Machine Intelligence:: Deep Learning
Week 7

Beate Sick, Elvis Murina, Oliver Dürr

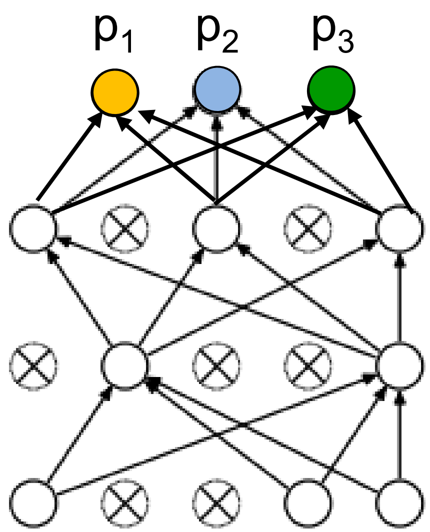
Institut für Datenanalyse und Prozessdesign
Zürcher Hochschule für Angewandte Wissenschaften

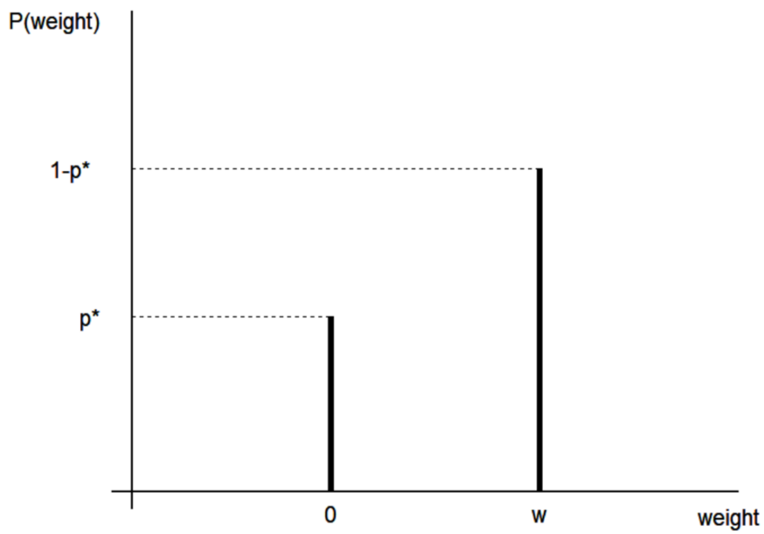
Part I: Preliminary, might change slightly before lecture

Winterthur, 7. April. 2020

Dropout

Recall: Classical Dropout only during training

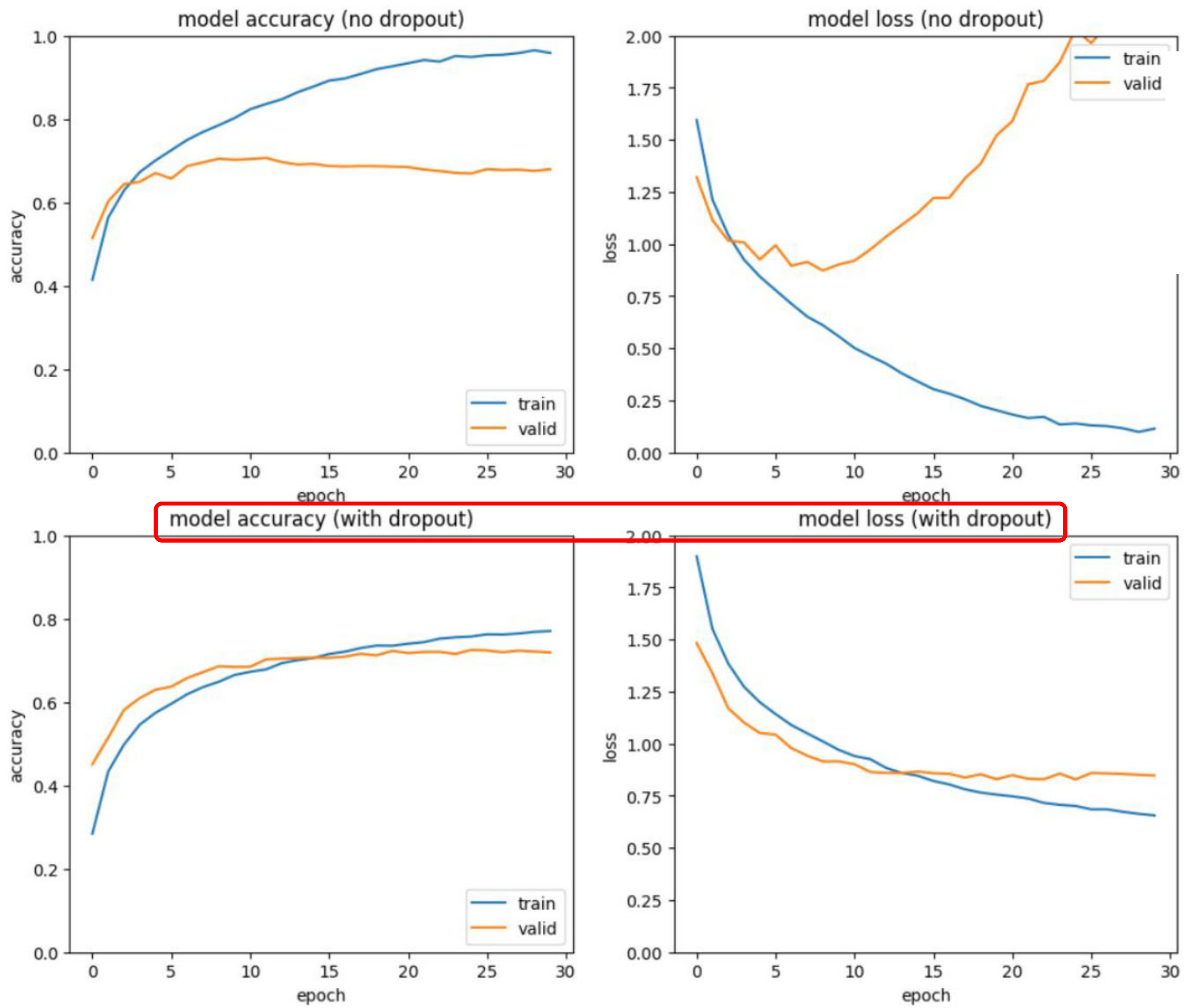


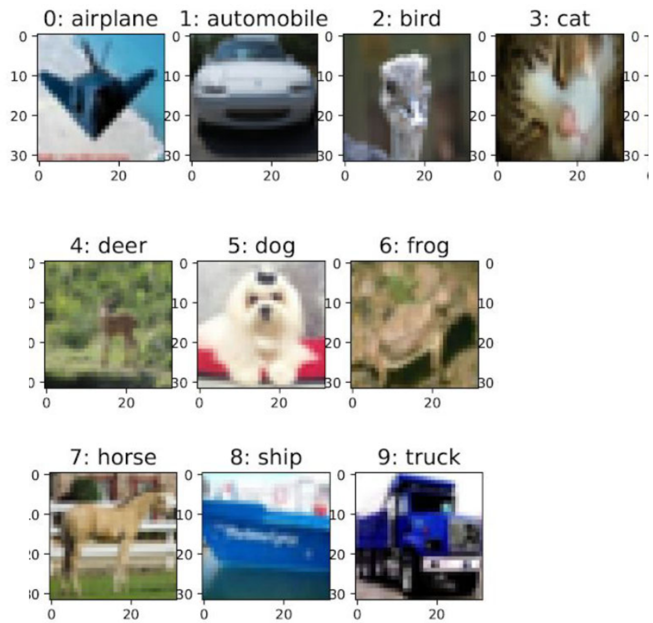


Using dropout during training implies:

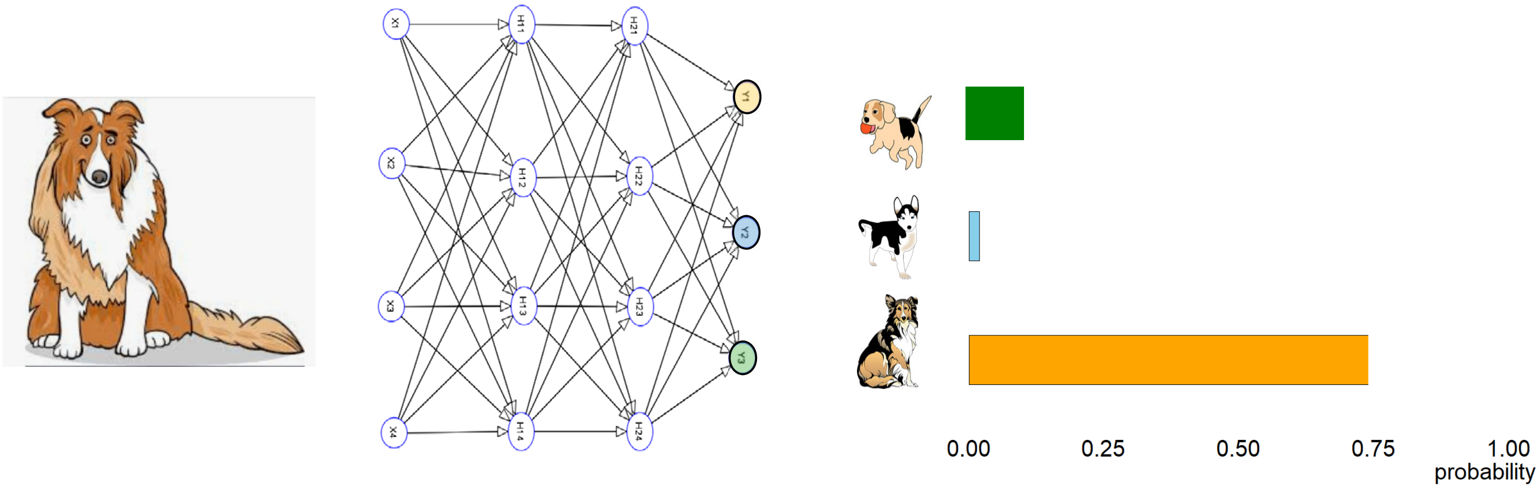
- In each training step only weights to not-dropped units are updated → we train a sparse sub-model NN
- For non-Bayesian NN we freeze the weights after training to a value $w \cdot p^*$

