Machine Intelligence:: Deep Learning
Week 8

Beate Sick, Jonas Brändli, Oliver Dürr

Bayesian Neural Networks

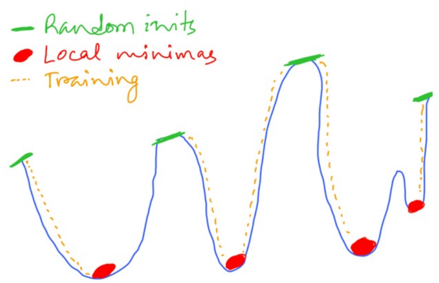
Importance to detect OOD

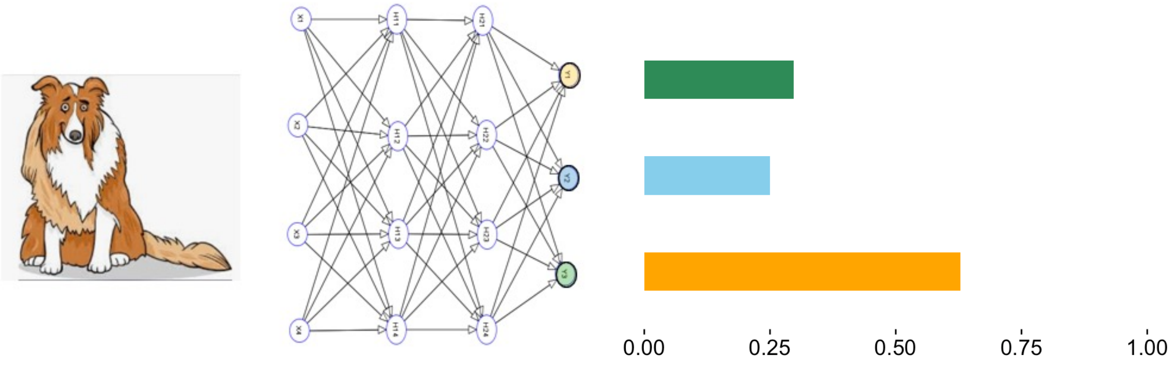
- Current DL Systems bad in out of distribution OOD situations
- Application need at least to detect OOD situations

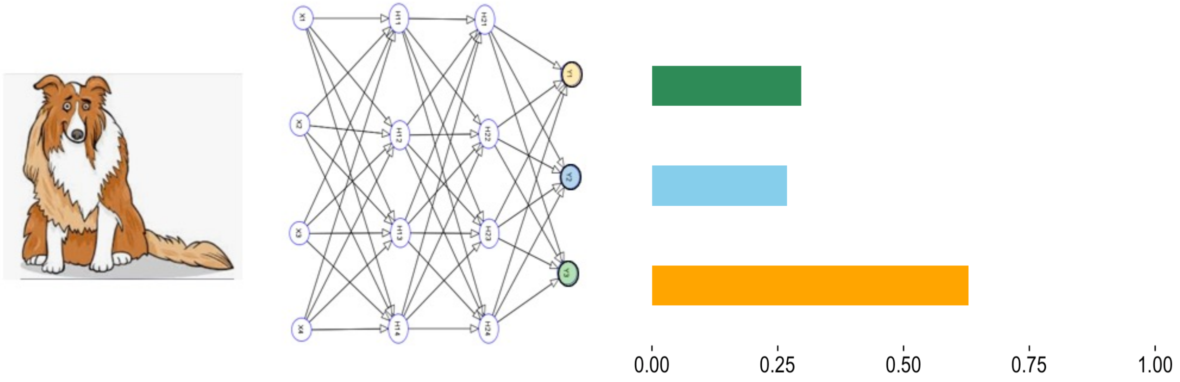
Foto: https://www.crn.de/software-services/halloween-training-fuer-google-autos.108409.html

Ensembling

Use two networks trained on same data

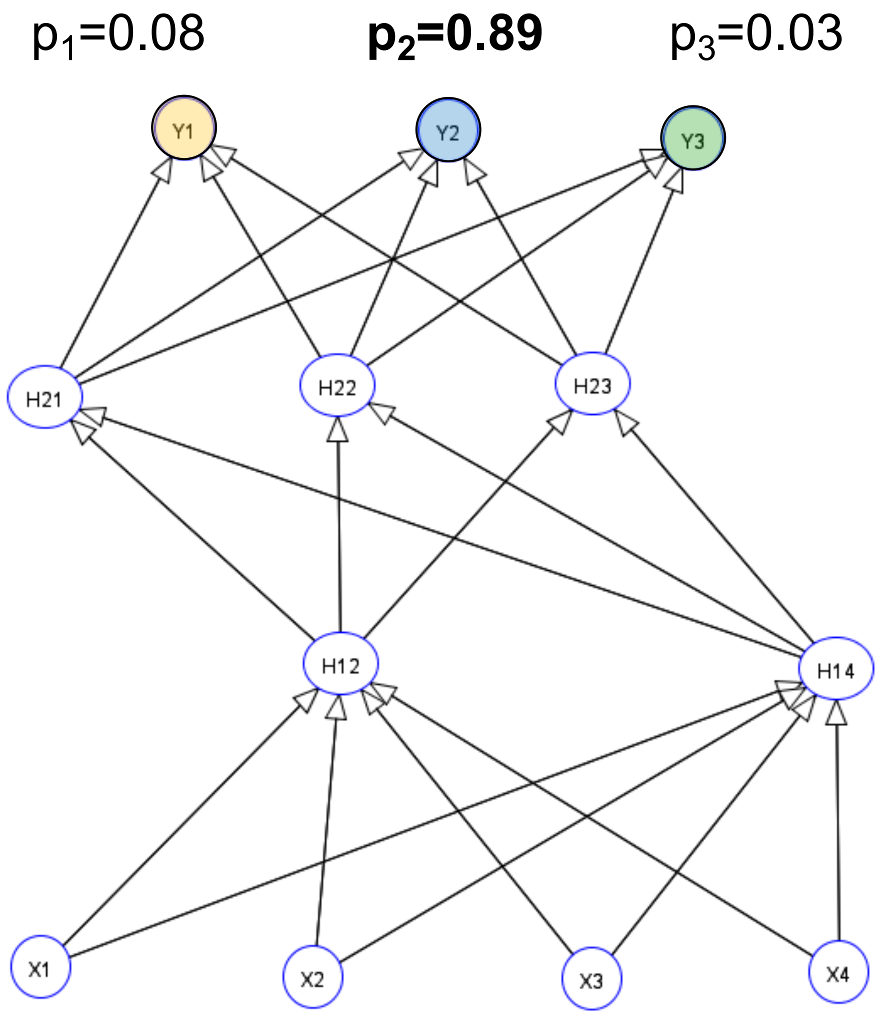


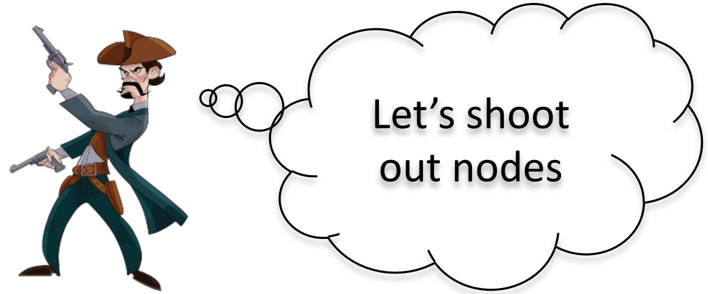




Small difference if example is know

MC Dropout during test time: Run 1 (Average over many runs)





