

## Uncertainty in DL models

*Beate Sick*

[sick@zhaw.ch](mailto:sick@zhaw.ch)

Remark: Much of the material has been developed together with Elvis Murina and Oliver Dürr

# Topics

- **Capturing uncertainty in statistics**
  - Functional uncertainty: which probability distribution to pick for the outcome?
  - Epistemic uncertainty: Parameter estimate of the CPD have uncertainty
  - Aleatoric uncertainty: Spread of CPD quantifies data variation
  - Frequentist's vs Bayesian's approach to capture epistemic uncertainty
- **Capturing uncertainty in DL?**
  - Quantifying prediction uncertain via Dropout
  - Calibration

# Which kind of uncertainties do talk about?

- **Functional uncertainty**

- How sure can we be to have fitted the correct model?  
→ this uncertainty depends on our field knowledge but gets usually also smaller if we collect more data which allows to detect a potential bias more easily

- **Aleatoric uncertainty**

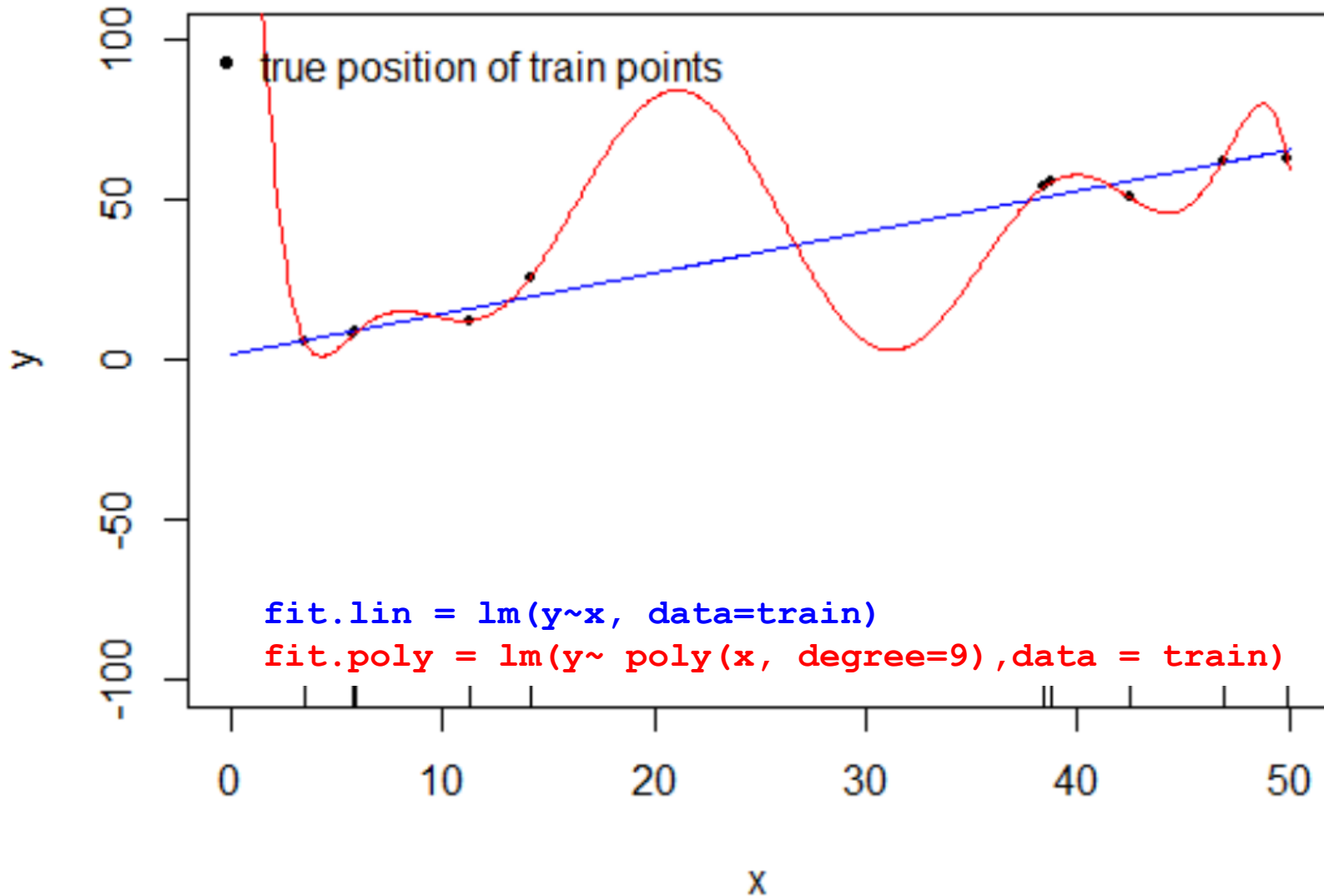
- data variability inherent to the data generating process  
→ this uncertainty does not get small when we collect more data

- **Epistemic uncertainty**

- Uncertainty of the model parameter estimates  
→ this uncertainty goes down when we collect more data

# Which is the right model?

Linear model or polynomial with 9 degrees of freedom?



# Recap: Linear regression in statistics

Model for the conditional probability distribution

CPD:  $Y_{X_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma^2)$

$$Y_x \in \mathbb{R}$$

$$\mu_x \in \mathbb{R}$$

$\mu_x$  is modeled

$\sigma^2$  is independent of the predictor values

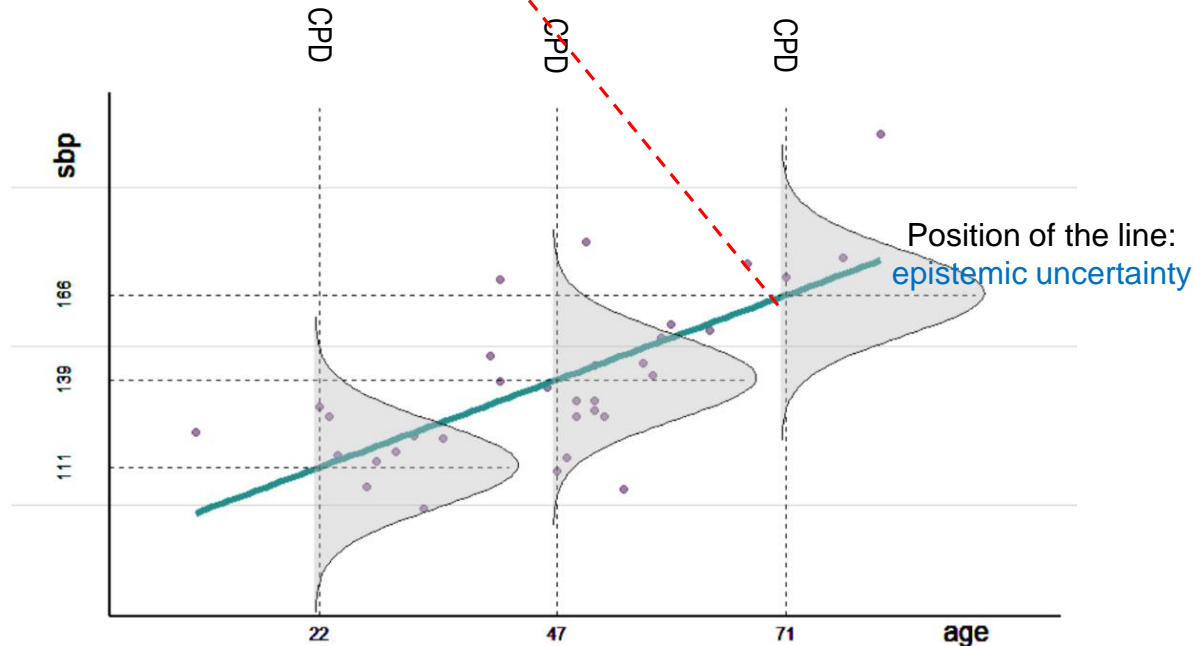
$$y_i = \beta_0 + \beta_1 \cdot x_{i1} + \varepsilon_i$$

$$E(Y_{X_i}) = \mu_{x_i} = (\mu|X=x_i) = \beta_0 + \beta_1 \cdot x_{i1}$$

$$\text{Var}(Y_{X_i}) = \text{Var}(Y|X_i) = \text{Var}(\varepsilon_i) = \sigma^2$$

