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:

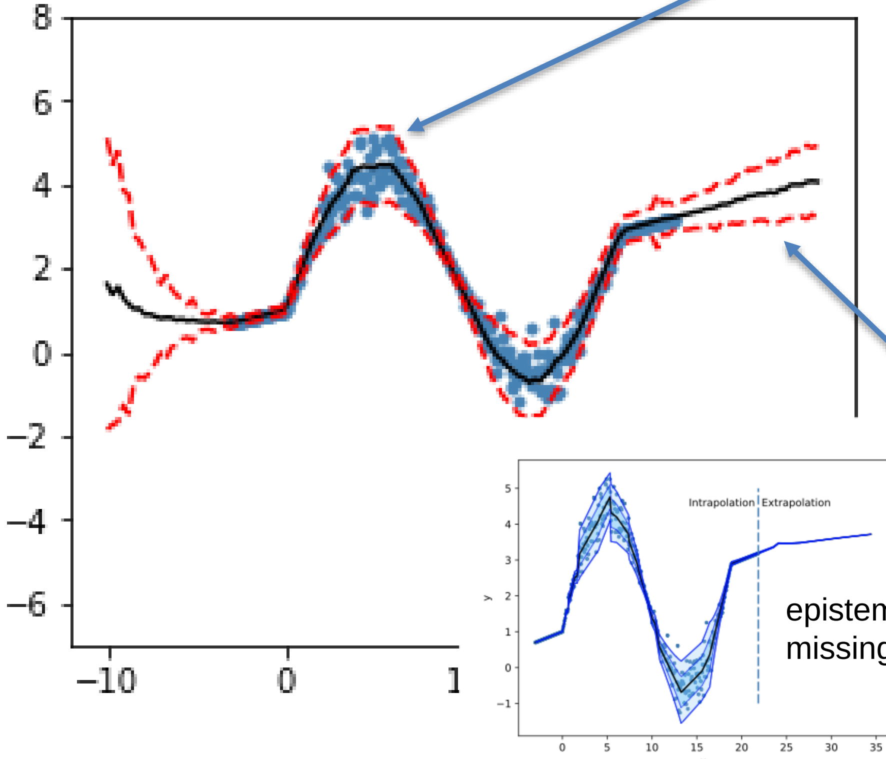
- Issues with current DL approach
  - No uncertainty for the fitted weights
    → epistemic uncertainty is ignored causing different problems:
    - No increased uncertainty in case of extrapolation
    - Deficits in prediction performance

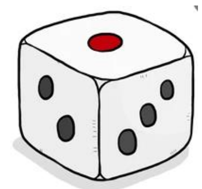
- Approaches to take epistemic uncertainty into account:
  - Deep Ensembling
  - MC Dropout

Aleatroic vs. Epistemic Uncertainty

Much spread in train data → aleatroic uncertainty (from latin “Alea Acta est”))

Caesar: Die Würfel sind gefallen!





We understand a dice perfectly (no epistemic uncertainty), but still there is a aleatoric uncertainty about the the number that will show up next when rolling the dice.

No (or few) train data

→ no (or few) “knowledge”

(from Ancient Greek ἐπιστήμη *(epistḗmē)* 'knowledge')

→ epistemic uncertainty

epistemic uncertainty is missing in classical NN

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

The elephant in the room

A high performant NN
(trained on imageNet data)
does not see the elephant!

General VIEW DOCS

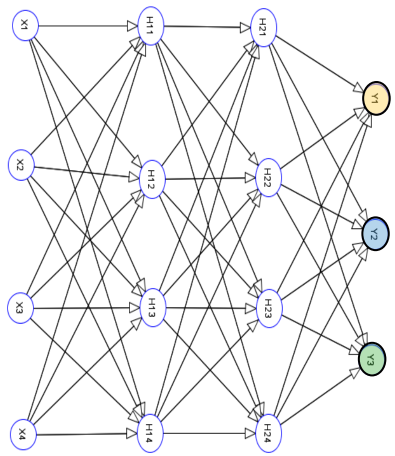
PREDICTED CONCEPT	PROBABILITY
group	0.979
adult	0.977
people	0.976
furniture	0.960
room	0.957
business	0.903
indoors	0.901
man	0.896
seat	0.895

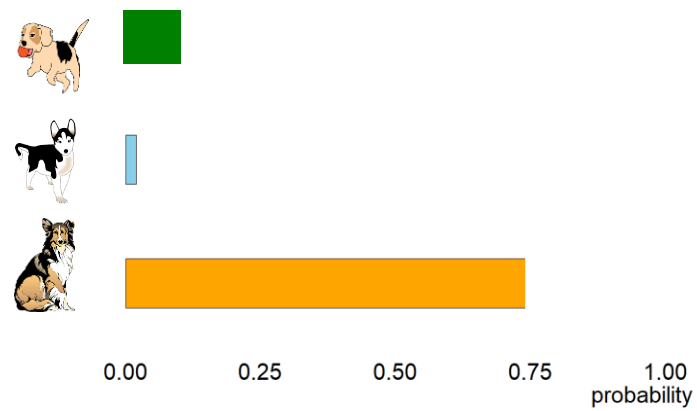
[NeurIPS 2018 Workshop on Causal Learning]

Elephant in the room

- Aufgabe: https://github.com/tensorchiefs/dl_course_2022/blob/master/notebooks/18_elephant_in_the_room.ipynb

CNNs have high performance on in-distribution examples

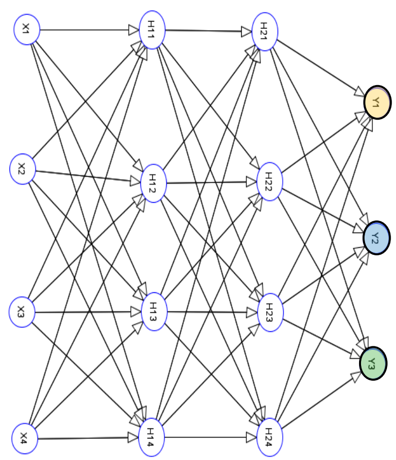




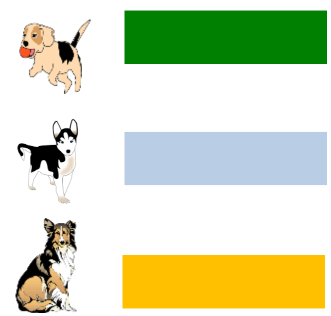
CNNs yield high accuracy and calibrated probabilities, but…

A classical NN cannot ring the alarm in case of out-of-distribution (OOD) examples

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



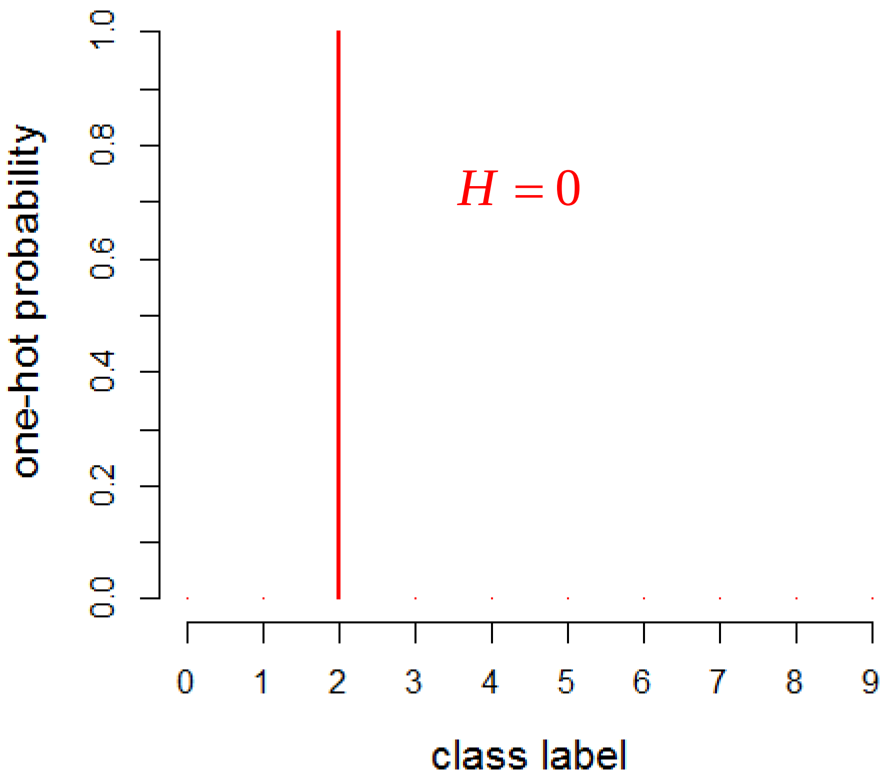
You might expect:

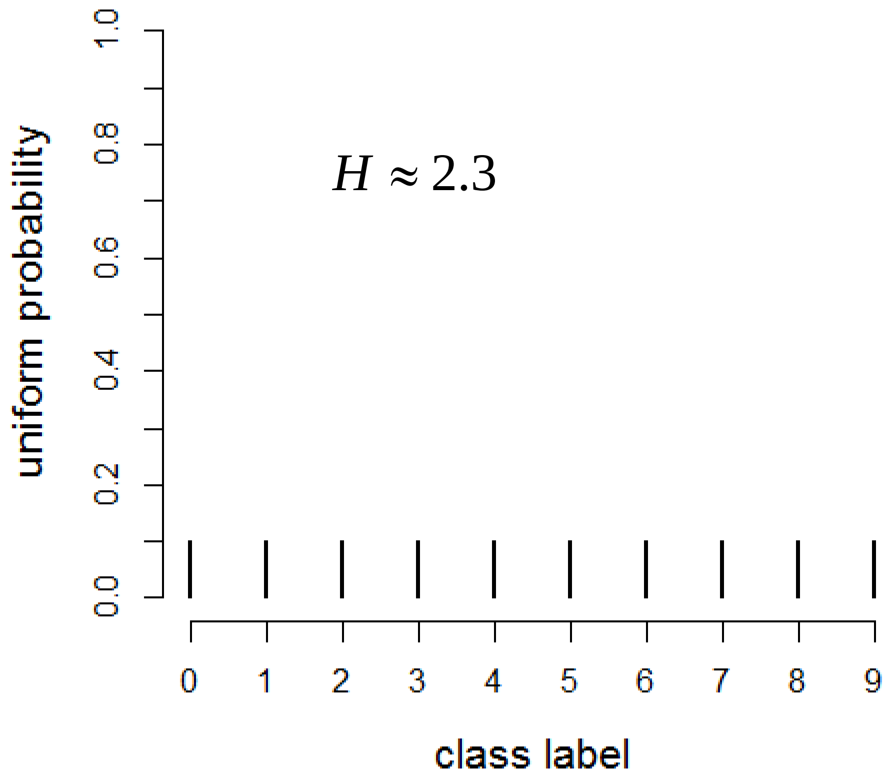


Recall: Entropy as measure for uncertainty

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

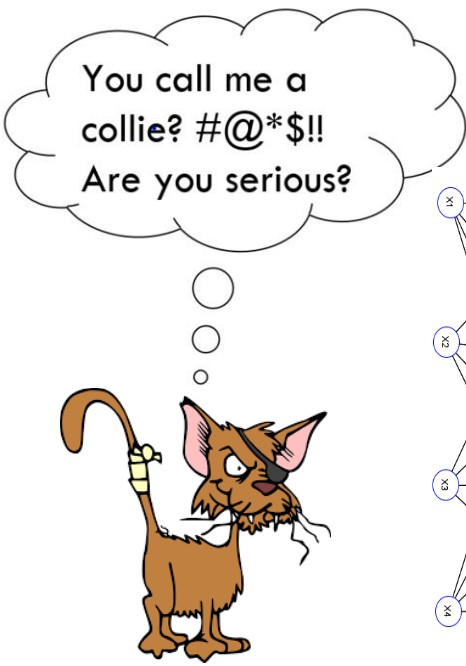
Entropy is a measure for “untidyness” or uncertainty.
If a distribution has only one peak, it is tidy and H=0
If all outcomes are equally probable it is maximal untidy

