Deep Learning

Grundlagen Grundbegriffe Grundübungen

Alfatraining Kurs 2025-08 Dozent: Karsten Flügge

Themen des Tages

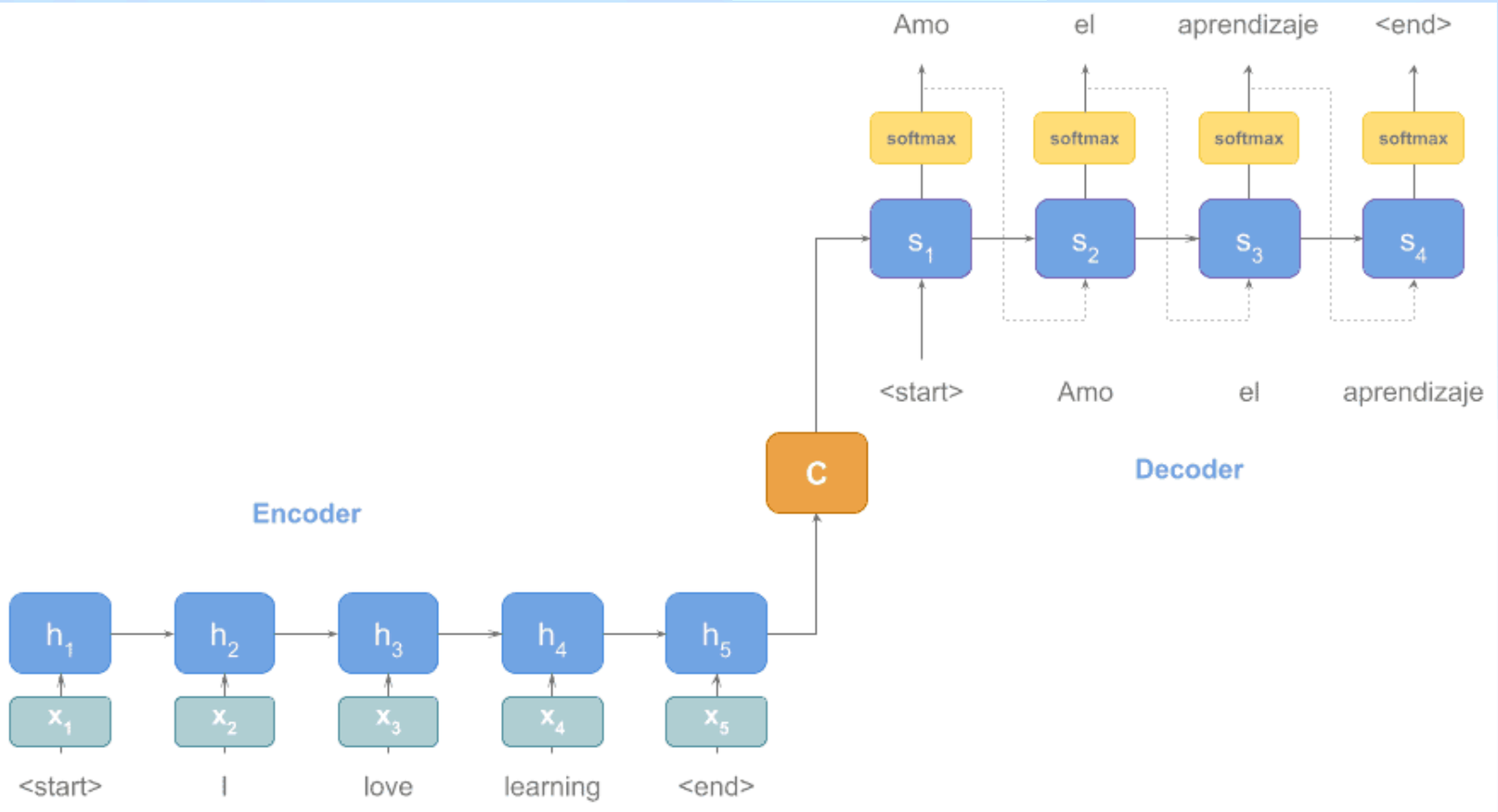
Tag 15

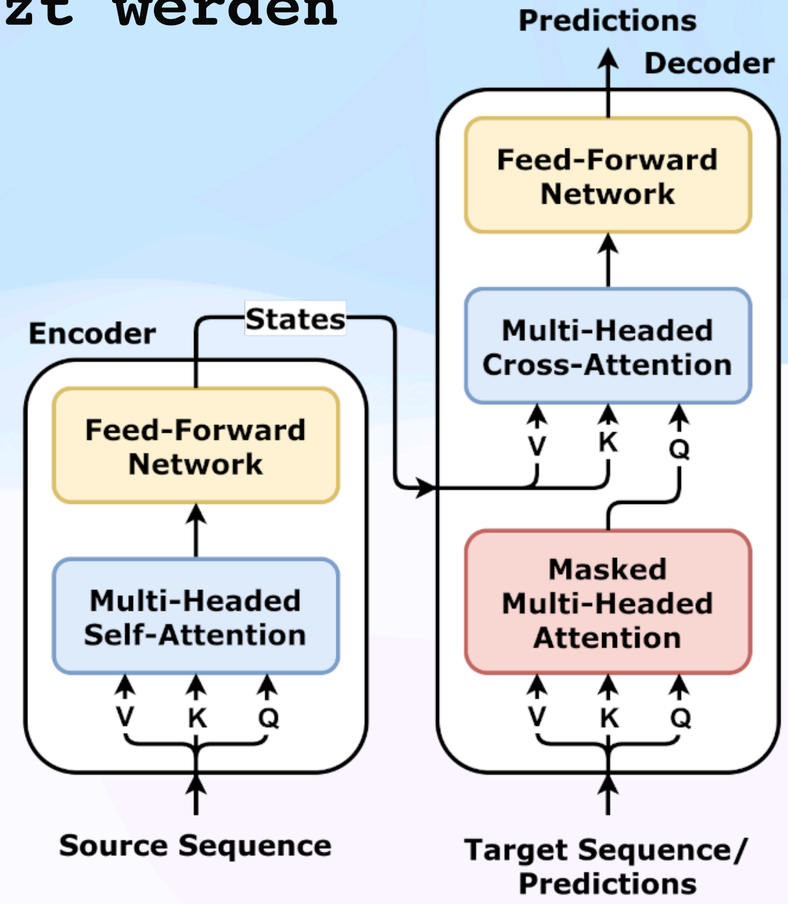
Attention-Schichten,
***Transformers*,**
Sprachmodelle
BERT, GPT
Textgeneration-Pipelines
Summarization,
Chatbots, fun

Epoch 48 … 60% done 0.0961
(['I', 'c', 'h', ' ', 'b', 'i', 'n', ' ', 'e', 'i', 'n', ' ',
'S', 't', 'u', 'd', *'e'*, *'r'*, *'t'*, ***'e'***, ***'n'***, '.', '<EOS>']

Transformers

Transformers sind seq2seq Modelle
bei denen RNNs durch Attention Blöcke ersetzt werden





Attention-Schichten

