Künstliche Intelligenz / Maschinelles Lernen

TECHNISCHE UNIVERSITÄT DARMSTADT

Tiefes Lernen — Deep Learning
Teil 2 - Trainieren von faltenden Neuronalen Netzwerken

Basierende auf Folien von Viktoriia Sharmanska, Geoff Hinton, Fei-Fei Li und viele anderen. Danke fürs Offenlegen ihrer Folien, auch an Binxu Wang für die Transformer Folien MLFS Tutorial Apr. 18th, 2023

Trainieren des AlexNets: Überblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Große Datenmenge: ImageNet
- 90 Epochen auf 2 Nvidia Geforce GTX 580 GPUs bracuth ungefähr 6 Tage (https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/#:~:text=AlexNet%20Training,weight%20decay%200.0005%20is%20used.)
- Zum Vergleich, 90 Epochen ImageNet-1k Training mit ResNet-50 auf einer NVIDIA M40 GPU benötigt ungefähr 14 Tage (You et al., arxiv 1709.05011, 2018)
- Mittels stochastischem Gradientenabstieg und Backpropagation

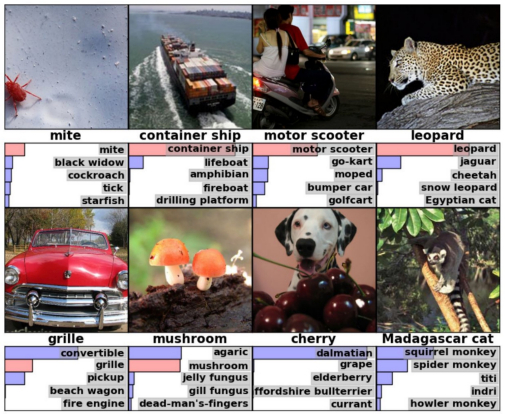
ImageNet

- 15 Millionen Bilder
- 22.000 Kategorien / Labels
- Bilder sind aus dem Internet
- Menschen haben die Bilder annotiert (Amazons Mechanical Turk)
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)
    - 1.000 Kategorien
    - 1.2 Millionen Bilder zum Trainieren (~1000 pro Kategorie)
    - 50,000 Bilder zur Validierung
    - 150,000 Bilder zum Testen
- RGB Bilder; Durschnittsbild wurde von allen Bildern abgezogen
- Unterschiedliche Auflösungen, aber AlexNet skaliert die Bilder auf 256x256

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
    - 1.2 M train
    - 100k test.



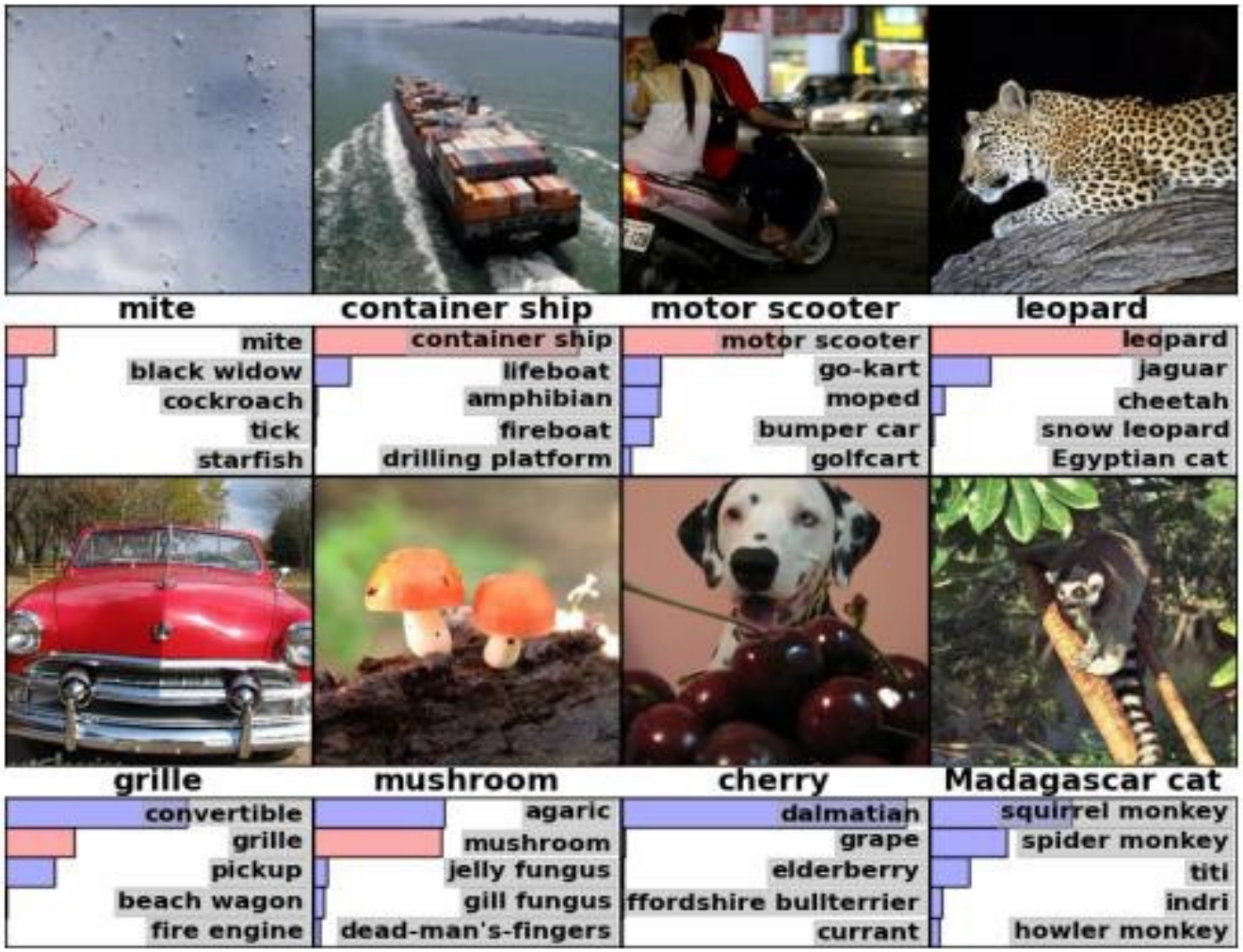
ImageNet: Klassifikationaufgabe

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine Vorhersage pro Bild (Top-1 error)
- Sage 5 Labels pro Bild vorher (Top-5 error)

TECHNISCHE UNIVERSITÄT DARMSTADT

ImageNet: Ergebnisse von AlexNet



AlexNet: Suche von ähnlichen Bildern

Anfragebild

Sechs Trainingsbilder, die zu Merkmalsvektoren in der letzten, unbeobachteten Schicht führen, die den kleinsten Euklidischen Abstand zum Anfragebild hatten

Das Trainieren von faltenden Neuronales Netzwerken (CNNs) im Detail

TECHNISCHE UNIVERSITÄT DARMSTADT

- Training
  - Stochastischer Gradientenabstieg
  - Backpropagation
  - Initialisierung
- Vermeidung von Überanpassung
  - Dropout Regularisierung
  - Datenaugmentierung
- Fine-tuning
- Visualisierung von CNNs

TECHNISCHE UNIVERSITÄT DARMSTADT

Training CNNs

- Stochastischer Gradientenabstieg
- Backpropagation
- Initialisierung

Stochastic gradient descent (SGD)

TECHNISCHE UNIVERSITÄT DARMSTADT

(Mini-batch) SGD

Initialisierung der Parameter

Gehe über alle Daten (mehrmals):

- **Sample** einen (Batch von) Datenpunkt(en)
- **Vorwärtspropagierung** der Daten durch das Netzwek, berechne die Klassifikationsgüte / den Verlust (loss).
- **Rückwärtspropagierung** des Gradientens der Parameter bezüglich der Verlustsfunktion
- **Update** der Parameter mittels des Gradientens

TECHNISCHE UNIVERSITÄT DARMSTADT

Stochastic gradient descent (SGD)

TECHNISCHE UNIVERSITÄT DARMSTADT

(Mini-batch) SGD

Initialisierung der Parameter **zufällig aber auf intelligente Art**

Gehe über alle Daten (mehrmals):

- **Sample** einen (Batch von) Datenpunkt(en)
- **Vorwärtspropagierung** der Daten durch das Netzwek, berechne die Klassifikationsgüte / den Verlust (loss). **Z.B.**

$$E = \frac{1}{2}(y_{predicted} - y_{true})^2$$

- **Rückwärtspropagierung** des Gradientens der Parameter bezüglich der Verlustsfunktion
- **Update** der Parameter mittels des Gradientens. **SGD**

$$w^{t+1} = w^t - \alpha \cdot \frac{dE}{dw}(w^t)$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Backpropagation

TECHNISCHE UNIVERSITÄT DARMSTADT

Backpropagation ist die rekursive Anwendung der Kettenregel zur Differenzierung entlang des Berechnungsgraphens des Netzwerkes zur Berechnung des Gradientes der Verlustfunktion nach alles Variablen, auch der Zwischenergebnisse

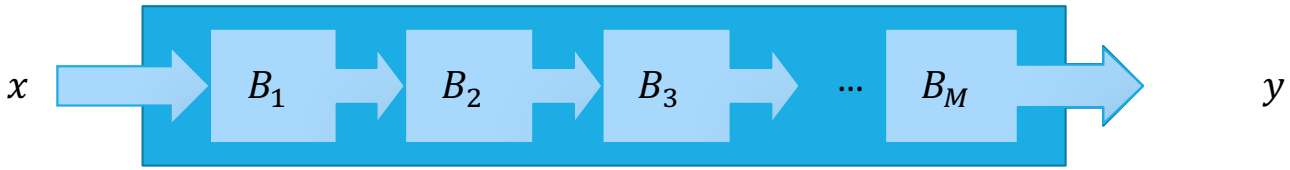
Die Lego-Architektur von tiefen Netzwerken spiegelt sich auch auf im Trainieren wider

TECHNISCHE UNIVERSITÄT DARMSTADT

Backpropagation

Implementierungen sind modular aufgebaut. Knoten/ Leogbausteine implementieren den Vorwärts- und Rückwärtspropagierungen

Sequential brick



Propagation

- Apply propagation rule to $B_1, B_2, B_3, \dots, B_M$.

Back-propagation

- Apply back-propagation rule to $B_M, \dots, B_3, B_2, B_1$.

