

Machine Intelligence:: Deep Learning

Week 7

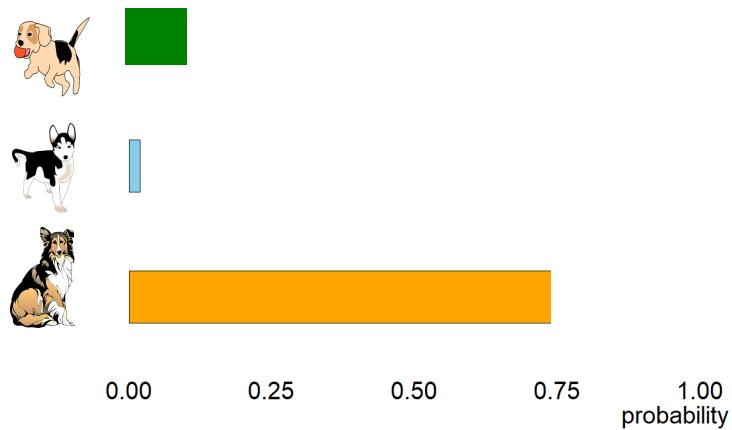
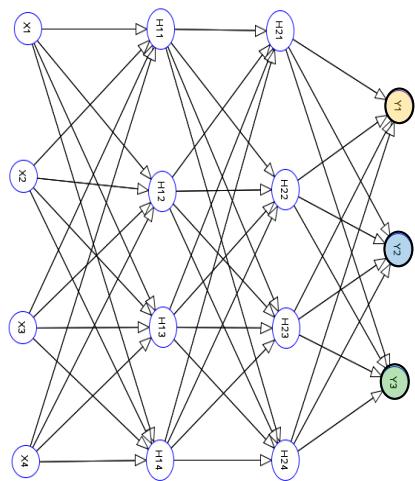
Beate Sick, Jonas Brändli, Oliver Dürr

Ensembling approaches for improving the performance and uncertainty estimates of NN models by taking into account the epistemic uncertainty.

Outline:

- Uncertainty in DL models
 - Epistemic uncertainty
 - Aleatoric uncertainty
- Approaches to take algorithmic and/or epistemic uncertainty into account:
 - Deep Ensembling
 - MC Dropout
- Bayes
 - Theoretical Background for all
 - Variational Inference (possible next week)

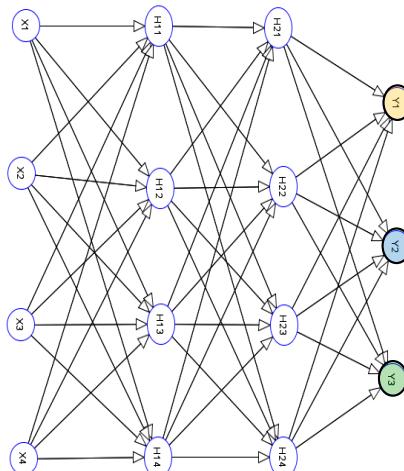
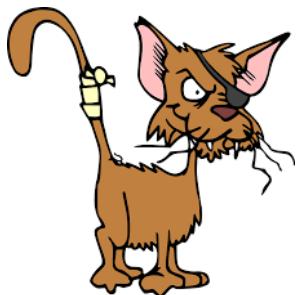
Probabilistic CNNs as we know them so far



CNNs yield high accuracy and calibrated (=unbiased) probabilities, but...

How good do we know probabilistic CNNs?

What happens if we present a novel class to the CNN?

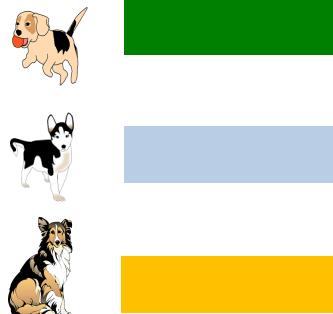
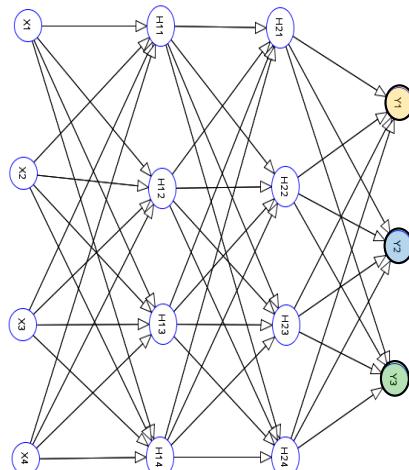
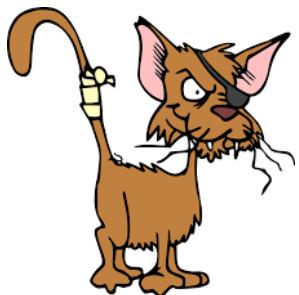


You might expect:

What do you
expect?

How good do we know probabilistic CNNs?

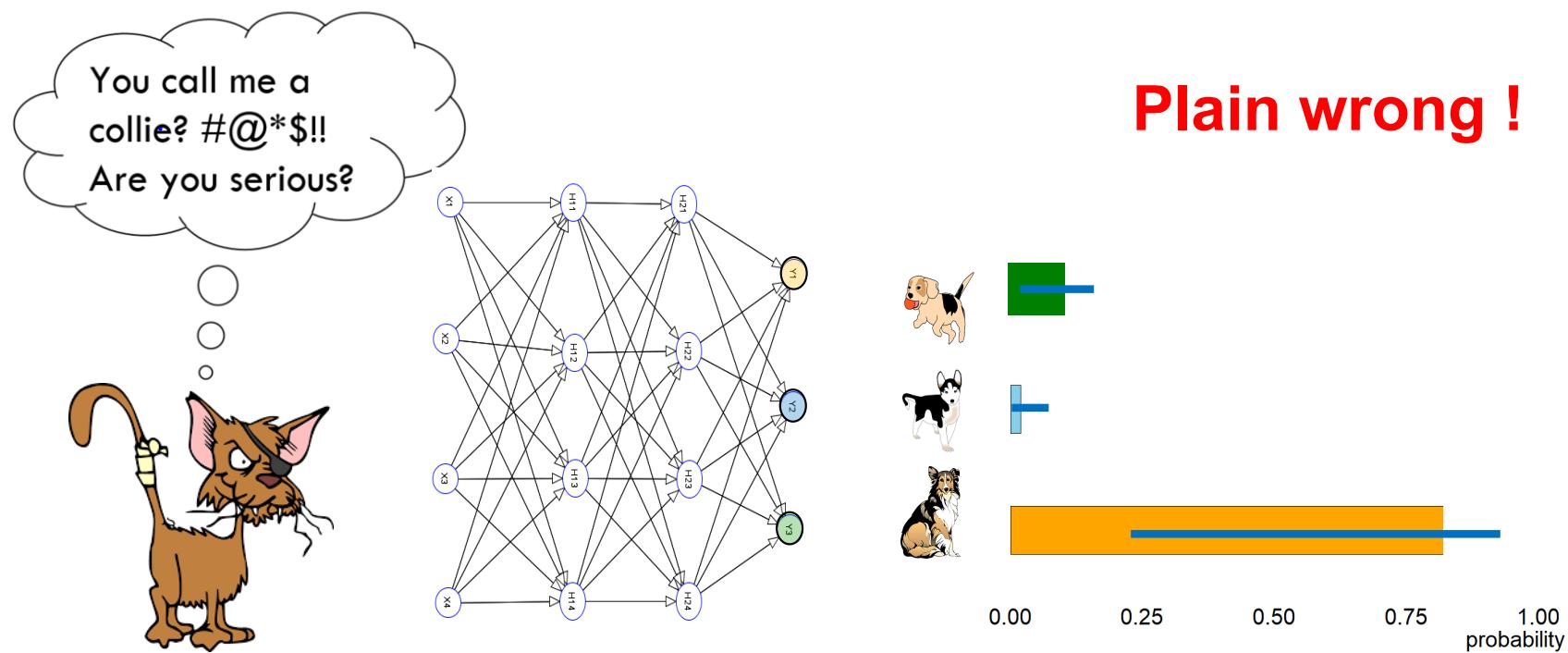
What happens if we present a novel class to the CNN?



You might expect:

A non-Bayesian NN cannot ring the alarm

What happens if we present a novel class to the CNN?



We need some error bars!

Importance to detect OOD



- Current DL Systems bad in out of distribution OOD situations
- Application need at least to detect OOD situations

Extrapolation: Causes Problems

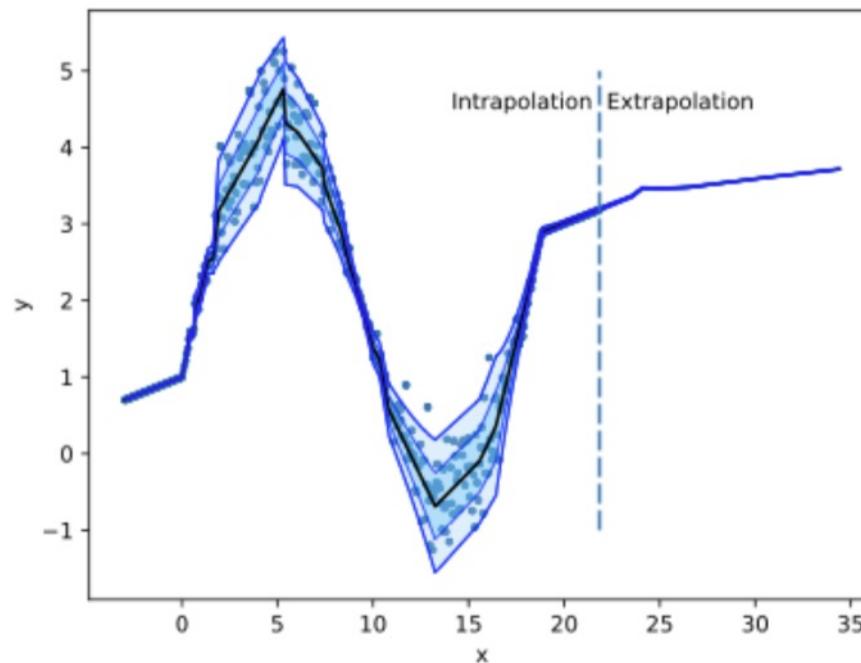
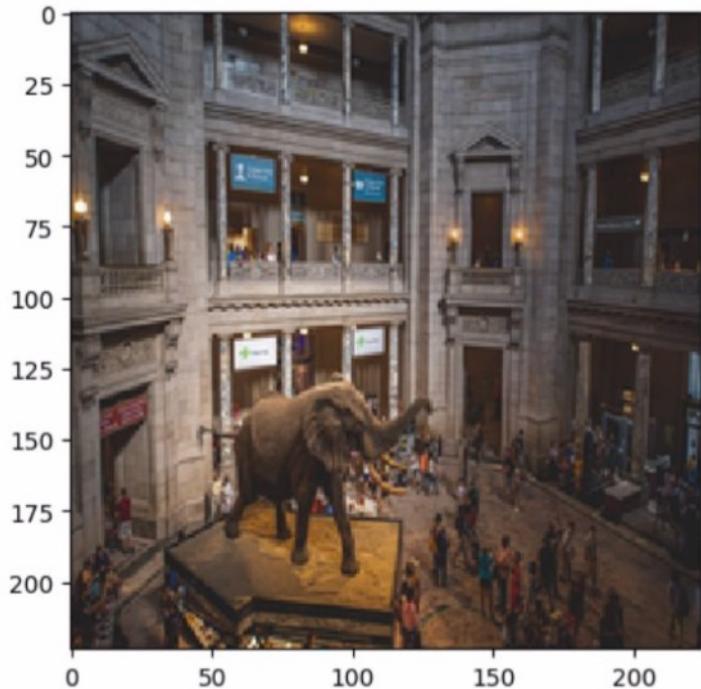
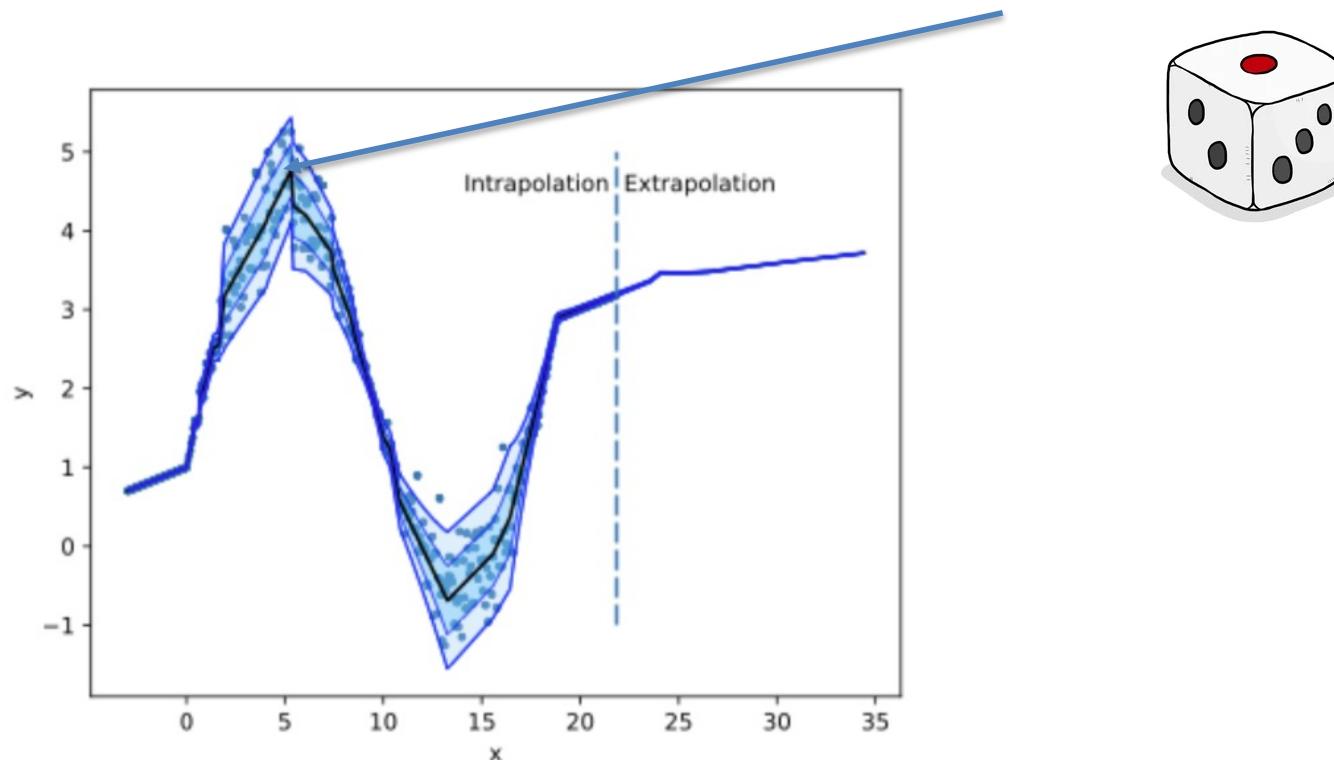


Figure 7.2 Bad case of DL. The high performant VGG16-CNN trained on ImageNet data fails to see the elephant in the room. The five highest-ranked class predictions of the objects in the image are horse_cart, shopping_cart, palace, streetcar, gondola; the elephant is not found! This image is an extrapolation of the training set. In the regression problem on the right side of the dashed vertical line, there's zero uncertainty in the regions where there's no data (extrapolation).

