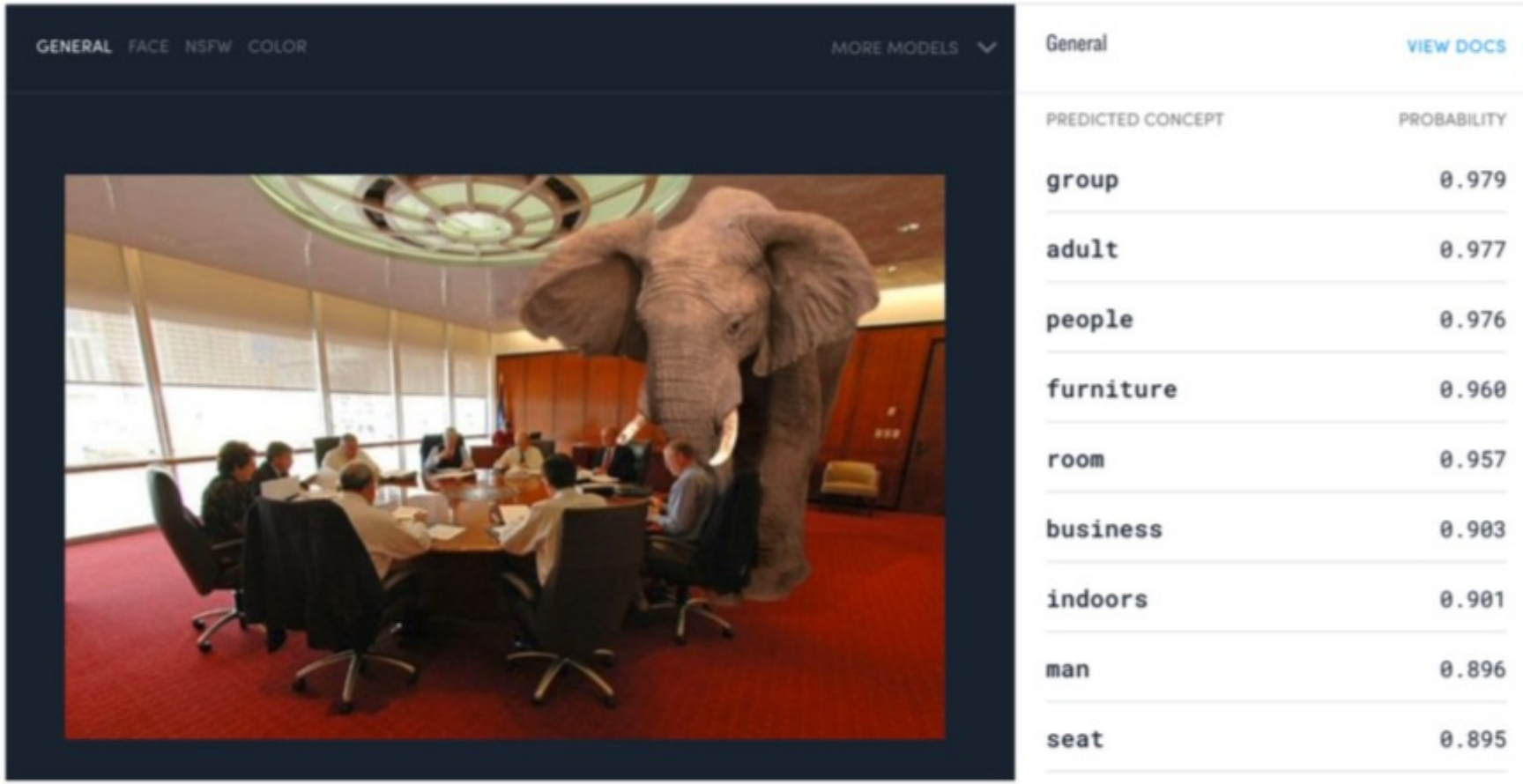


General	VIEW DOCS
no person	0.991
beach	0.990
water	0.985
sand	0.981
sea	0.980
travel	0.978
seashore	0.972
summer	0.954
sky	0.946
outdoors	0.944
ocean	0.936

NeurIPS 2018 Workshop on Causal Learning

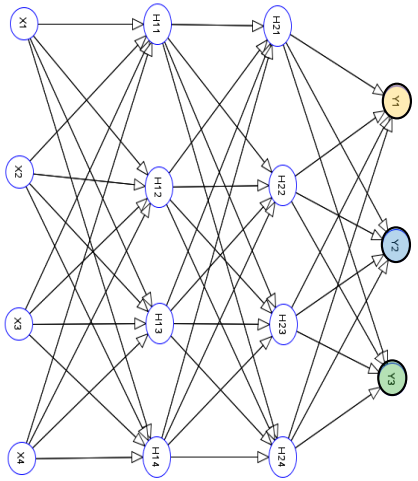
The elephant in the room

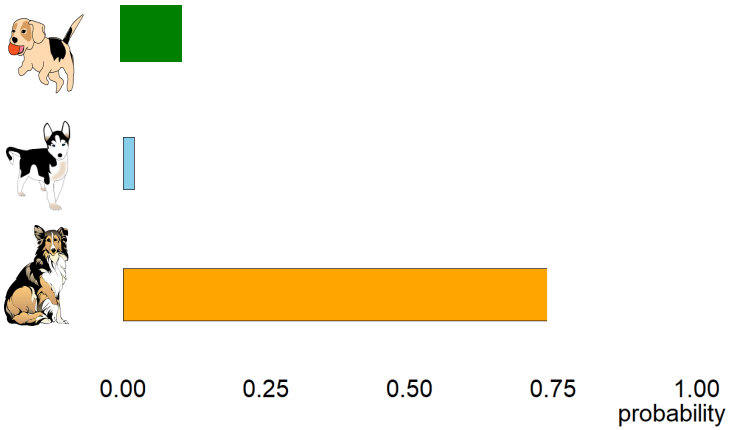
No elephant in prediction



What do we get from a DL classification model?

Suppose you train a classifier to discriminate between 3 dog breeds



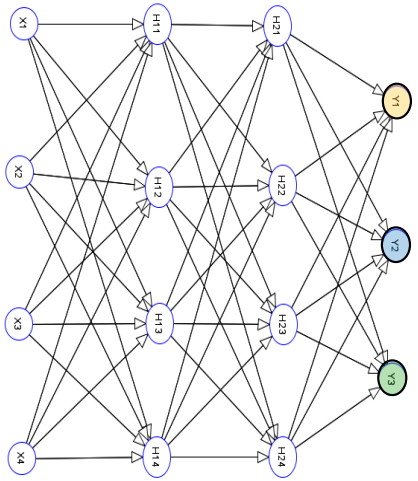


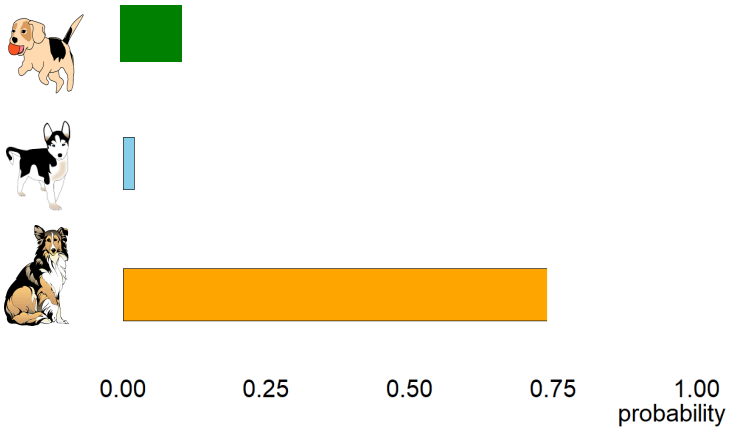
The prediction is “collie” because it gets the highest probability: p_{max}=0.75

Recap what is 0.75 telling us?

What do we get from a DL classification model?

Suppose you train a classifier to discriminate between 3 dog breeds

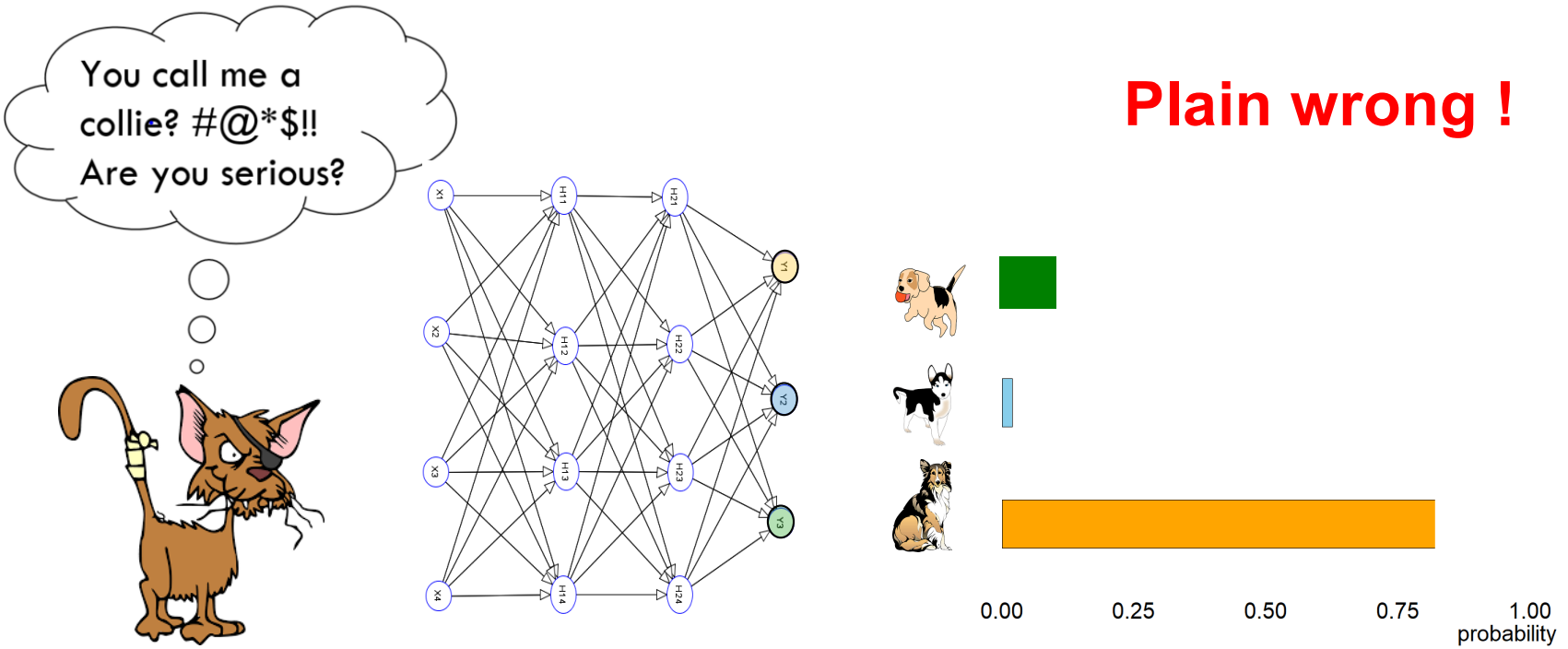




The prediction is “collie” because it gets the highest probability: p_{max}=0.75

What is 0.75 telling us?
In a calibrated classifier in 75 out of 100 predictions we have a Collie

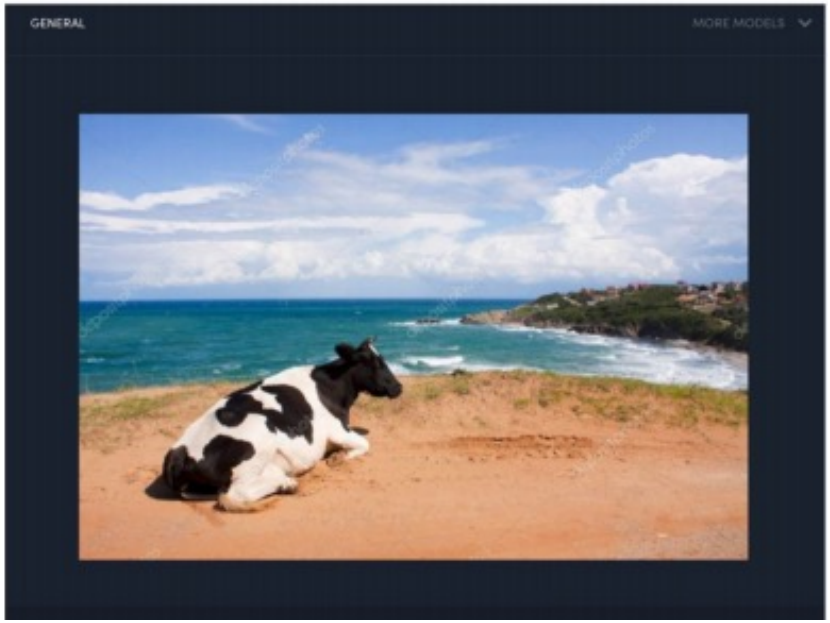
What happens if a novel class is presented to the CNN?

