Machine Intelligence:: Deep Learning
Week 6

Beate Sick, Oliver Dürr, Pascal Bühler

Probablistic models:
- count data
- with flexible CPDs

Bayes
- Problems with current DL

Might change slightly before lecture

Outline of the DL Module (tentative)

- Day 1: Jumpstart to DL
  - What is DL
  - Basic Building Blocks
  - Keras
- Day 2: CNN I
  - ImageData
- Day 3: CNN II and RNN
  - Tips and Tricks
  - Modern Architectures
  - 1-D Sequential Data
- Day 4: Looking at details
  - Linear Regression
  - Backpropagation
    - Resnet
  - Likelihood principle
- Day 5: Probabilistic Aspects
  - Likelihood principle (cont'd)
  - TensorFlow Probability (TFP)
  - Negative Loss Likelihood NLL
- Day 6: Probabilistic models in the wild
  - Complex Distributions
  - Count Data
  - Bayesian Modeling
    - Elephant in the room
- Day 7: Uncertainty in DL
  - Bayesian Modeling
- Day 8: Uncertainty cont'd
  - Bayesian Neural Networks
  - Projects

Projects please register (see website)
https://docs.google.com/spreadsheets/d/18VFrPbKq3YSOg8Ebc1q1wGgkfgaWI7IkcCCIGEDGj6Q/edit#gid=0

Besprechung der Aufgabe 13

13_linreg_with_tfp

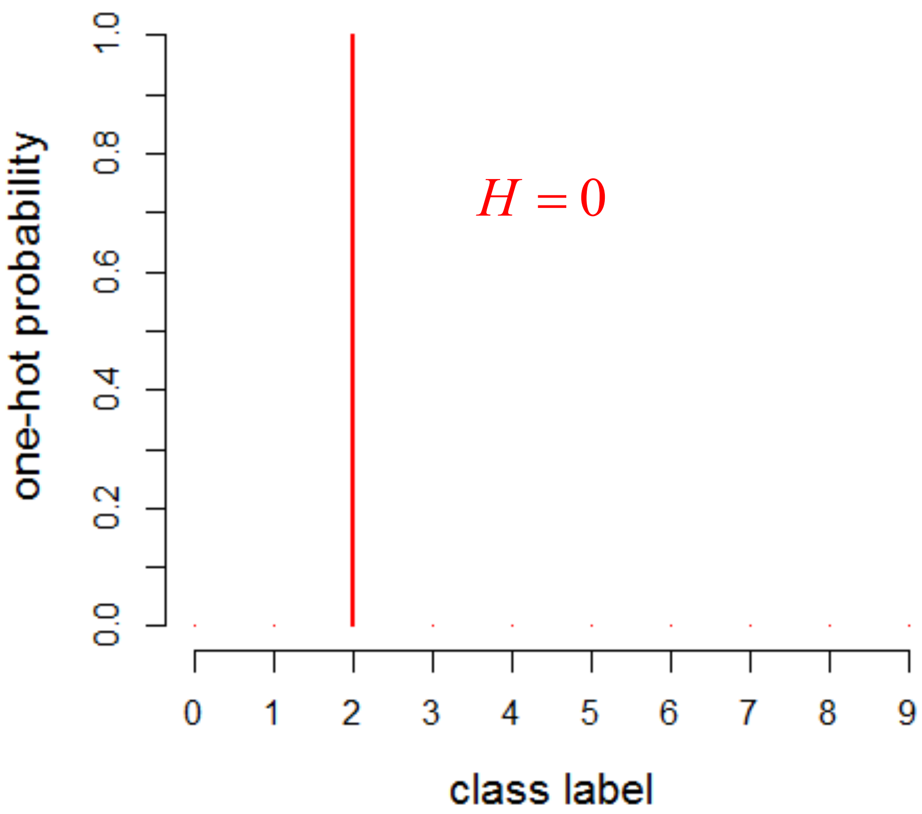
Side track:

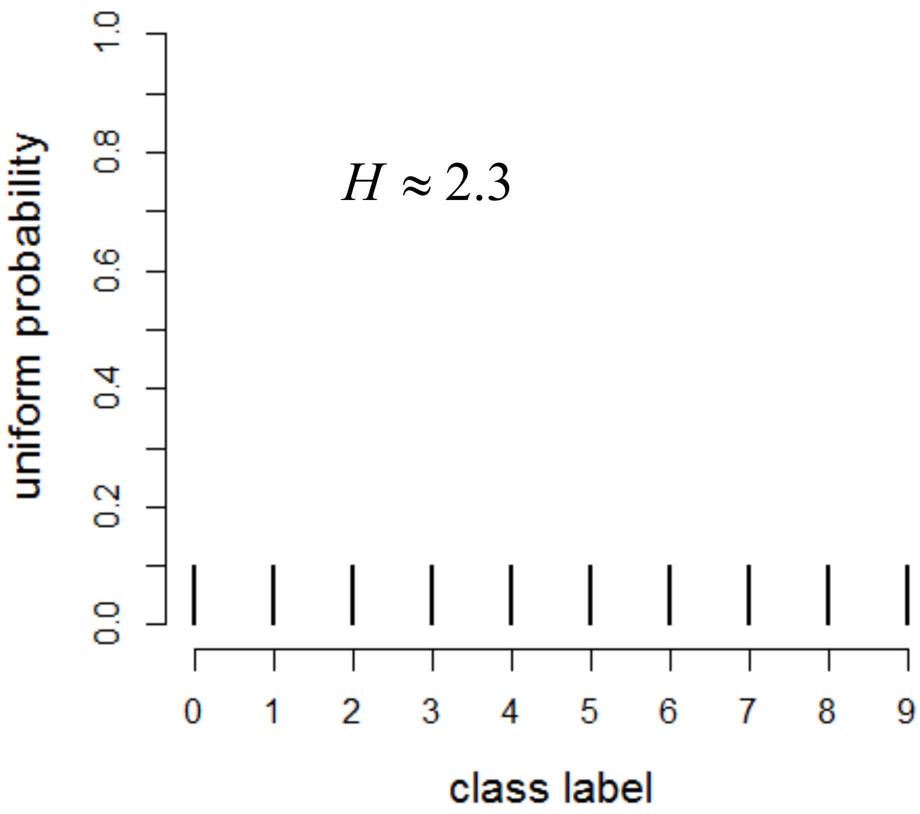
Why the name X-entropy?

Side track: Entropy

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

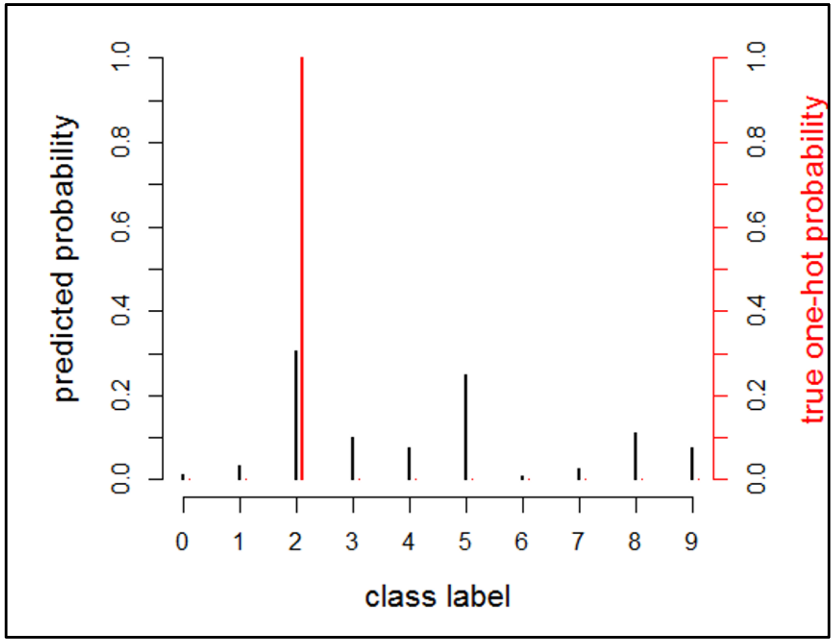
Entropy is a measure for “untidyness”.
If a distribution has only one peak, it is tidy and H=0
If all outcomes are equally probable it is maximal untidy





