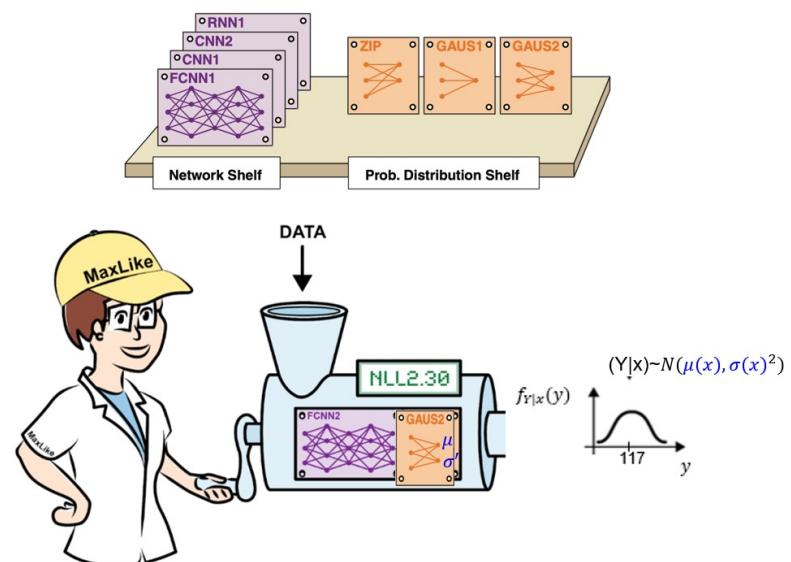


WBL Probabilistic Deep Learning:: Day 3

Slight changes possible before lecture

Beate Sick, Oliver Dürr

Day 3: Model evaluation and transformer NNs

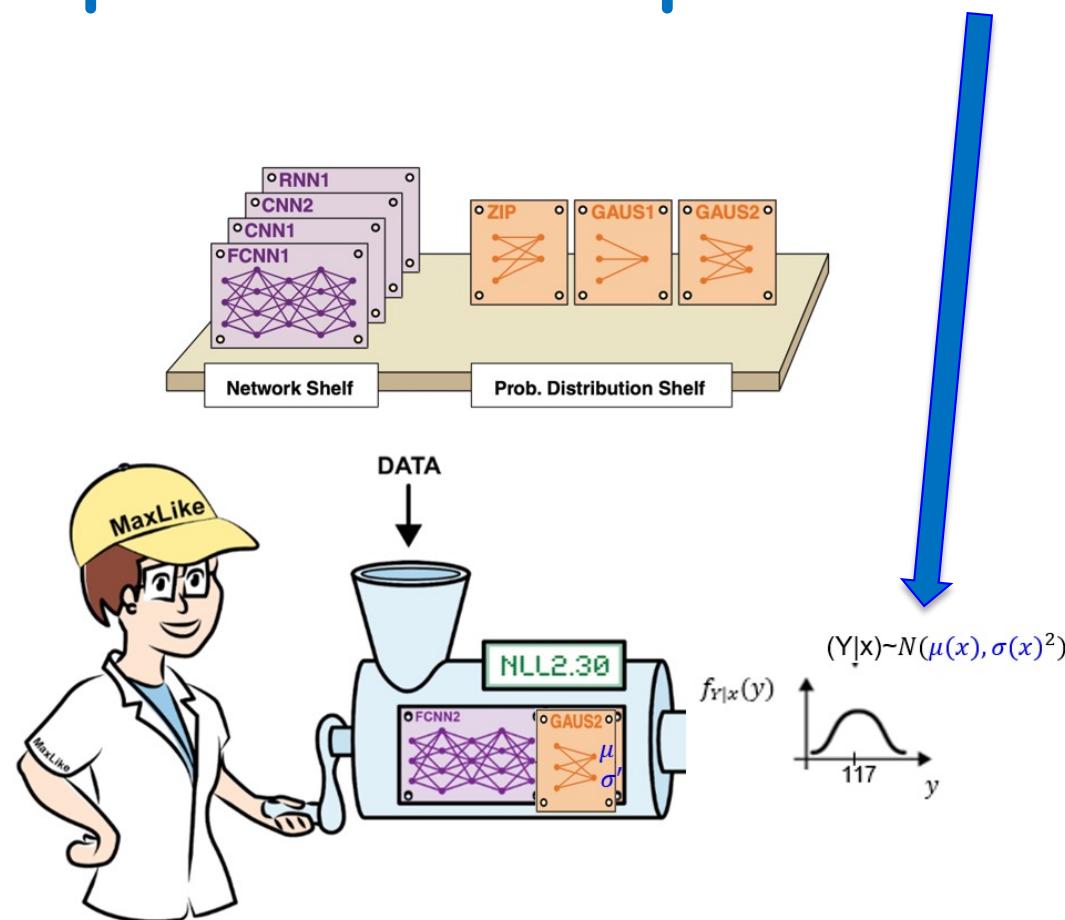


Topics (may change)

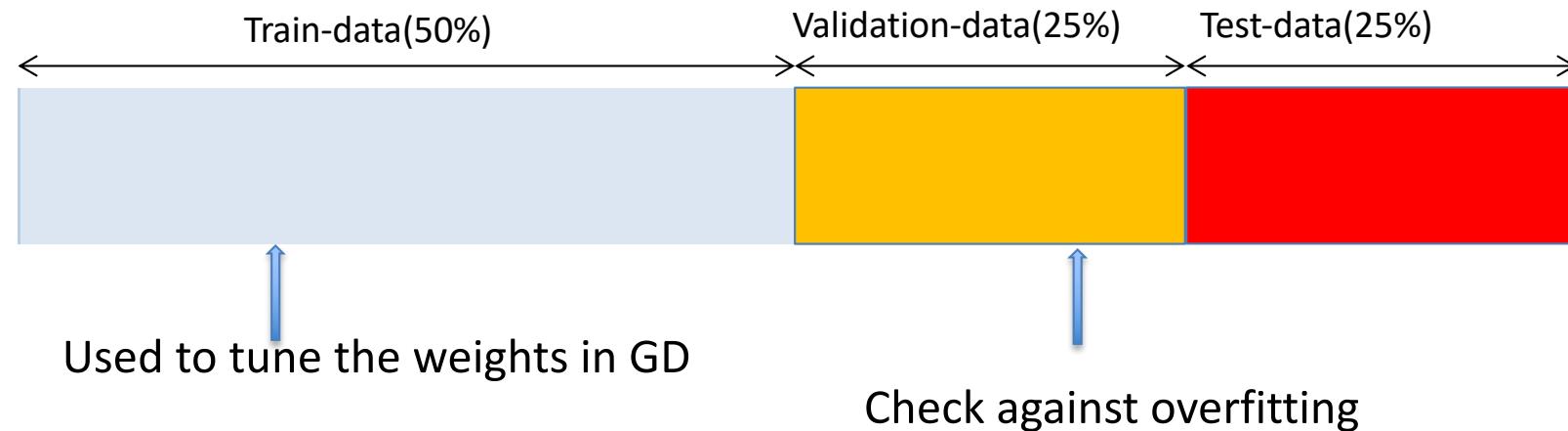
- Half day 1
 - Introduction to probabilistic DL
 - Introduction to Keras with pytorch backend
- Day 2
 - Loss function and optimization
 - Convolutional Neural Networks (CNN) for image data
 - Working with pretrained models
 - Transfer learning
 - Projects
- Day 3
 - Model evaluation
 - xAI: understand where the NN looks at
 - Transformer Architectures
 - Projects
- Half day 4
 - Project presentations
 - tba

Date
03.02.2025
10.02.2025
17.02.2025
24.02.2025

Evaluating the quality of the probabilistic predictions



Model evaluation is done on Validation and Test Set



Best practice: Lock an extra **test data set** away, and use it only at the very end, to evaluate the chosen model, that performed best on your validation set.

Reason: **When trying many models, you probably overfit on the validation set.**

Determine performance metrics, such as NLL, Brier score, AUC or MSE, to evaluate the predictions **on new validation or test data**

Evaluation of probabilistic models

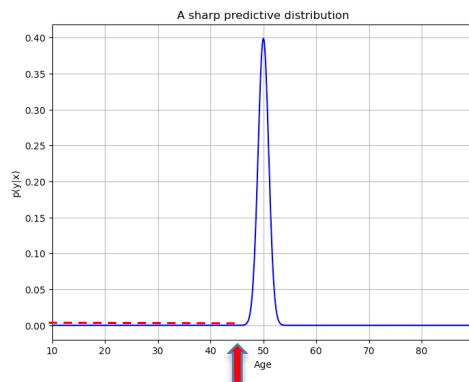
Which model is better / evaluating probabilistic models

Meryl Streep, 41y



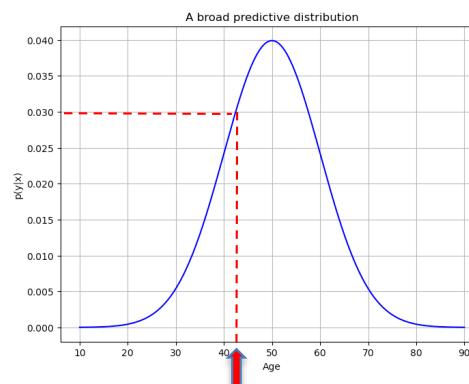
Model 1
Model 2

Predicted age distribution



$$p(y|x)$$

$$L_{\text{streep}} = 0.0001$$



$$p(y|x)$$

$$L_{\text{streep}} = 0.03$$



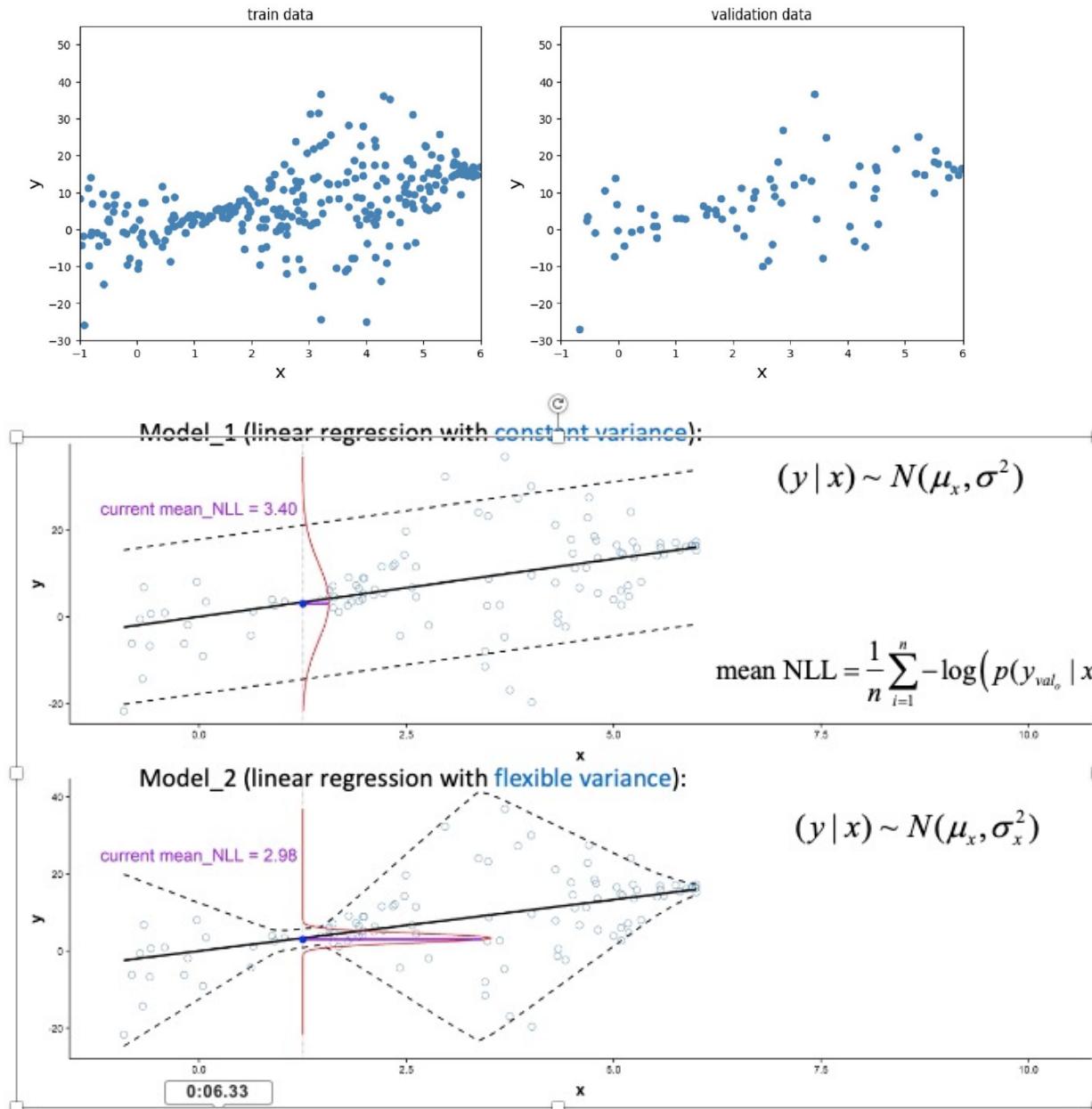
Which model is better?

For the evaluation, we don't care if $p(y|x)$ is from a NN or from another model. Evaluation of $p(y|x)$ has long tradition, e.g. in metrology (Good 1952).

Is the test Likelihood or NLL suited for evaluation?

- The best measures for evaluation of a probabilistic model:
 - test **NLL** (negative-log-likelihood) for continuous and discrete outcomes
 - test log-likelihood (aka log-score) for continuous and discrete outcomes
 - ... some few other “proper” scores you will see later
- Why?
 - **the objective in the training was to minimize the NLL**
→ the better the NLL the better the model!!
Otherwise we should have optimized another measure ;)
 - The NLL (or log-score) is “**strictly proper**” and the only local score which gets optimal if the predicted distribution matches the data generating distribution.
- Why not?
 - People like to see accuracy, AUC, MSE... → provide these measures additionally even though they are not “proper” and the model was not optimized for this purpose.

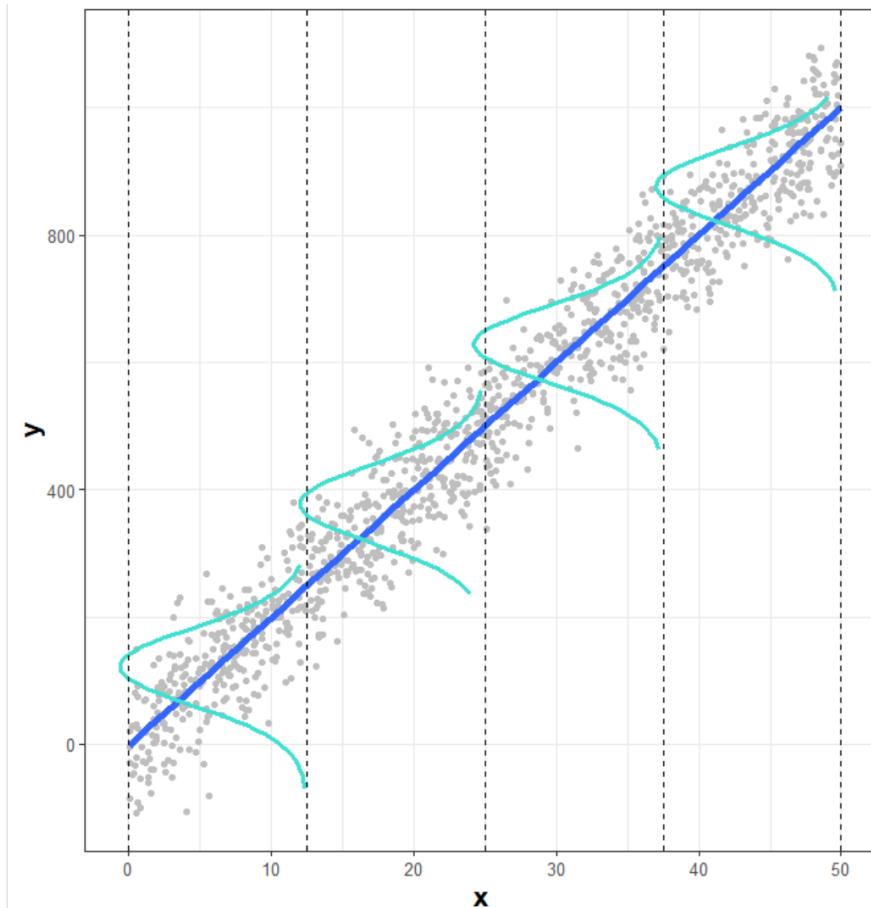
Which of the two probabilistic models is better?



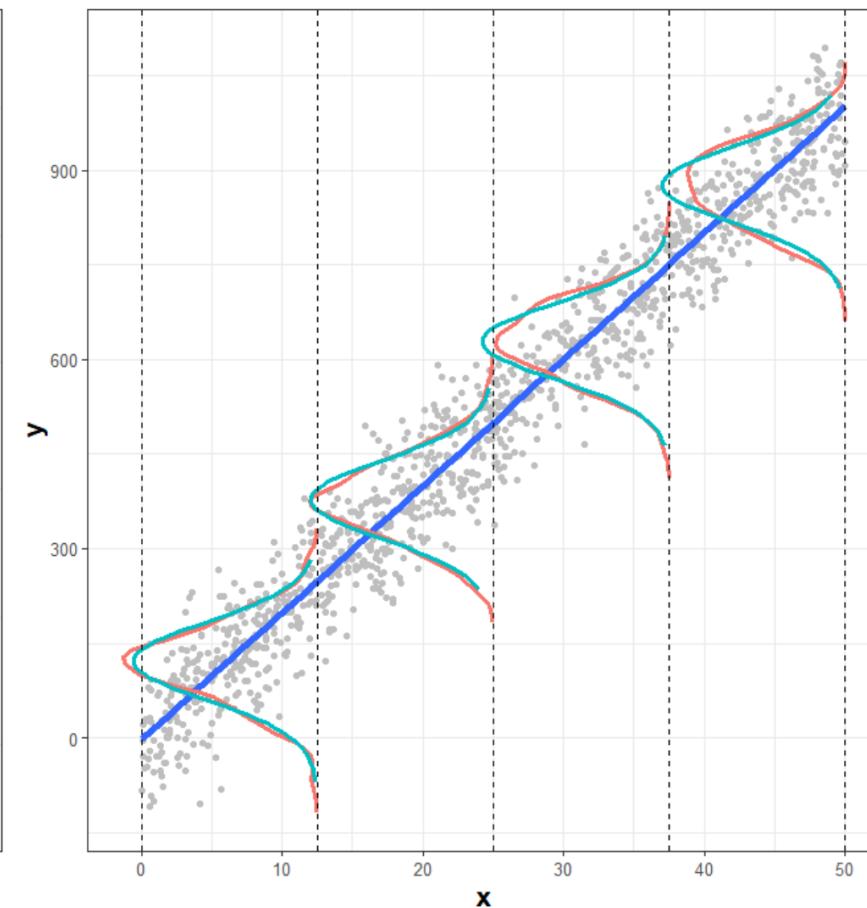
https://www.youtube.com/watch?v=QeWd0g_UzZY

Visually: Do predicted and observed outcome distribution match?

Validation data along with predicted outcome distribution (Gauss with const σ)



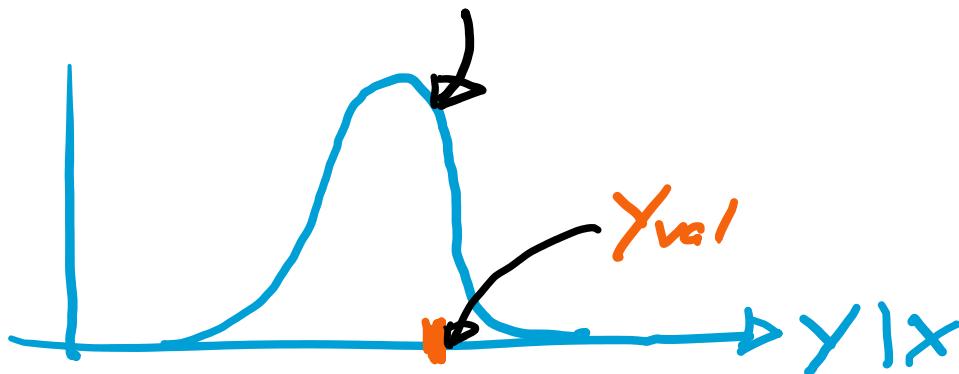
Validation data along with predicted and observed outcome distribution



A large validation data set is needed to ensure underlying assumption:
observed distribution = data generating distribution

What is a good “score” to evaluate a probabilistic model?