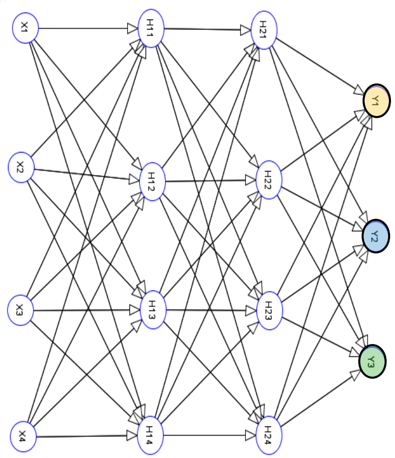




A classical NN cannot ring the alarm in case of out-of-distribution (OOD) examples

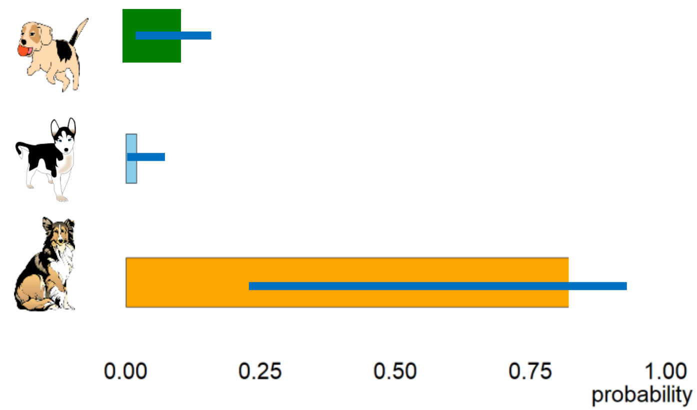
What happens if we present a novel class to the CNN?





But you probably get:

Plain wrong !



We need some error bars!
We need epistemic uncertainty!

Importance to detect OOD (out of distribution)

- 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

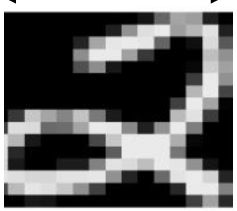
Deep ensembles

https://www.tutorialspoint.com/how-can-keras-be-used-to-implement-ensembling-in-python

Recall the training of NN models via SGD

k=28

k=28



Flatten to vector with k^2 elements

input
$\mathbf{x}^t$

$$\begin{bmatrix} 24.0 \\ -5.1 \\ 12.2 \\ 89.9 \\ \vdots \\ 3.2 \end{bmatrix}$$

$$\mathbf{x}_{(1,k^2)} \cdot \mathbf{W}_{(k^2,2)} + \mathbf{b}_{(1,2)} = \mathbf{z}_{(1,2)}$$

Score or logit
$\mathbf{z}^t$

$$\begin{bmatrix} 2.3 \\ 0.1 \end{bmatrix}$$

$$\hat{p}_k = \frac{e^{z_k}}{\sum_j e^{zj}}$$

Softmax or probability
$\mathbf{p}$=S($\mathbf{z}$)

$$\begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}$$

1-hot labels
$\mathbf{y}$

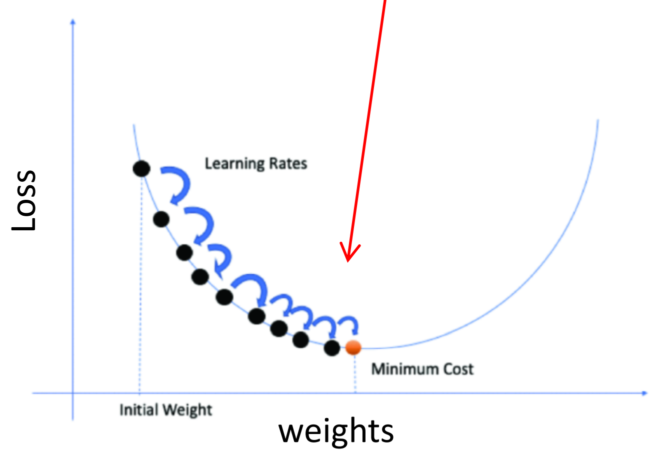
$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{matrix} \text{“2”} \\ \text{“1”} \end{matrix}$$

Loss: NLL, X-entropy $L(\mathbf{p}, \mathbf{y})$

$$= -\sum_{k=1}^{2} y_k \cdot \log\left(p_k\right)$$

Take step in direction of descent gradient:
(the gradient is oriented orthogonal to contour lines)

$$w_i^{(t)} = w_i^{(t-1)} - \varepsilon^{(t)} \frac{\partial L(\mathbf{w})}{\partial w_i}\bigg|_{w_i = w_i^{(t-1)}}$$



Loss = NLL averaged over all images in mini-batch

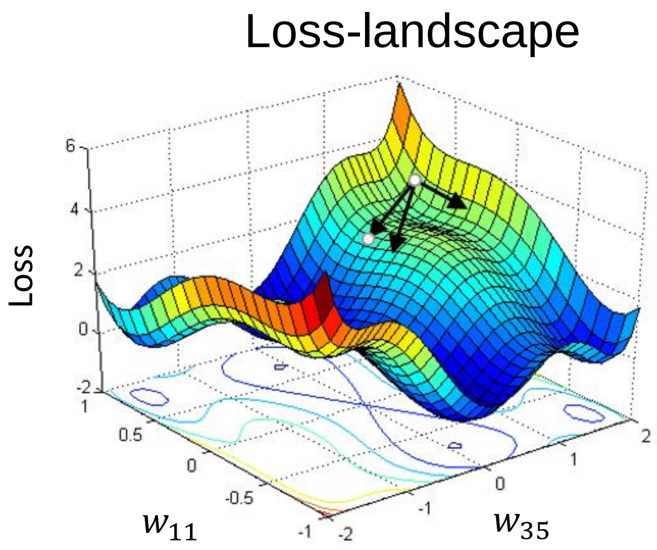
$$\mathrm{NLL} = \frac{1}{N} \sum_i L(\mathbf{p}_i, \mathbf{y}_i)$$

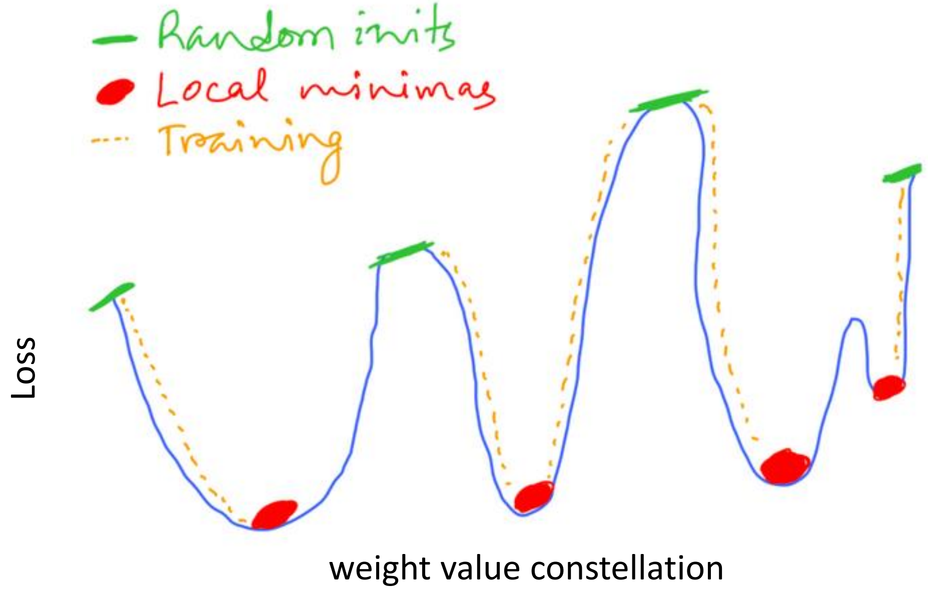
Loss

$\mathrm{C}\left(w_1, w_2\right)$

Image credicts

The loss-landscape in DL is usually not convex





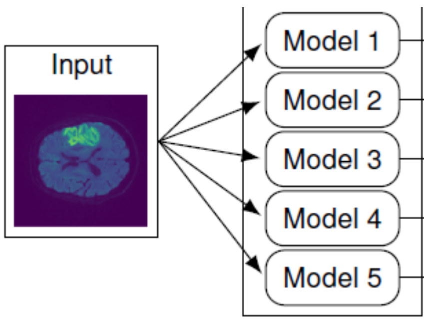
The loss-landscape of DL models has many local minima with similar depth.