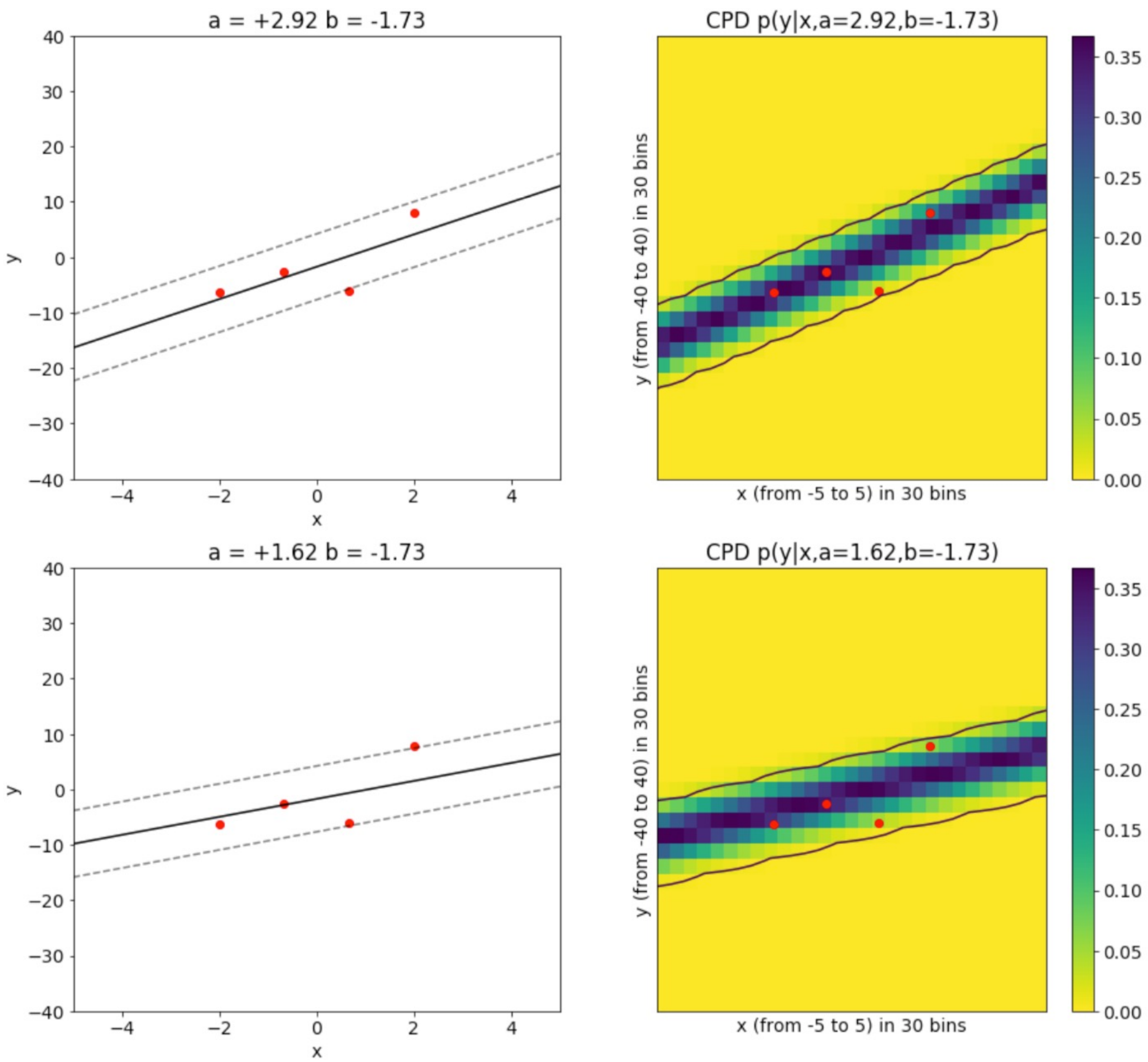
Stochastic dropout of units

Same input image

Bayes

Recap 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.



MaxLike Solution

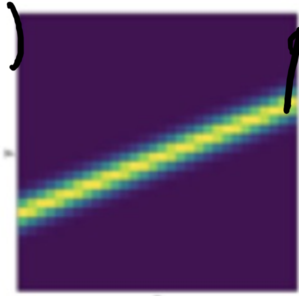
A bit off the MaxLike Solution

Combining different fits

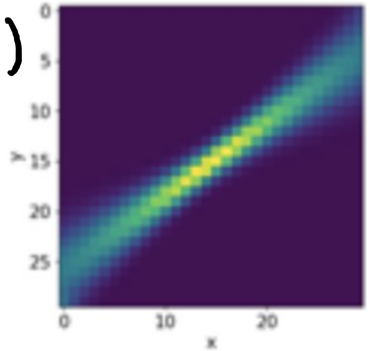
Also take the other fits with different parameters into account and weight them

$p(a,b|D)$ 0.09

$p(a,b|D)$ + 0.001



$p(a,b|D)$ + ... =



$p(y|x,D)$

$$p(y|x,D) = \int \underbrace{p(y|x,a,b)}_{N(y,\, \mu = a\cdot x + b,\, \sigma^2)} \underbrace{p(a,b|D)}_{\text{Posterior}} \, da \, db$$

Analyzing a Coin Toss Experiment

- We do an experiment and observe 3 times head $\rightarrow$ D='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)

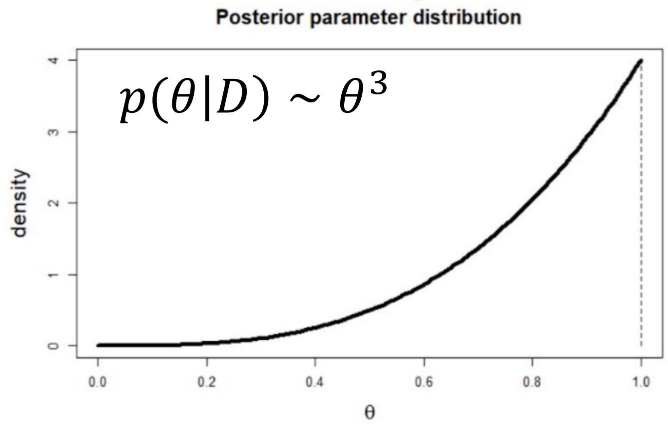
Prior (before)

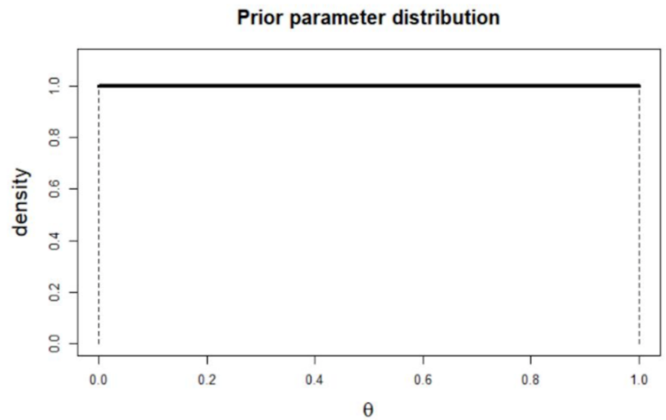
$$p(\theta|D) \sim p(D|\theta)p(\theta)$$