Backpropagation

TECHNISCHE UNIVERSITÄT DARMSTADT

Letzte Schicht implementiert die Klassifikation

Square loss brick

Propagation

$$E = y = \frac{1}{2}(x - d)^2$$

Back-propagation

$$\frac{\partial E}{\partial x} = (x - d)^T \frac{\partial E}{\partial y} = (x - d)^T$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Backpropagation

TECHNISCHE UNIVERSITÄT DARMSTADT

Typische Legobausteine

Loss bricks

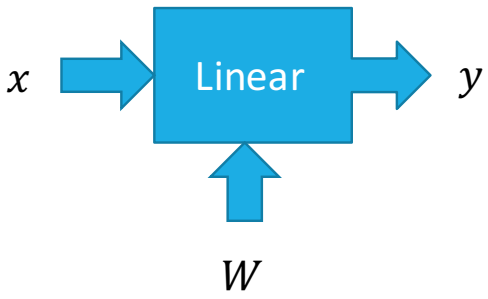
		Propagation	Back-propagation
Square		$y = \frac{1}{2}(x-d)^2$	$\frac{\partial E}{\partial x} = (x-d)^T \frac{\partial E}{\partial y}$
Log	$c = \pm 1$	$y = \log(1+e^{-cx})$	$\frac{\partial E}{\partial x} = \frac{-c}{1+e^{cx}} \frac{\partial E}{\partial y}$
Hinge	$c = \pm 1$	$y = \max(0, m-cx)$	$\frac{\partial E}{\partial x} = -c \ \mathbb{I}\{cx < m\} \frac{\partial E}{\partial y}$
LogSoftMax	$c = 1 \dots k$	$y = \log(\sum_k e^{x_k}) - x_c$	$\left[\frac{\partial E}{\partial x}\right]_s = (e^{x_s}/\sum_k e^{x_k} - \delta_{sc}) \frac{\partial E}{\partial y}$
MaxMargin	$c = 1 \dots k$	$y = \left[\max_{k \neq c}\{x_k + m\} - x_c\right]_+$	$\left[\frac{\partial E}{\partial x}\right]_s = (\delta_{sk^*} - \delta_{sc}) \ \mathbb{I}\{E > 0\} \frac{\partial E}{\partial y}$

Backpropagation

TECHNISCHE UNIVERSITÄT DARMSTADT

Vollständig-verbundene Schichte, Faltungsschichten (Skalarprodukt)

Linear brick



Propagation

$$y = Wx$$

Back-propagation

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial y} W$$

$$\frac{\partial E}{\partial W} = x \frac{\partial E}{\partial y}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Nicht-lineare Aktivierungen

Activation function brick

Propagation

$$y_s = f(x_s)$$

Back-propagation

$$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s f'(x_s)$$

Backpropagation

TECHNISCHE UNIVERSITÄT DARMSTADT

Typsiche Aktivierungsfunktionen

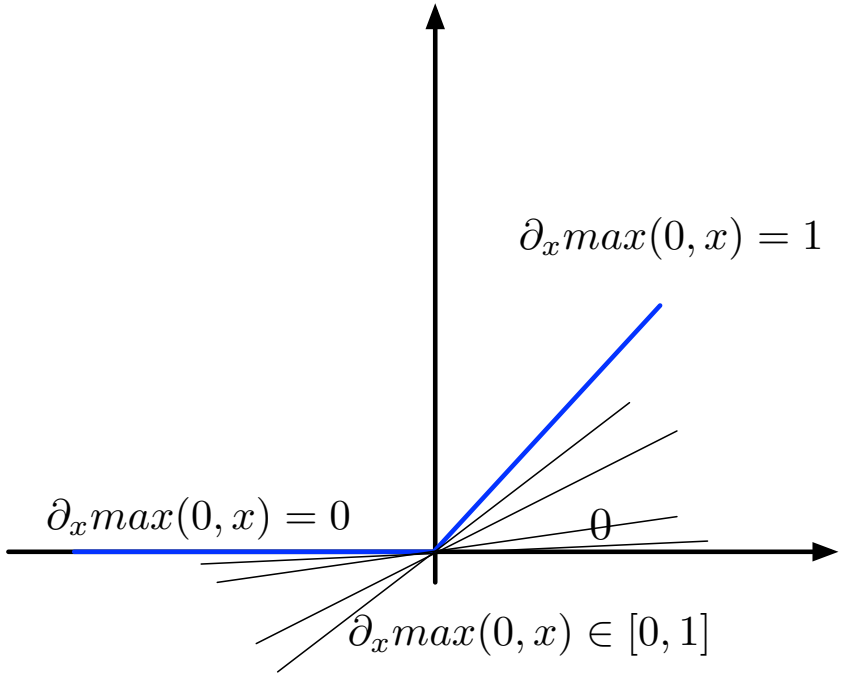
Activation functions

	Propagation	Back-propagation
Sigmoid	$y_s = \frac{1}{1+e^{-x_s}}$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \frac{1}{(1+e^{x_s})(1+e^{-x_s})}$
Tanh	$y_s = \tanh(x_s)$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \frac{1}{\cosh^2 x_s}$
ReLu	$y_s = \max(0, x_s)$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \mathbb{I}\{x_s > 0\}$
Ramp	$y_s = \min(-1, \max(1, x_s))$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \mathbb{I}\{-1 < x_s < 1\}$

TECHNISCHE UNIVERSITÄT DARMSTADT

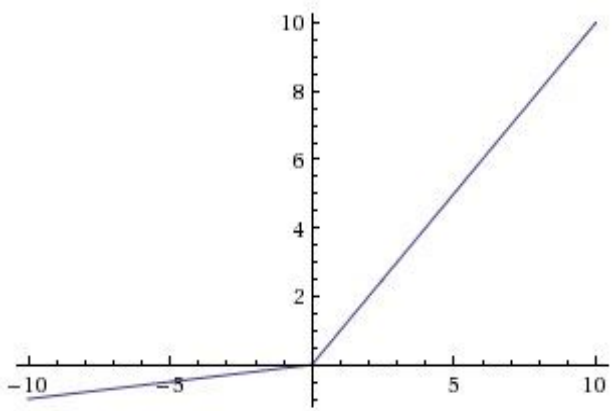
Subgradienten

ReLU Gradient ist für x=0 nicht definiert, daher benutzt man Subgradient



Leaky ReLU

In der Praxis gerne benutzt: *Leaky ReLU*, f(x) = max(0.01x, x)
Das vermeidet, dass der Gradient "saturiert", also überall gleich aussieht



Leaky ReLU

$$f(x) = \max(0.01x, x)$$

Training CNNs

- ☑ Stochastischer Gradientenabstieg
- ☑ Backpropagation
- ☐ Initialisierung

Stochastic gradient descent (SGD)

TECHNISCHE UNIVERSITÄT DARMSTADT

(Mini-batch) SGD

Initialisierung der (Filter-)Gewichte

- nicht mit 0 initialisieren
- Nicht alle mit dem selben Wert initialisieren
- sample gleichmäßig U[-b,b] um Null herum oder von der Normalverteilung

Abnahme (decay) der Schrittweite α

$$w^{t+1} = w^t - \alpha \cdot \frac{dE}{dw}(w^t)$$

Je näher wir am der “Lösung” sind, desto kleiner die Schritteweite

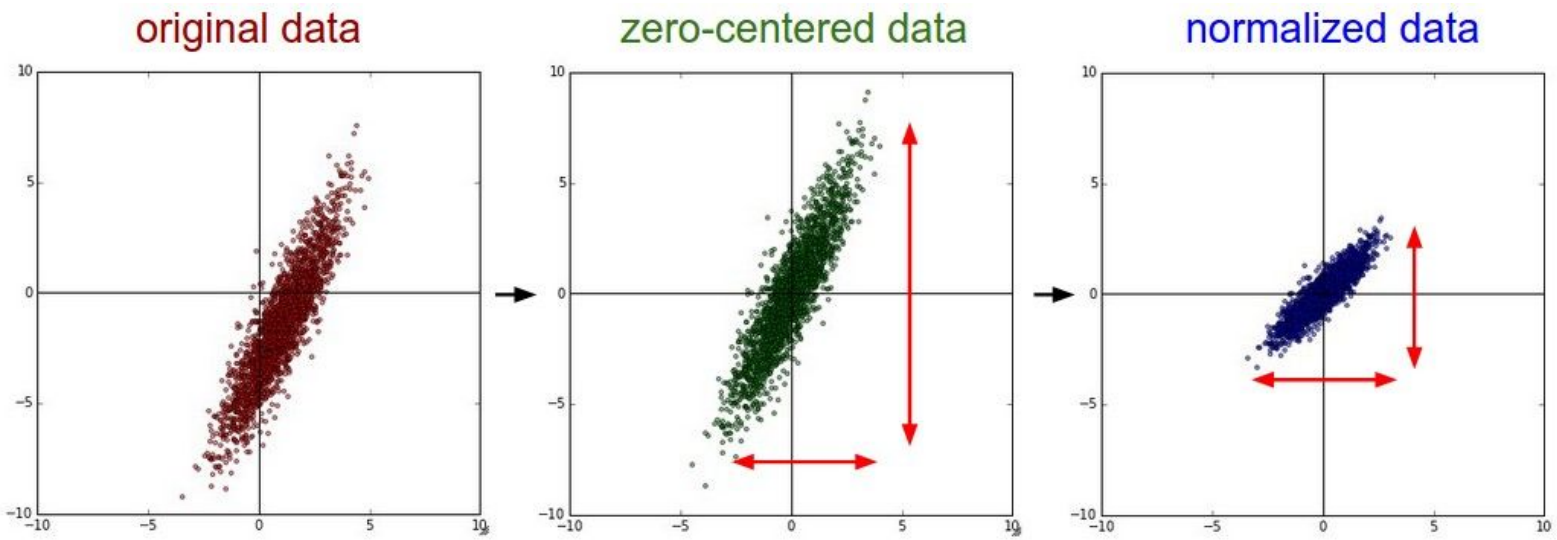
- Fange mit einer gr0ßen Schrittweite an (z.B. 0.1)
- Behalte diese bei, bis sich der Fehler auf der Validierungsmenge nicht mehr verbessert
- Halbiere die Schrittweite und verfolge wie zuvor

TECHNISCHE UNIVERSITÄT DARMSTADT

Stochastic gradient descent (SGD)

TECHNISCHE UNIVERSITÄT DARMSTADT

Normalisierung der Daten



Bei Bildern: Ziehe das Durchschnittsbild von allen Trainingsbildern ab.

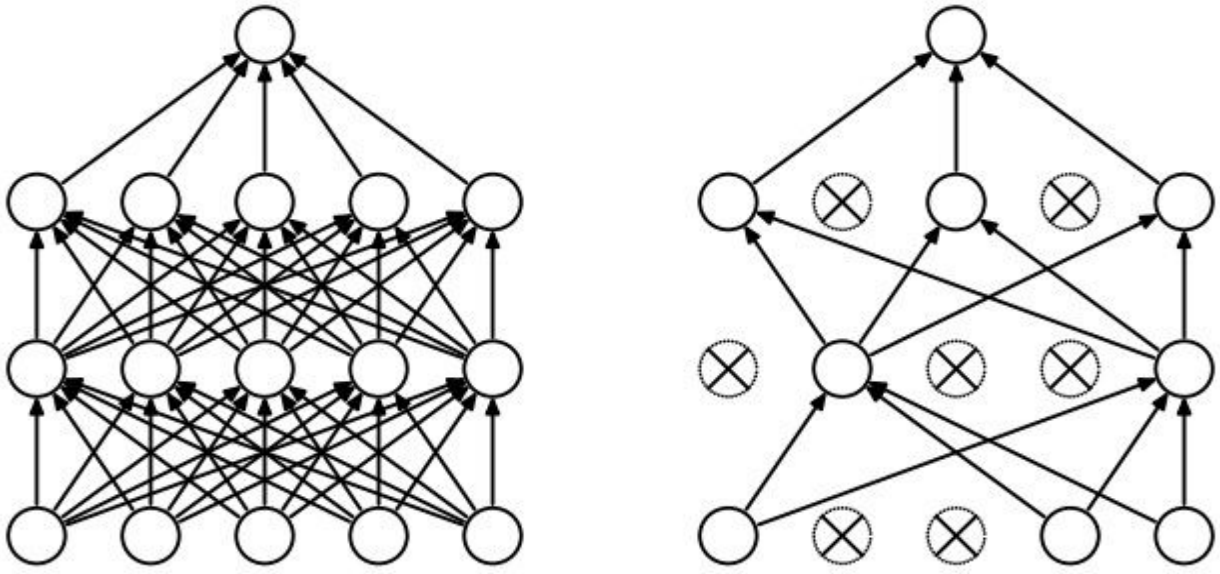
TECHNISCHE UNIVERSITÄT DARMSTADT

Vermeide Überanpassung

- Dropout regularization
- Data augmentation

Regularisierung: DropOut

Dropout: „setze die Eingabe für einige Neuronen auf 0 (mit Wahrscheinlichkeit 0.5)



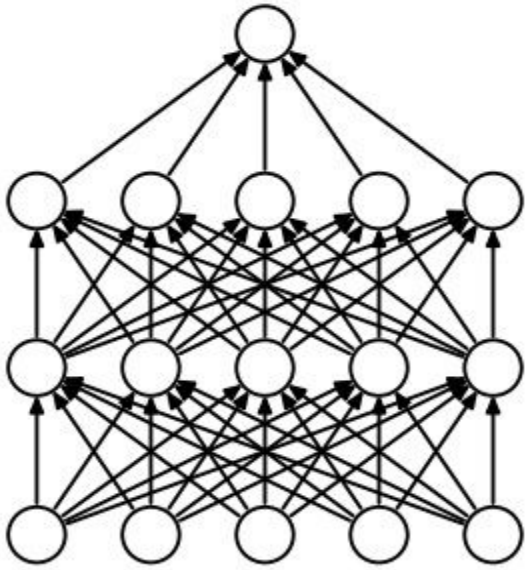
(a) Standard Neural Net

(b) After applying dropout.