Training is started with a random weight value initialization → training the NN with 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

Deep ensembling:
Train several NN models and average their predicitions

Trained models after random intitalization



Each model predictis probabilities for the 6 classes

p_1	p_2	p_3	p_4	p_5	p_6
0.1	0.1	0.2	0.4	0.1	0.1
0.05	0.05	0.2	0.3	0.2	0.2
0.2	0.2	0.1	0.05	0.05	0.4
0.05	0.05	0.05	0.05	0.2	0.6
0.3	0.25	0.25	0.1	0.1	0

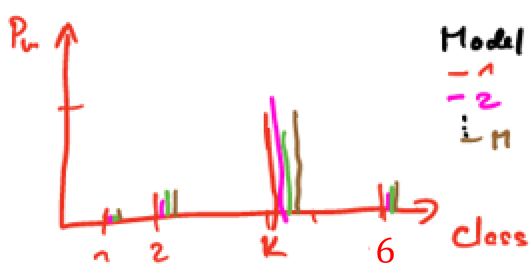
We get the ensemble predictions by averaging the probabilities that were predicted by the different models for each class

p_1^E	p_2^E	p_3^E	p_4^E	p_5^E	p_6^E
0.14	0.13	0.16	0.18	0.13	0.26

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

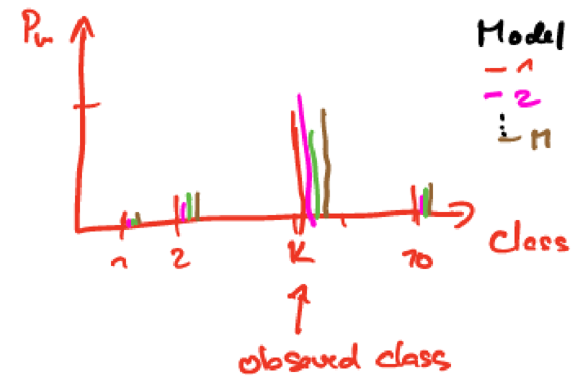
$sd_1^E \quad sd_2^E \quad sd_3^E \quad sd_4^E \quad sd_5^E \quad sd_6^E$



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.

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

Ensembling improves the NLL performance



Ensemble prediction for an observation with observed class = k, based on M models:

$$p_k^E = \frac{1}{M} \sum_{m=1}^{M} p_k \qquad k = 1 \ldots 10 \text{ for } 10 \text{ classes}$$

associated NLL contribution ℓ:

Model m: $\ell_m = -\log p_{km}$; Ensemble: $\ell^E = -\log p_k^E$

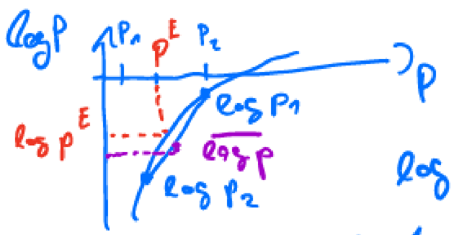
to show $\bar{\ell} \geq \ell^E$ with

$$\bar{\ell} = \frac{1}{M} \sum_{m=1}^{M} \ell_m = \frac{1}{M} \sum_m -\log p_{km}$$

$$= -\underbrace{\frac{1}{M} \sum_m \log p_{km}}_{\leq \log \frac{1}{M} \sum_m p_{km}}$$

$$\geq -\log p_k^E = \ell^E \quad \square$$

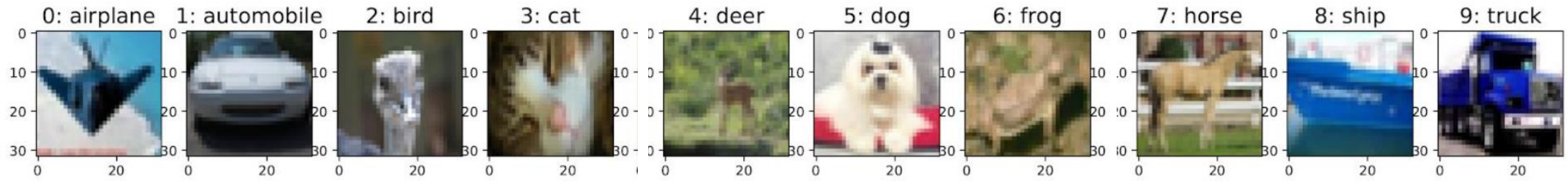
use Jensen inequality



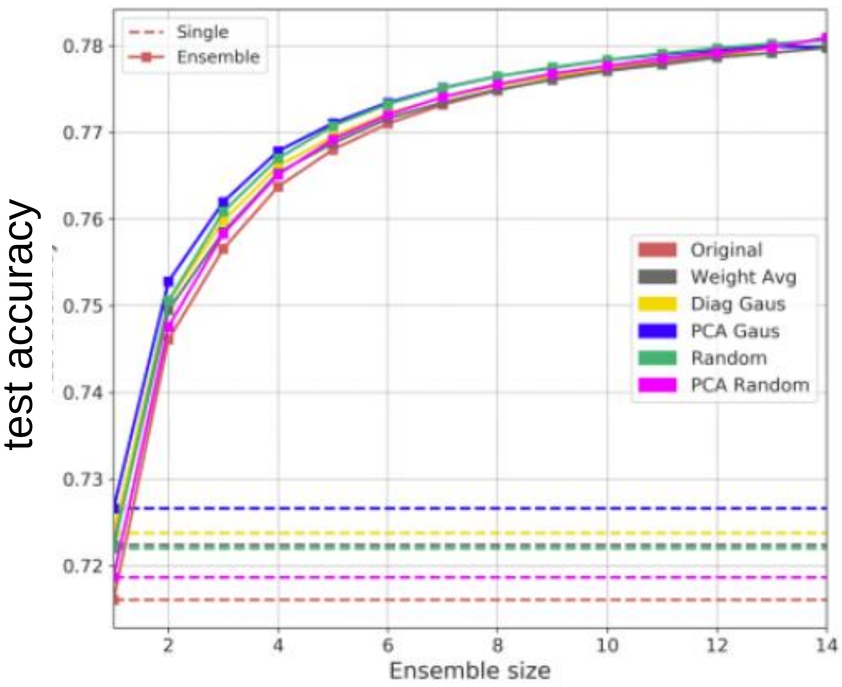
$$\log \bar{p} \geq \overline{\log p}$$

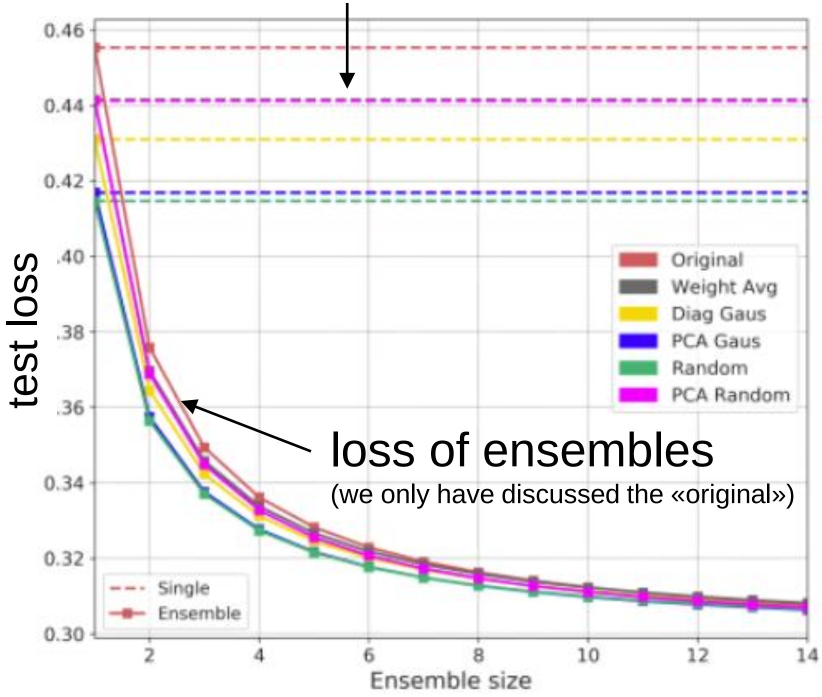
$$\log \frac{1}{M} \sum_m p_m \geq \frac{1}{M} \sum_m \log p_m$$

Deep ensembling improves prediction power



loss of 5 single models





loss of ensembles
(we only have discussed the «original»)

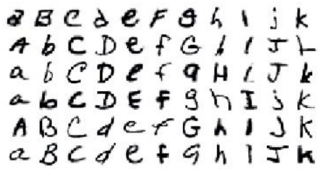
Ensembes with as few as 3 or 5 members are typically enough to achieve a perfromance gain.

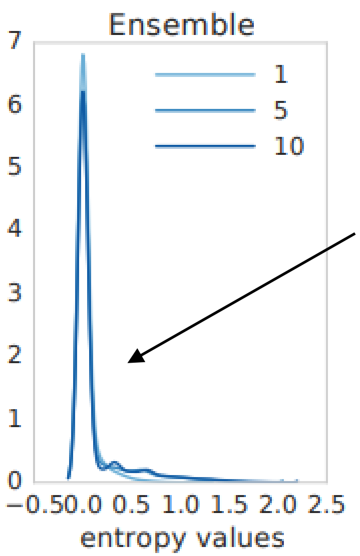
Image credit: https://wandb.ai/authors/loss-landscape/reports/Understanding-the-Effectivity-of-Ensembles-in-Deep-Learning--VmlldzoxODAxNjA

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.

Uncertainties for prediction of normal MNIST letter images

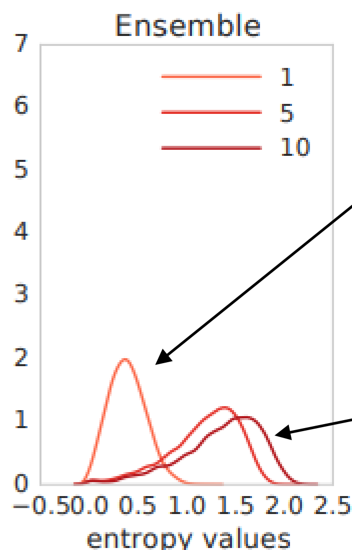




uncertainty measure

Ensemble uncertainty for a known class stays small for all sizes of the ensemble

Uncertainties for prediction of NOT-MNIST images



uncertainty measure

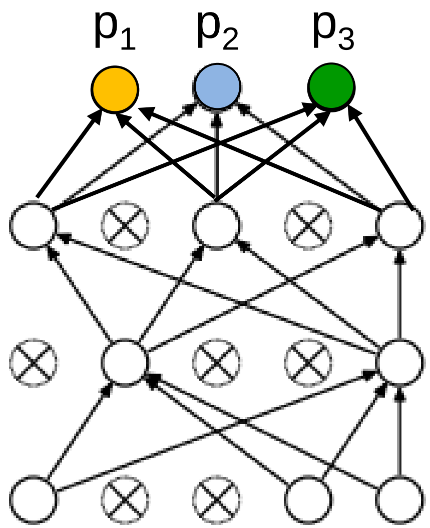
A single models shows less uncertainty for a novel class than an ensemble model