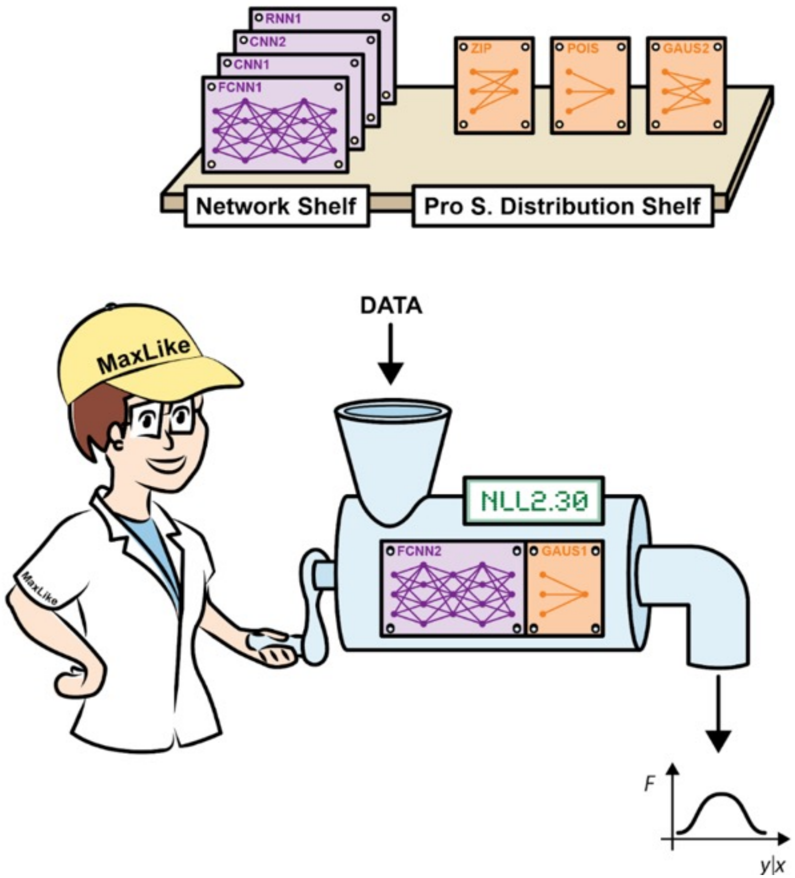
X-entropy

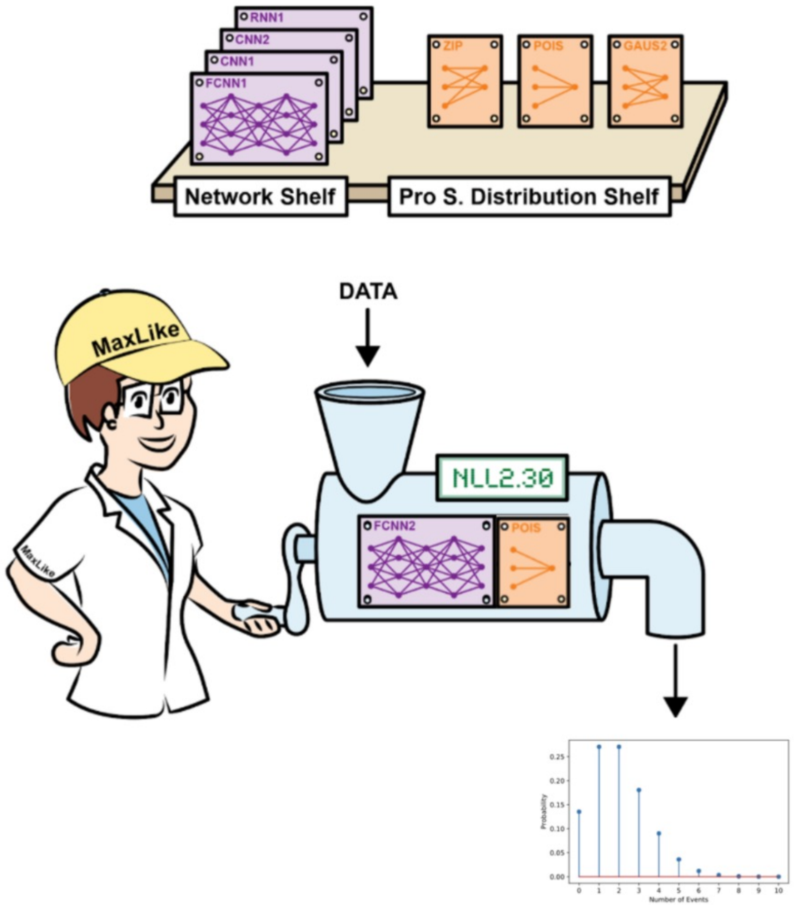
$$\text{X−entropy} = \sum_i \mathrm{H}({}^{true}\mathrm{p}_i, {}^{pred}\mathrm{q}_i) = \underbrace{-\sum_i {}^{true}\mathrm{p}_i \cdot \log({}^{pred}\mathrm{q}_i)}_{\text{Cross-entropy of } {}^{true}\text{p and } {}^{true}\text{q}} = \mathrm{NLL}$$



The cross-entropy is the same as the NLL, if the true probability distribution puts the whole weight on one class.

We have a flexible tool where the choice of the architecture and the choice of the outcome distribution is independent





Modeling count data:

M2: Poisson regression

The camper example

N=250 groups visiting a national park

Y=count: number of fishes cought

X1=persons: number of persons in group

X2=child: number of children in the group

X3=bait: indicates of life bait was used

X4=camper: indicates if camper is brought

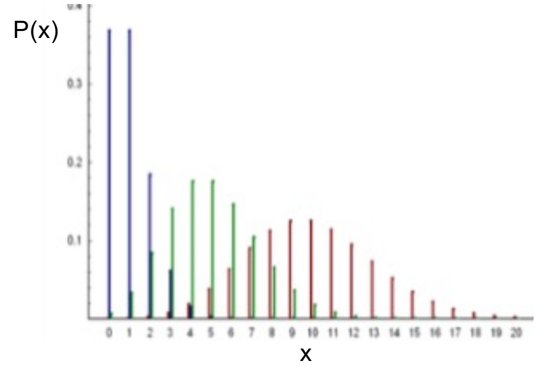
Data: https://stats.idre.ucla.edu/r/dae/zip

Recall the classical Poisson model for count data

X: number of incidences **per time unit**

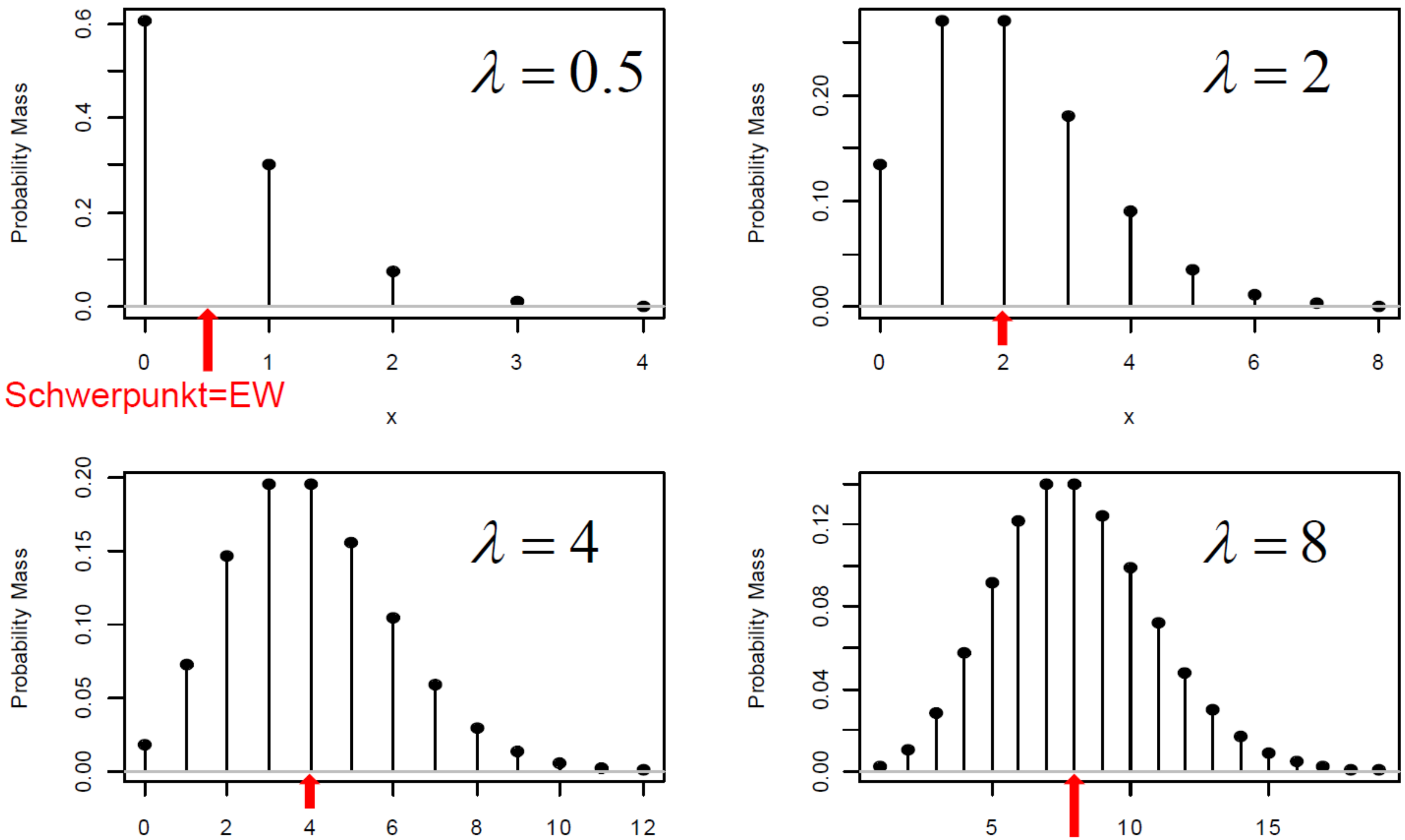
The Poisson model is appropriate to model **counts X per unit**, assuming that
1) the unknown incidence rate λ (per time unit) is constant and
2) the incidences occur independently

$$P(X = x) = \frac{\lambda^x}{x!} e^{-\lambda} \; , \qquad x = 0, 1, 2, \ldots$$



$$E(X) = Var(X) = \lambda$$: expected number of counts per time unit

The shape and mean of the Poisson distribution depends on λ



The Poisson distribution in tfp

```
dist = tfd.poisson.Poisson(rate = 2) #A

vals = np.linspace(0,10,11) #B

p = dist.prob(vals) #C
```

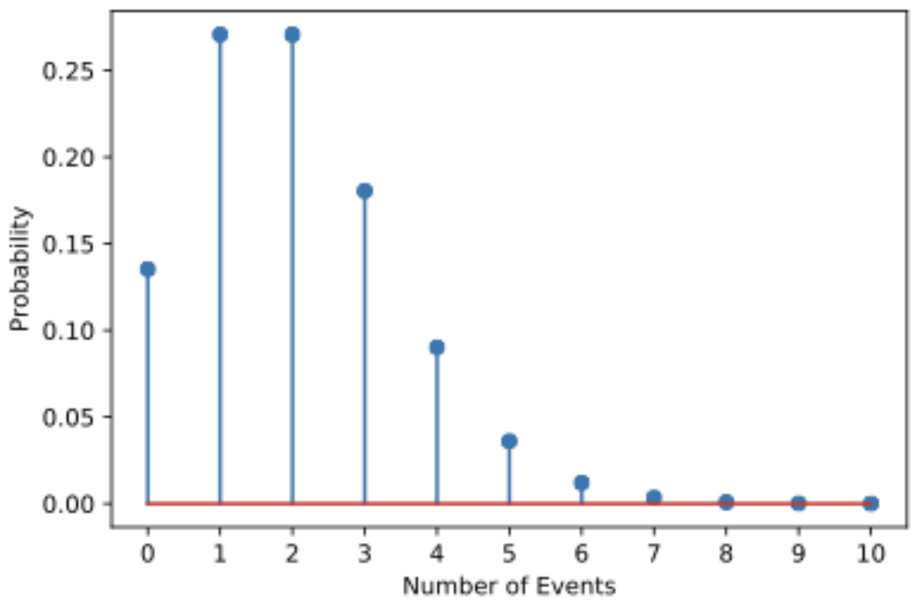


Figure 5.13 Poisson distribution for the case that there are, on average, two events per unit

Poisson regression for count data

Goal: Predict a Poisson CPD for (Y|X=x) which depends on predictor values

CPD: $Y_{X_i} = (\mathrm{Y}|\mathrm{X}_i) \sim \mathrm{Pois}(\lambda_{x_i})$

We only need to model λ_x to fix the Poisson CPD!