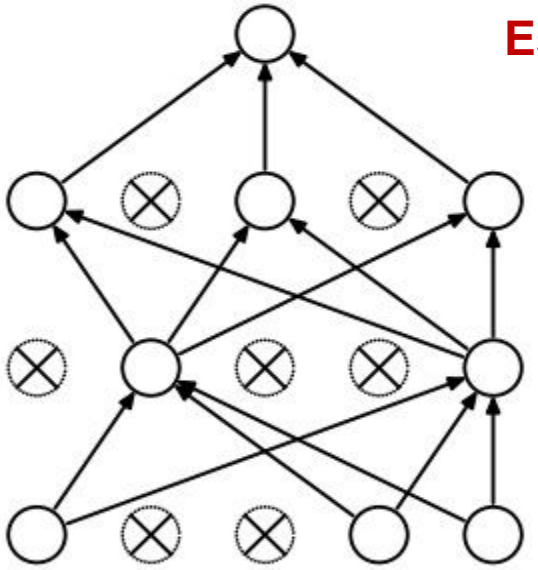
[Srivastava et al., 2014]

Regularisierung: DropOut

Dropout: „setze die Eingabe für einige Neuronen auf 0 (mit Wahrscheinlichkeit 0.5). Die „dropped out“ Neuronen tragen nicht zur Vorhersage bei und werden auch nicht bei Backpropagation beachtet.



(a) Standard Neural Net



(b) After applying dropout.

Es wird für jede Eingabe eine andere Netzwerk-Architektur benutzt. (Ensembles)

Zur Vorhersage auf der Testmenge benutze den Durschnitt aller Modelle

[Srivastava et al., 2014]

Regularisierung: DropOut

TECHNISCHE UNIVERSITÄT DARMSTADT

Verringert komplexe Ko-Adaptionen von Neuronen; ein einzelnes Neuron kann sich nicht mehr auf andere andere Neuronen verlassen

Jedes Neuron muss robuste Merkmale lernen, die in einem komplexen zusammenspiel mit anderen Neuronen gut funktionieren

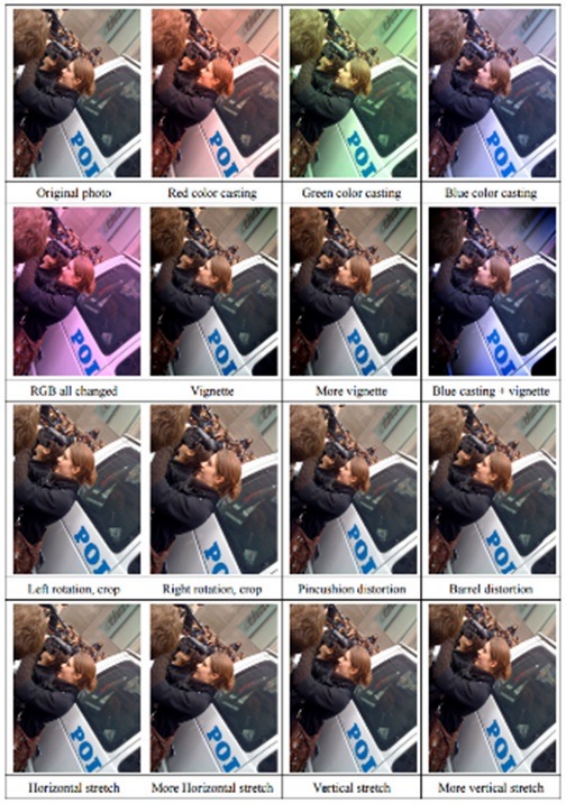
Ohne DropOut zeigen CNNs sarke Überanpassung

Aber DropOut verdoppelt (Pi*Daumen) den Trainingsaufwand

Alternative: Standard Regularisierungen wie z.B. L2-Regularisierung

Regularisierung: Datenaugmentierung

Die einfachste Art der Regularisierung ist die Datenaugmentierung. Wir reichern die Trainingsdaten mit vielen Varianten der Originaldaten an.



Typsicherweise

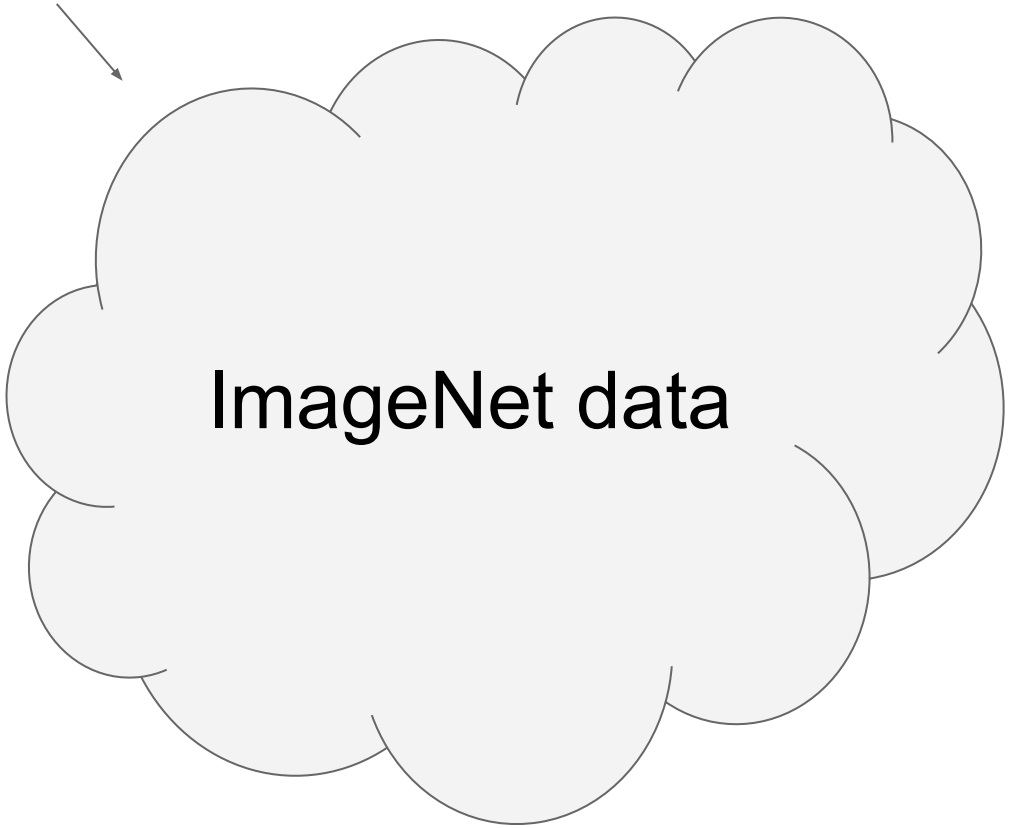
- horizontale Spiegelungen
- Zufällige Ausschnitte
- Änderungen der RGB_Werte
- Bildtransformatinoen

Fine-Tuning

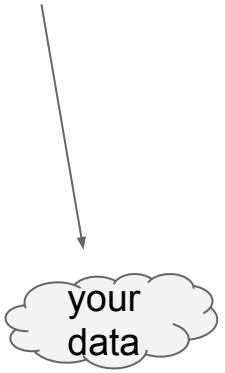
Fine-Tuning

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Train on ImageNet



2. Finetune network on your own data



TECHNISCHE UNIVERSITÄT DARMSTADT

Fine-Tuning

Transfer Learning with CNNs

1. Train on ImageNet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e. swap the Softmax layer at the end

3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

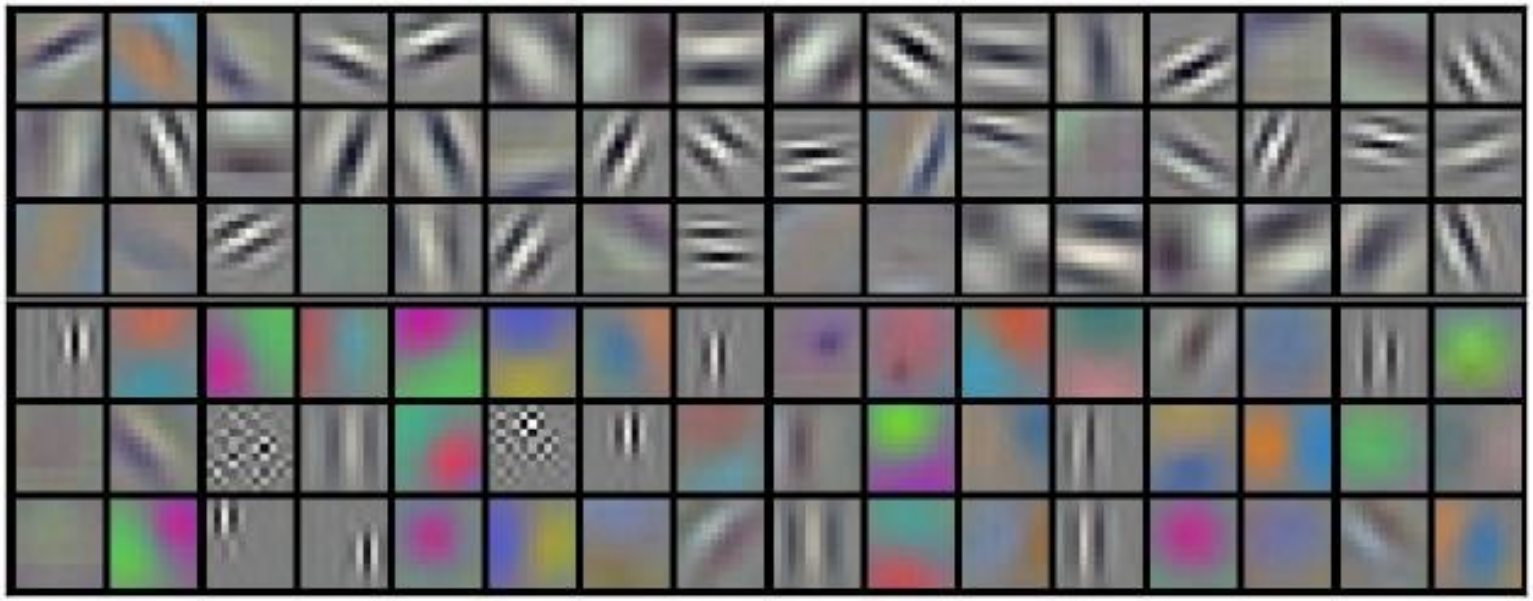
retrain bigger portion of the network, or even all of it.