Attention-Schichten sind essenzielle Bausteine in modernen Deep-Learning-Modellen, insbesondere in den Bereichen Natural Language Processing (NLP) und Computer Vision. Der bekannteste Einsatzort von Attention-Schichten ist der **Transformer**-Architektur, die die Grundlage für Modelle wie GP**T**, BER**T** und viele andere bildet.

Attention-Schichten

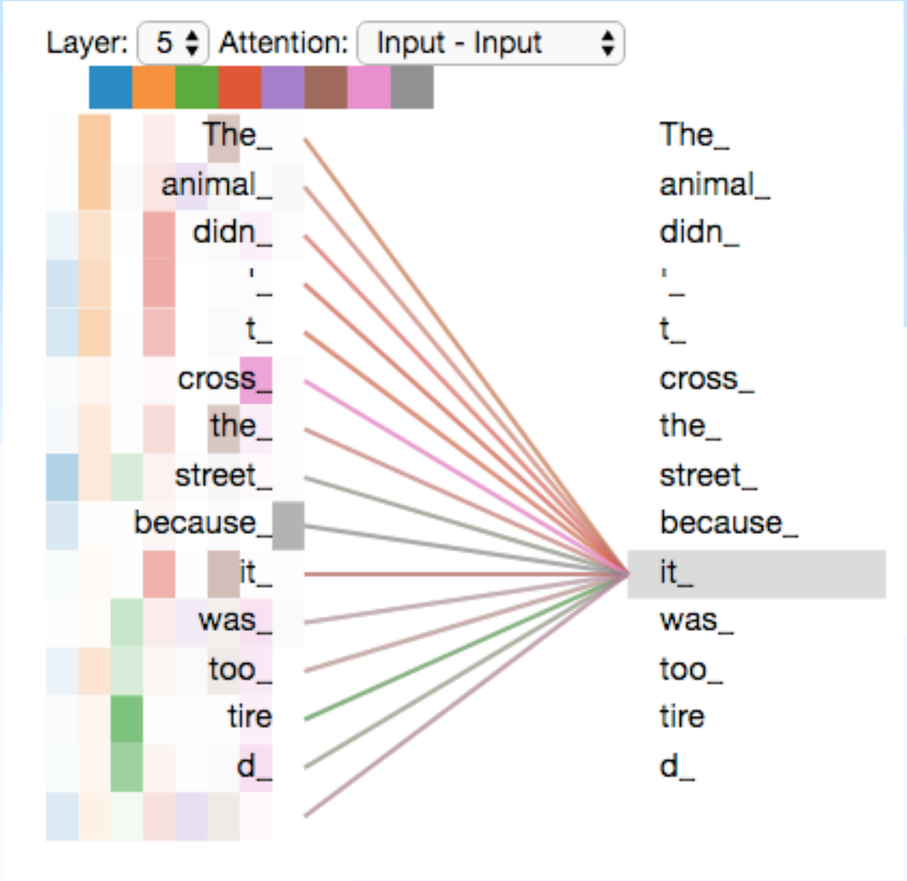
Dank des **Attention** Rechenschemas lernt ein Netz **korrelieren** und **attribuieren**,
also **Beziehungen** zwischen verschiedenen Teilen einer Eingabesequenz zu erkennen.
Zum Beispiel: worauf bezieht sich "**it**" im Satz:

"The cat sat on the mat because **it** was tired. What does **it** refer *to*? "
In diesem Satz bezieht sich das Pronomen "it" auf **"the cat"**.

"The cat sat on the mat because **it** was cosy."
In diesem Satz bezieht sich das Pronomen "it" auf **"the mat"**.

Kontext Abhängigkeit von Klassifizierung

Drei Worte sollen in Verbindung gebracht werden a b c
[cat + cozy] selten => it?
[cat < animals + tired] oft
[mat < object + cozy < external_state] oft
[mat < object + tired < emotional_state] selten



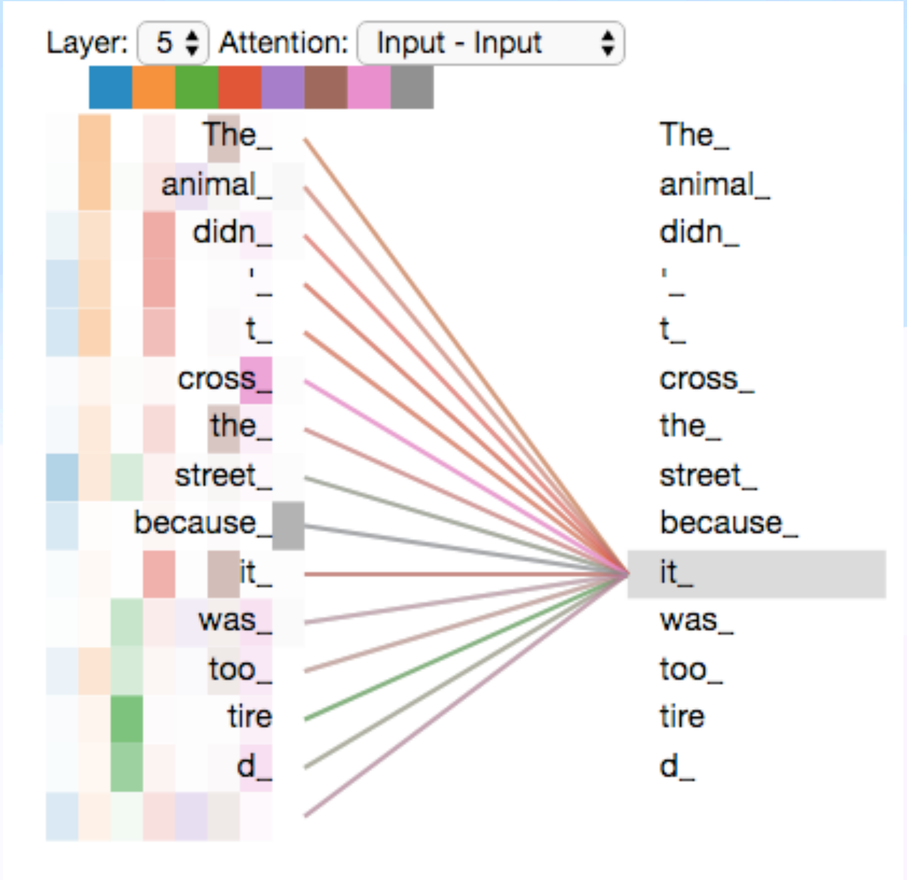
Attention-Schichten

Dank des **Attention** Rechenschemas lernt ein Netz **korrelieren** und **attribuieren**,
also **Beziehungen** zwischen verschiedenen Teilen einer Eingabesequenz zu erkennen.
Zum Beispiel: worauf bezieht sich "**it**" im Satz:

"The cat sat on the mat because **it** was tired. What does **it** refer *to*? "
In diesem Satz bezieht sich das Pronomen "it" auf **"the cat"**.

"The cat sat on the mat because **it** was cosy."
In diesem Satz bezieht sich das Pronomen "it" auf **"the mat"**.