Statistical Model:

$$\log(\lambda_i) = \underbrace{\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}}_{\text{Linear predictor } \eta_i}$$

link-function: ensures positive λ after back-transformation

Neural Network

$$\lambda_{x_i} = exp(z_i)$$

Model 2: Poisson regression via NNs in keras

We use a NN without hidden layer to control the rate λ.

```
inputs = Input(shape=(X_train.shape[1],))
rate = Dense(1,
         activation=tf.exp)(inputs) #A
p_y = tfp.layers.DistributionLambda(tfd.Poisson)(rate)
```

Using the exp-activation ensures that the rate λ is a positive number

```
model_p = Model(inputs=inputs, outputs=p_y) #C
```

Glueing the NN and the output layer together. Note that output p_y is a tf.distribution

```
def NLL(y_true, y_hat): #D
  return -y_hat.log_prob(y_true)
```

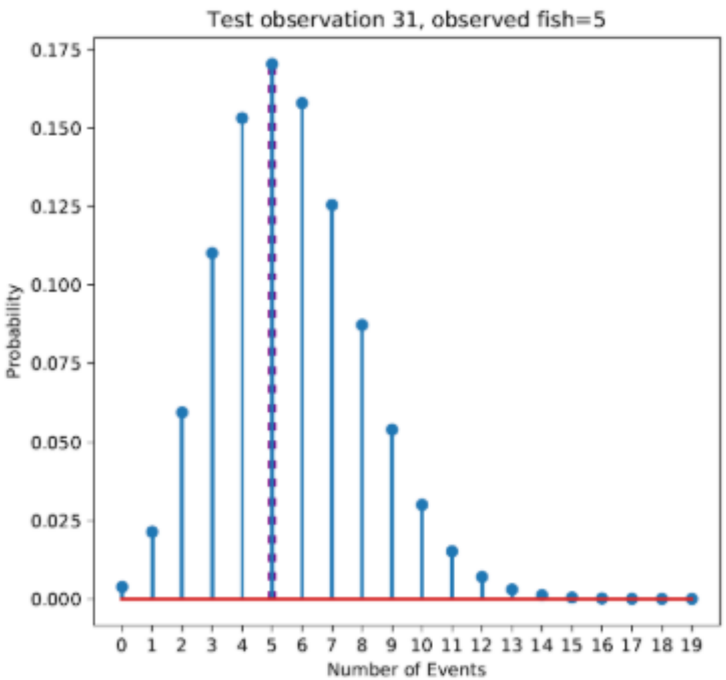
The second argument is the output of the model and thus a TFP distribution. It's as simple as calling log_prob to calculate the log probability of the observation that's needed to calculate the NLL

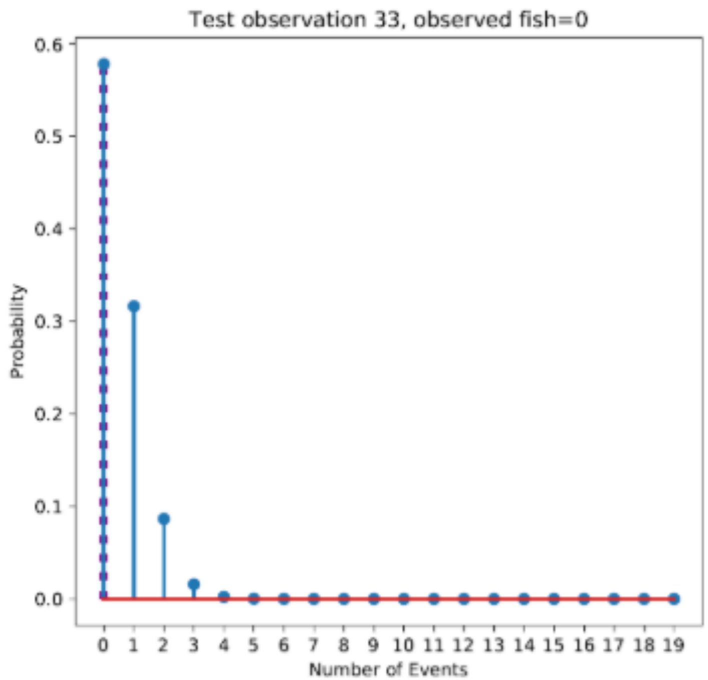
```
model_p.compile(Adam(learning_rate=0.01), loss=NLL)
model_p.summary()
```

Model 2: Poisson regression, get test NLL from Gaussian CPD

Predict CPD for outcome in test data:

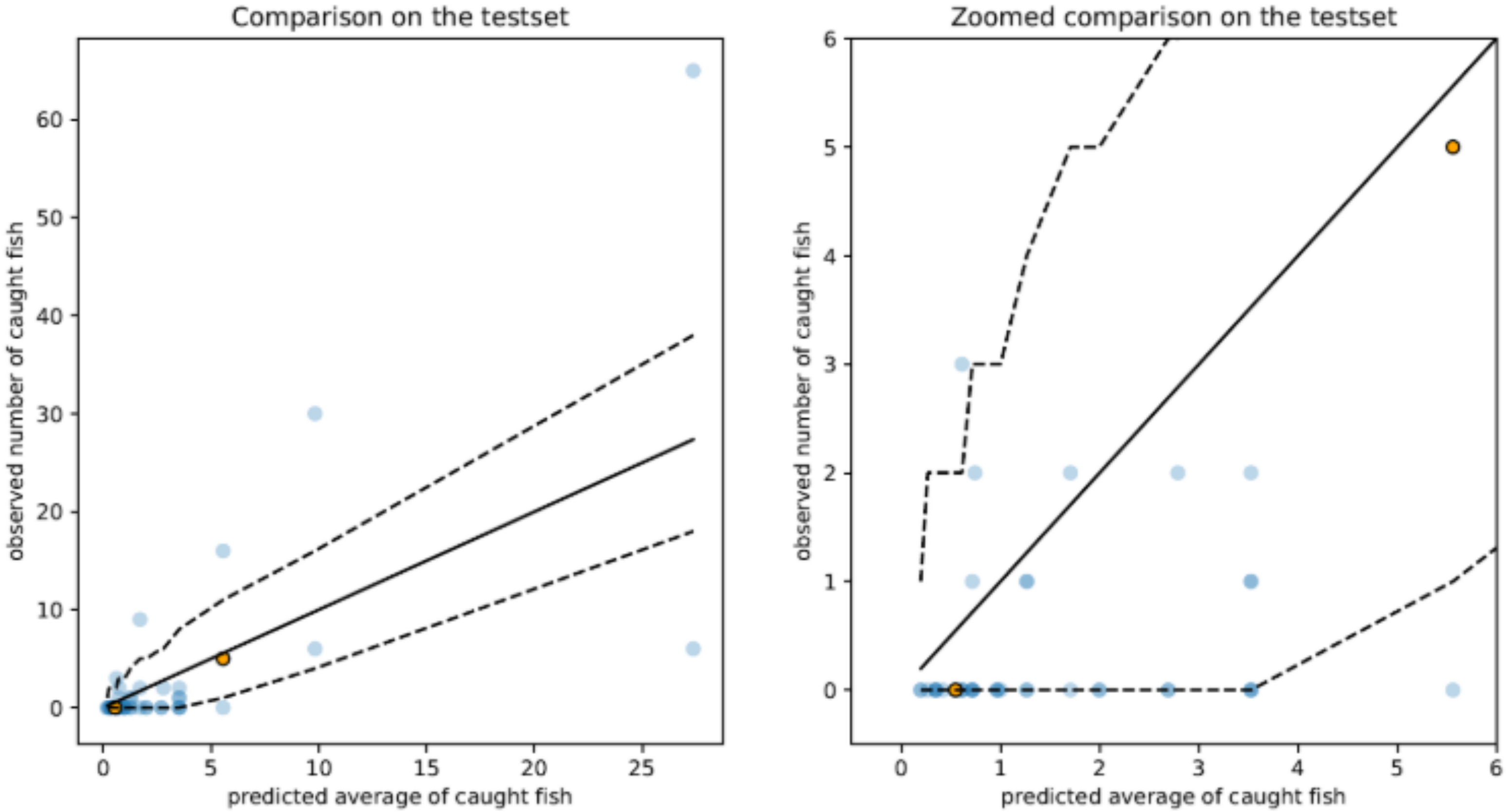
Group 31 used livebait, had a camper and were 4 persons with one child. Y=5 fish.
Group 33 used livebait, didn't have a camper and were 4 persons with two childern. Y=0 fish.





What is the likelihood of the observed outcome in test obs 31 and 33?

Model 2: Poisson regression, visualize the CPDs by quantiles



The mean of the CPD is depicted by the solid lines.
The dashed lines represent the 0.025 and 0.975 quantiles, yielding the borders of a 95% prediction interval.
Note that different combinations of predictor values can yield the same parameters of the CPD.

Ufzgi

- 14_poisreg_with_tfp.ipynb
- https://github.com/tensorchiefs/dl_course_2022/blob/master/notebooks/14_poisreg_with_tfp.ipynb

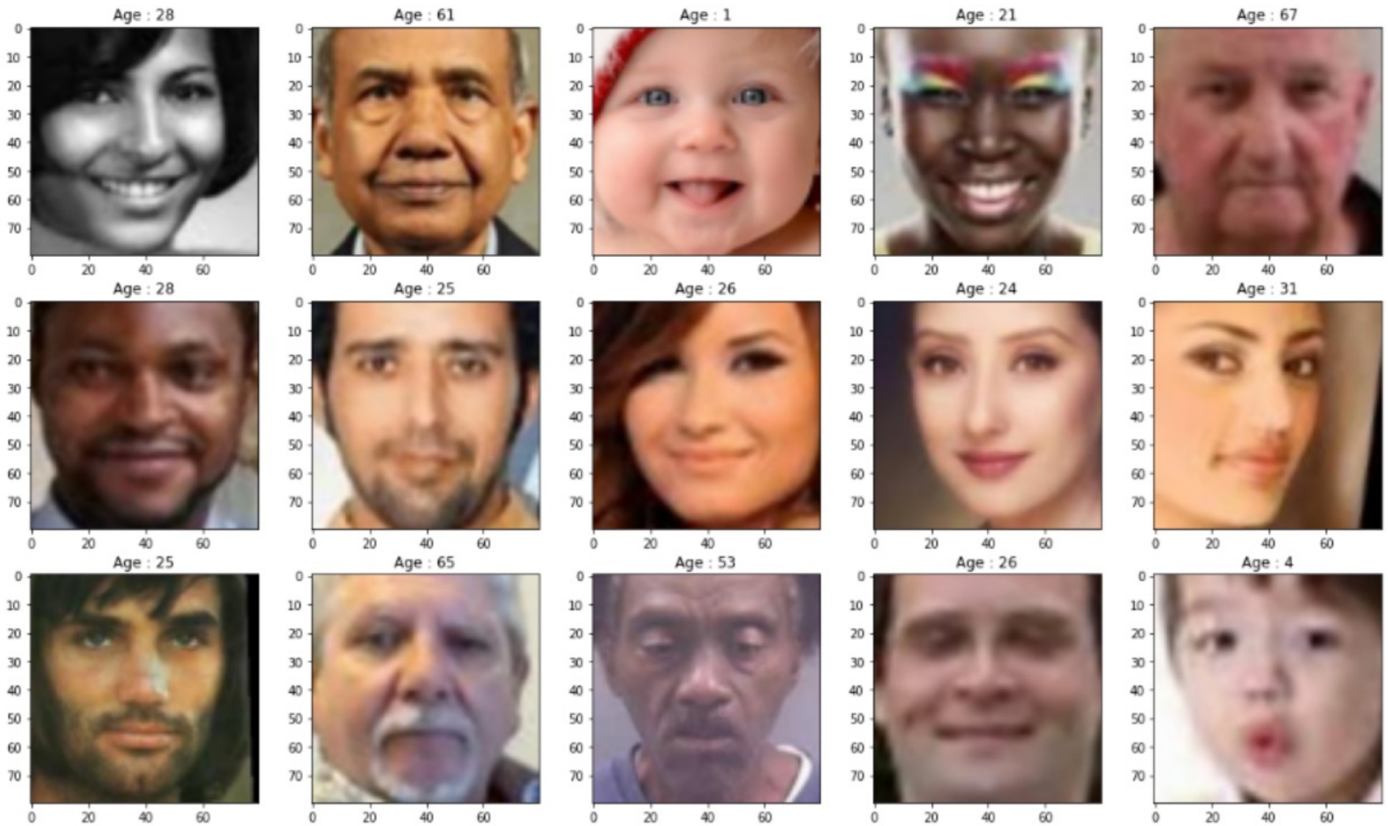
Besprechung der Aufgabe

Use NN and tfp to fit a poisson model

https://github.com/tensorchiefs/dl_course_2022/blob/master/notebooks/14_poisreg_with_tfp.ipynb