Definition Aleatory vs. Epistemic Uncertainty

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



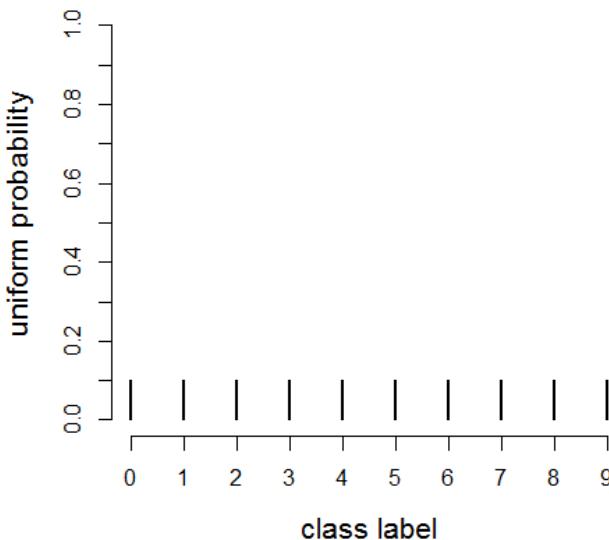
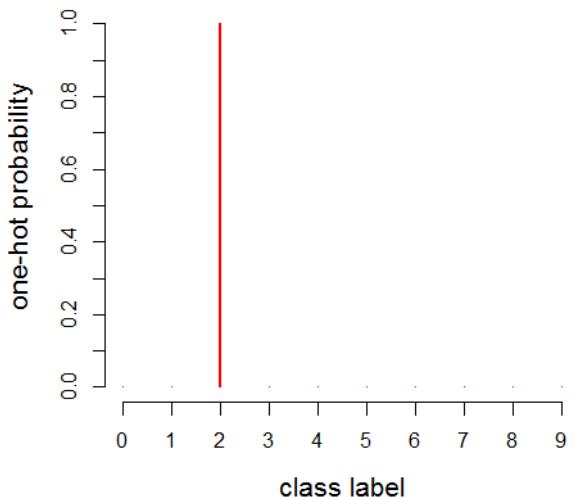
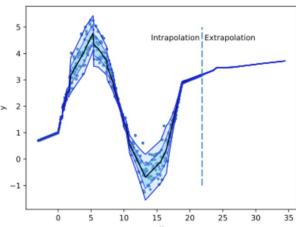
- *Aleatoric* uncertainty is due to the uncertainty in the data.
- The uncertainty when leaving the ‘known ground’ is called *epistemic* uncertainty.

We can model this uncertainty when we take the uncertainty with which we know the weights (called parameter uncertainty) into account. This can be done with Bayesian reasoning or phenomenological.

Aleatoric Uncertainty



- Regression
 - The spread of the Data.
- Classification
 - Spread?



$$H(P) = - \sum_i p_i \cdot \log(p_i)$$

One has $H \approx 2.3$ and one $H=0$. Which one

How to model the epistemic uncertainty?

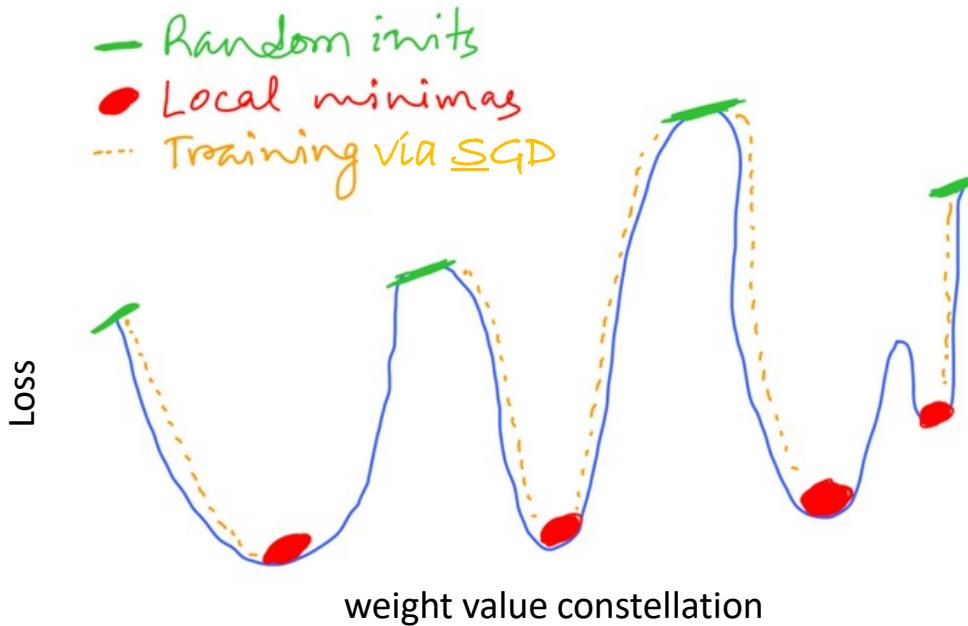
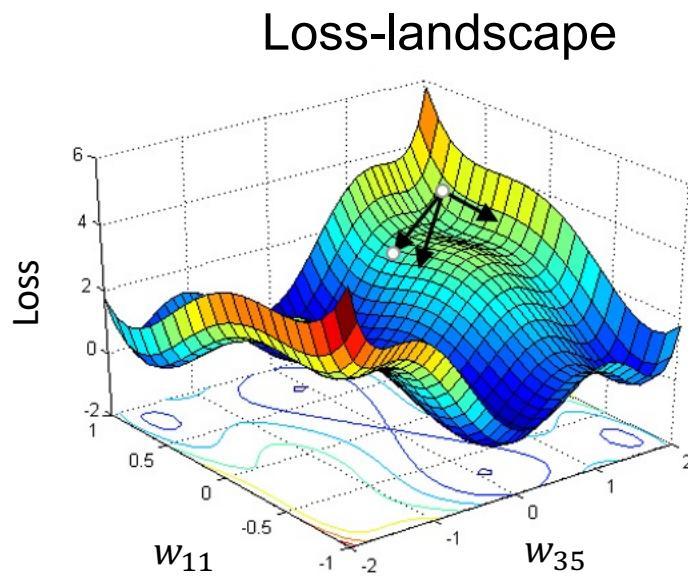
Deep Ensembling

Basic Idea of deep ensembling

- Use an ensemble of n-different models and see if they agree
- How to get different ensembles?
- 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.

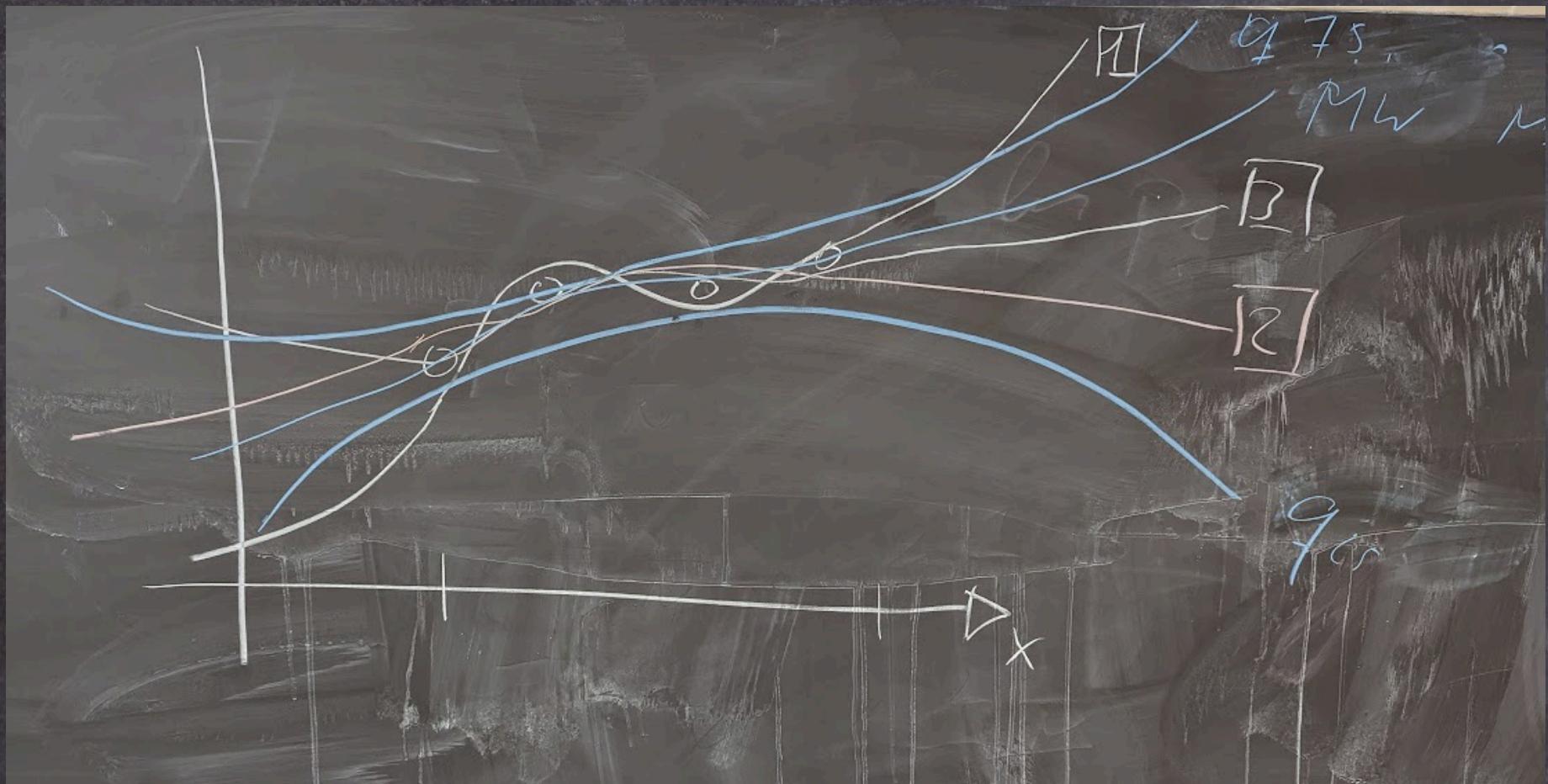
The loss-landscape in DL is usually not convex



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

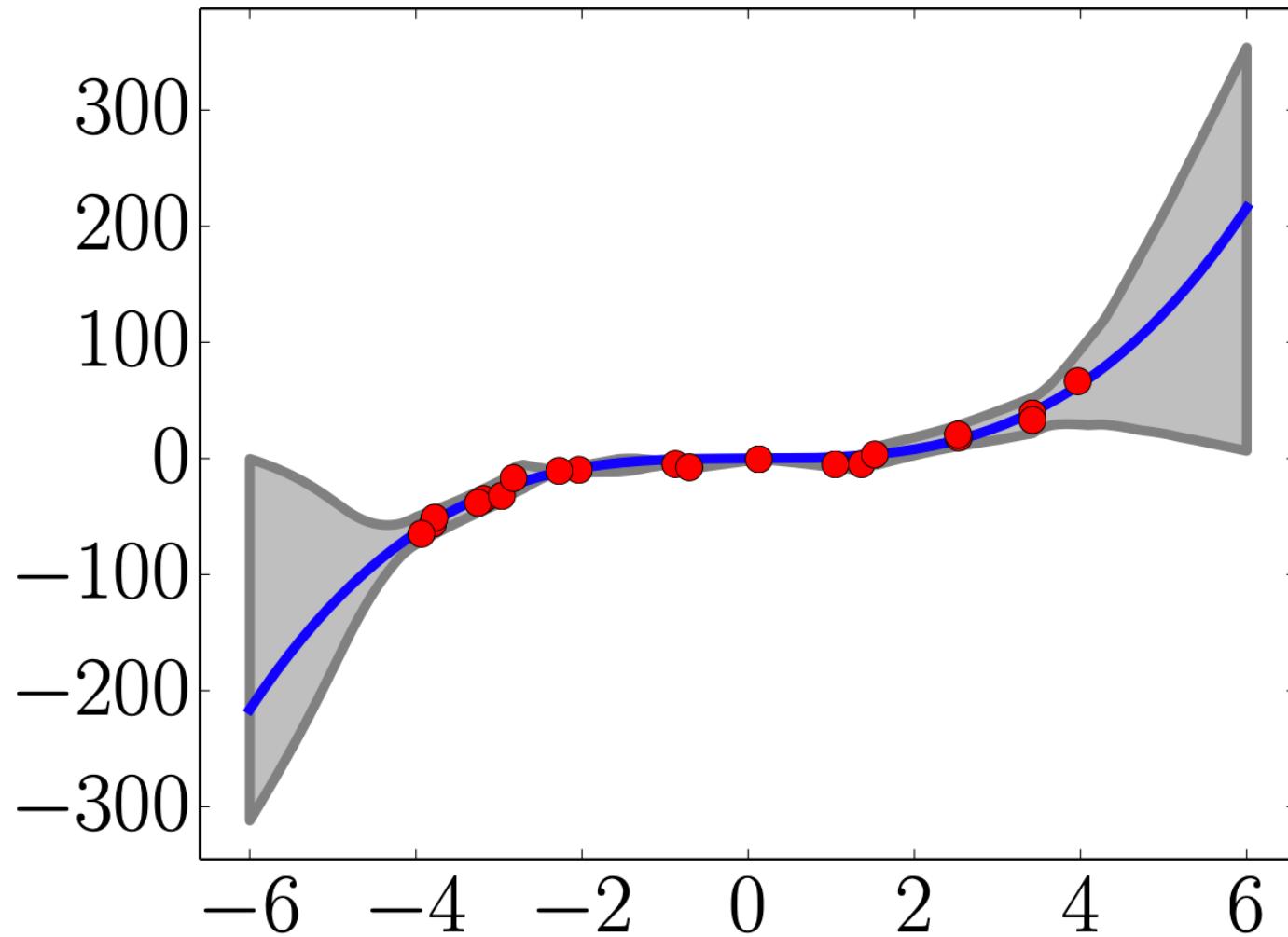
Training is started with a **random** weight value initialization → training the NN with **SGD** and the same data several times is usually ending in different local minima.