Kontext Abhängigkeit von Klassifizierung



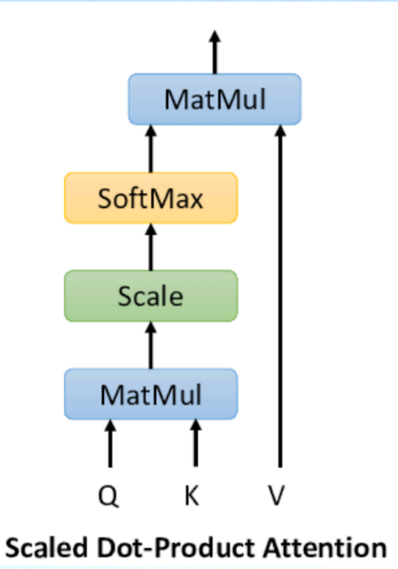
Wie ist der Einfluss von cosy/tired auf "it" kompatibel mit masking?
Antwort:
a) keine Maskierung;) oder
b) beim Lernen
c) tired can act 'on' it in later words, especially the last word/embedding before prediction

Attention-Schichten

Die Schlüsselidee ist, dass die **Aufmerksamkeit dynamisch und datenabhängig** verteilt wird. Die Berechnung erfolgt in der Regel durch eine gewichtete Summe der Eingabevektoren, wobei die Gewichte durch ähnliche Vektoren bestimmt werden. Die grundlegenden Begriffe hierbei sind:

- **Query (Abfrage):** Der Vektor, der beschreibt, worauf das Modell achten **soll**.
- **Key (Schlüssel):** Vektoren, die beschreiben, worauf das Modell achten **kann**.
- **Value (Wert):** Vektoren, die den tatsächlichen Inhalt repräsentieren, den das Modell **nutzt**.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Scaled Dot-Product Attention

c ≈ conditional activations **σ(Q·K)**, alle softmaxed
c ≈ conditional activation : Wert **V** verwenden (1) oder nicht (0) ? softmax: best passendstes paar
ΔE = ∑c*V = ∑conditions*Values : Embedding update conditionally applied (true if attending)

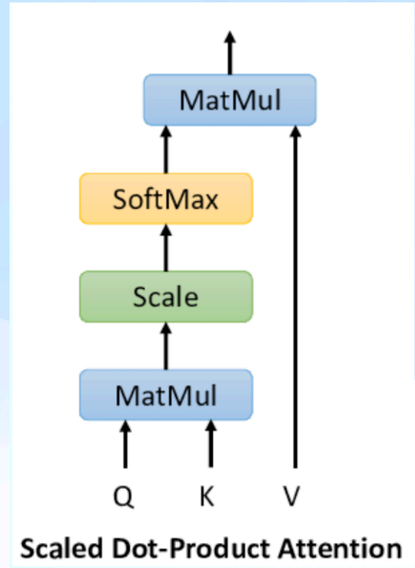
- condition ist ≈ 1 wenn Query zu Key "passt":
- condition ist ≈ 1 wenn **Feature** Query und Key ≥ 1 sind und somit das dot Produkt **Q·K** am Feature ≥ 1 ist
- [1,0,0]·[1,0,1]=1*1+0*0+0*1=**1** **softmax sucht das wichtigste / wahrscheinlichste 'Feature' in diesem Schritt**

Attention-Schichten

Drei Kanäle / Aspekte / Features können kombiniert werden:

- **Query (Abfrage):** Der Vektor, der beschreibt, worauf das Modell achten **soll**.
- **Key (Schlüssel):** Vektoren, die beschreiben, worauf das Modell achten **kann**.
- **Value (Wert):** Vektoren, die den tatsächlichen Inhalt repräsentieren, den das Modell **nutzt**.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Scaled Dot-Product Attention

Matrizen Q * K^T ≈ Haufen von Skalarprodukten
Extremfall Q: 1*n K: 1*n feature Vektoren mit n Einträgen =>
(1*n) * (n*1) => eine Zahl Aktivierung von **Key und Value**
Feature Vektoren werden Skalar multipliziert
a = Mensch ja / nein
b = essbar ja / nein
c = gross ja / nein

- [1,0,0]·[1,0,1]=1*1+0*0+0*1=**1 Mensch passt zu großer Mensch**
- **Matrizen Multiplikation als "Haufen von Skalar-Produkten"**

$$A \cdot B = (a_{il}) \cdot (b_{lk}) = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1L} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2L} \\ \dots & \dots & \dots & a_{il} & \dots \\ a_{I1} & a_{I2} & a_{I3} & \dots & a_{IL} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1K} \\ b_{21} & b_{22} & \dots & b_{2K} \\ b_{31} & b_{32} & \dots & b_{3K} \\ \dots & \dots & b_{lk} & \dots \\ b_{L1} & b_{L2} & \dots & b_{LK} \end{pmatrix} = C$$