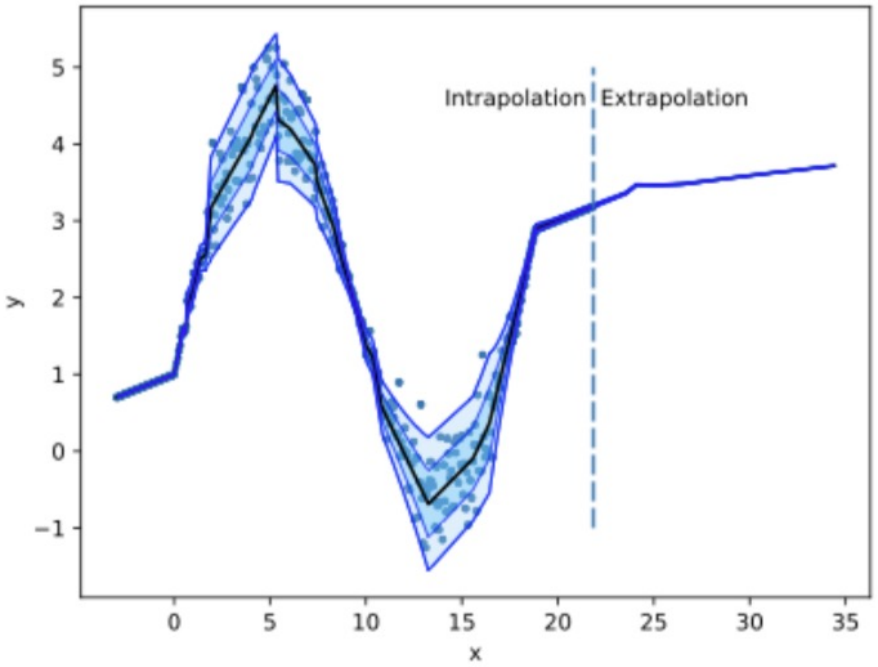


Plain wrong !

The reported probability is only valid if, we have the P(Train)=P(Test).
That's not the case. "The big lie deep learning is based on".
This is more evident, if we have a look at regression problems.

Extrapolation

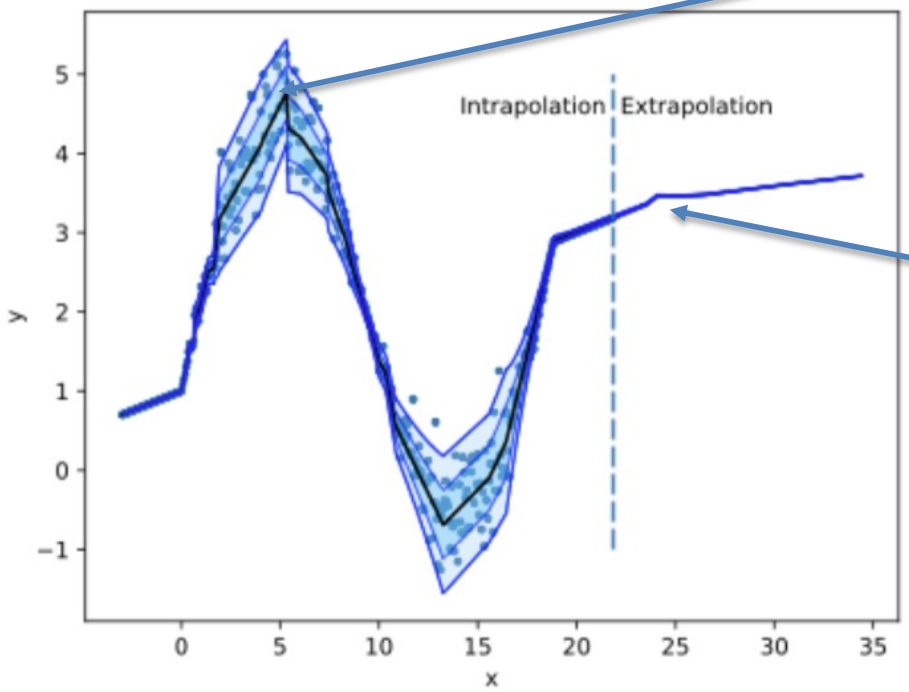




Deep Learning is not robust against out of distribution data (OOD)

Aleatroic vs. Epistemic Uncertainty

Much spread in data, aleatroic (from “Alea Acta est”))



No epistemic uncertainty, in standard DL

- *Aleatoric* uncertainty is due to the uncertainty in the data (also in the limit of $N \rightarrow \infty$)

- The uncertainty when leaving the ‘known ground’ is called *epistemic* uncertainty.

We can model the epistemic uncertainty when we take the uncertainty with which we know the weights (called parameter uncertainty) into account.

DL should know when it doesn't

- In many situations, we can't guaranty P(Train)=P(Test).
- DL better state this uncertainty

Foto: https://www.crn.de/software-services/halloween-training-fuer-google-autos.108409.html

Deep Ensembling

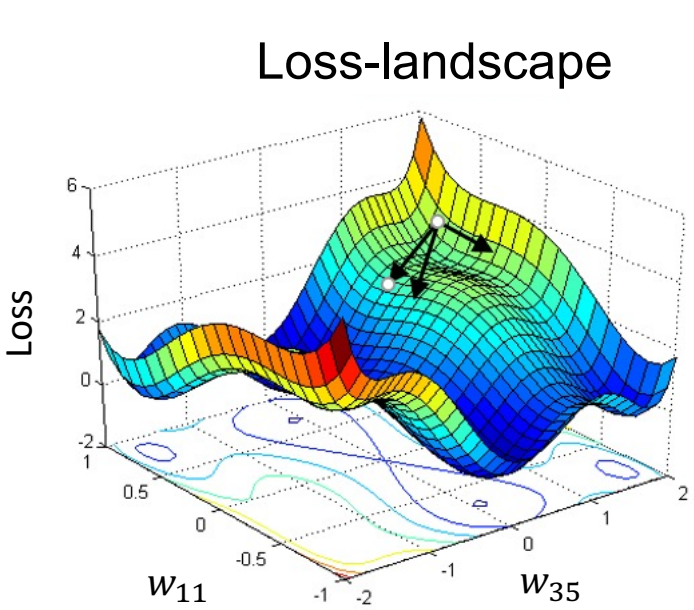
Original paper by Lakshminarayanan et al.: https://arxiv.org/pdf/1612.01474.pdf

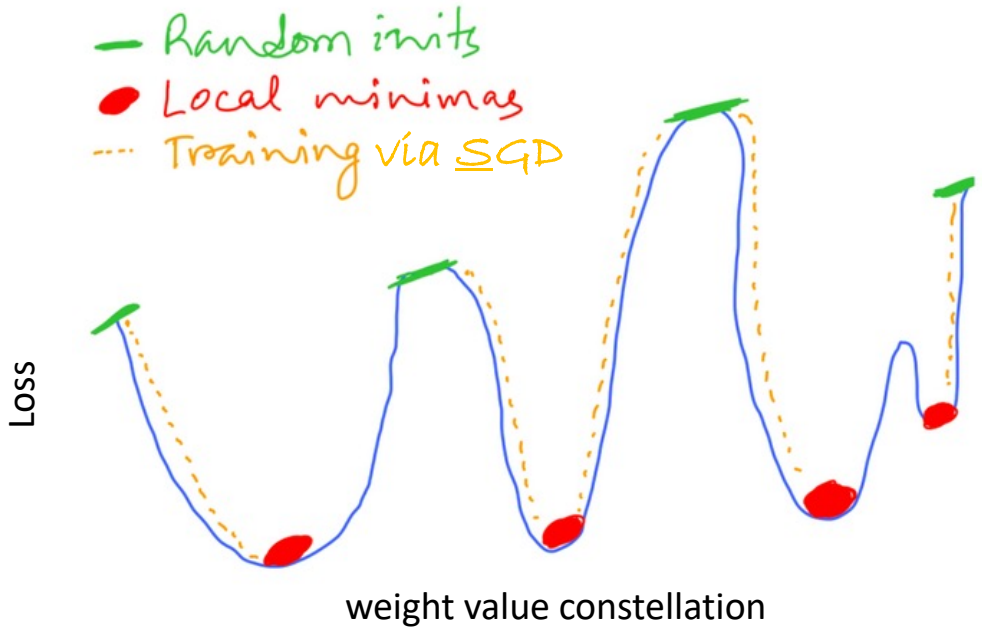
Basic Idea of deep ensembling

- Use an ensemble of n-different models and see if they agree
- How to get different ensemble members?
- Traditional Statics?
  - Bagging Boot Strapping and Aggregating
- Deep Learning
  - Deep Learning just average over different solutions of gradient descent

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, *30*.

The loss-landscape in DL is usually not convex



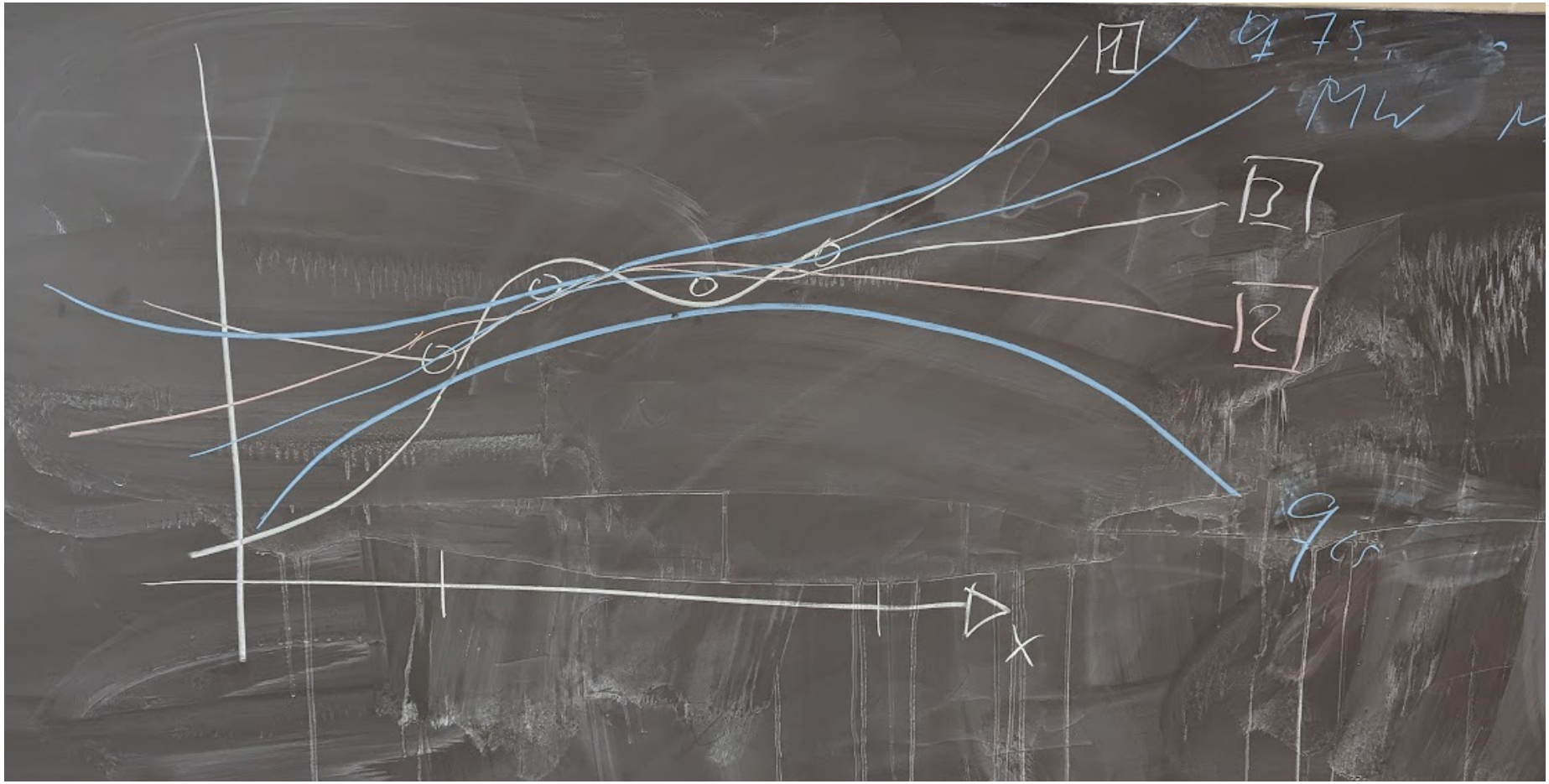


The loss-landscape of DL models has many local minima with similar depth.