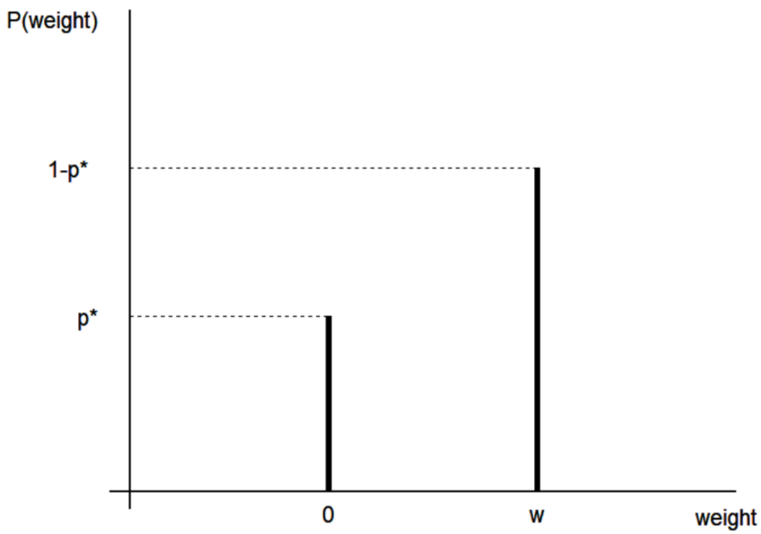
Ensemble uncertainty measures for novel classes improve with size of ensemblev

Image credits Lakshminarayanan et al.: https://arxiv.org/pdf/1612.01474.pdf

Dropout

Recall: Classical Dropout only during training

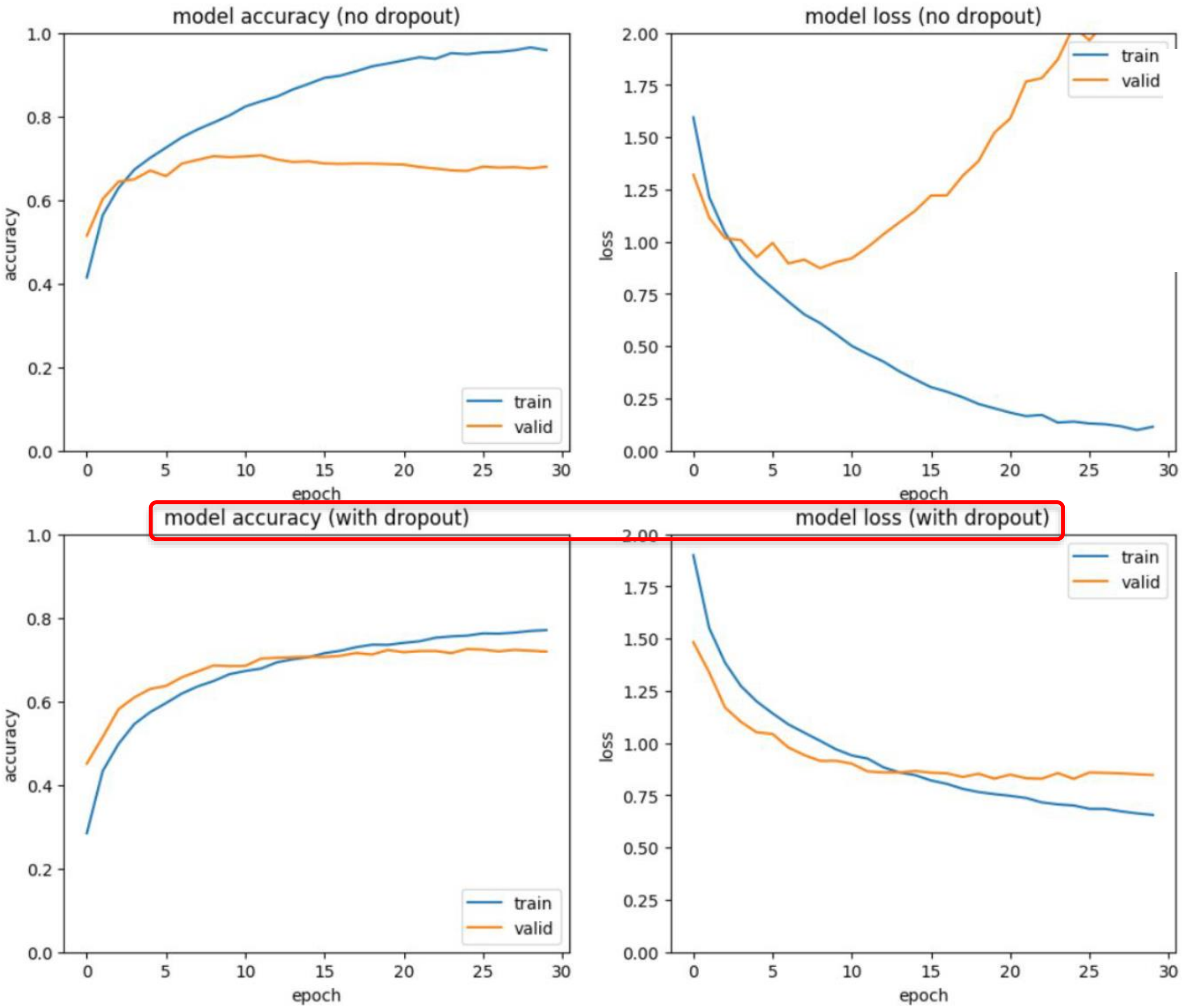


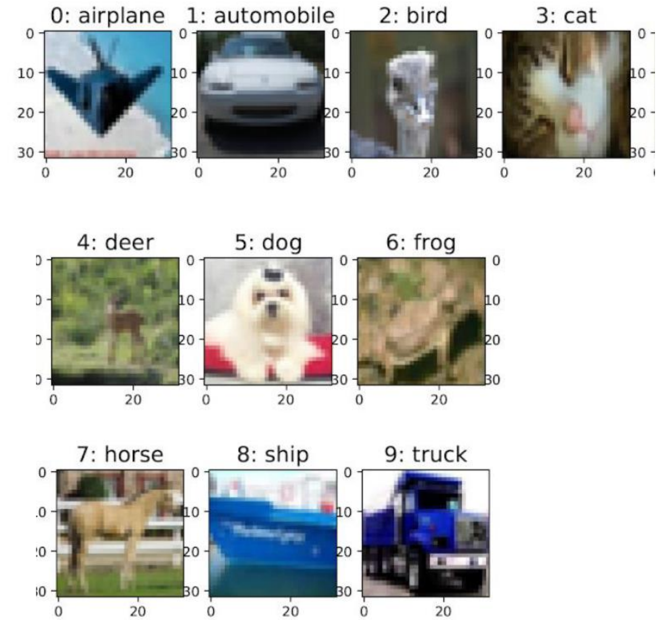


Using dropout during training implies:

- 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



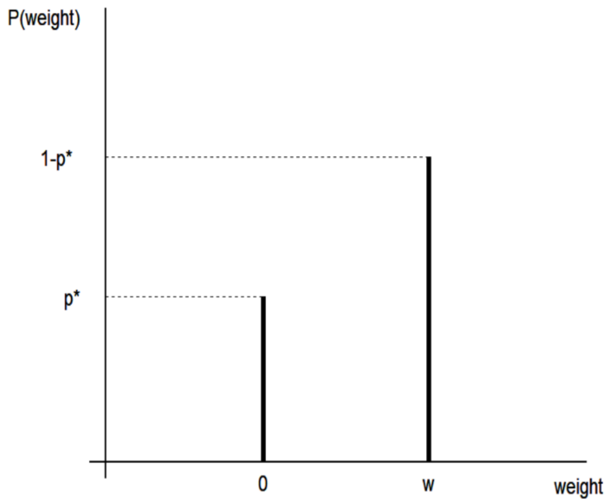


From Dropout during training to MC Dropout during test time

Bayesian NN via MC Dropout

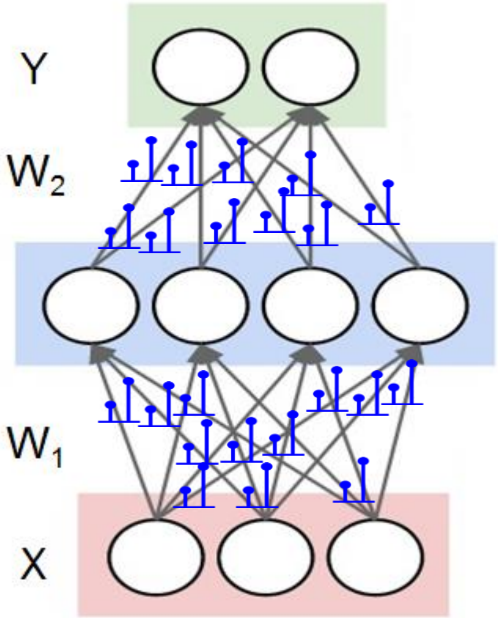
Yarin Gal et al. (2015):
Via Dropout training we learned a whole weight distribution for each connection.
We can sample from this Bernoulli-kind weight distribution by performing dropout during test time and use the dropout-trained NN as Bayesian NN.
Gal showed that doing dropout approximates VI with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

Learned
Bernoulli-kind distribution



Dropout in
test time

MC dropout NN



Weights have
Bernoulli-kind distribution

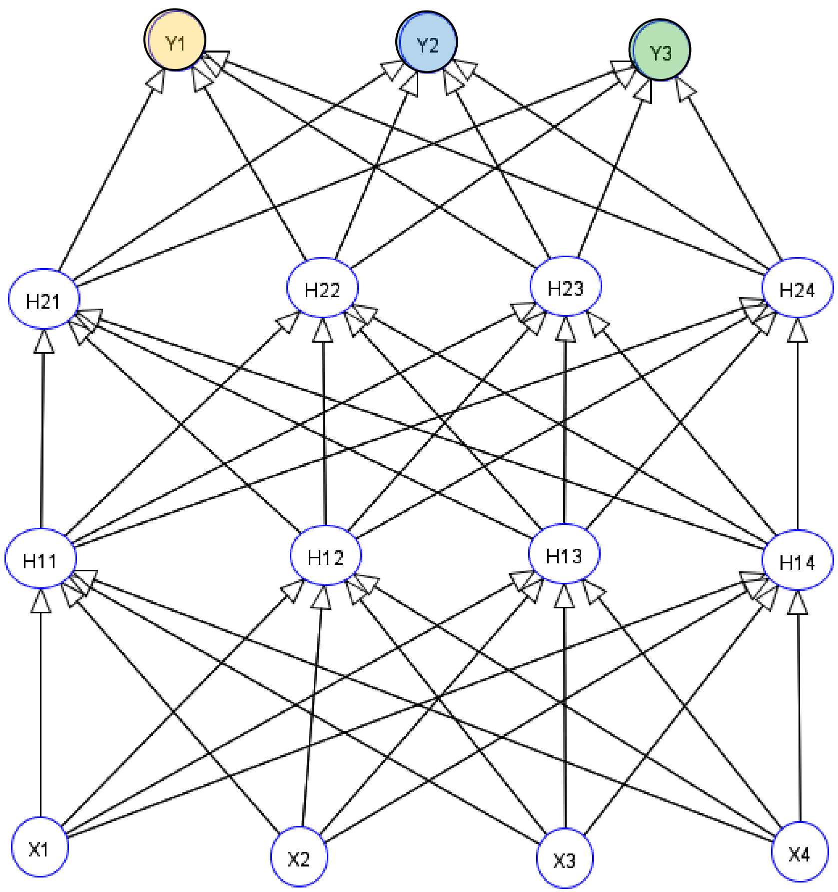
Which parameter has this q_θ?