$$\varepsilon_i \text{ i.i.d. } \sim N(0, \sigma^2)$$

Variability of data:  
aleatoric uncertainty

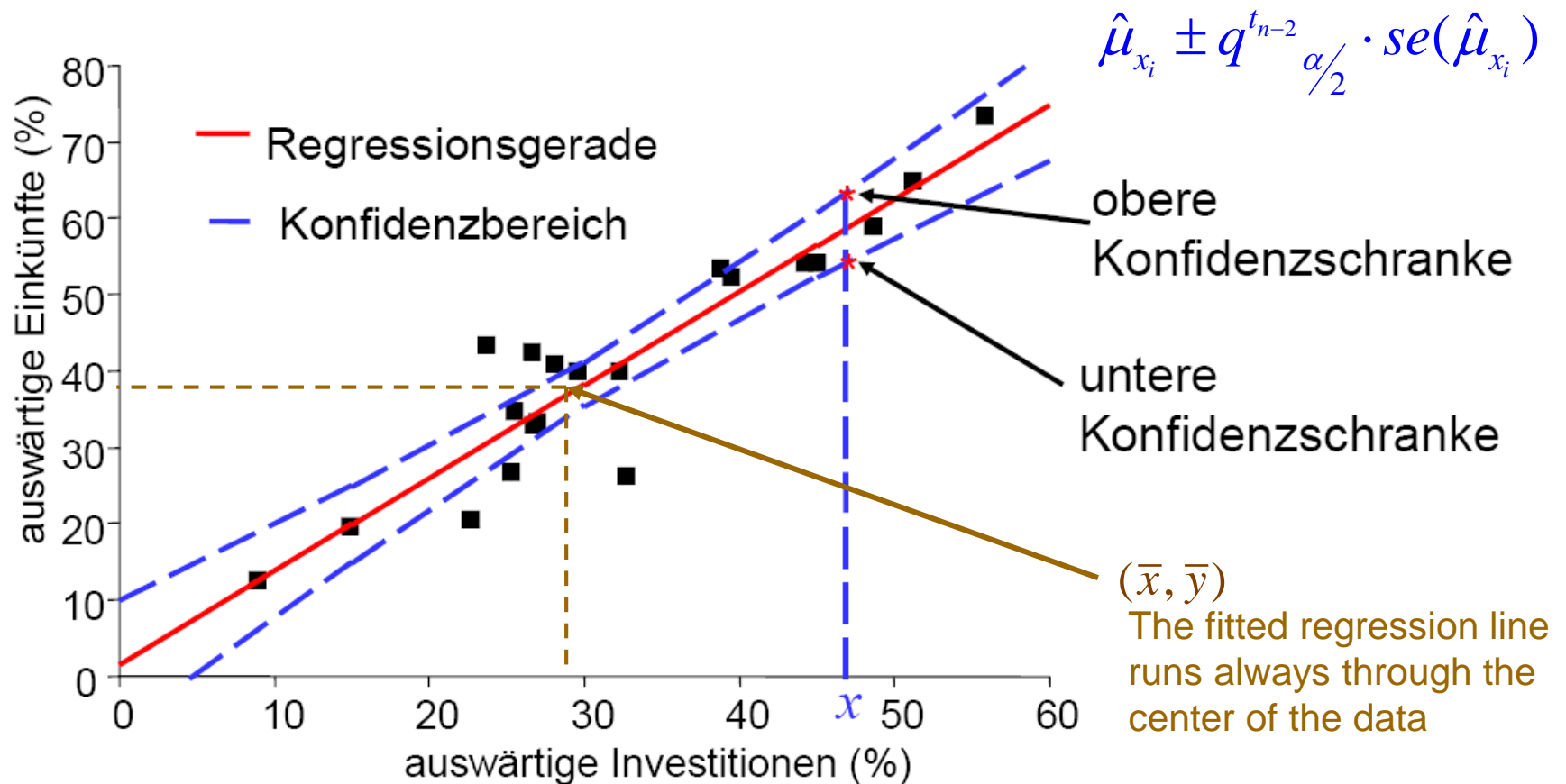


# Confidence Interval quantifies epistemic uncertainty

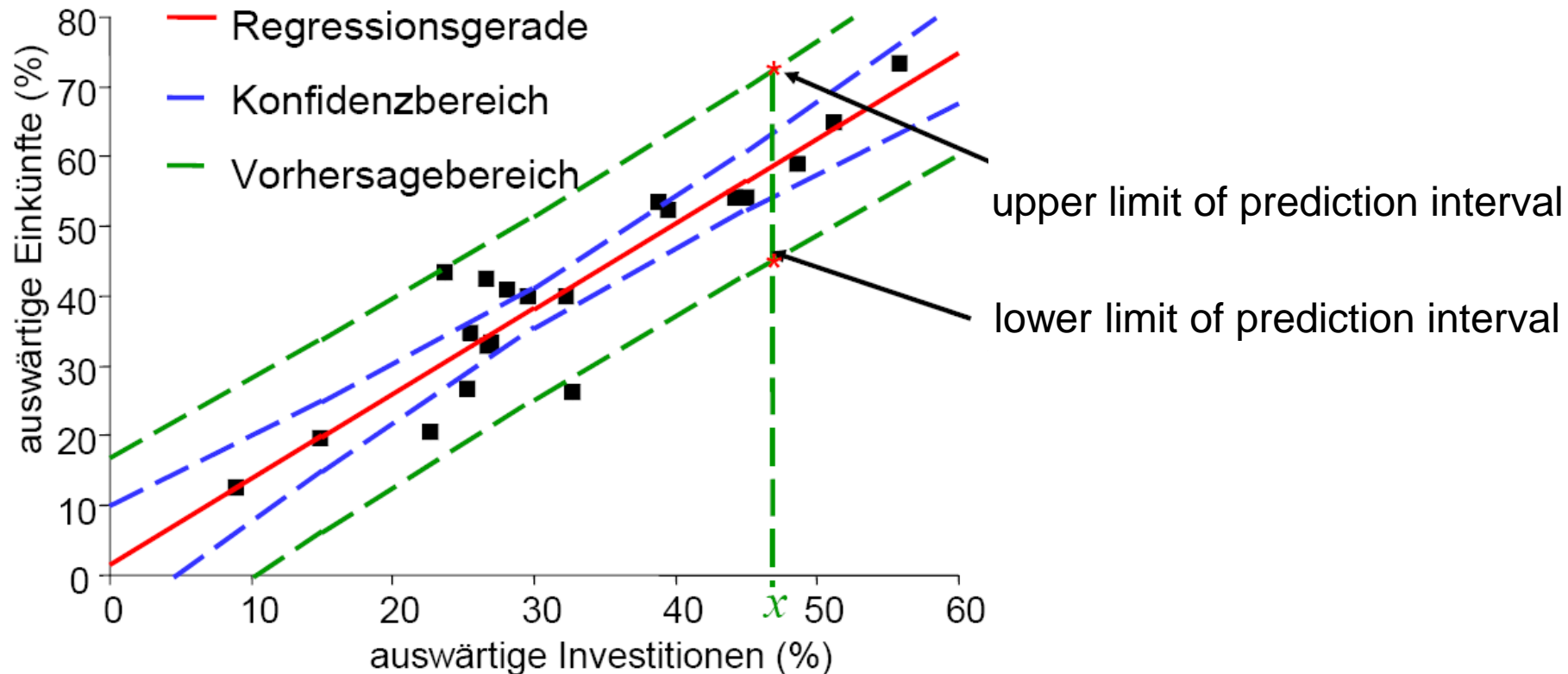
Model for the conditional probability distribution

$$\text{CPD: } Y_{x_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma^2)$$

We can determine at each position  $x_i$  the confidence interval for  $\mu_{x_i}$



# Prediction Interval quantifies also aleatoric uncertainties



The confidence interval quantifies the uncertainty of the parameter  $\mu$  ("epistemic uncertainty")

The prediction interval quantifies the variation of the data ("aleatoric uncertainty plus epistemic uncertainty")

# Confidence and Prediction Interval: Formula

$$Y_i = a + bX_i + \varepsilon_i$$

CI for  $\mu$        $\hat{\mu}_x \pm q^{t_{n-2}}_{\alpha/2} \cdot se(\hat{\mu}_x)$

epistemic uncertainty

$$se(\hat{\mu}_x) = \sqrt{\hat{\sigma}^2 \cdot \left( \frac{1}{n} + \frac{(x - \bar{x})^2}{\sum (x_i - \bar{x})^2} \right)}$$

CI for  $\sigma$        $\hat{\sigma} = \sqrt{\frac{1}{n-2} \sum_{i=1}^n r_i^2}$

epistemic uncertainty       $\hat{\sigma} \pm q^{t_{n-2}}_{\alpha/2} \cdot se(\hat{\sigma})$

$$se(\hat{\sigma}) = \sqrt{\hat{\sigma}^2 \cdot \left( 1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{\sum (x_i - \bar{x})^2} \right)}$$

PI for y       $\hat{\mu}_x \pm q^{t_{n-2}}_{\alpha/2} \cdot se(y_x)$

aleatoric uncertainty

$$se(y_x)^2 = se(\hat{\mu}_x)^2 + \hat{\sigma}^2$$



# Recap: Binary classification in statistics

CPD:  $Y_{X_i} = (Y|X_i) \sim \text{Ber}(p_{x_i})$

$Y_x \in \{0,1\}$  ,  $p_x \in [0,1]$

$p_x$  is modeled

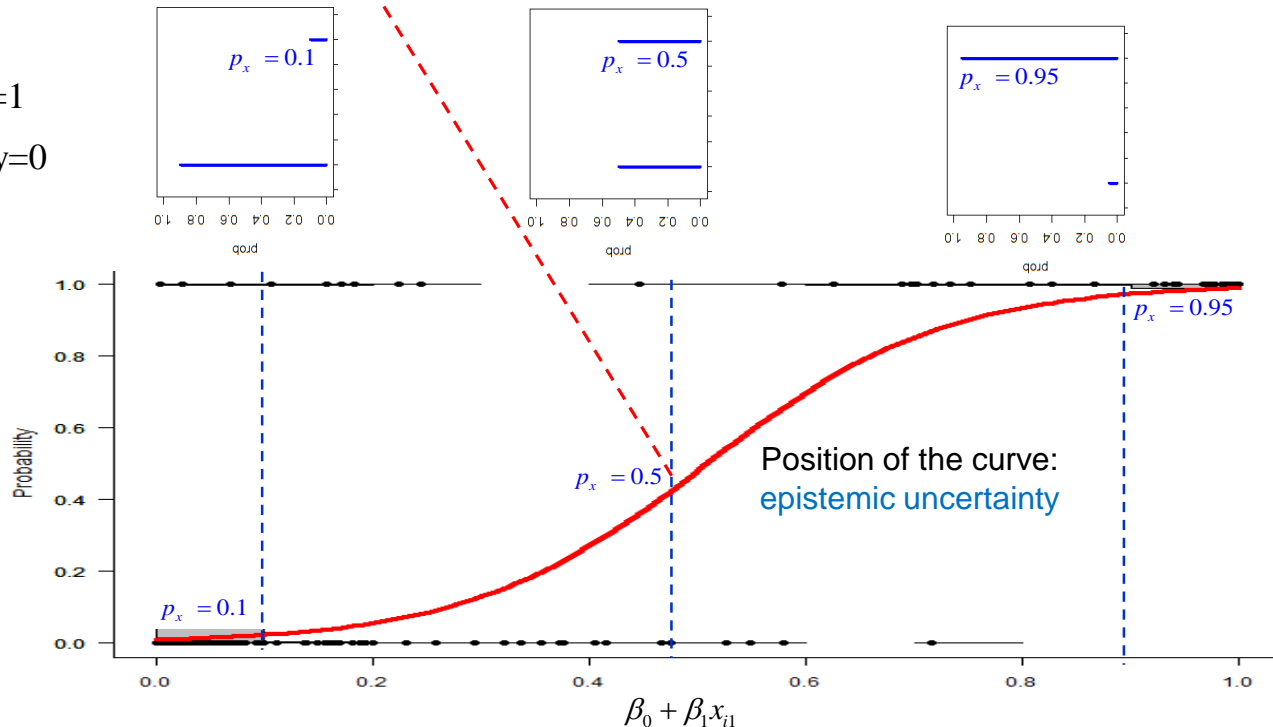
$$\log\left(\frac{p_{x_i}}{1-p_{x_i}}\right) = \beta_0 + \beta_1 x_{i1}$$