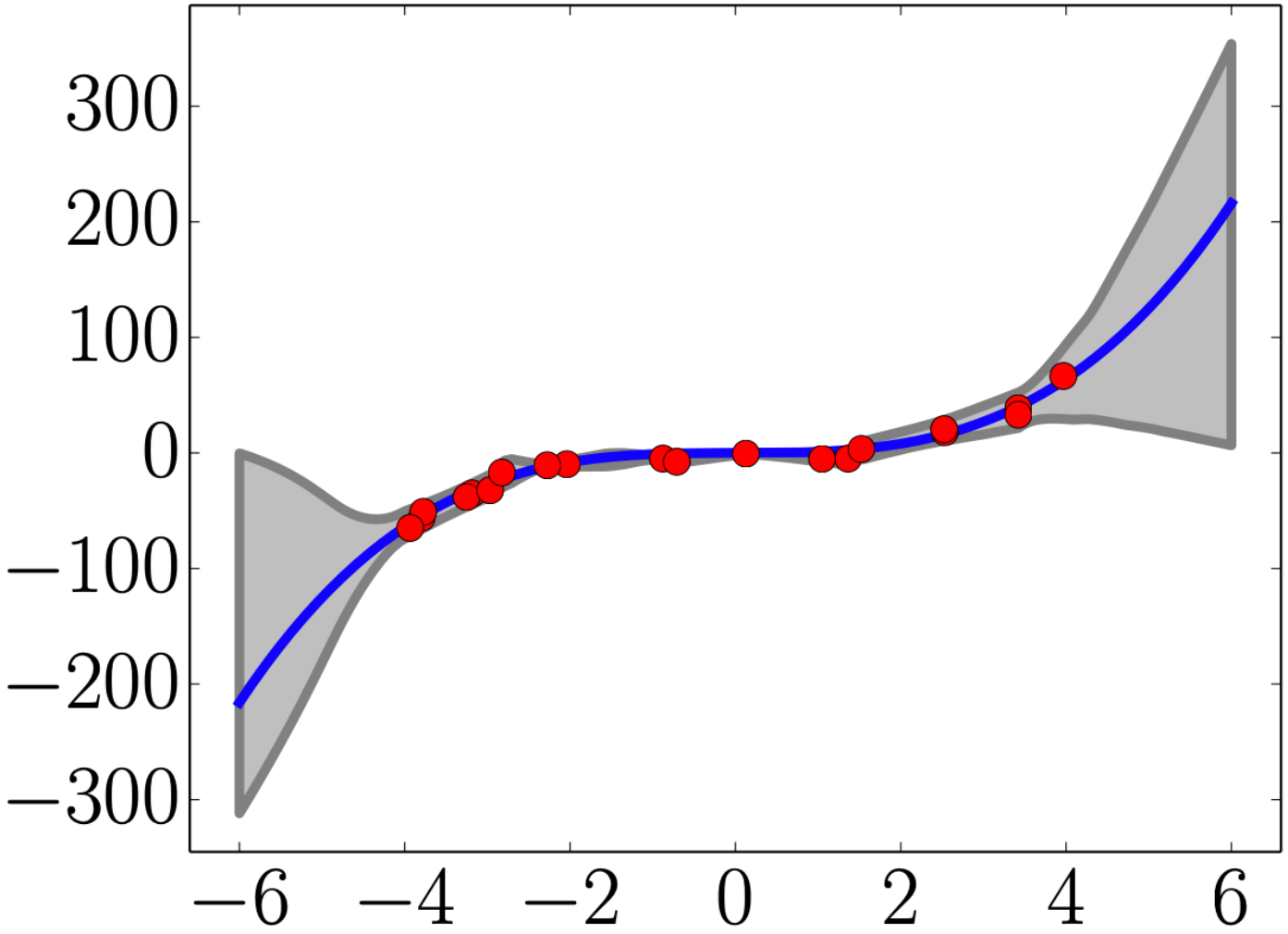
Training is started with a **random** weight value initialization → training the NN with **S**GD and the same data several times is usually ending in different local minima.

Image credit: https://wandb.ai/authors/loss-landscape/reports/Understanding-the-Effectivity-of-Ensembles-in-Deep-Learning--VmlldzoxODAxNjA

Blackboard Regression Ensembling with 3 solutions



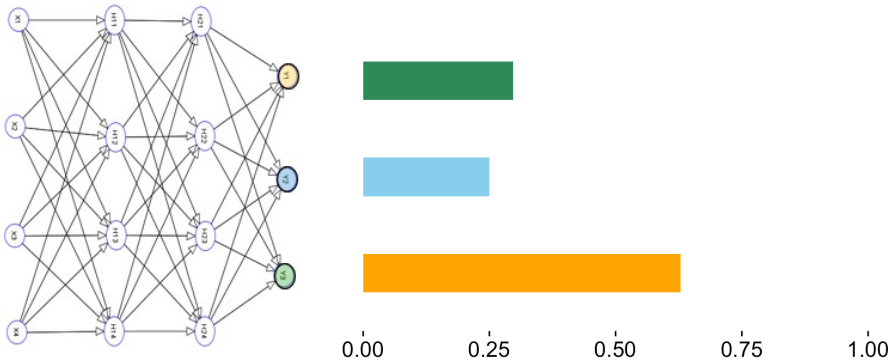
Ensemble of 5 Networks (Regression)

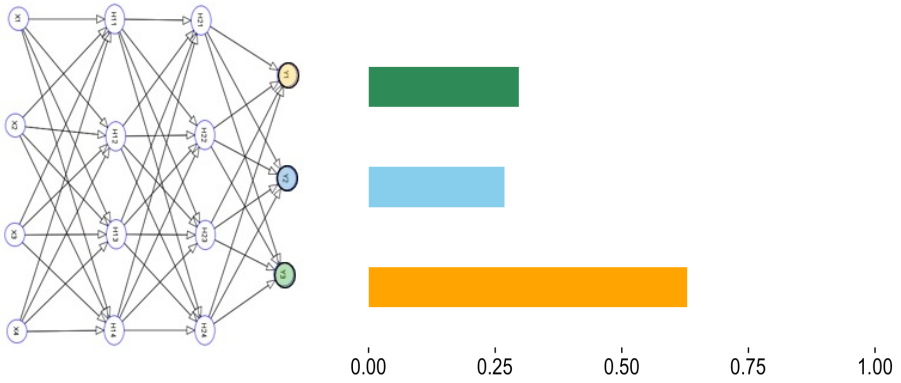


Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, *30*.

Classification

What happens if we haved trained the same CNN twice with the same data? Present example in training set.

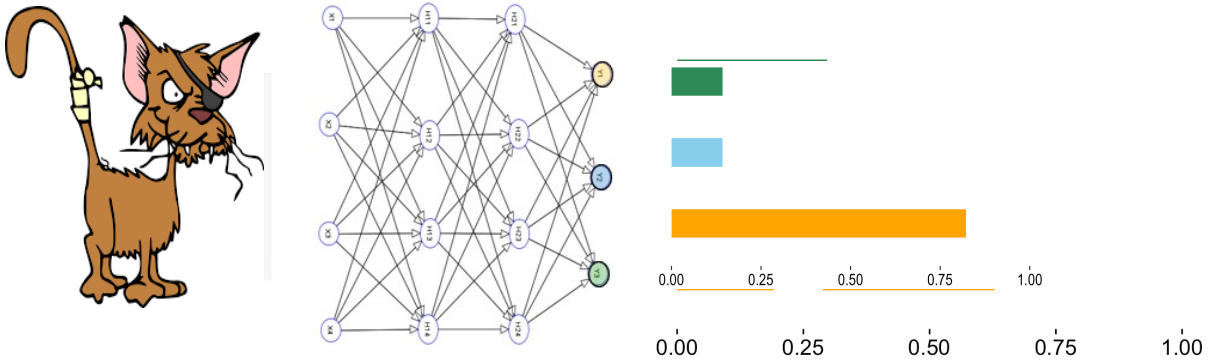


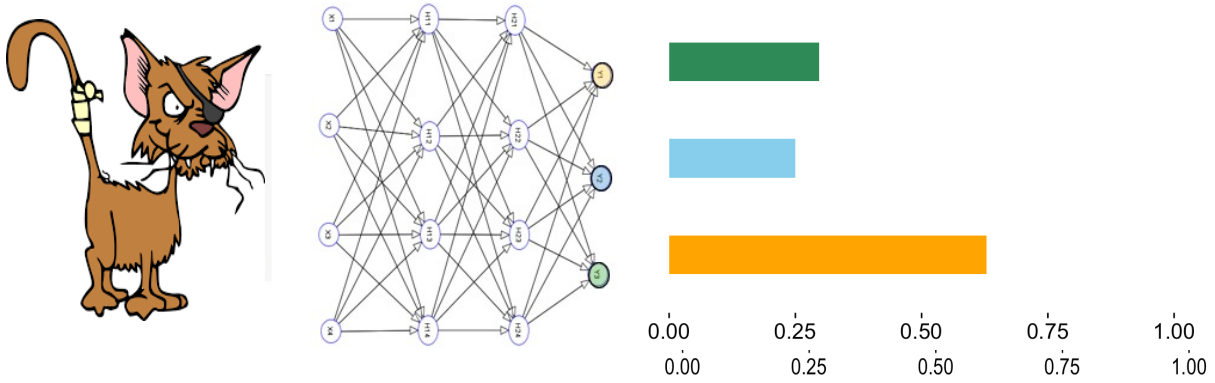


Small difference if example is know

Classification

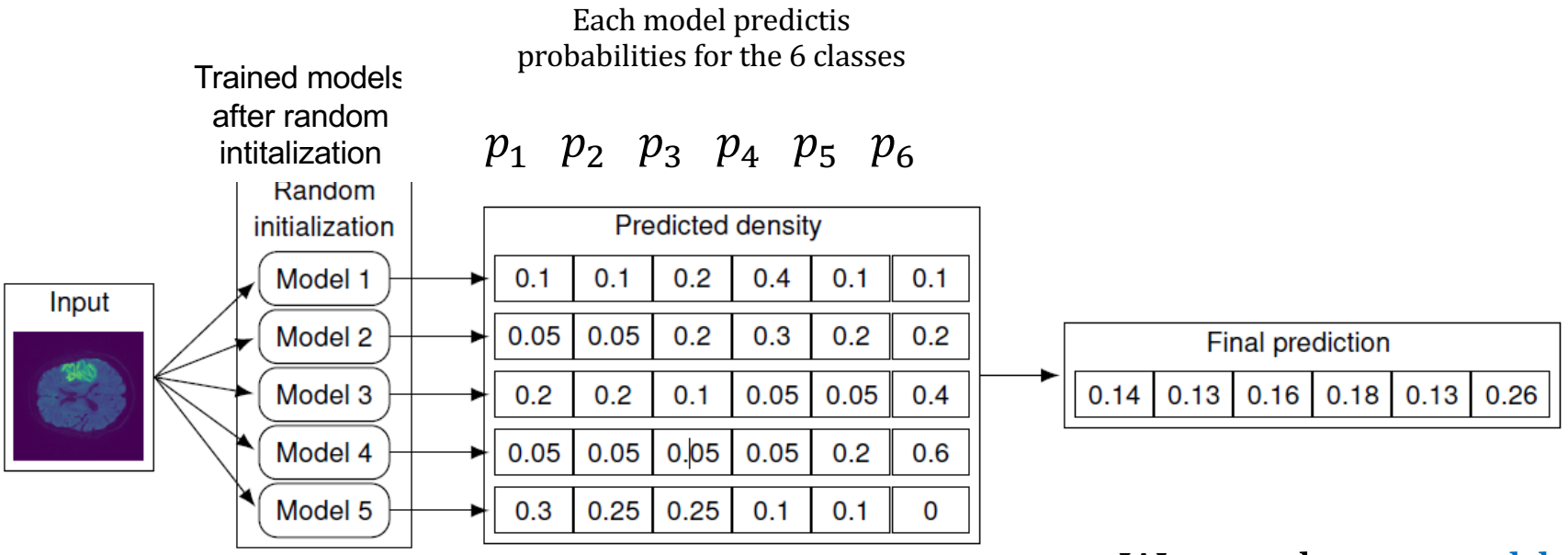
What happens if we haved trained the same CNN twice with the same data? But present OOD example.





Larger difference if not (cat).

Deep ensembling:
Train several NN models and average their predicitions



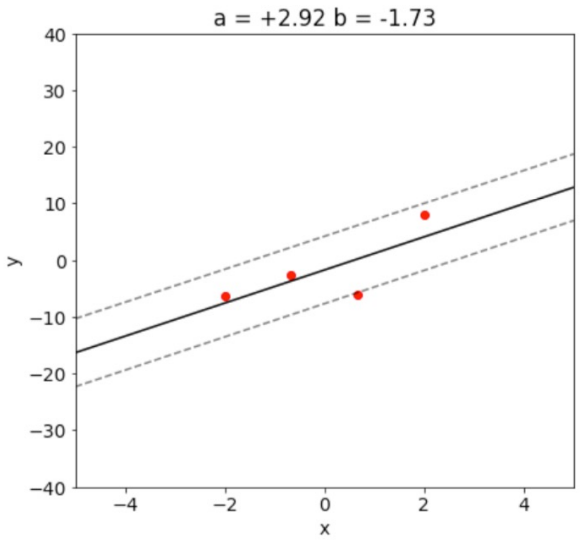
We get the ensemble predictions by averaging the probabilities that were predicted by the different models for each class

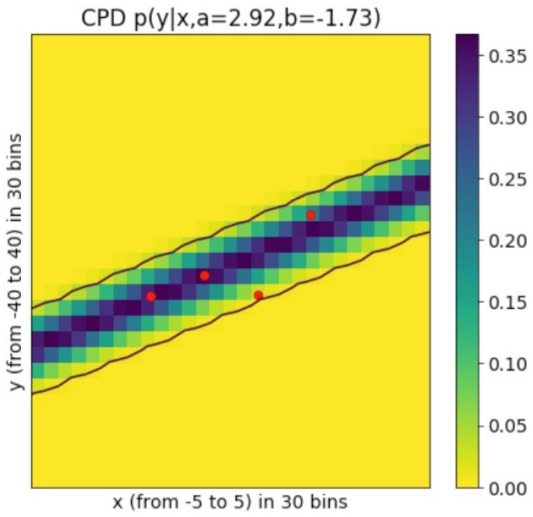
Added benefit. Ensembling also has better classification performance (e.g. NLL), then single average of the individual models.