$p(\theta|D)$

θ^3

$p(\theta)$





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

Bernoulli

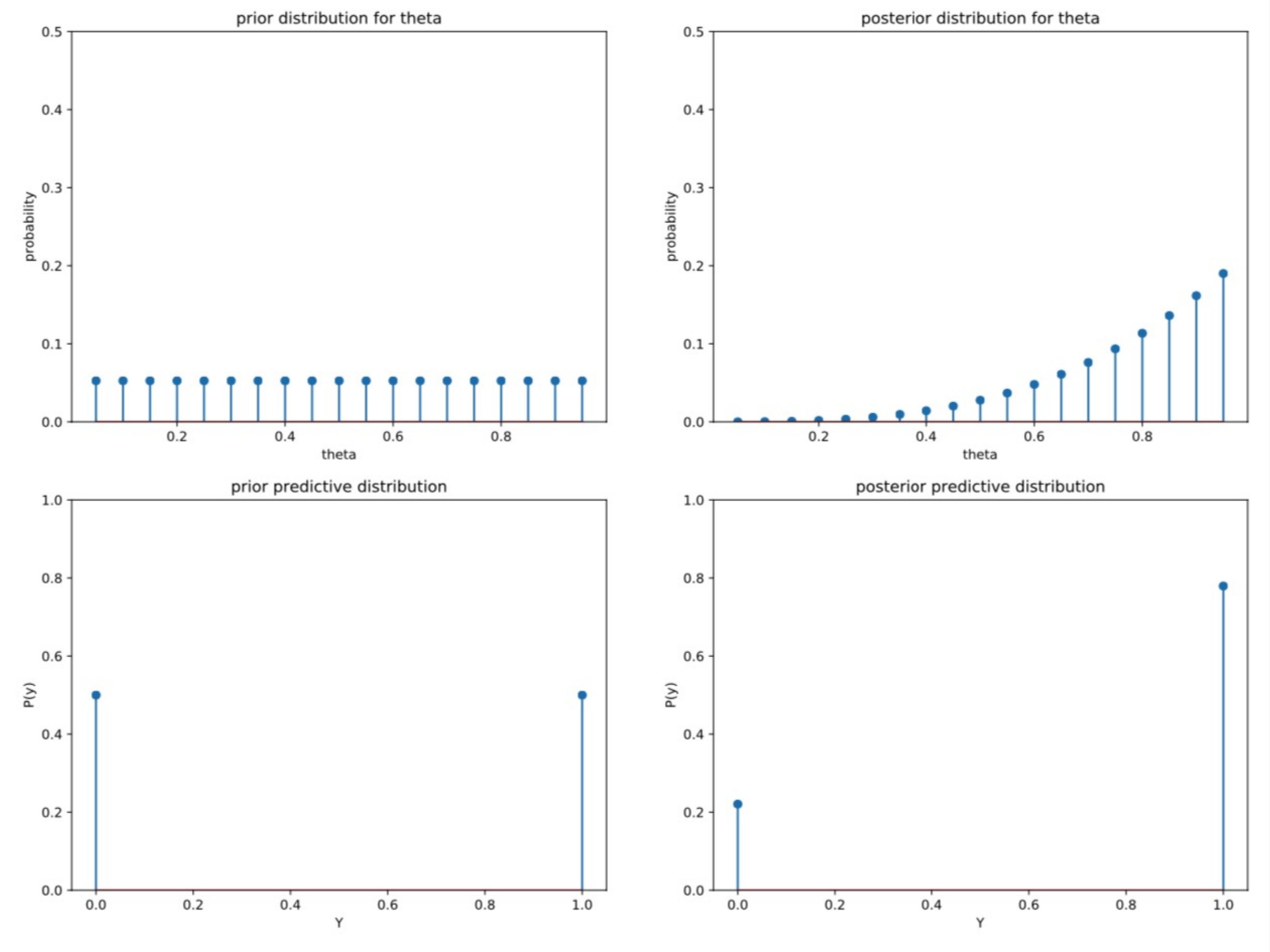
$$P(Y=1 \mid D) = \int \underbrace{P(\overset{\overset{1}{\shortparallel}}{Y} \mid \Theta)}_{\Theta} \underbrace{P(\Theta \mid D)}_{4\Theta^3} \, d\Theta$$

$$= \frac{4\Theta^5}{5}\Big|_0^1 = 0.8$$

Coin example «the hacker's way»

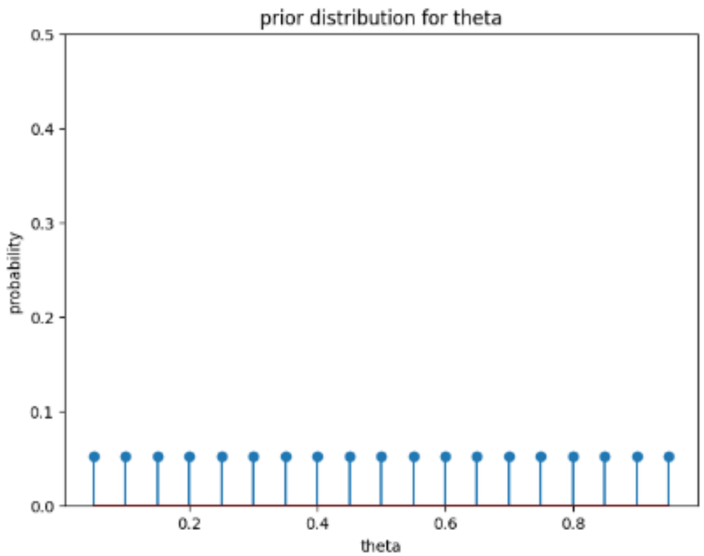
Work through the notebook NB21 and do the excerise therein.