- For evaluation we have
 - Values $(x_{\text{val}}, y_{\text{val}})$ from DGP (or validation set)
 - Predicted distribution $p_{\text{pred}}(Y|x_{\text{val}})$

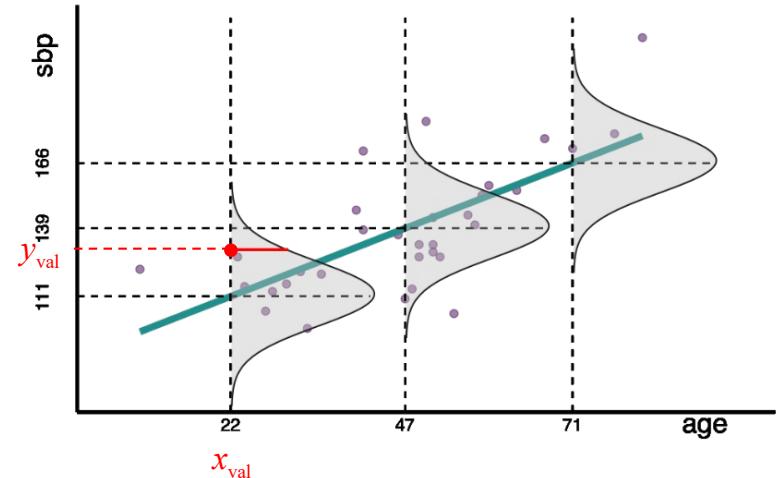


In the following:

1. We create a “score” which take as input $p_{\text{pred}}(Y|x)$ for a single pair $(x_{\text{val}}, y_{\text{val}})$
2. Evaluate if the score gets best if the predicted distribution p_{pred} matches the distribution of the data generating process (DGP) by using $(x_{\text{val}}, y_{\text{val}}) \sim DGP$

Scores to evaluate probabilistic prediction models

- We need validation data: $(x_{\text{val}}, y_{\text{val}})$
- We need predicted outcome distribution, given x : $p_{\text{pred}}(Y|x)$
- The score S takes *one instance* and yields a real number (smaller is better)
$$S(p_{\text{pred}}(Y|x_{\text{val}}), y_{\text{val}})$$



Example 1: NLL (aka log-score, ignorance):

$$S_{NLL}(p_{\text{pred}}(Y|x_{\text{val}}), y_{\text{val}}) = -\log(p_{\text{pred}}(Y|x_{\text{val}}))$$

Example 2: weighted MSE:

$$S_{MSE}(p_{\text{pred}}(Y|x_{\text{val}}), y_{\text{val}}) = \int (y_{\text{val}} - y)^2 p_{\text{pred}}(Y|x_{\text{val}}) dy$$

Empirical loss as average score

- If we use a validation set with n instances (x_{val_i}, y_{val_i}) to evaluate the model, the average score is used as empirical loss:

$$\text{empirical loss} = \frac{1}{n} \sum_{i=1}^n S(p_{\text{pred}}(y | x_{val_i}), y_{val_i})$$

- The empirical loss approximates the expected loss:

$$\text{expected loss} = \int_y S(p_{\text{pred}}(y | x'), y') \cdot p_{\text{true}}(y', x') \, dx' dy'$$

p_{pred} : predicted distribution

p_{true} : data generating distribution

Local scores

A score is local if the predicted distribution is evaluated only at the actual observed outcome of the validation data

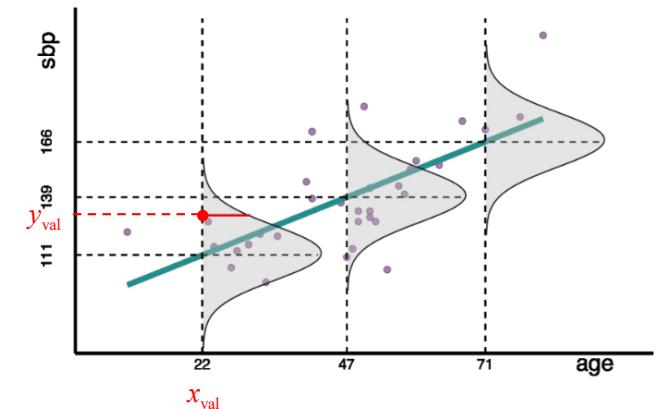
$$S(p_{\text{pred}}(Y|x_{\text{val}}), y_{\text{val}}) = S(p_{\text{pred}}(y_{\text{val}}|x_{\text{val}}), y_{\text{val}})$$

Example 1: NLL (aka log-score, ignorance):

$$S_{NLL}(p_{\text{pred}}(Y|x_{\text{val}}), y_{\text{val}}) = -\log(p_{\text{pred}}(y_{\text{val}}|x_{\text{val}}))$$

Example 2: linear score

$$S_{NL}(p_{\text{pred}}(Y|x_{\text{val}}), y_{\text{val}}) = -p_{\text{pred}}(y_{\text{val}}|x_{\text{val}})$$



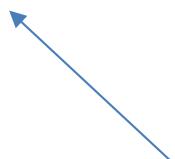
Proper Scores

For a proper score holds:

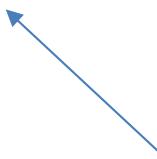
The expected value of a *proper score* takes its minimal (optimal) value, if predicted distribution $p_{\text{pred}} = p_{\text{true}}$ data generating distribution

The expected value of a *strictly proper score* takes its minimal value, *only* if predicted distribution $p_{\text{pred}} = p_{\text{true}}$ data generating distribution

$$\int \int S(p_{\text{true}}(y|x'), y') p_{\text{true}}(y', x') dy' dx' < \int \int S(p_{\text{pred}}(y|x'), y') p_{\text{true}}(y', x') dy' dx' \text{ if } p_{\text{pred}} \neq p_{\text{true}}$$



The score with true conditional distribution



The score with predicted cond. distribution

The log-score is strictly proper - proof for nerds

To show: $\int \int S(p_{pred}(y|x'), y') p_{true}(y', x') dy' dx' > \int \int S(p_{true}(y|x'), y') p_{true}(y', x') dy' dx'$

$$\begin{aligned} \int_{x,y} S(p_{pred}(y|x'), y') \cdot p_{true}(y', x') \, dx' dy' &= \int_{x,y} S(p_{true}(y|x'), y') \cdot p_{true}(y', x') \, dx' dy' + \\ &\quad \underbrace{\left\{ \int_{x,y} S(p_{pred}(y|x'), y') \cdot p_{true}(y', x') \, dx' dy' - \int_{x,y} S(p_{true}(y|x'), y') \cdot p_{true}(y', x') \, dx' dy' \right\}}_{> 0 \text{ for strictly proper scores } S} \end{aligned}$$

Proof that NLL is strictly proper $S_{NLL}(p(y|x_{val}), y_{val}) = -\log(p(y_{val}|x_{val}))$

$$\begin{aligned} &\int S_{NLL}(p_{pred}(y|x'), y') \cdot p_{true}(y', x') \, dx' dy' - \int S_{NLL}(p_{true}(y|x'), y') \cdot p_{true}(y', x') \, dx' dy' \\ &= \int -\log(p_{pred}(y'|x')) \cdot p_{true}(y', x') \, dx' dy' - \int -\log(p_{true}(y'|x')) \cdot p_{true}(y', x') \, dx' dy' \\ &= \int \log\left(\frac{p_{true}(y'|x')}{p_{pred}(y'|x')}\right) \cdot p_{true}(y'|x') p_{true}(x') \, dx' = \text{KL}(p_{true}(\cdot|x); p_{pred}(\cdot|x)) > 0 \quad \forall p_{pred} \neq p_{true} \end{aligned}$$

The linear score is not proper - proof for nerds

The linear score is not proper, meaning p_{true} does not yield the best expected score.

$$S_{\text{lin}}(p(y|x_{val}), y_{val}) = -p(y_{val} | x_{val})$$

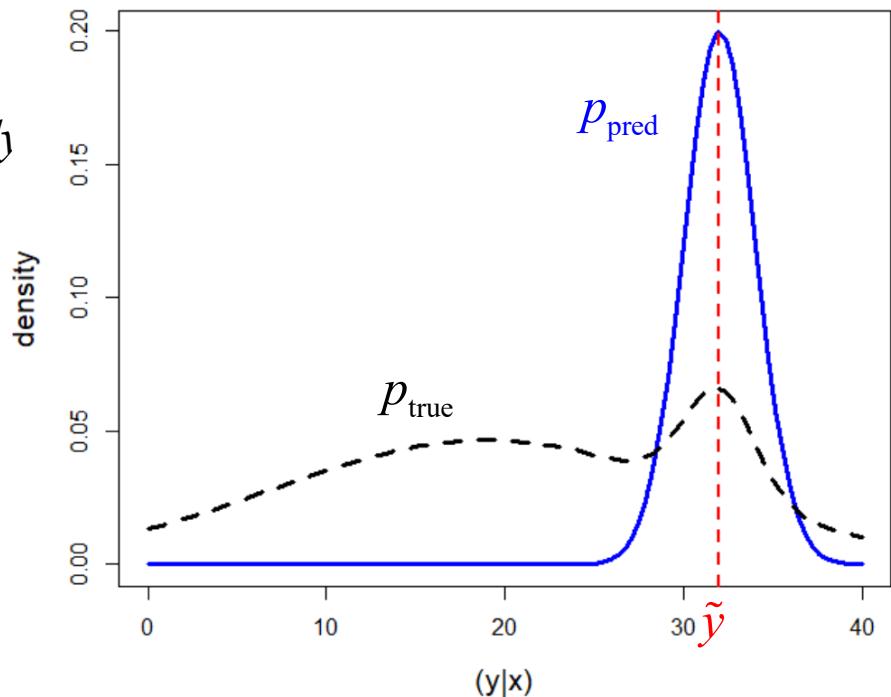
$$\begin{aligned} E_{p_{true}}(S_{p_{\text{pred}}}) &= \int_{x,y} S(p_{true}(y|x'), y') \cdot p_{true}(y',x') \, dx' dy' \\ &= \int_{x,y} -p_{true}(y'|x') \cdot p_{true}(y',x') \, dx' dy' \end{aligned}$$

If p_{true} is not constant, then there is a \tilde{y} higher than mean probability:

$$-p_{true}(\tilde{y}|x') < E_{p_{true}}(S_{p_{\text{true}}})$$

Proof:

Construct p_{pred} that scores better than p_{true} :



$$p_{\text{pred}}(\tilde{y}|x') = \frac{1}{\sigma} \cdot \text{kernel}\left(\frac{y' - \tilde{y}}{\sigma}\right)$$

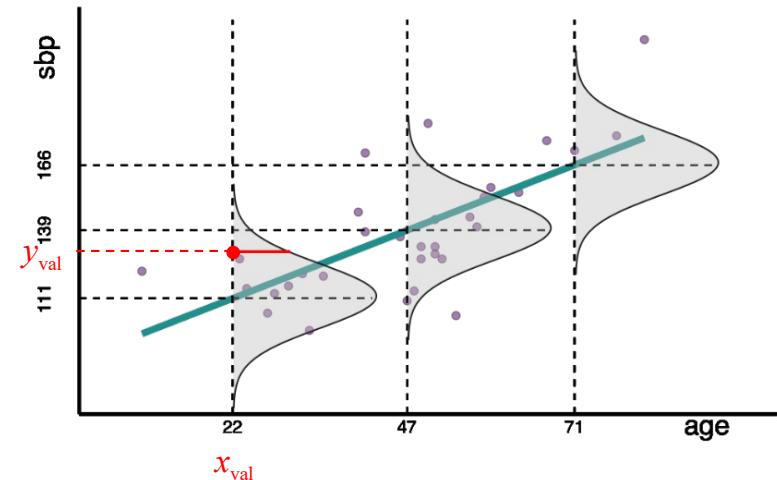
$$E_{p_{true}}(S_{p_{\text{pred}}}) = \int_{x,y} -p_{\text{pred}}(y'|x') \cdot p_{true}(y',x') \, dx' dy' \rightarrow -p_{true}(\tilde{y}|x') < E_{p_{true}}(S_{p_{\text{true}}})$$

The uniqueness of the log-score

It is provable that **the log-score is the only smooth, proper and local score for continuous variables**

(Bernardo, J. M., 1979: Expected information as expected utility. *Ann. Stat.*, 7, 686–690)

$$S_{\text{NLL}}(p(y|x_{\text{val}}), y_{\text{val}}) = -\log(p(y_{\text{val}}|x_{\text{val}}))$$



Scoring Probabilistic Forecasts: The Importance of Being Proper

JOCHEN BRÖCKER

Centre for the Analysis of Time Series, London School of Economics, London, United Kingdom

LEONARD A. SMITH

Centre for the Analysis of Time Series, London School of Economics, London, and Pembroke College, Oxford University, Oxford, United Kingdom

(Manuscript received 4 November 2005, in final form 23 May 2006)

ABSTRACT

Questions remain regarding how the skill of operational probabilistic forecasts is most usefully evaluated or compared, even though probability forecasts have been a long-standing aim in meteorological forecasting. This paper explains the importance of employing proper scores when selecting between the various measures of forecast skill. It is demonstrated that only proper scores provide internally consistent evaluations of probability forecasts, justifying the focus on proper scores independent of any attempt to influence the behavior of a forecaster. Another property of scores (i.e., locality) is discussed. Several scores are examined in this light. There is, effectively, only one proper, local score for probability forecasts of a continuous variable. It is also noted that operational needs of weather forecasts suggest that the current concept of a score may be too narrow; a possible generalization is motivated and discussed in the context of propriety and locality.

<https://journals.ametsoc.org/doi/full/10.1175/WAF966.1>

Prominent Scores for binary classifiers

Definition 9.9 (Scoring rules for binary predictions) Let $Y \sim B(\pi)$ be the predictive distribution for a binary event, i.e.

$$f(y) = \begin{cases} \pi & \text{for } y = 1, \\ 1 - \pi & \text{for } y = 0. \end{cases}$$

The *Brier score* BS, the *absolute score* AS and the *logarithmic score* LS are defined as

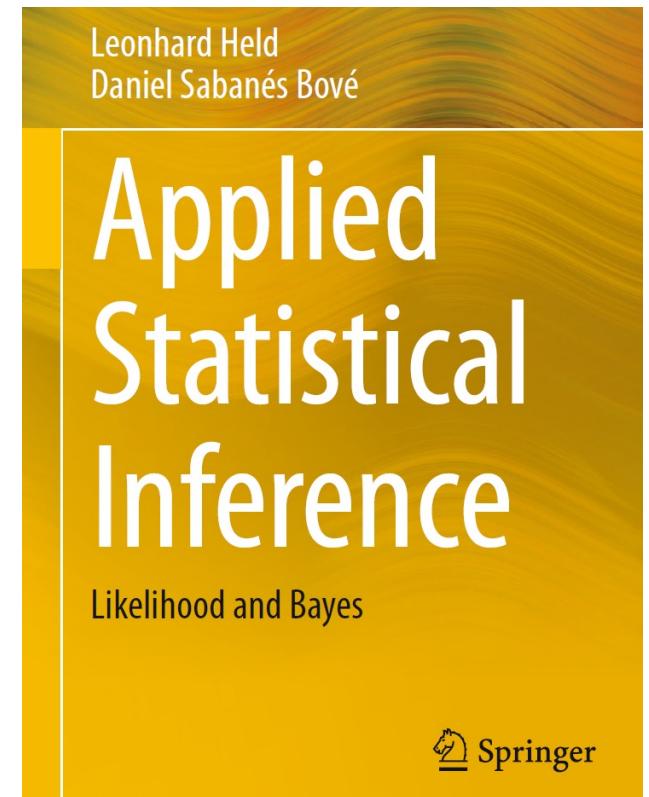
Strictly proper: $BS(f(y), y_0) = (y_0 - \pi)^2$,

Not proper: $AS(f(y), y_0) = |y_0 - \pi|$ and

Strictly proper: $LS(f(y), y_0) = -\log f(y_0)$,

respectively.

Remark: For binary classification, the log score is not the only strictly proper score.

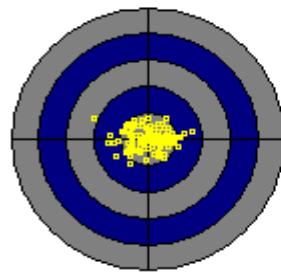


Non-proper scoring rules

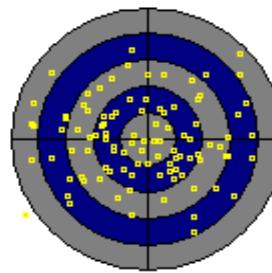
The following performance measures are very popular but **not proper**:

- Accuracy – for categorical outcome
- Kappa – for categorical outcome
- Area under the curve (AUC) - for binary outcome
- Mean Square Error (MSE) – for continuous outcome
- Mean Absolut Error (MAE) – for continuous outcome
- ...

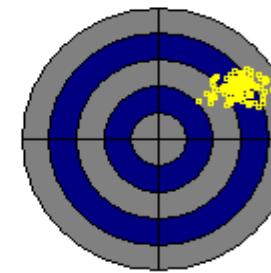
Calibration & Sharpness



calibrated
&
sharp



calibrated
&
not sharp



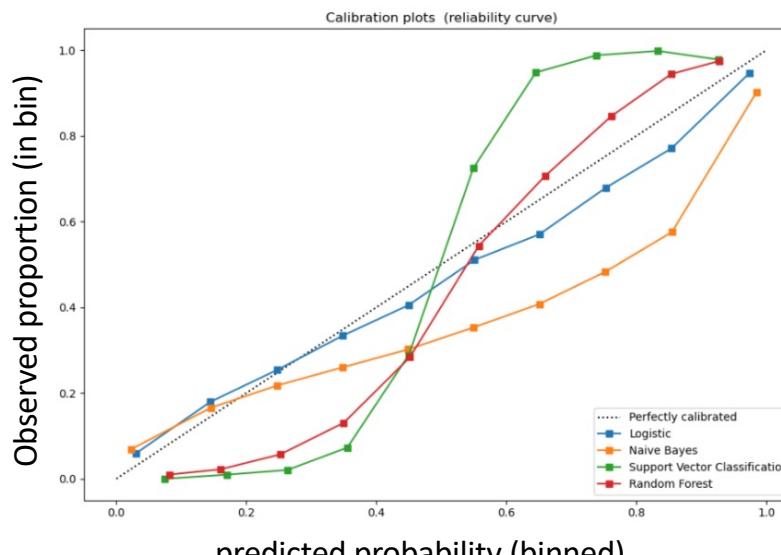
not calibrated
&
sharp

Calibration

- Consider the classification of tomorrow's weather
 - Y : Weather $\in \{\text{Sunny, Mixed, Rainy}\}$
- Example
 - Classifier $p_{\text{pred}}(Y = \text{sunny}) = 0.7$
 - In exactly 70% of those cases, it should be sunny
 - Check: Take all days with $p_{\text{pred}}(Y = \text{sunny}) = 0.7$ and check if on 70% of these days the weather was sunny. Do this for each probability bin.
- Calibration plot



$$\frac{\#(y_{\text{obs}} = \text{"rain"})}{\#y_{\text{obs}}}$$



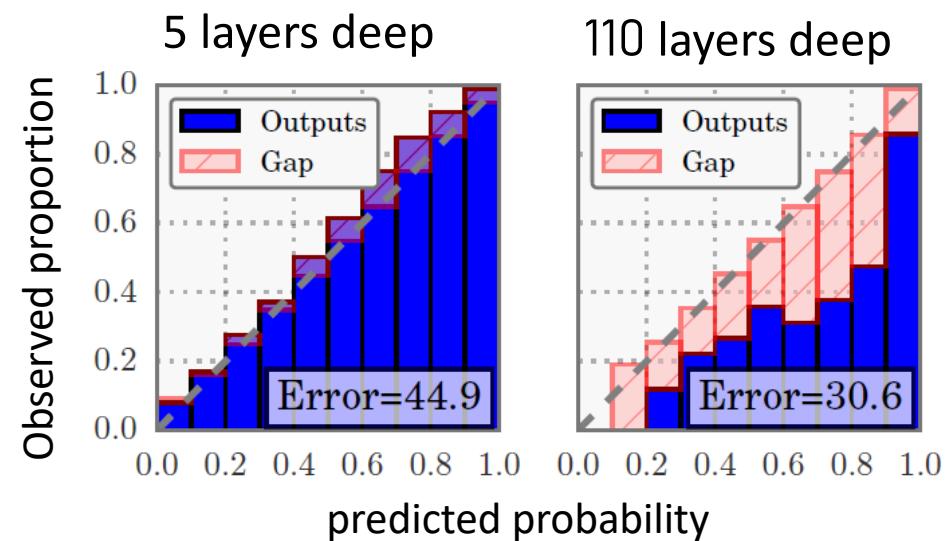
$$p_{\text{pred}}(Y = \text{"rain"})$$

Do NN 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.

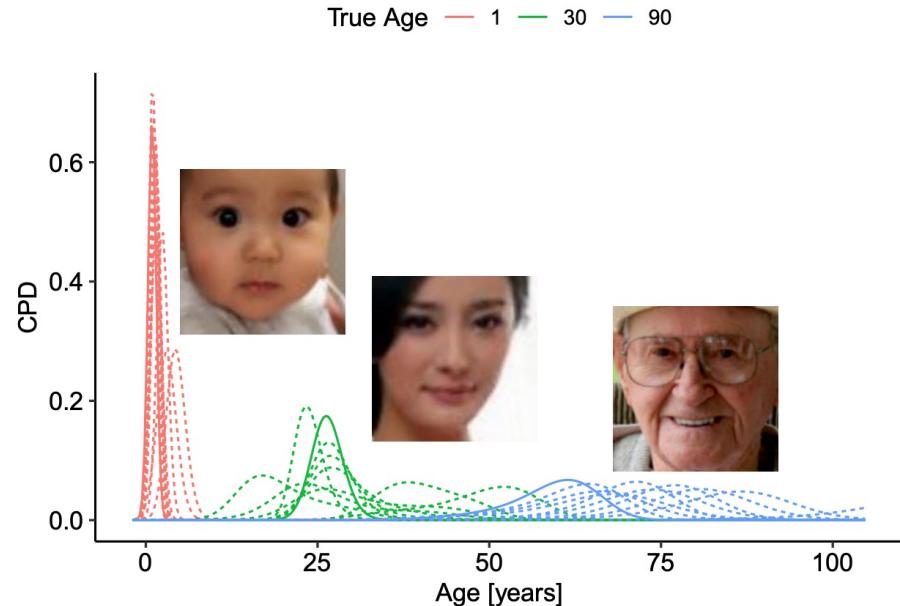
Calibration vs Sharpness

- E.g. in Konstanz, the rain probability for the next day is 20% (over a year)
- A Classifier that predicts $p_{rain} = 0.2$ for each day is well calibrated
- A good classifier should make sharp classifications (prob of rain close to 0% or 100%) but still be calibrated

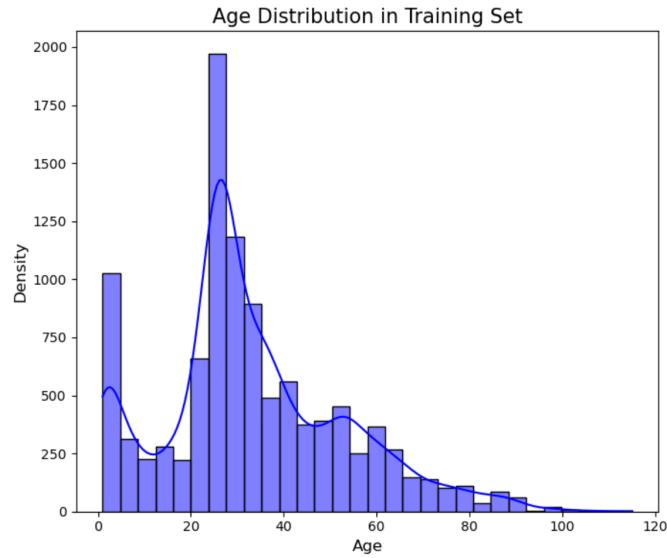
Example

	Good Classifier Prediction p_{rain}	Const Classifier Prediction p_{rain}	Actual Rain
Day 1	0.9	0.2	Yes
Day 2	0.05	0.2	No
Day 3	0.04	0.2	No
Day 4	0.03	0.2	No
Day 5	0.95	0.2	Yes

Calibration vs Sharpness $p(Y|x)$



Sharper predictor depends on x.
But is it calibrated?