$$E(Y_{x_i}) = E(Y=1 | X = x_i) = p_{x_i} = \frac{e^{\beta_0 + \beta_1 x_{i1}}}{1 + e^{\beta_0 + \beta_1 x_{i1}}}$$

$$\text{Var}(Y_{x_i}) = p_{x_i} \cdot (1 - p_{x_i})$$

Variability of data:  
aleatoric uncertainty

$$P(Y | X = x) = \begin{cases} p_x & , y=1 \\ 1 - p_x & , y=0 \end{cases}$$



# Bayes inference: linear regression

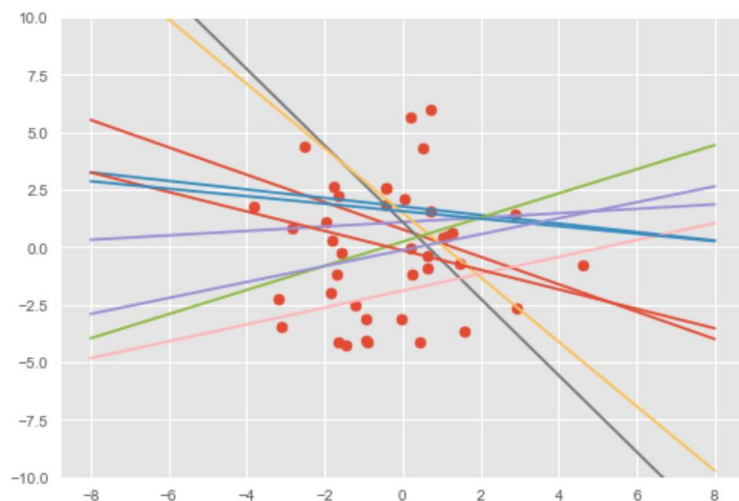
For a set of  $N$  data points  $(\mathbf{X}, \mathbf{y}) = \{(\mathbf{x}_n, y_n)\}$ , the model posits the following distributions:

$$\begin{aligned} p(\mathbf{w}) &= \text{Normal}(\mathbf{w} \mid \mathbf{0}, \sigma_w^2 \mathbf{I}), \\ p(b) &= \text{Normal}(b \mid 0, \sigma_b^2), \\ p(\mathbf{y} \mid \mathbf{w}, b, \mathbf{X}) &= \prod_{n=1}^N \text{Normal}(y_n \mid \mathbf{x}_n^\top \mathbf{w} + b, \sigma_y^2). \end{aligned}$$

before seeing data we set  
 $p(\omega)$  prior distribution  
 $\omega$  are the model parameter

In Bayes we assume that parameter come from a (prior) distribution.

We can sample parameter values and check possible realizations of models:

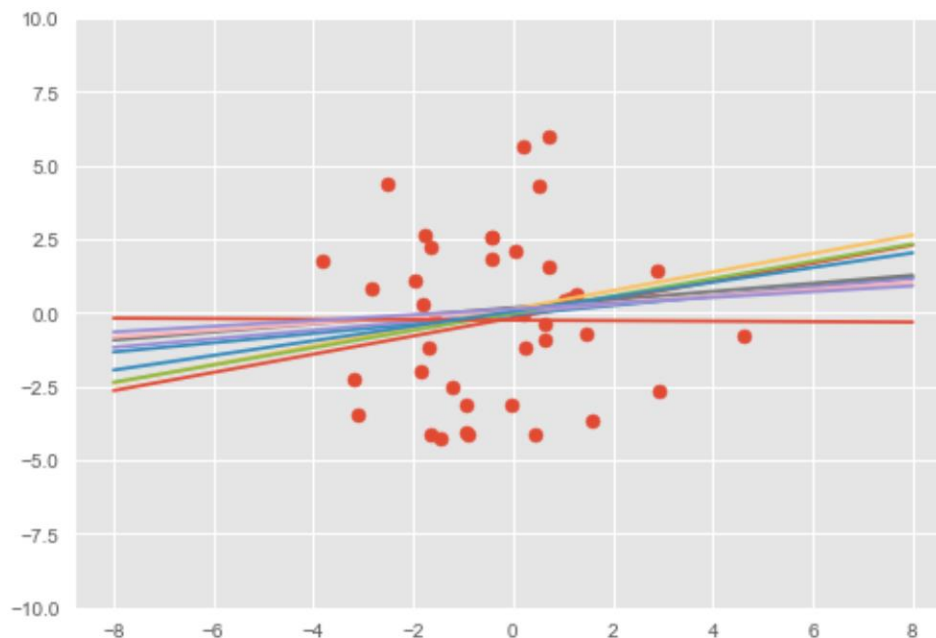


# Learning in the Bayesian framework

After we have seen some data  
we can update the distribution of  
the parameter  
→ posterior distribution

$$p(\omega | \mathbf{X}, \mathbf{Y}) = \frac{\overset{\text{likelihood}}{p(\mathbf{Y}|\omega, \mathbf{X})} \cdot \overset{\text{prior}}{p(\omega)}}{\underset{\substack{\text{normalizer}=\text{marginal likelihood}}}{p(\mathbf{Y}|\mathbf{X})}}$$
$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \omega) \cdot p(\omega) d\omega$$

We can capture the epistemic uncertainty by the interval in which 95% of all possible models are located.



# Prediction with flexible Frequentist's vs Bayesian's models

## Frequentist's strategy:

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

## Bayesian's strategy:

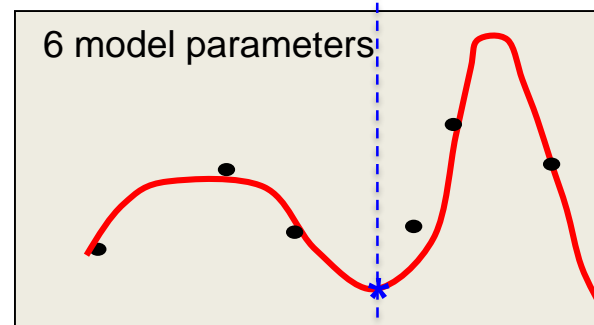
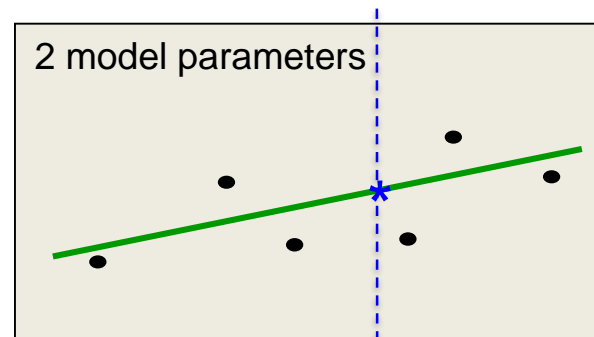
Use the model complexity you believe in.

Do not just use the best fitting model.

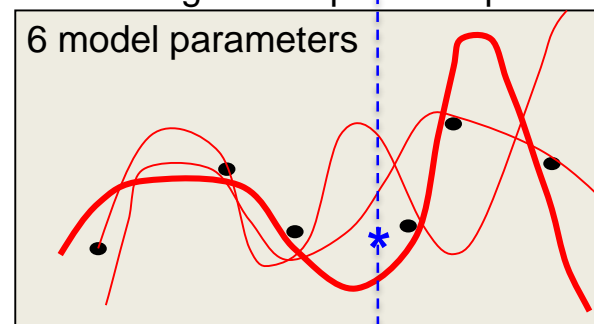
Do use the full posterior distribution over parameter settings leading to vague predictions since many different parameter settings have significant posterior probability.

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, \omega) \cdot p(\omega | X, Y) d\omega$$

prediction via marginalization over  $\omega$ :



Models with significant posterior probability



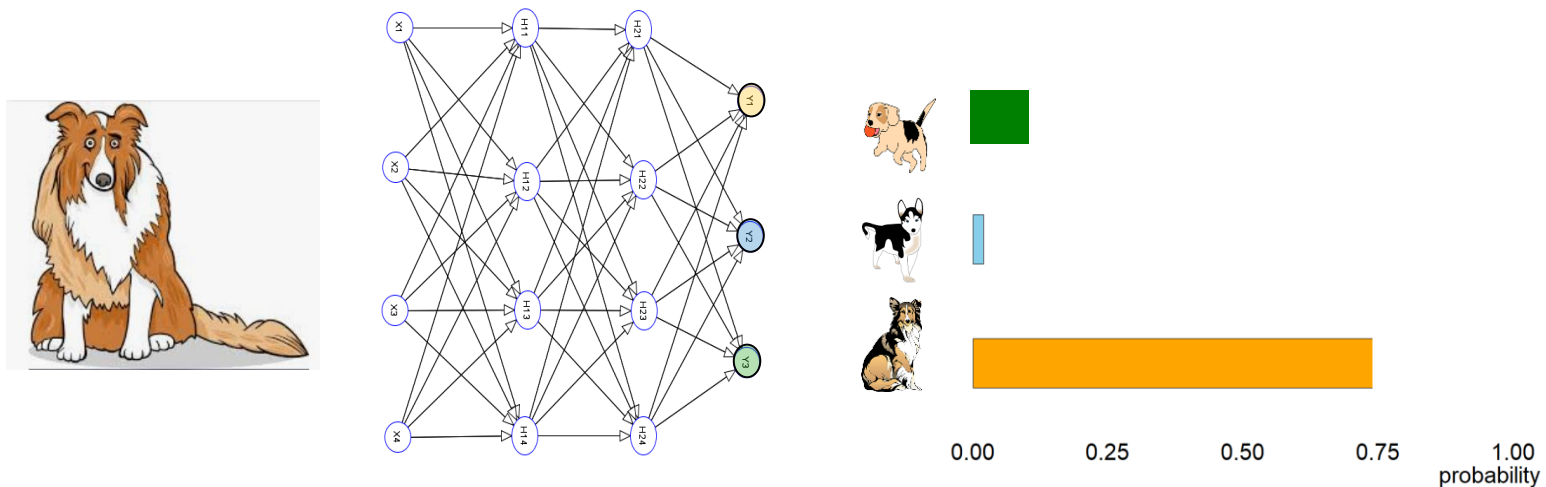
$x^*$ : new input

DL can also be used for regression and classification, but...