Bayesian Neural Networks for epistemic uncertainty

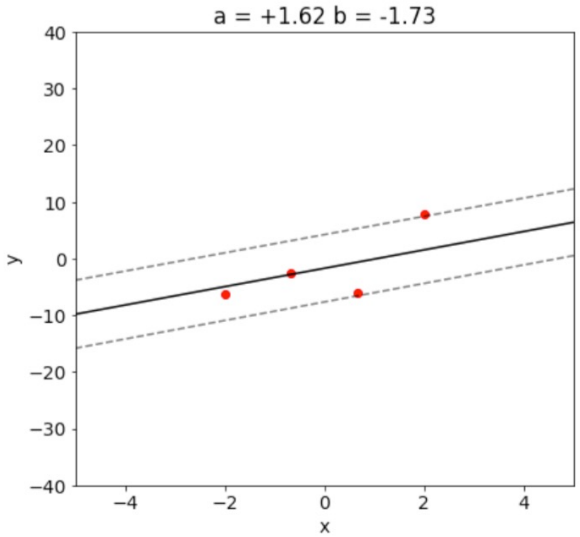
Motivation for Bayesian Model Averaging

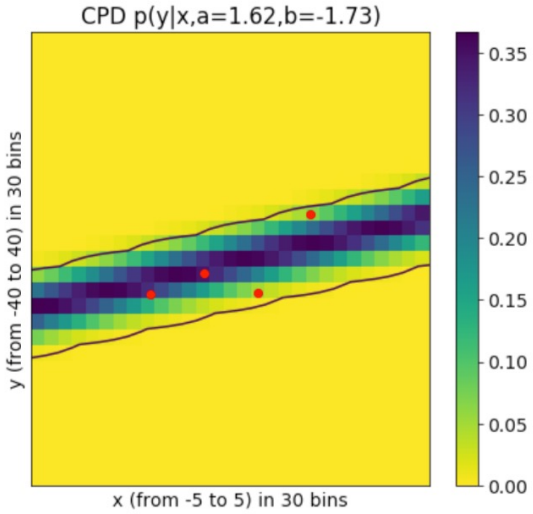
Let's look at good old linear regression to understand the gist of the Bayes idea. Assume $\sigma = 3$ to be known.





MaxLike Solution

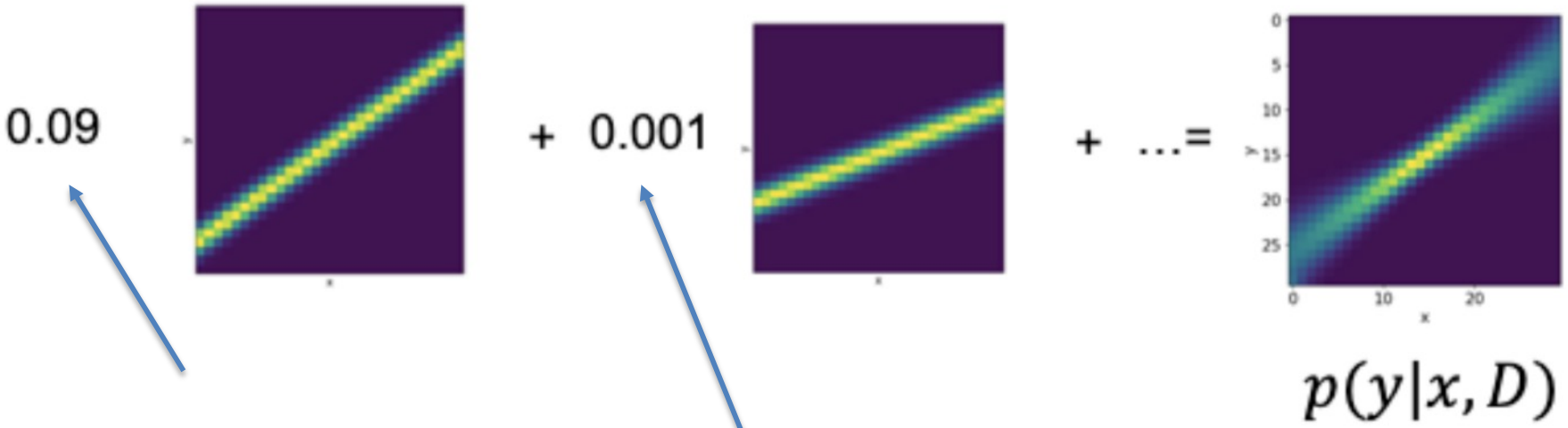




A bit off the MaxLike Solution

Combining different fits

Also take the other fits with different parameters into account and weight them



Highest weight for MaxLike

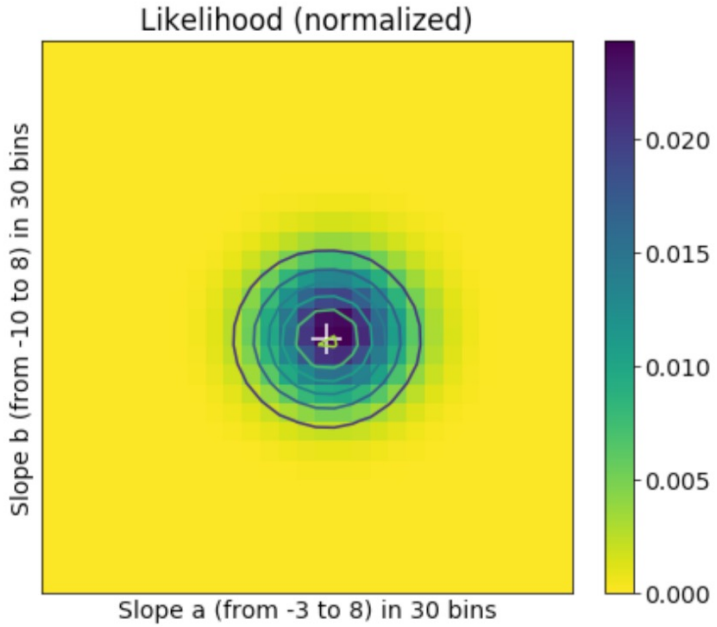
Other other less like solutions, should have a vote.

Question: How to get the weights (they need to sum up to one)?

Idea: use the (normalized) likelihood! $p_{\mathrm{norm}}((a,b)|D)$ with D is training data

Don't put all egg's in one Basket

- Also take other solutions for a,b into account



$$p_{norm}((a,b)|D) = \frac{p((a,b)|D)}{\sum p((a,b)|D)}$$

$$\mathrm{p}(y|x,D) = \sum_a \sum_b p(y|x,(a,b)) \cdot p_{norm}((a,b)|D)$$

Likelihood at 30x30 different positions of a and b. Normalized to be one.
https://github.com/tensorchiefs/ dl_book/blob/master/chapter_07 /nb_ch07_02.ipynb

Result

$$\mathrm{p}(y|x,D) = \sum_a \sum_b p(y|x,(a,b)) \cdot p_{norm}((a,b)|D)$$

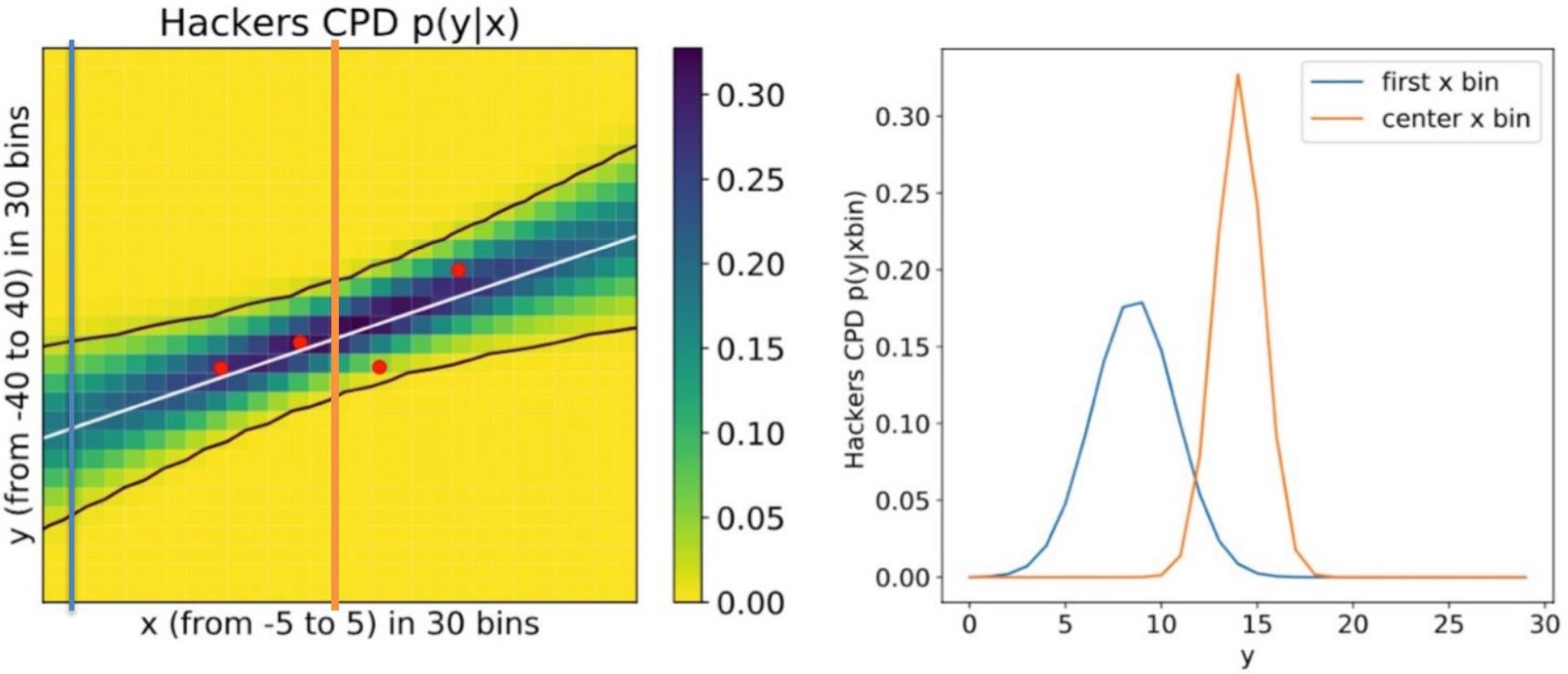


Figure 7.6 The predictive distribution for the Bayesian linear regression model, trained with the four data points shown on the left side by the color-code and on the right as conditional distribution at two different *x* positions. You can clearly see that the uncertainty gets larger when leaving the *x*-regions where there's data.

What have we done?

- We calculated
  - $\mathrm{p}(y|x,D) = \sum_a \sum_b p(y|x,(a,b)) \cdot p_{norm}((a,b)|D)$
- Or with the parameters $\theta = (a,b)$
  - $\mathrm{p}(y|x,D) = \sum_\theta p(y|x,\theta) \cdot p_{norm}(\theta|D)$
- We choose as weight for the weighted average
  - $p_{norm}(\theta|D) = \frac{p(\theta|D)}{\sum_\theta p(\theta|D)}$
  - It turns out that we where not the first to do so…

Bayesian statistics

- The Bayesian Theorem:
  - $p(A|B) = \frac{P(B|A)P(A)}{P(B)}$

- Applied to $A = \theta$ and $B = D$
  - Parameters θ of a model e.g. weights *w* of NN
  - Training data *D*

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} = \frac{p(D|\theta)p(\theta)}{\sum_\theta p(D|\theta)p(\theta)} \sim p(D|\theta)p(\theta)$$

  - $p(\theta|D)$ posterior see next slide
  - $p(D|\theta)$ likelihood our good old friend
  - $p(\theta)$ prior see next slide
  - $p(D)$ evidence just a normalization constant

The Bayesian Mantra (say it loud)

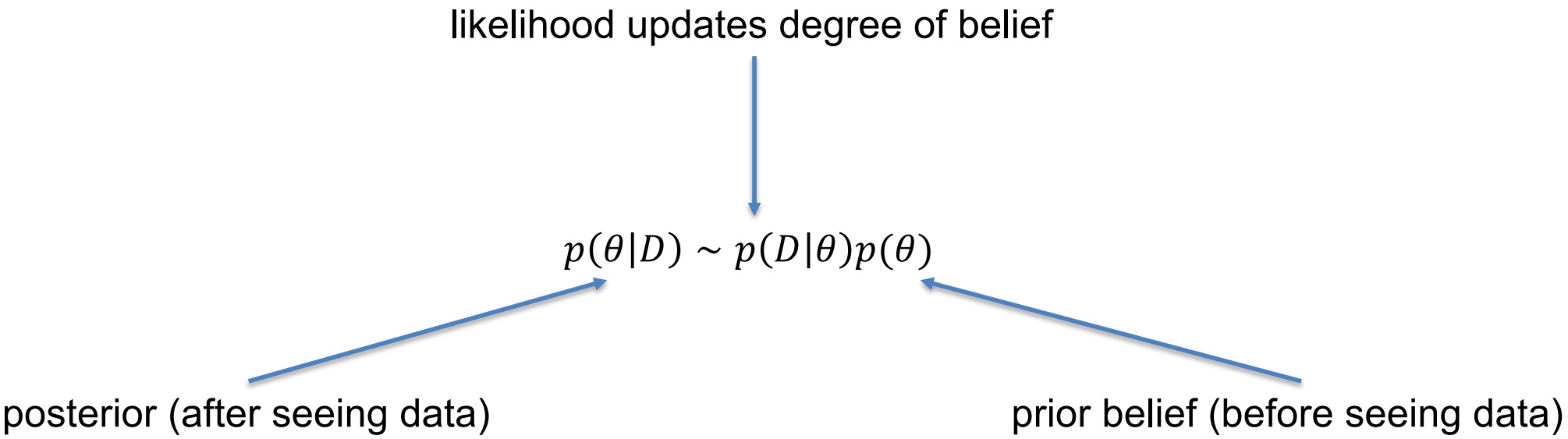
$$p(\theta|D) \sim p(D|\theta)p(\theta)$$

"The posterior is proportional to the likelihood, times the prior"

Interpretation updating the degree of belief

- Parameters θ are random variables, with distribution
  - We interpret the distribution as our degree of belief in a certain value