Probabilistic models with complex input data

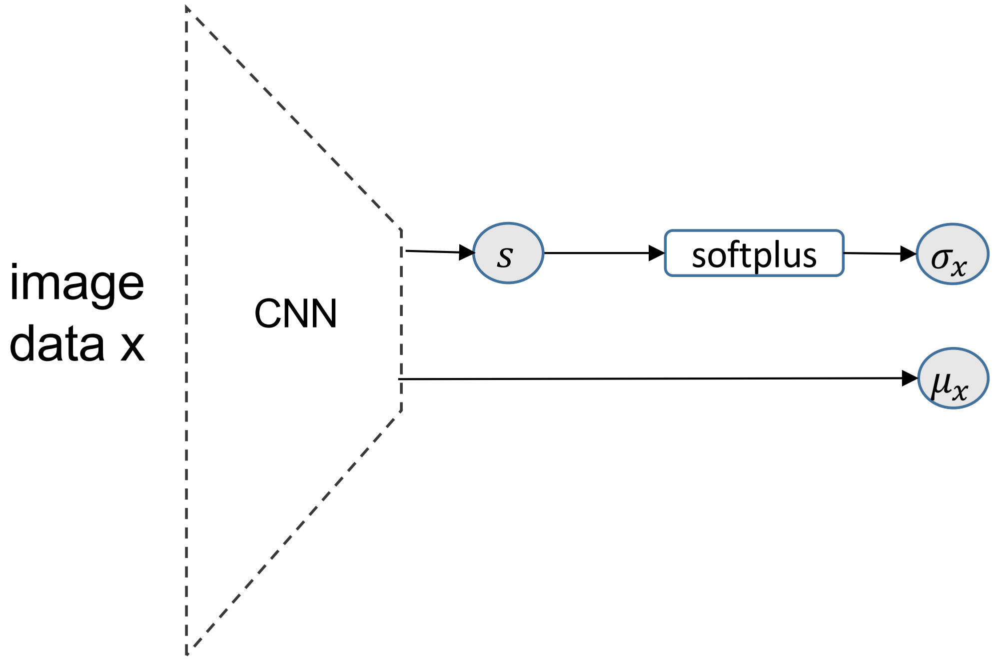
The UTK face data – face image data with known age



Data: https://stats.idre.ucla.edu/r/dae/zip

UTKFace data set containing N= 23′708 images of cropped faces of humans with known age ranging from 1 to 116 years.

Modeling a Gaussian CPD with flexible mean & variance



Minimize the negative log-likelihood (NLL):

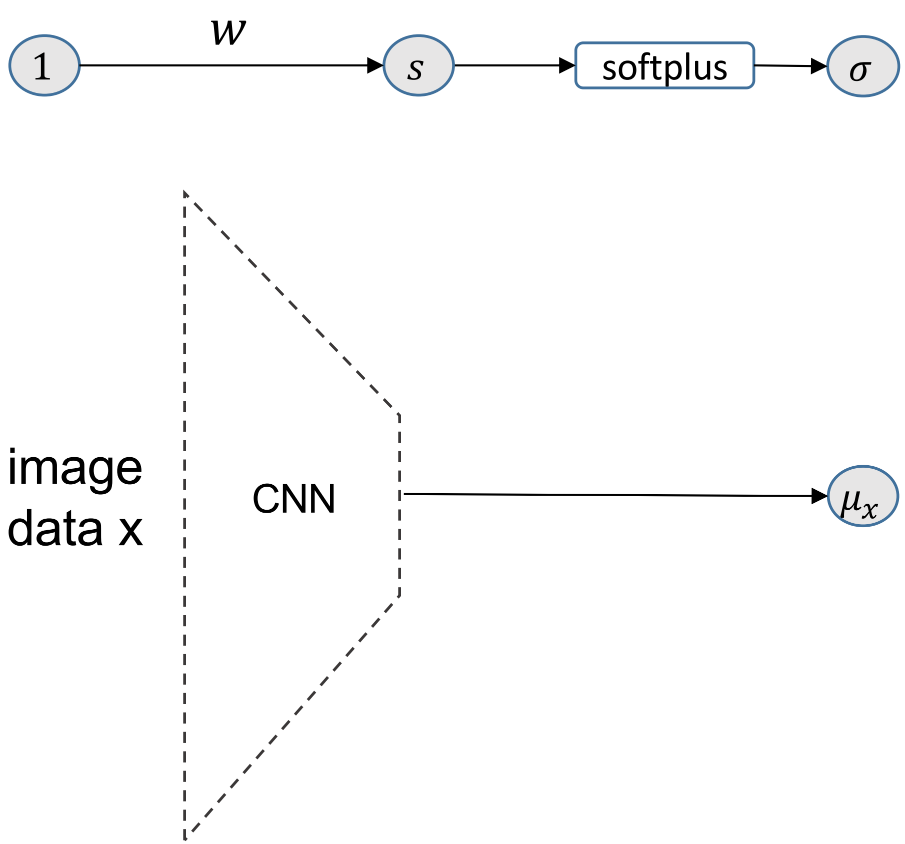
$$\hat{\mathbf{w}}_{\mathrm{ML}} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} -\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \frac{\left(y_i - \mu_{x_i}\right)^2}{2\sigma^2}$$

CNNs for modeling Gaussian CPDs

```
def NLL(y, distr):
  return -distr.log_prob(y)

def my_dist(params):
  return tfd.Normal(loc=params[:,0:1], scale=1e-3 + tf.math.softplus(0.05 * params[:,1:2]))# both paramete
rs are learnable

inputs = Input(shape=(80,80,3))
x = Convolution2D(16,kernel_size,padding='same',activation="relu")(inputs)
x = Convolution2D(16,kernel_size,padding='same',activation="relu")(x)
x = MaxPooling2D(pool_size=pool_size)(x)

x = Convolution2D(32,kernel_size,padding='same',activation="relu")(x)
x = Convolution2D(32,kernel_size,padding='same',activation="relu")(x)
x = MaxPooling2D(pool_size=pool_size)(x)

x = Convolution2D(32,kernel_size,padding='same',activation="relu")(x)
x = Convolution2D(32,kernel_size,padding='same',activation="relu")(x)
x = MaxPooling2D(pool_size=pool_size)(x)

x = Flatten()(x)
x = Dense(500,activation="relu")(x)
x = Dropout(0.3)(x)
x = Dense(50,activation="relu")(x)
x = Dropout(0.3)(x)
x = Dense(2)(x)
dist = tfp.layers.DistributionLambda(my_dist)(x)

model_flex = Model(inputs=inputs, outputs=dist)
model_flex.compile(tf.keras.optimizers.Adam(), loss=NLL)
```

We control both parameters (μ_x, σ_x) of a Gaussian CPD $N(\mu_x, \sigma_x)$ by a CNN
→ More flexible than in classical regression where $\sigma = constant$

Modeling Gaussian CPD with flexible mean & constant variance



Minimize the negative log-likelihood (NLL):

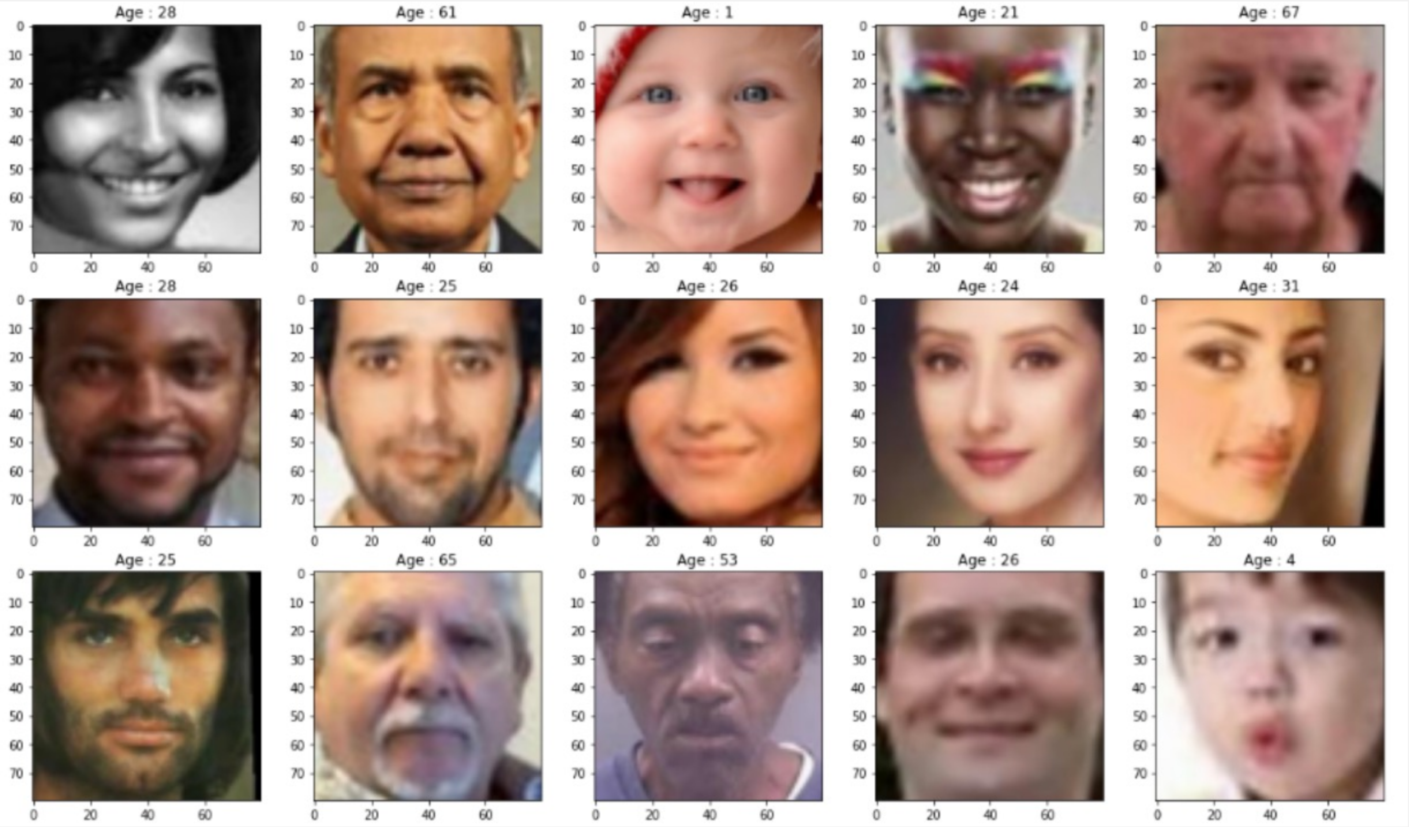
$$\hat{\mathbf{w}}_{\text{ML}} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} -\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \frac{\left(y_i - \mu_{x_i}\right)^2}{2\sigma^2}$$