A vererbt C
Zeilenzahl

$$C = (c_{ik}) = \begin{pmatrix} c_{11} & c_{12} & c_{13} & \dots & c_{1K} \\ c_{21} & c_{22} & c_{23} & \dots & c_{2K} \\ \dots & \dots & \dots & c_{ik} & \dots \\ c_{I1} & c_{I2} & c_{I3} & \dots & c_{IK} \end{pmatrix}$$

B vererbt C
Spaltenzahl

Attention-Schichten

Ableitbares Dictionary

Query ≈ Feature / Attribute

beim Training

Query[Key]:=Value ≈ **hashmap**

captial[germany]=berlin

captial[france]=paris

Key[Query]:=Value ≈ **relational property store**

germany**.**captial=berlin

france**.**captial=paris

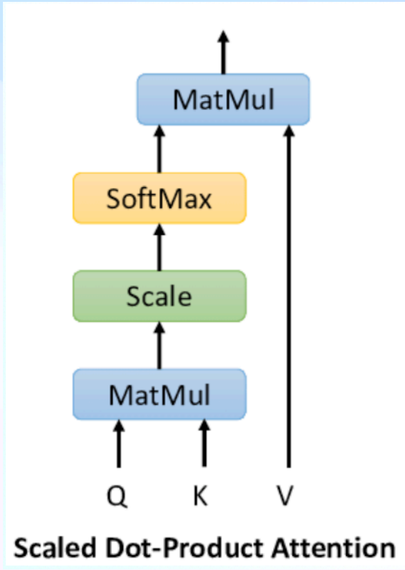
bei Anfragen

Query ° Key => Value

?capital[Deutschland] = softmax()*cities … ≈ "Berlin"!

- **Q·K** am Feature ≥ 1 ist

💡 query und key sind mathematisch symmetrisch! alles nur interpretation



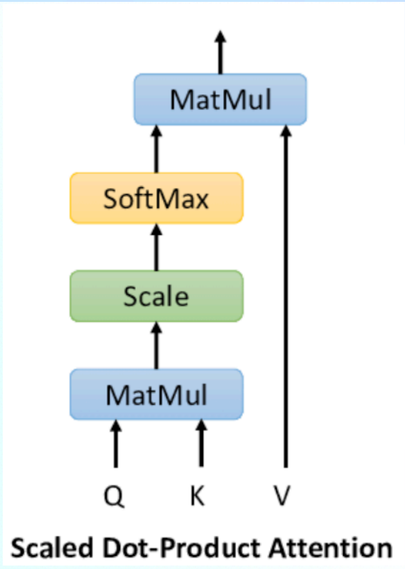
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Attention-Schichten

"Attention is all you need"

legendäres Paper ≈ 200.000 citations

https://arxiv.org/abs/1706.03762



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Attention-Schichten

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

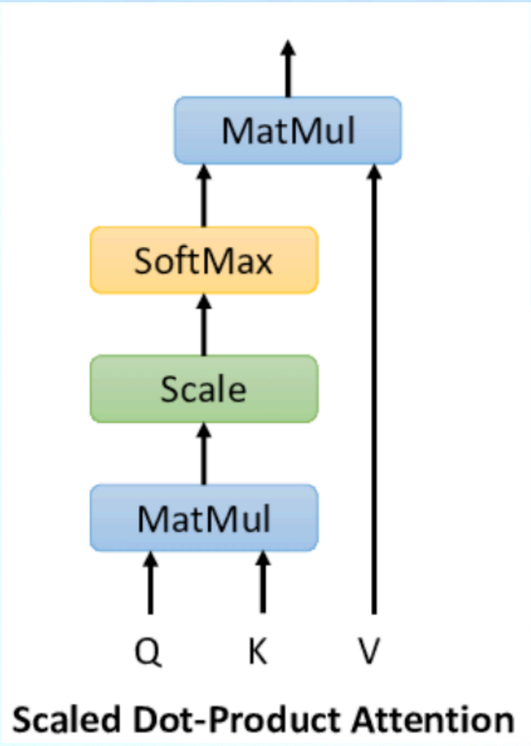
Input: **embedding Vector h** (hidden / context)

Output: **neuer embedding Vector? delta h, wird zum embedding Vector hinzuaddiert**

W_Q W_K W_V sind lernbare Matrizen (oder höhere Tensoren)

$Q = W_Q * h$ $K = W_K * h$ $V = W_V * h$

Q K V sind Aktivierungs**matrizen** (bzw mehrere Vektoren als Tensoren zusammengefasst)



Scaled Dot-Product Attention

Aspekte / Features können kombiniert werden:

- **Query (Abfrage):** Vektor(en), der beschreibt, worauf das Modell achten **soll**.
- **Key (Schlüssel):** Vektoren, die beschreiben, worauf das Modell achten **kann**.
- **Value (Wert):** Vektoren, die den tatsächlichen Inhalt repräsentieren, den das Modell **nutzt**.