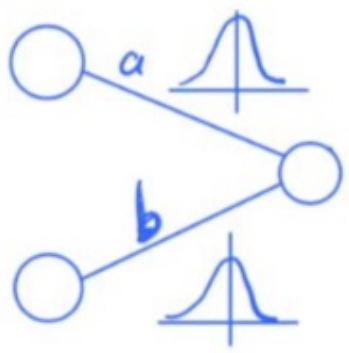
- The Bayes formula is seen as an update of our belief in the light of data



Revisiting the Hackers' example

- $\mathrm{p}(y|x,D) = \sum p(y|x,\theta) \cdot p_{\mathrm{norm}}(\theta|D)$
  - $p_{\mathrm{norm}}(\theta|D) = \frac{p(\theta|D)}{\sum p(\theta|D)}$
  - Bayes
  - $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} = \frac{p(D|\theta)p(\theta)}{\sum_\theta p(D|\theta)p(\theta)}$

- What, we did was to set the prior $p(\theta) = \mathrm{const}$
- Graphical Representation, of Bayesian linear regression:



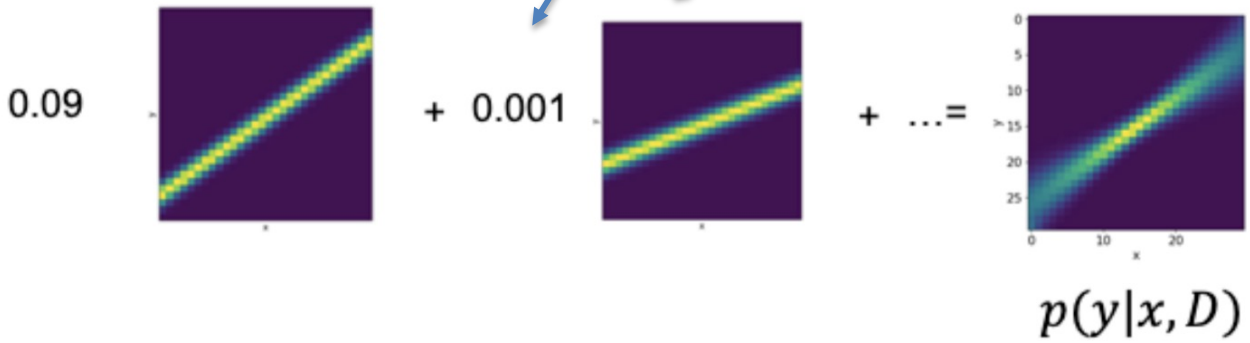
The weights are replaced by distribution.
E.g. for a

a --> $p(a|D)$ with $p(a)$ prior

Summary

Bayesian Model Averaging

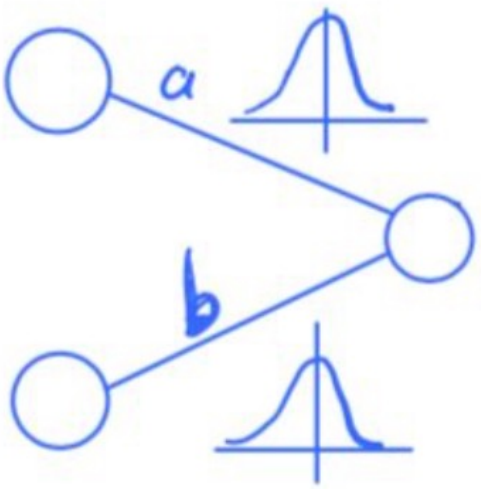
- $\mathrm{p}(y|x, D) = \int p(y|x, \theta) \cdot p\,(\theta|D)\; d\theta$

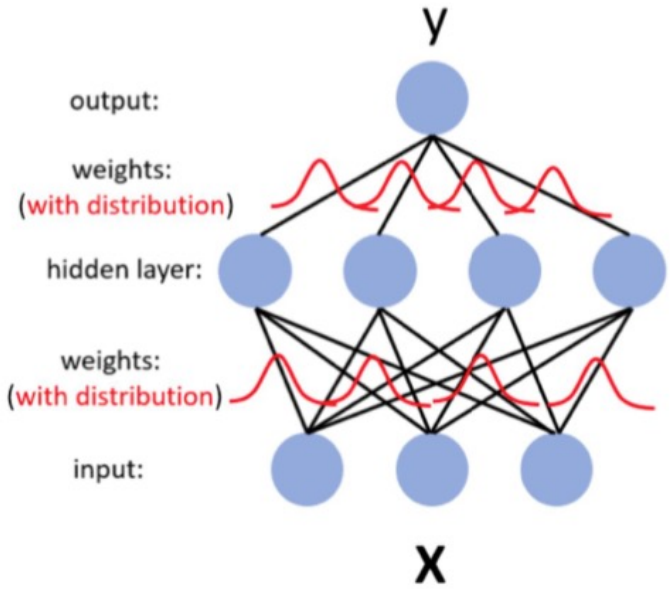


- Not just a single solution
  - "Marginalizing instead of optimization"
- Bayes usually achieves
  - Better uncertainty estimates
  - Better prediction performance

Bayesian Neural* Networks (BNN)

- Linear Regression with Gaussian prior and fixed sigma can be solved analytically





- Bayesian Neural Network cannot be solved analytically

*Don’t get confused Bayesian Network is something completely different

Approximations to BNN

- A BNN would require to calculate

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} = \frac{p(D|\theta)p(\theta)}{\sum_\theta p(D|\theta)p(\theta)}$$

- Usually no analytical solution exists (only for simple problems)
- It's possible to calculate $p(D|\theta)p(\theta)$ for a few values fast
- But calculate $\sum_\theta p(D|\theta)p(\theta)$ is impossible for high-dimension θ
  - Thing we need to calculate 10 values per dimension, then 10^dimensions evals

Approximations

- MCMC (only for very small NN)
  - Sample from $p(\theta|D)$ with knowledge of $p(D|\theta')p(\theta')$ / $p(D|\theta'')p(\theta'')$
- Variational Inference VI
  - Replace $p(a|D)$ with an approximation e.g. $N(\mu_a, \sigma_b)$ and determine μ_a,σ_a
- MC-Dropout
  - Dropout also during predictions (magically) samples from a posterior

Variational Inference

The principle of VI

- Replace $p(\theta|D)$ with $q_\lambda(\theta)$ (Variational Ansatz)
- Typically, independent Gaussian for each weight
  - $p(a|D) = q_{\mu,\sigma}(a)$

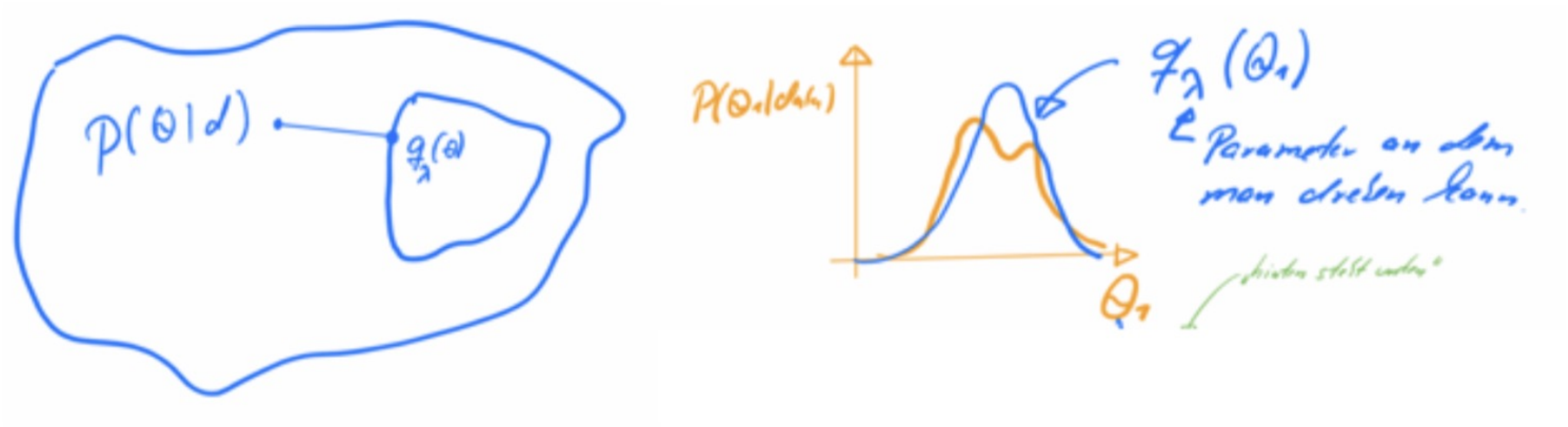
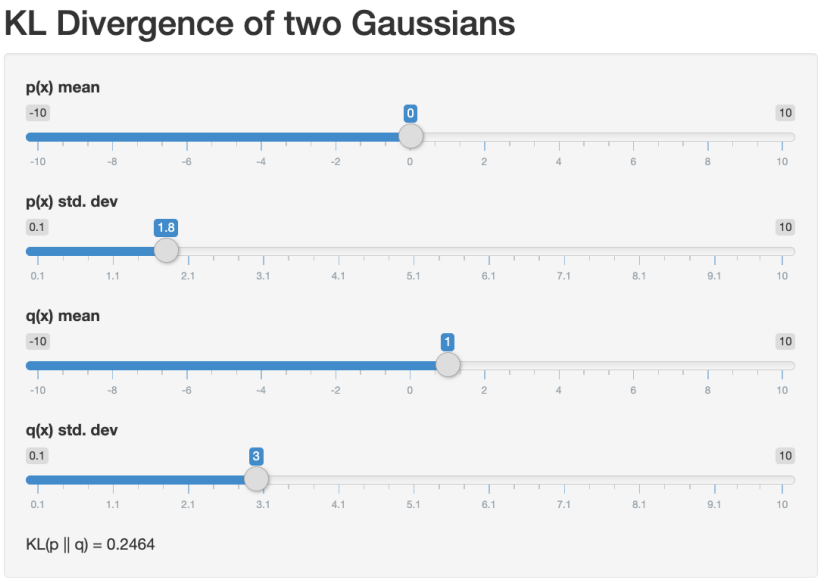


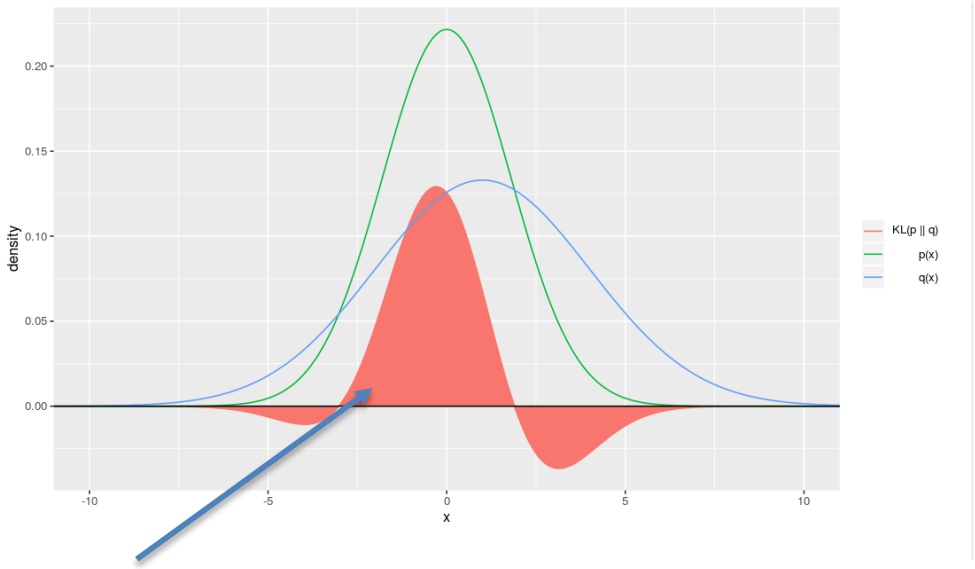
Figure 8.3 The principle idea of variational inference (VI). The larger region on the left depicts the space of all possible distributions, and the dot in the upper left represents the posterior $p(\theta|D)$ (corresponding to the dotted density on the right panel). The inner region depicts the space of possible variational distributions $q_\lambda(\theta)$. The optimized variational distribution $q_\lambda(\theta)$ (illustrated by the point in the inner loop in the left panel, corresponding to the solid density on the right panel) has the smallest distance to the posterior (shown by the dotted line on the right).

KL-Divergence

$$D_{KL}(p||q) = \int \log\left(\frac{p(y)}{q(y)}\right) p(y)dy$$

Very popular measure in Deep Learning to measure the “Distance” between 2 Distribution.