CNNs for modeling Gaussian CPDs

```
def NLL(y, distr):
  return -distr.log_prob(y)

def my_dist(params):
  return tfd.Normal(loc=params[:,0:1], scale=1e-3 + tf.math.softplus(0.05 * params[:,1:2]))# both paramete
rs are learnable

input1 = Input(shape=(80,80,3))
input2 = Input(shape=(1,))
x = Convolution2D(16,kernel_size,padding='same',activation="relu")(input1)
x = Convolution2D(16,kernel_size,padding='same',activation="relu")(x)
x = MaxPooling2D(pool_size=pool_size)(x)

x = Convolution2D(32,kernel_size,padding='same',activation="relu")(x)
x = Convolution2D(32,kernel_size,padding='same',activation="relu")(x)
x = MaxPooling2D(pool_size=pool_size)(x)

x = Convolution2D(32,kernel_size,padding='same',activation="relu")(x)
x = Convolution2D(32,kernel_size,padding='same',activation="relu")(x)
x = MaxPooling2D(pool_size=pool_size)(x)

x = Flatten()(x)
x = Dense(500,activation="relu")(x)
x = Dropout(0.3)(x)
x = Dense(50,activation="relu")(x)
x = Dropout(0.3)(x)
out1 = Dense(1)(x)
out2 = Dense(1)(input2)
params = Concatenate()([out1,out2])
dist = tfp.layers.DistributionLambda(my_dist)(params) #

model_const_sd = Model(inputs=[input1,input2], outputs=dist) ## use a trick with two inputs, input2 is jus
t ones
model_const_sd.compile(tf.keras.optimizers.Adam(), loss=NLL)
```

We control both parameters (μ, σ) of a Gaussian CPD $\mathrm{N}(\mu_x, \sigma)$ → But assume a constant variance

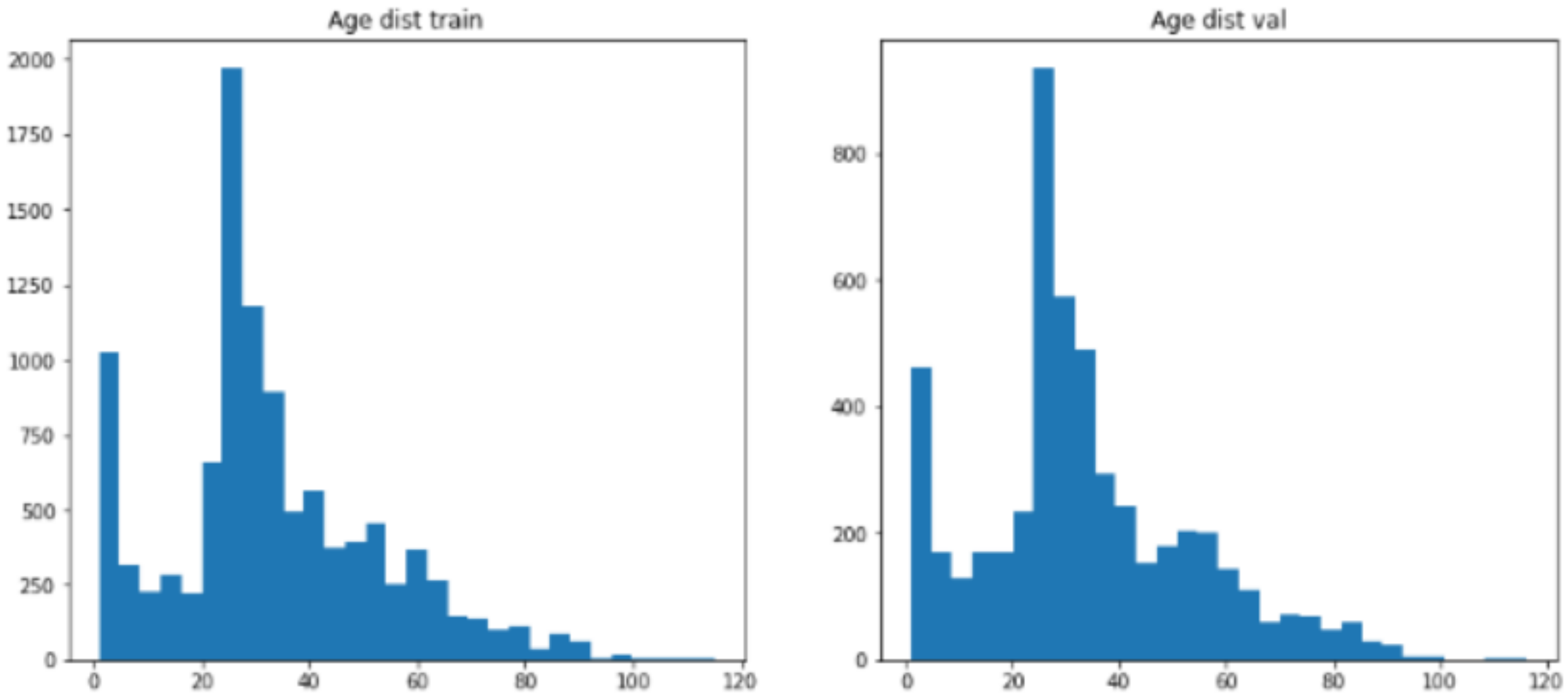
Exercise: bis um 16:00



Check out the models for age prediction with flexible and constant variance and try to understand the code and to answer the questions.

https://github.com/tensorchiefs/dl_course_2022/blob/master/notebooks/17_faces_regression.ipynb

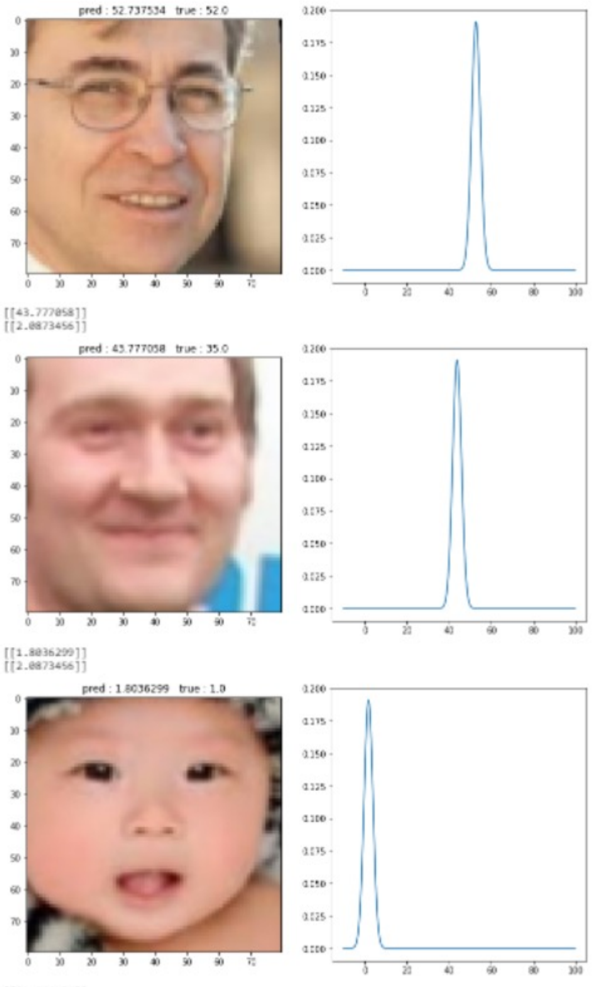
Age distribution



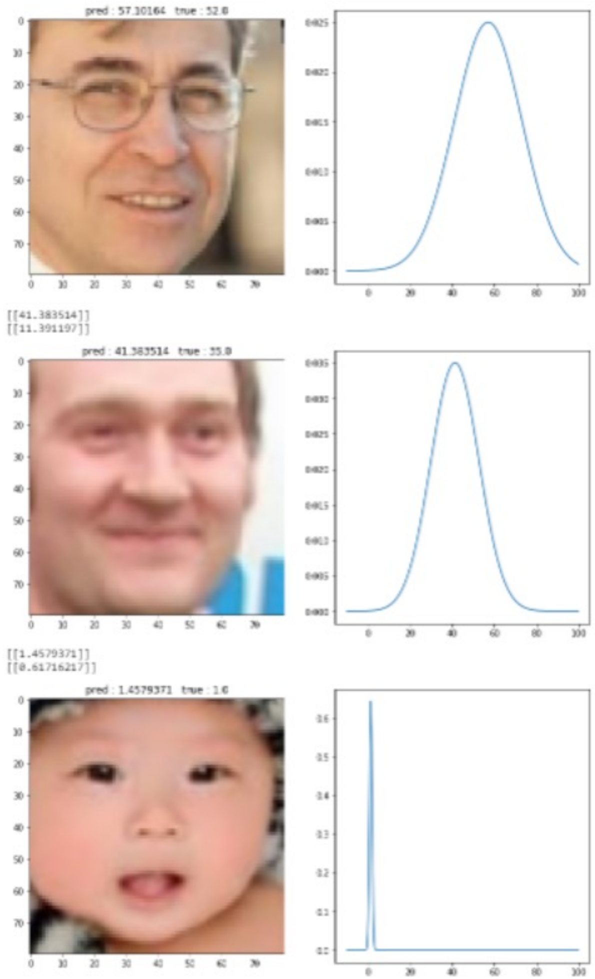
We have a lot of small children in the data set, for whom the age estimation is probably not so difficult.