Can DL quantify uncertainty?

# 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:  $p_{\max}=0.75$

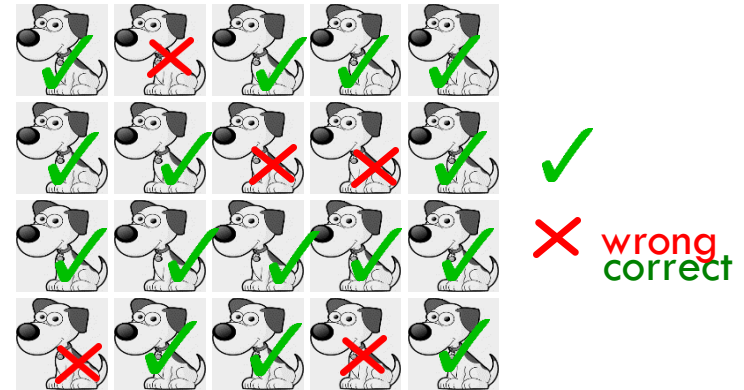
# What is the probability telling?

$$p_{\max}=0.75$$

Among many predications that had  $p_{\max}=0.75$ , we expect that on average 75% of these predictions are correct and only 25% predictions are wrong

→ Then the classifier produces  
*calibrated probabilities*

Sample of images where the predicted class got  $p=0.75$ :



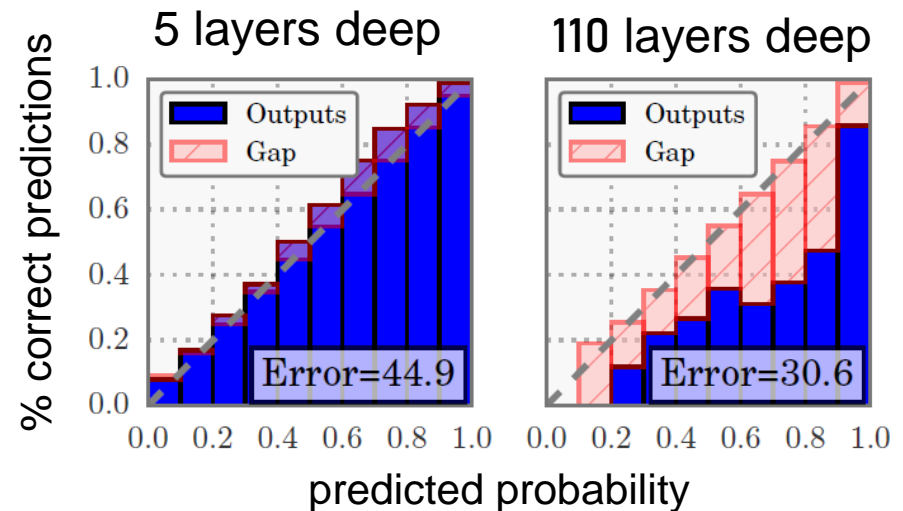
# Do CNNs produce calibrated probabilities?

Guo et al. (2017)

## On Calibration of Modern NN

The deeper CNNs get

- the fewer miss-classifications
- the less well calibrated they get



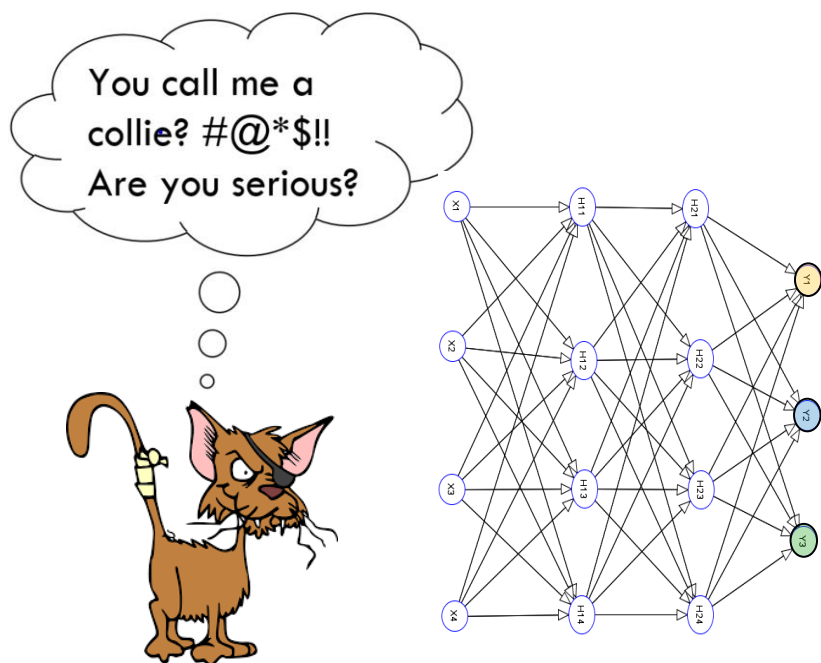
Good news:

deep NN can be “recalibrated” and then we get calibrated probabilities.

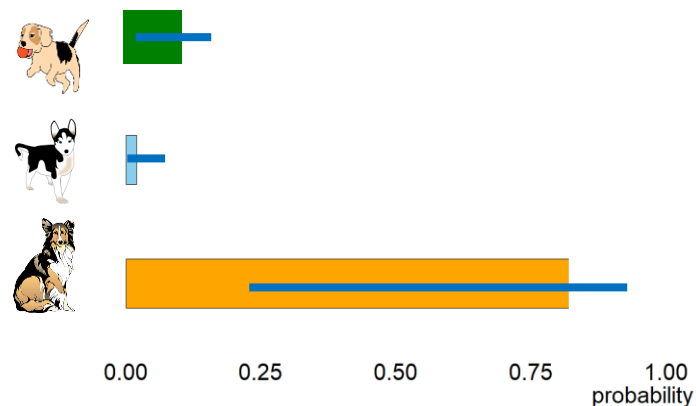


CNNs yield high accuracy  
and calibrated probabilities, but...

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

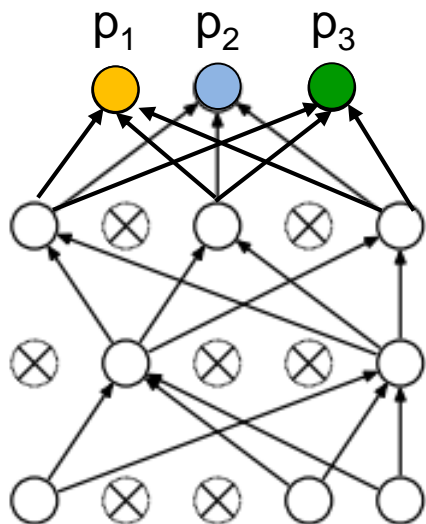


**Plain wrong !**



**We need some error bars!**

# MC Dropout and Bayesian Neural Networks



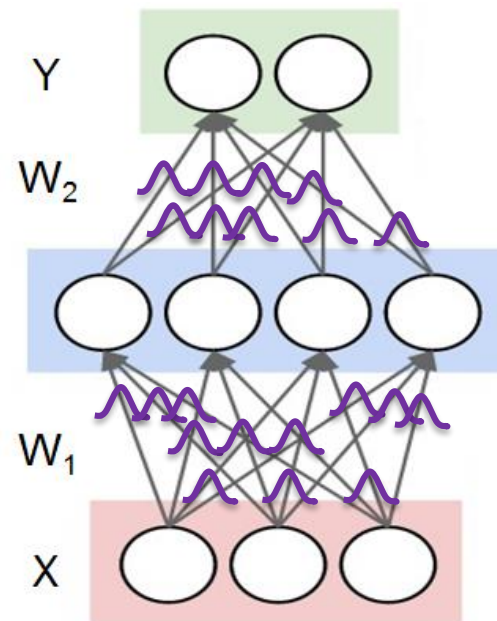
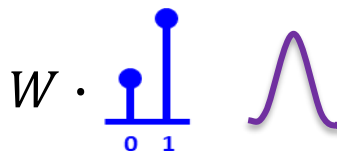
## MC Dropout

Randomly drop nodes in each run  
→ Usually done during training

Dropout in test time



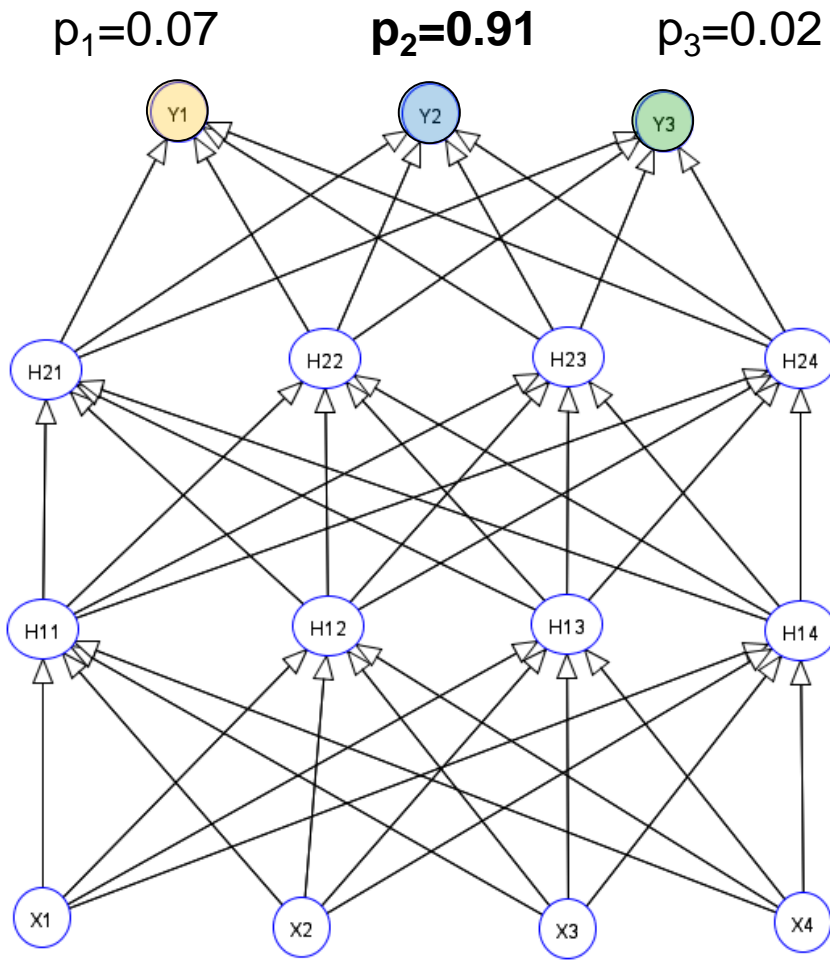
Yarin Gal\* (2015):  
we learn a whole weight distribution