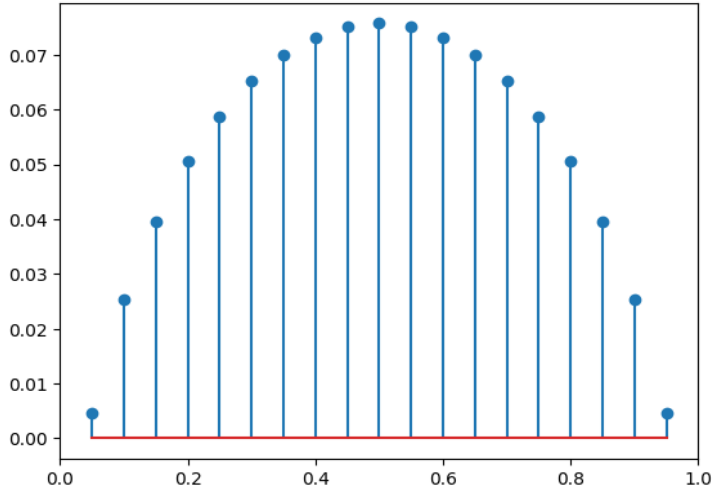
War als Hausi auf



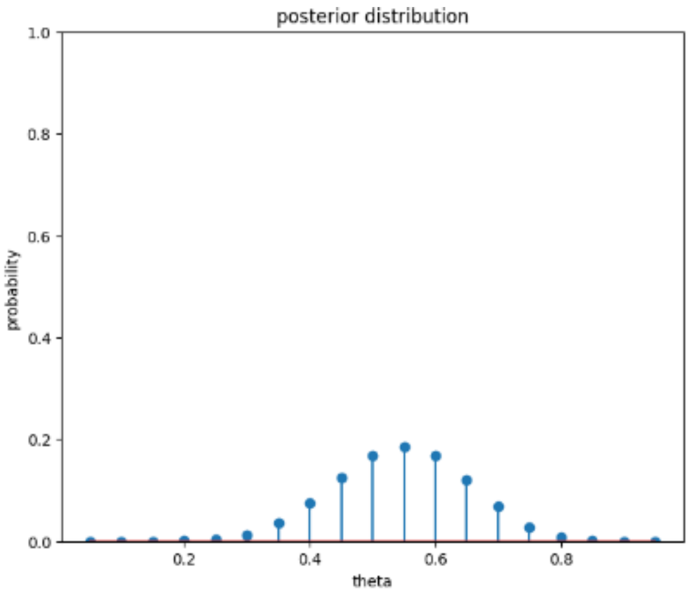
Result 40+11 times head and 9 tails

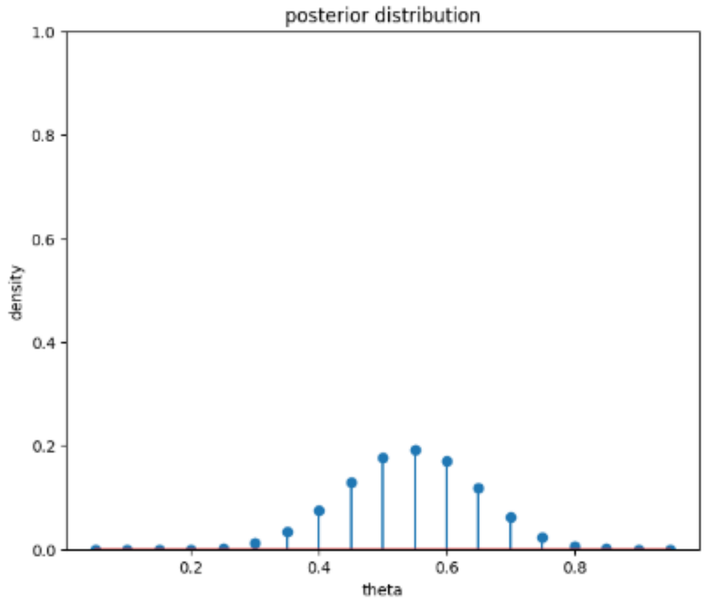
Prior





Posterior

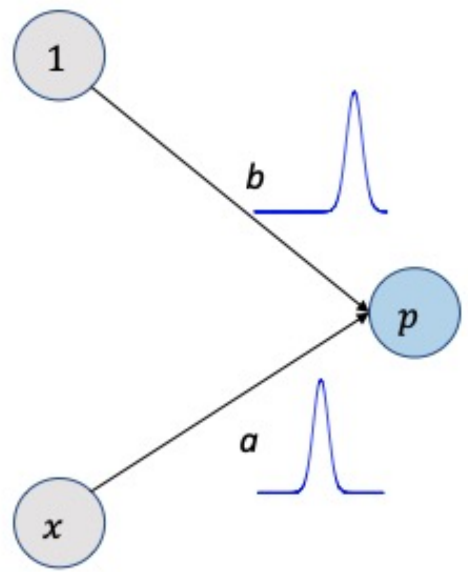


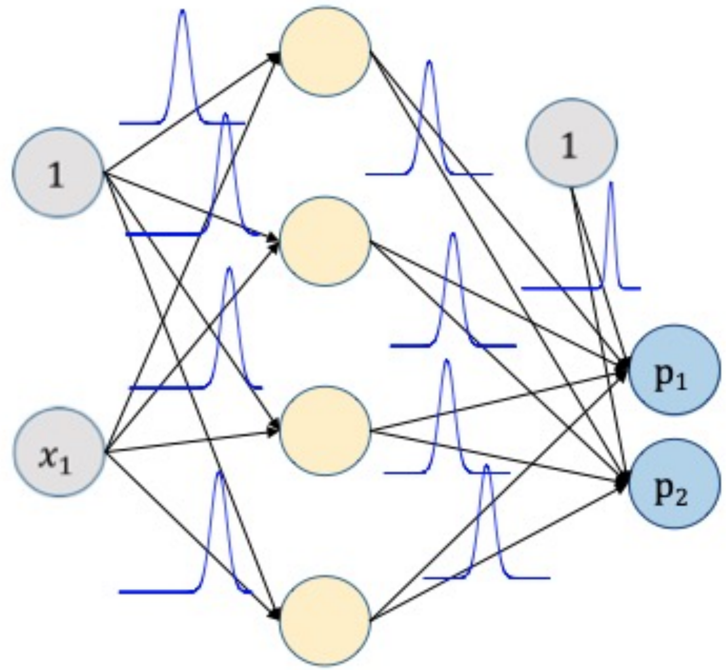


Bayesian Neural Networks

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 confuse Bayesian Networks (DAGs) with Bayesian Neural Networks

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)
- Computing $\sum_\theta p(D|\theta)p(\theta)$ is impossible for high-dimension θ

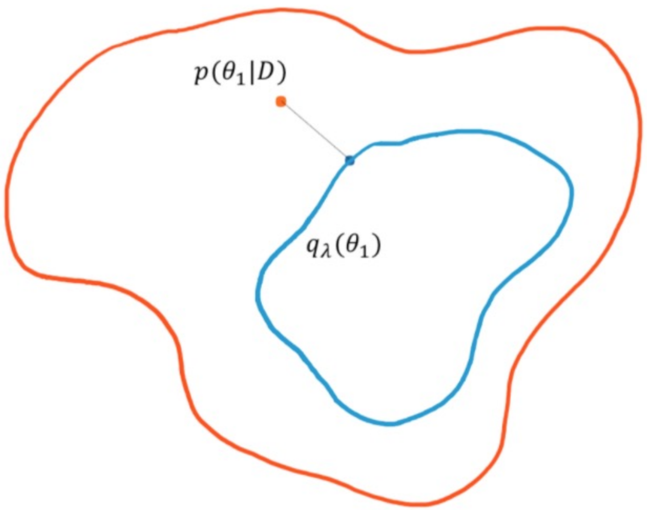
Approximations

- MCMC (only for very small NN feasible)
  - Sample from $p(\theta|D)$ with knowledge of $p(D|\theta')p(\theta')$ / $p(D|\theta'')p(\theta'')$
- Gaussian variational Inference VI
  - Approximate $p(w|D)$ by a Gaussian $N(\mu,\sigma)$ and tune μ,σ
- MC-Dropout
  - MC-Dropout during predictions (magically) samples from a variational approximation