Blackboard Regression Ensembling with 3 solutions



In the data points, solutions are similar.
Outside the data there are difference

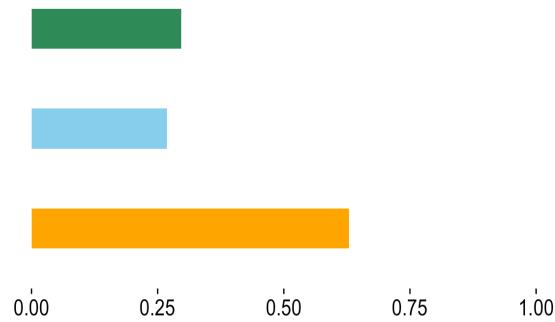
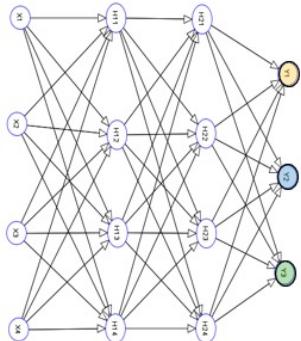
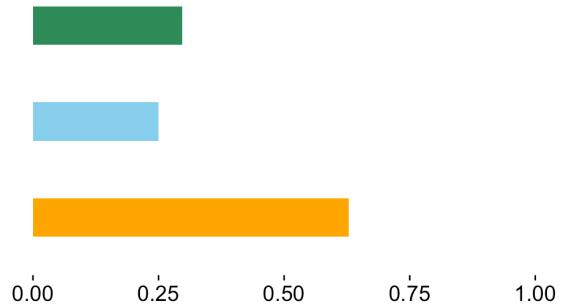
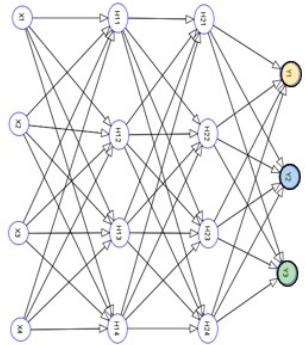
Ensemble of 5 Networks (Regression)



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

Classification

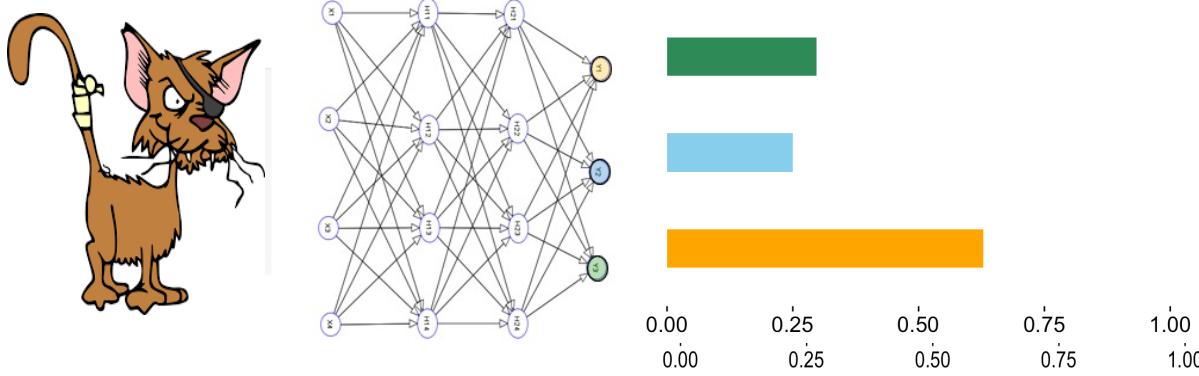
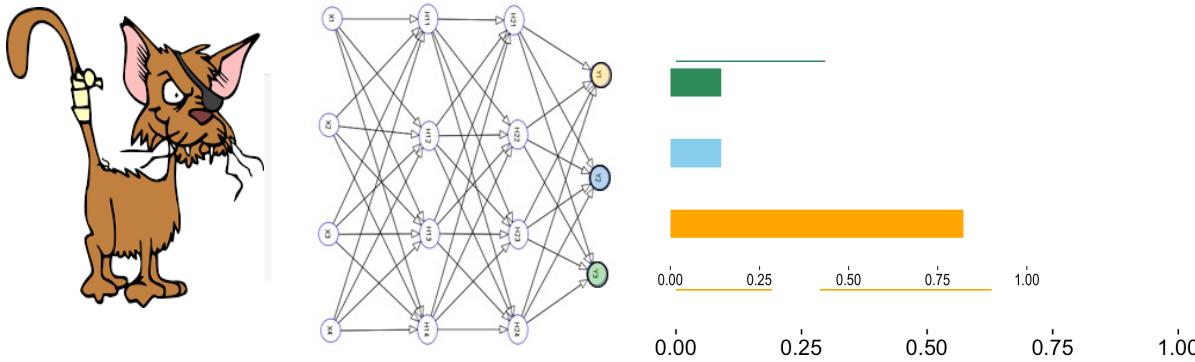
What happens if we have trained the same CNN twice with the same data? Present example in training set.



Small difference if example is known

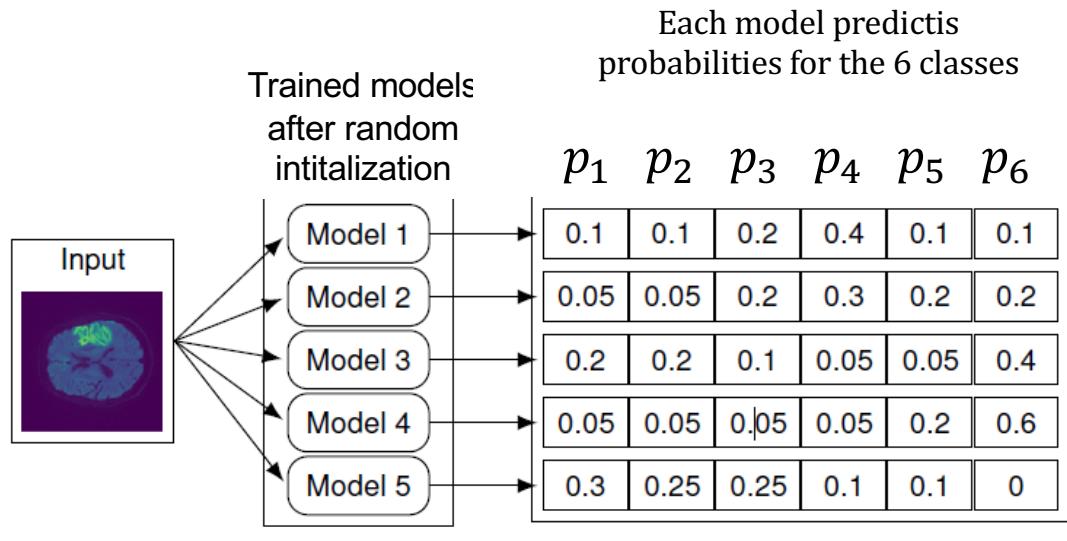
Classification

What happens if we have 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 predictions

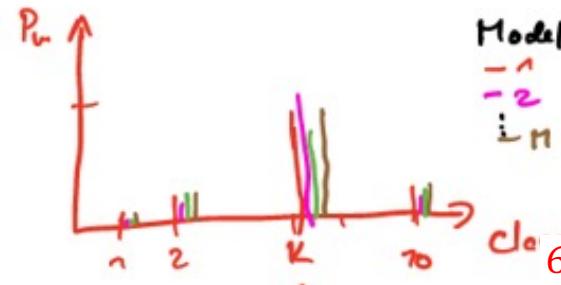


We get the **ensemble predictions** by averaging the probabilities that were predicted by the different models for each class

$$p_1^E \quad p_2^E \quad p_3^E \quad p_4^E \quad p_5^E \quad p_6^E$$

$$\begin{bmatrix} 0.14 & 0.13 & 0.16 & 0.18 & 0.13 & 0.26 \end{bmatrix}$$

We can get the **aleatoric ensemble uncertainties** by computing the entropy of the ensemble prediction $H(P^E) = -\sum_i p_i^E \cdot \log(p_i^E)$.



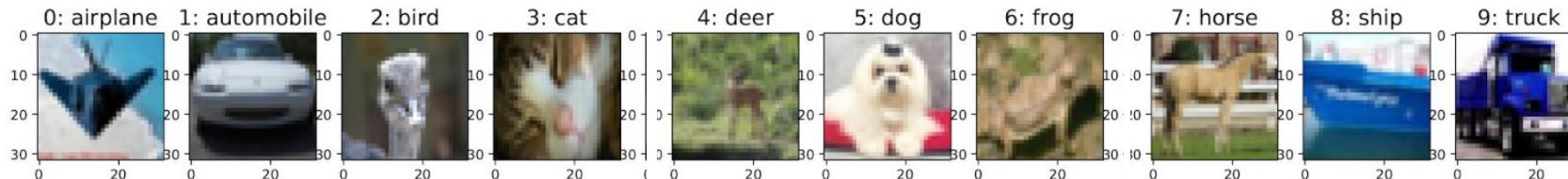
We can get the **epistemic ensemble uncertainties** by e.g. computing the standard deviation of the probabilities that were predicted by the different models for each class

$$sd_1^E \quad sd_2^E \quad sd_3^E \quad sd_4^E \quad sd_5^E \quad sd_6^E$$

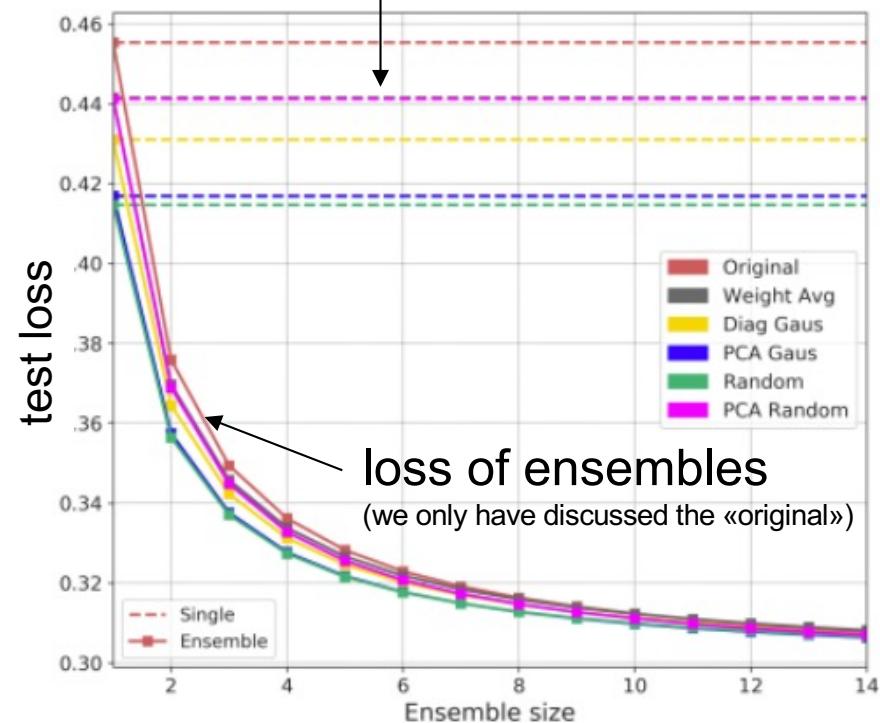
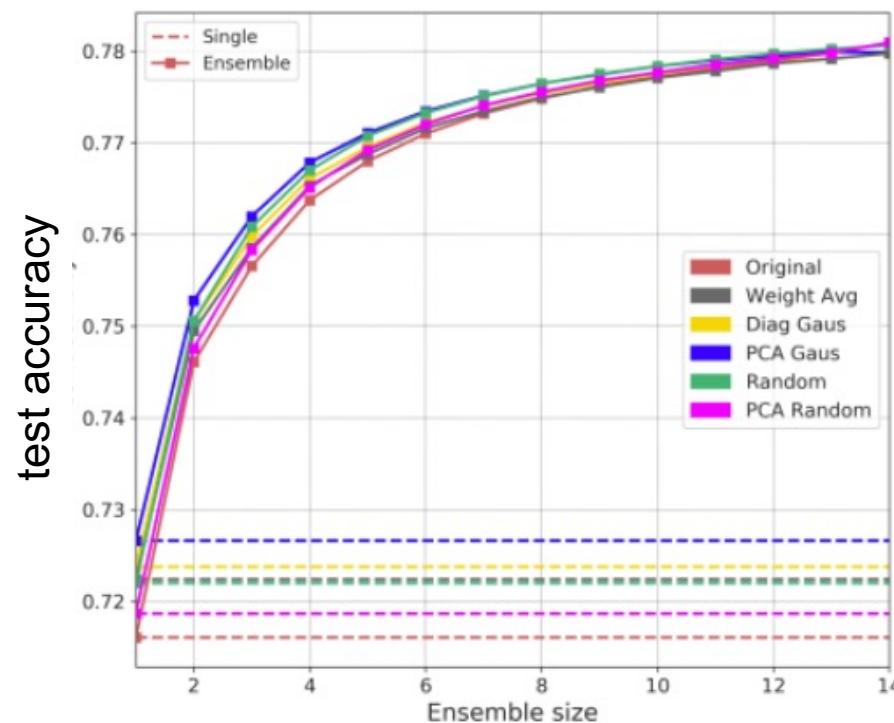
Nice:

For the convex NLL loss, it is guaranteed, that the NLL of the ensemble prediction is better (smaller or equal) than the average NLL of the individual models.

Deep ensembling improves prediction power



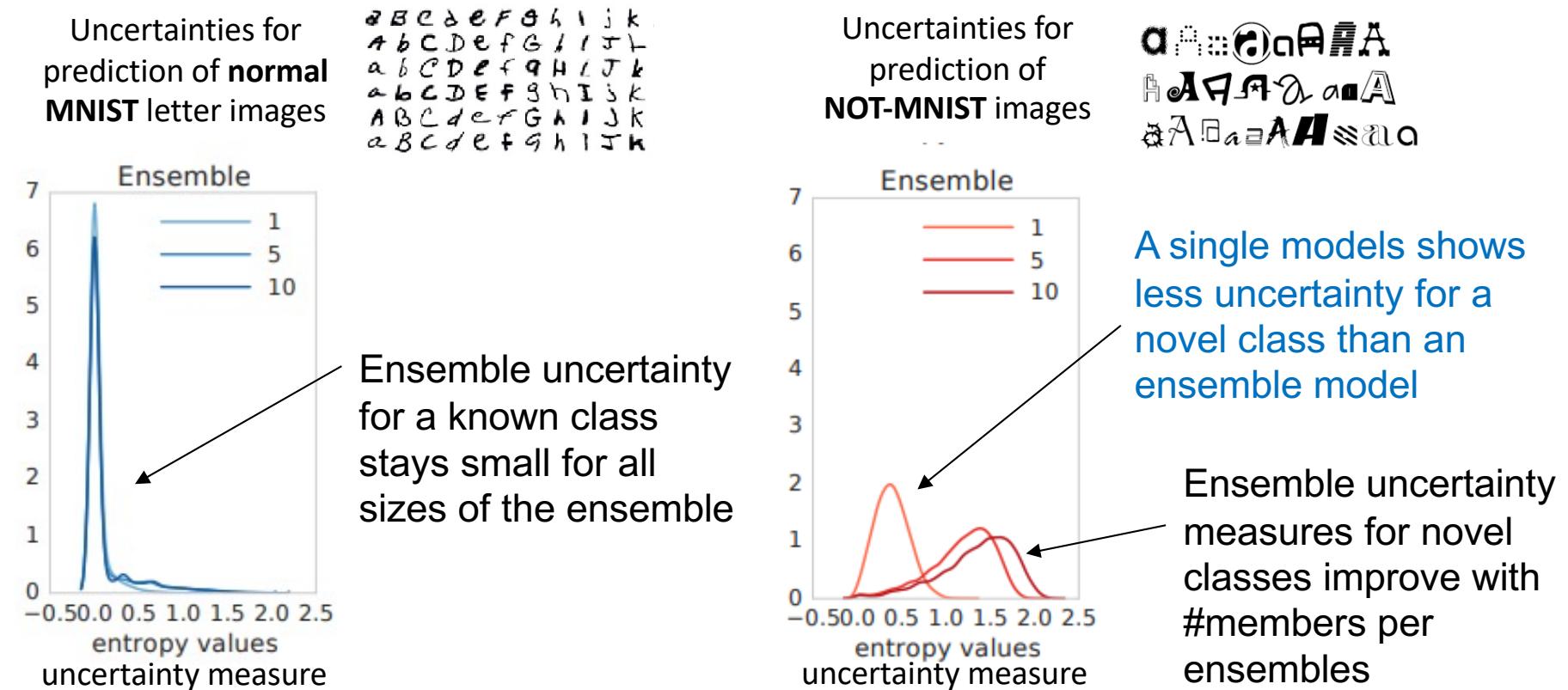
loss of 5 single models



Ensembles with as few as 3 or 5 members are typically enough to achieve a performance gain.