A very uninformative but calibrated forecaster would just predict the population distribution (for every) x.

Is there a score that takes into account calibration and sharpness?

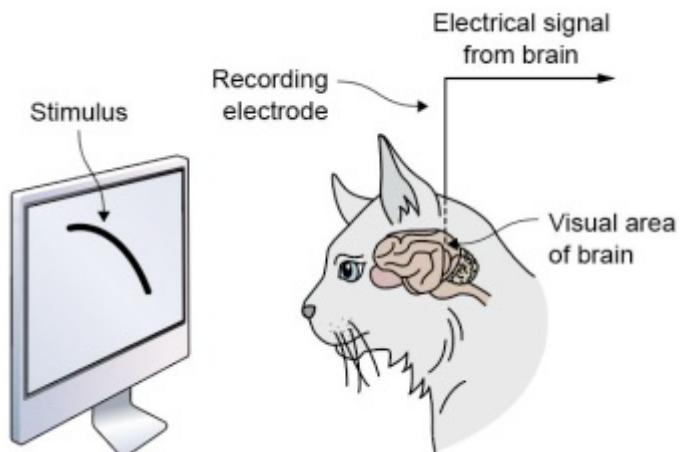
There is: the **NLL** 😊

NLL as general cure-all in probabilistic modeling

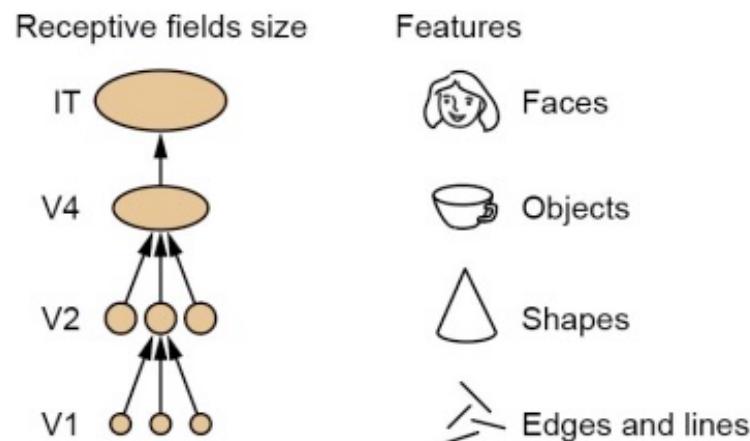
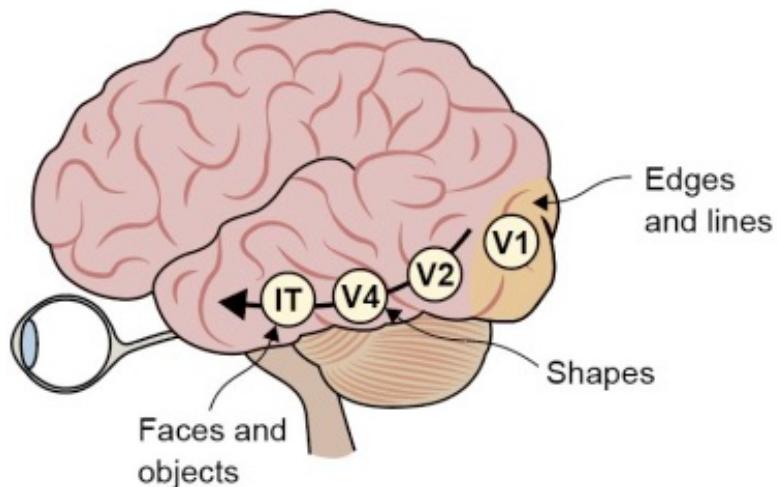
- Maximize likelihood \leftrightarrow minimize negative log-likelihood (NLL)
- The log-score (NLL) is strictly proper score for regression.
- The log-score (NLL) is also strictly proper for classification models.
- To train a probabilistic model: minimize NLL!
- To evaluate or compare probabilistic models: use the validation NLL!

Biological Inspiration of CNNs

How does the brain respond to visually received stimuli?

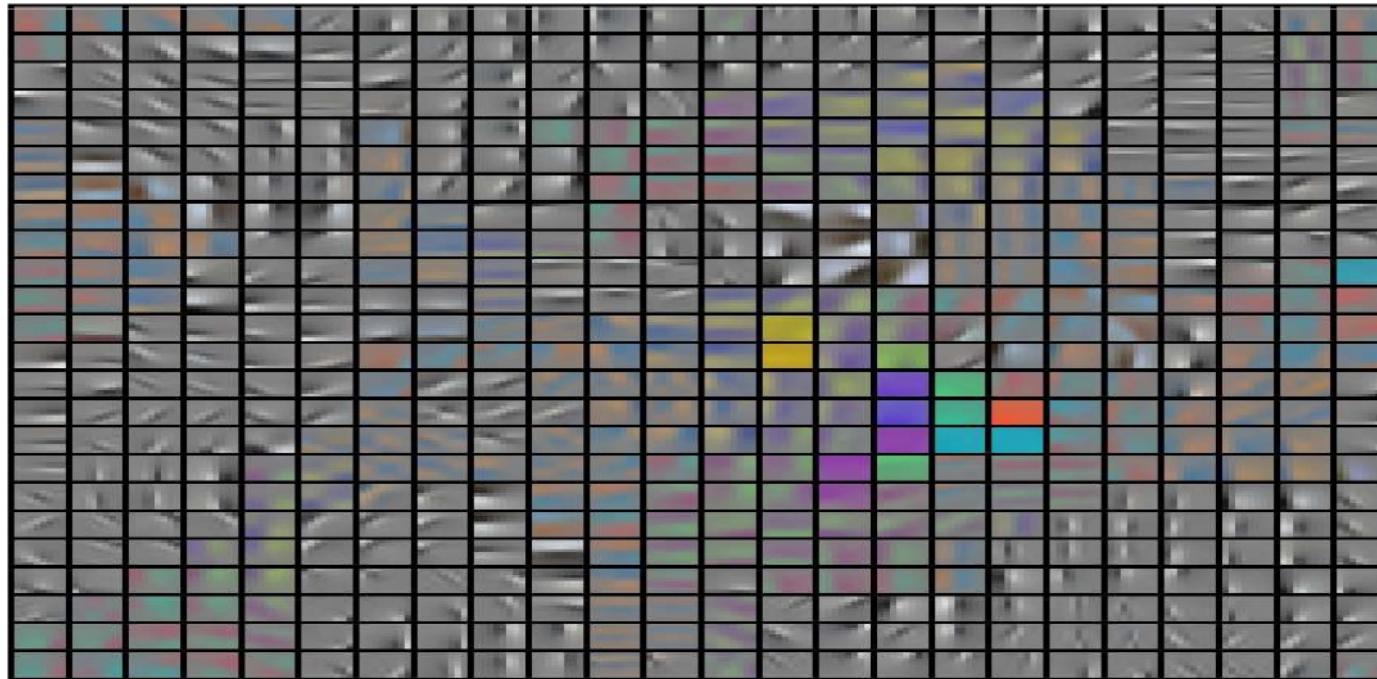
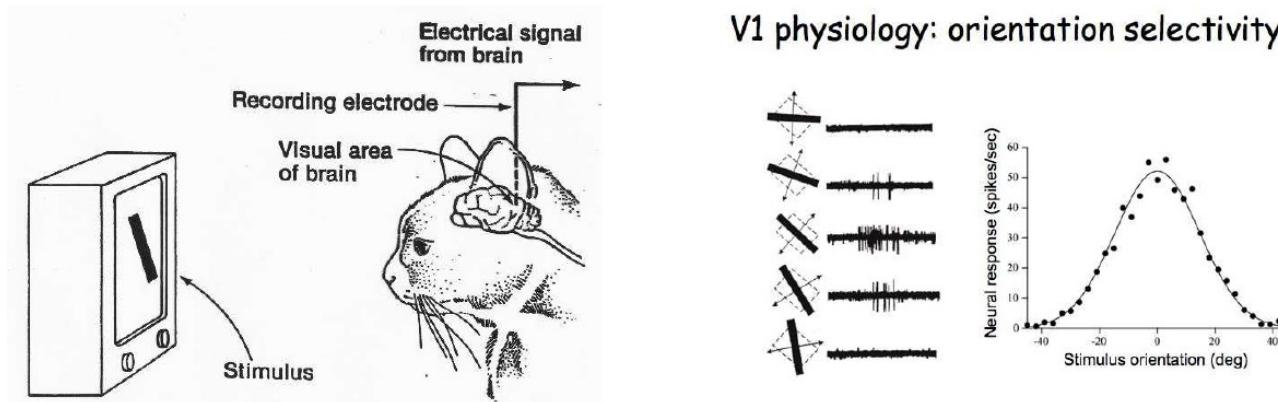


Setup of the experiment of Hubel and Wiesel in late 1950s in which they discovered **neurons** in the visual cortex that **responded** when moving **edges** were shown to the cat.



Organization of the visual cortex in a brain, where neurons in different regions respond to more and more complex stimuli

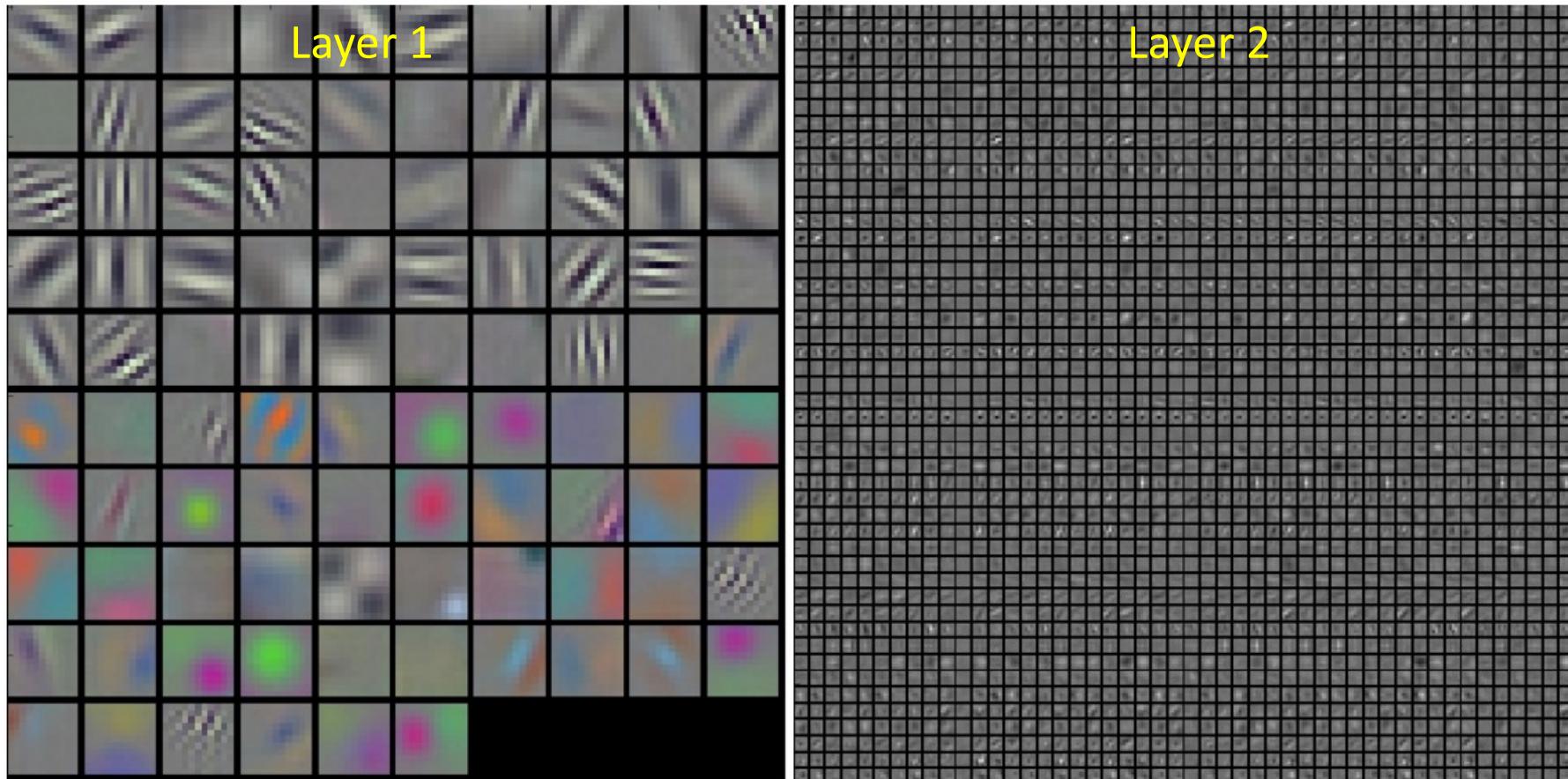
Compare neurons in brain region V1 in first layer of a CNN



Neurons in brain region V1 and neurons in 1. layer of a CNN respond to similar patterns

Visualize the weights used in filters

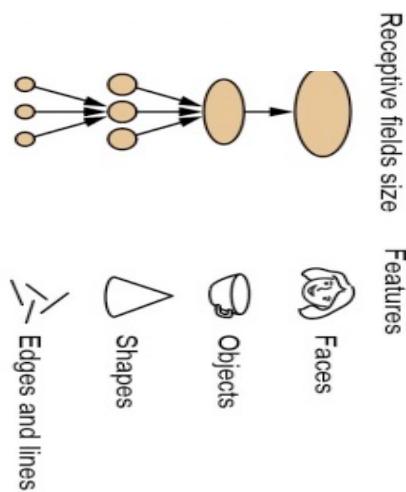
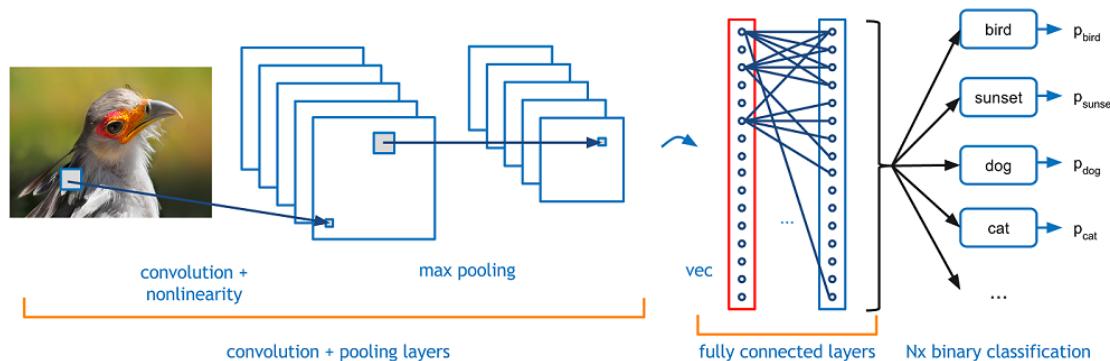
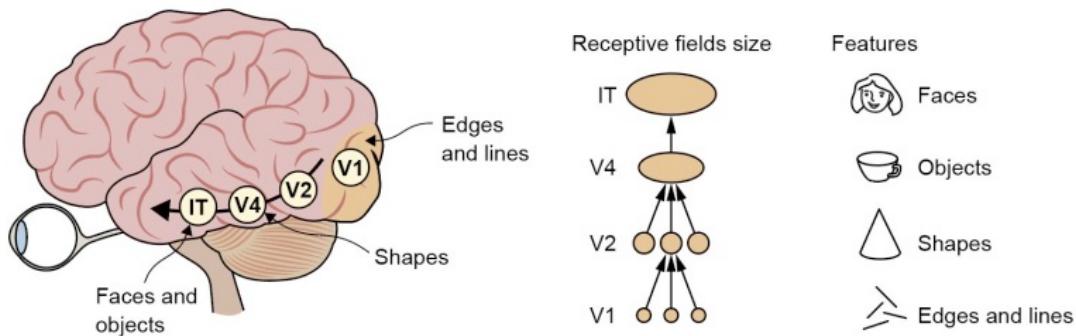
Filter weights from a trained Alex Net



Only in layer 1 the filter pattern correspond to extracted patterns in the image.

In higher layers we can only check if patterns look noisy, which would indicate that the network that hasn't been trained for long enough, or possibly with a too low regularization strength that may have led to overfitting.

Weak analogies between brain and CNNs architecture



xAI = explainable AI
Where does a CNN look at?

Visualize patches yielding high values in activation maps

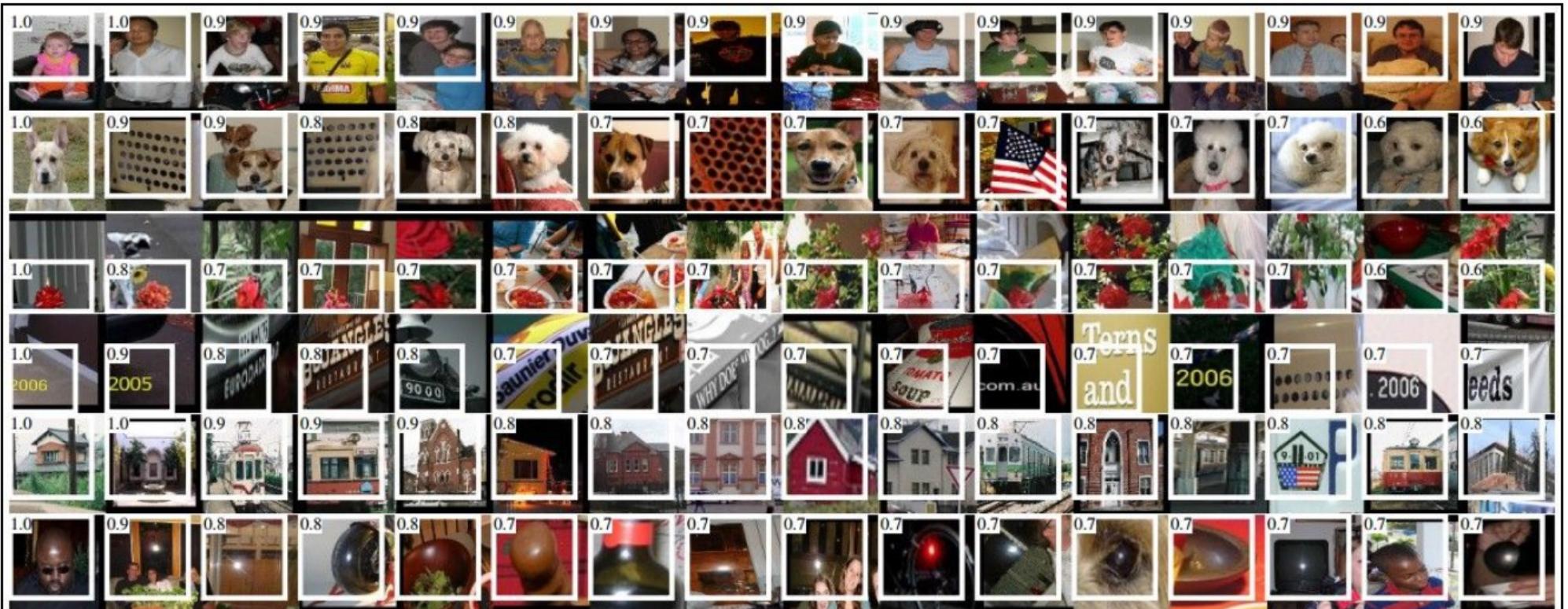
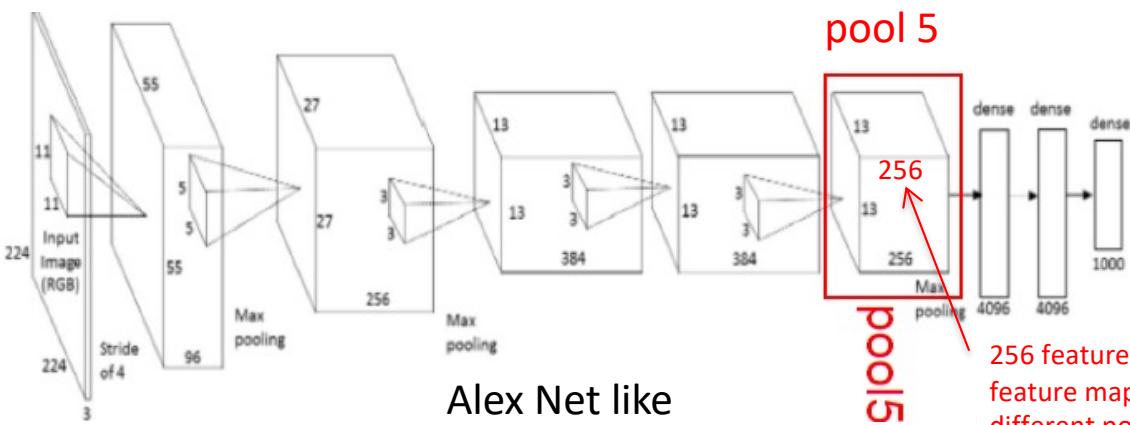


Figure 4: Top regions for six pool₅ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).