Recall: Dropout fights overfitting in a CIFAR10 CNN





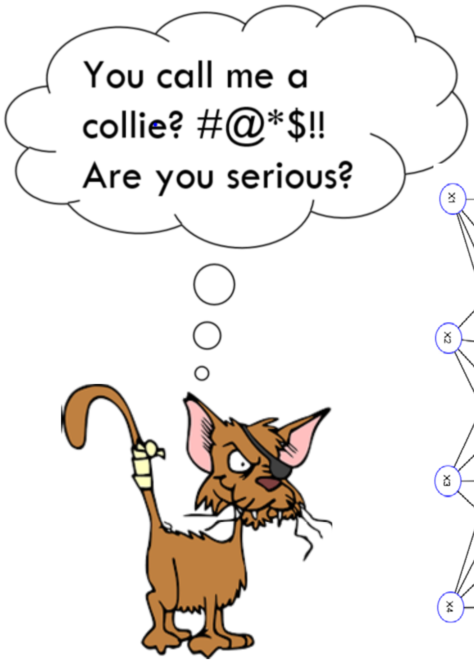
Recall: Nice properties of CNNs

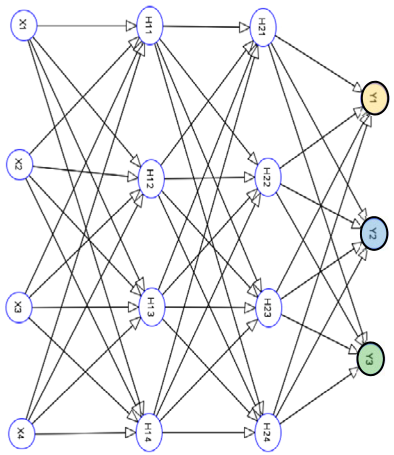


CNNs yield high accuracy and calibrated probabilities, but…

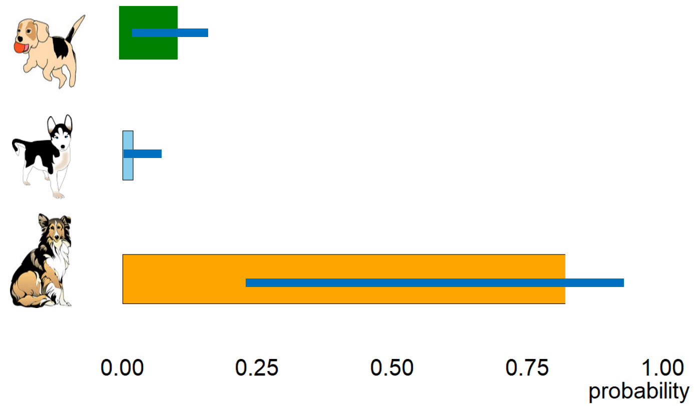
A non-Bayesian NN cannot ring the alarm

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





Plain wrong !



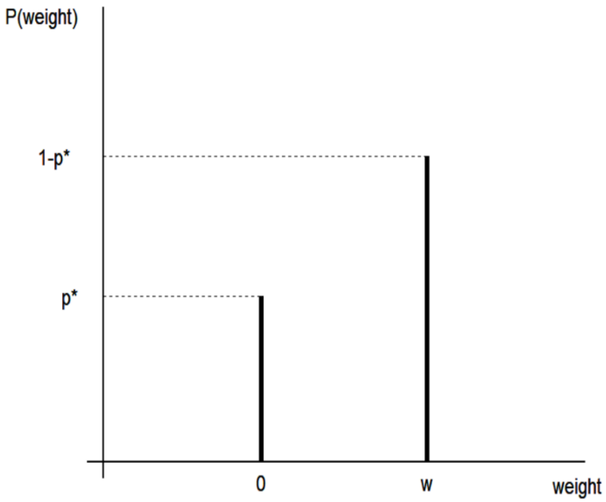
We need some error bars!

From Dropout during training
to MC Dropout during test time

Bayesian NN via MC Dropout

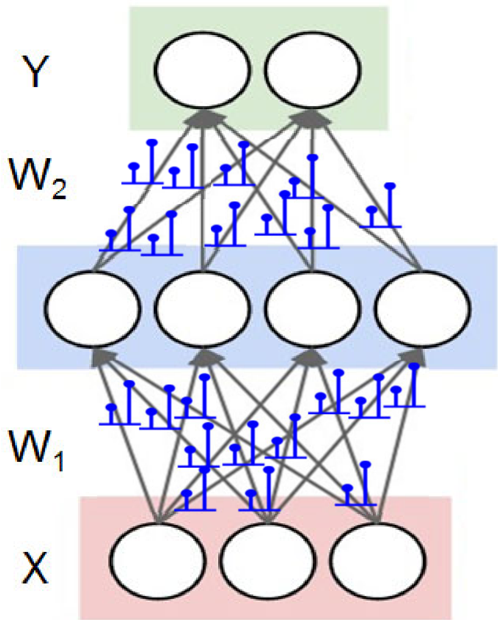
Yarin Gal et al. (2015):
Via Dropout training we learned a whole weight distribution for each connection. We can sample from this Bernoulli-kind distribution by performing dropout during test time and use the dropout-trained NN as Bayesian NN.
Gal showed that doing dropout approximates VI with a Bernoulli-kind variational distribution (instead of a Gaussian).

Learned
Bernoulli-kind distribution



Dropout in
test time

MC dropout NN

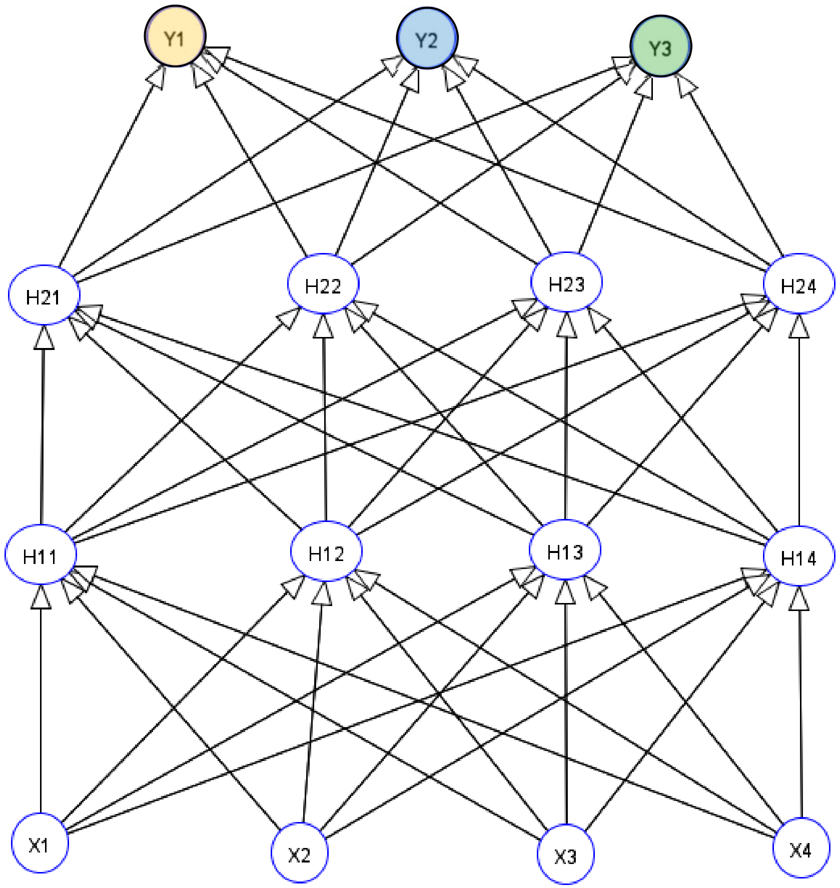


Weights have
Bernoulli-kind distribution

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:

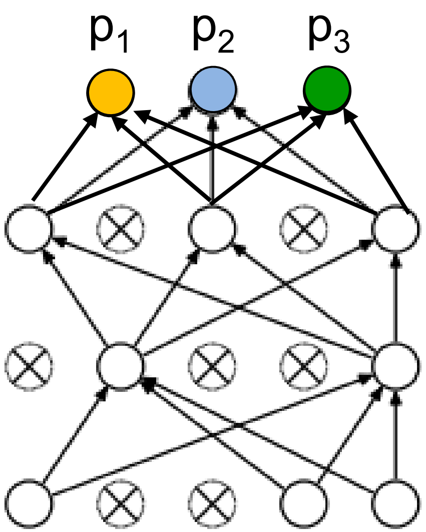
p_1=0.07 **p_2=0.91** p_3=0.02

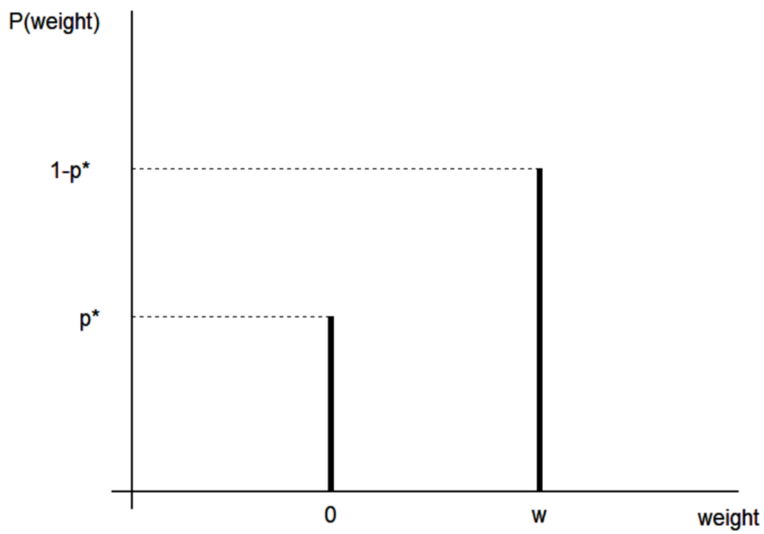


Probability of predicted class: $\mathbf{p_{max}}$

Input: image pixel values

Recall: Classical Dropout only during training

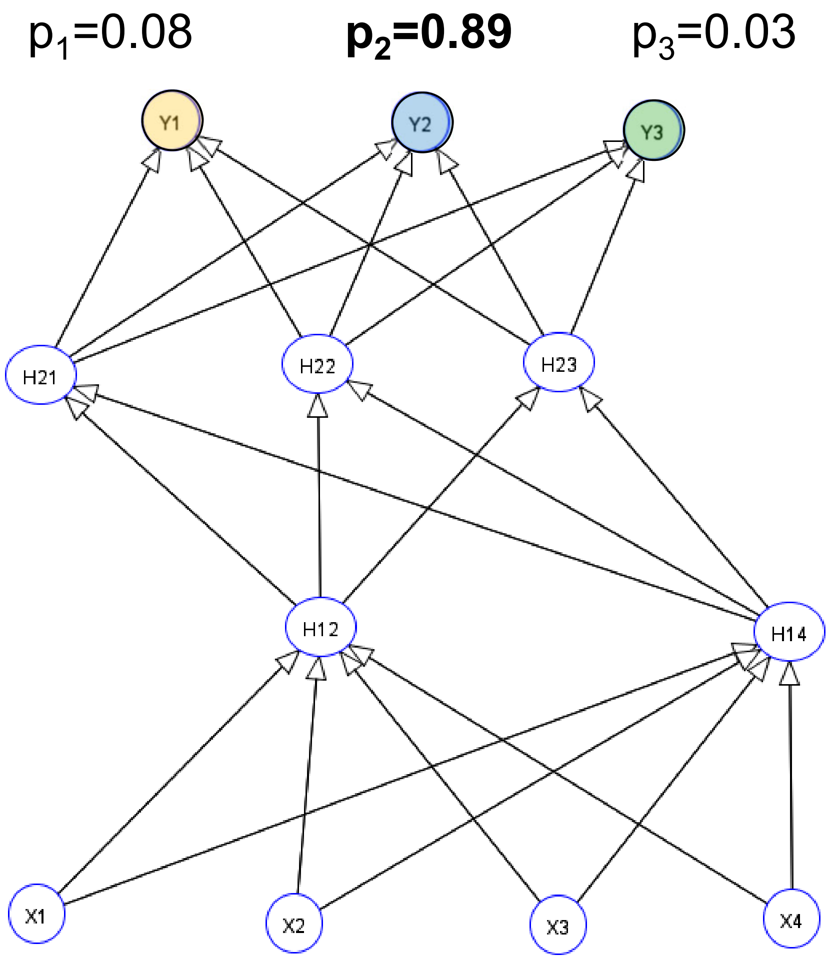


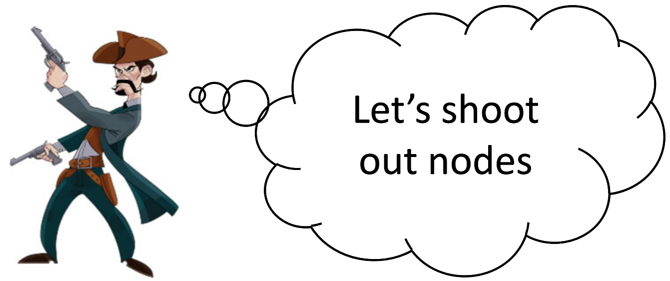


Using dropout during training implies:

- 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 $w \cdot p^*$

MC Dropout during test time: Run 1

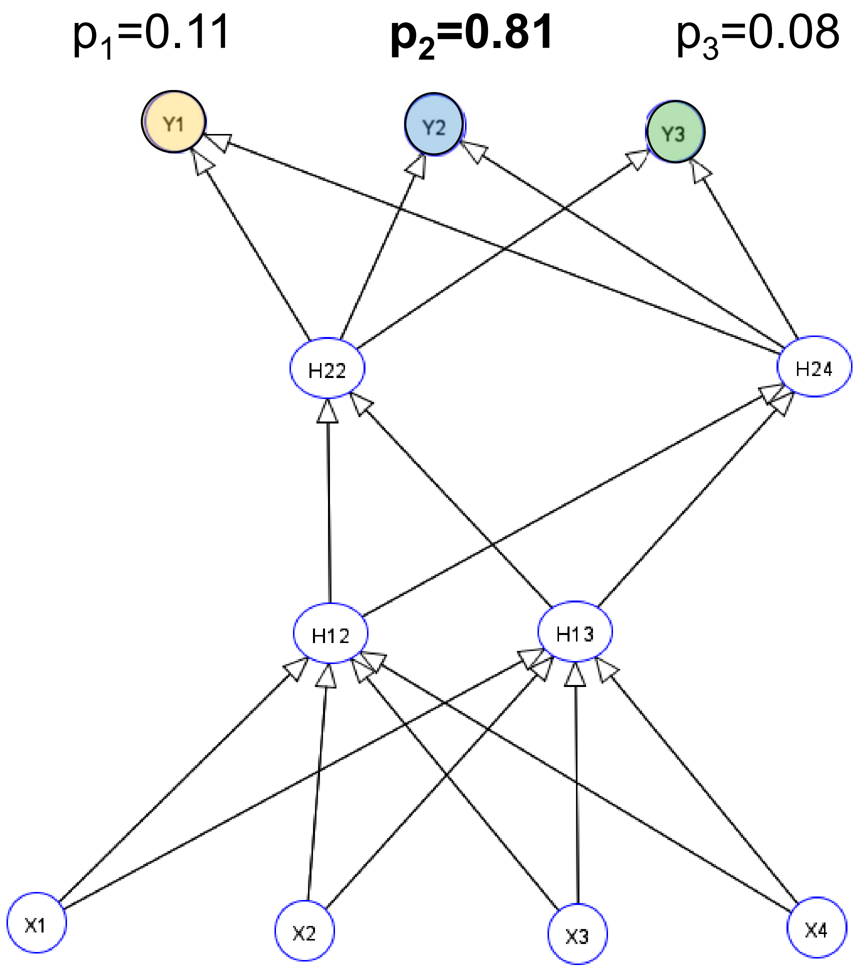


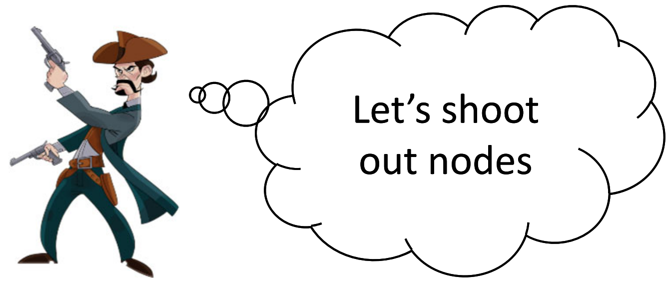


Stochastic dropout of units

Same input image

MC Dropout during test time: Run 2

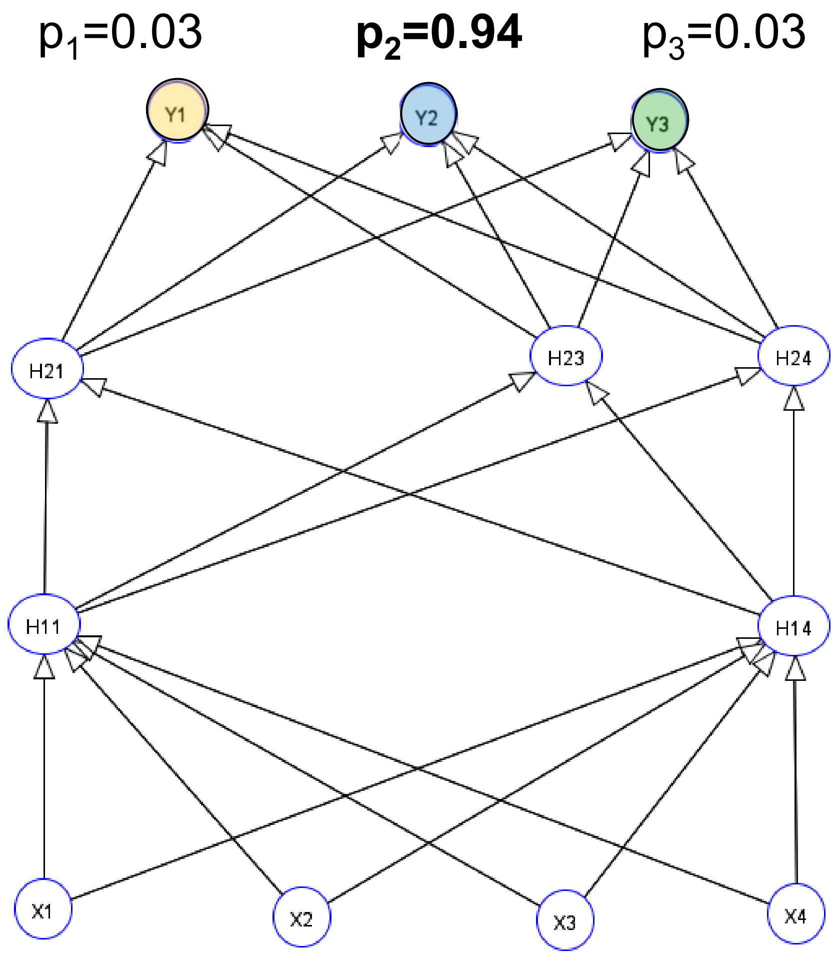


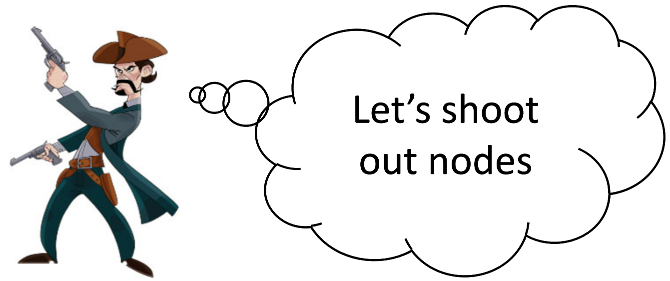


Stochastic dropout of units

Same input image

MC Dropout during test time: Run 3

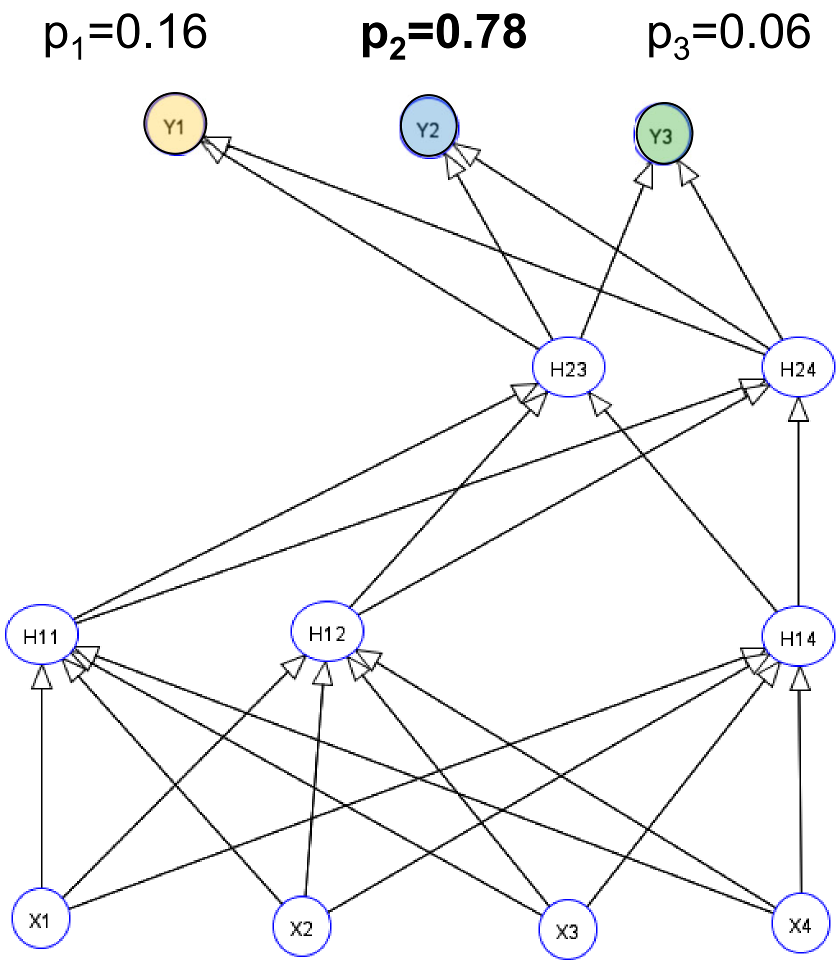


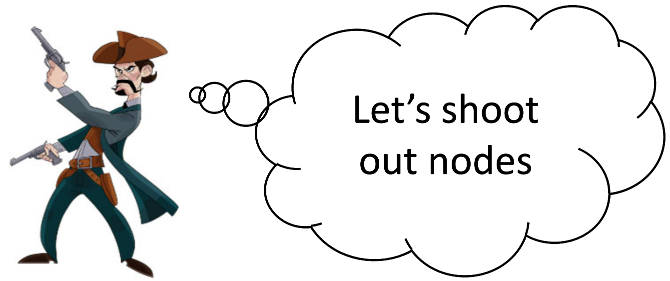


Stochastic dropout of units

Same input image

MC Dropout during test time: Run 4



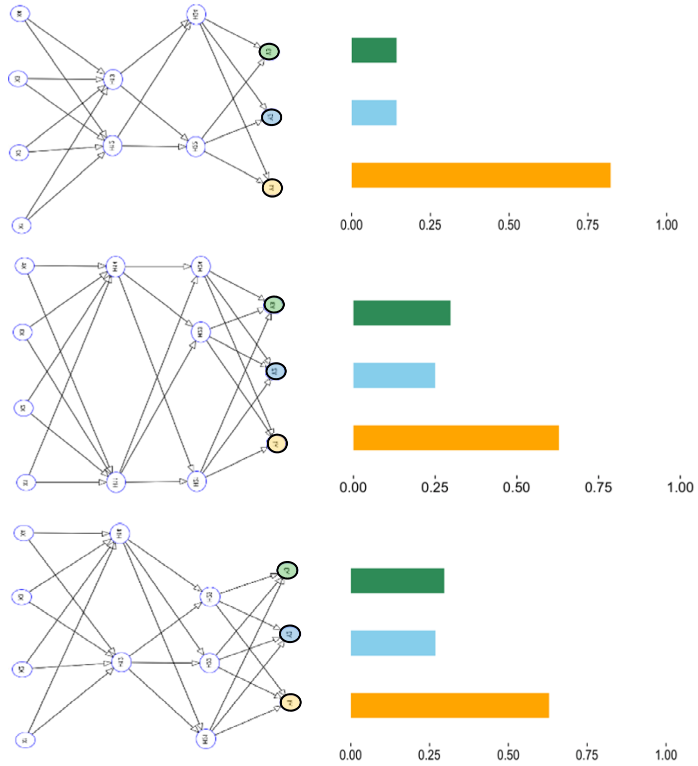


Stochastic dropout of units

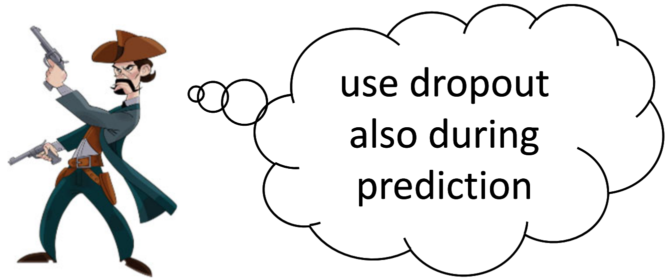
Same input image

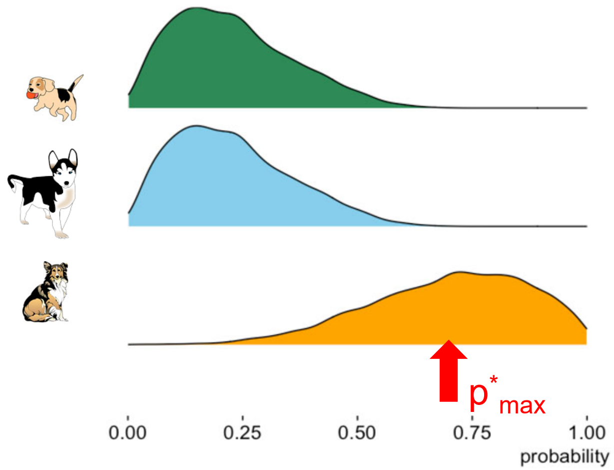
MC Dropout during test time yields a multivariate predictive distribution for the parameters

Many Dropout Runs in forward pass



...