The value w.

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:

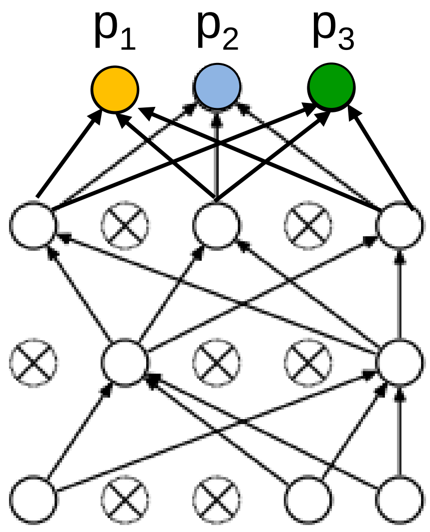
p_1=0.07 **p_2=0.91** p_3=0.02

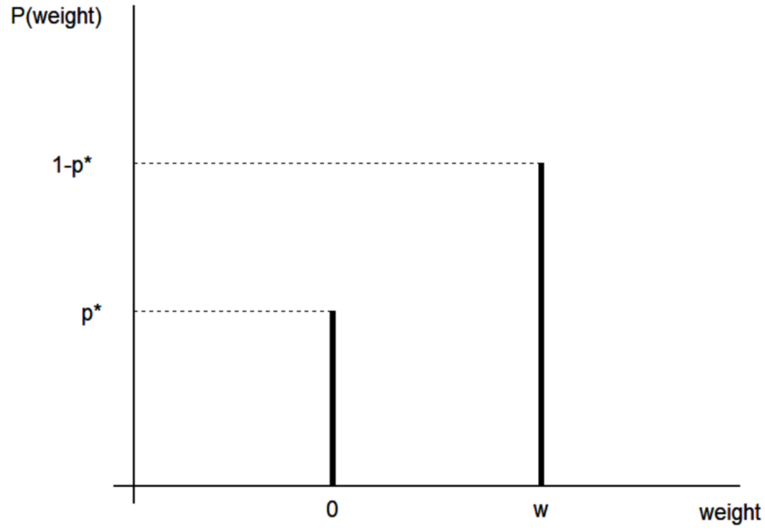


Probability of predicted class: $\mathbf{p_{max}}$

Input: image pixel values

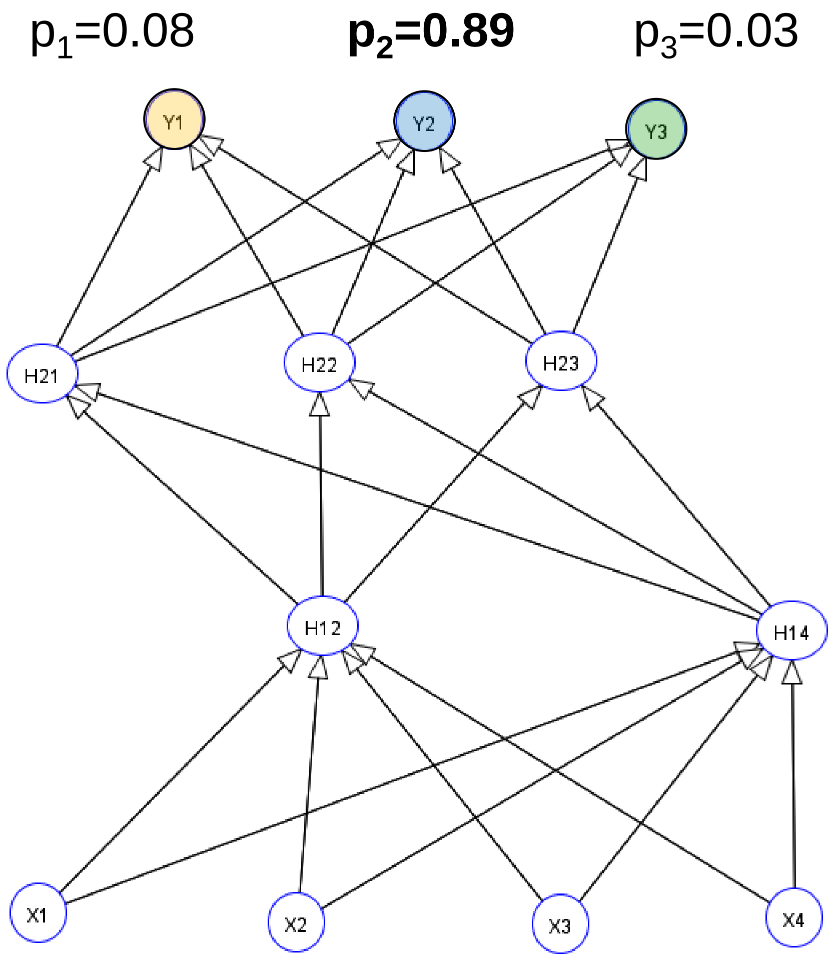
MC Dropout: we also perform dropout during test time

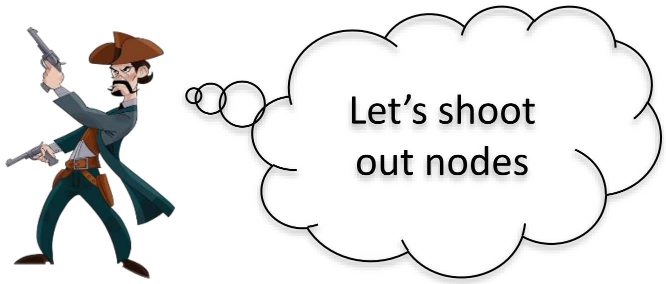




In each prediction instance we dropout a random subset of nodes, which corresponds to setting all weights starting from these nodes to zero.