## Bayesian NN

→ We should sample from weight distribution during test time

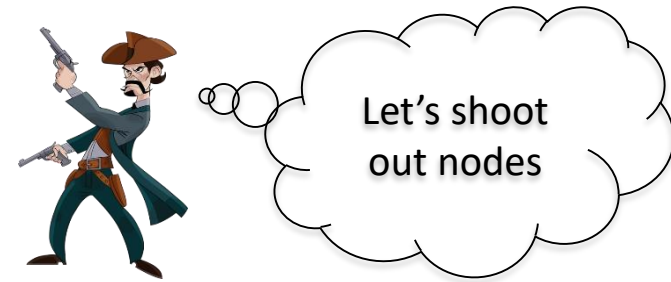
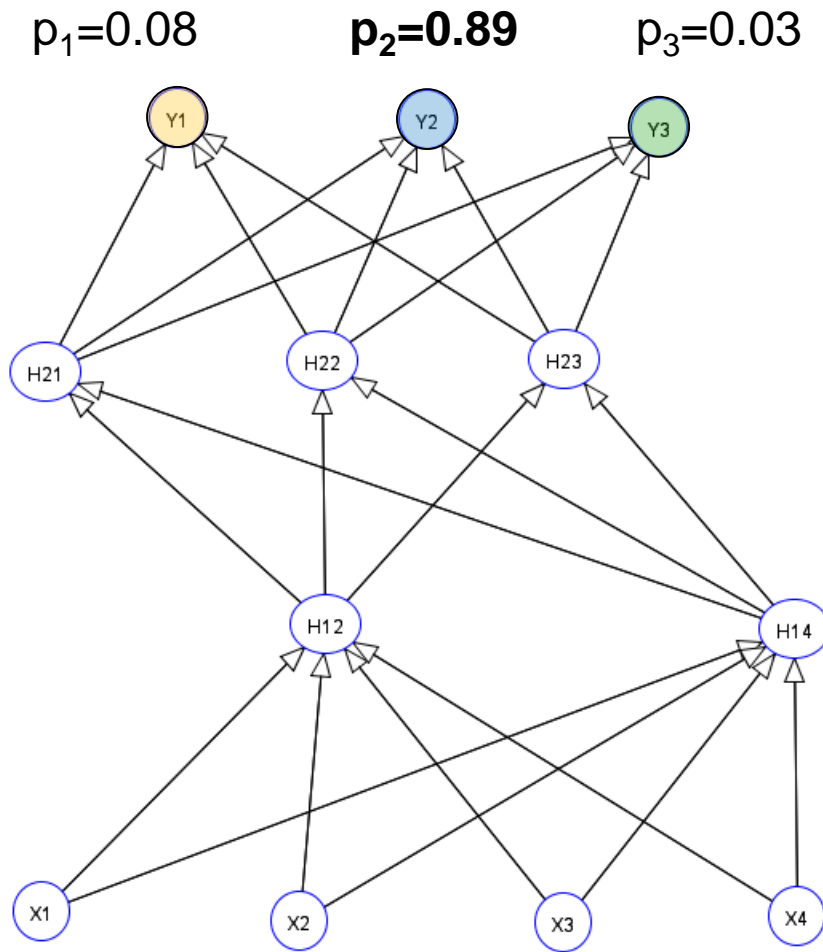
# No MC Dropout



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

Input: image pixel values

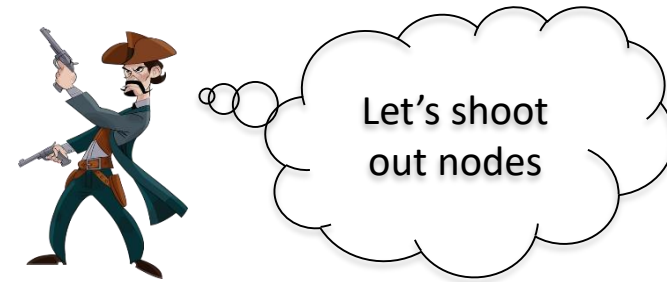
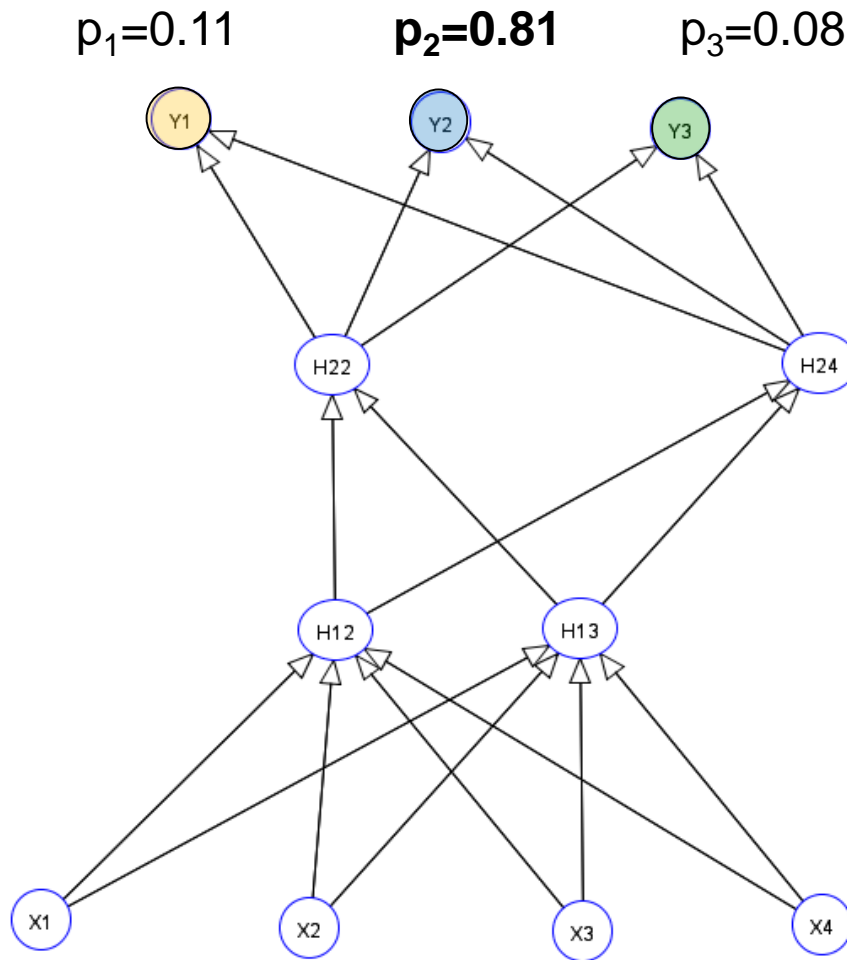
# Use dropout at test time: Run 1



Stochastic dropout of units

Same input image

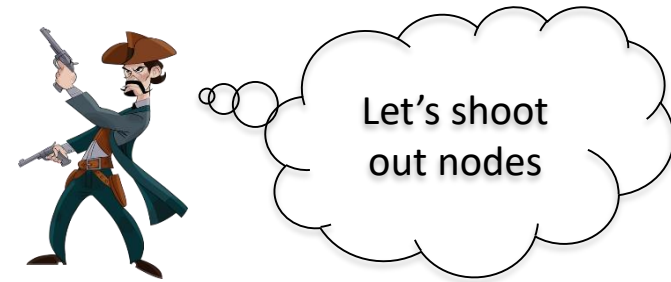
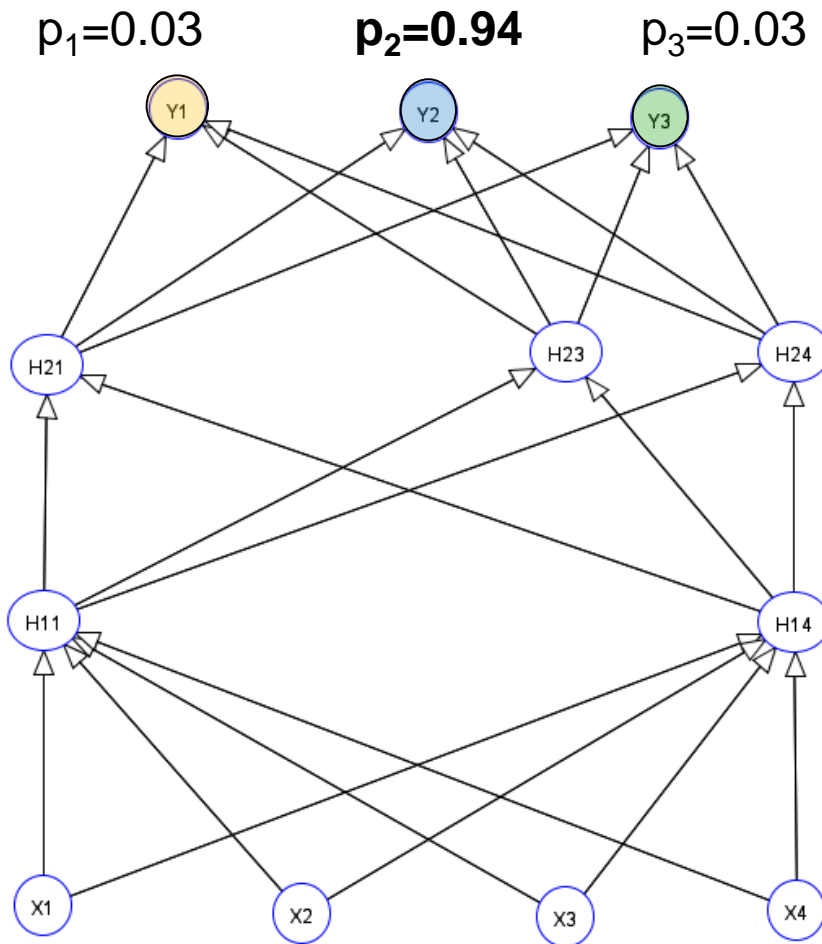
# Use dropout at test time: Run 2