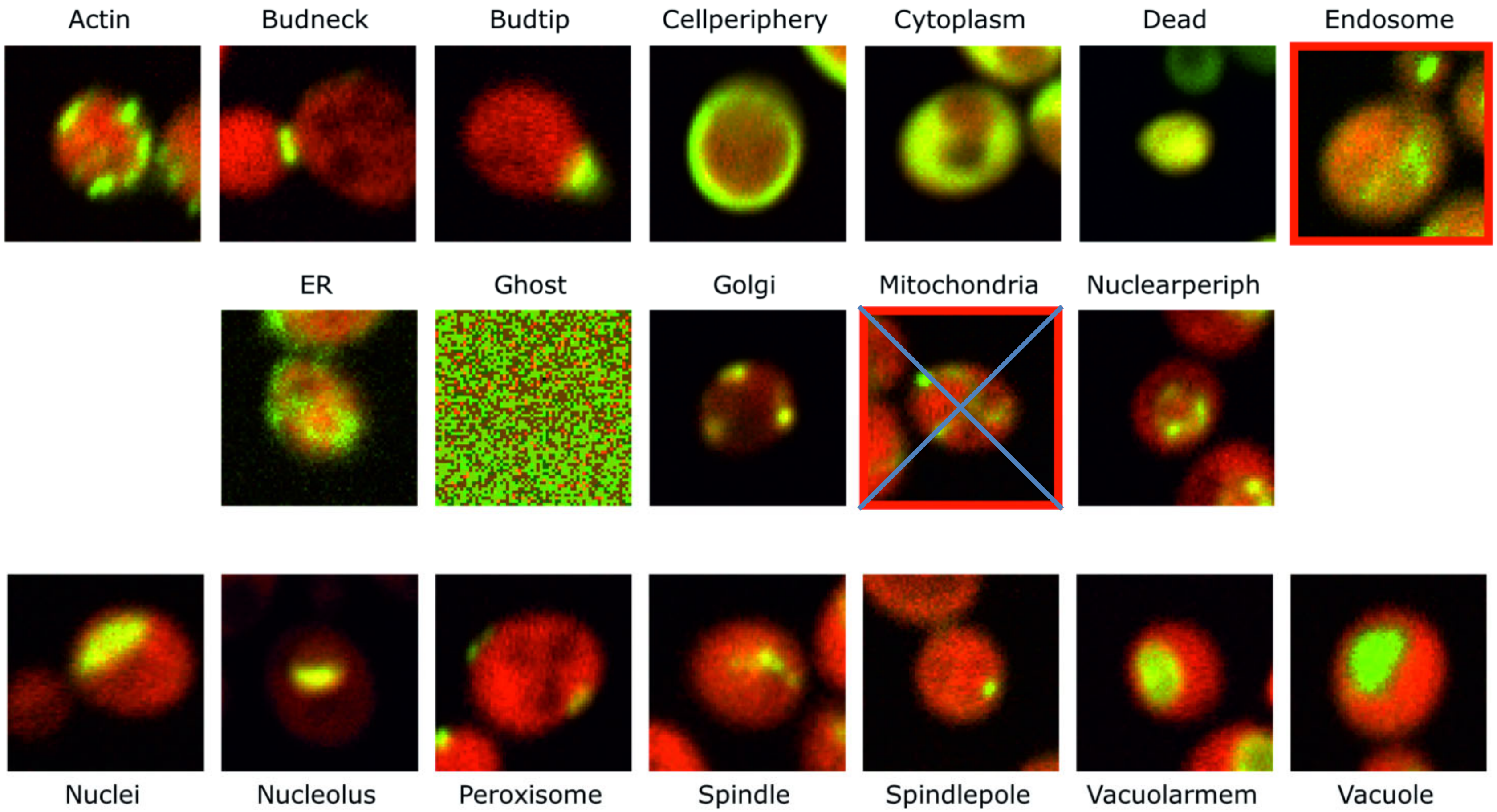




CNN predicts class “collie”
but with high uncertainty

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

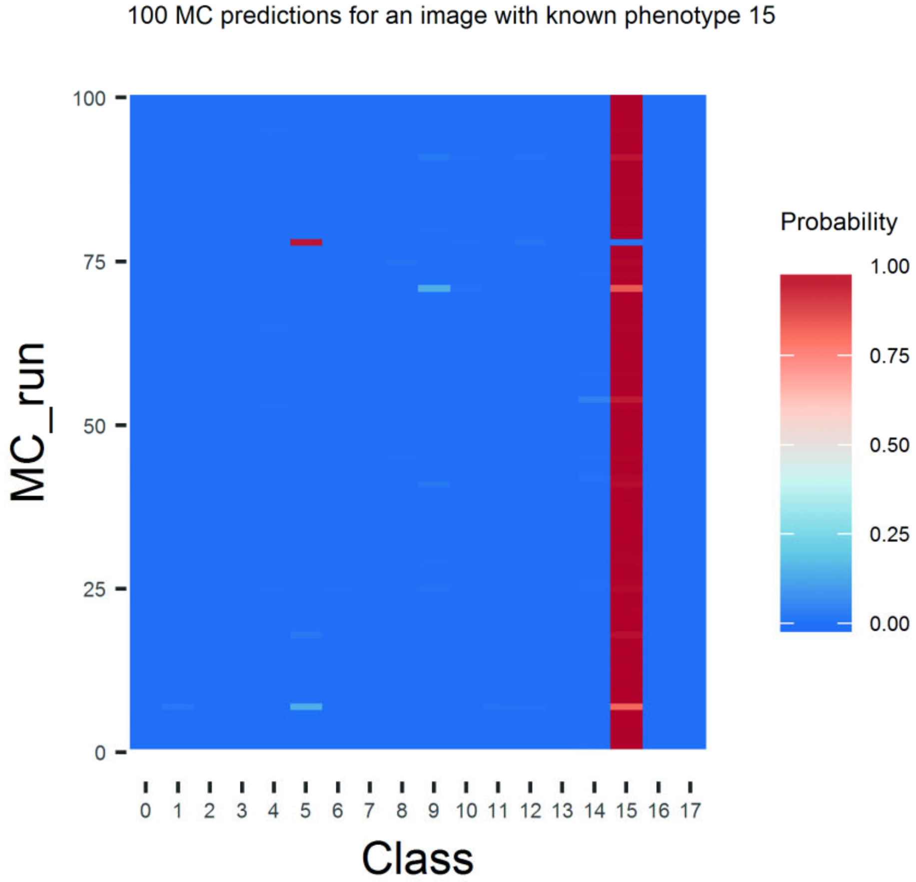
Experiment with unknown phenotype



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

Probability distribution from MC dropout runs

Image with known class 15



Probability distribution from MC dropout runs

Image with known class 15

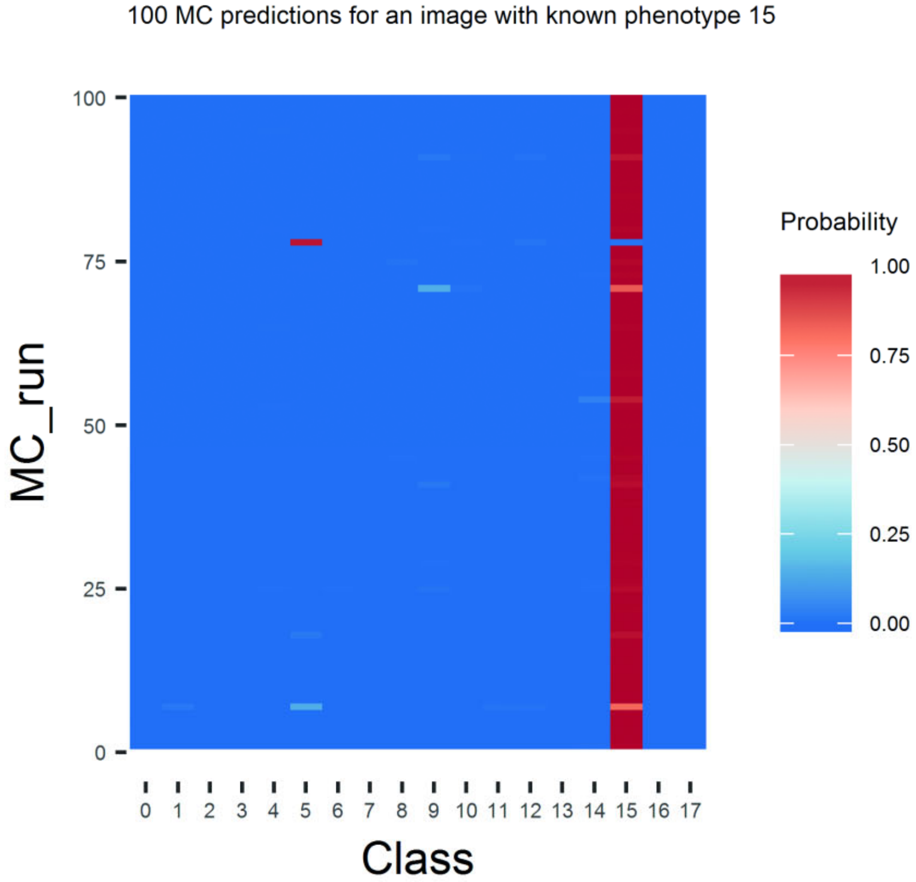
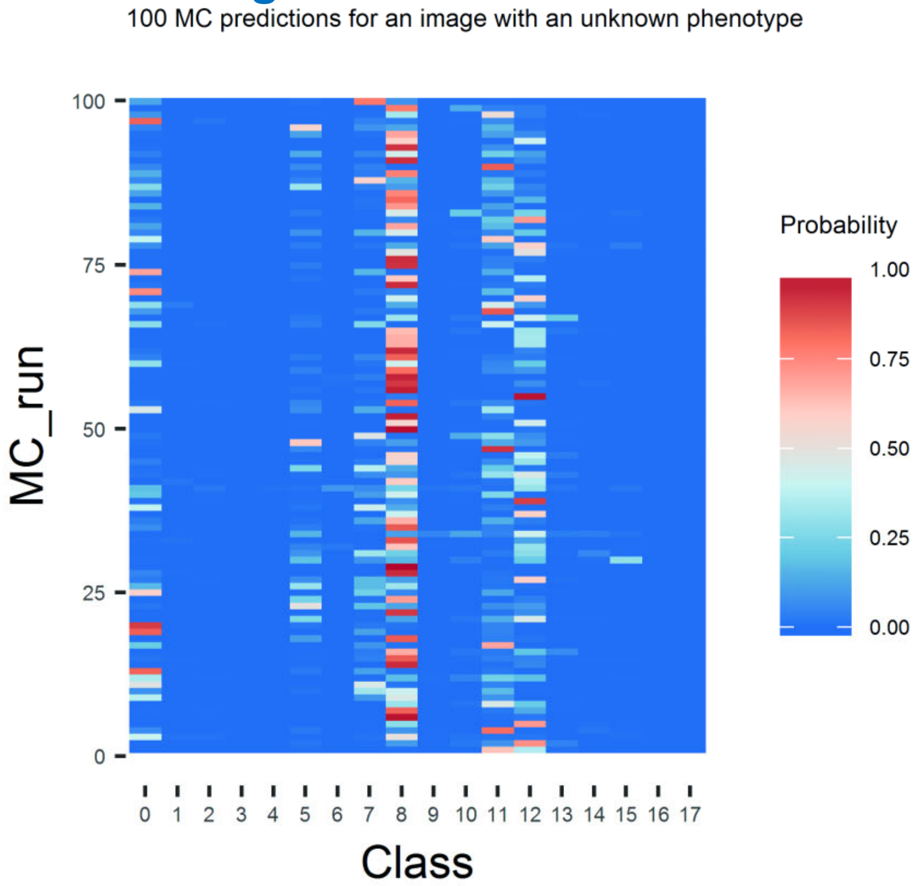