Variational Inference

The principle of VI

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



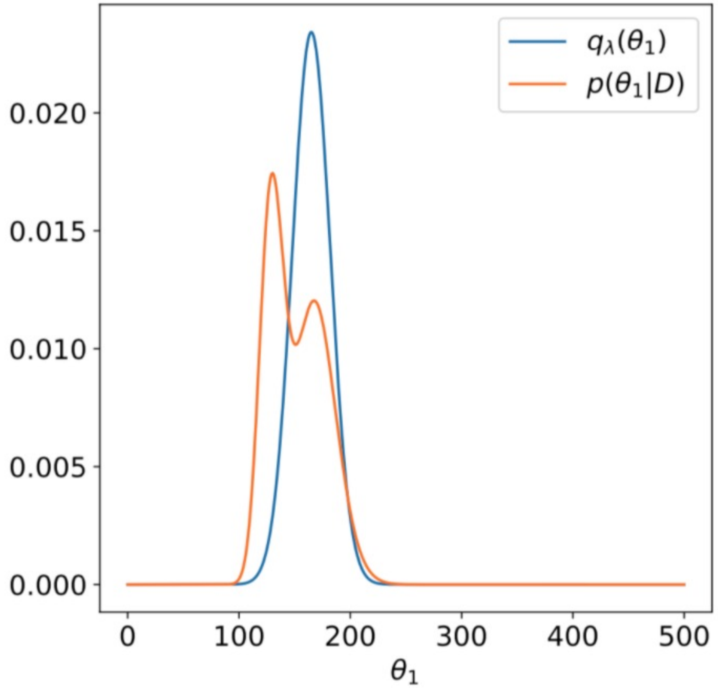


Figure 8.3 The principle idea of variational inference (VI). The larger region on the left depicts the space of all possible distributions, and the dot in the upper left represents the posterior $p(\theta_1|D)$ **(corresponding to the dotted density on the right panel). The inner region depicts the space of possible variational distributions** $q_\lambda(\theta_1)$**. The optimized variational distribution** $q_\lambda(\theta_1)$ **(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).**

Distance between two distributions

- To get $q_\lambda(\theta)$ close to $p(\theta|D)$ we need a distance

- Typical "Distance" is KL-Divergence

- "Distance" between two distributions $f(x)$ and $g(x)$

$$KL(f(x)||g(x)) = \int \log\left(\frac{f(x)}{g(x)}\right) f(x)dx = E_{x\sim f(x)}[\log\left(\frac{f(x)}{g(x)}\right)]$$

- Properties of KL-Divergence
  - KL ≥ 0
  - KL = 0 if $f(x) = g(x)$
  - $KL(f(x)||g(x)) \neq KL(g(x)||f(x))$ Not symmetrical not a real distance

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

Tradeoff of good fit (low NLL) and regularization small KL to prior

TF Particularies

- Layers for VI:
  - DenseReparameterization
  - Convolution{1D,2D,3D}Reparameterization
  - Further a method called Flipout to speed up training

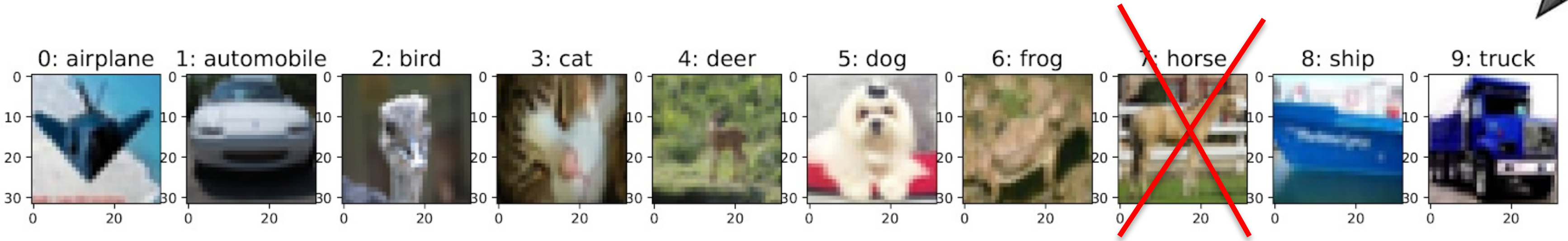
From documentation (Convolution2DFlipout)

> When doing minibatch stochastic optimization, make sure to scale this loss such that it is applied just once per epoch (e.g. if kl is the sum of losses for each element of the batch, you should pass kl / num_examples_per_epoch to your optimizer)
>
> num_examples_per_epoch = number of training data

```
kl = tfp.distributions.kl_divergence
divergence_fn=lambda q, p, _: kl(q, p) / (num * 1.0)

DenseReparameterization(1,kernel_divergence_fn=divergence_fn)
```

Hands-on Time cntd.: Fit the VI Bayesian NN



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.

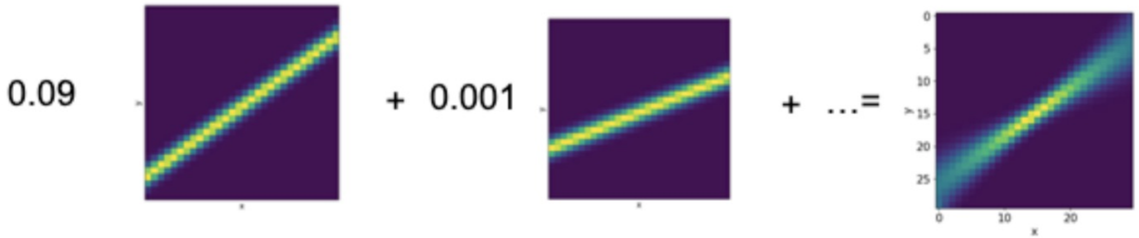
https://github.com/tensorchiefs/dl_course_2021/blob/master/notebooks/20_cifar10_classification_mc_and_vi.ipynb

Comparison

Bayes

Bayes:

- Averages a all possible solution weighted by using posterior weights



$p(y|x, D)$

Ensembling:

- just average a few possible solutions (obtained via SGD) without weights.