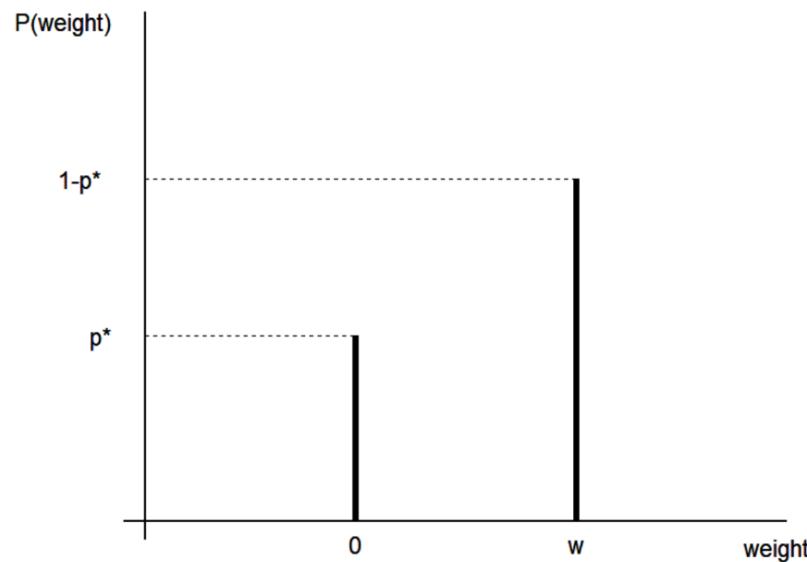
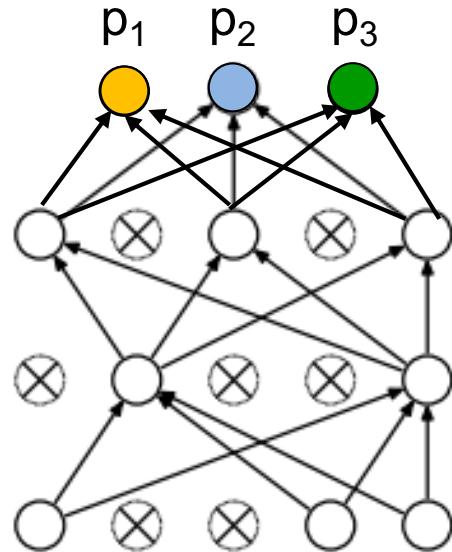
Deep ensembles improve uncertainty measures

We want that a model, that is trained on normal MNIST letter data, should provide large uncertainties when applied on novel (NOT-MNIST) letter images.



Epistemic uncertainty
Dropout during test time

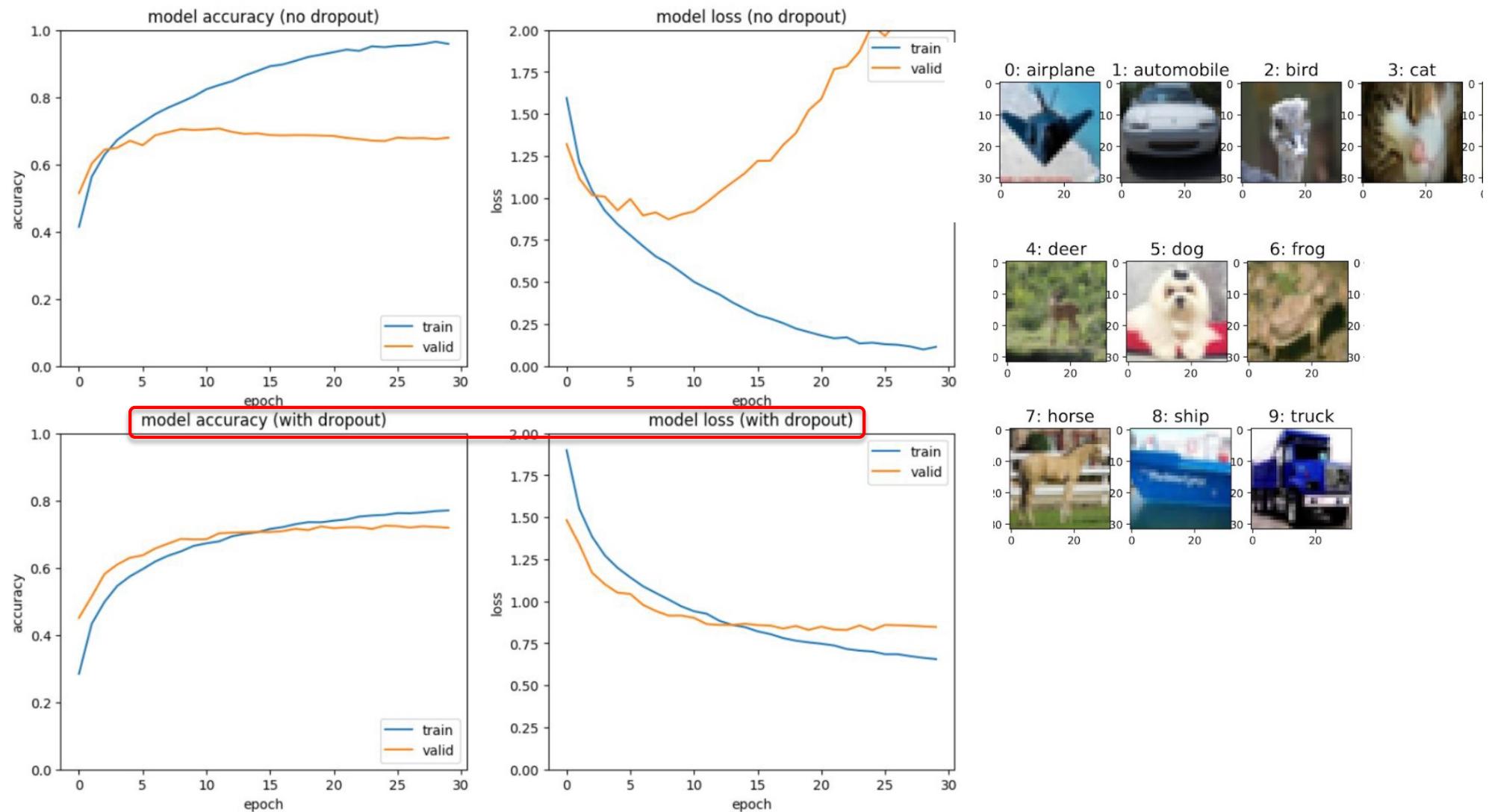
Recall: Classical Dropout only during training



Using [dropout during training](#) implies:

- In each training step only weights to not-dropped units are updated → we train a sparse sub-model NN
- For non-Bayesian NN we freeze the weights after training to a value $w \cdot p^*$

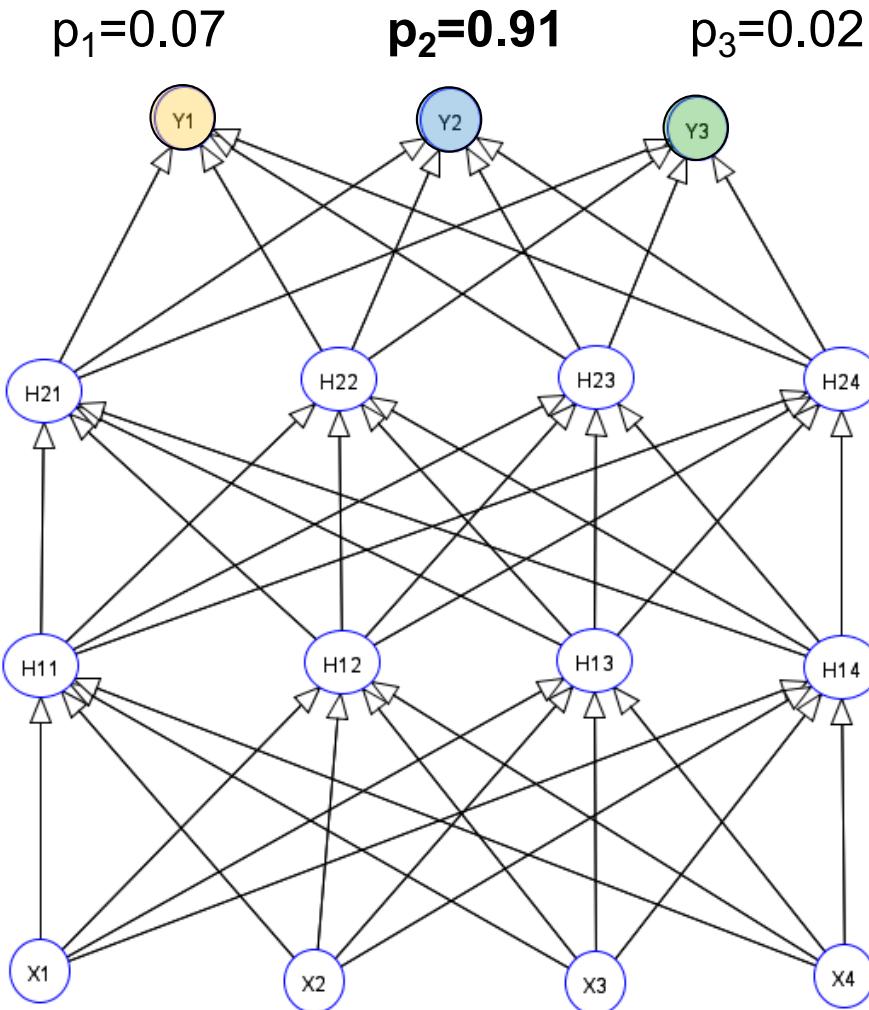
Recall: Dropout fights overfitting in a CIFAR10 CNN



From Dropout during training
to MC Dropout during test time

When using Dropout only during training

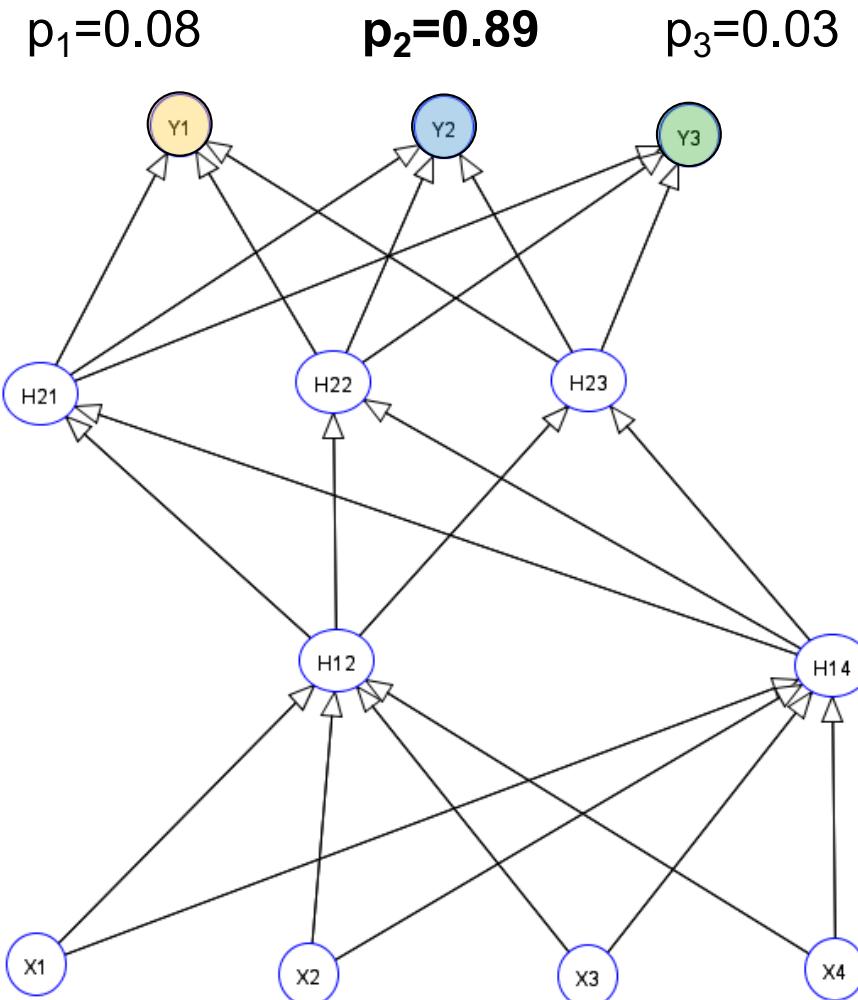
For non-Bayesian NN we freeze the weights after training to a value $w \cdot p^*$ and use then the trained NN for prediction:



Probability of predicted class: p_{\max}

Input: image pixel values

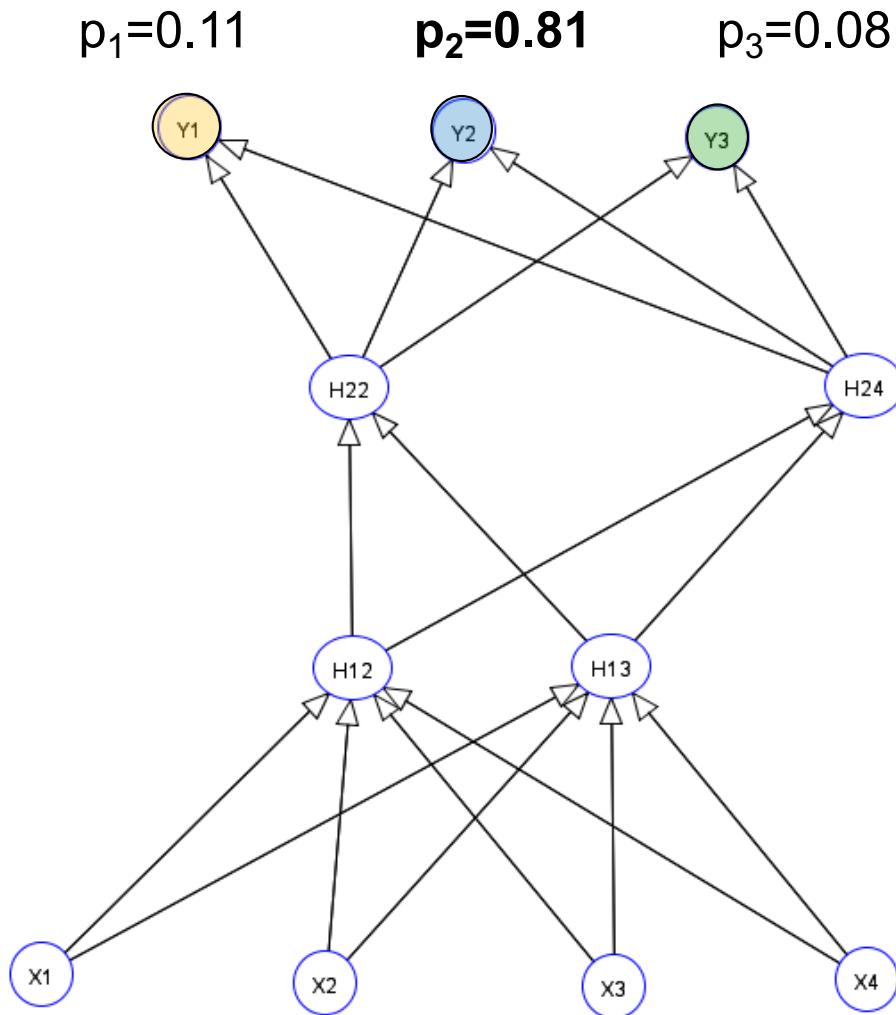
MC Dropout during test time: Run 1



Stochastic dropout of units

Same input image

MC Dropout during test time: Run 2

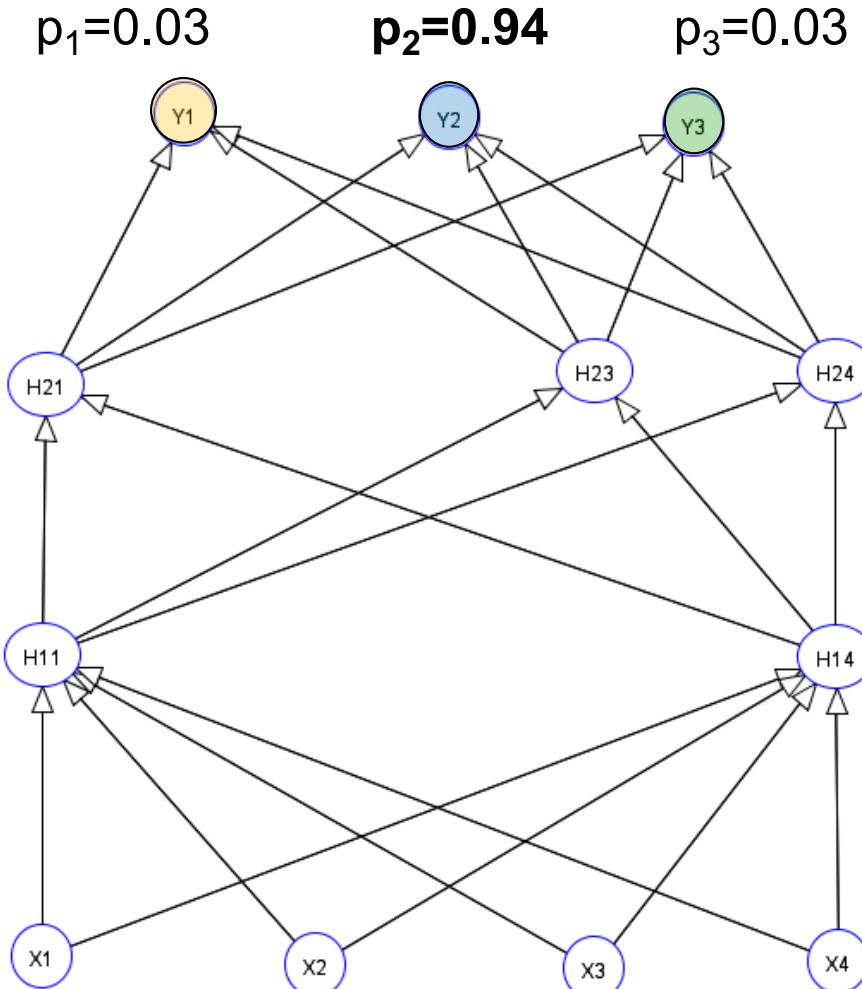


Let's shoot
out nodes

Stochastic dropout of units

Same input image

MC Dropout during test time: Run 3

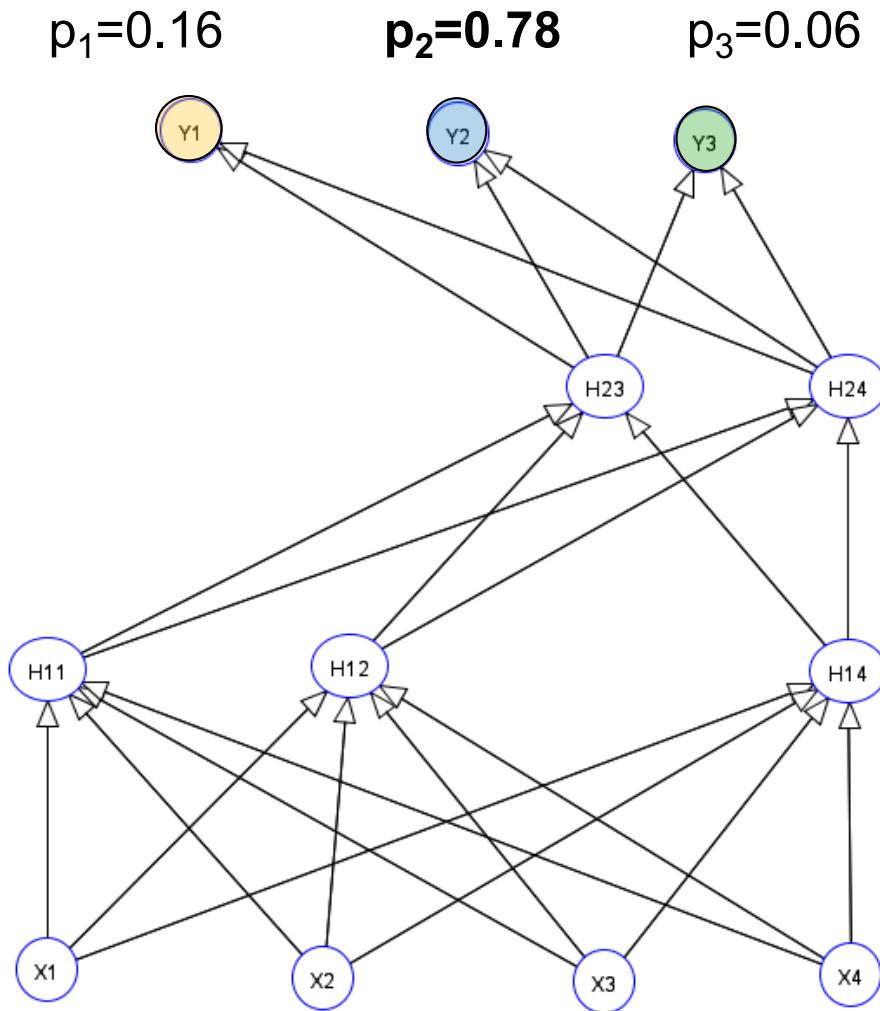


Let's shoot
out nodes

Stochastic dropout of units

Same input image

MC Dropout during test time: Run 4



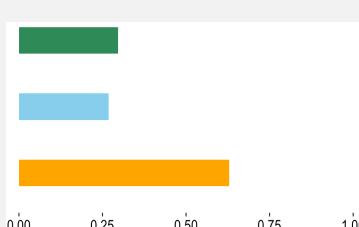
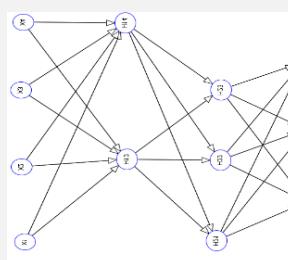
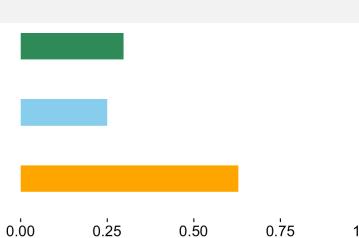
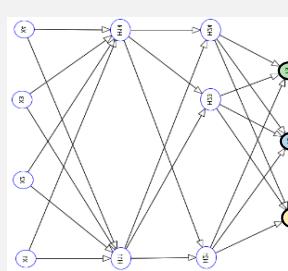
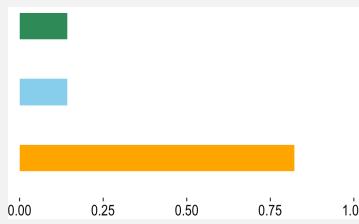
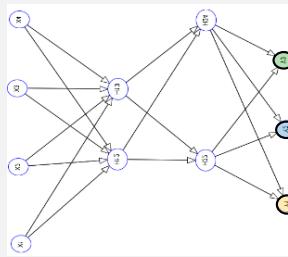
Stochastic dropout of units

Same input image

MC Dropout during test time yields a multivariate predictive distribution for the parameters



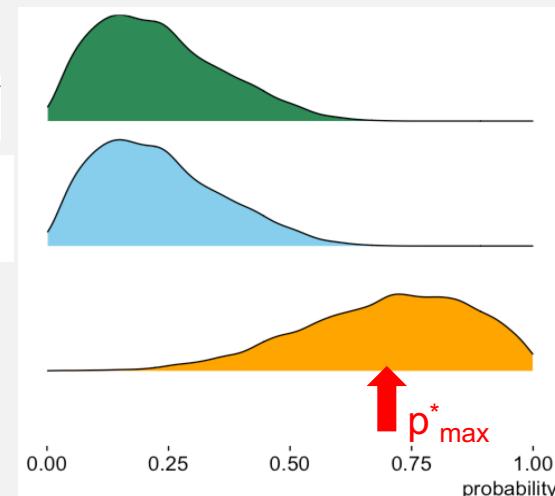
Many Dropout Runs in forward pass



...