<http://cs231n.github.io/understanding-cnn/>

Here we show image patches that activate maps in layer 5 most.

256 feature maps generated by 256 different 3x3 filters. Each feature map consists of equivalent neurons looking at different positions of the input volume.

What kind of image (patches) excites a certain neuron corresponding to a large activation in a feature map?

10 images from data set leading to high signals 6 feature maps of **conv6**



10 images from data set leading to high signals 6 feature maps of **conv9**

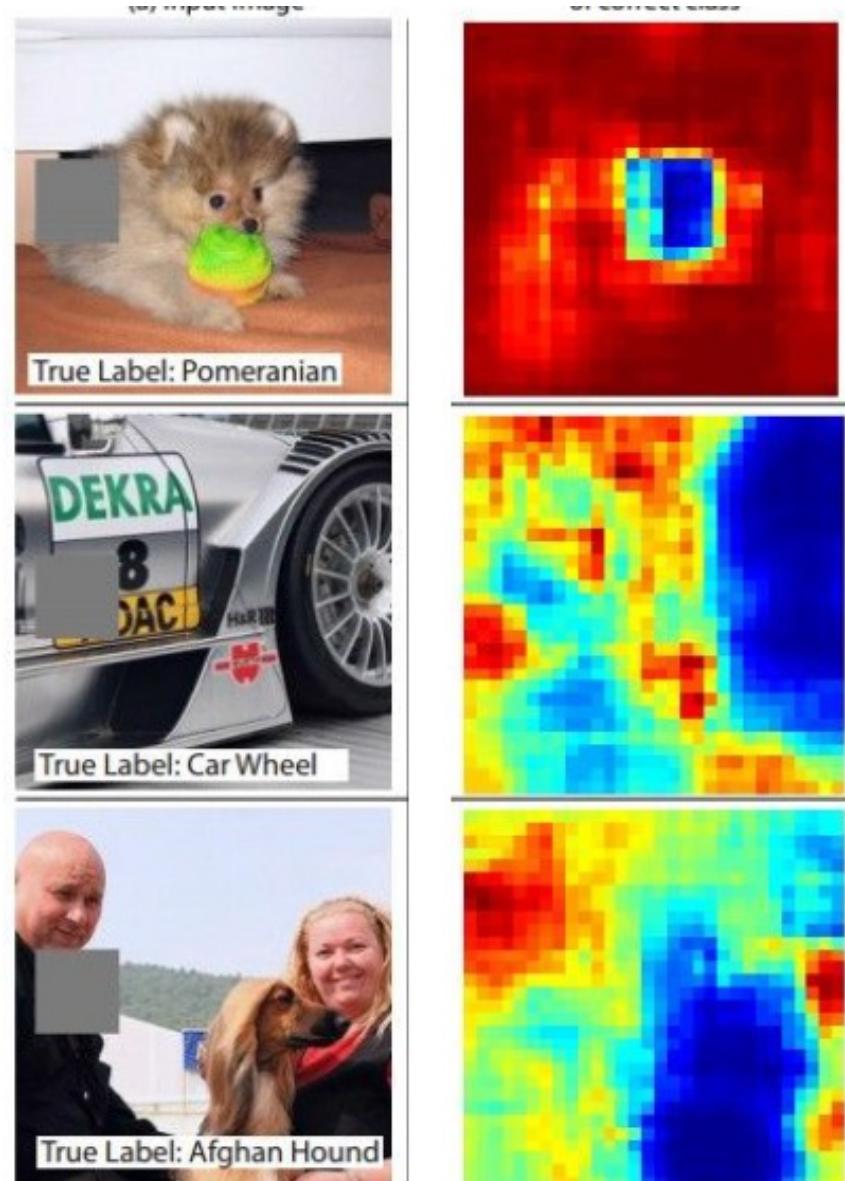


Which pixels are important for the classification?

Occlusion experiments

Occlude part of the image with a mask and check for each position of the mask how strongly the score for the correct class is changing.

Warning:
Usefulness depends on application...



Occlusion experiments [\[Zeiler & Fergus 2013\]](#)

image credit: cs231n

Which pixels are important for the classification? *LIME: Local Interpretable Model-agnostic Explanations*

Idea:

- 1) perturb interpretable features of the instance – e.g. randomly delete super-pixels in an image and track as perturbation vector such as $(0,1,1,0,\dots,1)=x$.
- 2) Classify perturbed instance by your model, here a CNN, and track the achieved classification-score= y
- 3) Identify for which features/super-pixels the presence in the perturbed input version are important to get a high classification score (use RF or lasso for $y \sim x$)

<https://arxiv.org/abs/1602.04938>



(a) Husky classified as wolf



Don't trust

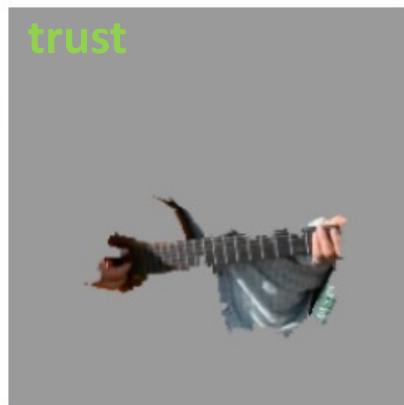
(b) Explanation

-> presence of snow was used to distinguish wolf and husky

-> Explain the CNN classification by showing instance-specific important features
visualize important feature allows to judge the individual classification



(a) Original Image



(b) Explaining Electric guitar



(c) Explaining Acoustic guitar



(d) Explaining Labrador

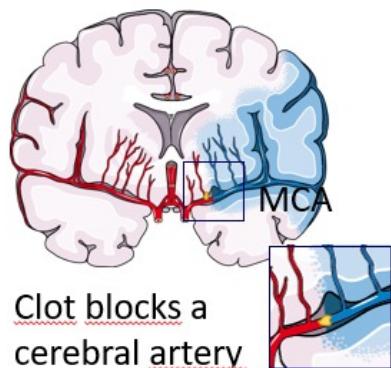
Tf, python code: <https://github.com/marcotcr/lime> -> image tutorial

<https://github.com/marcotcr/lime/blob/master/doc/notebooks/Tutorial%20-%20Image%20Classification%20Keras.ipynb>

xAI in our research

Stroke application: Motivation

Ischemic stroke (88%)



Large vessel occlusion stroke

1.9 million
neurons/min.



Time is brain!

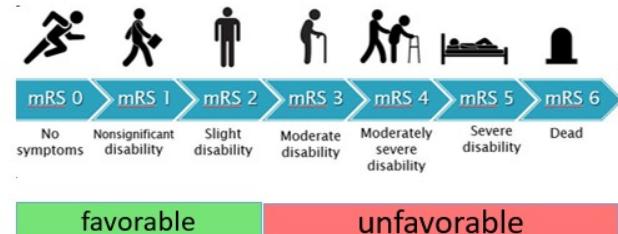
Semi-structured patient data ($n = 407$)

ordinal outcome

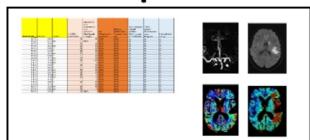
binary outcome

Tabular data
 $x_1 | x_2 | \dots | x_d$
1 2
⋮
n

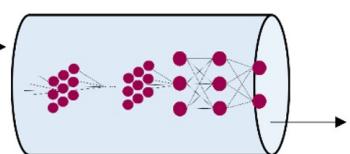
3D image data



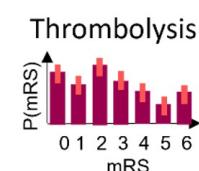
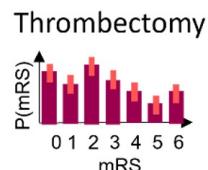
Input



Model

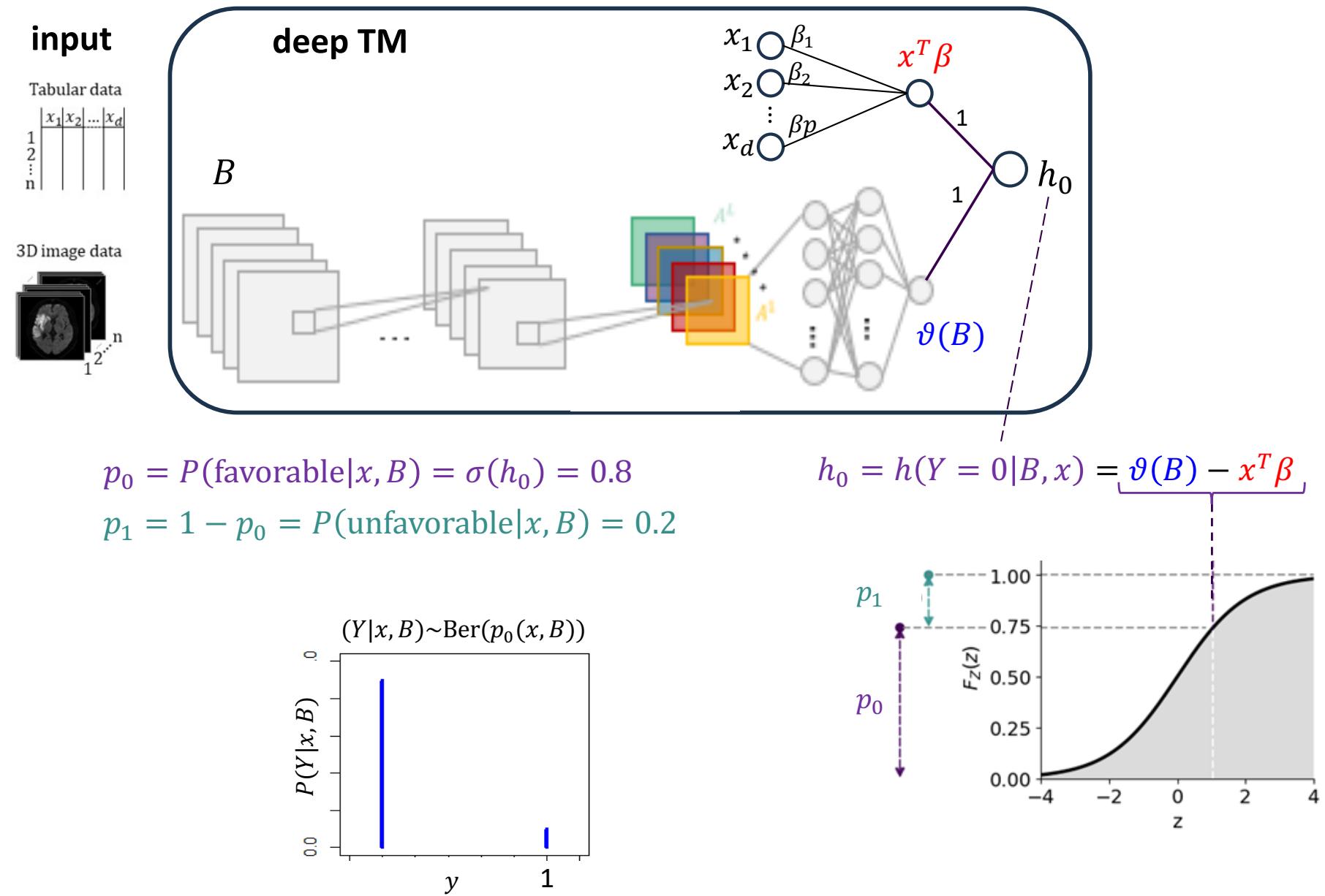


Output

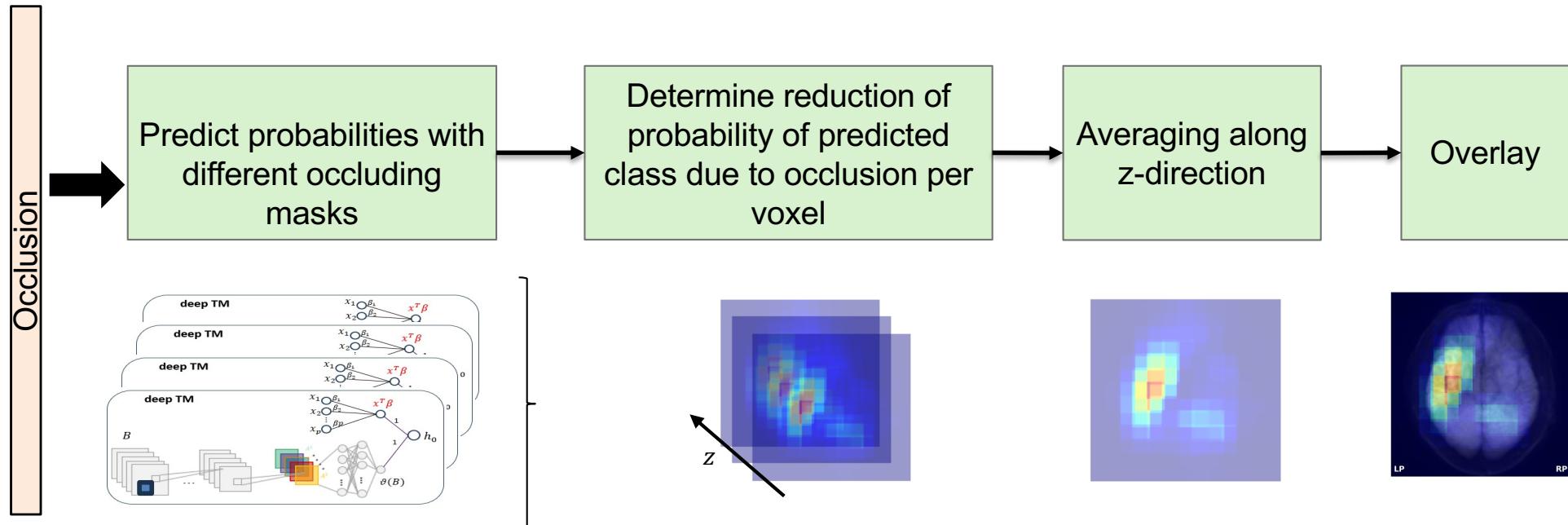
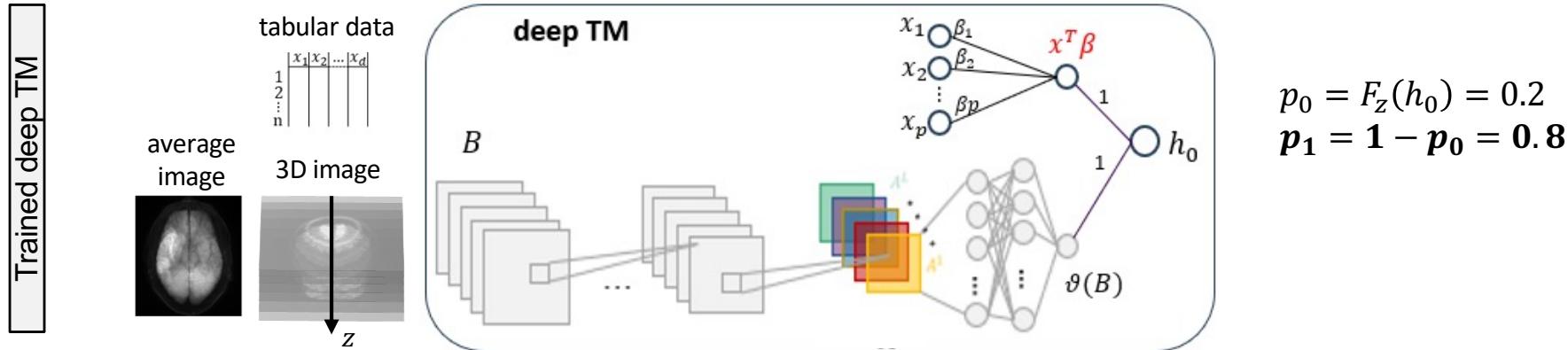


Treat or not?
Why?
Is it certain?

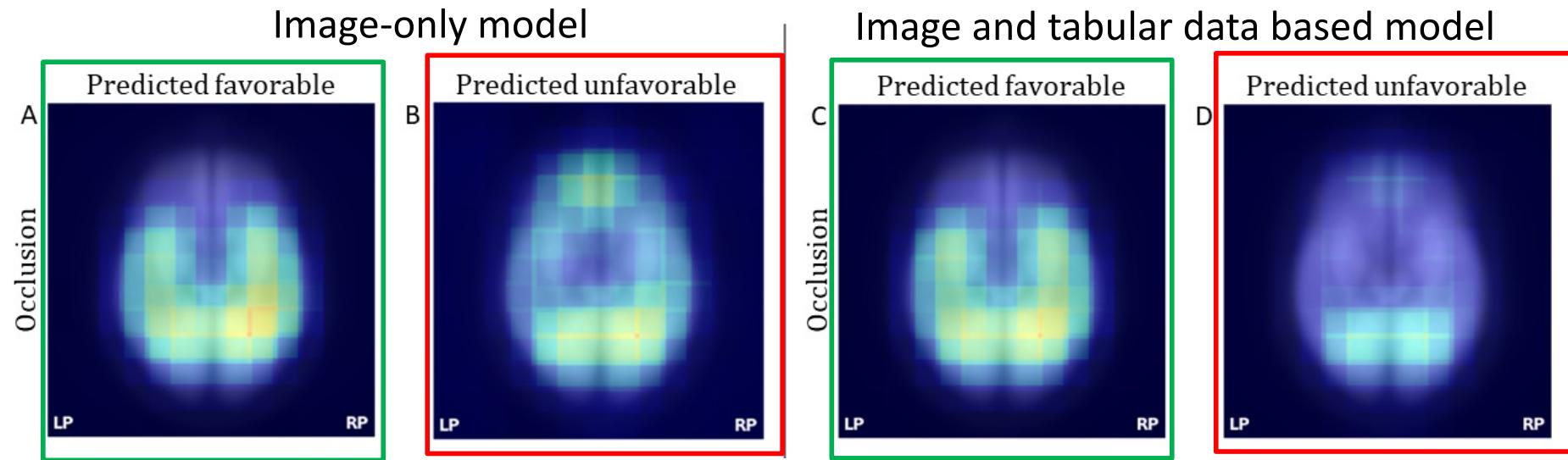
Architecture of a semi-structured dTM for a binary Y



Occlusion for dTMs in stroke application



Explanation maps for patients predicted as un/favorable



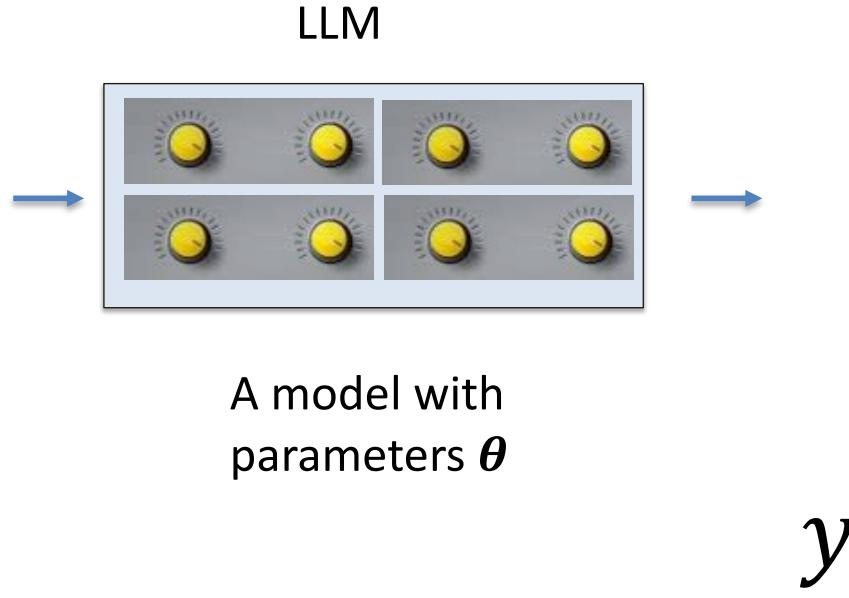
- xAI for image only based model highlights frontal lobe if bad outcome is predicted
- Neurologist know that the frontal lobe indicates the age of the patient
- When providing age as tabular data in addition to the image to the data, the frontal lobe is not anymore hightlighted

Small Excuse on Large Language Models

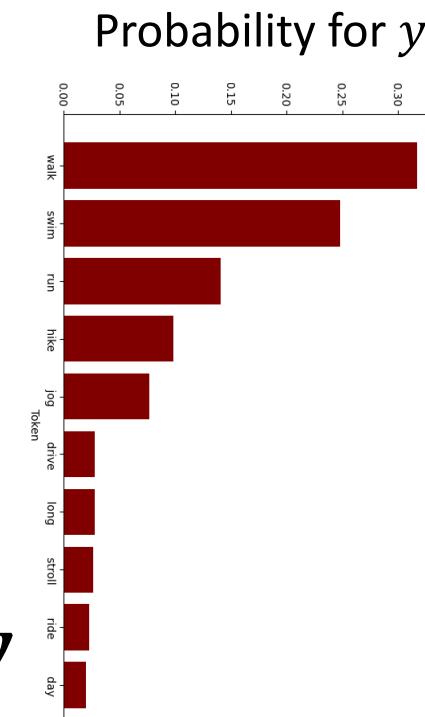
LLM are probabilistic models

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

x



A model with
parameters θ

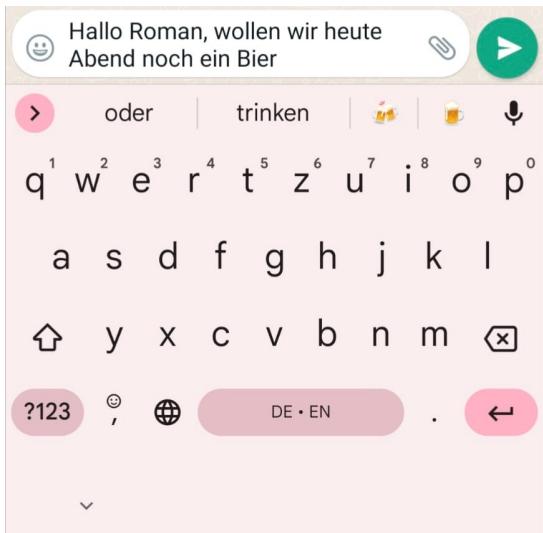


Quiz: Number of parameters in GTP-3.5?