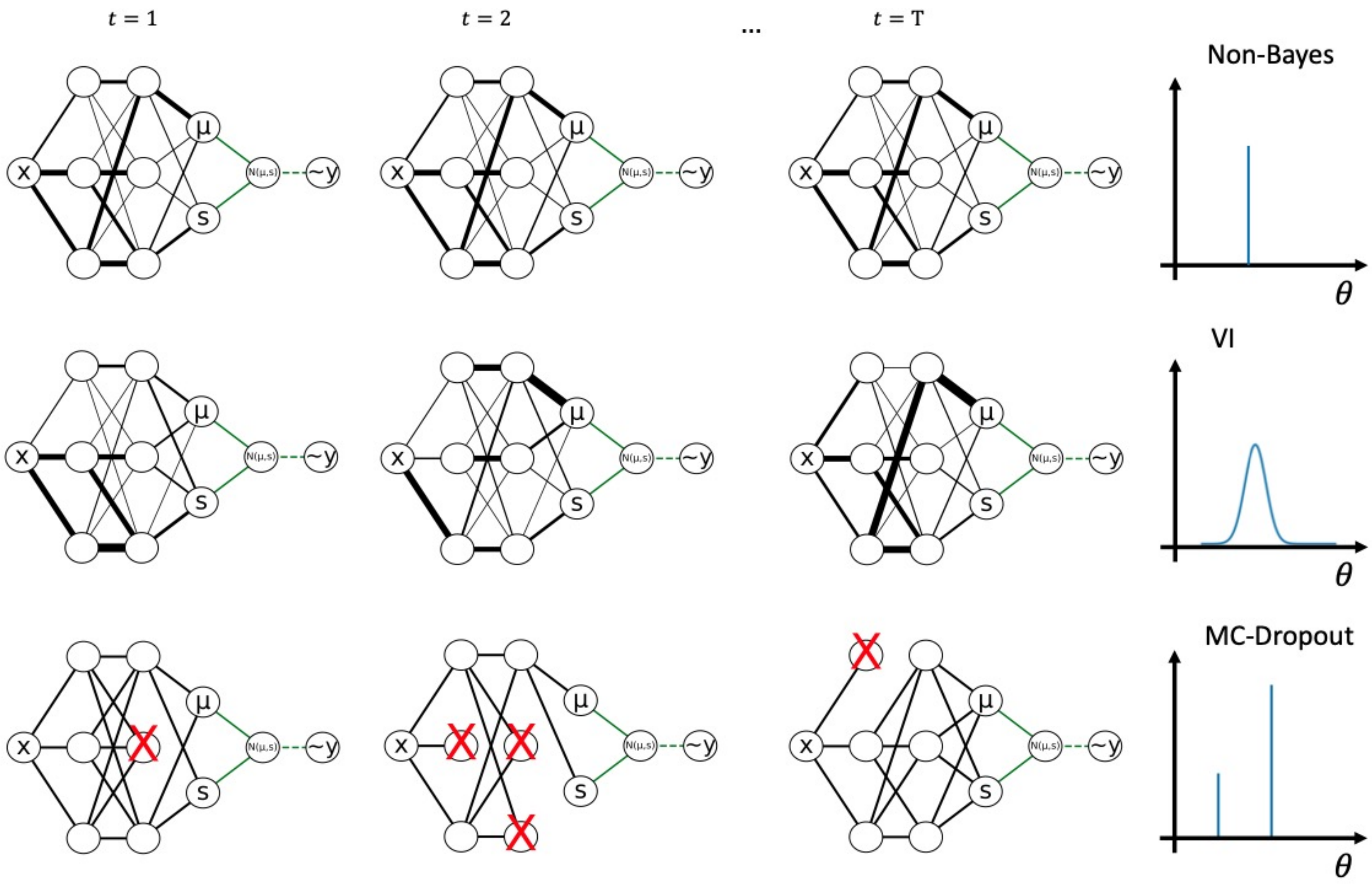
MC-Dropout:

- Averages over many possible solutions, can be seen as Bayesian. Paper called "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning"

VI:

- Clear Bayesian method approximates posterior.

Comparing different Network types



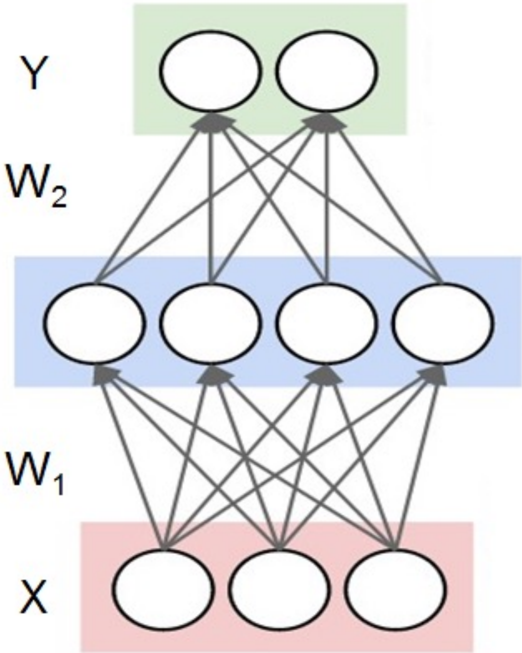
https://www.youtube.com/watch?v=mQrUcUoT2k4