The Integral is positive

Demo of KL https://gnarlyware.com/blog/kl-divergence-online-demo/

Determining the variational parameters by minimizing KL-Divergence

- We want $q_\lambda(\theta)$ close to $p(\theta|D)$
- We start with KL(q||p) (does only work this direction)

$$\mathrm{KL}(q_\lambda(\theta) \parallel p(\theta|D)) = \int q_\lambda(\theta) \log\left(\frac{q_\lambda(\theta)}{p(\theta|D)}\right) d\theta$$

$$= \log(p(D)) - \underbrace{(\mathbb{E}_{\theta\sim q_\lambda}(\log(p(D|\theta))) - \mathrm{KL}(q_\lambda(\theta) \parallel p(\theta)))}_{\mathrm{ELBO}(\lambda)}$$

- After a bit of math (using Bayes Rule)
  - The math is quite similar to the one used for VAE
- ELBO Evidence Lower Bound is maximized (minimize –ELBO)

Derivation of ELBO [just for reference]

(continued)

Let's write the definition of the KL divergence and indicate the data with *D*. If you can't remember how to write the KL divergence of, say, two functions *f* and *g*, maybe this rule of thumb will help: it's "back alone down," meaning that the second function *g* in $\mathrm{KL}[f(\theta)\|g(\theta)]$ will be alone and in the denominator. The following definition of the KL divergence shouldn't surprise you too much:

$$\mathrm{KL}[q_\lambda(\theta)\|p(\theta|D)] = \int q_\lambda(\theta)\log\frac{q_\lambda(\theta)}{p(\theta|D)}d\theta$$

$p(\theta|D)$ is the second function ("in the back") and, thus, only appears once ("alone") in the integral and, further, it is in the denominator ("down"). (You could also look up the definition of the KL divergence.) Now, there are some algebraic manipulations ahead. Feel free to follow the steps using a pen and paper. The first thing we do is to use the definition of the conditional probability, $p(\theta|D) = p(\theta, D)/p(D)$:

$$\mathrm{KL}[q_\lambda(\theta)\|p(\theta|D)] = \int q_\lambda(\theta)\log\frac{q_\lambda(\theta)}{p(\theta, D)/p(D)}d\theta$$

Then we use the calculation rules of the logarithm $\log(A \cdot B) = \log(A) + \log(B)$ and $\log(B/A) = -\log(A/B)$ to split the integral into two parts:

$$\mathrm{KL}[q_\lambda(\theta)\|p(\theta|D)] = \int q_\lambda(\theta)\log p(D)d\theta - \int q_\lambda(\theta)\log\frac{p(\theta, D)}{q_\lambda(\theta)}d\theta$$

Because log *p*(*D*) doesn't depend on *θ*, we can put it before the integral:

$$\mathrm{KL}[q_\lambda(\theta)\|p(\theta|D)] = \log p(D)\cdot\int q_\lambda(\theta)d\theta - \int q_\lambda(\theta)\log\frac{p(\theta, D)}{q_\lambda(\theta)}d\theta$$

And because $q_\lambda(\theta)$ is a probability density and for all probability densities the integral is one, we have $\int q_\lambda(\theta)d\theta = 1$:

$$\mathrm{KL}[q_\lambda(\theta)\|p(\theta|D)] = \log p(D) - \int q_\lambda(\theta)\log\frac{p(\theta, D)}{q_\lambda(\theta)}d\theta$$

The first term doesn't depend on the variational parameter *λ*; therefore, all you need to minimize is $-\int q_\lambda(\theta)\log\frac{p(\theta, D)}{q_\lambda(\theta)}d\theta$. The optimal value *λ* is thus:

$$\lambda^* = \mathrm{argmin}\left\{-\int q_\lambda(\theta)\log\frac{p(\theta, D)}{q_\lambda(\theta)}d\theta\right\}$$

Now, let's arrange it into the form of equation 8.1 with $p(\theta, D) = p(\theta|D)\cdot p(\theta)$

$$\lambda^* = \mathrm{argmin}\left\{-\int q_\lambda(\theta)\log\frac{p(D|\theta)\cdot p(\theta)}{q_\lambda(\theta)}d\theta\right\}$$

and with the calculus rules for the logarithm:

$$\lambda^* = \mathrm{argmin}\left\{\int q_\lambda(\theta)\log\frac{q_\lambda(\theta)}{p(\theta)}d\theta - \int q_\lambda(\theta)\cdot\log p(D|\theta)d\theta\right\}$$

The first term is the definition of the KL divergence between the variational and the prior distribution: $\mathrm{KL}[q_\lambda(\theta)\|p(\theta)]$ (remember "back alone down"). The second term is the definition of the expectation of the function $\log p(D|\theta)$. So finally, we have

$$\lambda^* = \mathrm{argmin}\left\{\mathrm{KL}[q_\lambda(\theta)\|p(\theta)] - E_{\theta\sim q_\lambda}[\log(p(D|\theta))]\right\}$$

and are done with deriving the expression in equation 8.1. Wasn't so hard, was it?

Intuition of the optimization

Distance of prior to variational approximation (regularization)

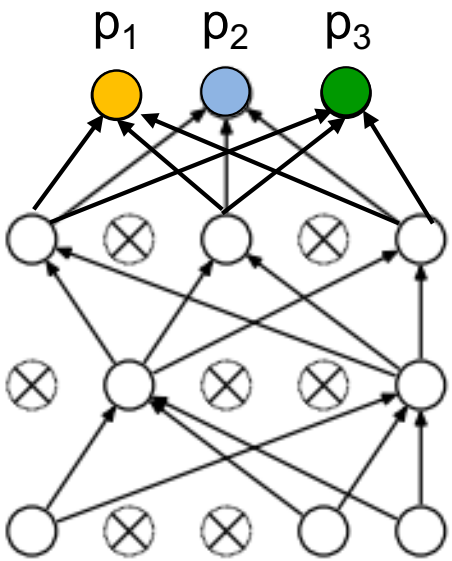
$$\lambda^* = argmin\{KL[q_\lambda(\theta)||p(\theta)] - E_{\theta\sim q_\lambda}[log(p(D|\theta)]\}$$

NLL of trainings data D, now averaged over different weights from variational distribution

- Typical: $q_\lambda(\theta)$ and $p(\theta)$ Gaussian → Analytical Approximation possible
- $E_\theta[\log(p(D|\theta)]$ estimated by the mean of a few samples
- Interpretation
  - Tradeoff of good fit (low NLL) and regularization small KL to prior.
    - The KL is ∝ Number of weights
    - The NLL terms is ∝ number of datapoints
  - The more Data the less important the priors.

MC Dropout as Bayesian Approximation

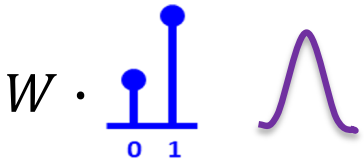
MC Dropout and Bayesian Neural Networks

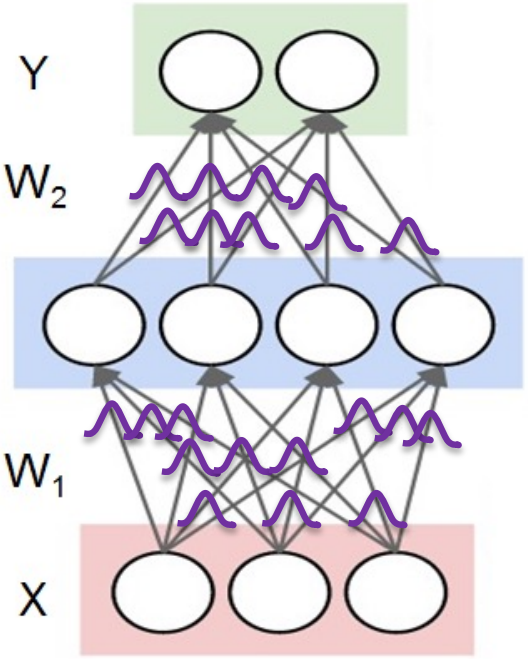


MC Dropout
Randomly drop nodes in each run
→ Ususally done during training

Dropout in test time

Yarin Gal* (2015): we learn a whole weight distribution



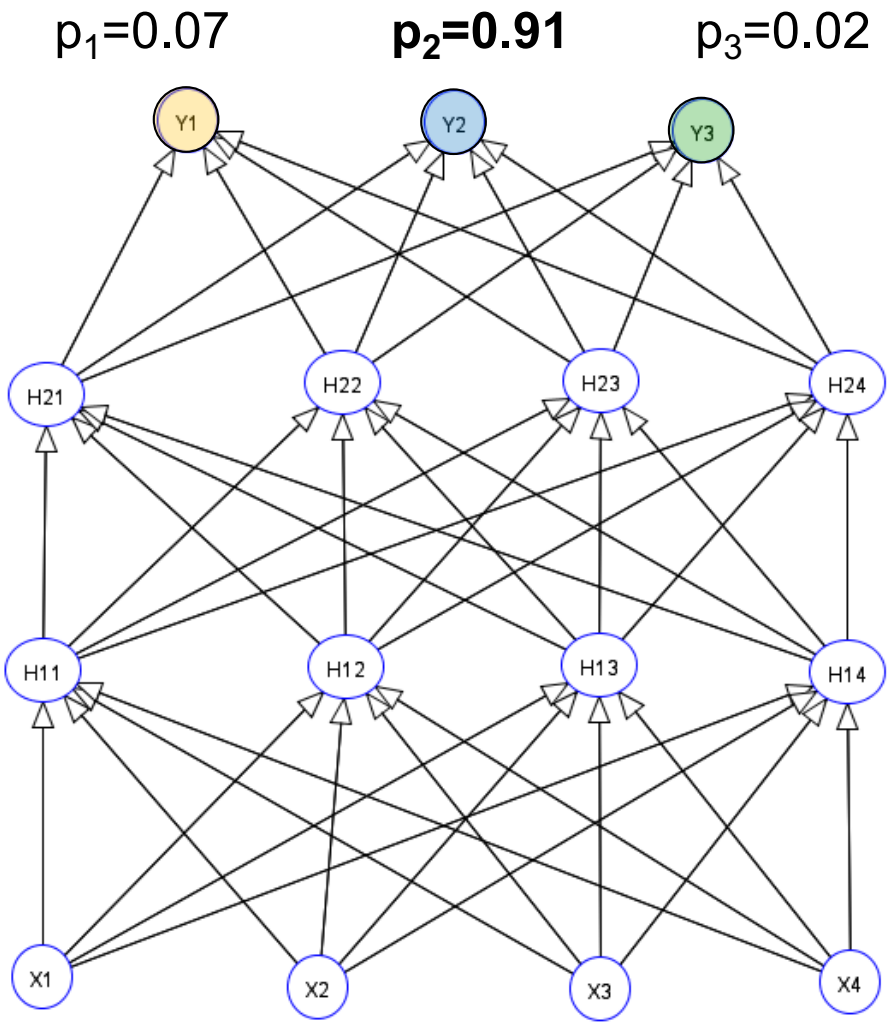


Bayesian NN

→ We should sample from weight distribution during test time

*Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning https://arxiv.org/abs/1506.02142

No MC Dropout

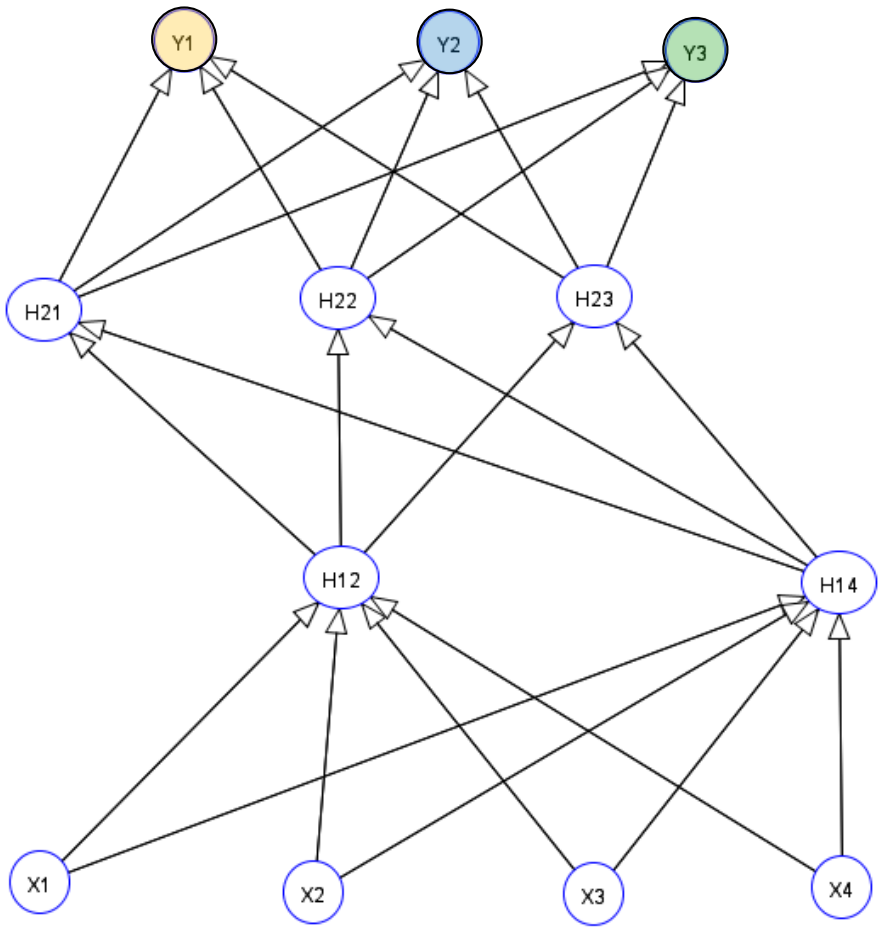


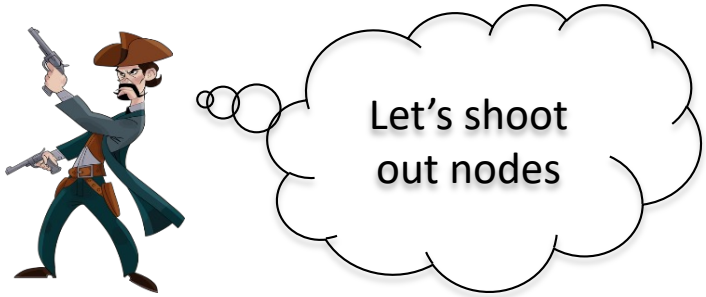
Probability of predicted class: $\mathbf{p_{max}}$