Conditional Activation

σ(Q·K), alle softmaxed *per row* ≈ riesen **gates** viele Zeilen mit Werten in **(0,1)**

∑ = 1 pro Feature / row / aspekt

Generell: Ableitbarkeit

Wir können nur über Ableitungen lernen, daher
müssen alle Bausteine approximieren als **ableitbare Funktion!**

Softmax normierung so dass das maximum den höchsten Wert hat

Attention: wir approximieren die Idee vom dictionary/map, also value lookup per key

Attention-Schichten

Arten von Attention

Self-Attention: Jeder Teil der Eingabe "achtet" auf andere Teile derselben Eingabe. Dies ist der Kern des Transformer-Modells.

Multi-Head Attention: Die Self-Attention wird parallel über mehrere sogenannte "Heads" berechnet, die es dem Modell ermöglichen, verschiedene Aspekte der Eingabe gleichzeitig zu betrachten.

Cross-Attention: Hierbei achtet ein Teil des Modells auf eine andere Eingabesequenz, beispielsweise wenn in einer seq2seq-Architektur (z. B. bei Übersetzungsmodellen) der Decoder auf die Ausgabe des Encoders achtet.

Self Attention

softmax e^x/T / ∑ ...

Embedding Vorschau

20 questions

1 1 1 1 1 1 1 1 (1) 1 1 1

2^20 = 1048576

Um Macht des Embedding Vektors intuitiv zu machen.

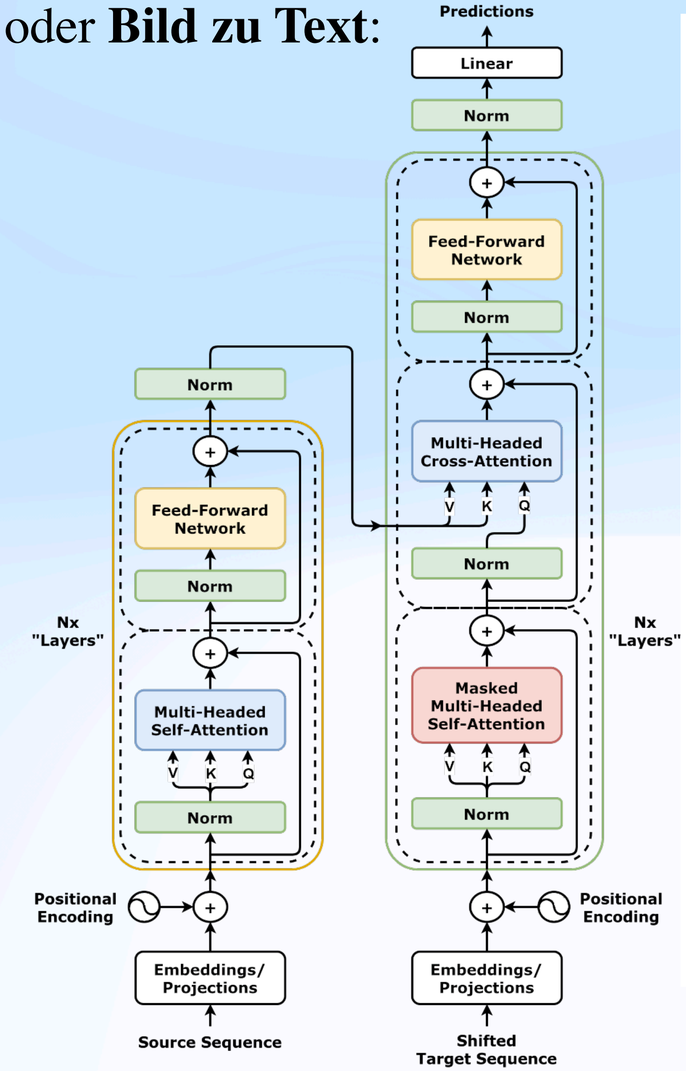
Quintessence: schon mit einem Vektor von 20 Nullen und Einsen kann man jeweils 1 Millionen verschiedene Objekte referenzieren. Tatsächlich sind es pro Spiel 1 Millionen, je nach kreativer Fragestellung können beliebig viele Objekte erfragt werden, vorausgesetzt beide Parteien haben einen gemeinsamen Wissensschatz!