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

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

Generative Language Models: "predict next token"



Generative Language models:

Like you pressing (most) probable suggestion



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.



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

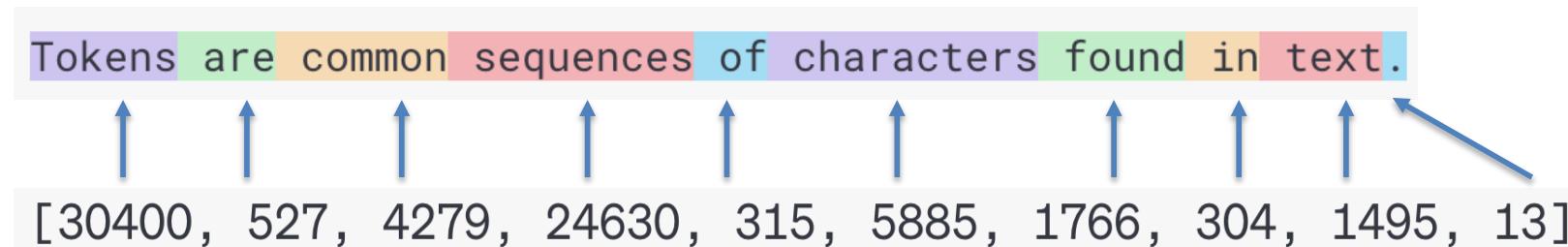
Ingredient to LLM

- Coding Strings to number
 - Tokenization
 - Embedding
- Transformer Architecture
 - Here we give a short introduction

Tokenization: Byte Pair Encoding

ChatGPT works with tokens not with words. It has 50257 different tokens.

<https://platform.openai.com/tokenizer>



Mein Luftkissenfahrzeug ist voller Aale

Mein Luftkissenfahrzeug ist voller Aale

Emergence or stochastical parrot 🦜?

Emergence or stochastical parrot 🦜?

差不多

??不多

[17597, 106, 16937, 43240]

English is coded shorter (since most of text is in English for BPE Algorithm)

Fun Fact: “spell lollipop backwards” used to trick ChatGPT(before 25 Sep Version 2024)

Tokenization: One-Hot-Encoded

- Tokens are categorical data cat ≠ sat
- One Hot Encoding needed
- N size of the relevant vocabulary, levels of categories

Example: Look at vocabulary with N=9 to tokenize the 2 text-samples below

Vocabulary: {'the': 1, 'cat': 2, 'sat': 3, 'on': 4, 'mat': 5, 'dog': 6, 'ate': 7, 'my': 8, 'homework': 9}

Text-sample: “The cat sat on the hat”

“The dog ate my homework”

Tokenized: [1, 2, 3, 4, 1, 5]

[1, 6, 7, 8, 9]

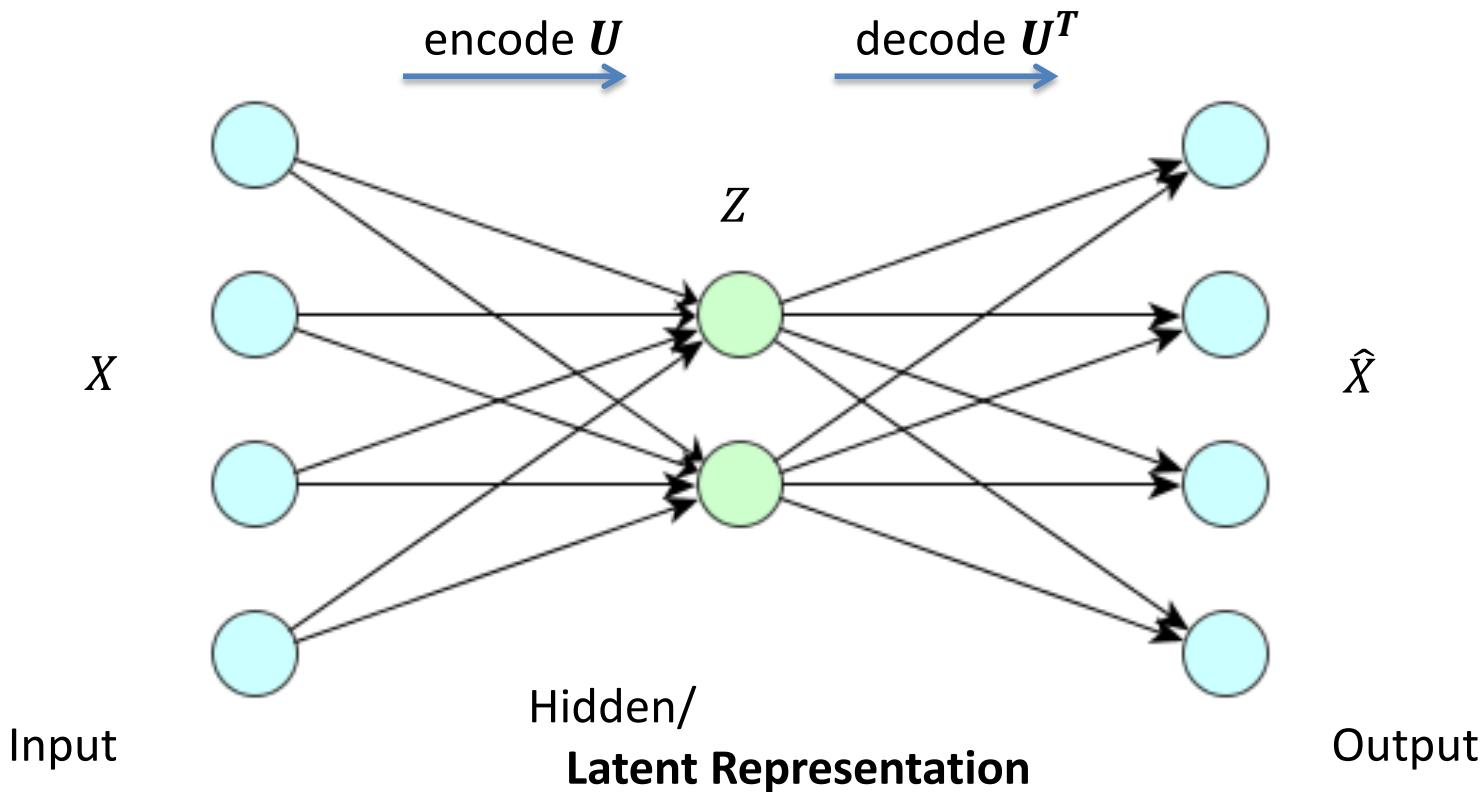
1	0	0	0	0	0	0	0	0	0	the
0	1	0	0	0	0	0	0	0	0	cat
0	0	1	0	0	0	0	0	0	0	sat
0	0	0	1	0	0	0	0	0	0	on
1	0	0	0	0	0	0	0	0	0	the
0	0	0	0	1	0	0	0	0	0	hat

N elements (=vocabulary size)

1	0	0	0	0	0	0	0	0	0	the
0	0	0	0	0	0	1	0	0	0	dog
0	0	0	0	0	0	0	1	0	0	ate
0	0	0	0	0	0	0	0	1	0	my
0	0	0	0	0	0	0	0	0	1	homework

N elements (=vocabulary size)

Recap: PCA Going from High in Low Dimensional Space

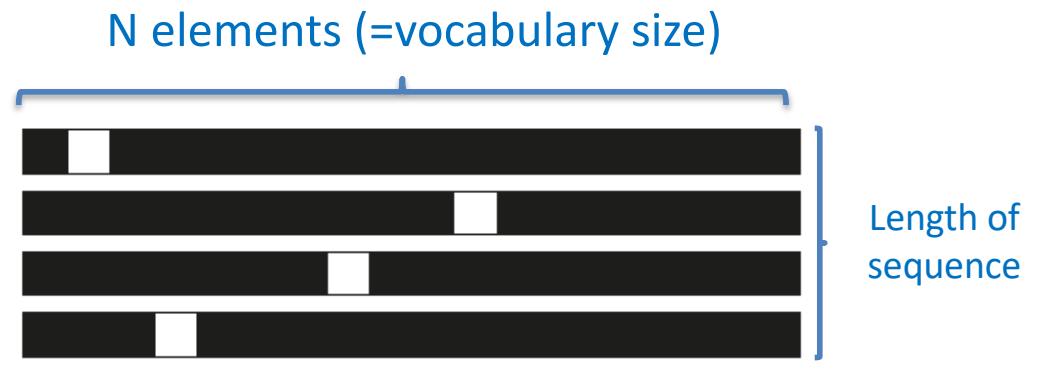


- In PCA the training is done so that reconstruction error $\|X - \hat{X}\|^2$ is minimized

Embeddings

1-hot encodings or tokens

- Based on vocabulary of size N
- sparse: 1 one and $N-1$ zeros
- High-dim: vector-length = N

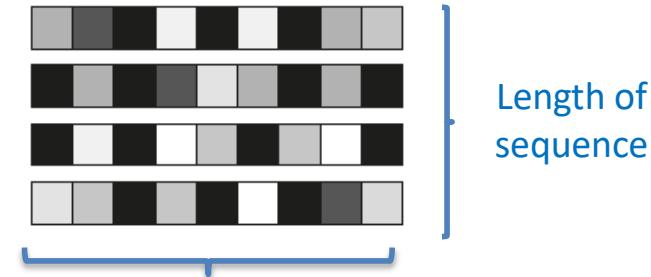


Word embedding's are

- Dense
- Low-dimensional: vector-length = p
- Learned from data via fcNN $N \rightarrow p$

There are pretrained word embeddings like word2vec.

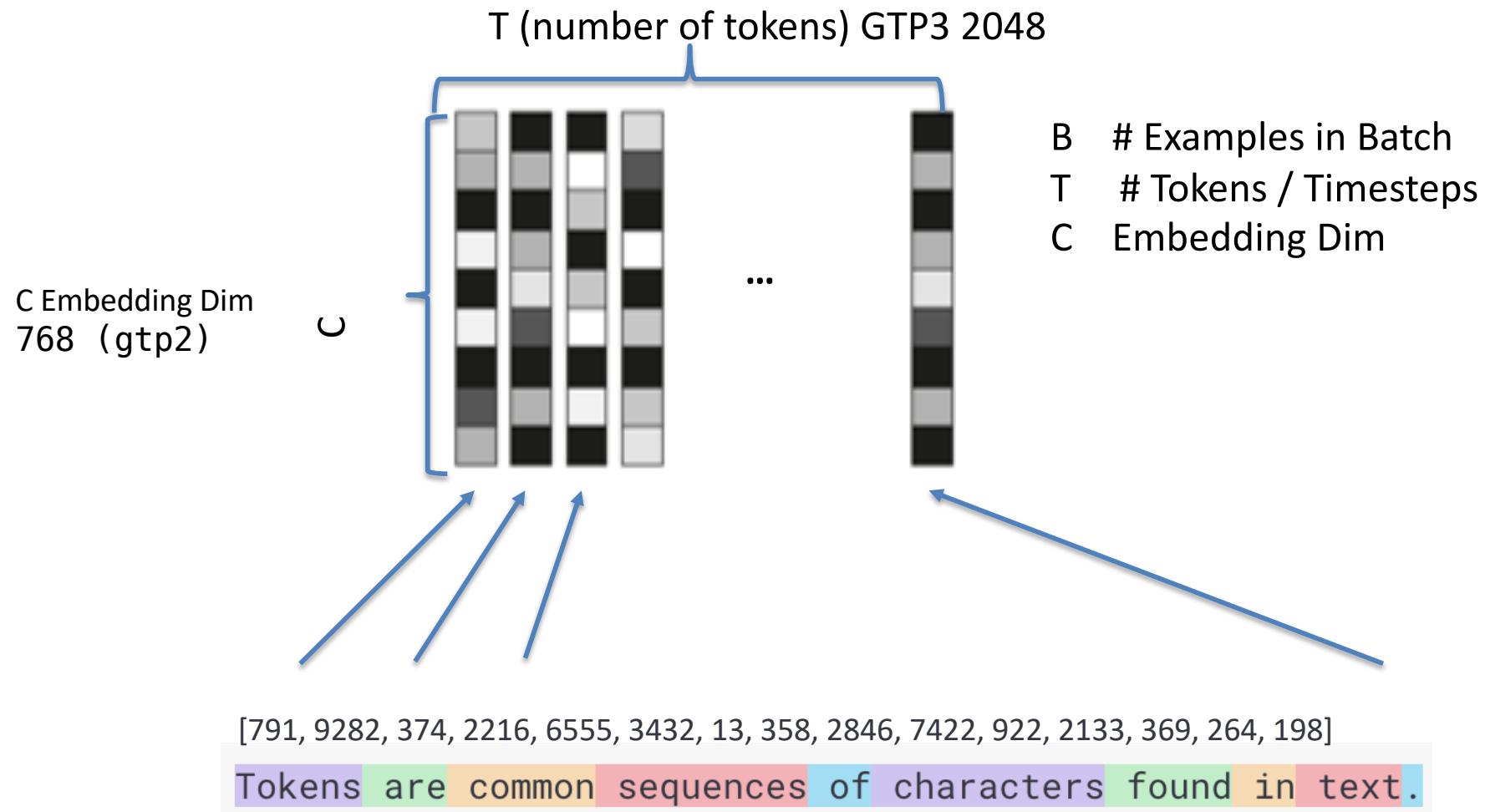
Fully connected NN
with $N \cdot C$ weights



C elements
= dimension of embedding

Recap Understand Text as a Sequence of Vectors

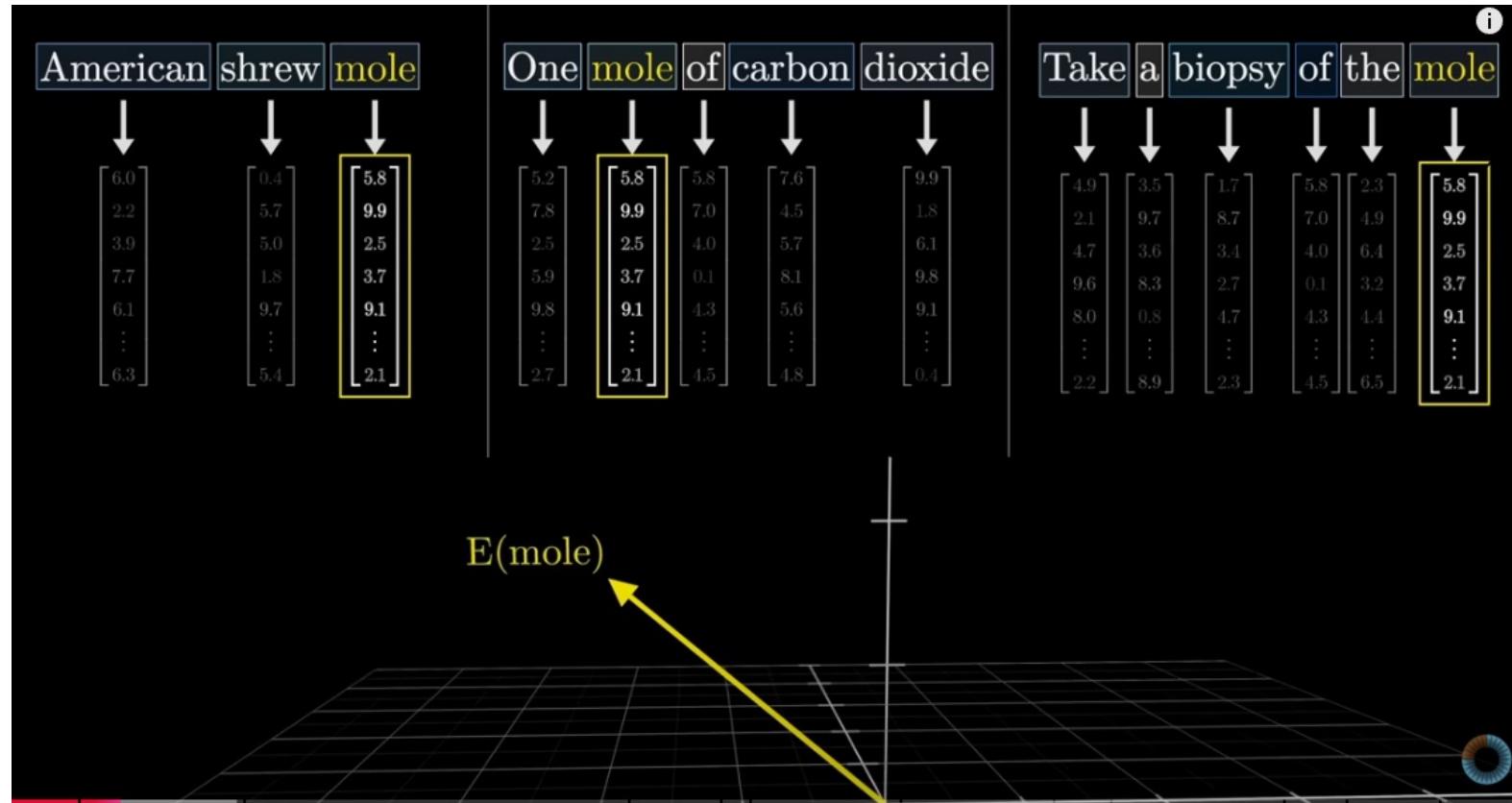
- After Tokenization and Embedding Text looks like



In DL we usually take batches the primary object of interest
a tensors of size 3 with (B, T, C) dimensions

Note, embeddings have no context

Visual



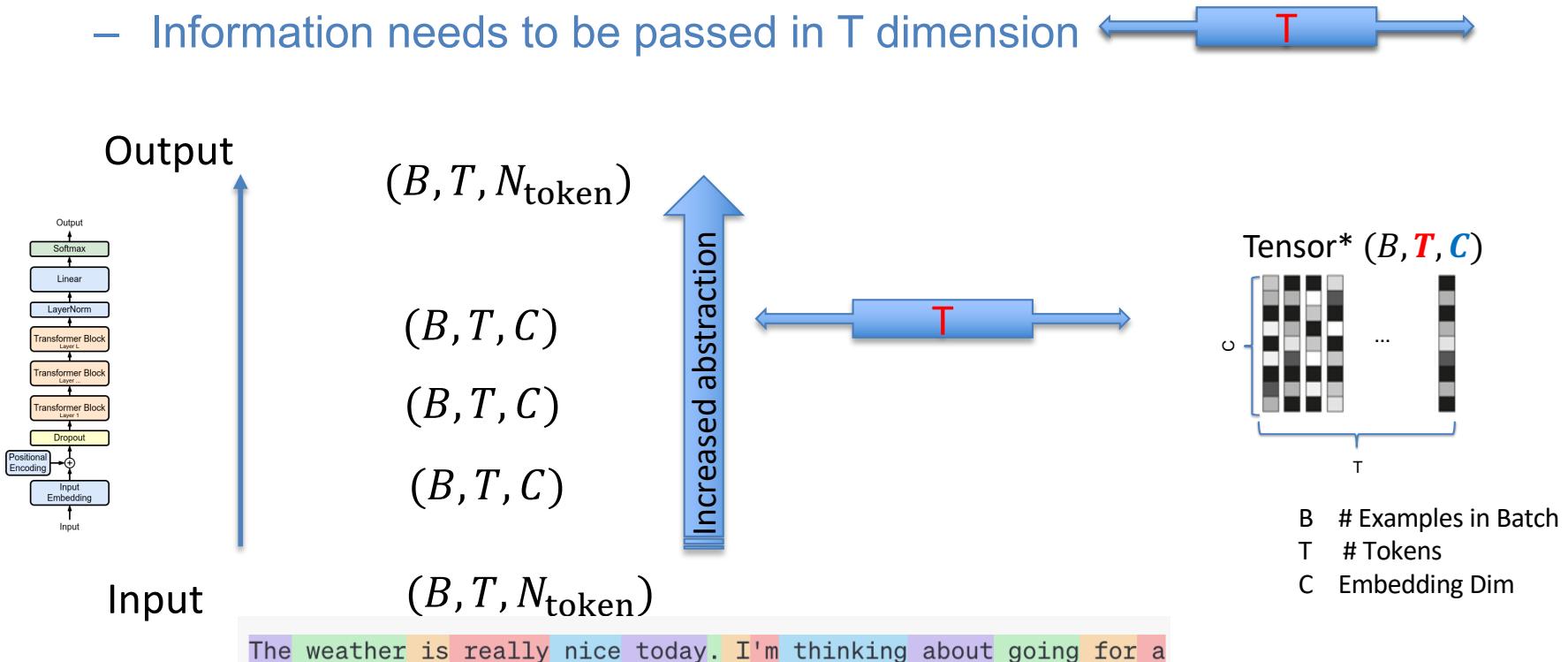
The context should change the embedding.