Stochastic dropout of units

Same input image

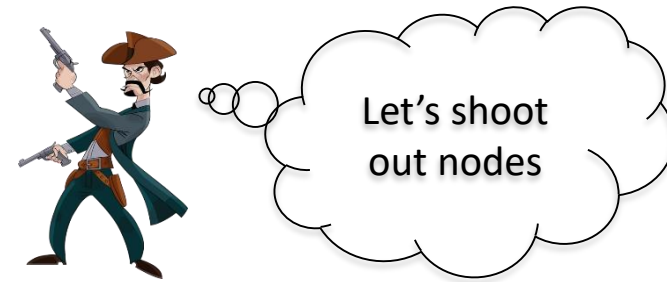
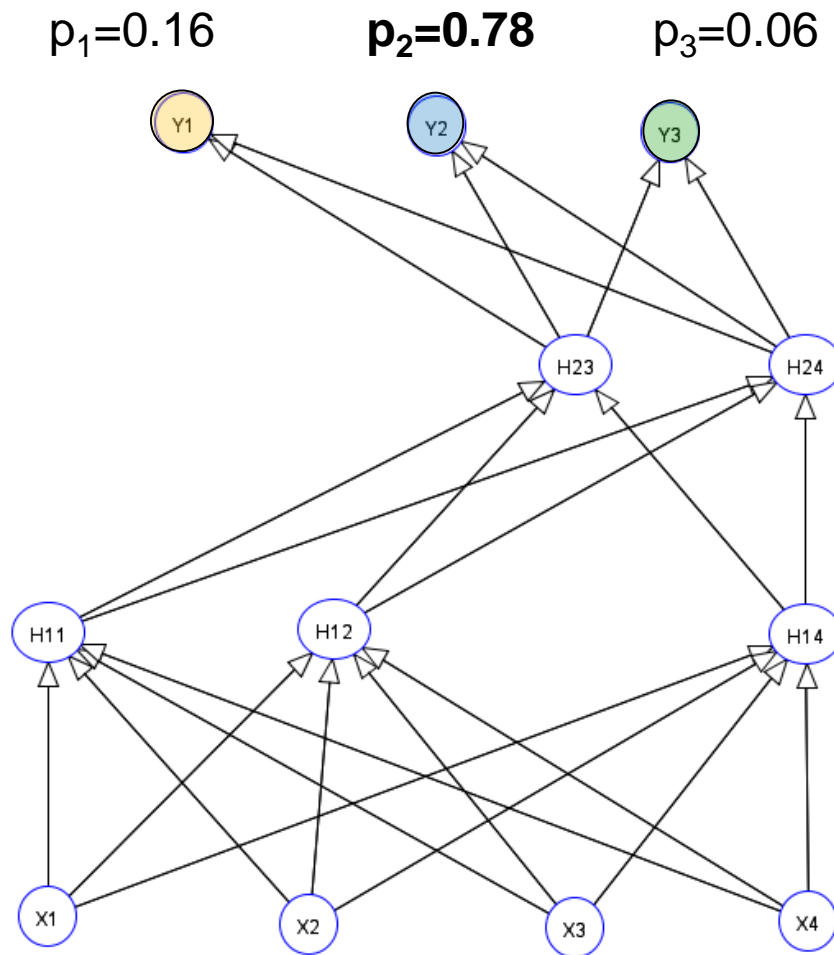
# Use dropout at test time: Run 3



Stochastic dropout of units

Same input image

# Use dropout at test time: Run 4



Stochastic dropout of units

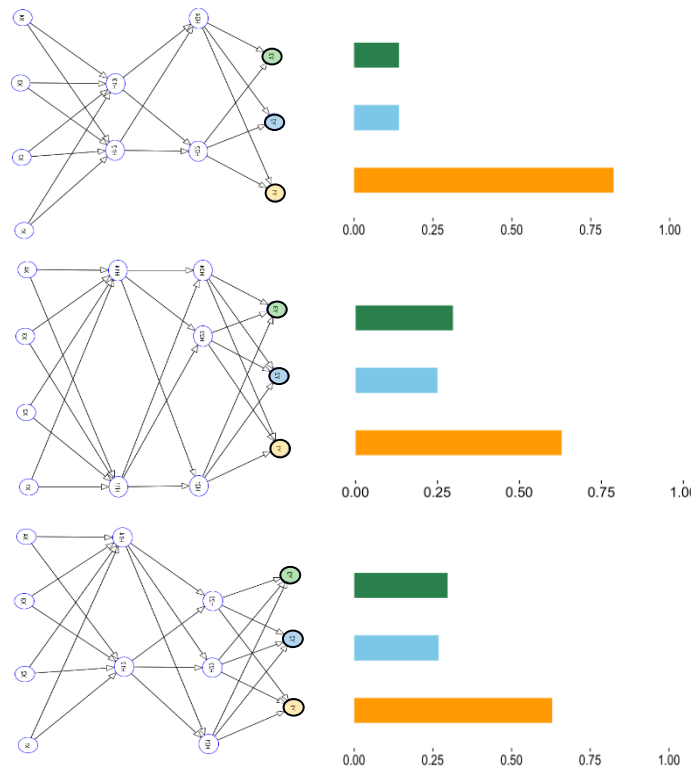
Same input image



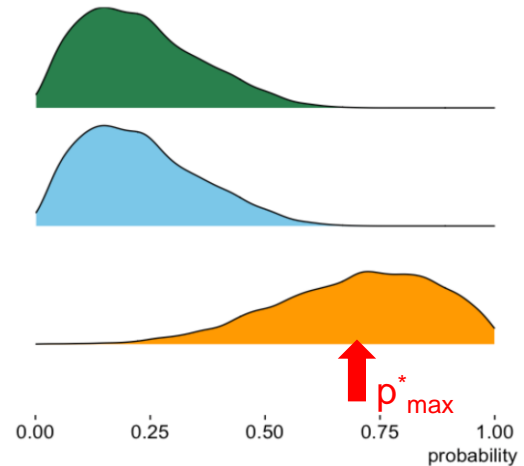
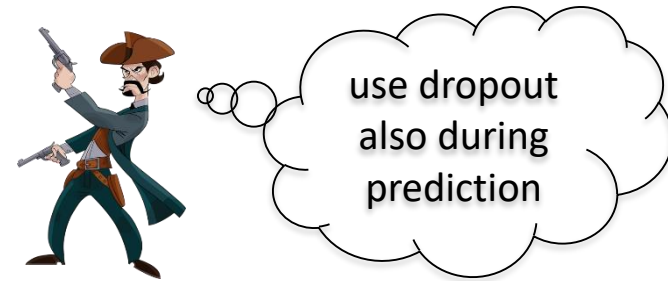
# MC probability prediction



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.

# What to get from the MC\* probability distribution

The **center of mass** quantifies the **predicted probability**

$p_{\max}^*$

The **spread** quantifies in addition the **uncertainty** of the predicted probability