Resulting age CPDs

Constant variance



flexible variance



In case of a flexible variance, a broad predicted Gaussian CPD does indicate high uncertainty about the age.

Summary

- A probabilistic model predicts for each input a whole conditional probability distribution (CPD).
- The predicted CPD assigns for each possible outcome *y*, a probability with which it's expected.
- The negative log-likelihood (NLL) measures how well the CPD matches the actual distribution of the outcomes (lower is better).
- The NLL is used as a loss function when training a probabilistic model.
- The NLL on new data is used to measure, and to compare, the prediction performance of different probabilistic models.
- Using a proper choice for the CPD enhances the performance of your models.
- For continuous data, a common first choice is a Normal distribution
- For count data, common choices for distribution are Poisson, Negative-Binomial, or Zero-inflated Poisson (ZIP).

LLM are also just probabilistic models

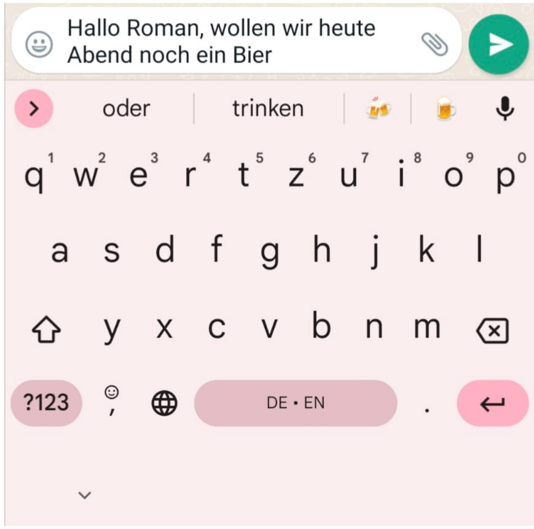
Principles of ChatGPT

Describe your technology in one sentence.

I am a Generative Language model based on the transformer architecture that uses self-attention mechanisms and is trained with maximum likelihood principle to predict the next token in a sequence of tokens and generate new text.

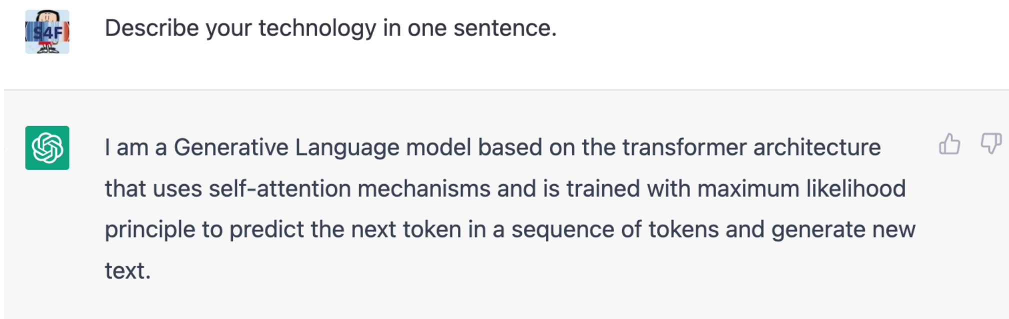
Token = Part(s) of Word

Generative Language Models: “predict next token“



Generative Language models:

Like you pressing (most) probable suggestion

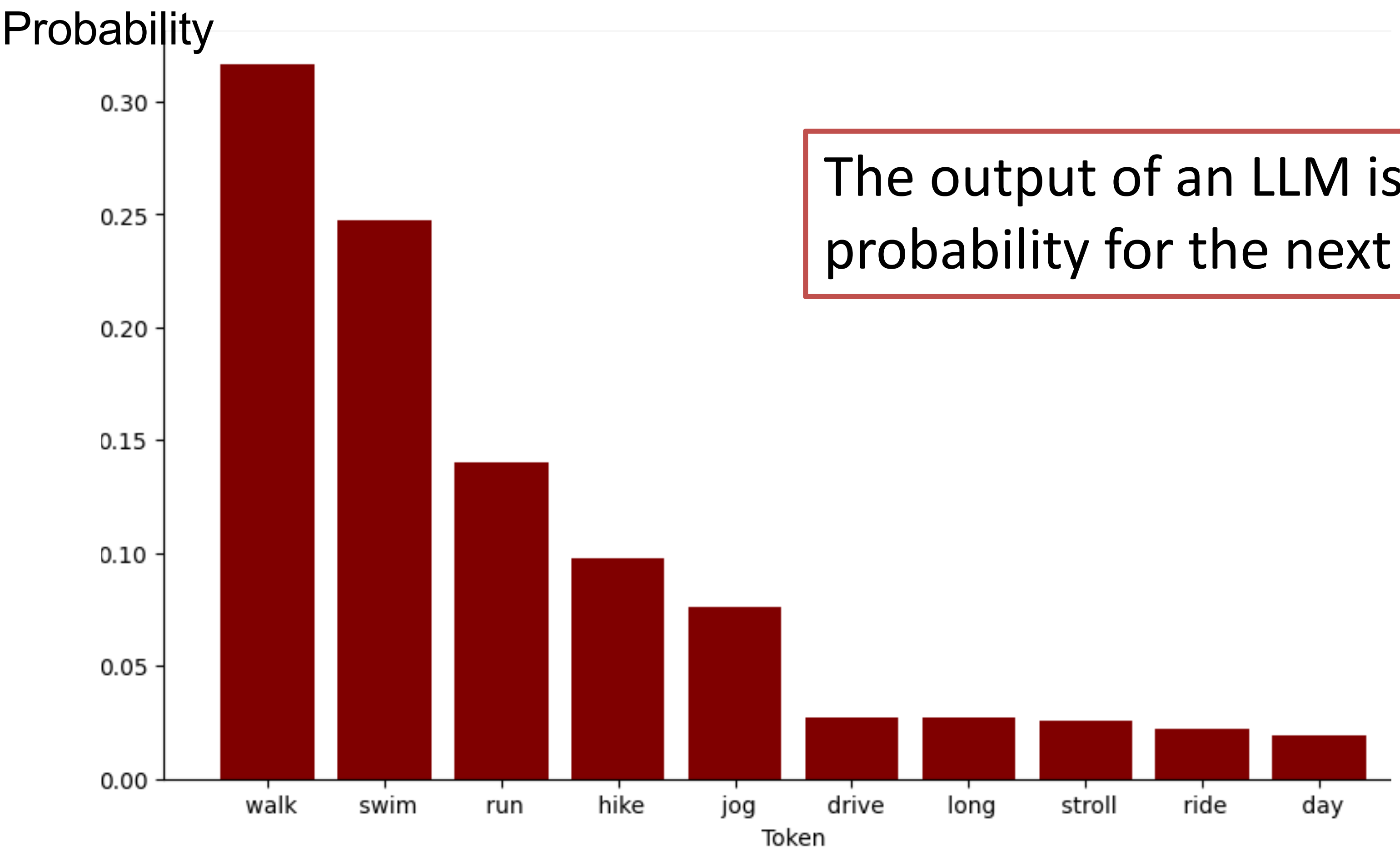


Steps 1: Describe your technology in one sentence. → I
Steps 2: Describe your technology in one sentence. I → am
Steps 3: Describe your technology in one sentence. I am → a
Steps 4: Describe your technology in one sentence. I am a → generative

Step 36 Describe your technology in one sentence. I am a … new text. → END

Probabilities for next word

Prompt
The weather is really nice today. I'm thinking about going for a

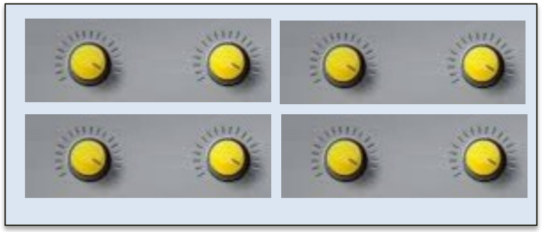


The output of an LLM is the probability for the next word

https://github.com/oduerr/llm_playground/tree/main/hugging_face/sampling_from_LLM

LLM are probabilistic models

LLM

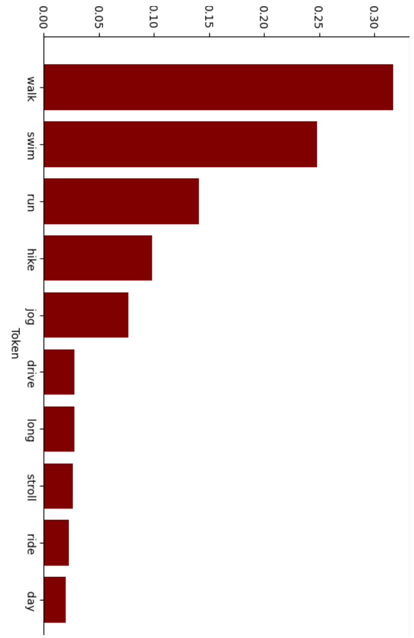


The weather is really nice today. I'm thinking about going for a

$\boldsymbol{x}$

A model with parameters $\boldsymbol{\theta}$

Probability for y



y

Quiz: Number of parameters in GTP-3.5?

Quiz: Write this in math

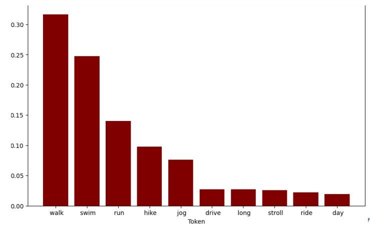
$$p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$$

GTP (Generative Pretrained Transformer) and Llama are causal/autoregressive

Other LLMs (BERT) might depend on “future”.