Mole:

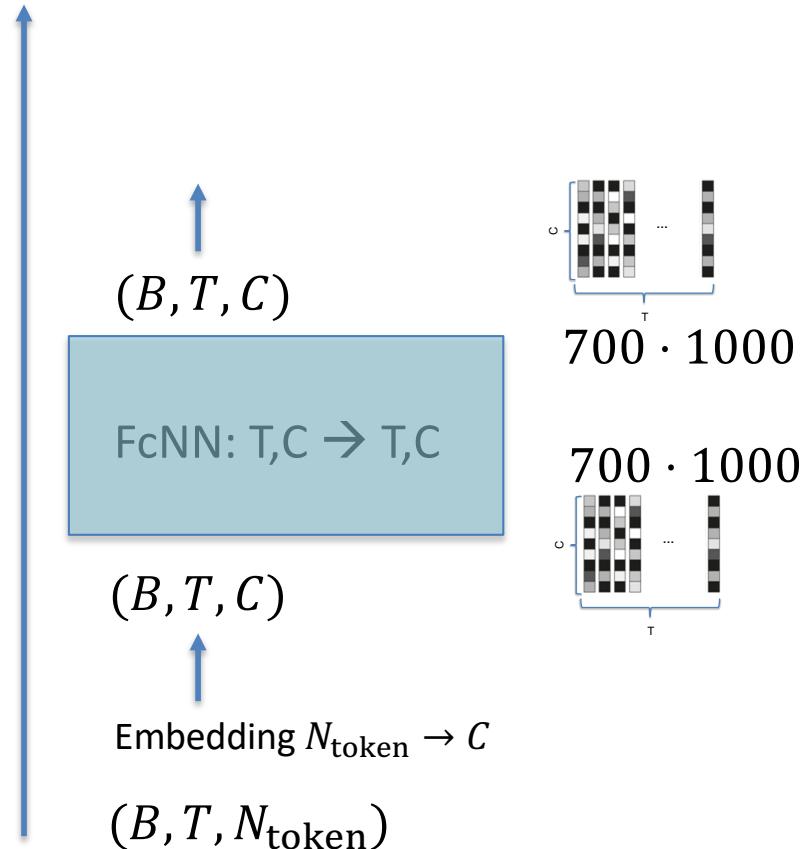
Maulwurf (Tier), Leberfleck (Med), Maulwurf (Spionage), Mol (Chemie), Mole (Landbrücke)

In Language: need for context

- Example
 - Server, can I have the check?
 - Looks like I just crashed the server
- We need the context, to understand the meaning of server.
 - Information needs to be passed in T dimension



A naïve approach for context Visual improvement



One Block would have
 $(700 \cdot 1000)^2 \approx 490B$ parameter

Too much for a single layer

Other architectures have been developed in the past.
RNNs and LSTMs

Context: Transformer

.03762v5 [cs.CL] 6 Dec 2017

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Ilia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including

<https://arxiv.org/abs/1706.03762>

- Introduced for translation
- Basis for Language models (GPT-2, GPT-3, Chat-GPT, Bert)

Google Scholar search results for "Attention is all you need". The search bar contains the query. Below it, there are sections for "Articles" (1 result) and "Lookup". The main result is a summary of the paper by Vaswani et al., followed by a link to the PDF on neurips.cc. At the bottom, there are options to save, cite, and view related articles.

Articles 1 result (0.02 sec)

Lookup

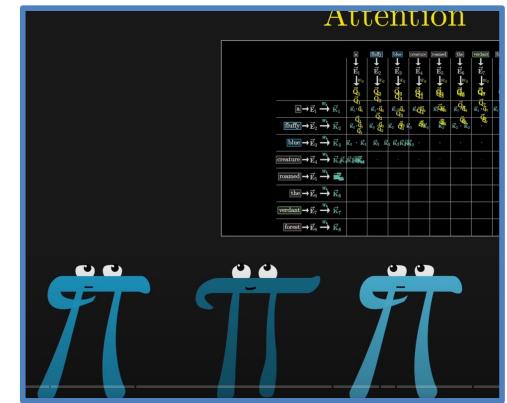
[Attention is all you need](#)
A Vaswani, N Shazeer, N Parmar... - Advances in neural ..., 2017 - proceedings.neurips.cc

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder and decoder configuration. The best performing such models also connect the encoder and decoder through an attention mechanism. We propose a novel, simple network architecture based solely on an attention mechanism, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more ...

☆ Save ⚏ Cite Cited by 62992 Related articles All 46 versions ☰

Sources for Transformer

- 3Blue1Brown:
 - https://www.youtube.com/watch?v=eMlx5fFNoYc&t=559s&ab_channel=3Blue1Brown
- Live Coding (and explanation) Andrej Karpathy nano-GPT
<https://youtu.be/kCc8FmEb1nY>

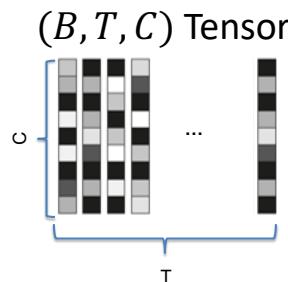
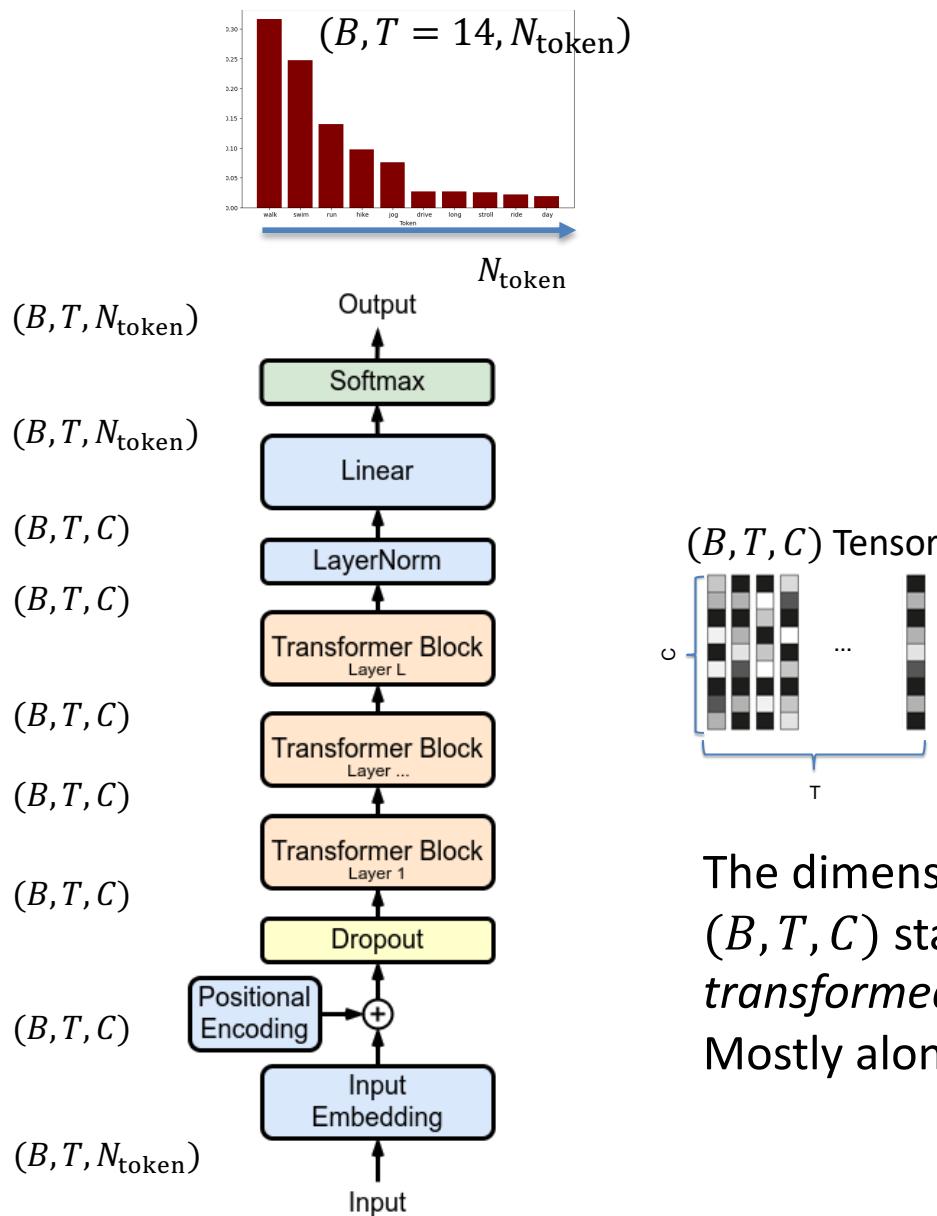


Further Resources

- CS25 Andrej Karpathy Introduction to Transformers
 - Nice intuition with message parsing <https://www.youtube.com/watch?v=XfpMkf4rD6E>
- Lennart Svensson: <https://youtu.be/0SmNEp4zTpc>
 - Good intuition (weighted average, mathematical sound)
- Ava Soleimany: <https://www.youtube.com/watch?v=QvkQ1B3FBqA>
 - Good intuition (Search query idea)
- <https://jalammar.github.io/illustrated-transformer/>
 - Nice illustrations
- CS25 Andrej Karpathy Introduction to Transformers
 - Nice intuition with message parsing <https://www.youtube.com/watch?v=XfpMkf4rD6E>

Follow the forward pass

B # Examples in Batch
 T # Tokens
 C Embedding Dim



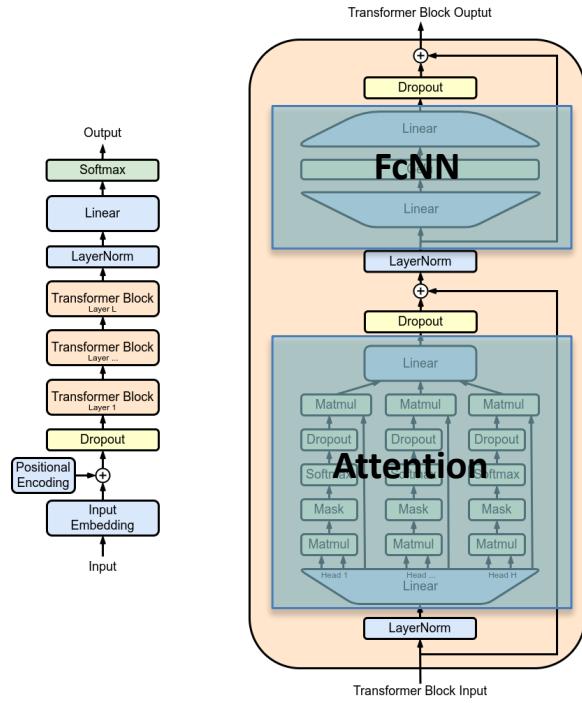
The dimension of the tensor (B, T, C) stays, it is just *transformed*.
 Mostly along C dimension.

[791, 9282, 374, 2216, 6555, 3432, 13, 358, 2846, 7422, 922, 2133, 369, 264, 198]

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

Transformer Block

B # Examples in Batch
 T # Tokens
 C Embedding Dim



(B, T, C)

FcNN Block

FcNN with C in and out.
 Updates representation

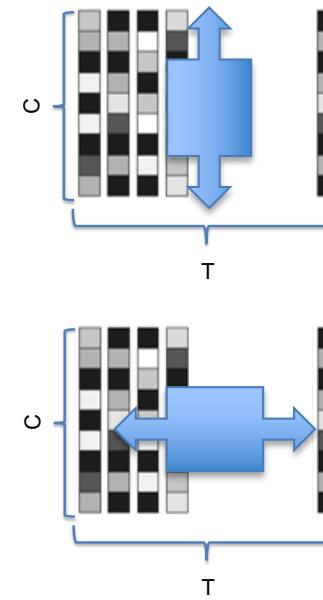
(B, T, C)

Attention Block:

Updates with information
 from (left) neighbors

(B, T, C)

Information Processing



Information is processed in two steps first

- Along T
- Along C

Total Memory $O(T) + O(C)$

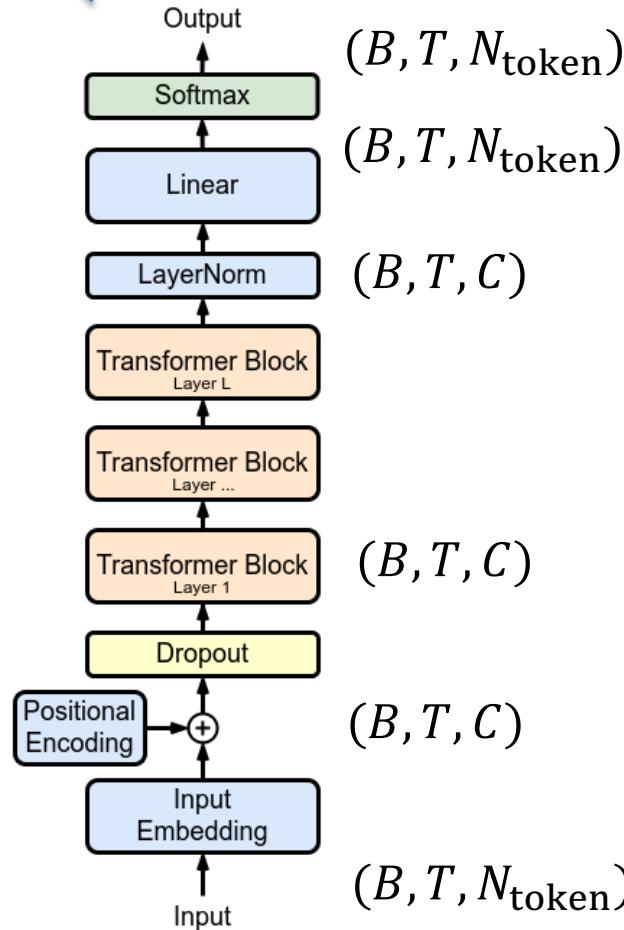
Most operations along C-Direction

Training parallelism / Causal LLM

B # Examples in Batch
 T # Tokens
 C Embedding Dim

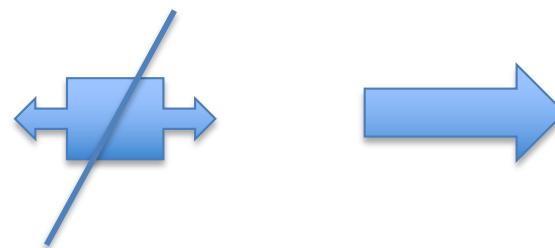
Shift input one token to the left

The weather is really nice today. I'm thinking about going for a walk
 [128000, 791, 9282, 374, 2216, 6555, 3432, 13, 358, 2846, 7422, 922, 2133, 369, 264]



Attention (“Causality”)

When predicting next word only information from previous token must be used.



[128000, 791, 9282, 374, 2216, 6555, 3432, 13, 358, 2846, 7422, 922, 2133, 369, 264]

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

Using Transformers

```
### Getting the model
model_name = "gpt2-medium" #'deepseek-ai/DeepSeek-R1-Distill-Llama-8B", ...
model = AutoModelForCausalLM.from_pretrained(model_name) #← Downloading model
model.num_parameters() / (1E9) #0.35 Billion Weight ChatGPT 3.5 has 175B

### Tokenizing
tokenizer = AutoTokenizer.from_pretrained(model_name)
input_context = "The weather is really nice today. I'm thinking about going for a"
input_ids = torch.tensor(tokenizer.encode(input_context)).unsqueeze(0) #128000, 791,...,264
input_ids.shape #1, 14 this is short for (1,14,50257)

### Predicting the next Tokens
outputs = model(input_ids) # We ask the model for the next token
probs = torch.nn.functional.softmax(outputs.logits, dim=-1)[0]
```

Input:

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

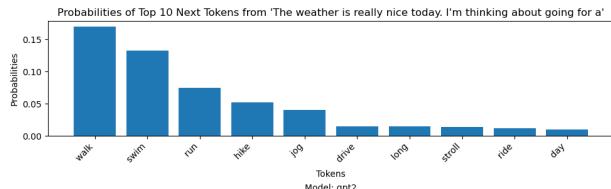
probs[,2,]

is the probability distribution of the word after “The weather is”

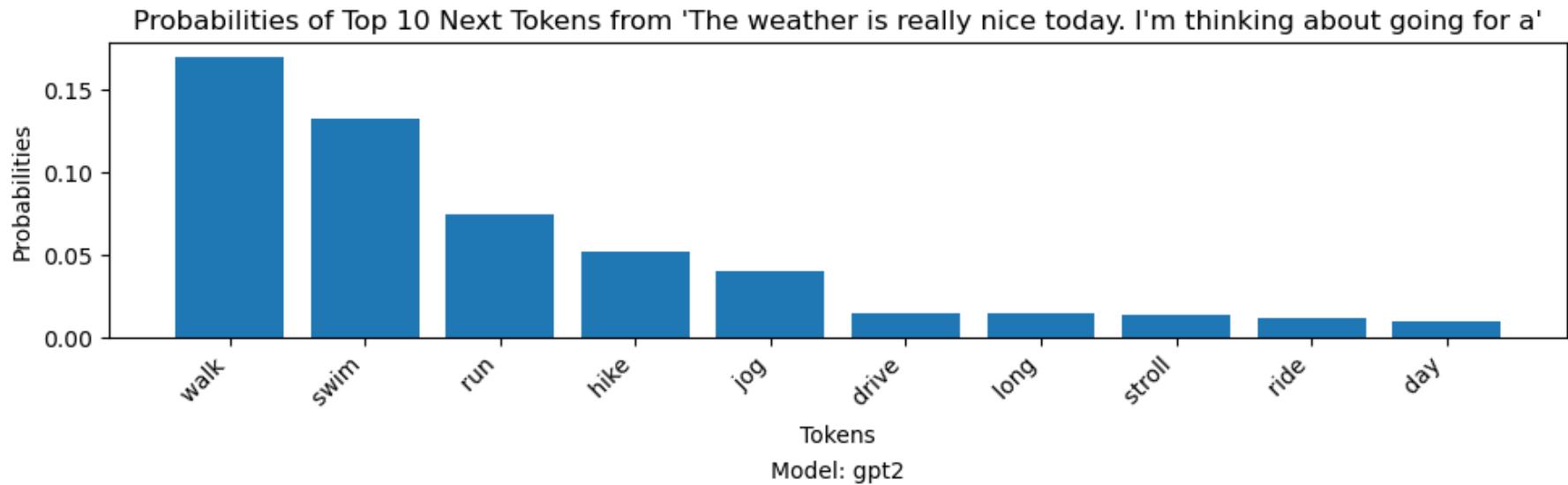
More interesting is

probs[,-1,]the probability for the next word / token after

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



Results



Now sample from that distribution, and add this token to generate new text. Say, we sampled swim

```
new_input = "The weather is really nice today. I'm thinking about going for a"
```

Sampling next token from distribution

Model: gpt2-medium

Prompt: The weather is really nice today. I'm thinking about going for a jog around the neighborhood." Beth dropped her rain boots, pulled out a notebook from her bag and scanned the documents that filled up the notable shelf above her desk. A few days ago, Mr. Richard Fair asked to borrow a laptop to help him look swim today," he said, gesturing to a hastily painted house in a garden at the expense of some gardening students who had come to try and solve matching English numbers.

crash tomorrow." Beetle gold: Ranzbash Talib was part of Russia's 2008 Olympic hockey gold medal team We are really looking forward to lift and have great club setting over Christmas than for beer and burro rounds today

Only take the most probable token

walk today. I'm thinking about going for a walk today.