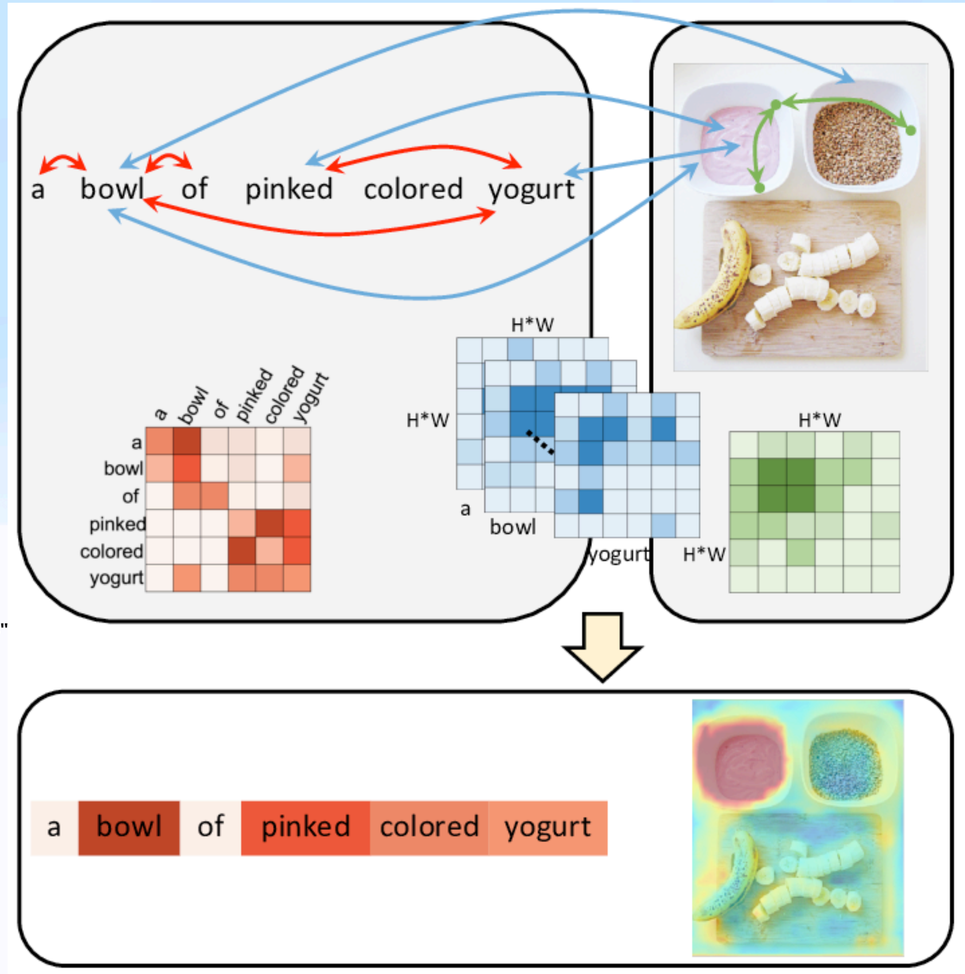
Ist das gesuchte Wort RegenBogenForellenSchnurz?

Wie bei Komprimierung: Wenn man bekannte Daten komprimiert wird die Datei kleiner.

Attention-Schichten

Bei **Cross-Attention** lernt das Netz Zusammenhänge zwischen verschiedenen Daten herzustellen zum Beispiel beim **übersetzen** oder **Text zu Bild** oder **Bild zu Text**:





Attention-Schichten

Vorteile:

Ermöglicht **parallele** Verarbeitung der Eingabedaten (im Gegensatz zu 'Schneeball' RNNs).

Kann **lange Abhängigkeiten** modellieren.

Nachteile:

Hoher Rechenaufwand bei großen Sequenzen (Quadratische Komplexität).

Potenziell hoher Speicherverbrauch.

Weitere Aspekte:

Kombination mit anderen Schichten, z. B. Feed-Forward-Schichten.

Regularisierung: Dropout auf Attention-Werte.

Transformer-Architektur

Der Transformer ist *im wesentlichen* Sequence to Sequence mit Attention statt RNN

Kann man RNN *überall* mit Attention ersetzen?

Mit genug compute: ja! Ansonsten:

GRU können mit weniger Speicher und Zeit oft ähnliche Ergebnisse erzielen, bis zu einer Grenze.

Attention kann generell **viel komplexere Zusammenhänge** lernen und über größere Distanzen. "Wie ein Magnet;)"

Needle in Haystack problem: Text mit 100000 Worten und eine Information versteckt.

Wie ist es mit Variabler Eingabe Länge?