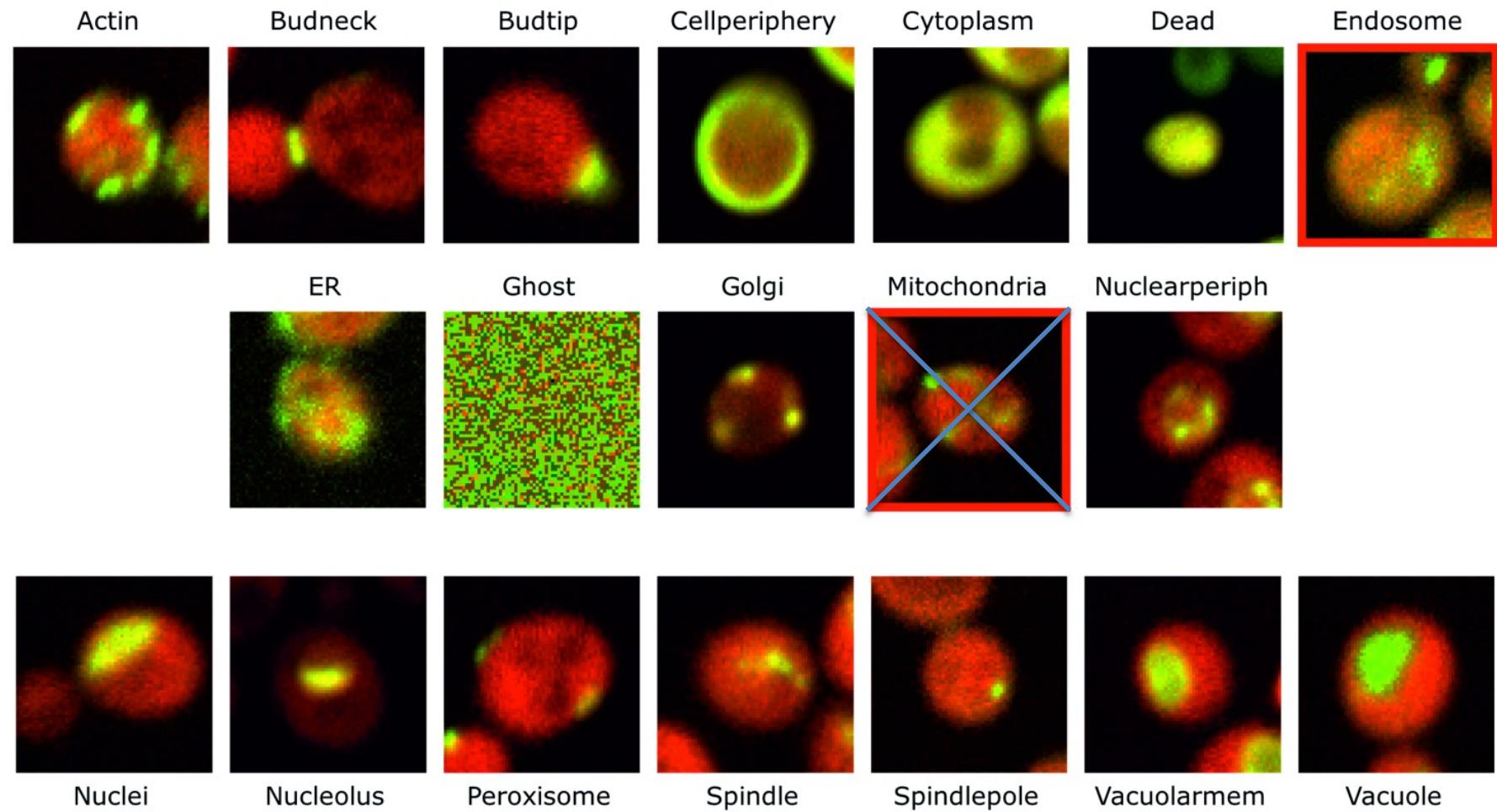
use dropout
also during
prediction



CNN predicts class “collie”
but with high uncertainty

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

Experiment with unknown phenotype

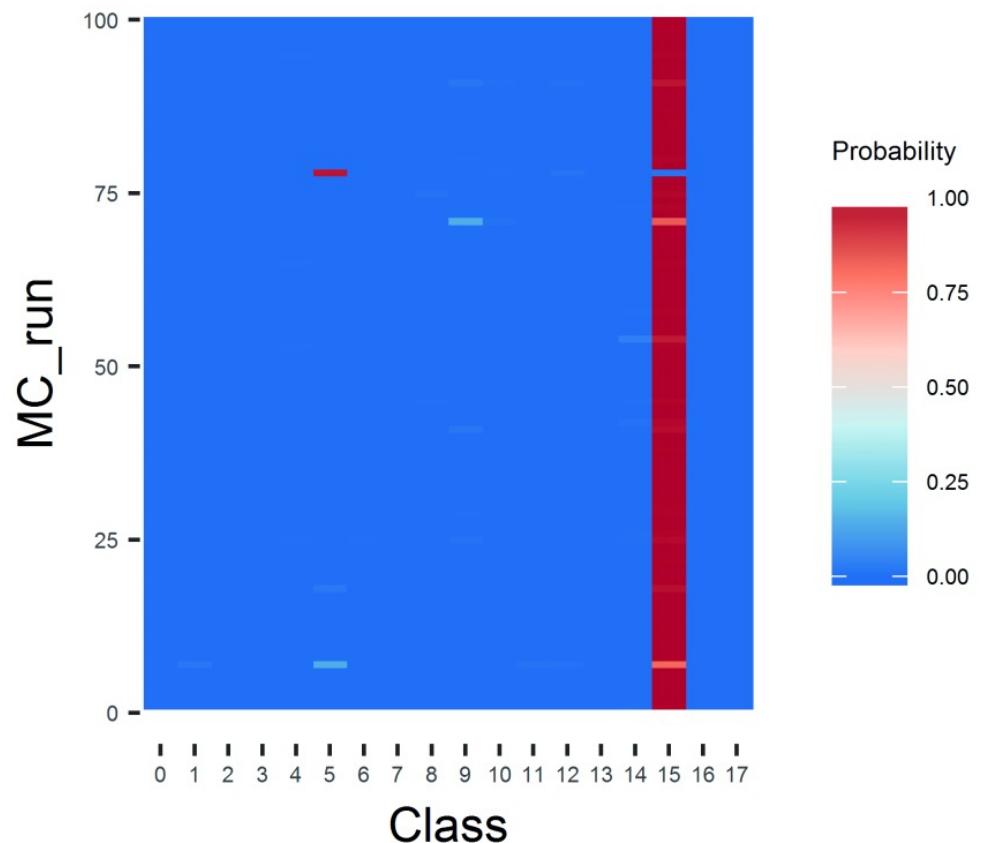


Dürr O, Murina E, Siegismund D, Tolkachev V, Steigrale S, Sick B. Know when you don't know, Assay Drug Dev Technol. 2018

Probability distribution from MC dropout runs

Image with known class 15

100 MC predictions for an image with known phenotype 15



Probability distribution from MC dropout runs

Image with known class 15

100 MC predictions for an image with known phenotype 15

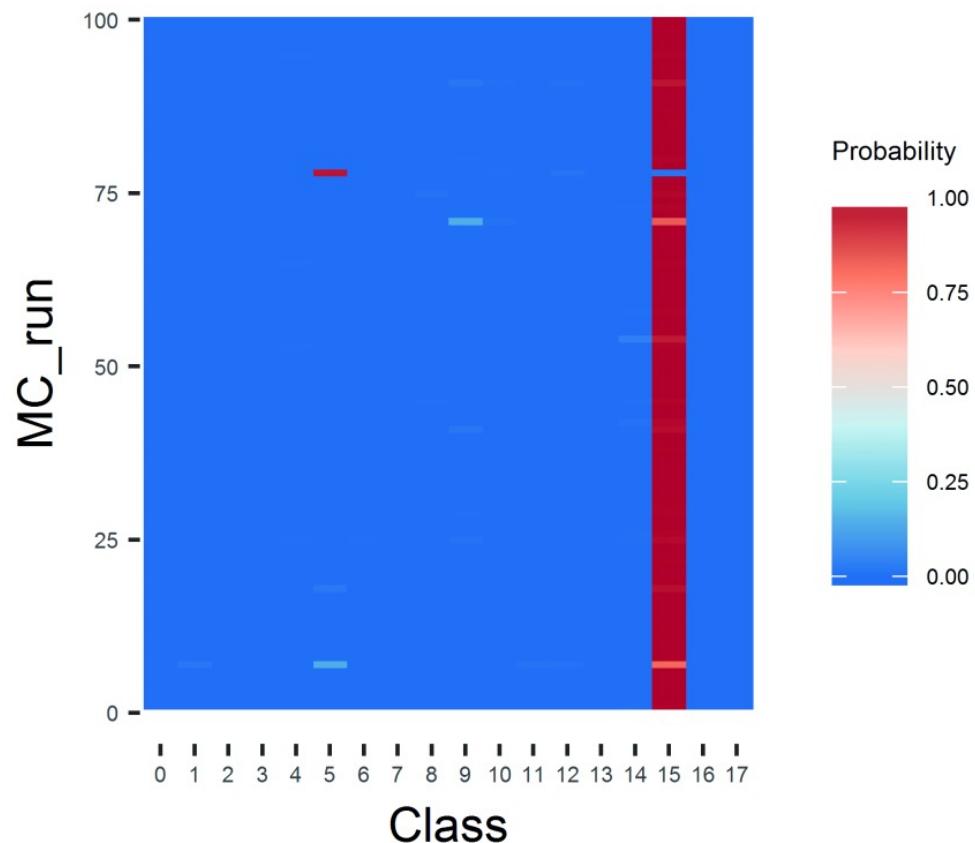
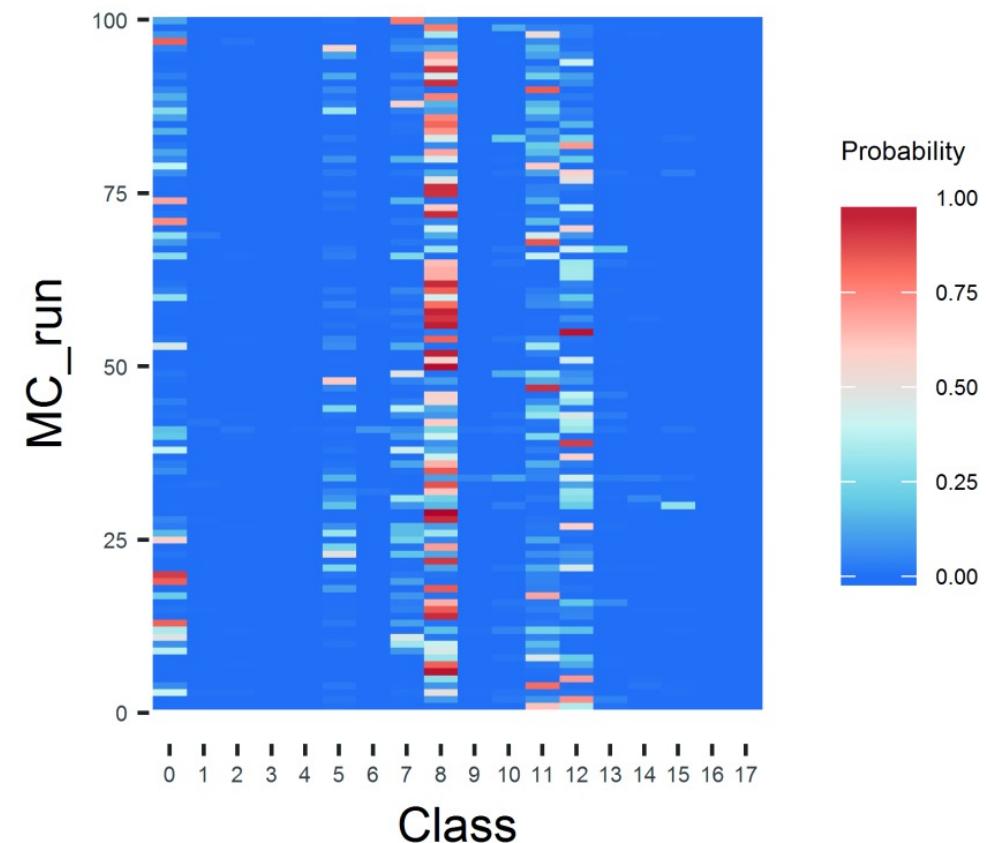


Image with unknown class

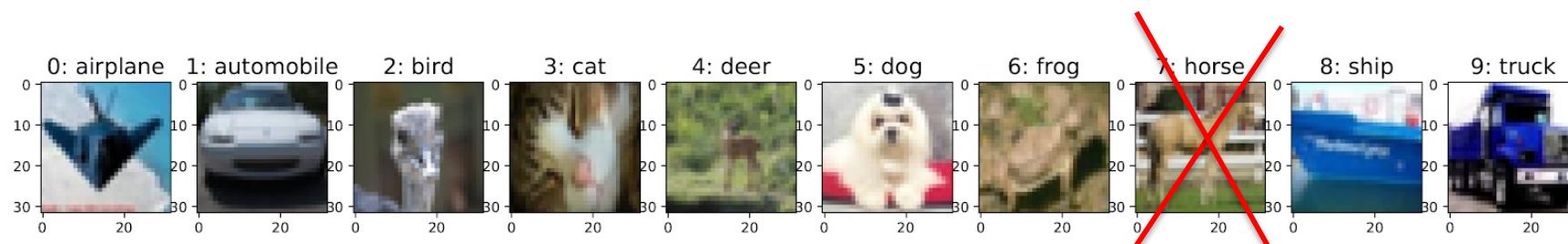
100 MC predictions for an image with an unknown phenotype



Hands-on Time



Train a CNN with only 9 of the 10 classes and investigate if the uncertainties are different when predicting images from known or unknown classes.



How does MC dropout compare with deep ensembles?

- For MC dropout we only need to compare one NN with as many parameters as a classical NN. We then average different MC dropout predictions
- For deep ensembles we need to train several NNs (typical 3 to 5) with different random initialization. We then average the predictions of these NNs
- Deep ensembles are computationally more costly but provide typically better prediction performance (and also better uncertainty measures) than MC dropout

Bayesian Neural Networks

Bayesian NN

Ensembling and MCMC Dropout work

But why?

Bayesian Statistics is the language to formulate epistemic uncertainty

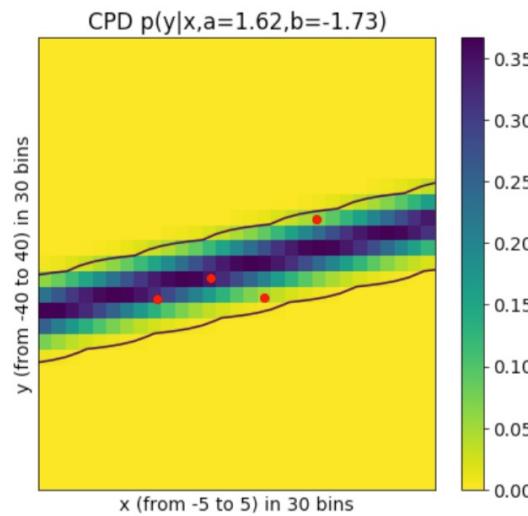
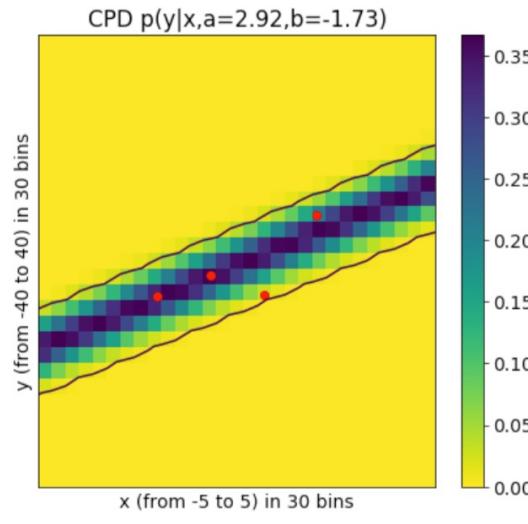
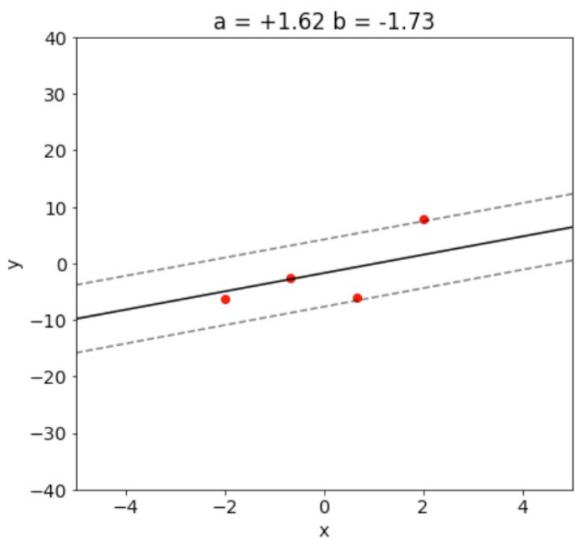
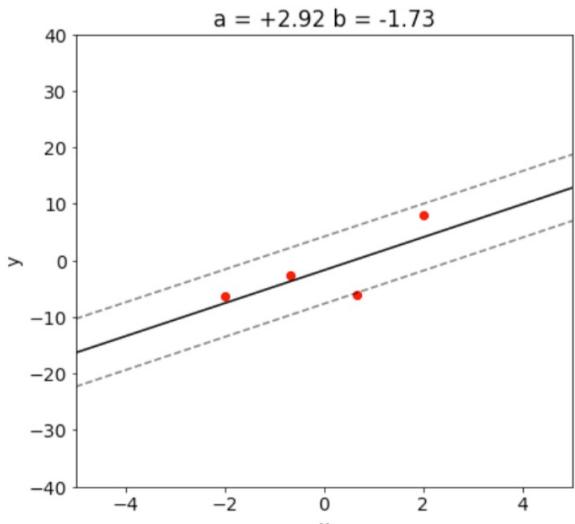
In the following a sneak preview into Bayesian Statistics