MC Dropout during test time: Run 1



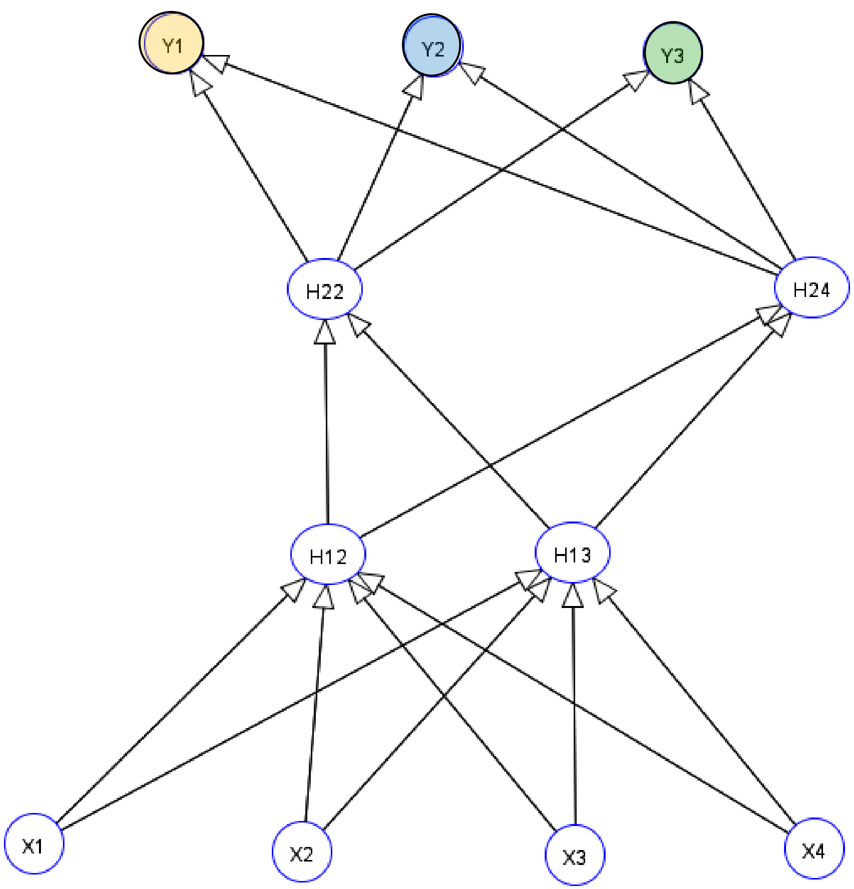


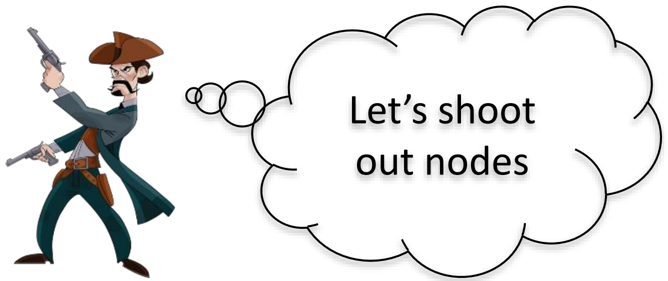
Stochastic dropout of units

Same input image

MC Dropout during test time: Run 2

p_1=0.11 **p_2=0.81** p_3=0.08

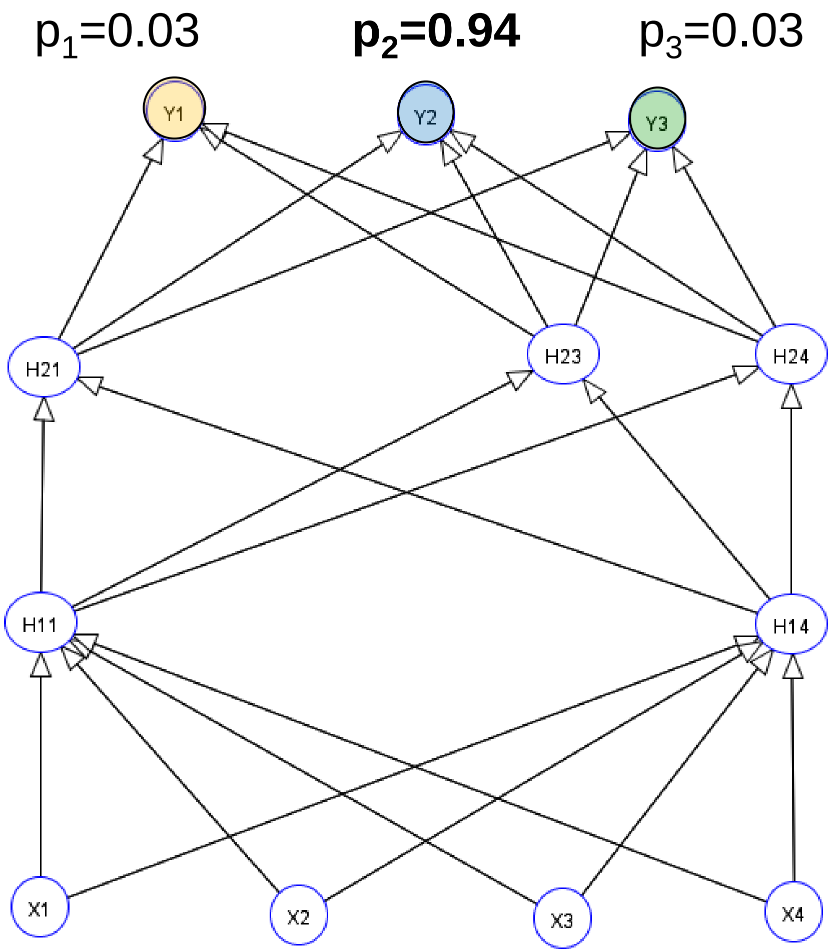


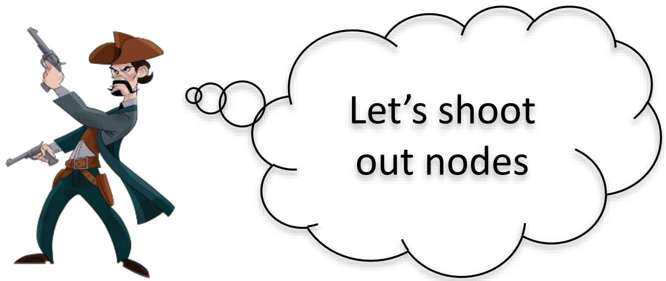


Stochastic dropout of units

Same input image

MC Dropout during test time: Run 3

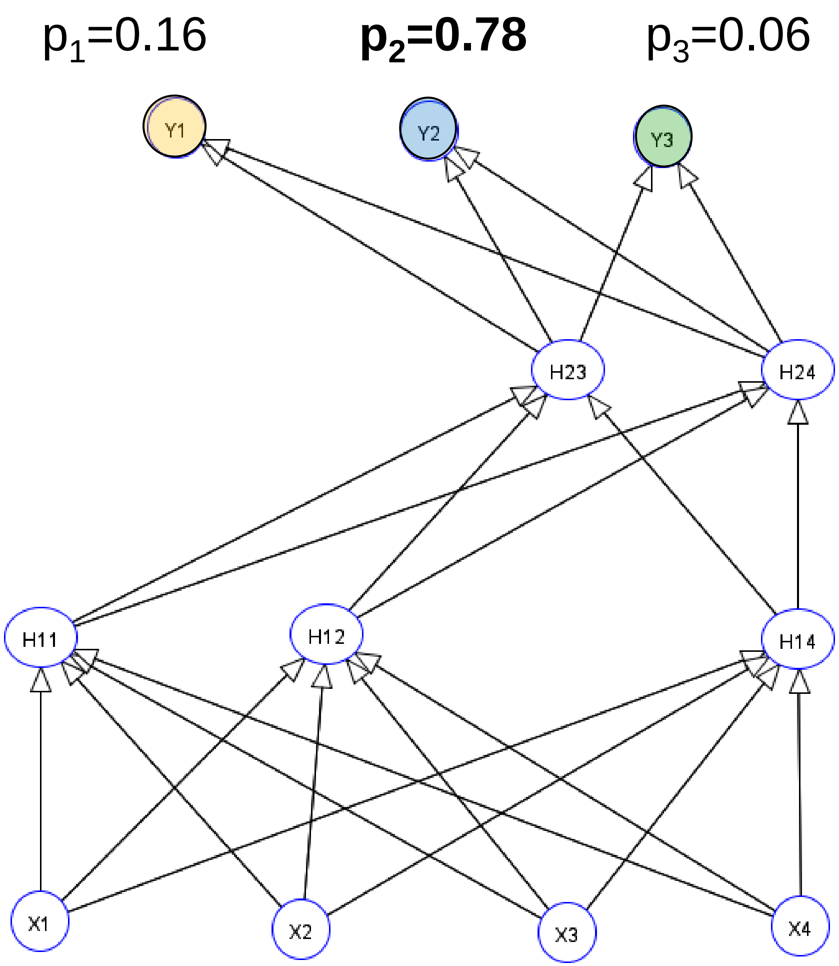


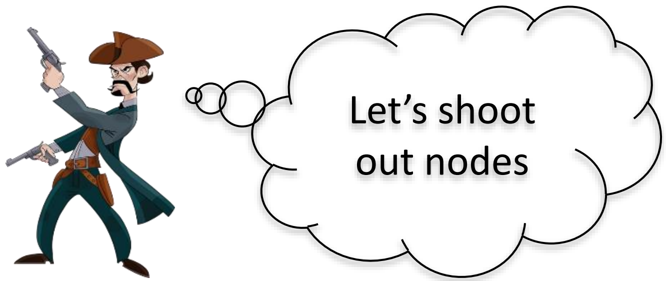


Stochastic dropout of units

Same input image

MC Dropout during test time: Run 4

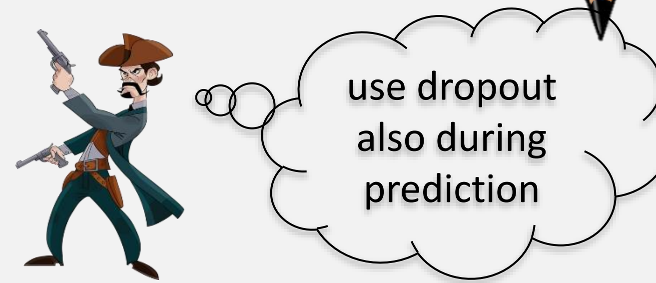




Stochastic dropout of units

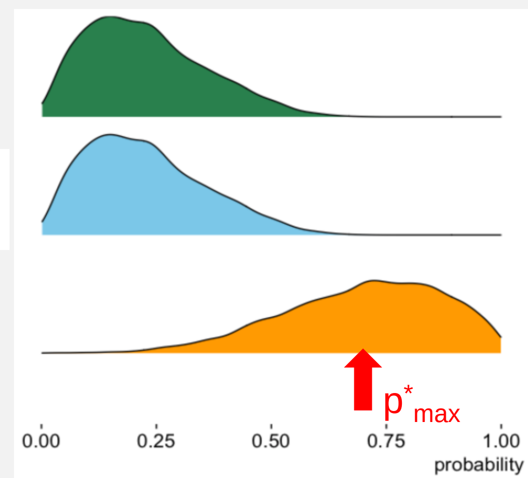
Same input image

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



Many Dropout Runs in forward pass

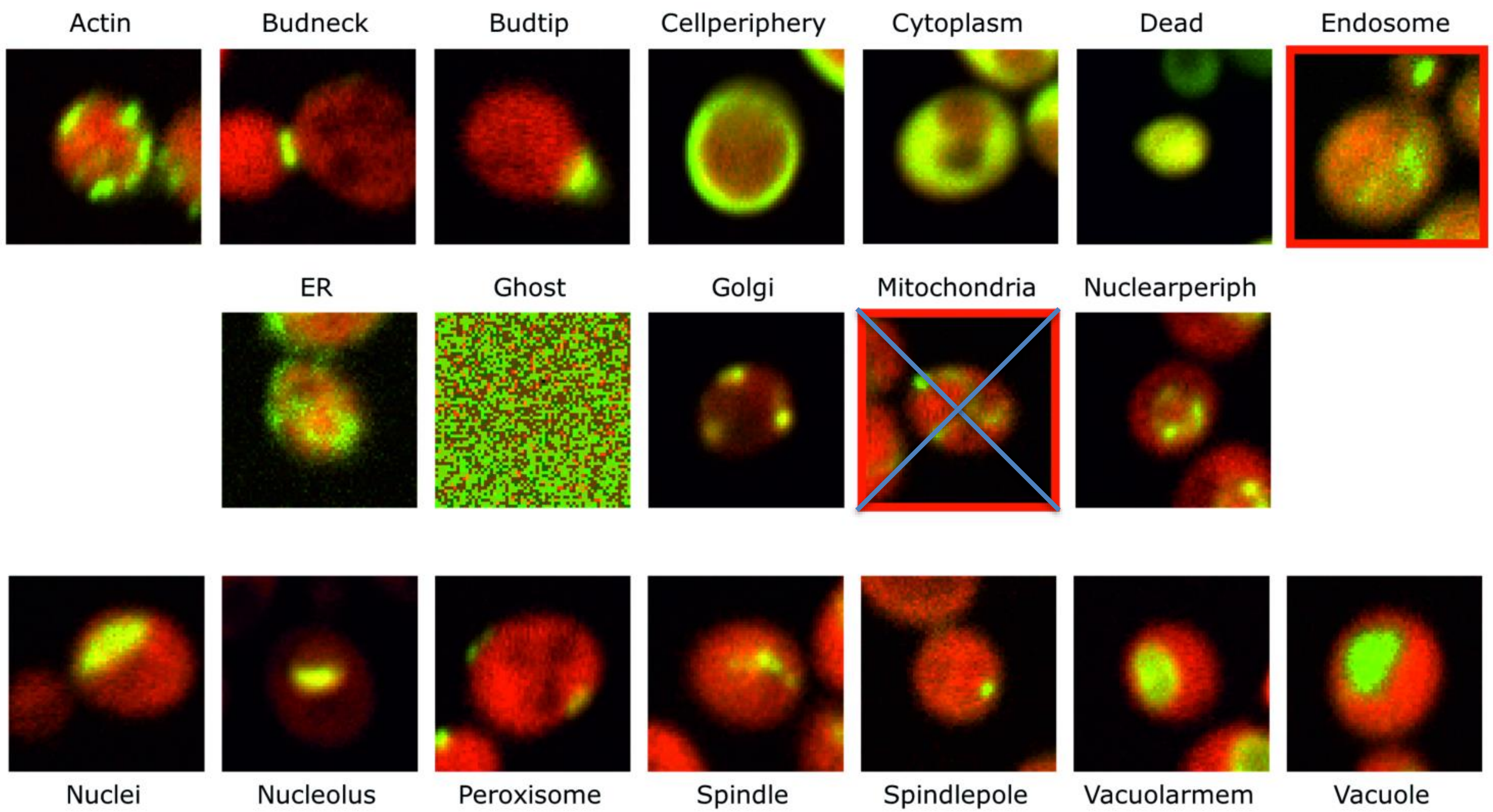
...