A Non-Baysian NN learns one set of weights: the same input same output

A Bayesian NN learns distribution of weights: same input different outputs

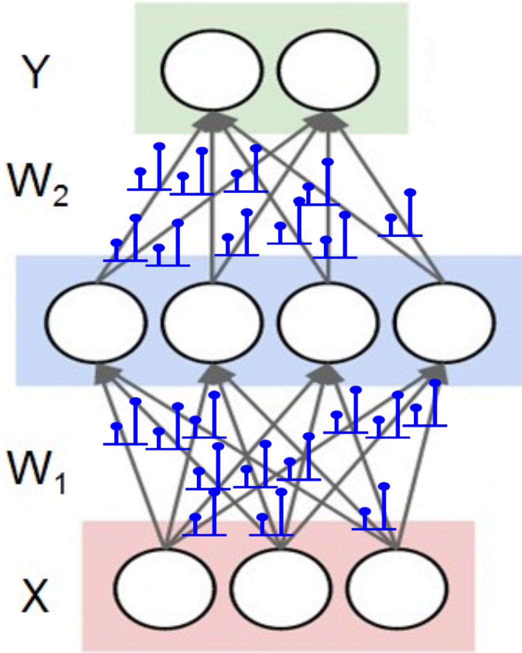
MC-Dropout vs VI

Non-Bayesian NN



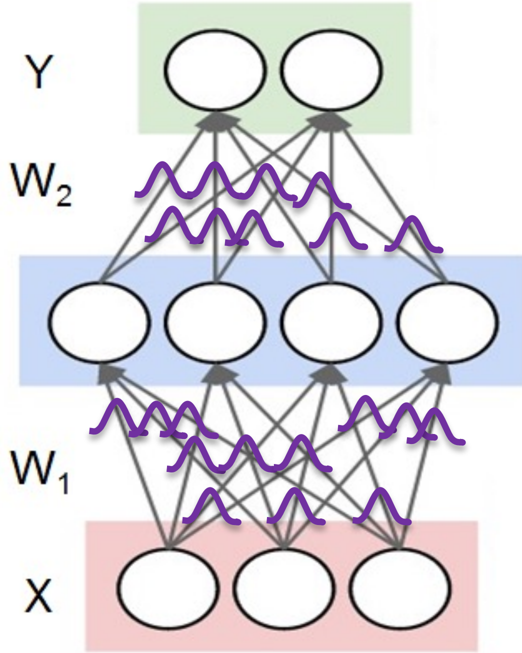
Weights are fixed

MC dropout Bayesian NN



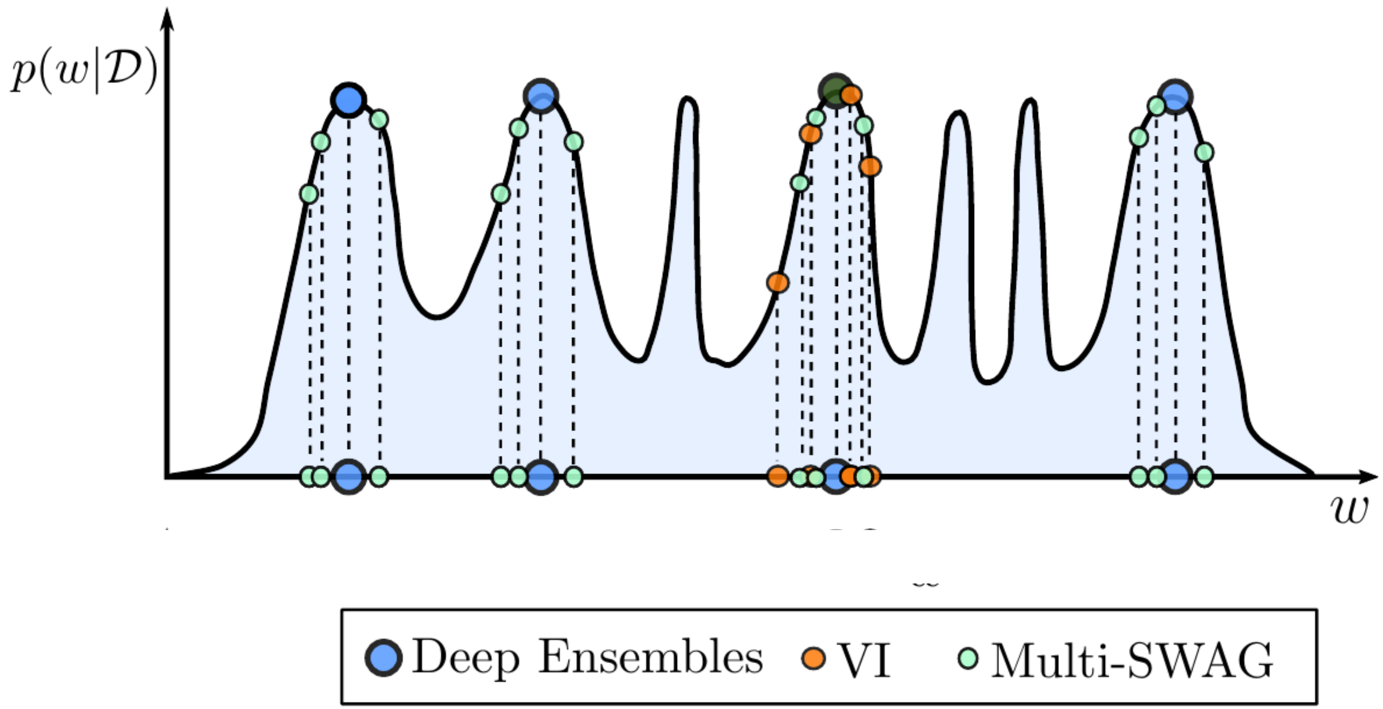
Weights have Bernoulli-kind distribution

VI Bayesian NN



Weights have Gaussian distribution

Comparison Ensembling vs. VI



Deep Ensembles as Approximate Bayesian Inference
https://cims.nyu.edu/~andrewgw/deepensembles/

A.G. Wilson, P. Izmailov. *Bayesian Deep Learning and a Probabilistic Perspective of Generalization*. Advances in Neural Information Processing Systems, 2020

"We Bayesians also have a not-so-secret super-weapon: we can take algorithms that work well, reinterpret them as approximations to some form of Bayesian inference, and voila, we can claim credit for the success of an entire field of machine learning as a special case of Bayesian machine learning. We are the BORG of machine learning." Quote from

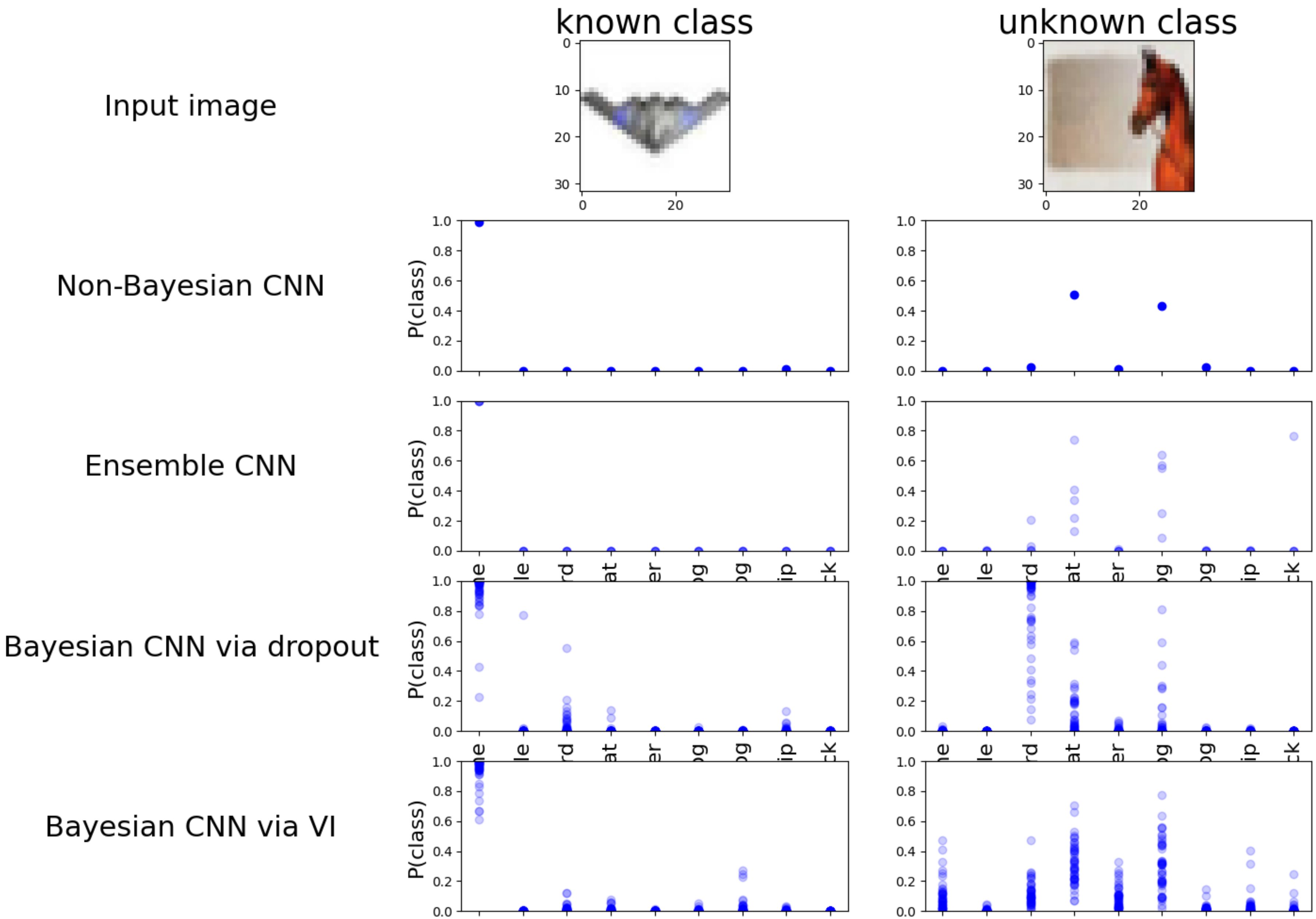
https://www.inference.vc/everything-that-works-works-because-its-bayesian-2/