For NN we will introduce a direct approximation. Variational Inference

MC Dropout and Ensembling can be also seen a Bayesian Approximations

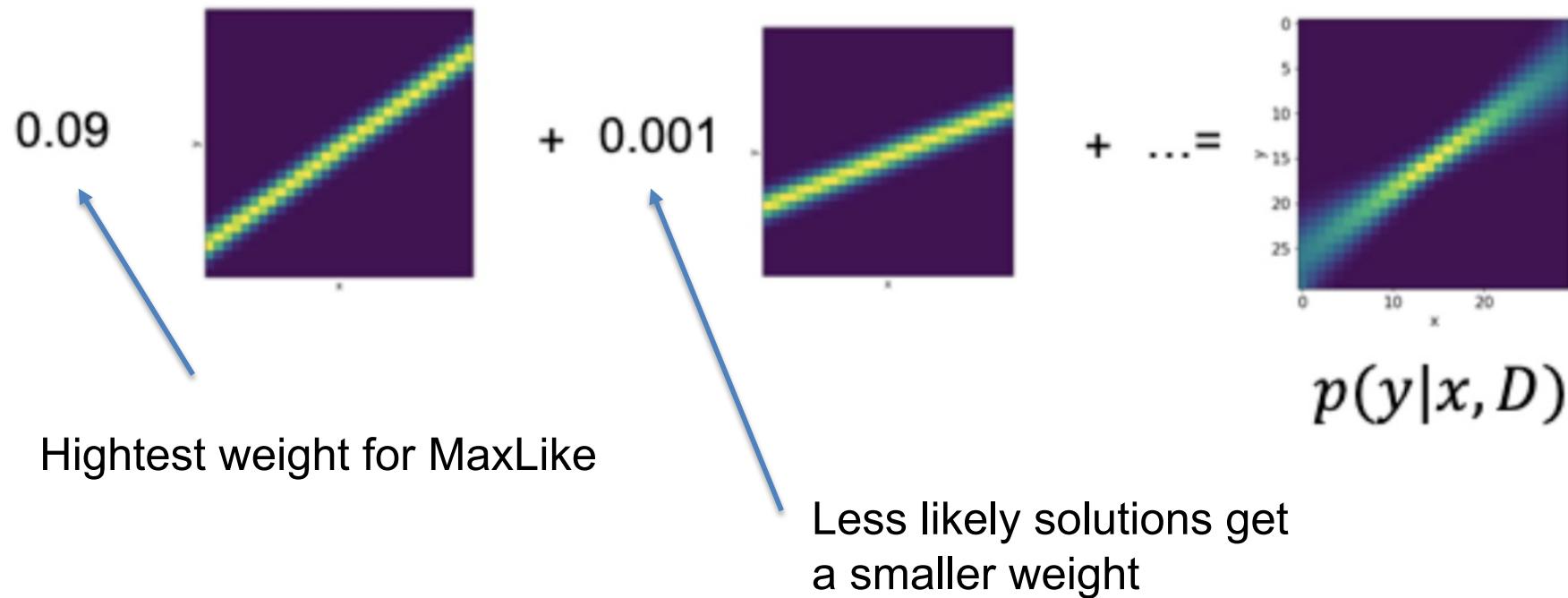
Bayes the hackers' way

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



Combining different fits

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

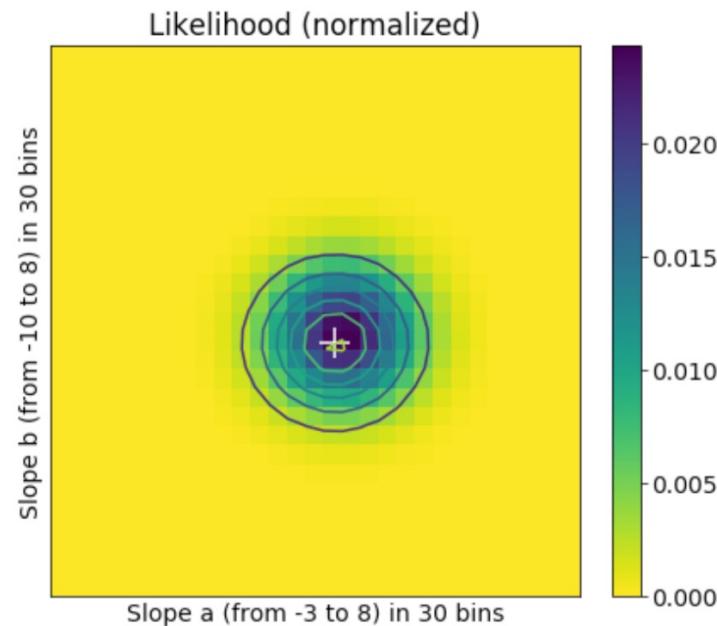


Question: How to get the weight?

Idea: use the (normalized) likelihood $p_{\text{norm}}(D|(a, b))$!

Don't put all egg's in one Basket

Also take other solutions for parameters a, b into account



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

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

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



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

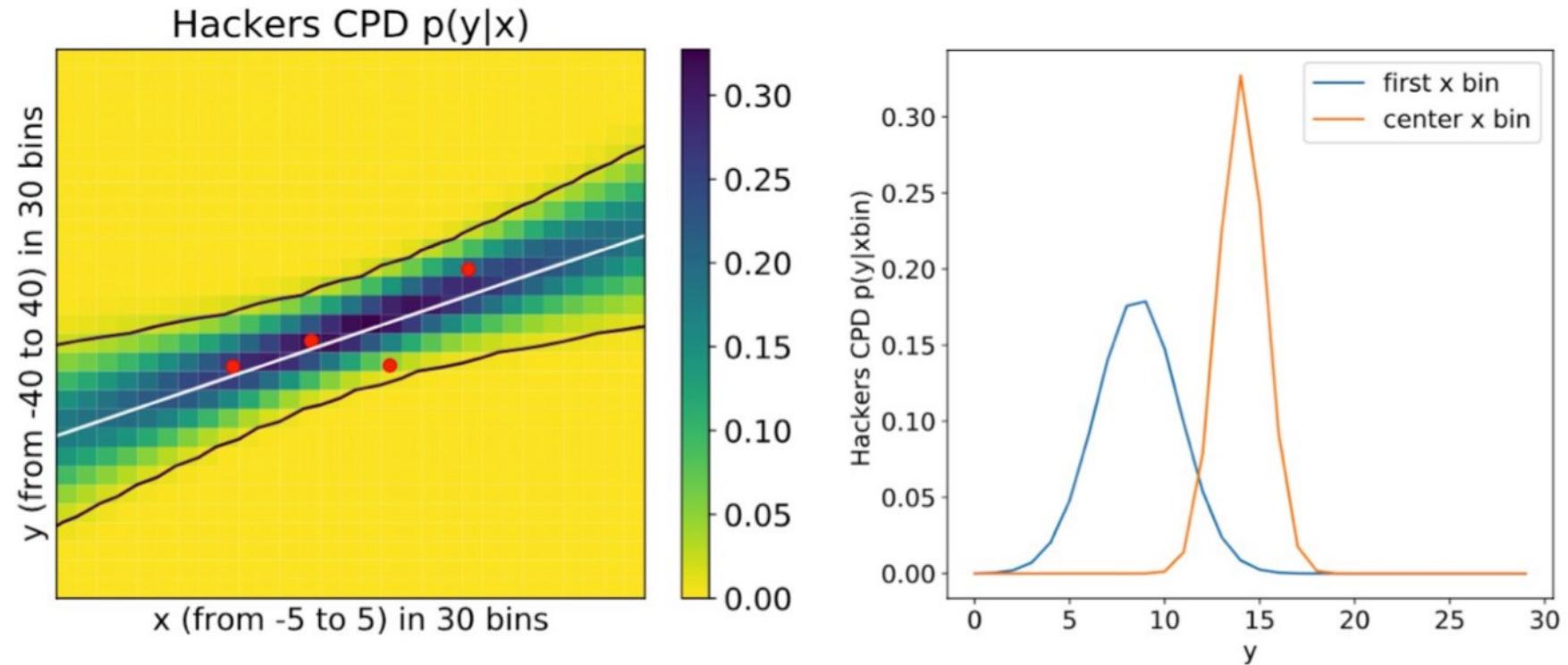


Figure 7.6 The predictive distribution for the Bayesian linear regression model, trained with the four data points shown on the left side by the color-code and on the right as conditional distribution at two different x positions. You can clearly see that the uncertainty gets larger when leaving the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

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

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

Revisiting the Hackers' example

Hacker's way

$$p(y|x, D) = \sum p(y|x, w) \cdot p_{\text{norm}}(D|w)$$

$$p_{\text{norm}}(D|w) = \frac{p(D|w)}{\sum_w p(D|w)}$$

Bayes

$$p(y|x, D) = \sum_w p(y|x, w) \cdot p(w|D)$$

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)} = \frac{p(D|w)p(w)}{\sum_w p(D|w)p(w)}$$

→ The Hacker's way is Bayes, if we assume a uniform prior $p(w) = \text{const}$

Bayesian modeling has less problems with complex models

Frequentist's strategy:

You can only use a complex model if you have enough data!

Bayesian's strategy:

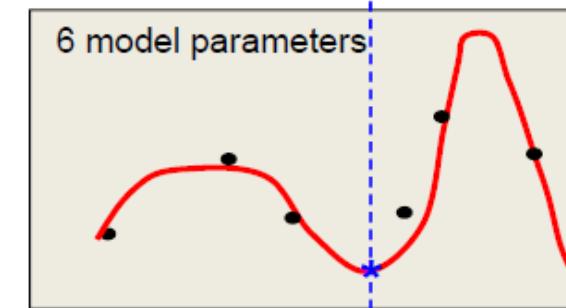
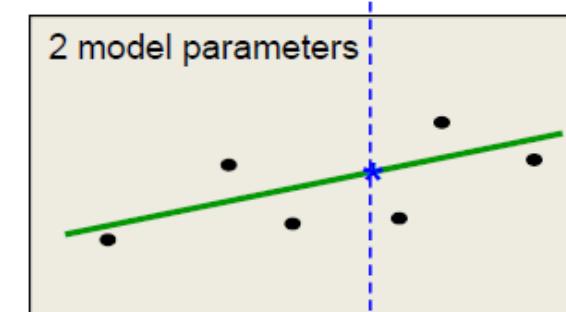
Use the model complexity you believe in.

Do not just use the best fitting model.

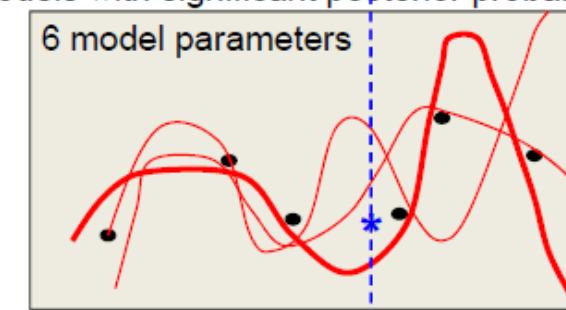
Do use the **full posterior distribution over parameter settings** leading to vague predictions since **many different parameter settings have significant posterior probability**.

$$p(y|x^*, \text{traindata}) = \int p(y|x^*, \theta) \cdot p_{\text{norm}}(\theta | \text{traindata}) d\theta$$

Image credits: Hinton coursera course



Models with significant posterior probability



x^* : new input

Bayesian terminology



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

$p(\theta|D)$ posterior

$p(D|\theta)$ likelihood

$p(\theta)$ prior

$p(D)$ evidence just a normalization constant

The Bayesian Mantra (say it loud)

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

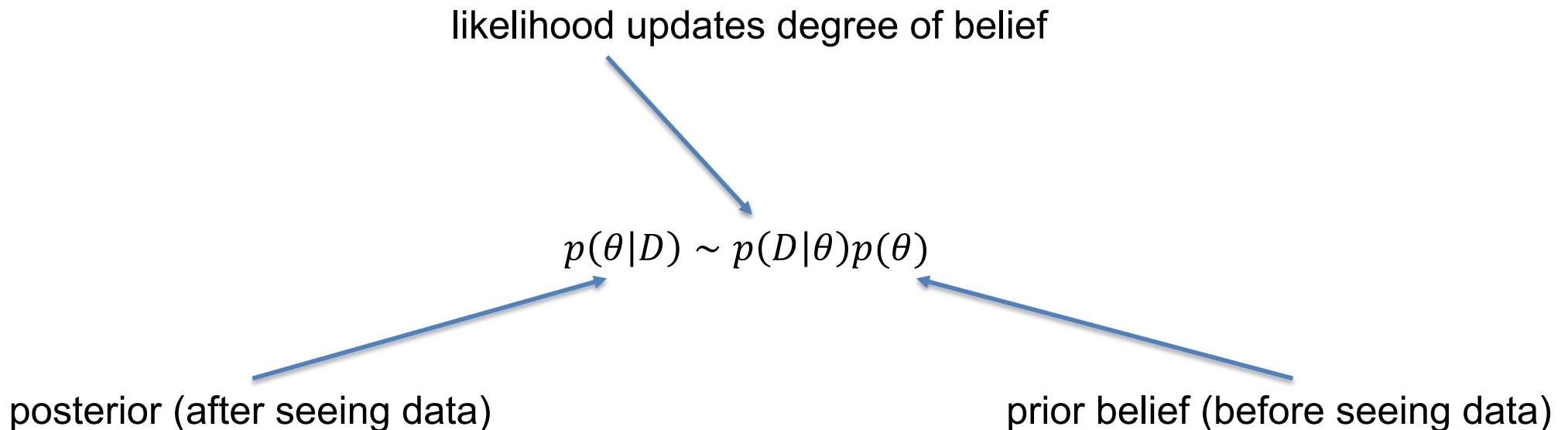
“The posterior is proportional to the likelihood, times the prior”

Interpretation updating the degree of belief

Parameters θ are random variables, following a distribution

This distribution reflects our belief about which values are probable for θ

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



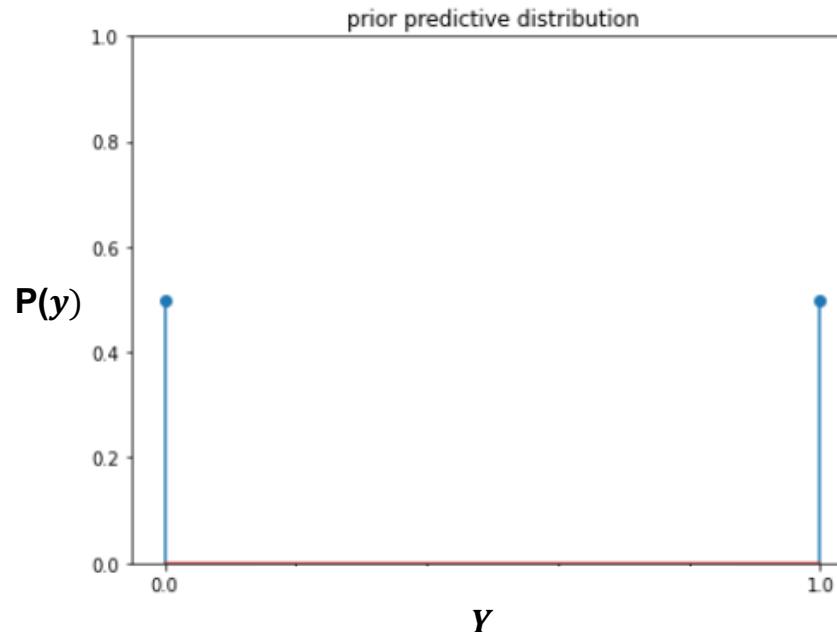
Example Coin Toss

Head or tail?