CNN predicts class “collie” but with high uncertainty

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

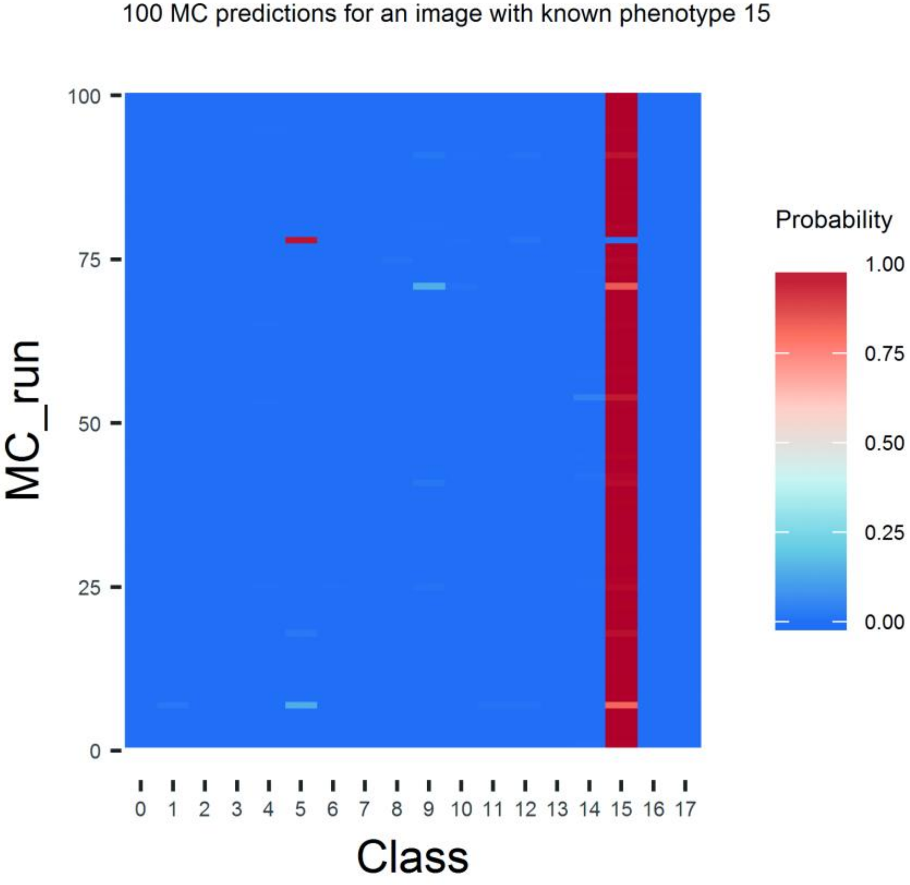
Experiment with unknown phenotype



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

Probability distribution from MC dropout runs

Image with known class 15



Probability distribution from MC dropout runs

Image with known class 15

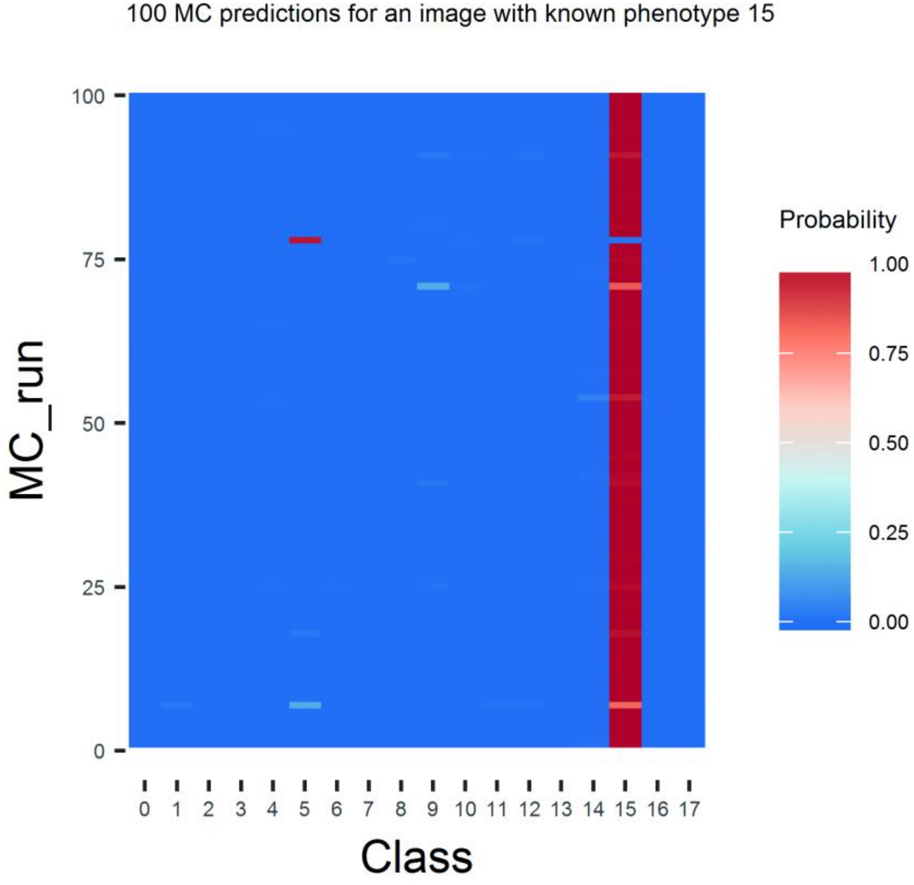
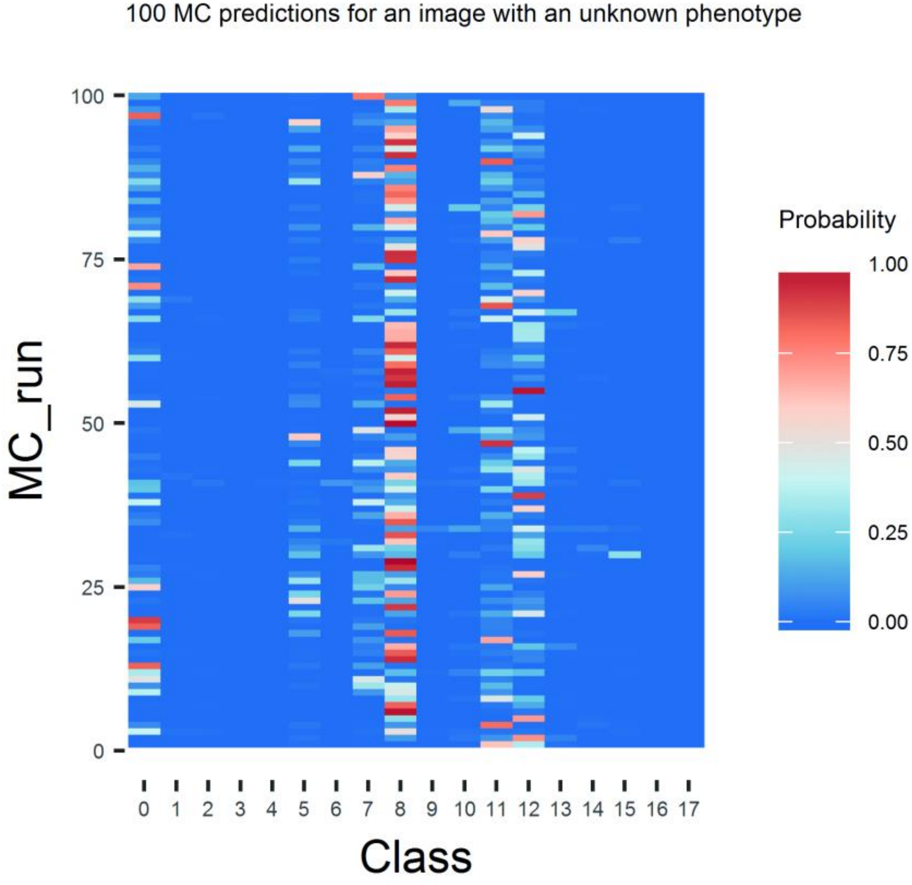
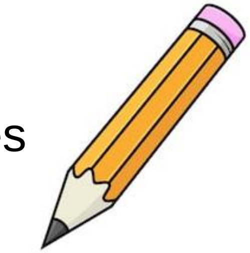


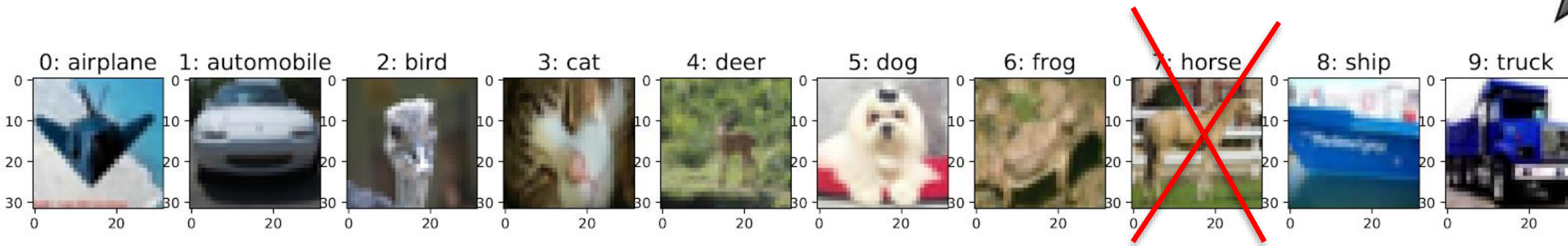
Image with unknown class



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.



Cifar10 classification case study with novel class

- Imports
- Loading and preparation of the dataset
- Non-Bayesian CNN
- **Variational Inference**
- MC Dropout
  - Getting mc dropout predictions
- Accuracy on the the known lables in the train set for all three models
  - Non-Bayesian prediction
  - Bayesian VI prediction
  - Bayesian MC prediction
  - Predicted classes for the unknown class
  - Compare the predictions for a known and unknown image

Please focus on MC Dropout

https://github.com/tensorchiefs/dl_course_2022/blob/master/notebooks/20_cifar10_classification_mc_and_vi.ipynb