Im Internet findet man viele pre-trained Modelle

Visualisierung

Erste Faltungsschicht

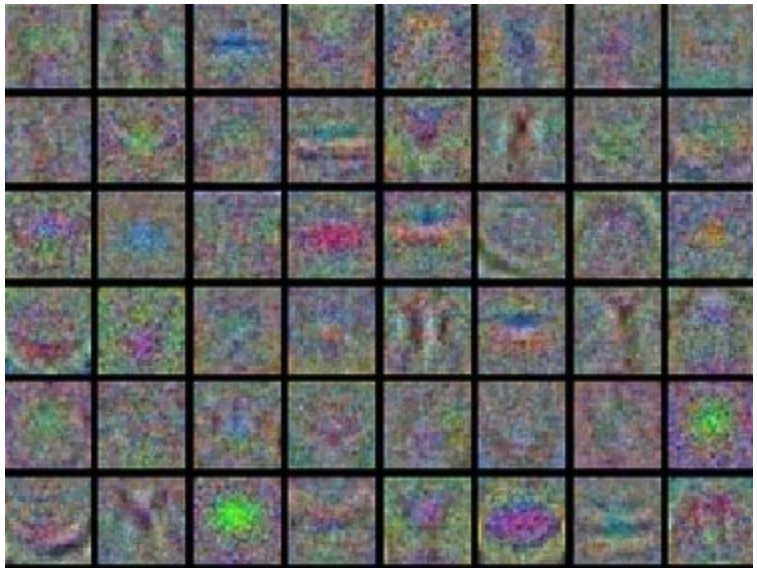
AlexNet

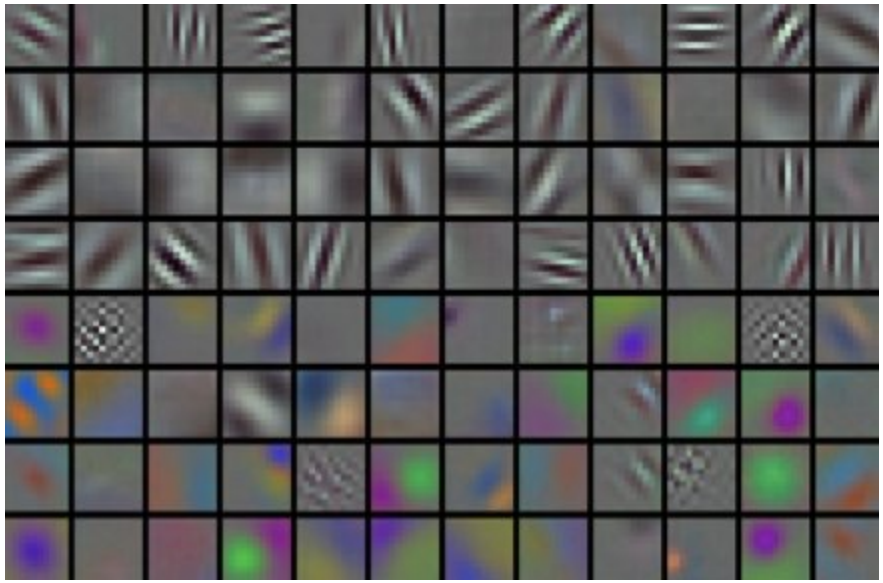


96 Faltungskerne (11×11×3) gelernt durch die erste Faltungsschicht auf 227×227×3 Eingabebildern.

Erste Faltungsschicht

AlexNet

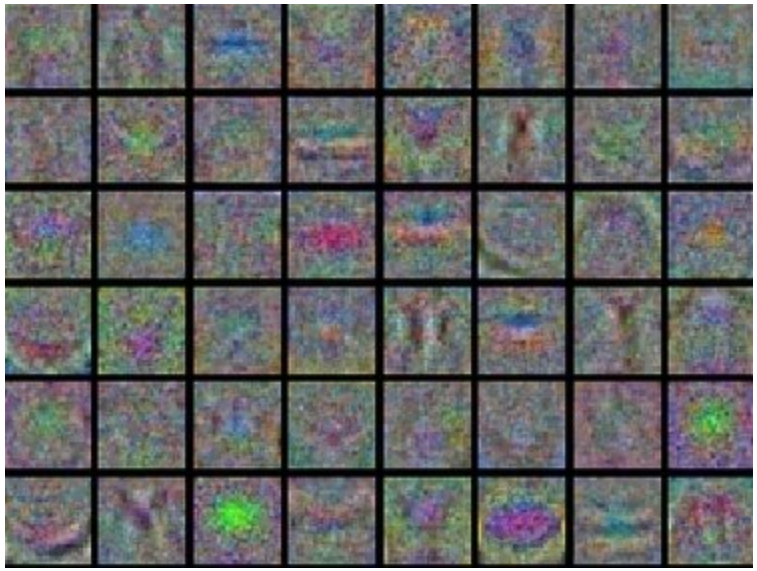


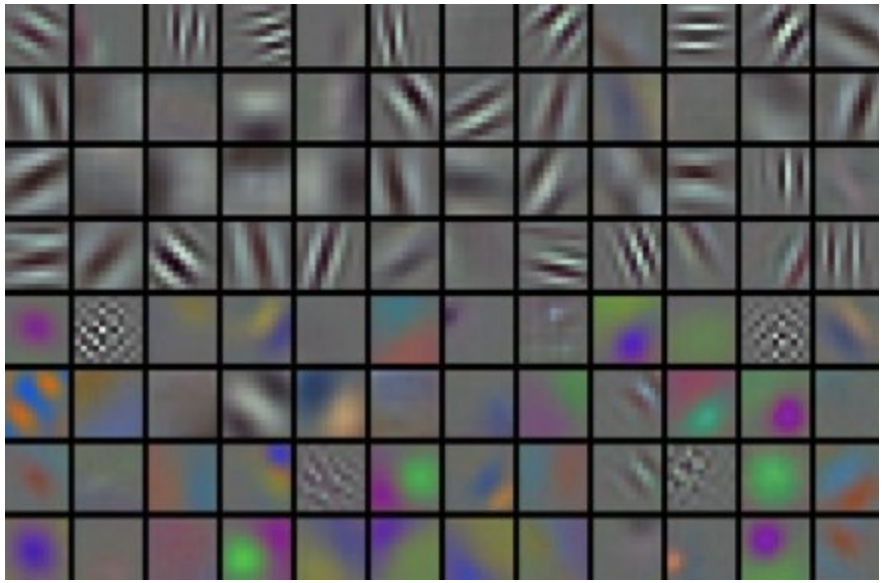


Welches Netzwerk ist besser? Links oder rechts?

Erste Faltungsschicht

AlexNet



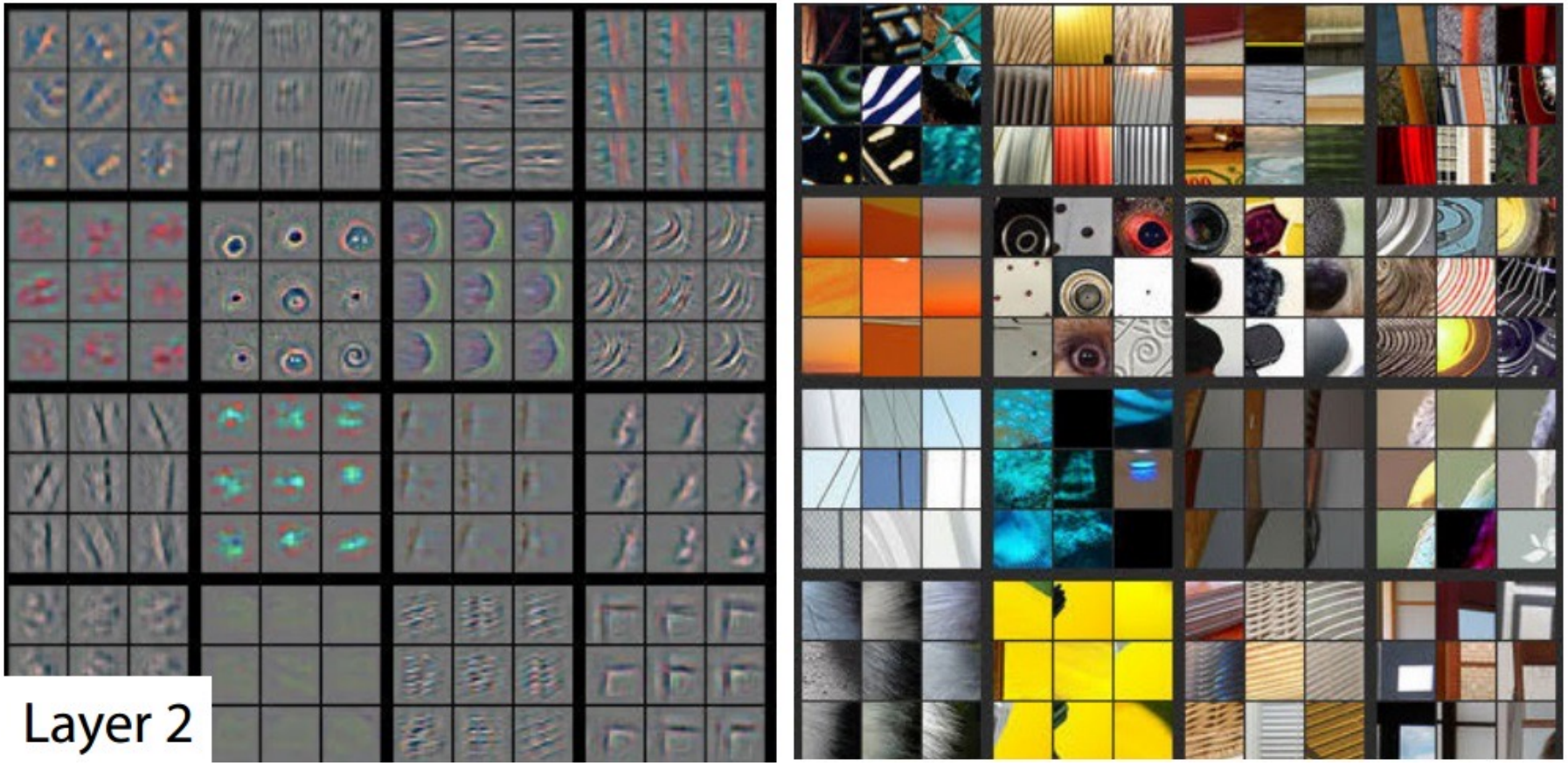


Links: vielleicht nicht konvergiert? Falsche Schrittweiter? Nicht normalisiert?