Input: image pixel values

Use dropout at test time: Run 1

p_1=0.08 **p_2=0.89** p_3=0.03

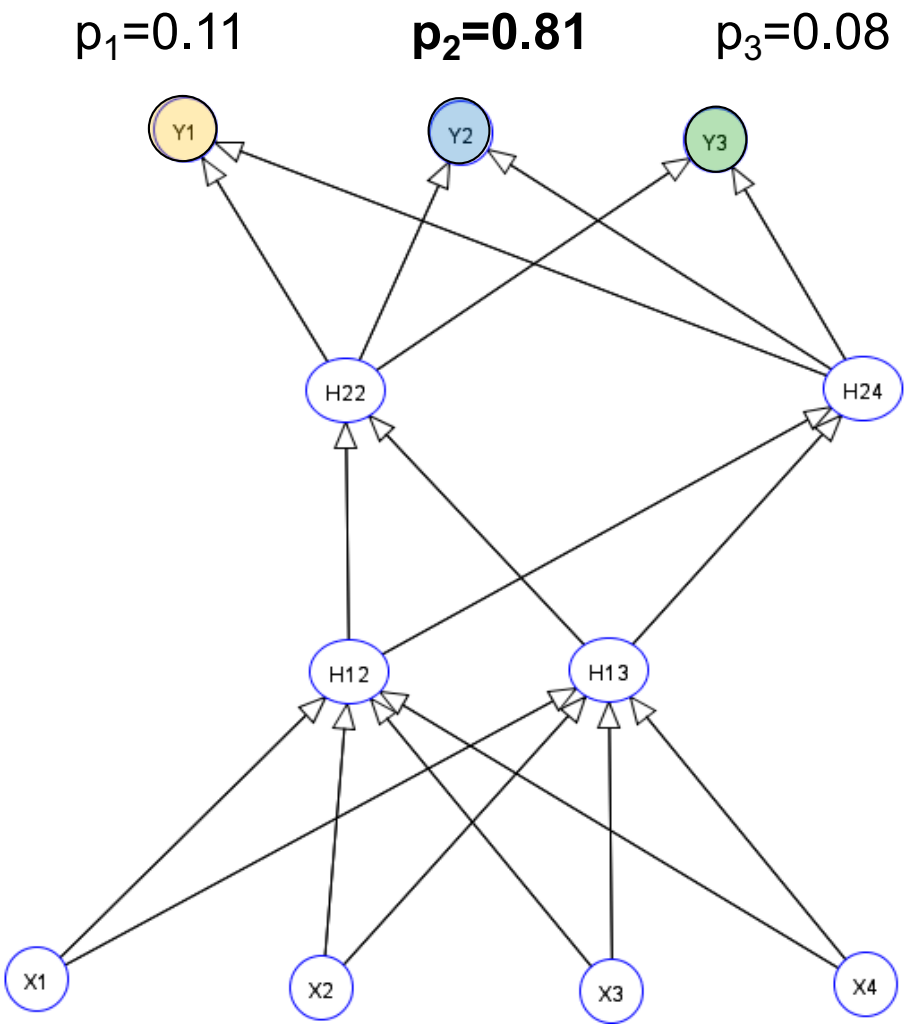


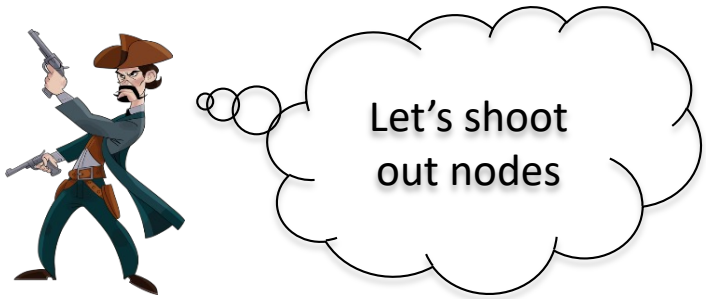


Stochastic dropout of units

Same input image

Use dropout at test time: Run 2

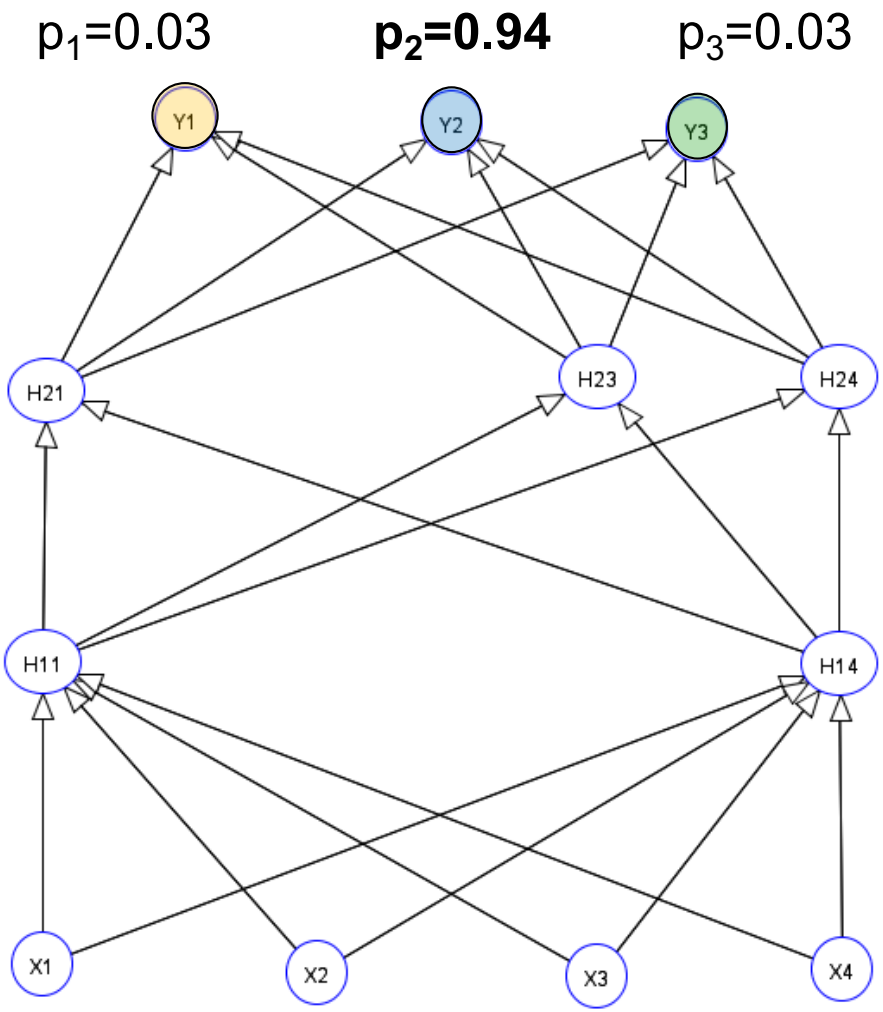


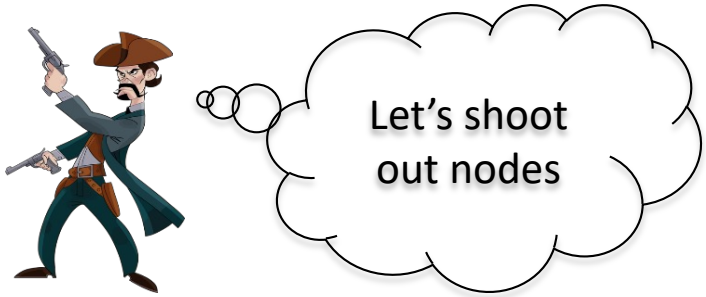


Stochastic dropout of units

Same input image

Use dropout at test time: Run 3



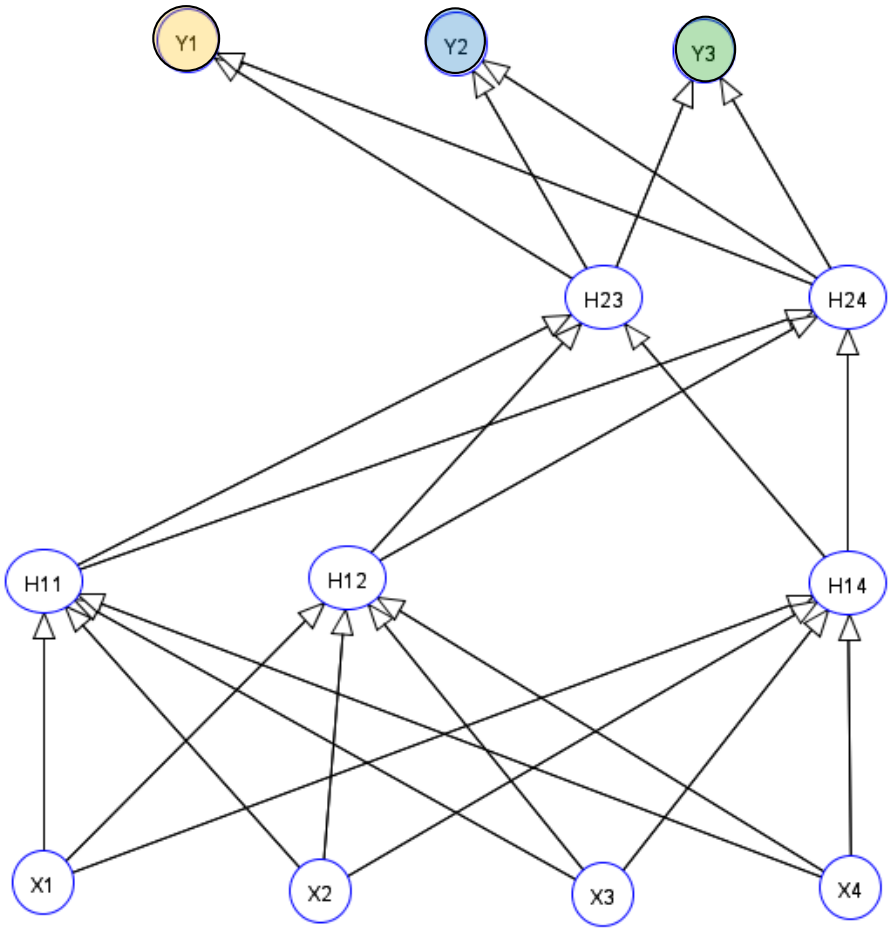


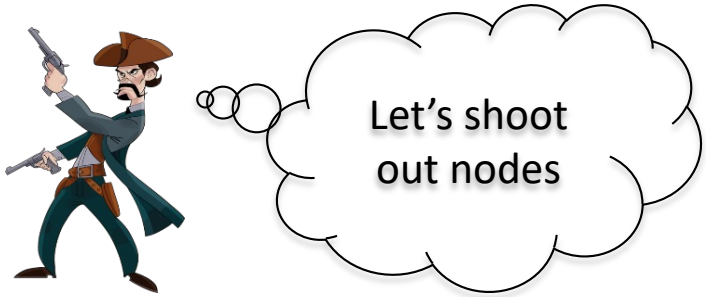
Stochastic dropout of units

Same input image

Use dropout at test time: Run 4

p_1=0.16 **p_2=0.78** p_3=0.06



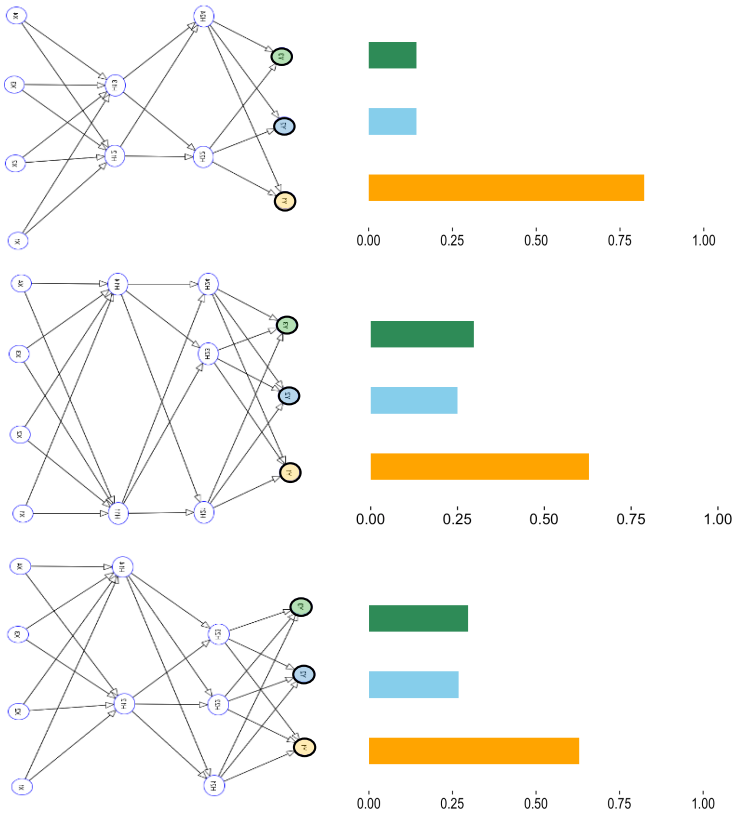


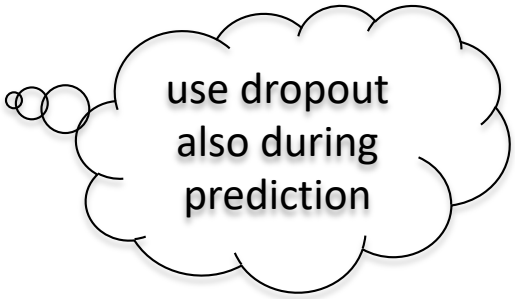
Stochastic dropout of units

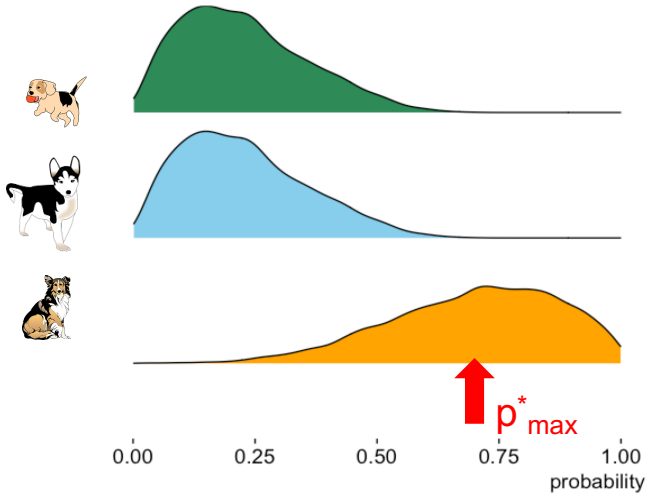
Same input image

MC probability prediction

Many Dropout Runs in forward pass





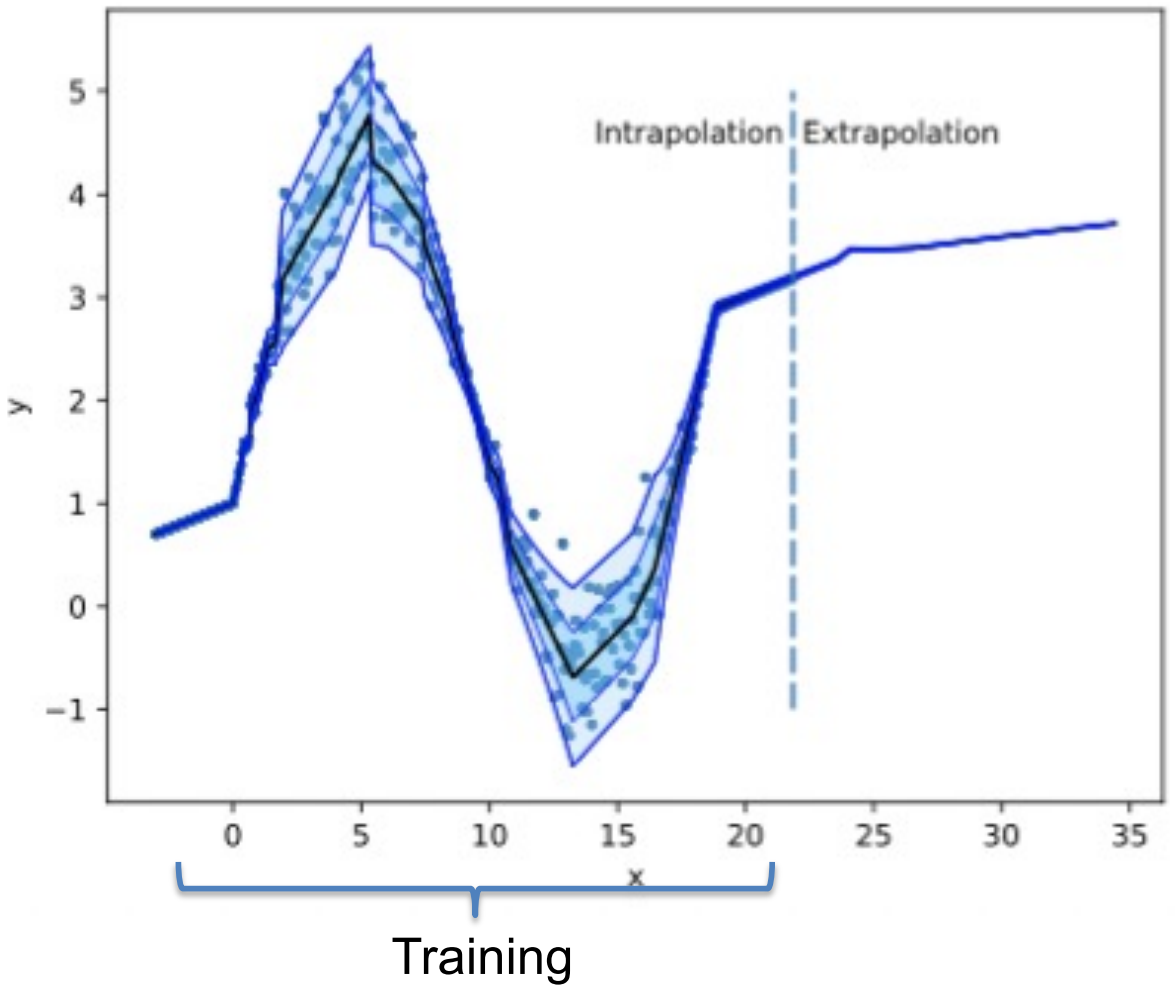


CNN predicts class
“collie”
but with high uncertainty

Remark: Mean of marginal give components of mean in multivariate distribution.

Case Study

Task


Intrapolation
Extrapolation
y
x
5
4
3
2
1
0
−1
0
5
10
15
20
25
30
35
Training

Comparison of different methods

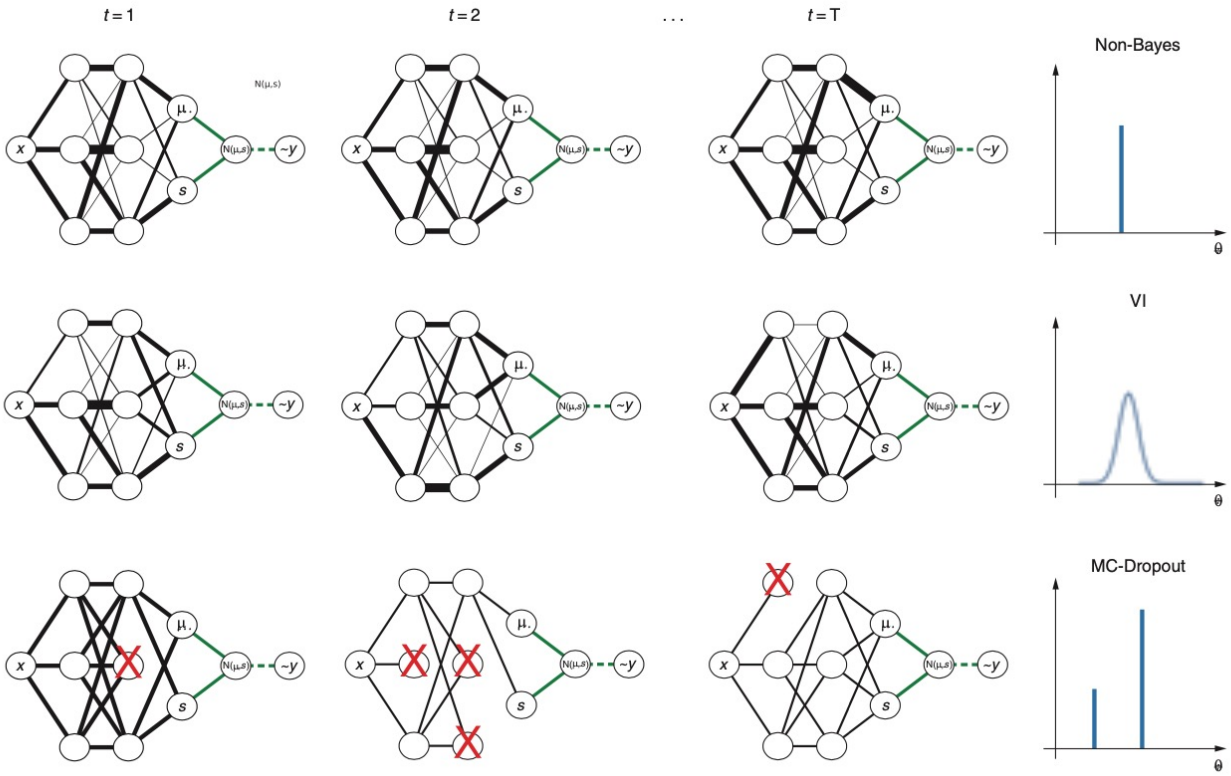


Figure 8.13 Sketch of the sampling procedure of three models used in the regression case study. In the last column (on the right), one edge of the network is picked, showing its distribution. The remaining columns show realizations of different networks for T different runs. In