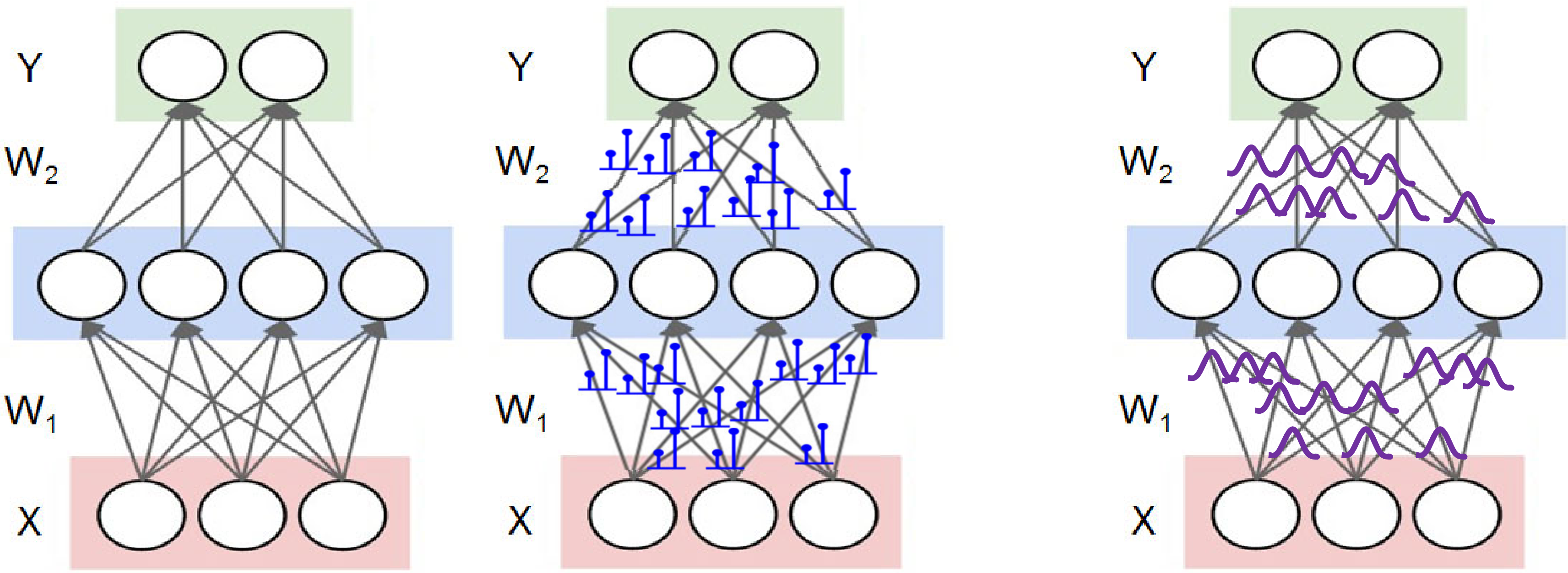
Image with unknown class



Comparing non-Bayesian with Bayesian NN

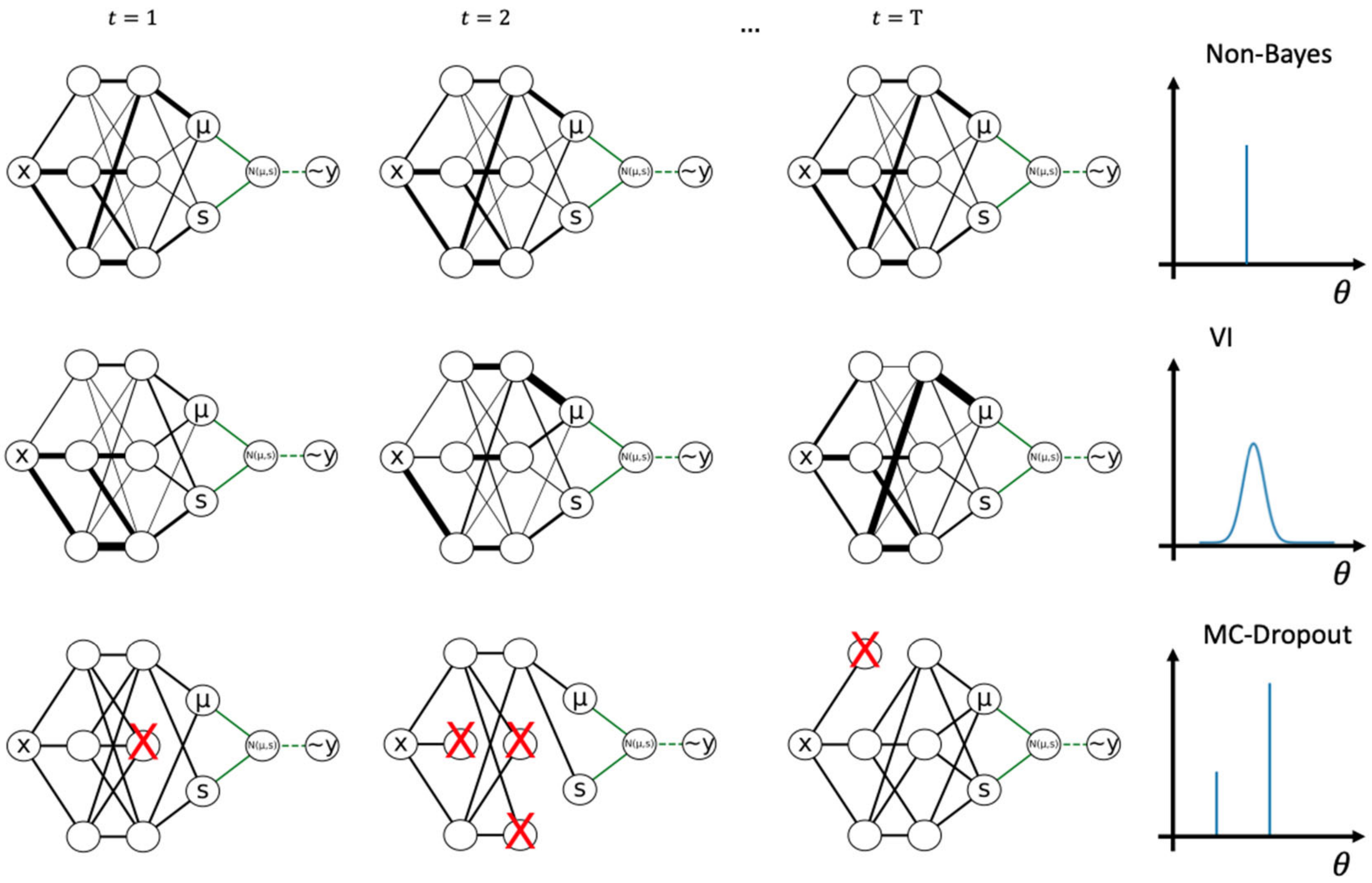
Non-Bayesian and Bayesian NNs

Non-Bayesian NN | MC dropout Bayesian NN | VI Bayesian NN



Weights are fixed | **Weights have Bernoulli-kind distribution** | **Weights have Gaussian distribution**

Comparing different Network types

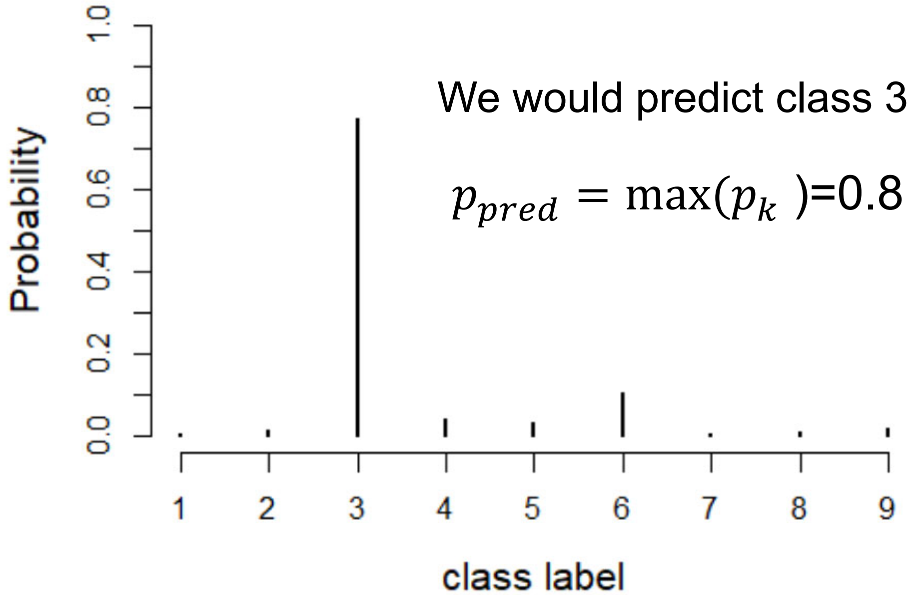


A Non-Baysian NN learns one set of weights: the same input same output
A Bayesian NN learns distribution of weights: same input different outputs

Uncertainty measures in classification

Uncertainty in non-Bayesian classification

Multinomial CPD
$MN(p_1(x,w), p_2(x,w), \ldots, p_9(x,w))$



We would predict class 3

$p_{pred} = \max(p_k\ )$=0.8

In a non-Bayesian NN we make for each input x ONE CPD:

Image x
MN(p1(x,w), …, p9(x,w))

Uncertainty measures capturing the **aleatoric** uncertainty :

Negative log-Likelihood: $NLL = -\log(p_{pred})$

Entropy: $H = -\sum_{k=1}^{9} p_k \cdot \log(p_k)$

Uncertainty in Bayesian classification

In a Bayesian NN we sample T-times from the weight distributions and get each time a slightly different multinomial CPD

predict_no	**Image x**
1	MN(p1(x,w1), …, p9(x,w1))
2	MN(p1(x,w2), …, p9(x,w2))
…	
T	MN(p1(x,wT), …, p9(x,wT))

For each class k ($k \in \{1,2,\dots,9\}$) we determine the mean probability: $p_k^* = \frac{1}{T}\sum_{i=1}^{T} p_{k\,i}$

The predicted class has the highest mean probability: $p_{pred}^* = \max(p_k^*)$

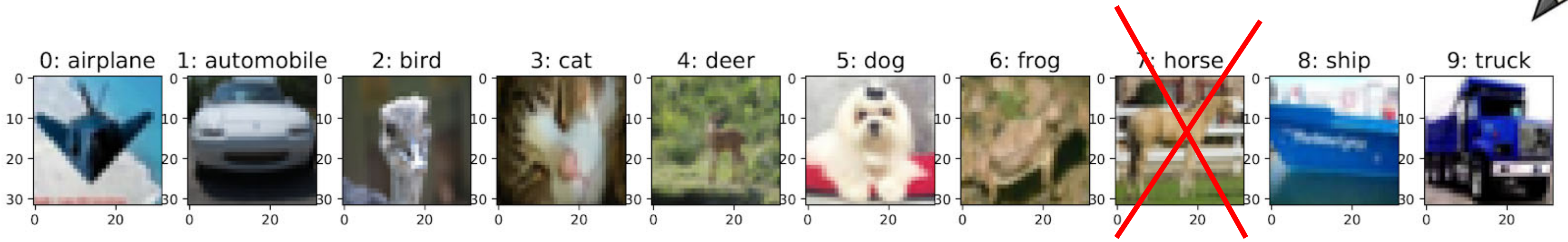
Uncertainty measures including **aleatoric and epistemic** contributions:

Negative log-Likelihood: $NLL^* = -\log(p_{pred}^*)$

Entropy: $H^* = -\sum_{k=1}^{9} p_k^* \cdot \log(p_k^*)$

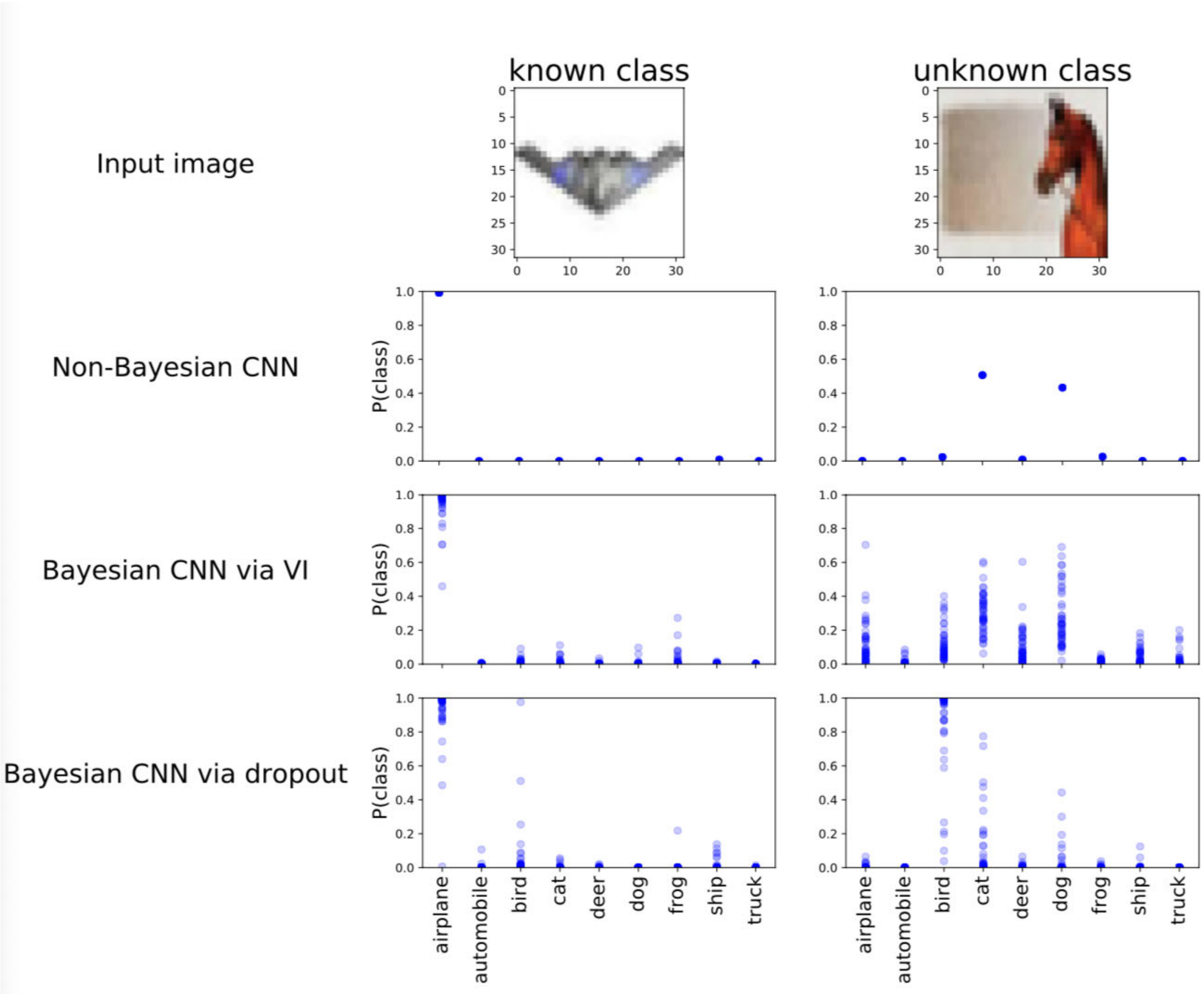
Total variance: $V_{tot} = \sum_{k=1}^{9} var(p_k) = \sum_{k=1}^{9}\sum_{i=1}^{T}\left(p_{kt} - p_k^*\right)^2$

Hands-on Time

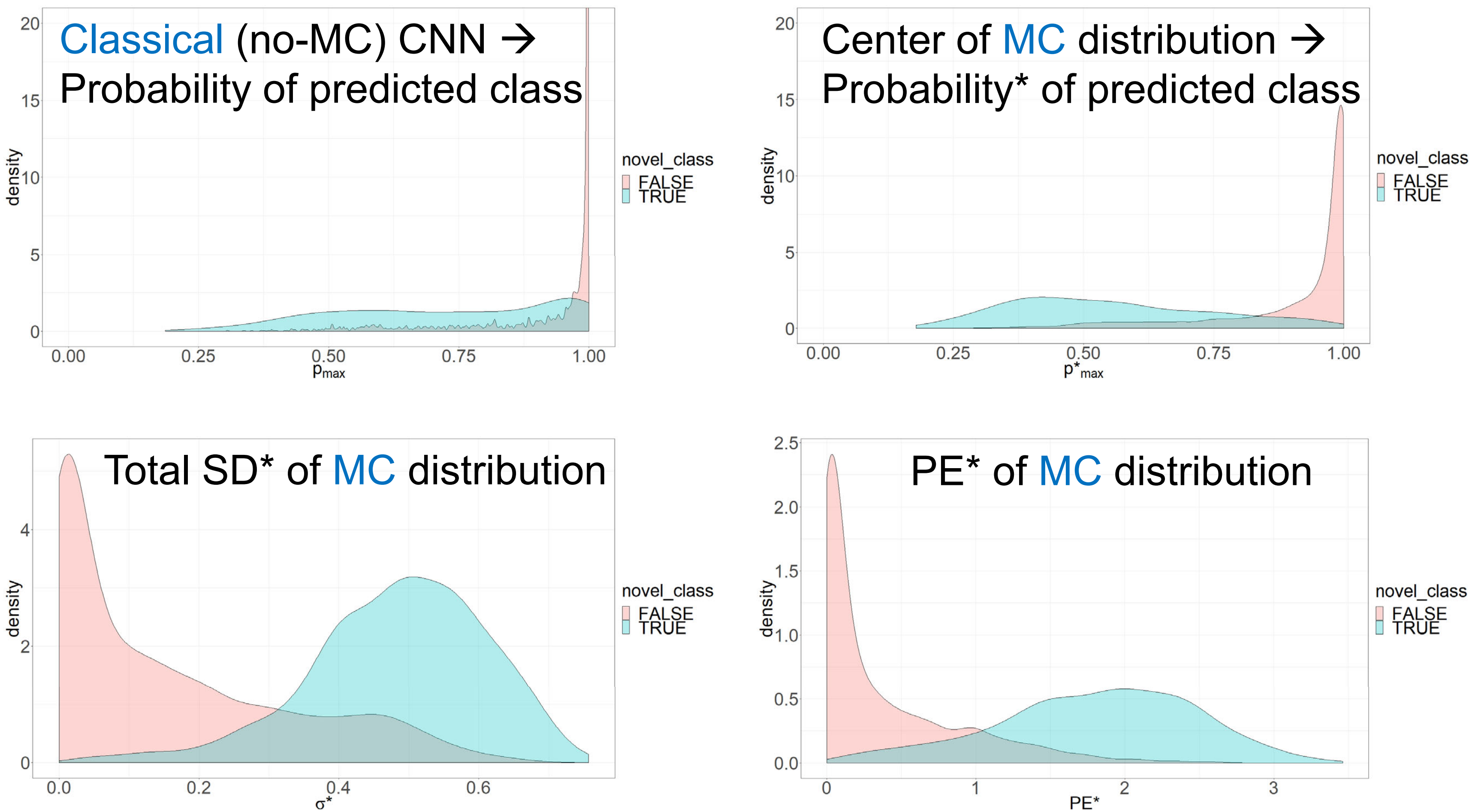


Train a CNN with only 9 of the 10 classes and investigate if the uncertainties are different when predicting images from known or unknown classes.

Looking at the predictive distribution!



Do known/novel classes yield different values for probability and uncertainty measures?

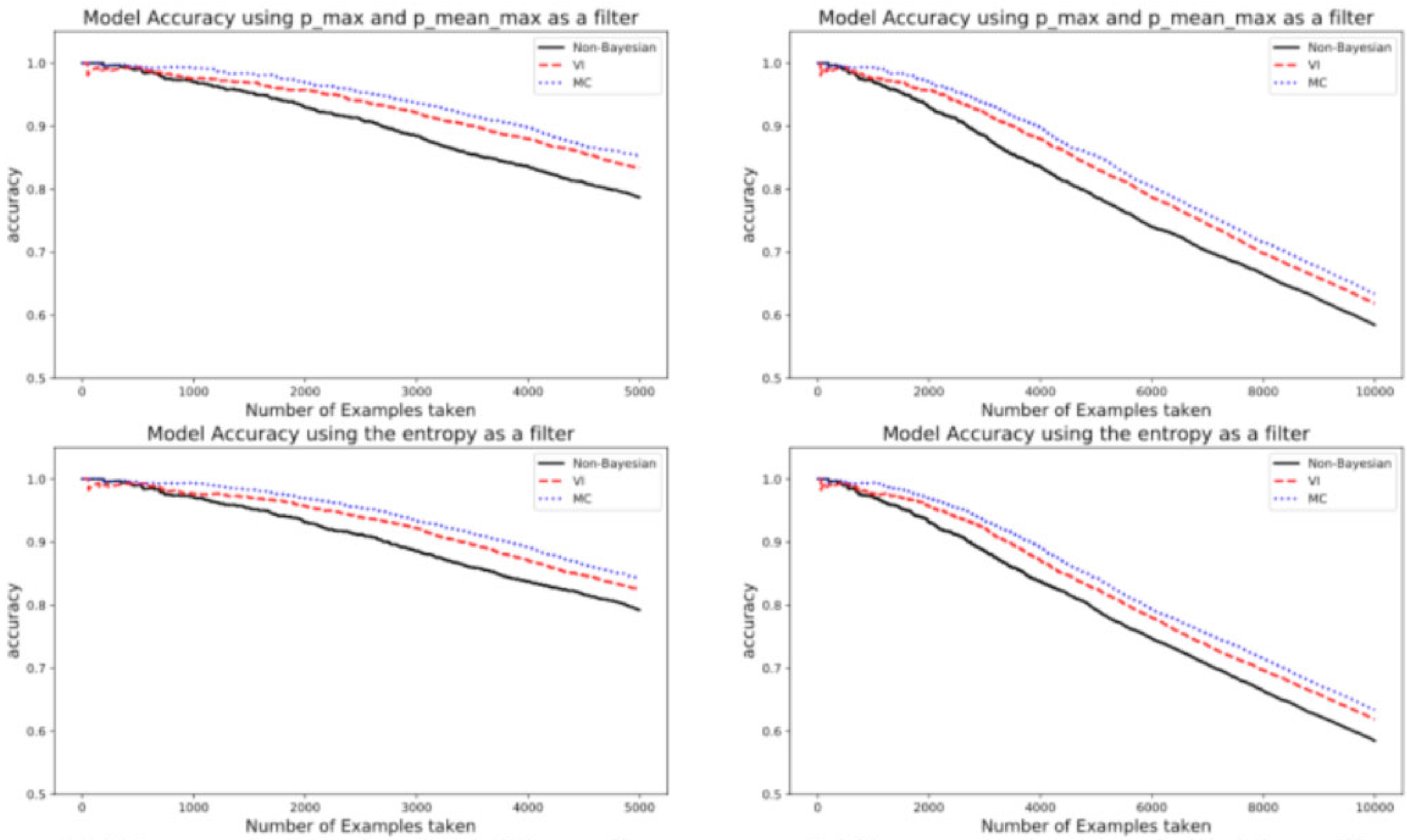


Filtering experiment based on uncertainty

Goal: Get higher accuracy by filter only predictions which are quite certainly correct

- Each prediction has an attached uncertainty measure
- Sort predictions according to the uncertainty measures
- The prediction with lowest uncertainty should yield highest prediction performance
- By successively adding predictions with increasing uncertainties should yield an decreasing prediction performance (e.g. accuracy)

Filtering experiment to compare uncertainty measures



Uncertainty from non-Bayesian NN is less good in identifying wrong classifications than uncertainty measures from Bayesian variants of the NN.