Experimental Results

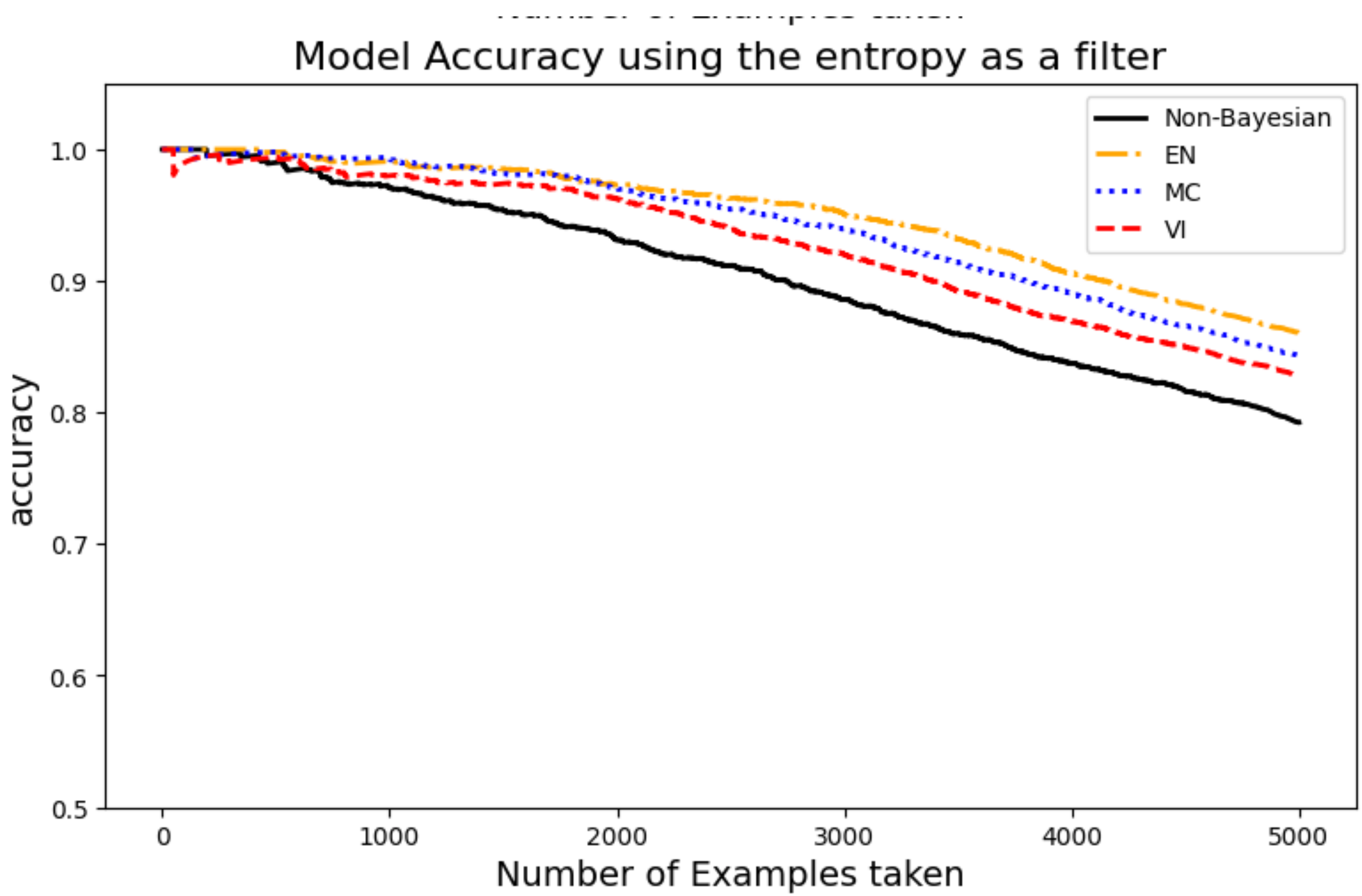
Predictive Performance (Notebook)

	Non-Bayesian	EN	MC	VI
test acc on known labels	0.649444	0.730889	0.706444	0.684444

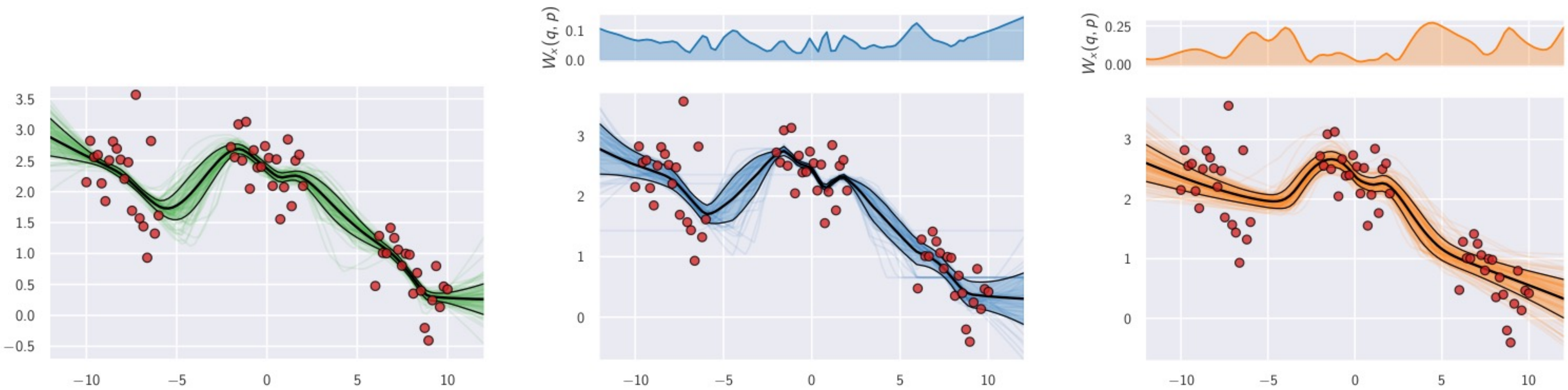
Looking at the predictive distribution!



Filter Experiment



Comparison Ensembling vs. VI



(a) Exhaustive HMC (b) Deep Ensembles (c) Variational Inference

See also https://cims.nyu.edu/~andrewgw/deepensembles/

A.G. Wilson, P. Izmailov. *Bayesian Deep Learning and a Probabilistic Perspective of Generalization*. Advances in Neural Information Processing Systems, 2020

Conclusion

- Standard neural networks (NNs) fail to express their uncertainty (can’t talk about the elephant in the room).

- The following Algorithms (can express their uncertainty and usually gain a higher predictive performance)

- Ensembling
  - Usually the best, however needs ~5 networks training of 5 networks
- MC dropout
  - Easy to implement, needs only one training
- VI (Bayesian by nature)
  - Clear Bayesian, needs a bit more effort in training
- Many other methods have been developed
  - Overview