Training causal models: To predict the next token

- Training data:
  - “Whole Internet”: arXiv + StackOverflow + …
    - LLaMMa*: 1T (1E12) Token
- Take samples
  - Take a text example where to know the answer
    - The weather is really nice today. I'm thinking about going for a walk
  - Use input x=“The weather is really nice today. I'm thinking about going for a”
  - Observed value y=“walk”
  - Output of model $p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$



  - Tune the model so that $p_{\boldsymbol{\theta}}$(“walk”$|\boldsymbol{x}$) is high
- Models like BERT need different training

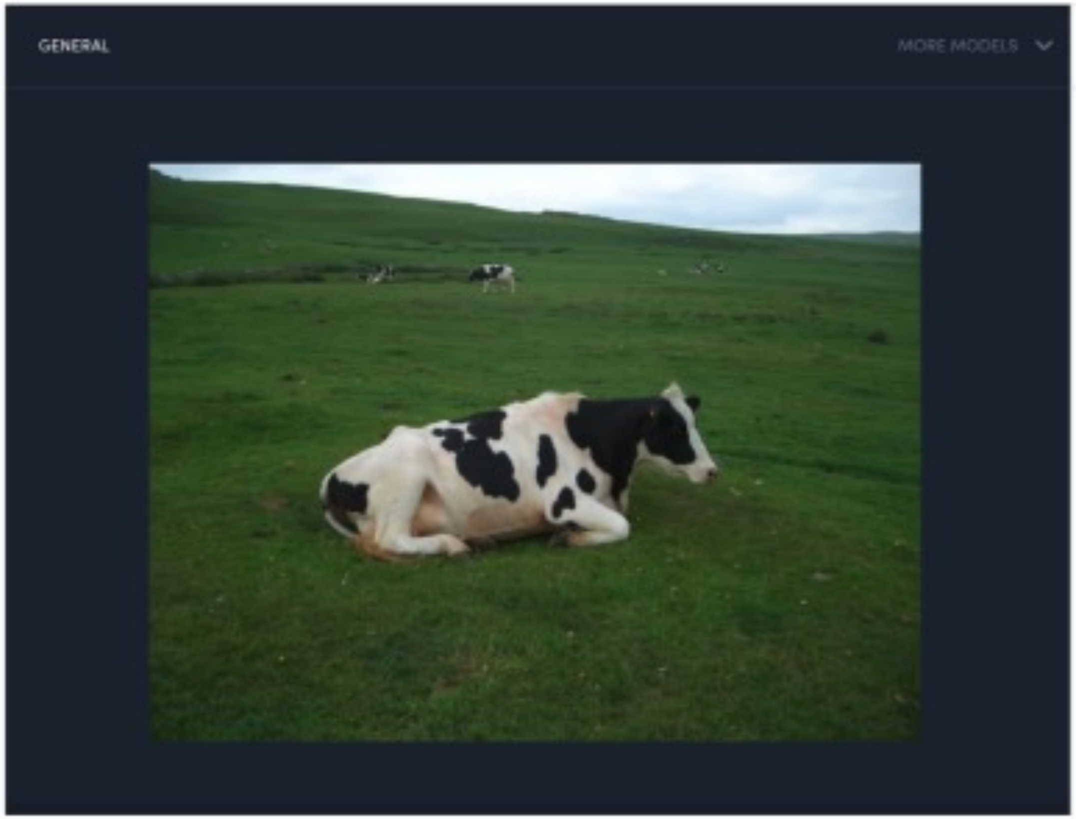
*LLaMA arXiv:2302.13971

Bayesian Deep Learning

Outline

- Issues with current DL approach
  - No extrapolating / no epistemic uncertainty
- Bayesian Statistics
  - A simple example (Bayes the Hacker's way)
  - Introduction to Bayesian Statistics
- Bayesian Neural Networks
  - Variational Approximation
  - MC-Dropout

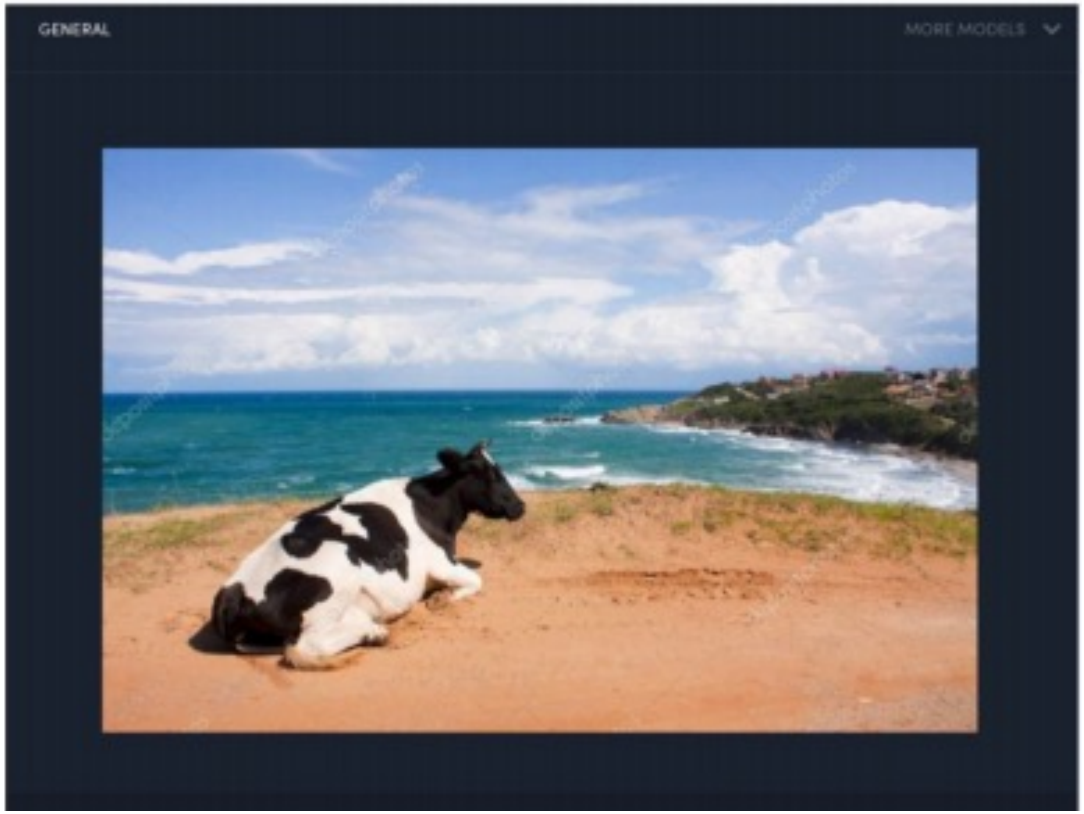
Typisches Beispiel einer Kuh



General	VIEW DOCS
cow	0.992
cattle	0.983
mammal	0.979
grass	0.978
livestock	0.966
farm	0.964
landscape	0.963
pasture	0.954
grassland	0.949
agriculture	0.948
no person	0.945

Untypisches Beispiel einer Kuh

Dieses Bild ist zu weit weg von den Trainingsdaten. Kuh nicht unabhängig von Hintergrund



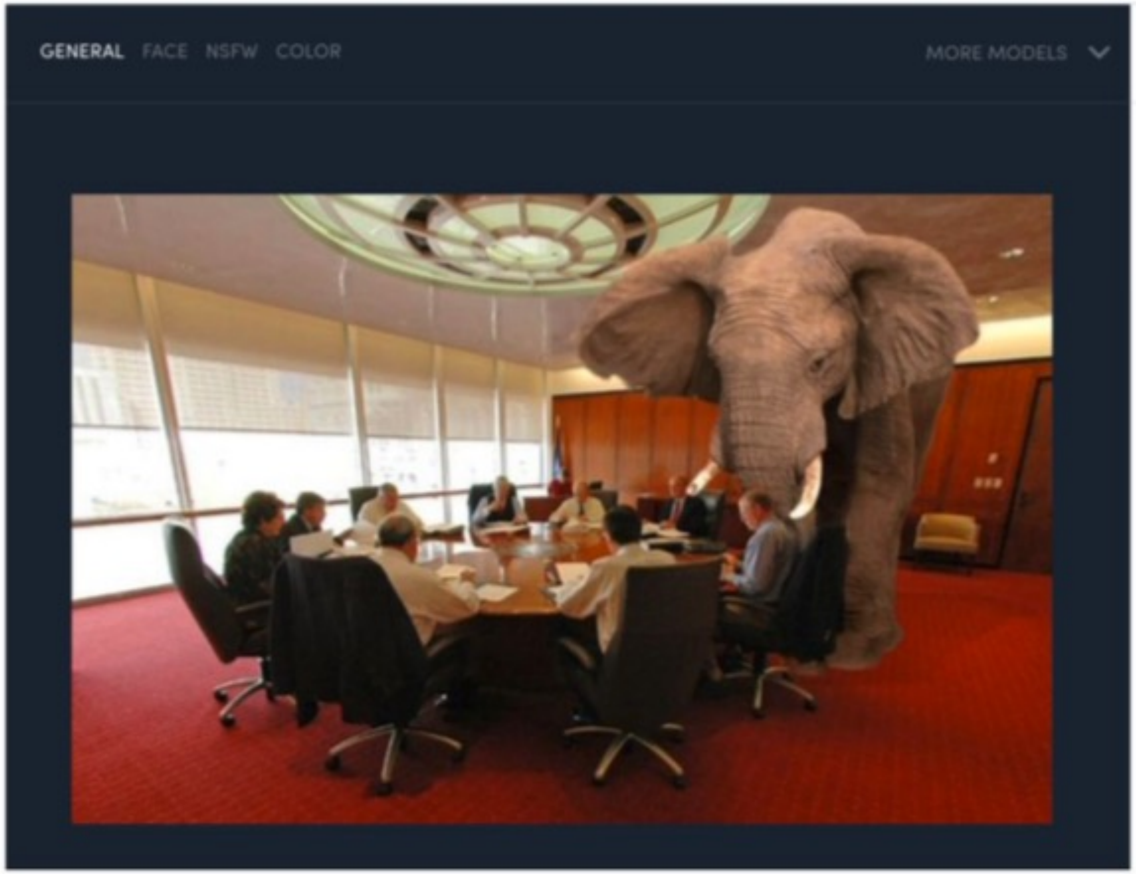
Neuronales Netz erkennt keine Kuh!

General	VIEW DOCS
no person	0.991
beach	0.990
water	0.985
sand	0.981
sea	0.980
travel	0.978
seashore	0.972
summer	0.954
sky	0.946
outdoors	0.944
ocean	0.936

[NeurIPS 2018 Workshop on Causal Learning]

The elephant in the room

Neuronales Netz erkennt
keinen Elefanten!