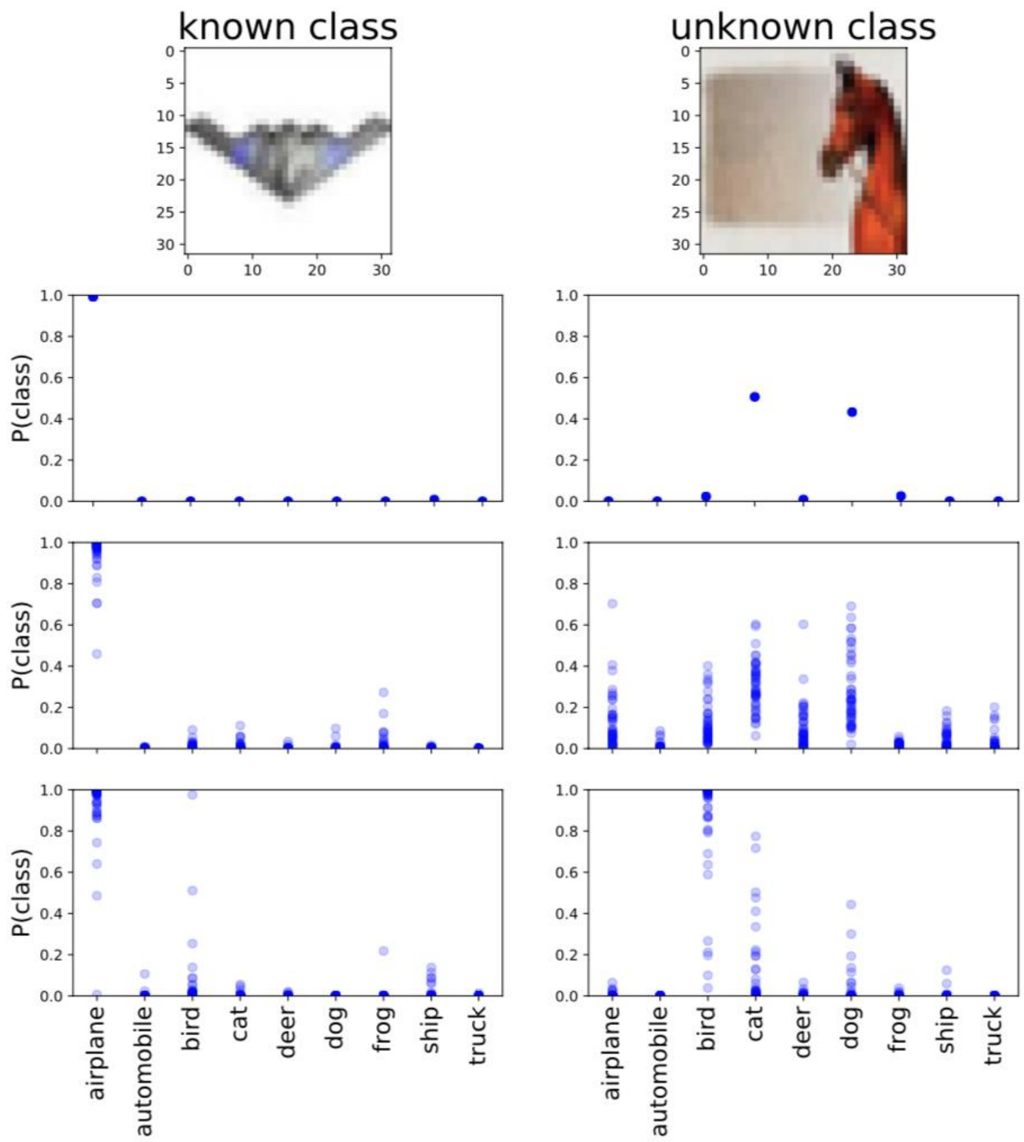
Run the Variational Inference without trying to understand it – we will care later!

Looking at the predictive distribution!

Classical CNN (no Bayes)

Gaussian VI

MC dropout



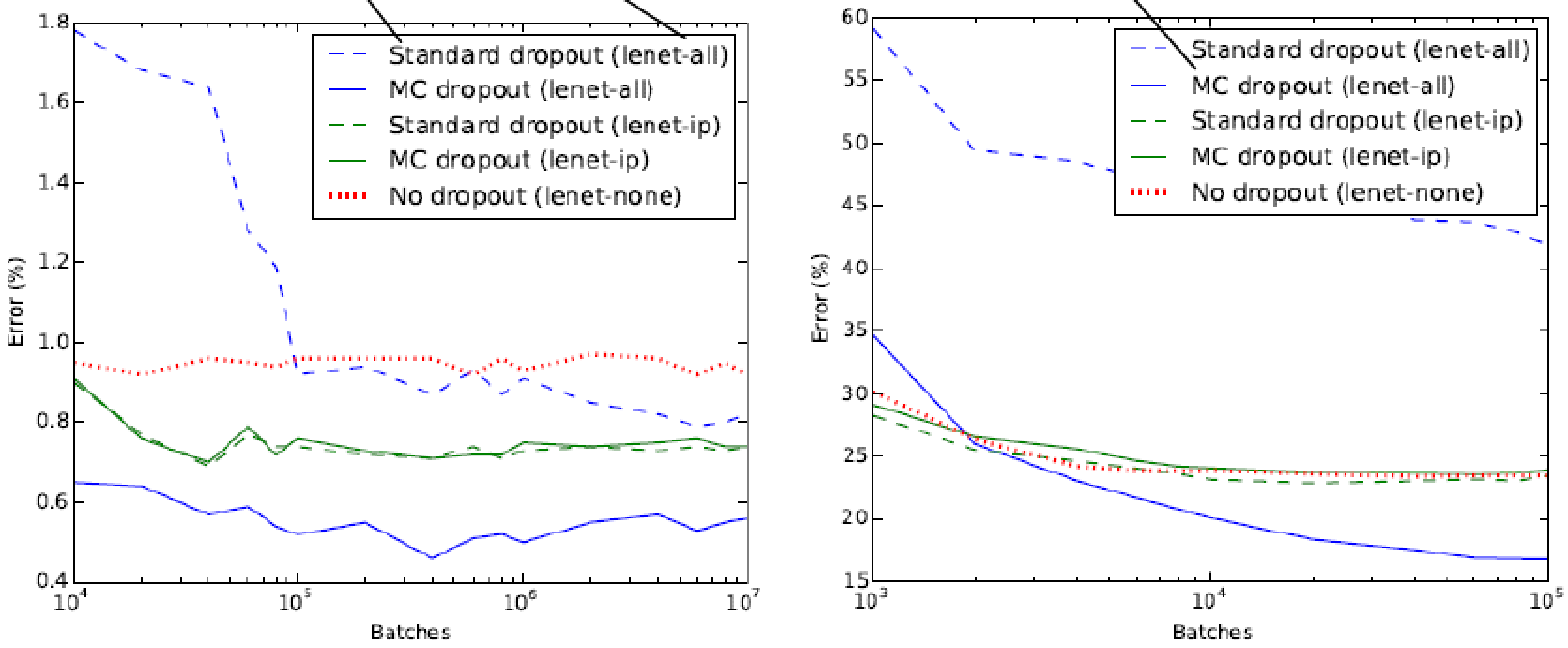
MC Dropout increases prediction performance

MC dropout is equivalent to performing *several* stochastic forward passes through the network and averaging the results. By doing so Yarin Gal was able to outperform state of the art error rates on MNIST and CIFAR-10 without changing the architecture of the used CNNs or anything else beside dropout.

"Standard" means no dropout in test time

"all" means dropout was done in all layers

"MC dropout" means that dropout was done in test time and the results were averaged



(a) MNIST (b) CIFAR-10

Image credits: PhD thesis of Yarin Gal: http://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf

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

MNIST classification

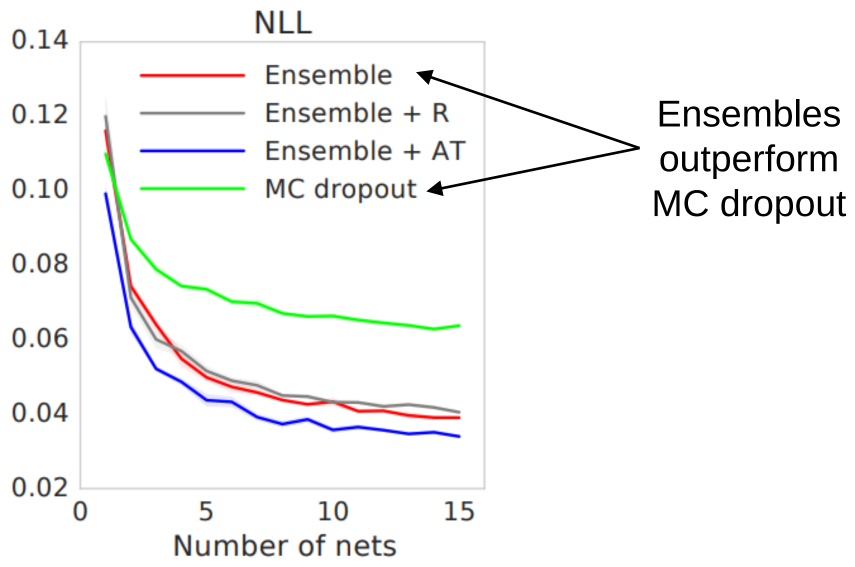
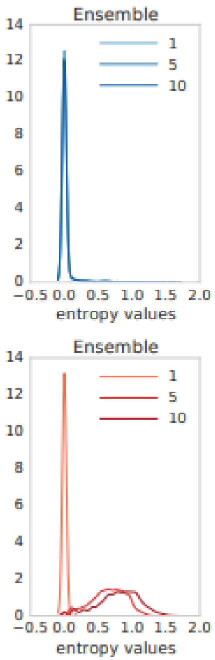
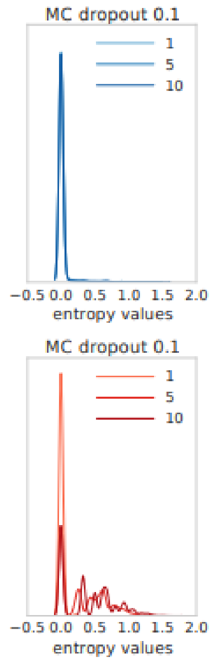


Image credits Lakshminarayanan et al.: https://arxiv.org/pdf/1612.01474.pdf

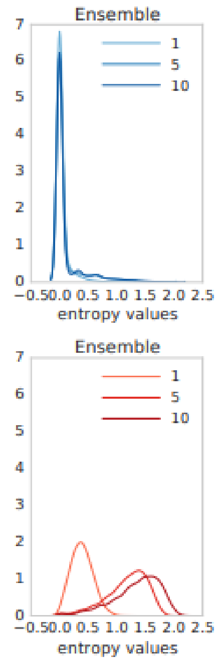
Compare MC dropout and deep ensembles uncertainties



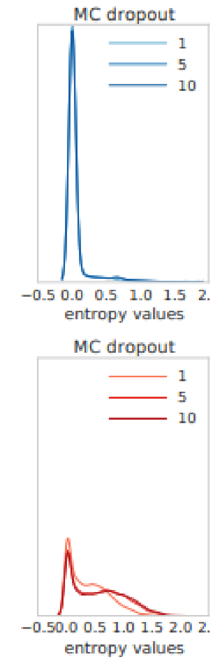
The novel class uncertainty is better in ensemble models than in MC Dropout models



(a) MNIST-NotMNIST



The novel class uncertainty is better in ensemble models than in MC Dropout models



(b) SVHN-CIFAR10

Figure 3: : Histogram of the predictive entropy on test examples from known classes (top row) and unknown classes (bottom row), as we vary ensemble size M.

Image credits Lakshminarayanan et al.: https://arxiv.org/pdf/1612.01474.pdf