Rechts: Schön, glatt, unterschiedliche Filter. Lernen scheint gut funktioniert zu haben

Visualisierung von Faltungsnetzwerken

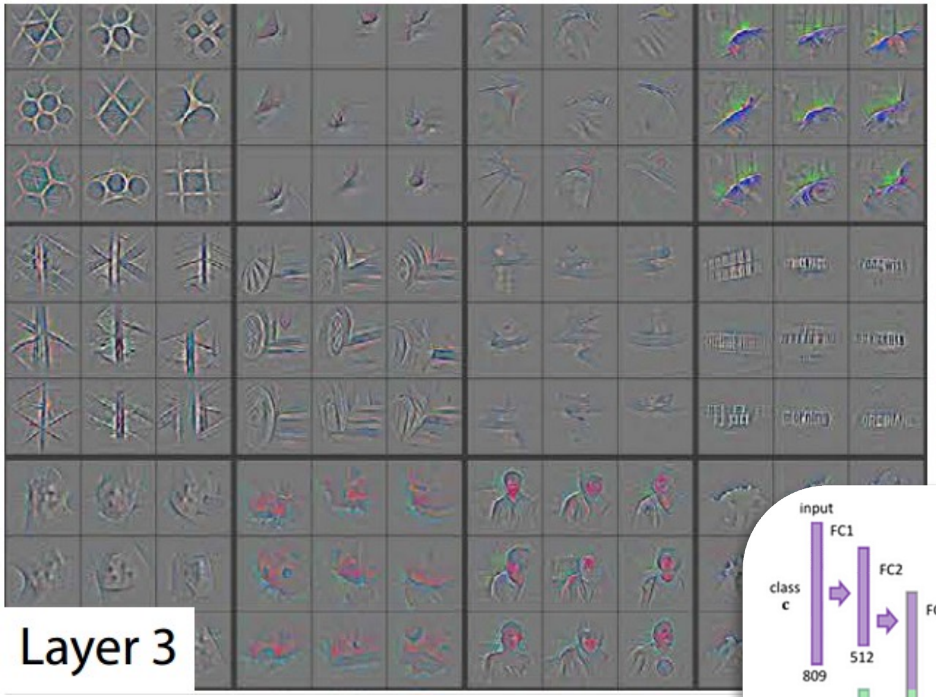
TECHNISCHE UNIVERSITÄT DARMSTADT

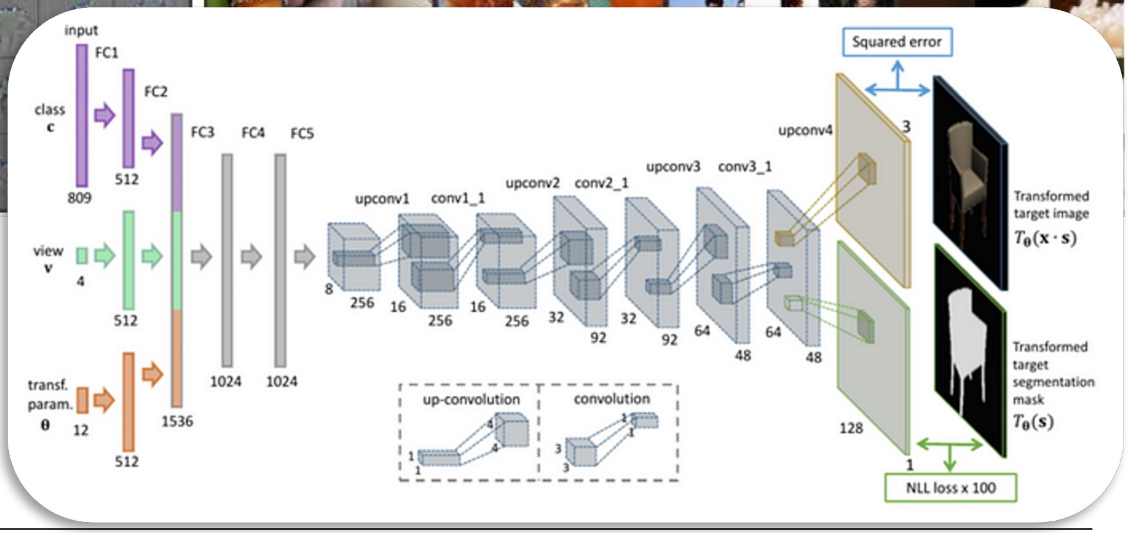


Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Idea: map activations of neurons back to the input pixel space

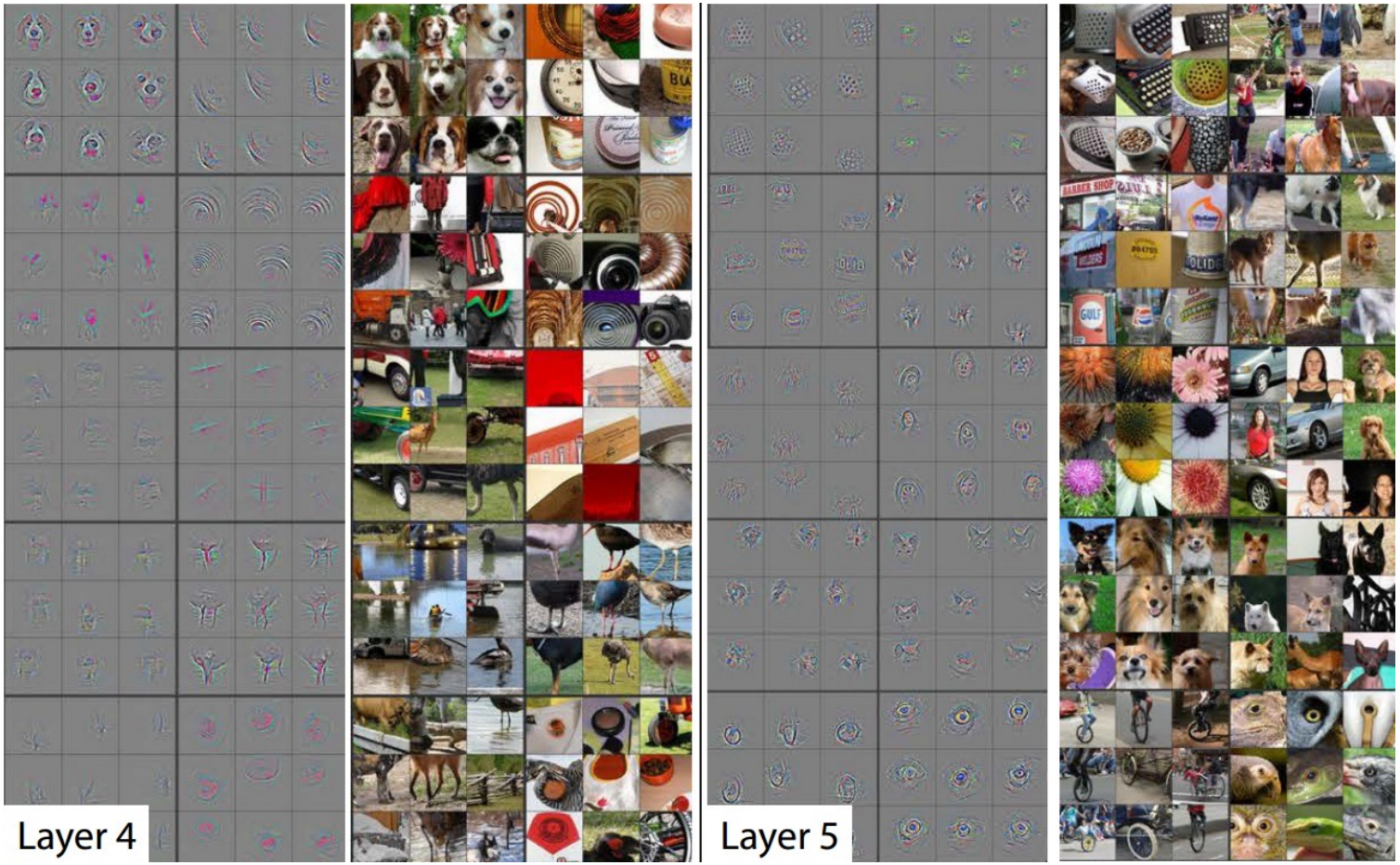
Visualisierung von Faltungsnetzwerken





Use e.g. a Deconvolutional Network, i.e., a CNN in reverse

Visualisierung von Faltungsnetzwerken

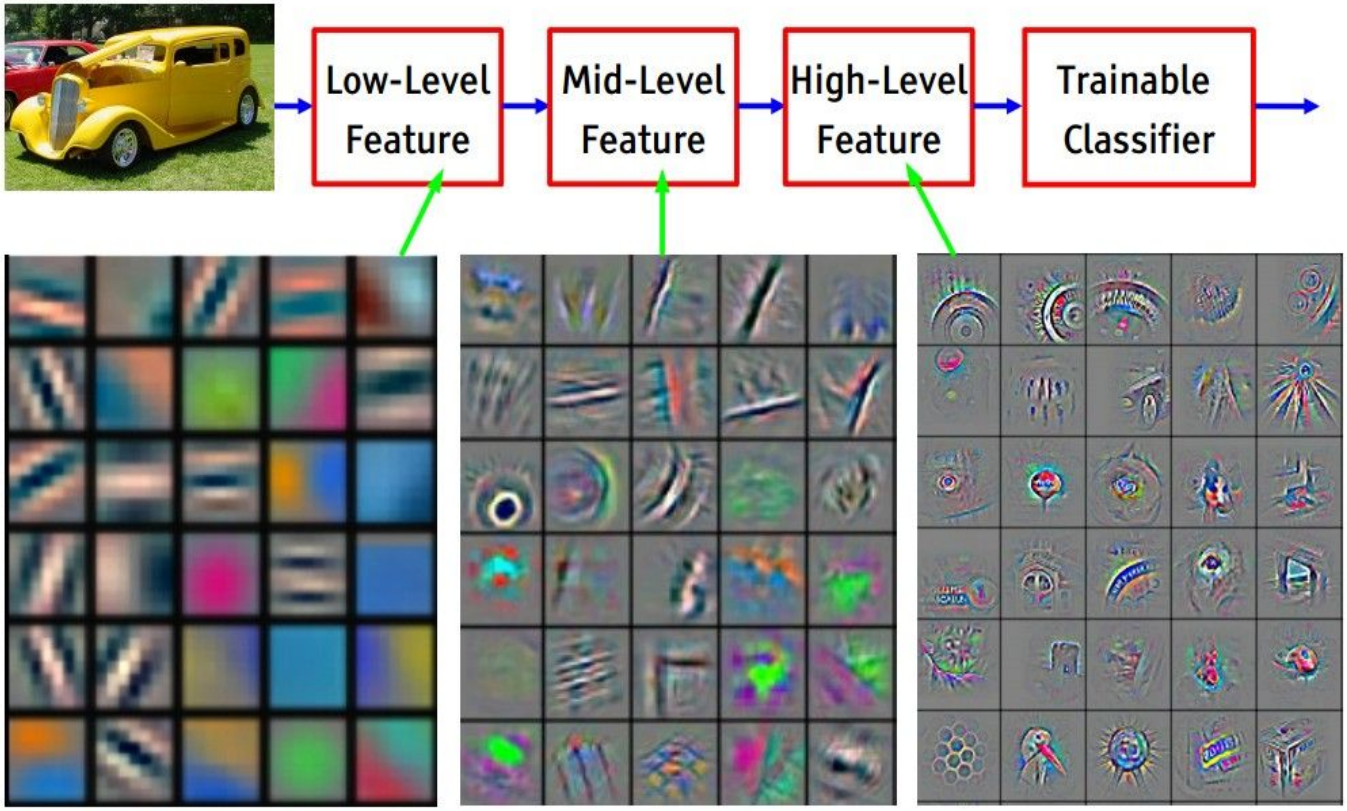


Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Visualisierung von Faltungsnetzwerken