Findings:

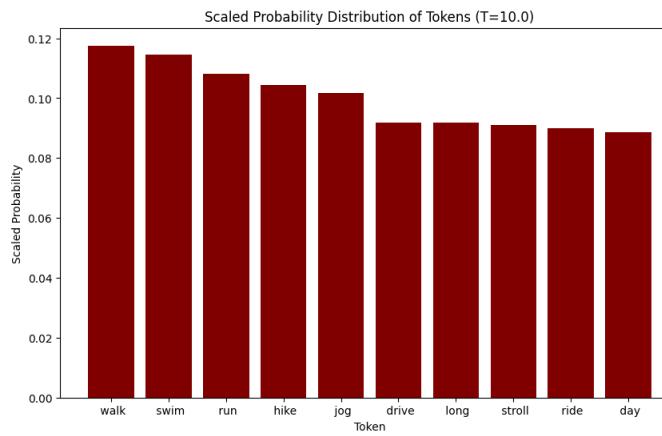
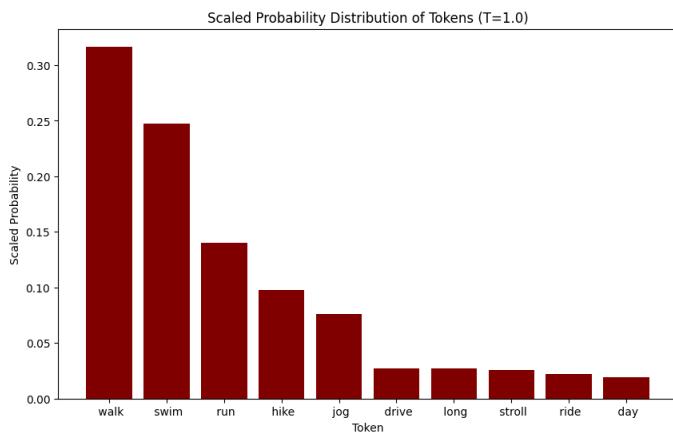
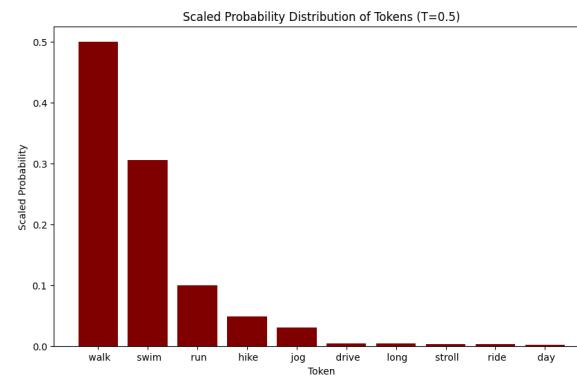
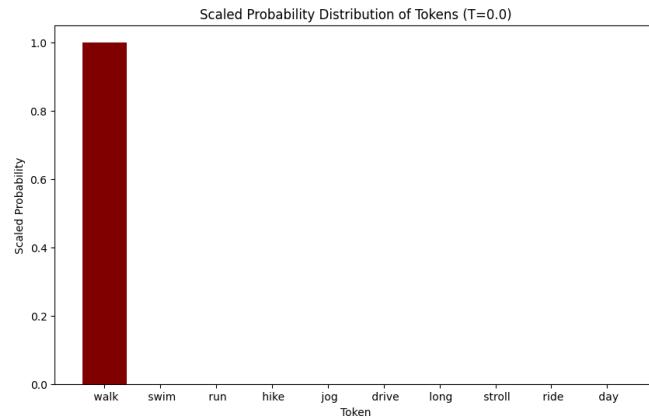
Sampling from distribution, yields incoherent text. Sampling using maximum can yields repetitive text

Tricks:

Apply a temperature, top-k, top-p and p_min

Temperature scaling: scaled probability

$$p(y|\text{"prompt"}) = \frac{\exp(\text{logits}/\text{temperature})}{\sum_{y'} \exp(\text{logits}_{y'}/\text{temperature})}$$



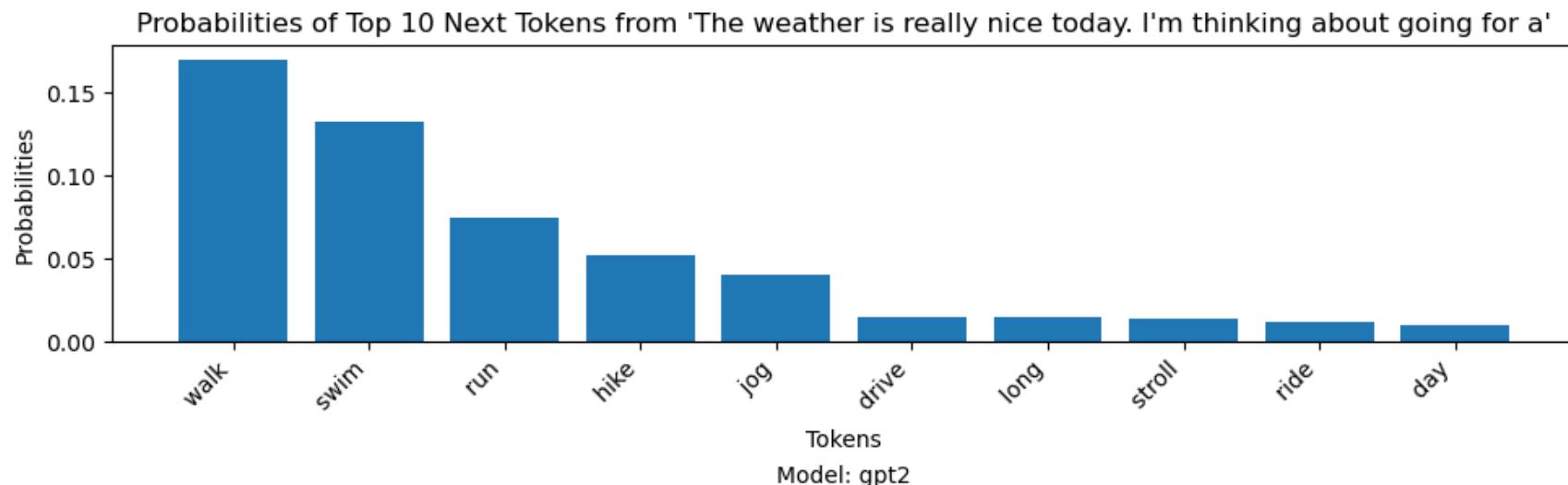
Additional Sampling Tricks

- Top-p (Nucleus Sampling):
 - Dynamically selects tokens until **cumulative probability exceeds p** (e.g., 90%).
 - **Adaptive:** Number of tokens changes based on model confidence.
 - Focuses sampling on the **most likely tokens**, maintaining flexibility
- Min-p Filtering:
 - Removes tokens with probabilities below a minimum threshold (e.g., 5%).
 - Acts as a final confidence filter to exclude low-probability outliers.
 - Ensures all selected tokens have a baseline likelihood.
- Guidelines
 - Lower Temperature less creative
 - Instructions based models, have low temperature
 - For [SmallLM2-1.7B-Instruct](#)
 - max_new_tokens=50, temperature=0.2, top_p=0.9

Playaround with LMMs

Have a look at the following NB:

https://github.com/tensorchiefs/dlwl_eth25/blob/master/notebooks/03_sampling_Lmm.ipynb



Training

- Training (GPT-3)
 - Trained on “complete Internet” (common crawl 400 10^9 tokens) + Wikipedia ...
 - Trained to predict next token (maximum likelihood principle)
 - Trained for few months (CO₂ equivalent 120 cars for one year)
 - Prediction 100 pages 0.4 kwh
 - Data before **Q4 2021**
- In Language Modelling, instead of NLL the average of NLL over a sequence of length N is calculated, as follows.
 - $PPL = \exp(NLL/N)$

Training LLaMA2

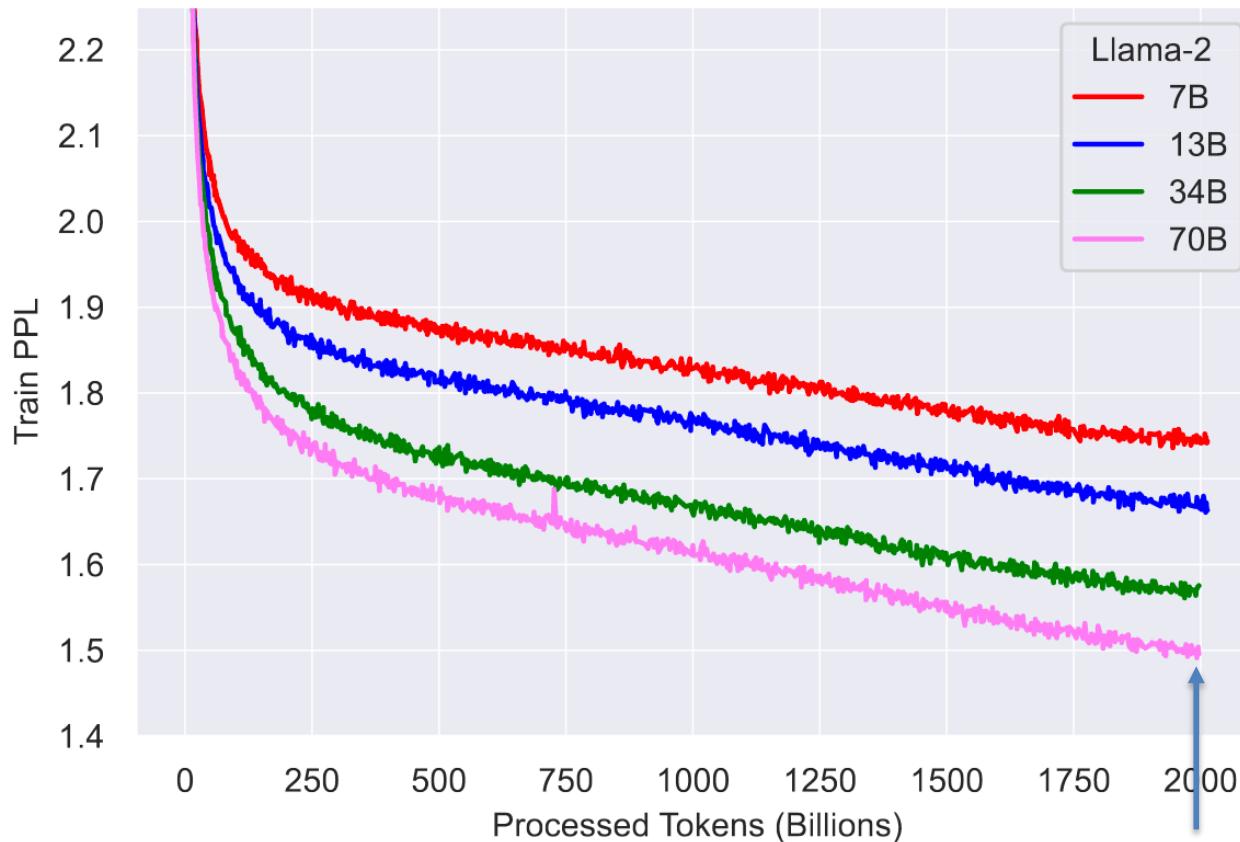
Number of parameters 7B-70B

Number of Tokens $N_{\text{token}} = 32k$

Context Length $T = 2k - 4k$

Number of Layers 32-80

Context Size C=4096-8192



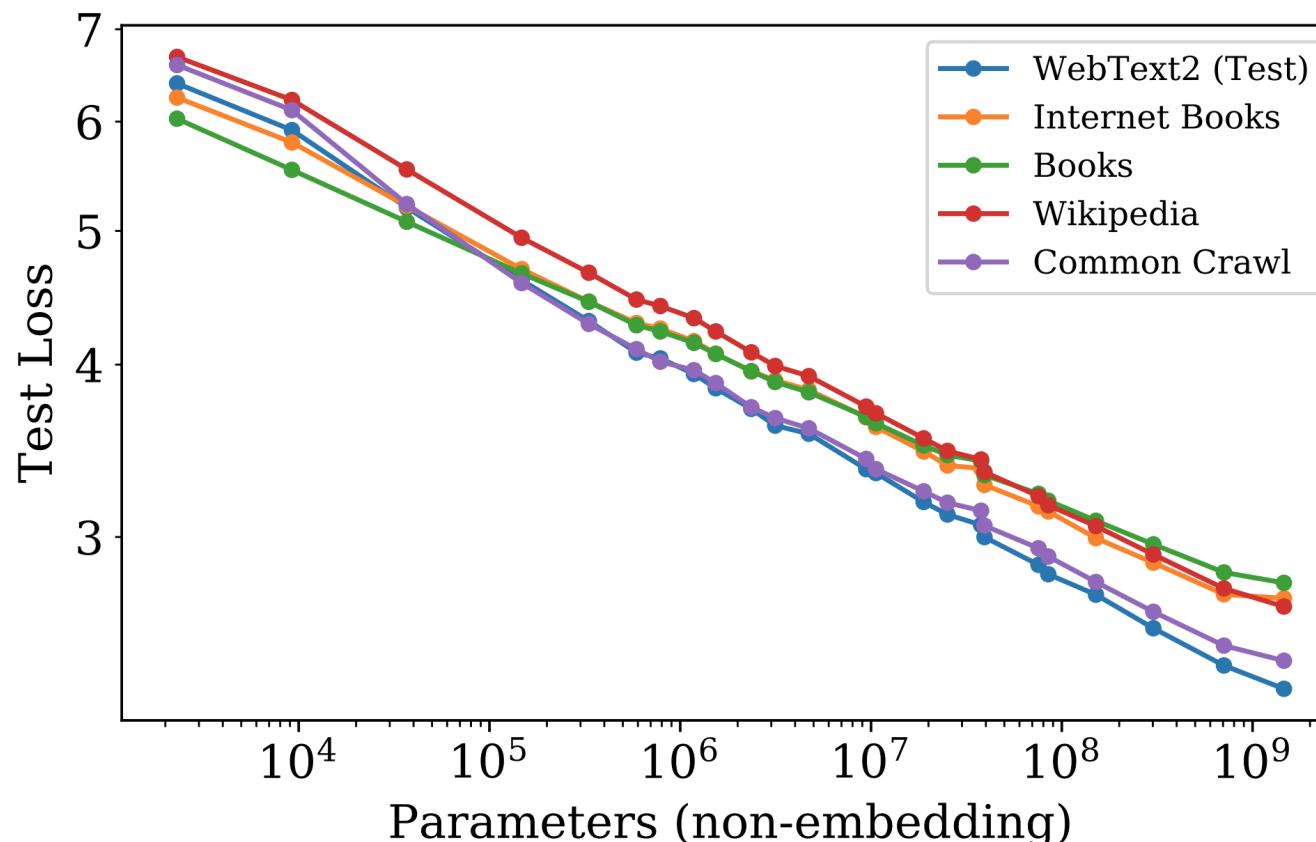
For 70B
1.7 Million GPU Hours
291.42 tCO₂eq
(70 Household for a year)

Scaling Laws of LMM

- Performance L in NLL (hold-out-set) scales like

$$- L(N, D) = A + \frac{B}{N^{0.34}} + \frac{C}{D^{0.28}}$$

- N Number of parameters of model
- D Data set size (A, B, C) constants



Alignment Finetuning

Plain vanilla LLM

prompt **What is the capital of France?**

What is France's largest city?

What is France's population?

What is the currency of France?

Finetuned versions (Chat-GPT and Chat LLama)

prompt **What is the capital of France?**

Paris

Further aspects of fine tuning (Helpful, Honest, Harmless)

Exercise LLM



- Do Exercise 03_sampling_llm

Emergent properties of LLM



Yann LeCun
Jan 2022

"I take an object, I put it on the table, and I push the table [...] GPT-5000 is never gonna learn what happens, this information is not present in any text.

ChatGPT-3.5 (Dec 2022)



I put an object on the table and I push the table. What happens to the object? Give your best guess in max 3 sentences.

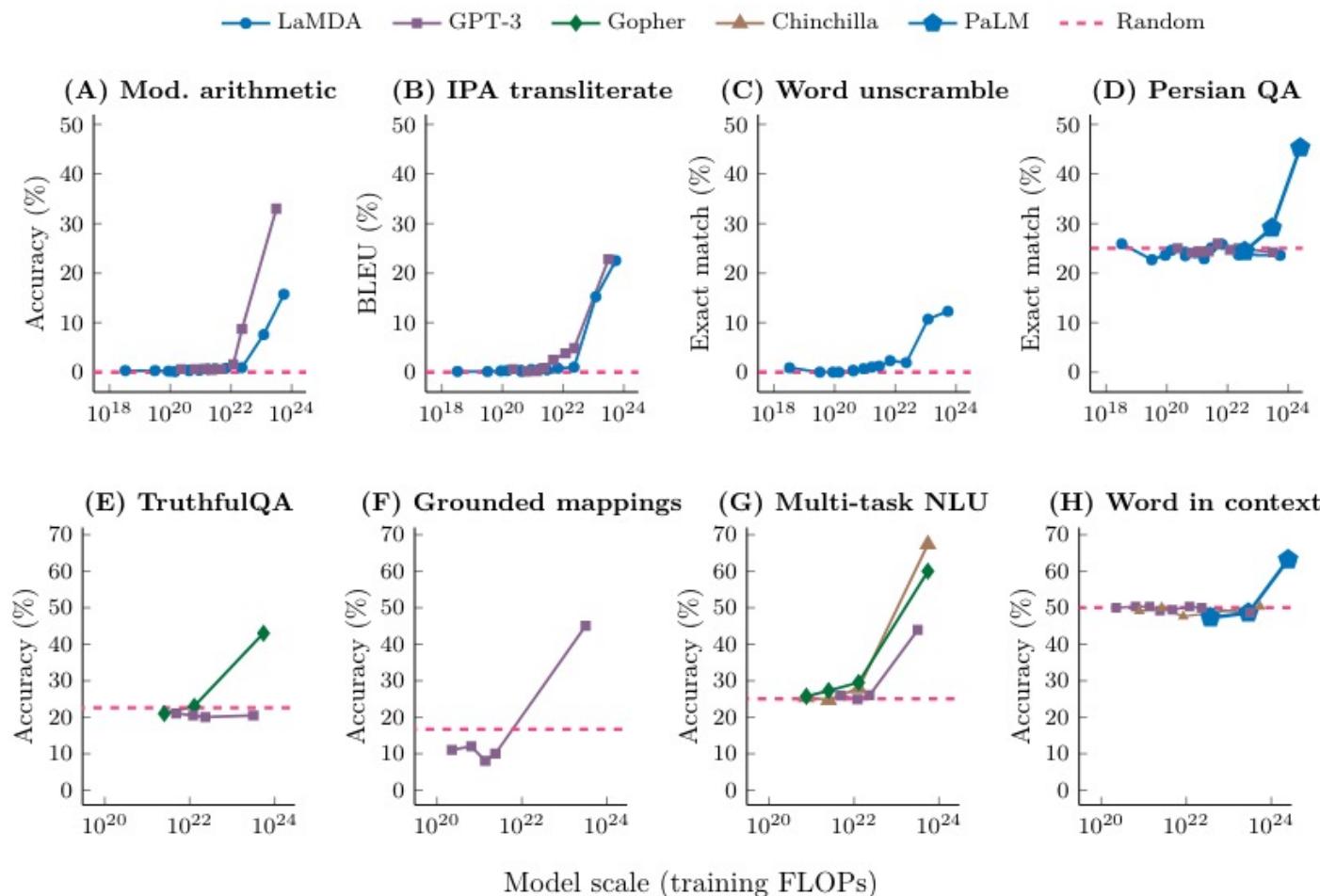


When you push the table, the object on it is likely to move along with the table, assuming the push isn't too violent or the object isn't overly fragile or unstable. The object's movement will be subject to the principles of inertia and friction; it will try to remain in its current state of motion until an external force acts upon it. However, if the push is too forceful, the object might slide or topple over.

Why, scientific challenges

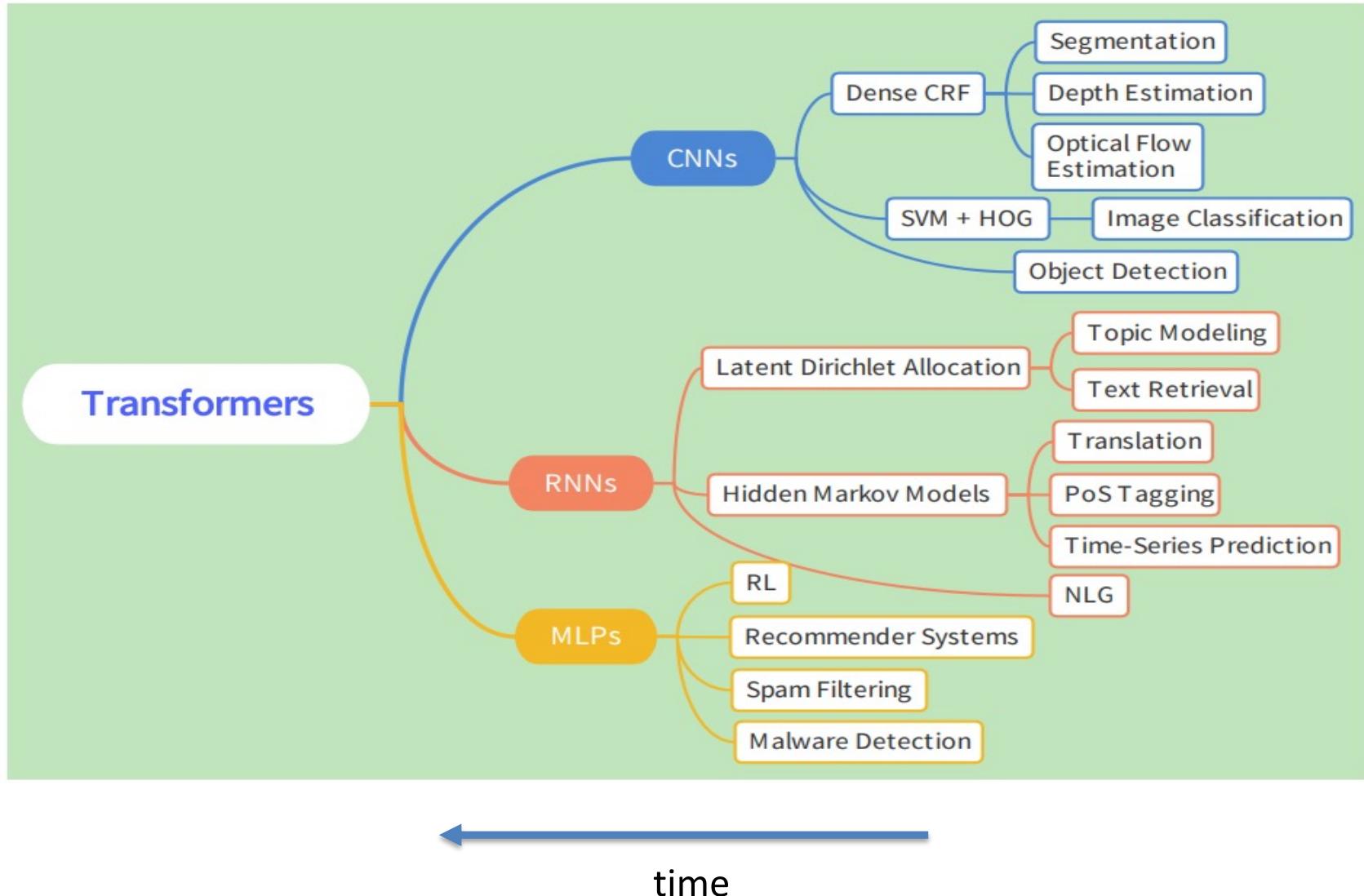
Stochastical parrots
?