$Y_i \in \{0, 1\}$, here 0 stands for tail and 1 for head
 $Y \sim \text{Ber}(\theta)$, with 1 parameter $\theta = P(Y = 1)$

For a fair coin $\theta = 0.5$



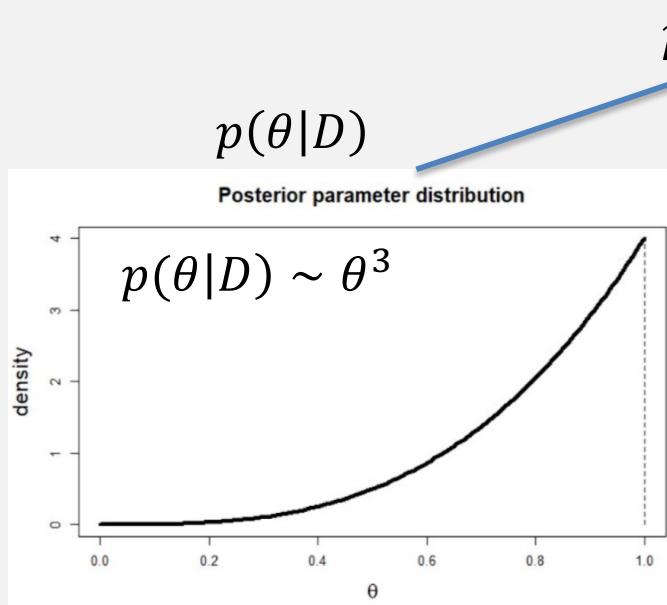
But how to know if a coin is fair?

Analyzing a Coin Toss Experiment

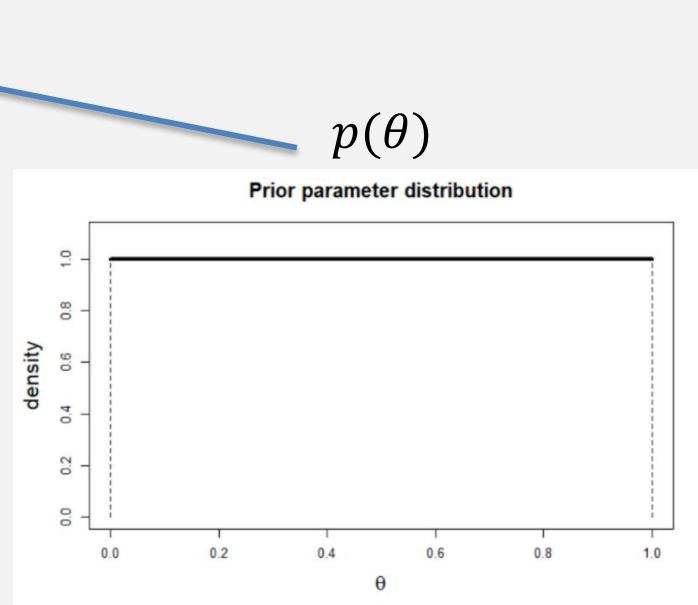


- We do an experiment and observe 3 times head $\rightarrow D = \text{'3 heads'}$
- θ parameter for the Bernoulli-distribution (probability of head)
- Before the experiment we assume all value of θ are equally likely $p(\theta) = \text{const.}$
- Calculate likelihood $p(D|\theta) = p(y=1) \cdot p(y=1) \cdot p(y=1) = \theta \cdot \theta \cdot \theta = \theta^3$
- Posterior $p(\theta|D) \sim p(D|\theta)p(\theta) = \theta^3$ beliefs more in head

Posterior (after)



Prior (before)



$$p(\theta|D) = 4 \cdot \theta^3 \quad (\text{the factor 4 is needed for normalization so that the posterior integrates to 1})$$

Posterior Predictive Distribution

Posterior predictive distribution

$$p(y|x, D) = \int_{\theta} p(y|x, \theta) \cdot p(\theta|D) d\theta$$

The coin example is unconditional (there is no predictor x)

$$p(y|D) = \int_{\theta} p(y|\theta) \cdot p(\theta|D) d\theta$$

To compute the posterior predictive probability for head in the coin example, we need:

- $p(y = 1|D) = \theta$
- $p(\theta|D) = 4 \cdot \theta^3$

Posterior predictive distribution

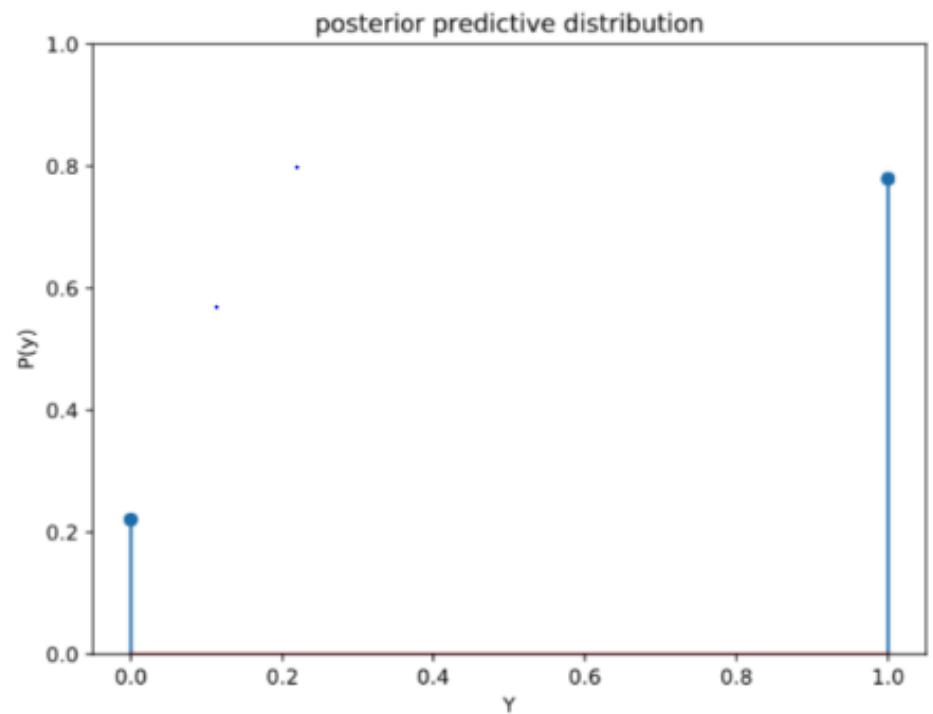
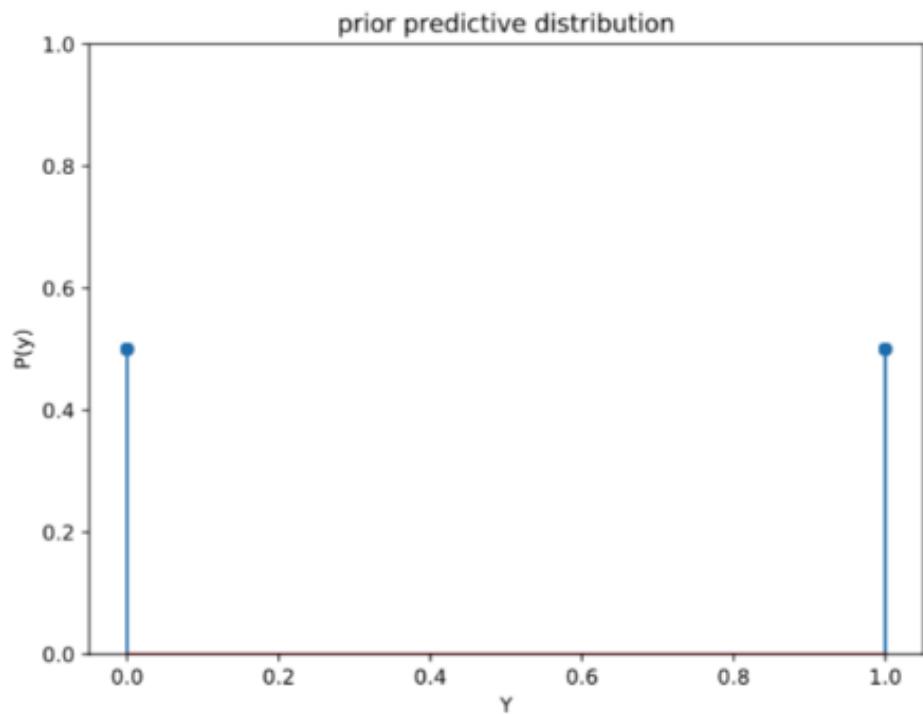
$$p(y = 1|D) = \int_{\theta} \theta \cdot 4 \cdot \theta^3 d\theta = 0.8$$

$$P(Y = 1|D) = \int_0^1 \theta \cdot 4 \cdot \theta^3 d\theta = \frac{4}{5} \cdot \theta^5 \Big|_0^1$$

$$P(Y = 1|D) = \frac{4}{5} \cdot 1^5 - \frac{4}{5} \cdot 0^5 = 0.8$$

$$P(Y = 0) = 1 - P(Y = 1) = 0.2$$

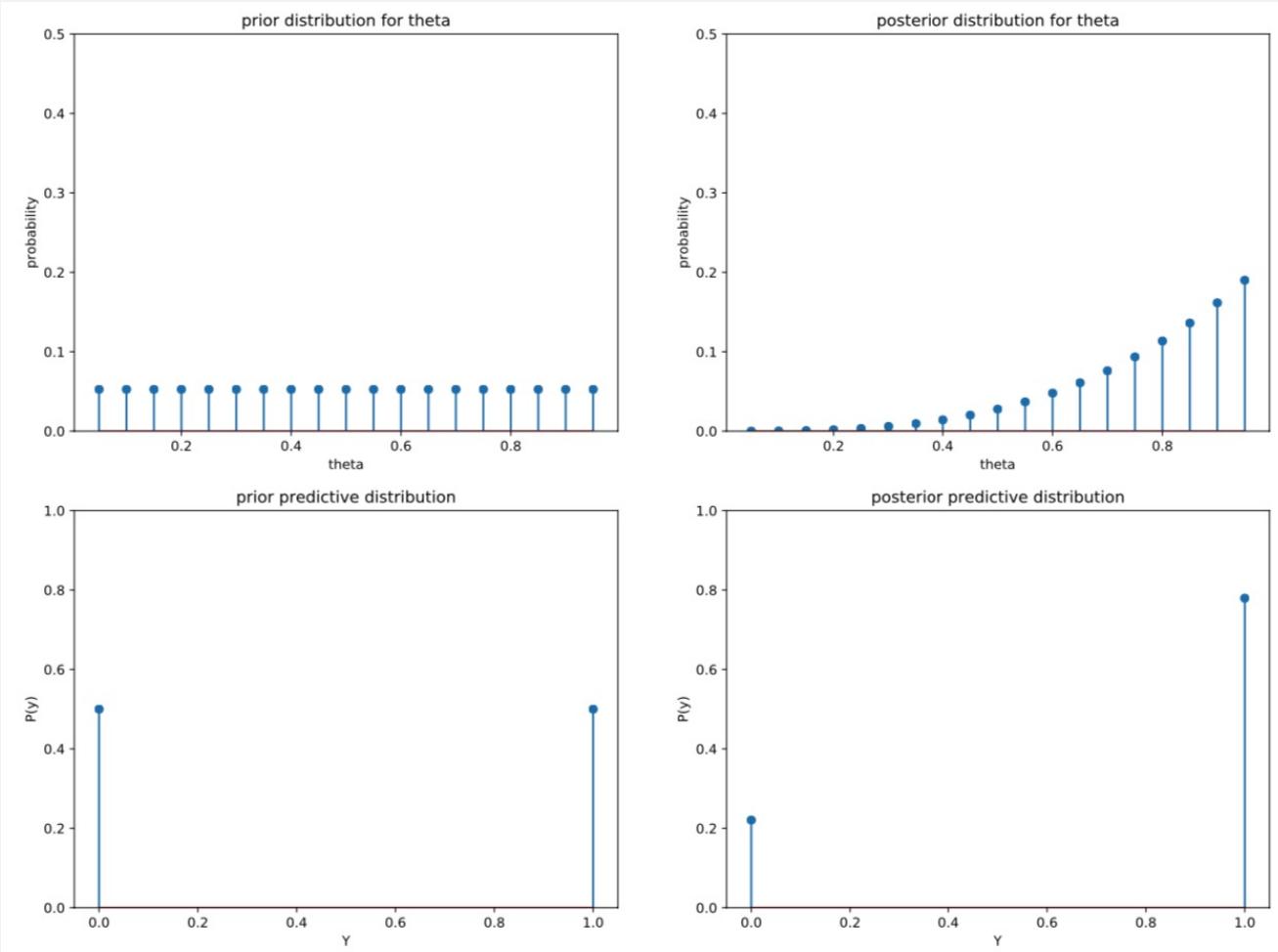
Prior and Posterior Predictive distribution



Coin example «the hacker's way»



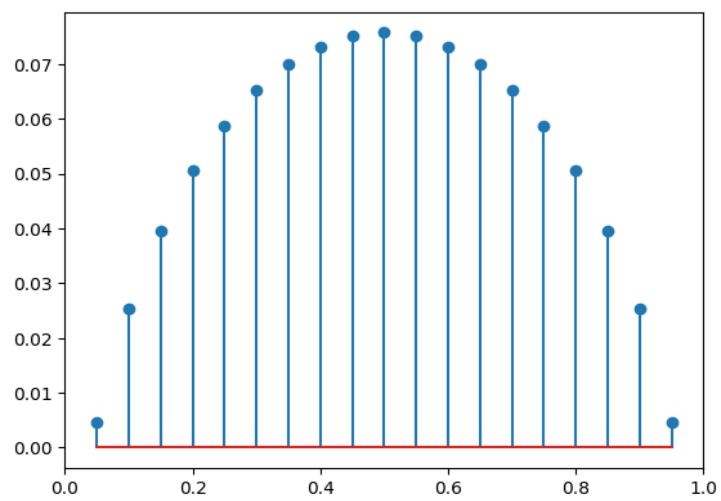
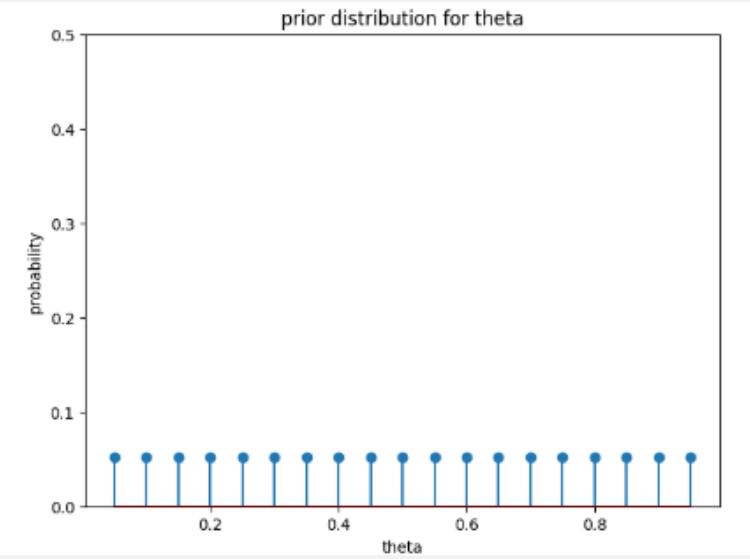
Work through the notebook NB21 and do the exercise therein.



Result 40+11 times head and 9 tails



Prior



Posterior