TECHNISCHE UNIVERSITÄT DARMSTADT

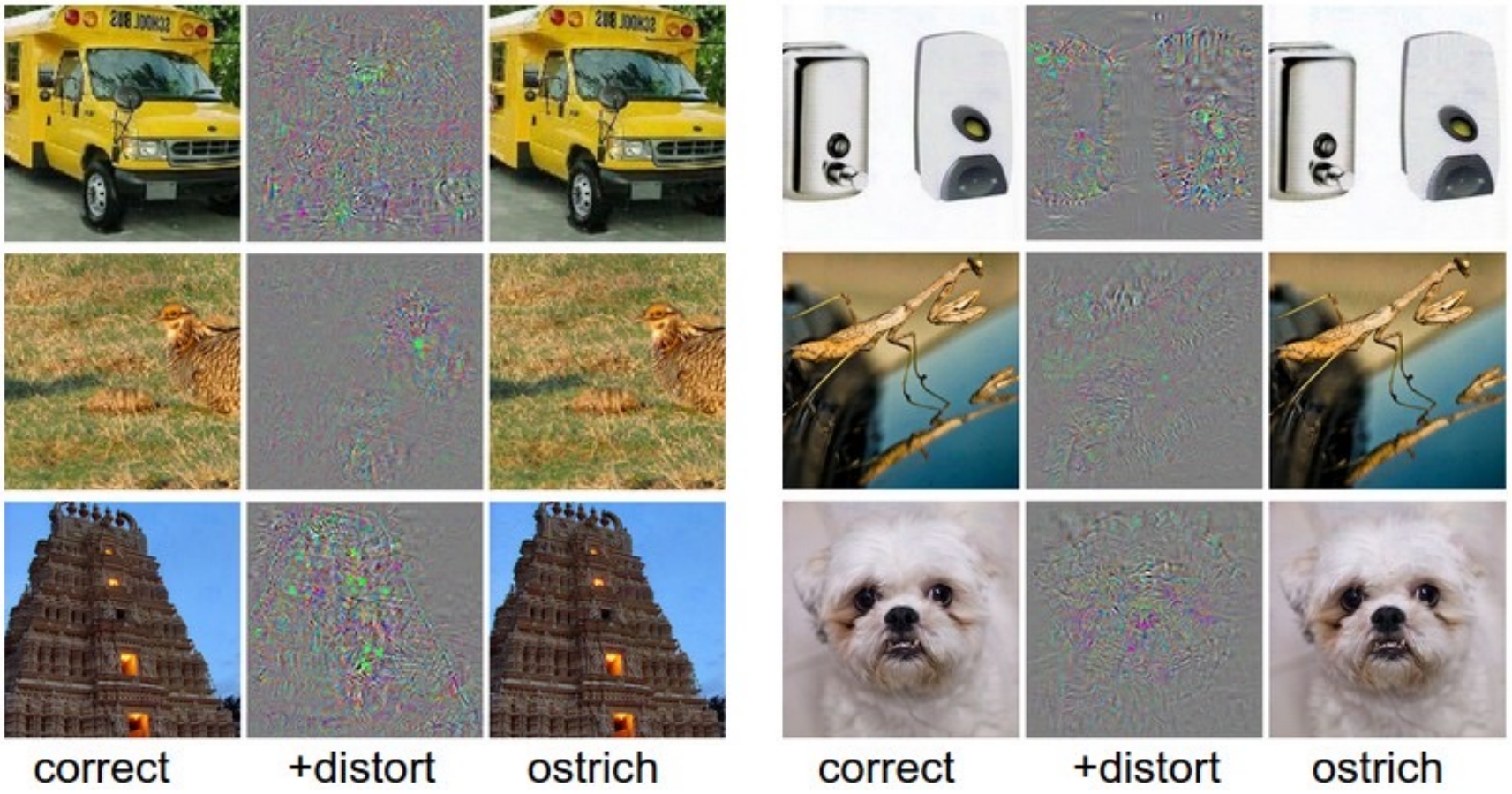
Die Schichten lernen hierarchische Repräsentationen der Eingabedaten



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

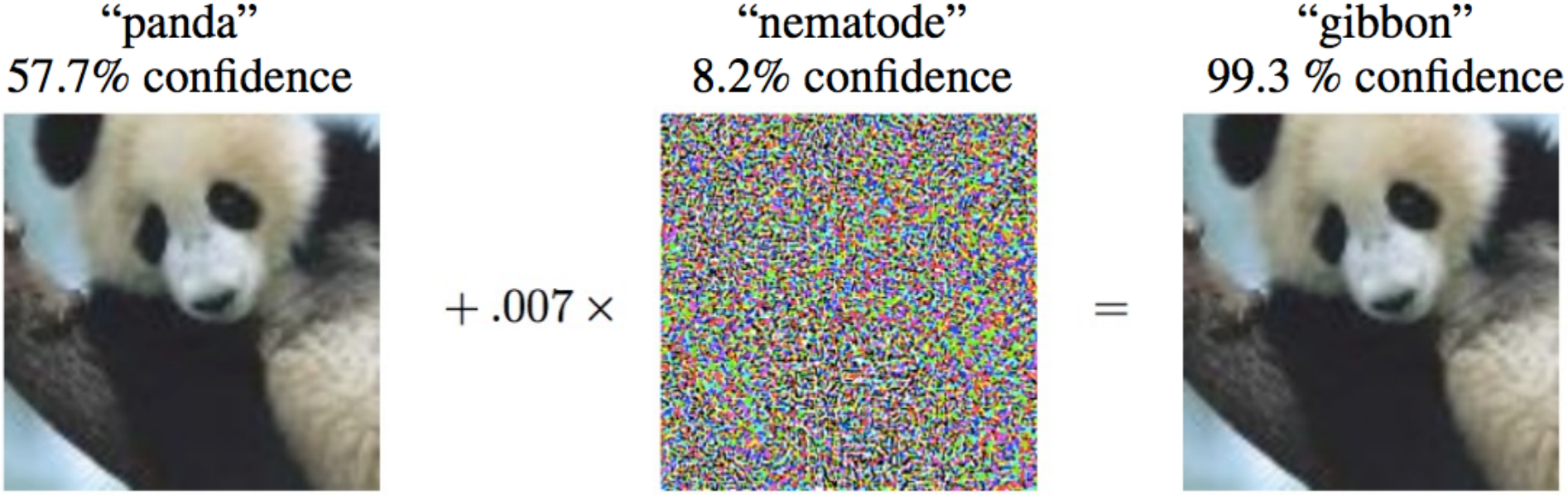
TECHNISCHE UNIVERSITÄT DARMSTADT

Aber man kann CNNs auch „täuschen“



Etwas Rauschen kann das Ergebnis komplett ändern

Aber man kann CNNs auch „täuschen“



Explaining and Harnessing Adversarial Examples [Goodfellow ICLR 2015]

Etwas Rauschen kann das Ergebnis komplett ändern

Was haben Sie bisher kennengelernt

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das Trainieren von tiefen Neuronalen Netzwerken folgt dem Lego-Prinzip mittels Vorwärts- und Rückwärtspropagierung
- Meistens wird ein stochastischer Gradientenabstieg benutzt
- DropOut und Datenaugmentierung vermeiden Überanpassung
- Trainieren von tiefen Netzwerken kostet viel Zeit und Energie. Daher, wenn möglich, Transferlernen mittels pre-trained Modellen
- **Es gibt noch viel mehr zu sagen: Recurrent Networks, LSTM, Siam Networks, Variational Neural Netowrks, …**

(Folien auf English)

TRANSFORMER

Two Pillars of NLP

Representation

- How to represent language to machines?

Modelling

- How to model language statistically?

Simultaneously solved by DL models,
e.g. transformers

Models have different emphases

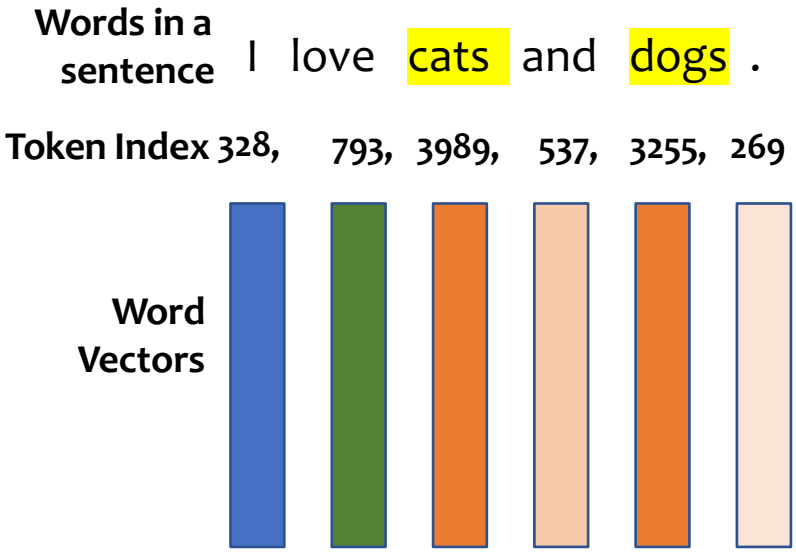
BERT | Openai-embedding

GPT | ChatGPT

Idea adapted from MIT 6.8610

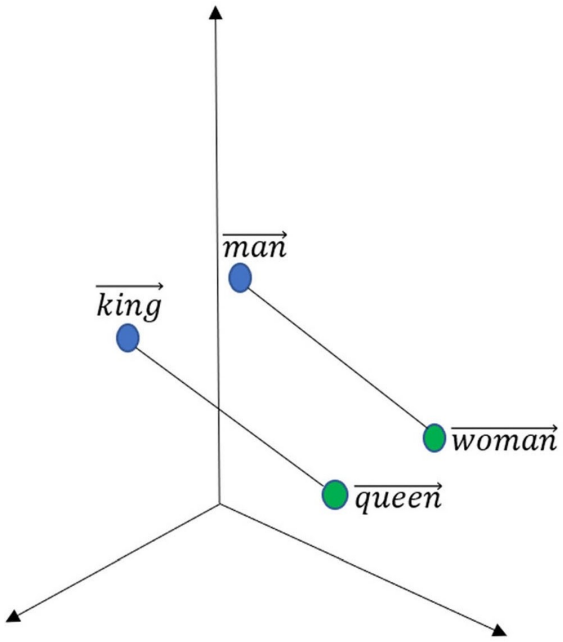
Representation: Word as Vectors

- Represent words in a vector space
  - Encode features in a distributed way.
  - Vector distance ⇒ similarity
  - Vector geometry ⇒ relation.
- Desirable features
  - Part of speech / usage (*Noun*)
  - Relation to other words (*cat – cats - kitten*)
  - Perceptual feature (for object.) (*cats are cute*)

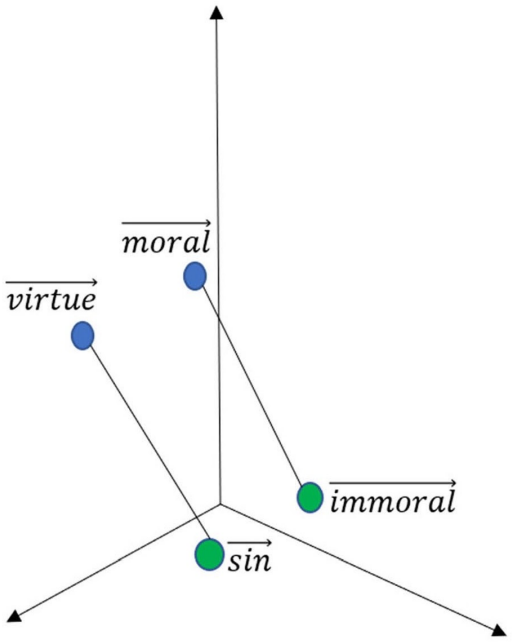


Word as Vectors:

Vector space geometry captures semantic relationship



a "gender" dimension

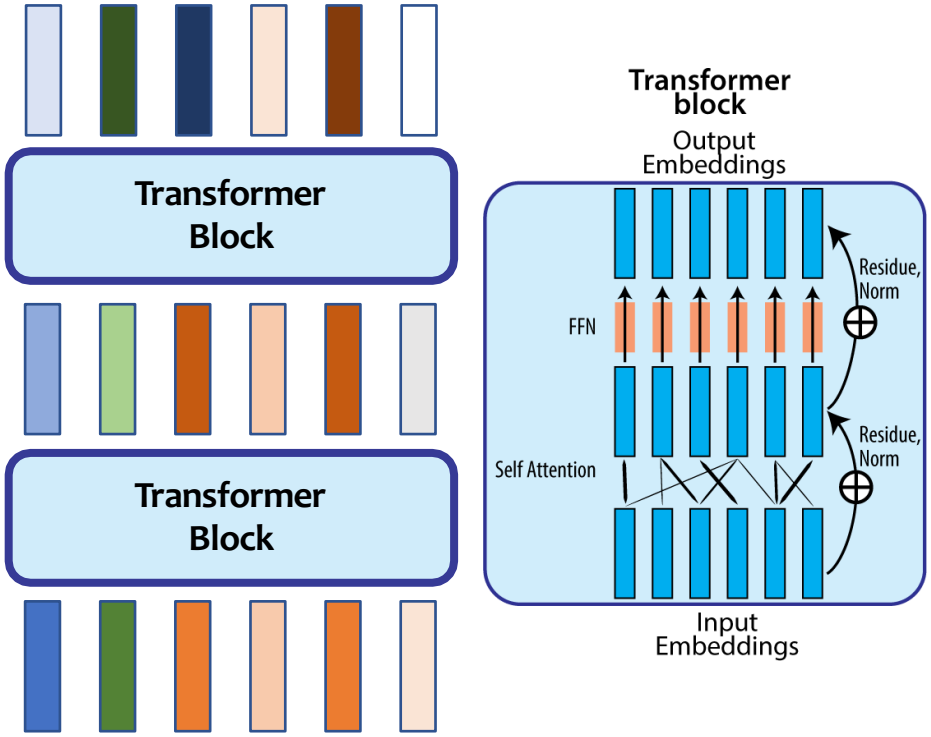


a "morality" dimension

Arseniev-Koehler, Alina. "Theoretical foundations and limits of word embeddings: what types of meaning can they capture?." *Sociological Methods & Research* (2021)

Word Vector in Context: Transformers

- Meaning of word depends on context.
  - "I can buy a can of fish."
- Word vectors should depend on context, not just word itself.
- Transformers let each word "absorb" influence from other words to be *contextualized*



Word Representation can help …

- Text classification / understanding
  - Sentiment analysis (*happy or angry*)
  - Text similarity assessment
  - Information retrieval (*find most similar text*)
  - Machine translation

Slide adapted from MIT 6.8610

Language model

- LM predicts the likelihood of any sequence of words, for a given *language corpus**.
  - $p(\text{"This is a fluffy dog."}) = p(w_1 w_2 w_3 w_4 w_5) = 0.132$ …
  - $p(\text{"This are a purple flying dear."}) = p(w_1 w_2 w_3 w_4 w_5 w_6) = 0.0002$ …

Note:
Each person / author has their individual LM. English ***marginalize*** *over those.*

LM of Shakespeare
LM of Trump

Language model

- Given such model, we can use conditional probability to
  - Guess missing words
    - "This _ a fluffy dog."
    - $\arg\max p(w_2|w_1 w_3 w_4 w_5) \rightarrow$ "is"
  - Predict next words
    - "This is a fluffy ..."
    - $\arg\max p(w_5|w_1 w_2 w_3 w_4) \rightarrow$ "$catc$"
  - Answer questions
    - "The Vatican locates in the city of ..."
    - $\arg\max p(w_7|w_1 w_2 w_3 w_4 w_5 w_6) \rightarrow$ "$Romec$"

Remark:

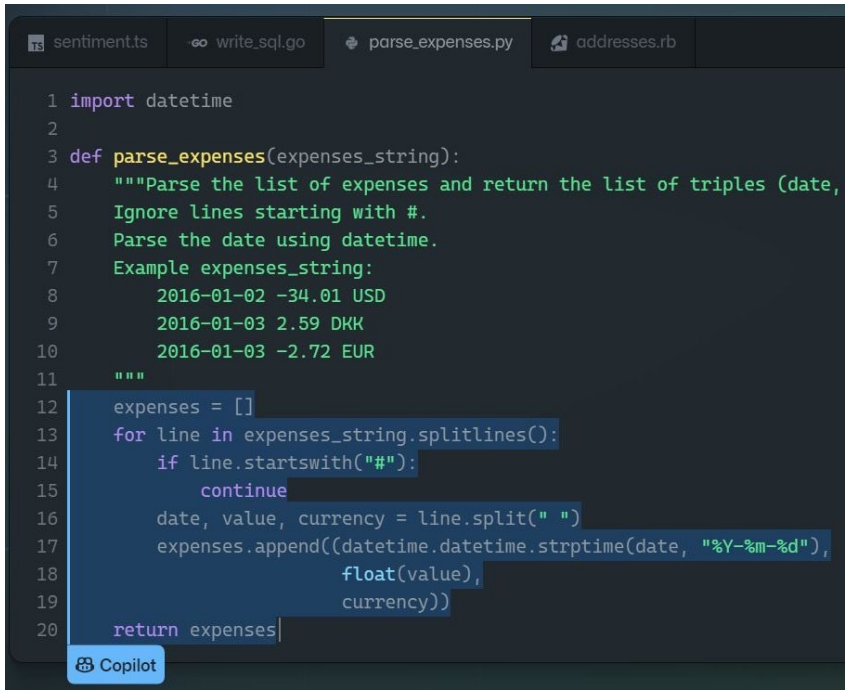
By modelling statistics, language model implicitly learns grammar, semantics, common sense, factual knowledge…

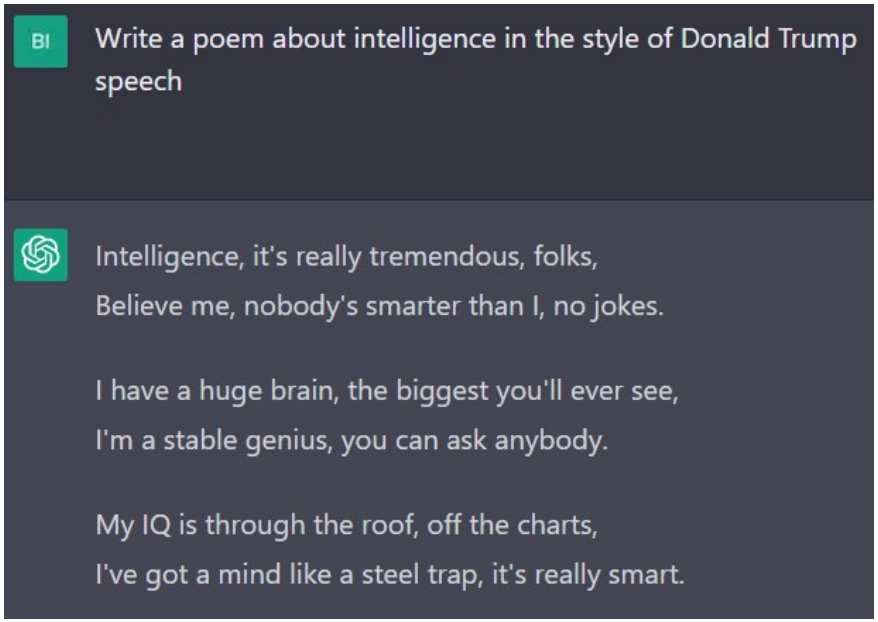
Language model can help …

- Text generation
  - Code completion
  - Chat bot
  - Text summary
  - Text correction
  - Machine translation
- Text classification
  - Detect spam
  - Authorship attribution
  - (*per likelihood comparison of two language models*)

Slide adapted from MIT 6.8610

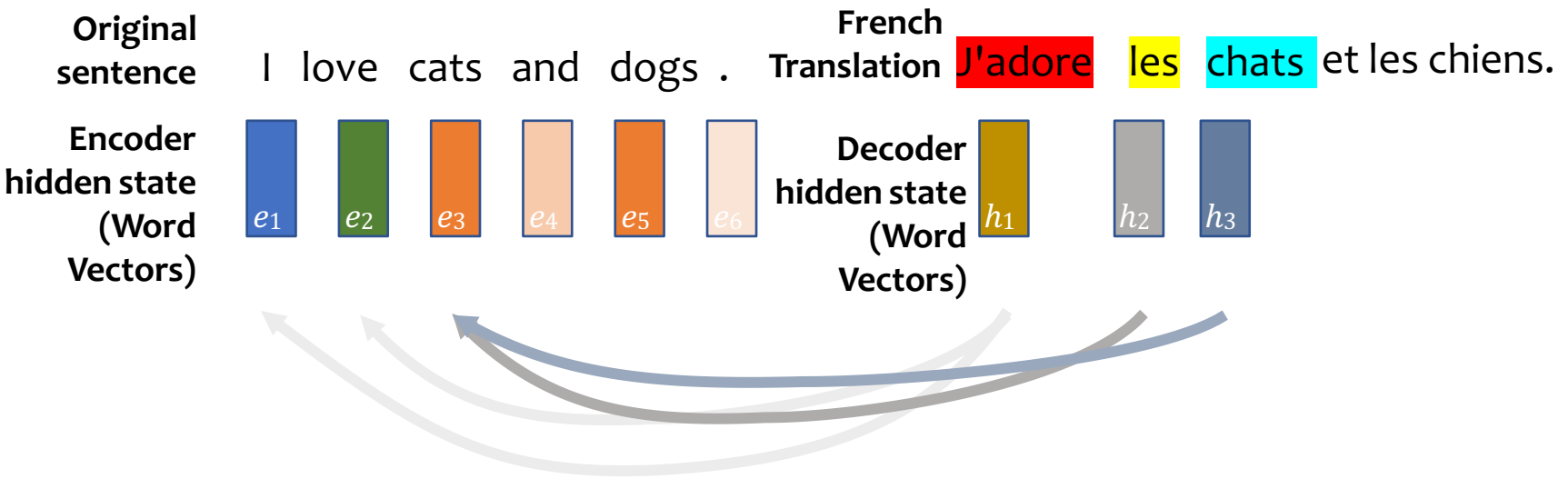
E.g. Github Copilot, ChatGPT





Origin of Attention:

Machine Translation (Seq2Seq)



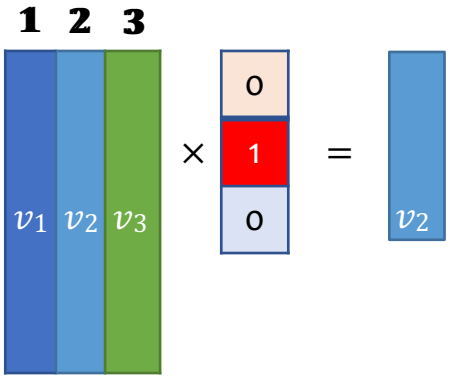
- Use **Attention** to retrieve **relevant info** from a batch of vectors.

From Dictionary to Attention
Dictionary: Hard-indexing

- Dictionary maps keys to values
  - `dic = $\{1 : v_1, 2 : v_2, 3 : v_3\}$`
  - Keys $1,2,3$
  - Values v_1, v_2, v_3

- Query retrieves information related to a key
  - `dic[2]` Query 2
  - Find 2 in keys
  - Get corresponding value.