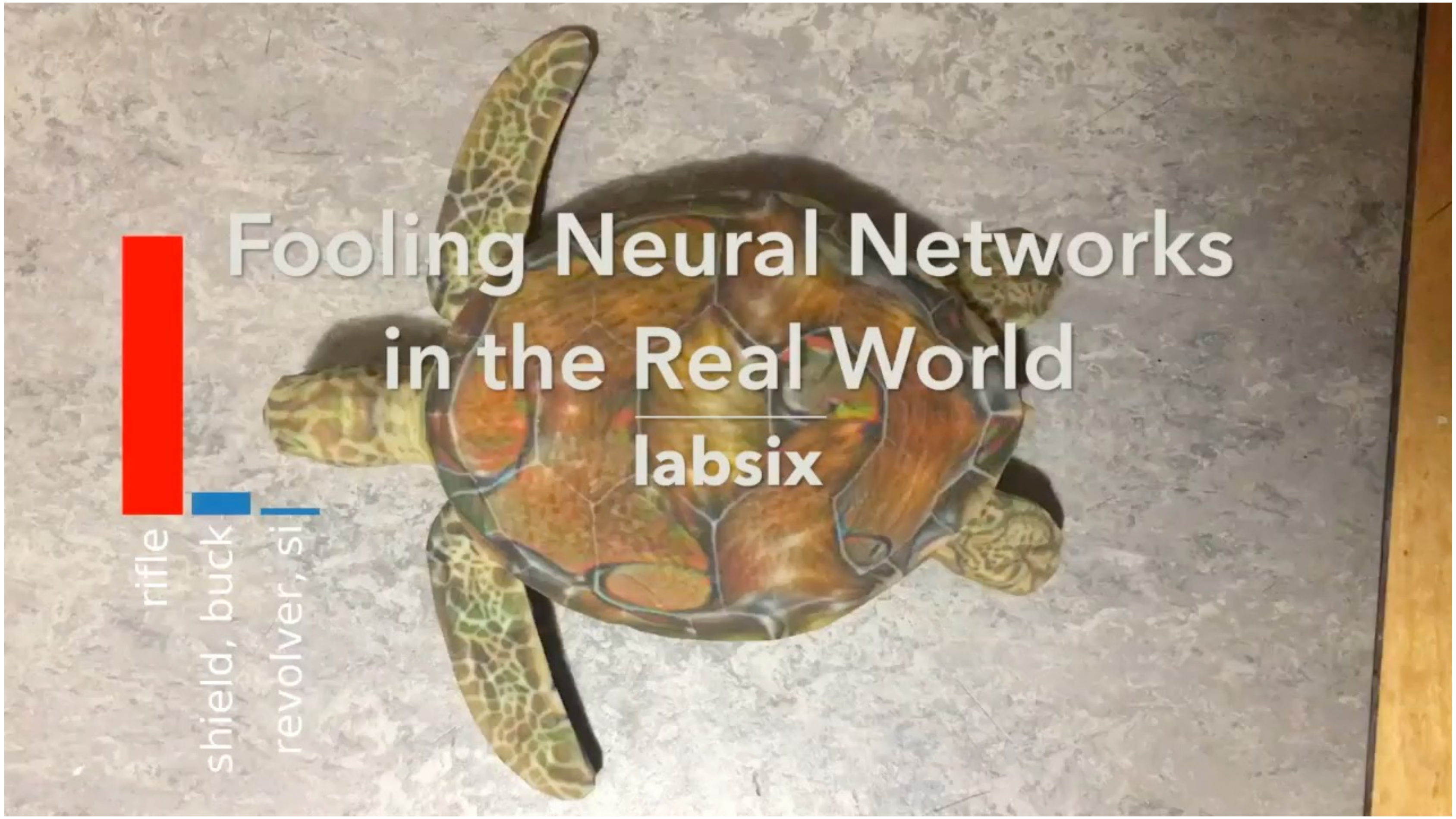


General VIEW DOCS

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]

Turtle as Rifle

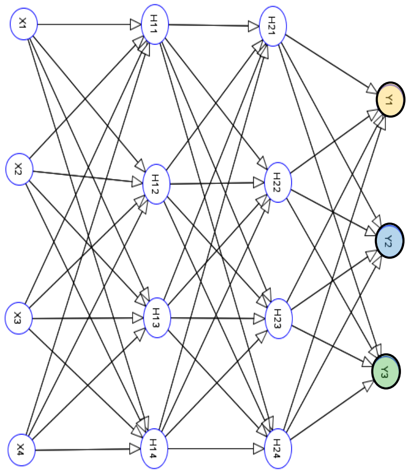


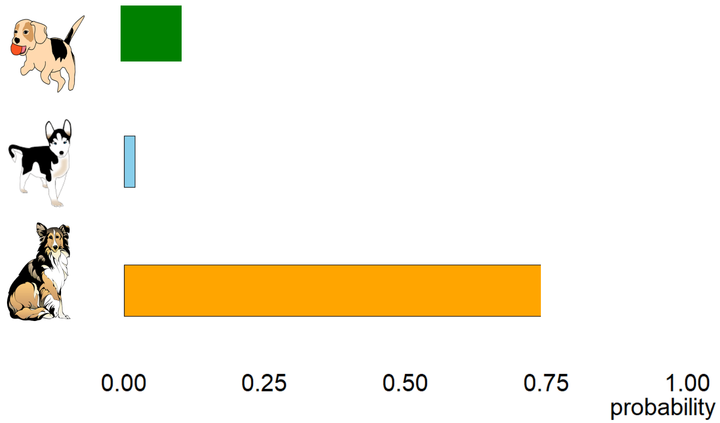
Ausgedruckes Exemplar

Source: https://www.labsix.org/physical-objects-that-fool-neural-nets/

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_{pred}=0.75

What is 0.75 telling us.

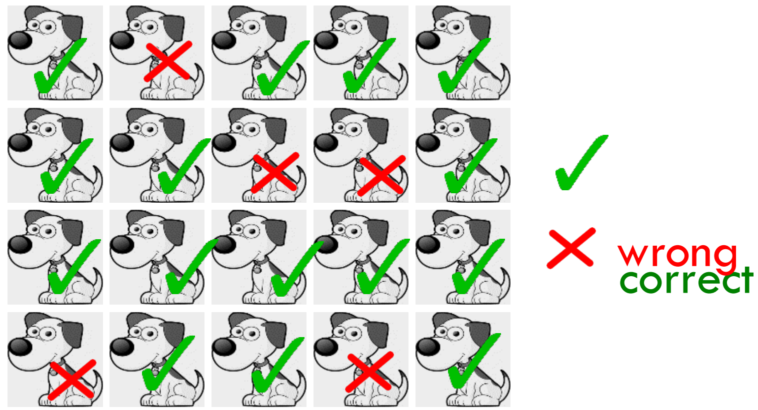
What is the probability telling?

p_{pred}=0.75

Among many predications that had p_{pred}=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*

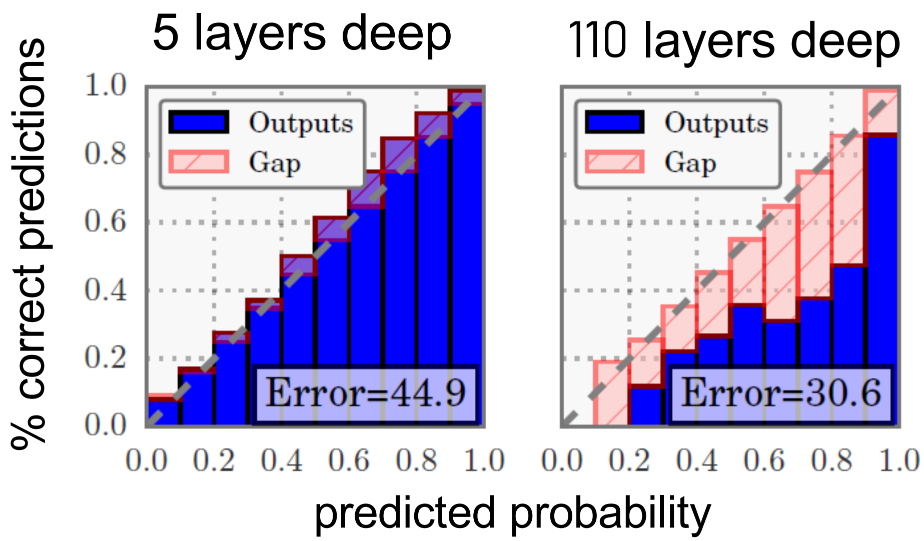
Sample of images where the predicted class got p=0.75:



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

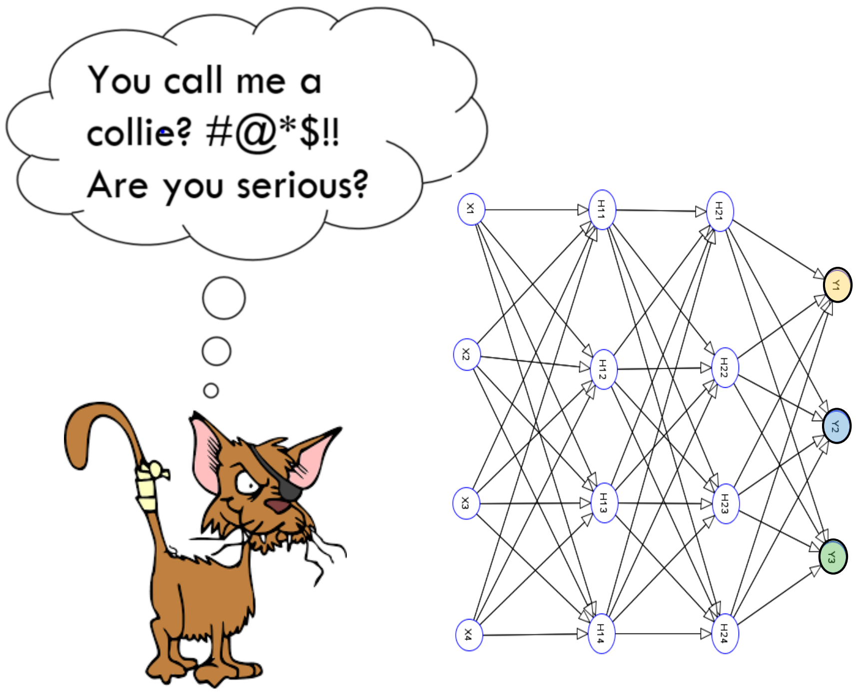


Good news:
deep NN can be “recalibrated” and then we get calibrated probabilities.

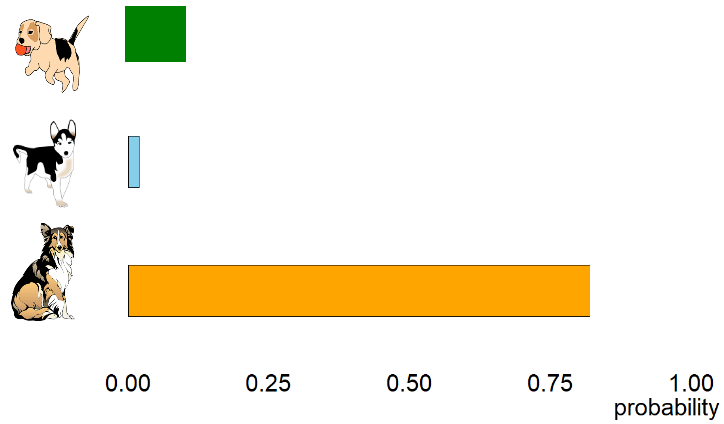
46

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

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



Plain wrong !



The reported probability is only valid if, we have the P(Train)=P(Test).
That's not the case. "The big lie deep learning is based on".
This is more evident, if we have a look at regression problems.

48

Elephant in the room

- Aufgabe
  https://github.com/tensorchiefs/dl_book/blob/master/chapter_07/nb_ch07_01.ipynb