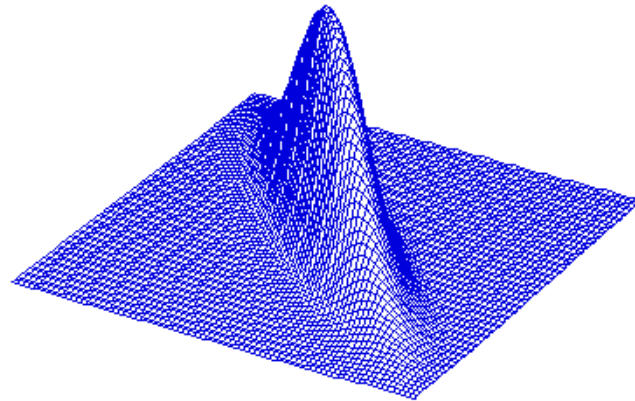
$\sigma^*$  total standard deviation

PE\* entropy,

MI\* mutual information

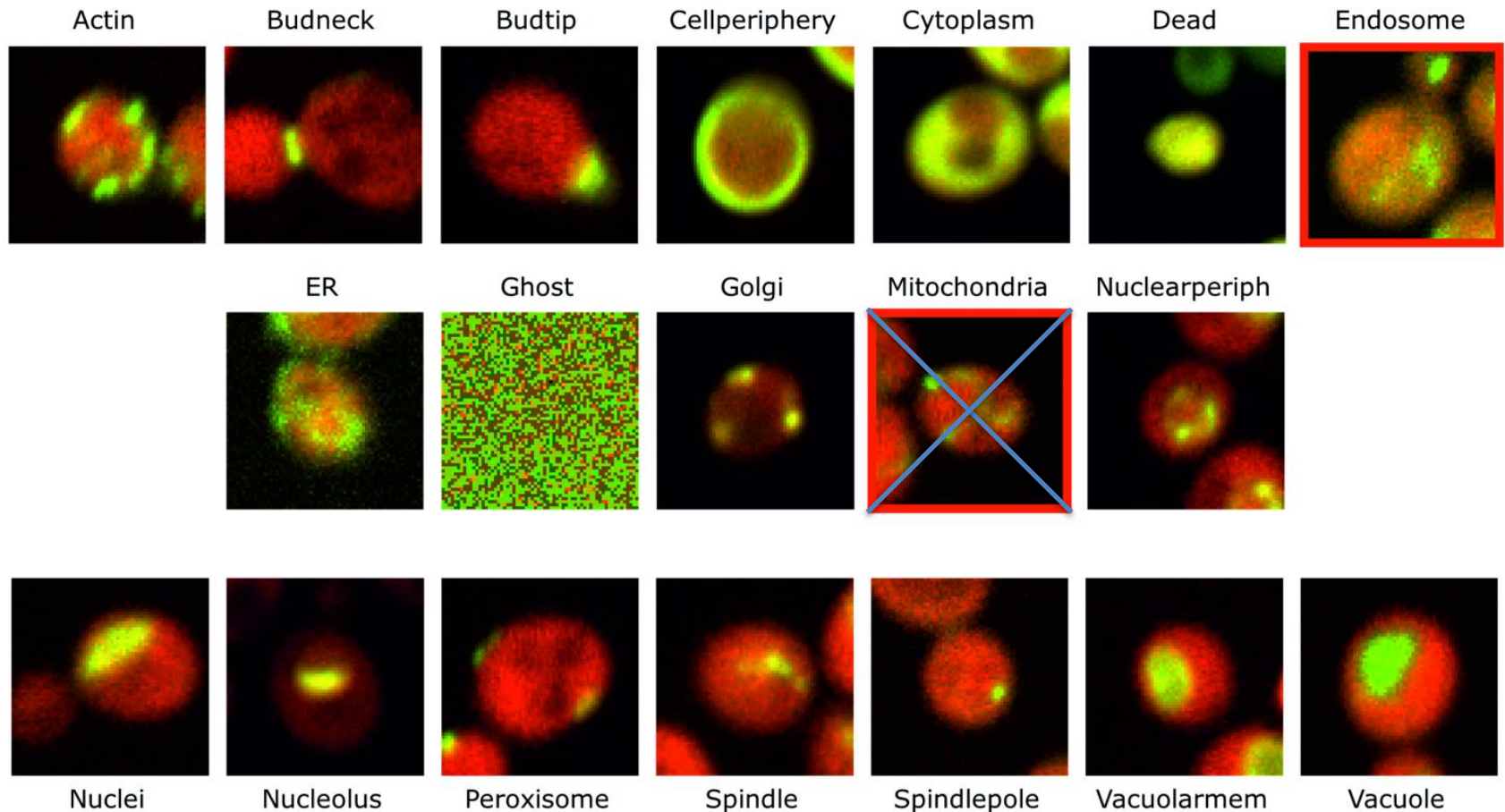
VR\* variation ratio

f\* vote ratio



Evaluate method on some real data

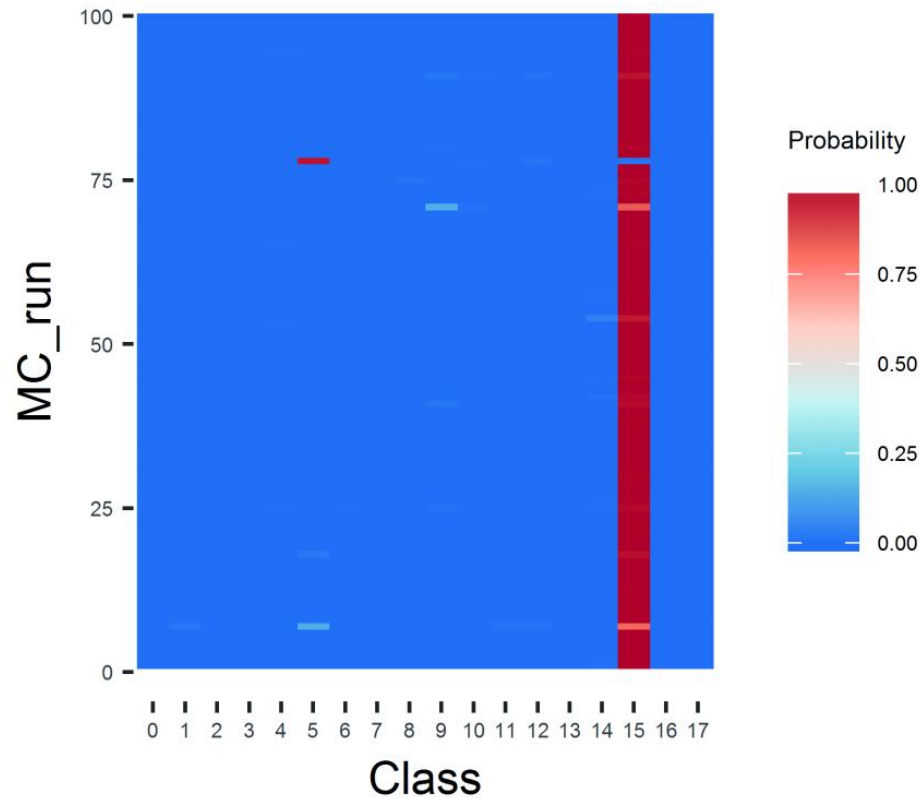
# Experiment with unknown phenotype



# Probability distribution from MC dropout runs

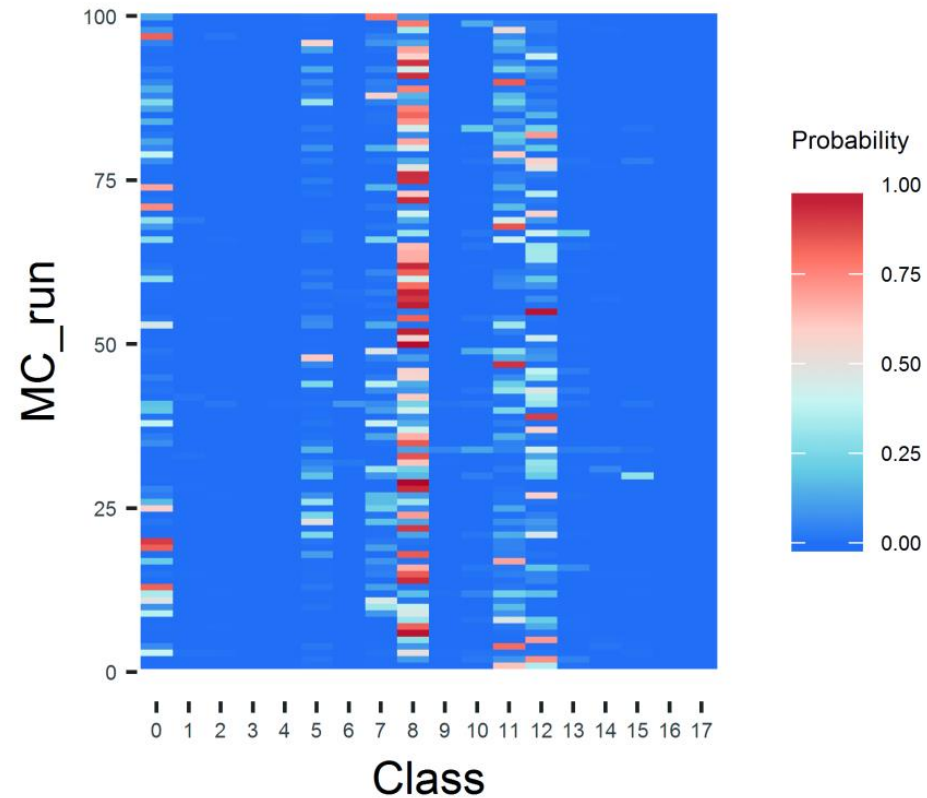
## Image with known class 15

100 MC predictions for an image with known phenotype 15

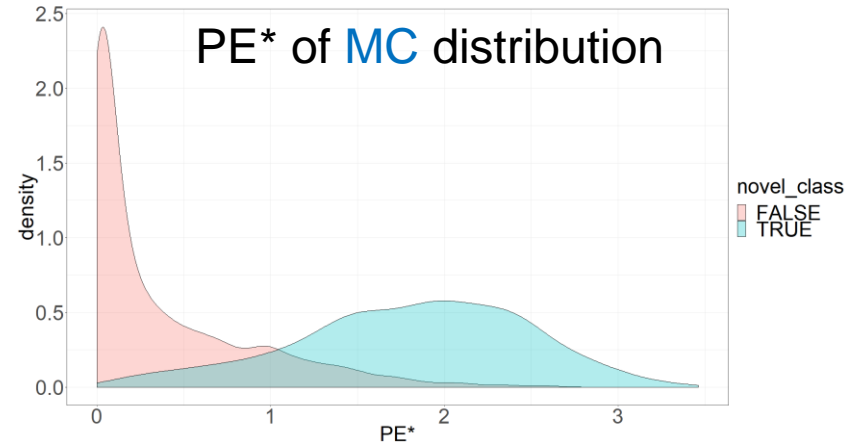
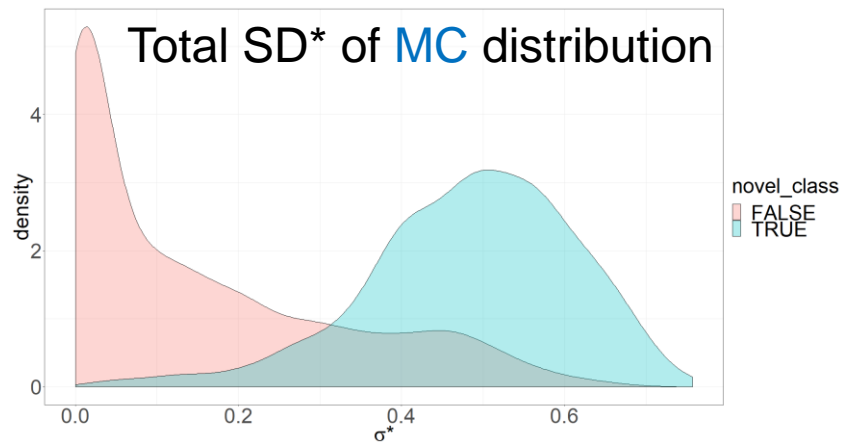
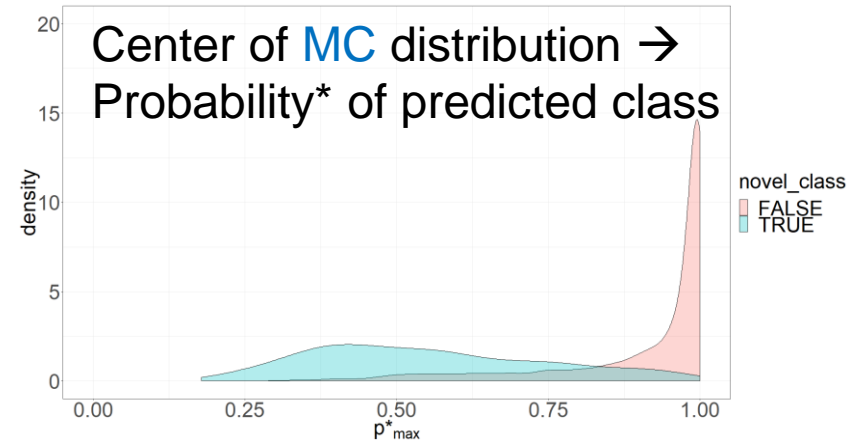
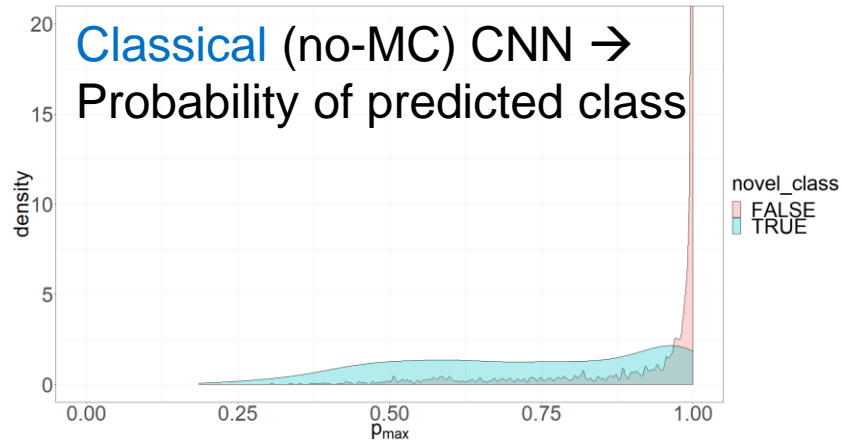