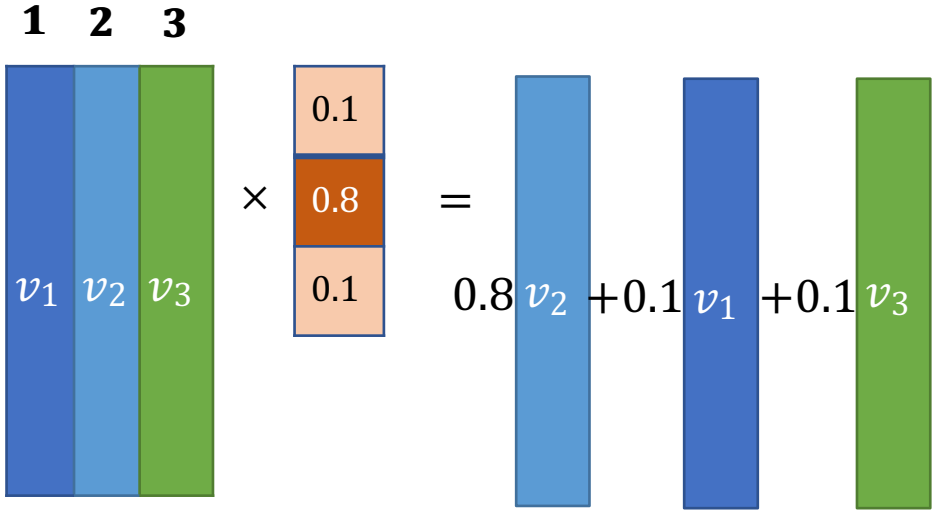
- Retrieving values as matrix vector product
  - a is one hot vector over the keys
  - Weighted sum the value vectors.



From Dictionary to Attention

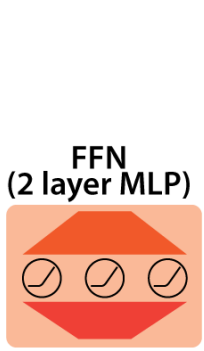
Attention: Soft-indexing

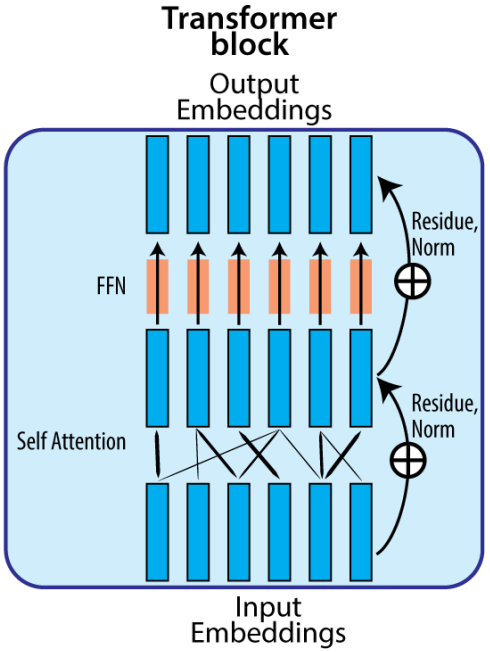
- Soft indexing
  - a as a distribution over the keys, represents how much information I want from this key.
  - Matrix vector product, weighted combines the values.
- How to compute a?
  - *Feature based attention:* based on similarity of query and key.



From Attention to **TransformerBlock**

- A **transformer block** has
  - Self Attention
    - information exchange ***between tokens***
  - Feed forward network
    - Information transform ***within tokens***
    - E.g. a multi-layer perceptron with 1 hidden layer
    - GeLU activation is commonly used.
  - Normalization (Layer normalization)
- A transformer model contains N x **transformer block**





„**Transformer** is all you need."

- **Position encoding**
  - Sinusoidal function
  - Learnt from scratch (**position embedding**)
- **Encoder:** process input in parallel
  - Self Attention + FFN
- **Decoder:** generate output autoregressively
  - Masked Self Attention
  - Cross attention (fetch into)
  - FFN

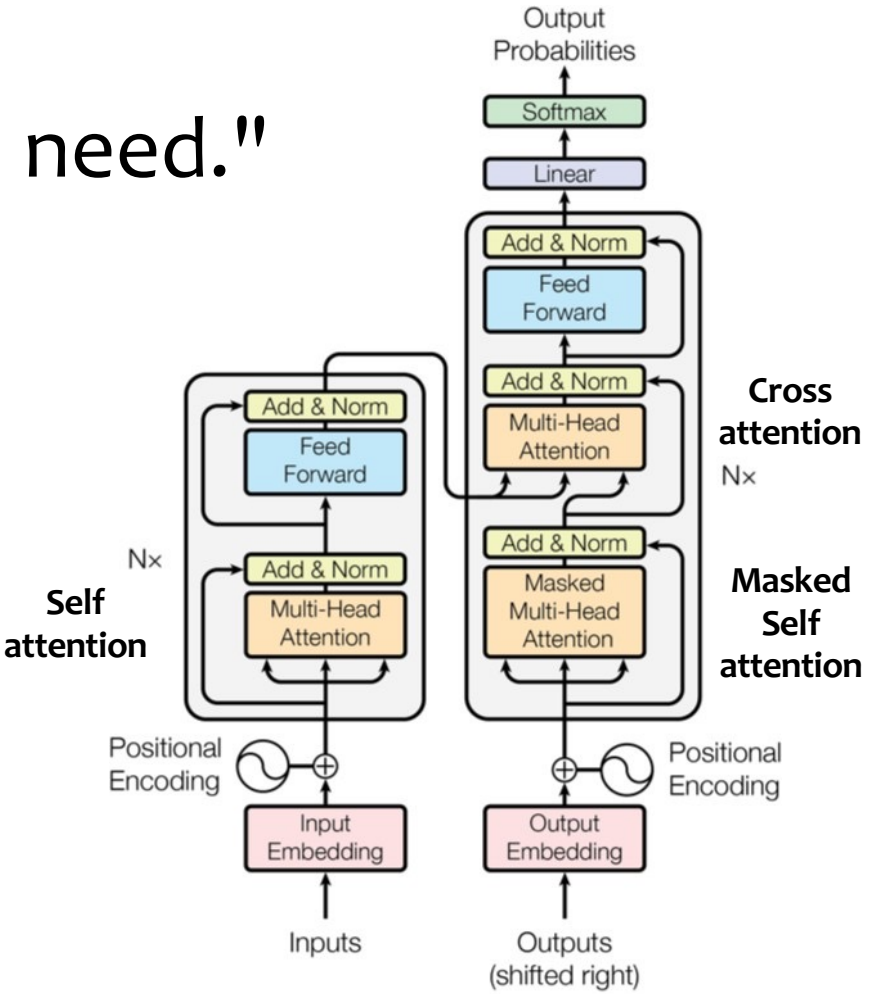
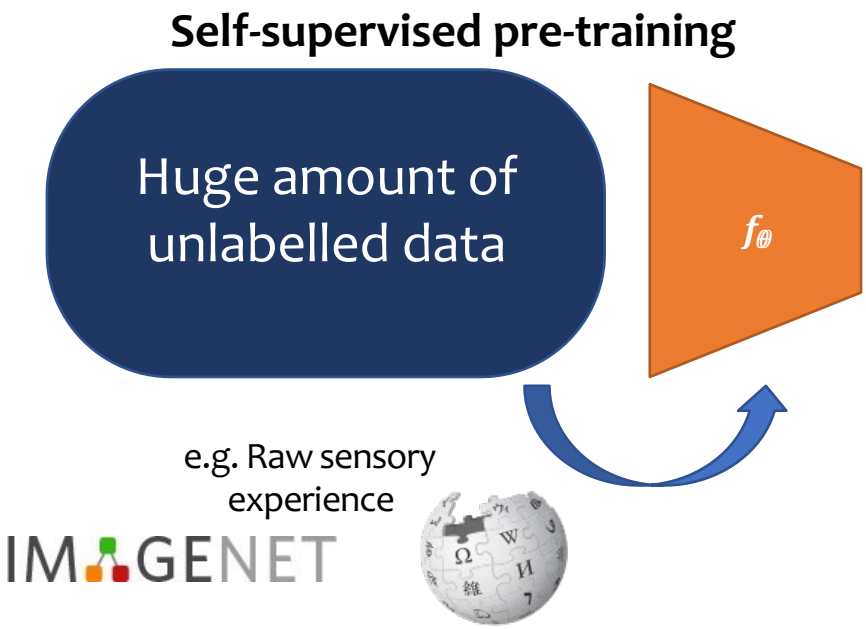


Figure 1: The Transformer - model architecture.

https://arxiv.org/abs/1706.03762

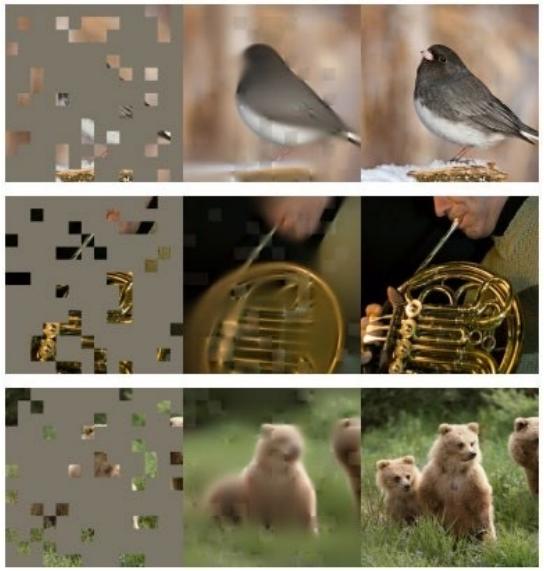
Self-supervised learning paradigm



- Objective of pretraining
  - Learn ***good representation*** , *s.t.*
    - downstream tasks can be easily solved on these representations. (e.g. linear decodable)
    - with fewer labeled samples

No supervision?
Create supervised tasks and solve them.

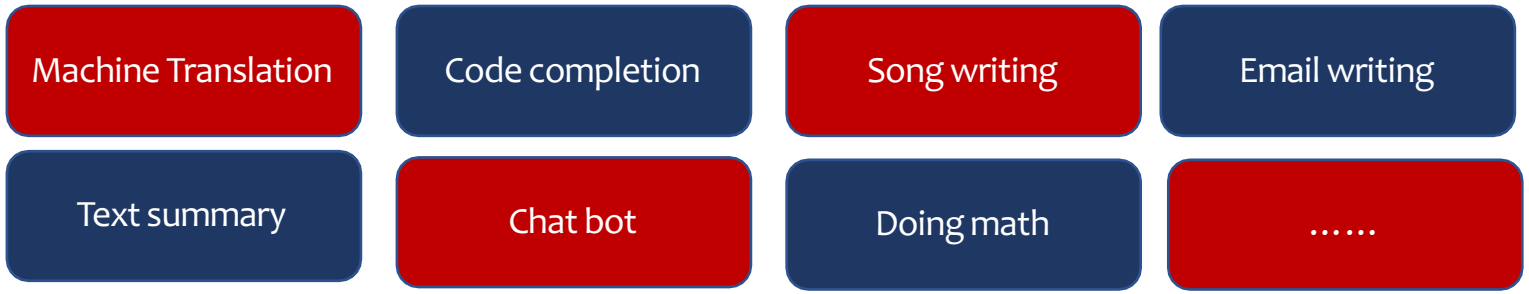
- Self-supervised learning
  - Mask out some part of data and learning by predicting
  - Learning by comparing similarity.



Self-prediction
(Masked Autoencoder, BERT, GPT)

Contrastive learning
(SimCLR, CLIP)

Two expectations for AI



Specialist

- Pro:
  - Solve one task far better than anyone.
- Con:
  - Perform poorly on any other tasks …

Generalist

- Pro
  - Flexibly solve many tasks with little modifications.
- Con
  - May not perform as well as specialist in one task.

Hungyi-Lee https://www.youtube.com/watch?v=F58vJcGgjt0&t=4s

Two major ways of using Language Models

Fine-tuning

- Gradient descent on weights to optimize performance on *one task.*

- What to fine-tune?
  - *Full network*
  - *Readout heads*
  - *Adapters*

  } *Parameter efficient fine-tuning*

Change the model "itself"

Prompting

- Design special prompt to cue / condition the network into specific mode to solve any tasks.

  - No parameter change. one model to rule them all.

Change the way to use it.

Hungyi-Lee
https://www.youtube.com/watch?v=F58vJcGgjto&t=4s