each run, the input x_i is the same, and the values of the edges are sampled according to their distributions. In the upper row, you see the non-Bayesian approach, where all realized NN are the same. In the middle, you see the VI-Bayesian approach, where the values of the edges are sampled from Gaussians. On the bottom right, the MC dropout approach, where the values of edges come from binary distributions. For animated versions, see **https://youtu.be/mQrUcUoT2k4** (for VI); **https://youtu.be/0-oyDeR9HrE** (for MC dropout); and **https://youtu.be/FO5avm3XT4g** (for non-Bayesian).

MC
https://www.youtube.com/watch?v=0-oyDeR9HrE&feature=youtu.be

VI
https://www.youtube.com/watch?v=mQrUcUoT2k4&feature=youtu.be

Results

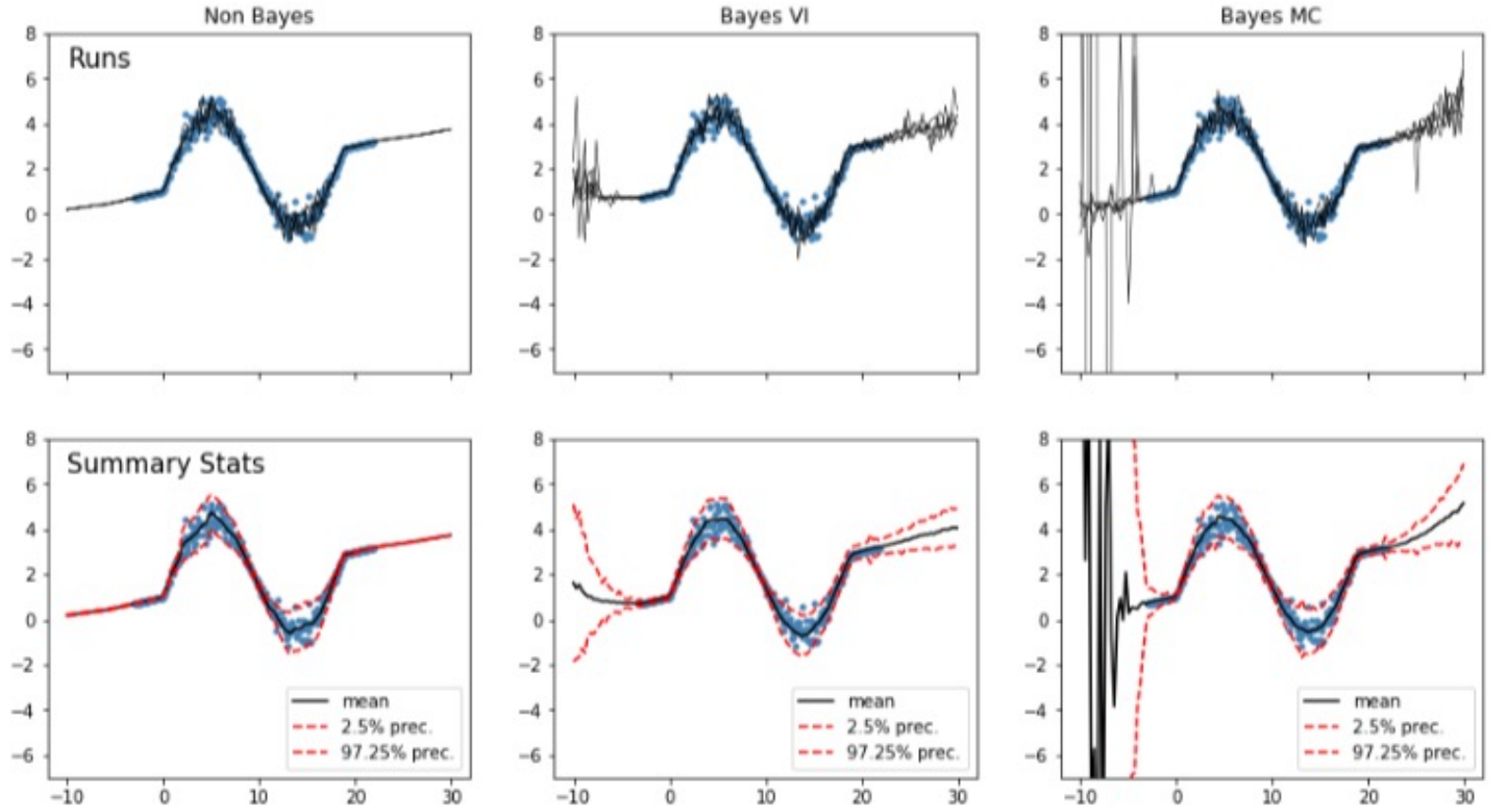


Figure 8.13 Predictive distributions. The solid lines in the top row show five samples from the outcome distribution for the three models. The second row shows summary statistics: the solid line represents the mean, and the lower and upper dashed lines depict the upper and lower borders of the 95% prediction interval.

Summary

- Deep Learning only models the aleatory uncertainty
- If $P(Train) \neq P(Test)$ epistemic uncertainty is needed

- Model averaging
  - Ensembling over different runs
    - Simple practical approach
    - Covers multiple modes in the loss landscape
  - Bayes
    - Theoretically well-founded
    - Requires approximations; choice of priors not clear
    - VI and Dropout usually in a single mode