Uncertainty measures in regression

Uncertainty in non-Bayesian NN

We do predictions for 400 x-values between -10 and 30 yielding for each x a Gaussian CPD.

x1= -10	x2= -9.9	...	x400= 30
$N\left(\mu_{x_{1,w}}, \sigma_{x1,w}\right)$	$N\left(\mu_{x_{2,w}}, \sigma_{x2,w}\right)$		$N\left(\mu_{x_{400,w}}, \sigma_{x400,w}\right)$

Uncertainty measures capturing the **aleatoric** uncertainty at $\boldsymbol{x}$:

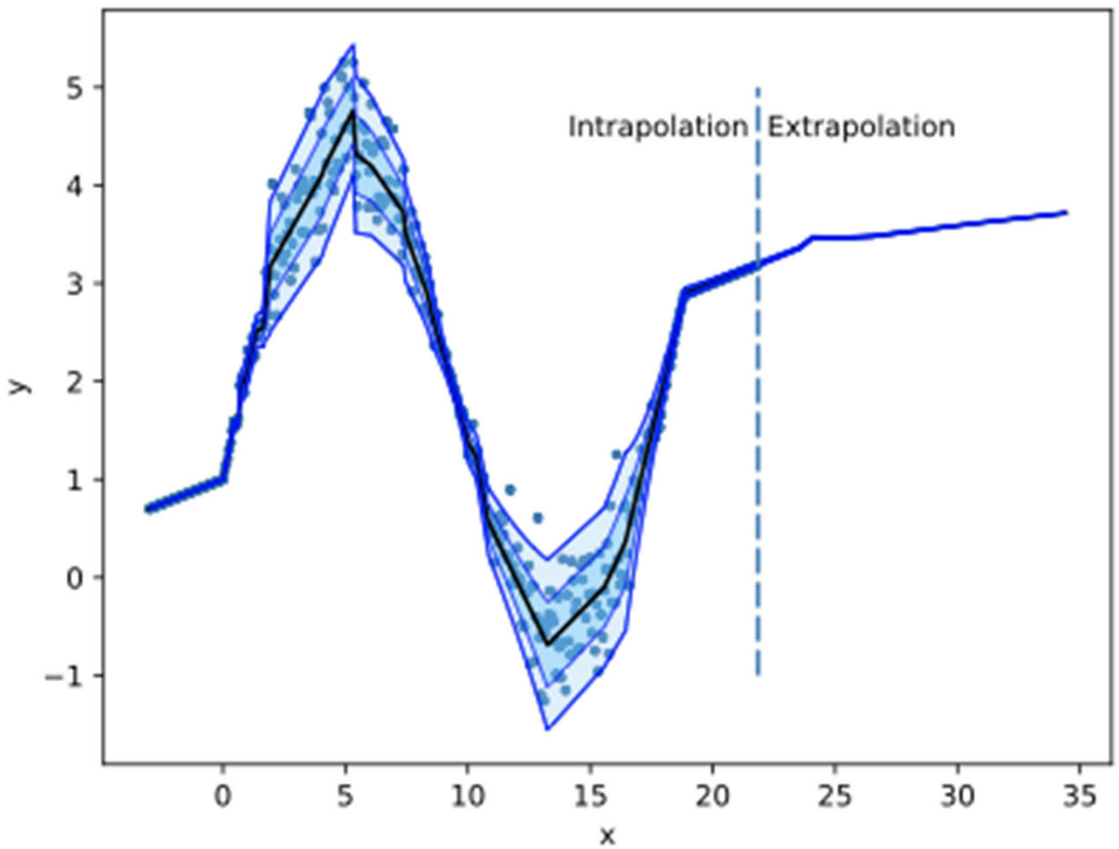
Standard deviation: σ_x

95% CI: $\mu_x \pm 1.96 \cdot \sigma_x$

Remark:
We could also estimate the 95% CI at position x by sampling several times from the CPD and determine the 0.025 and 0.975 quantiles, yielding :
95% CI: $[q_{0.025}; q_{0.975}]$

The problem of non-Bayesian NN

Problem:
A non-Bayesian NN does extrapolation with very small uncertainty



Uncertainty in Bayesian regression NN

In a Bayesian NN we sample T-times from the weight distributions and get each time a slightly different CPD. In regression the CPD is often Gaussian.

We do predictions for 400 x-values between -10 and 30 yielding in each of the T runs a different Gaussian CPD at each x-position.

predict_no	x1= -10	x2= -9.9	**...**	x400= 30
1	N(x1,w1,x1,w1)	N(x2,w1,x2,w1)		N(x400,w1,x400,w1)
2	N(x1,w2,x1,w2)	N(x2,w2,x2,w2)		N(x400,w2,x400,w2)
...				
T	N(x1,wT,x1,wT)	N(x2,wT,x2,wT)		N(x400,wT,x400,wT)

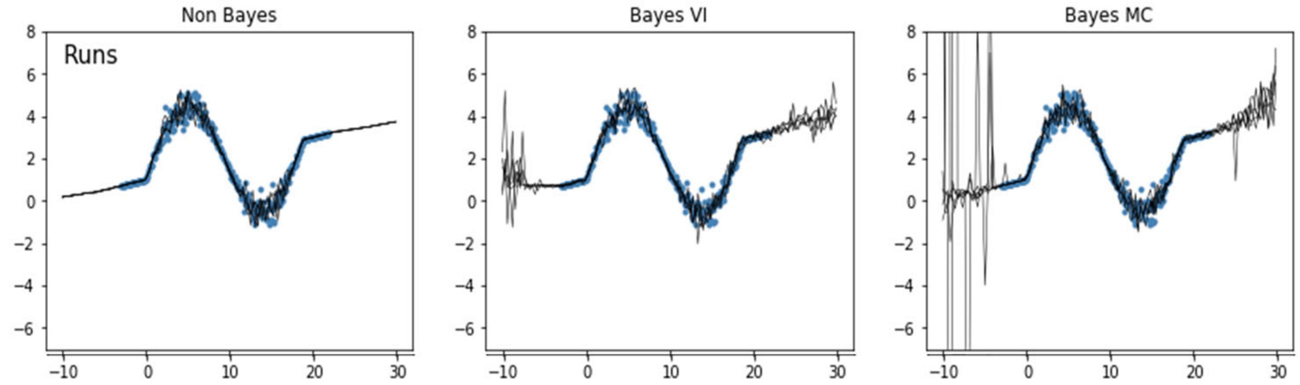
Uncertainty measures including **aleatoric and epistemic** contributions:

To estimate the 95% CI at position x by from each of the T CPDs and determine the 0.025 and 0.975 quantiles, yielding :

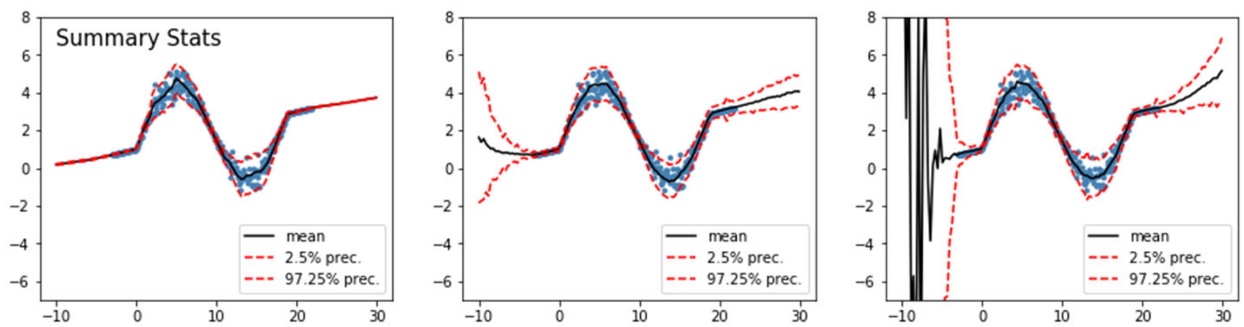
95% CI: $[q_{0.025}; q_{0.975}]$

How Bayesian NN indicate uncertainty

The solid lines show five predicted y-vectors corresponding to 5 CPDs at each x-position.



https://youtu.be/FO5avm3XT4g https://youtu.be/mQrUcUoT2k4 https://youtu.be/0-oyDeR9HrE



Conclusion

- Standard neural networks (NNs) fail to express their uncertainty (can’t talk about the elephant in the room).
- Bayesian neural networks (BNNs) can express their uncertainty.
- BNNs often yield better performance than their non-Bayesian variant.
- Novel classes can be better identified with BNNs, which combine epistemic and aleatoric uncertainties compared to standard NNs.
- Variational inference (VI) and Monte Carlo dropout (MC dropout) are approximation methods that allow you to fit deep BNNs.
- TFP provides easy to use layers for fitting a BNN via VI.
- MC dropout can be used in Keras for fitting BNNs.