## Image with **unknown** class

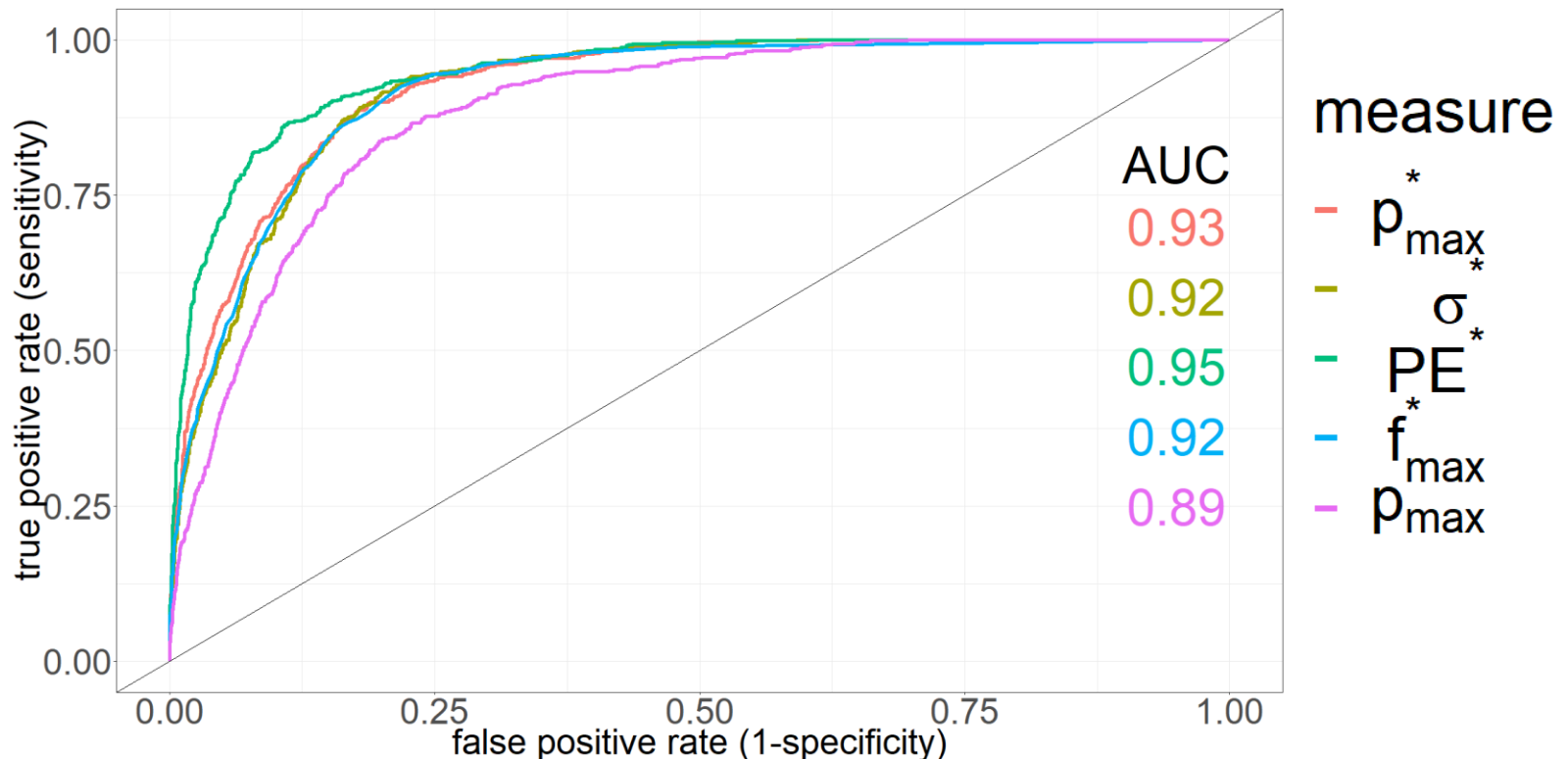
100 MC predictions for an image with an unknown phenotype



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



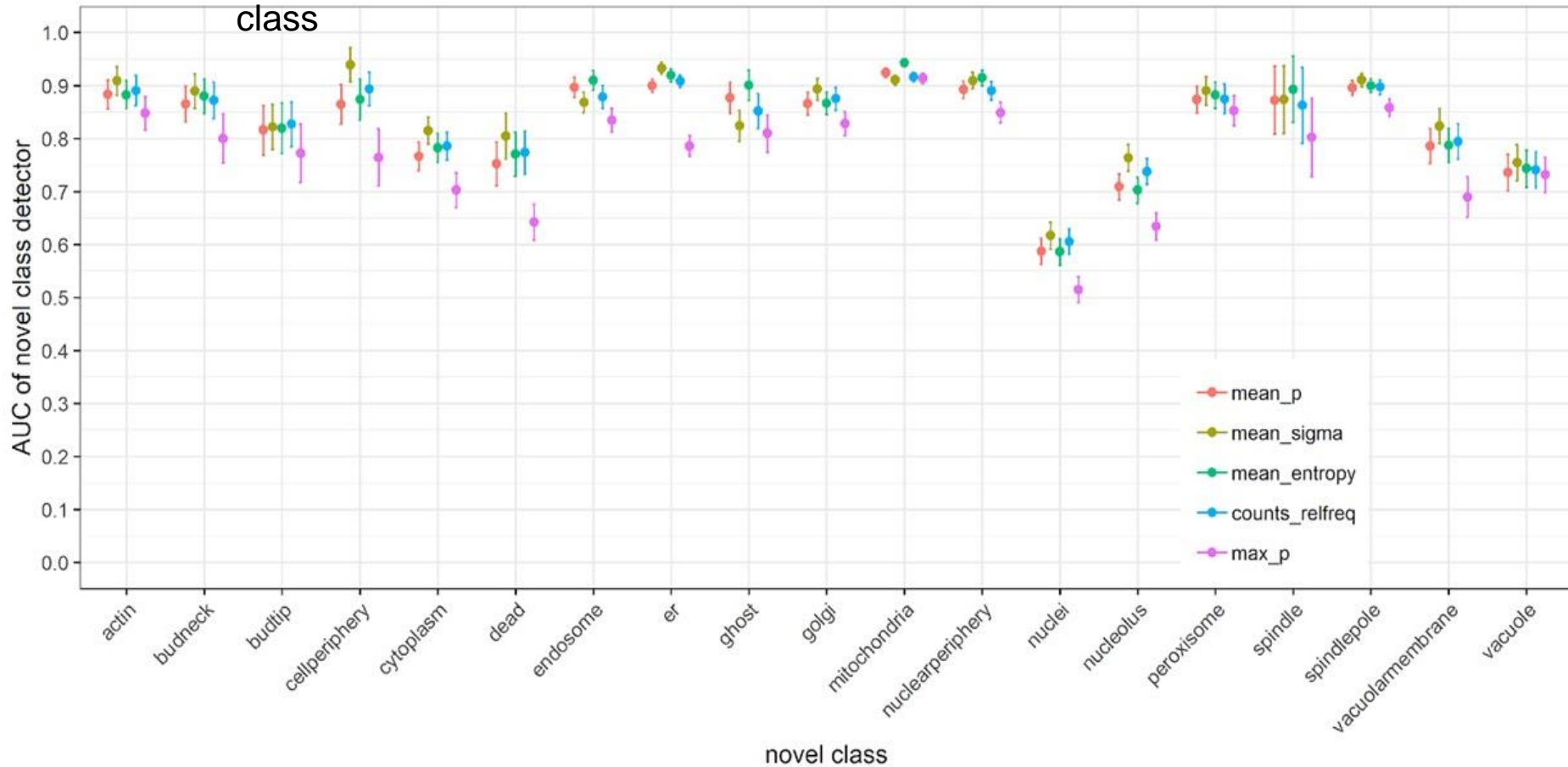
# How good are novel/unusual classes identifiable?



All **MC Dropout based approaches** are superior compared to the **non-MC** approach.

# Dropout uncertainty measures outperform traditional CNN probability

CV experiment where each phenotyp is once in the role of the novel class





# Creating custom layers in keras

```
keras.layers.Dropout(0.5)(x, training=True)
```

Or define a lambda layer which does the trick:

```
from keras.layers.core import Lambda # needed to build the custom layer
from keras import backend as K #Now we have access to the backend (could be tensorflow,
```

Define your custom function

```
def mcdropout(x):
    #return tf.nn.dropout(x=x, keep_prob=0.33333) #using TensorFlow
    return K.dropout(x, level=0.5) # being agnostic of the backend
```

Here you could do anything possible in TF

```
tf.add(x, 10)
```

...

Include your custom function as a layer

```
model = Sequential()
model.add(Lambda(mcdropout, input_shape=(5,)))
#model.add(Dense(10))
#... Usually you would have many more layers
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

# Conclusion

## MC Dropout during test time

- yields uncertainty measures for each individual classification
- helps to identify uncertain cases
- allows to indicate novel classes
- yields new probability estimates leading to higher accuracy