- LLM where trained to predict the next token in a sequence of tokens, but
- ... suddenly, if the model is large enough, they can solve other tasks



Transformers for Vision

A general trend in DL Architectures



Overview

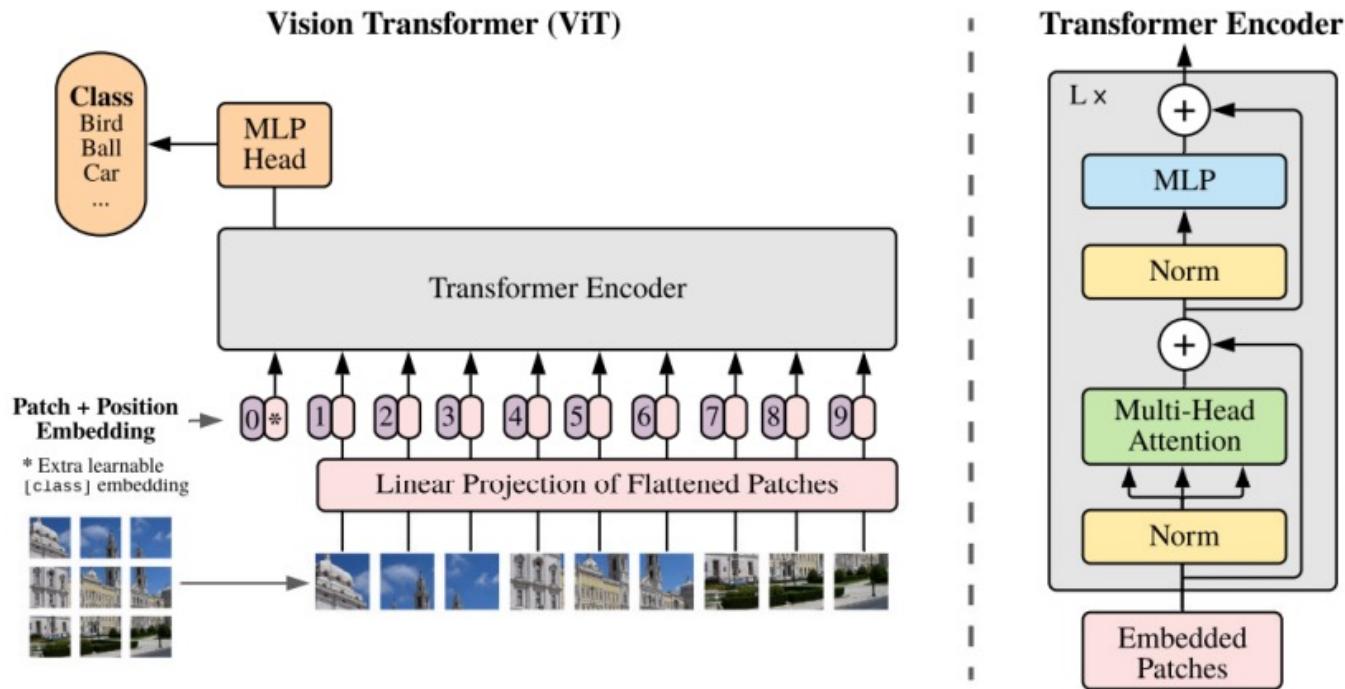


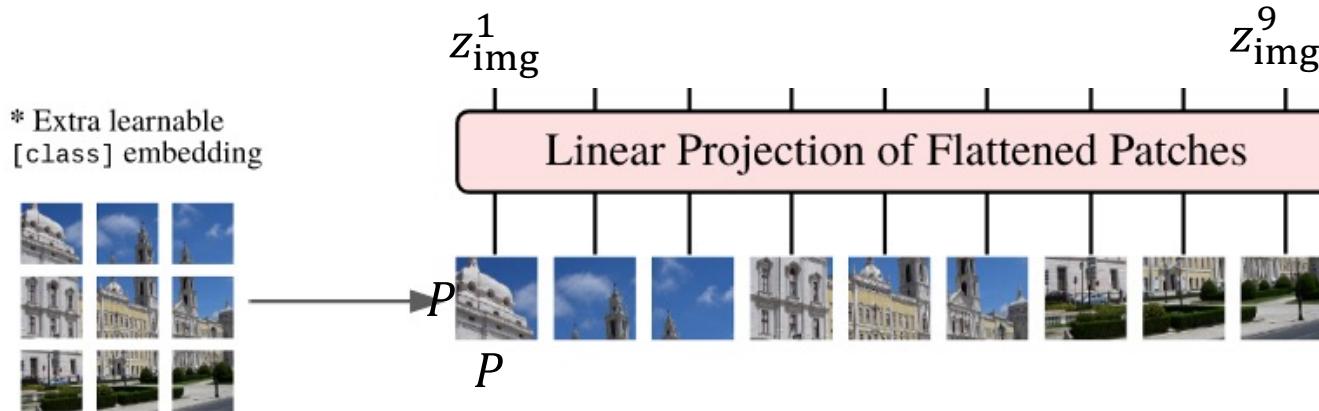
Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable “classification token” to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Question: How to bring Images (W, H, C) to (T, C) Structure*?

* We ignore the Batch B first for notational convenience.

Step 1: Transform image to patches

- We use Patches creates from original images dimension.
- One Patch of Dimension (P, P) corresponds to one token in time Dimension

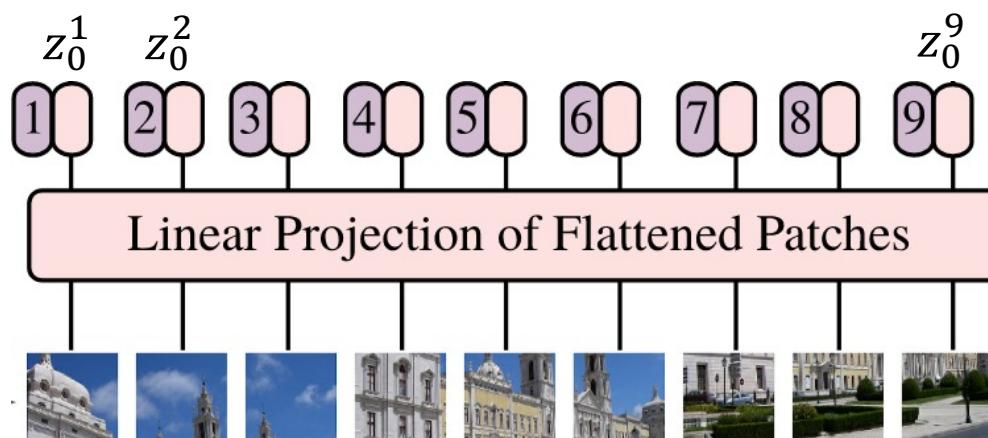


- We use Patches creates from original images dimension.
- One Patch of Dimension (P, P, Col) corresponds to one token in time Dimension.
- Embedding of the images
 - Flatten the patch in a vector x_p^t of size $P^2 \cdot \text{Col}$ and project it to a vector of Dimension C .
 - A image embedding matrix E of shape $(P^2 \cdot \text{Col}, C)$
 - $z_{\text{img}} = (x_p^1 E, x_p^2 E, \dots x_p^T E)$

Step 2: Encoding the position

- To encode the position, we use numbers T (number of patches numbers)
- This is categorical data, hence we use one hot encoding
- We use a position embedding matrix E_{pos}
 - We learn the matrix E_{pos}
- $z_{pos} = E_{pos} \cdot (x_1, x_2, \dots, x_T)$ with one-encoded vectors
- Linear Algebra
 - “Die Spalten einer Matrix sind die Bilder der Einheitsvektoren”
- $z_{pos} = E_{pos}$
- Adding positions embedding z_{pos} and image embedding z_{img}

$$z_0 = z_{img} + z_{pos}$$

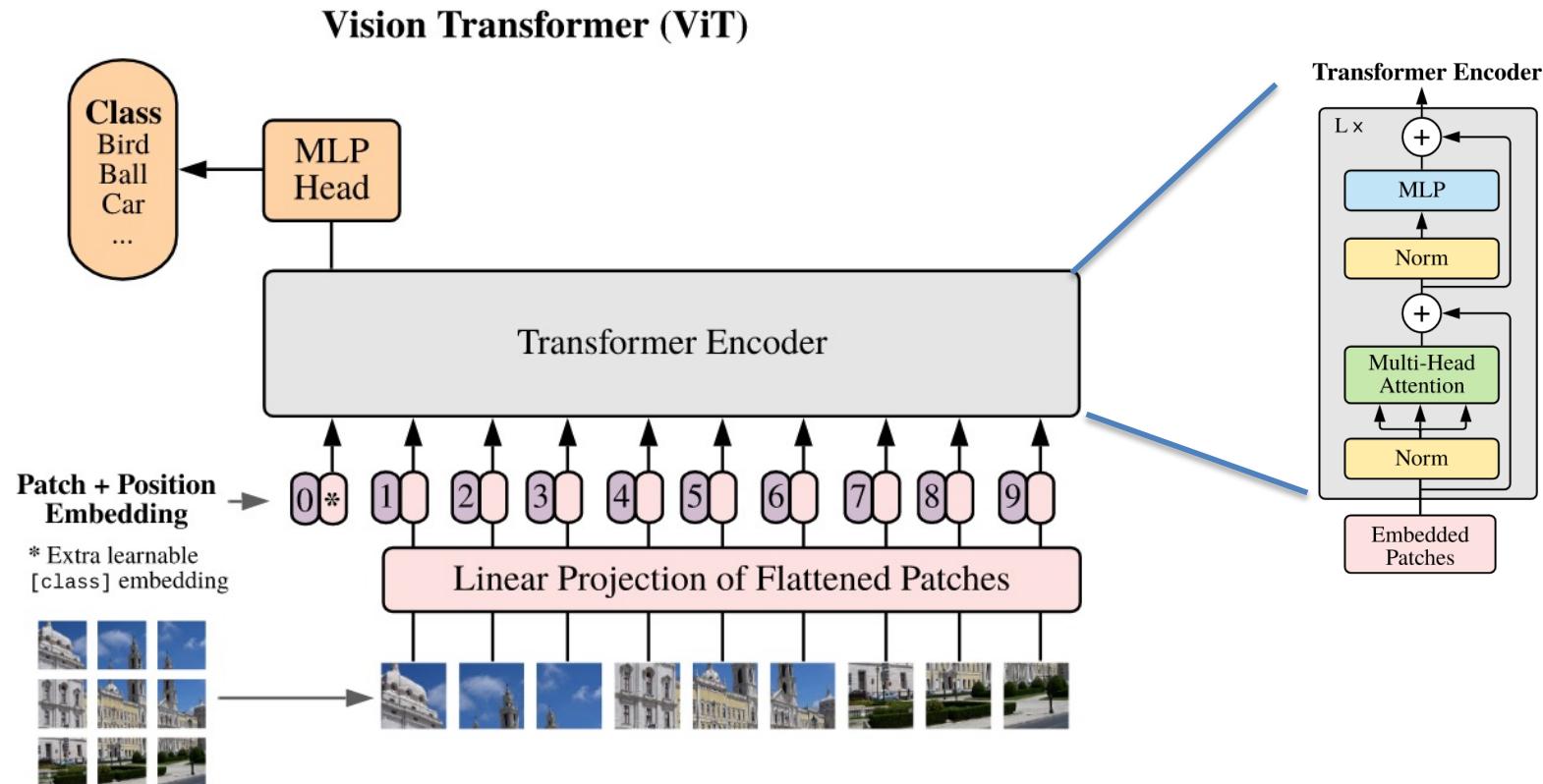


Training / Class Token

- z_0 is a (T, C) tensor and suitable for transformers
- Causal transformers in LLM predict next token in a sequence of tokens
- Here
 - No ordering, no causality
 - We use bidirectional attention influence from “future” is OK
 - Not sensible to predict next patch in a sequence of patches
 - We train using classification
 - Therefore, we use a classification token at position 0 (all other would be also OK) and do MLP with softmax at the end.
- Our start is then

$$z_0 = [x_{\text{class}}, x_p^1 \mathbf{E}, x_p^2 \mathbf{E}, \dots, x_p^T \mathbf{E}] + \mathbf{E}_{pos},$$

The complete Architecture



$$z_0 = [x_{\text{class}}, x_p^1 \mathbf{E}, x_p^2 \mathbf{E}, \dots, x_p^T \mathbf{E}] + \mathbf{E}_{\text{pos}},$$

$$z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \quad \ell = 1 \dots L,$$

$$z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell, \quad \ell = 1 \dots L,$$

$$y = \text{LN}(z_0^L).$$

$$\mathbf{E} \in \mathbb{R}^{(P^2 \cdot \text{Col}) \times C},$$

$$\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(T+1) \times C}.$$

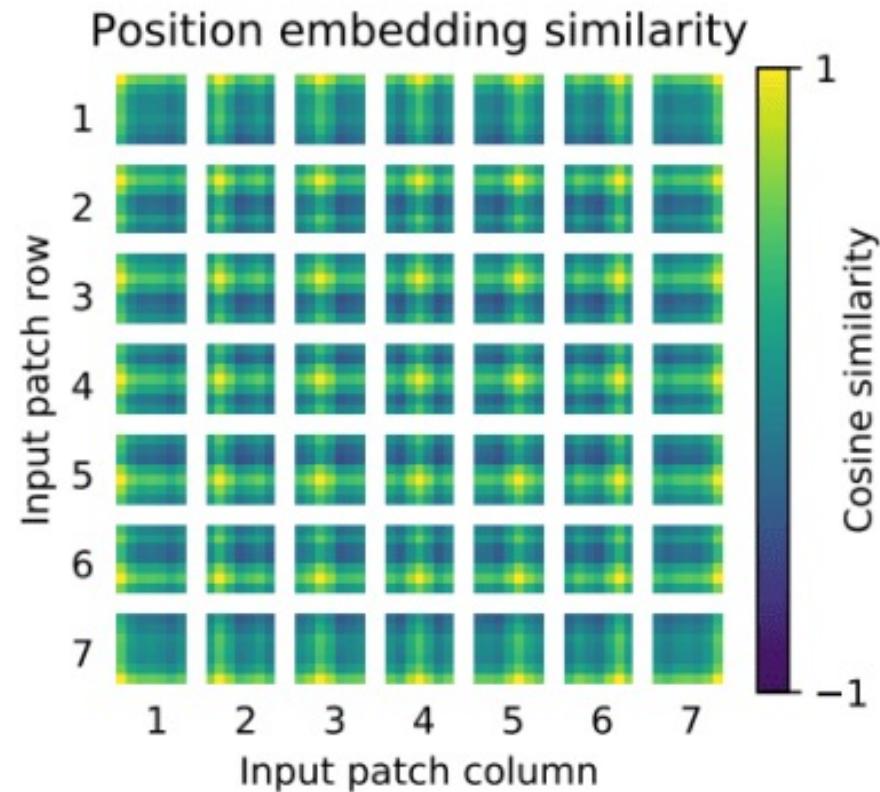
Results

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	—
CIFAR-100	94.55 ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	—
Oxford-IIIT Pets	97.56 ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	—
Oxford Flowers-102	99.68 ± 0.02	99.74 ± 0.00	99.61 ± 0.02	99.63 ± 0.03	—
VTAB (19 tasks)	77.63 ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	—
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

- ViT is trained on datasets with more than 14M images it can approach or beat state-of-the-art CNNs (ResNets or EfficientNets).

Results: Learned Position Embeddings

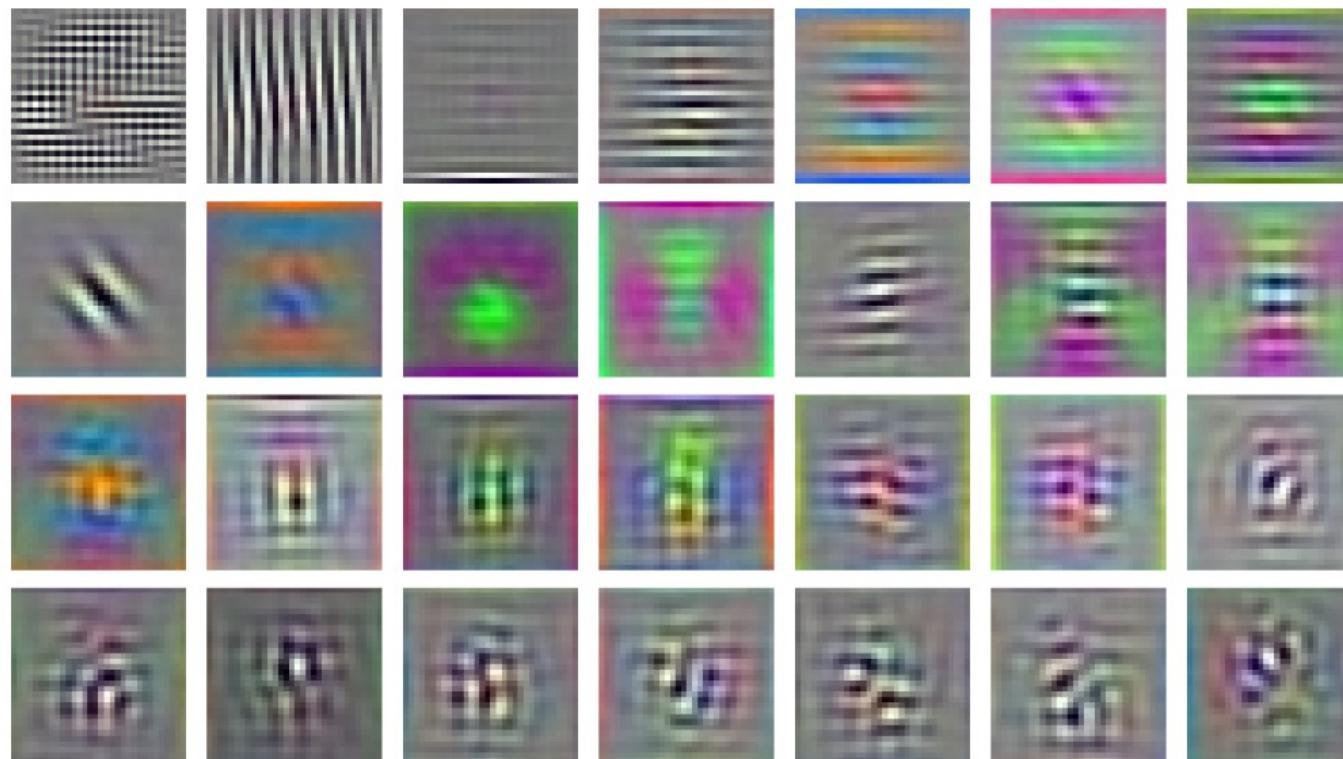


- Similarity of position embeddings of ViT-L/32.
- Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches.
- The first patch (1,1) has high correlation with others which are on same row (1,.) and col(.1)



Result: Learned Weight Embeddings

RGB embedding filters
(first 28 principal components)



Similar to Alex Net, typical patterns are most important for embedding

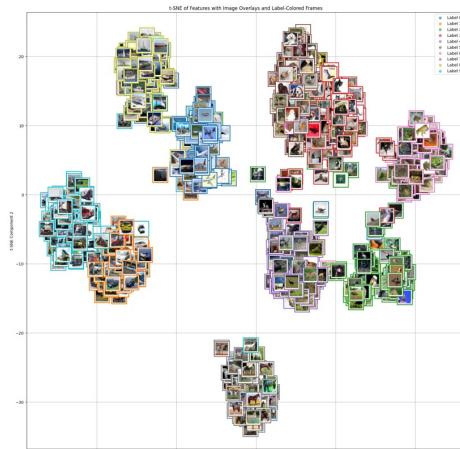
Possible Exercise



- Use a VIT for feature extraction 02_transfer_learning.ipynb

Possible Exercise: Results

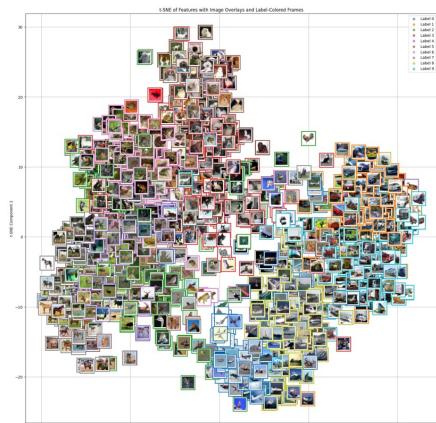
- Use a ViT for feature extraction in NB



Data Set: CIFAR10
Extractor: vit
Training Data: 800
Validation Data: 200
Test Data: 2000

Data	Model	Accuracy
RAW	Random Forest	0.3195
RAW	CNN	0.1280
Features	Random Forest	0.9415
Features	Neural Network	0.9500

- Use a VGG16 for feature extraction (last week)



Data Set: CIFAR10
Extractor: VGG16
Training Data: 800
Validation Data: 200
Test Data: 2000

Data	Model	Accuracy
RAW	Random Forest	0.3195
RAW	CNN	0.3020
Features	Random Forest	0.7495
Features	Neural Network	0.7645

Summary on ViT

- Less inductive bias compared to CNN
- Need much data to learn features
- Good Feature Extractor