GPT Context Window 4096 => bis zu n Millionen

Transformer-Architektur

Der Transformer ist eine revolutionäre Architektur in der Welt des Deep Learning, die insbesondere für Natural Language Processing (NLP) entwickelt wurde. Seit der Veröffentlichung des ursprünglichen **"Attention is All You Need"**-Papers von Vaswani et al. (2017) hat der Transformer das Gebiet der Sequenzmodellierung grundlegend verändert. Im Gegensatz zu früheren Modellen wie RNNs und LSTMs nutzt der Transformer keine **rekursiven oder sequentiellen Verbindungen**, sondern setzt vollständig auf das Konzept der **Self-Attention**.

1. **Encoder**: mehreren Schichten **Self-Attention** und Feed-Forward
2. **Decoder**: mehrere Schichten **Self-Attention** und Feed-Forward sowie **Cross-Attention** enthalten, um die Informationen aus dem Encoder zu nutzen.

Transformer-Architektur

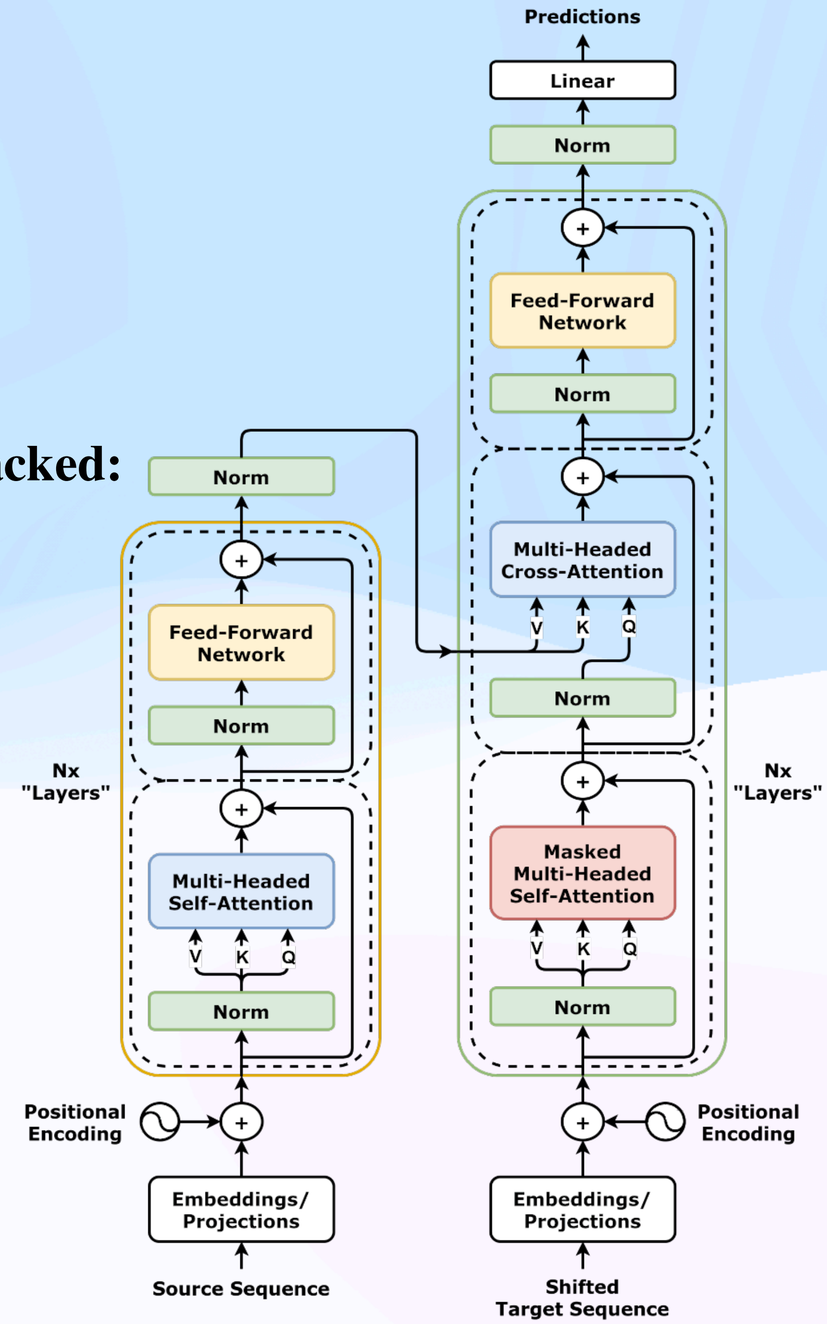
Die **original Transformer-Architektur** enthielt einige Komponenten,
welche die Architektur *leicht komplizierter* erscheinen lässt als **seq2seq:**

Vor oder hinter die Blöcke wurden verschiedene Norm Schichten gestacked:

post-LN LayerNorm hinter Block (original)
pre-LN LayerNorm vor Block (besser)
Scaling: Vektoren werden mit Faktor multipliziert

LayerNorm kann zum Beispiel so etwas sein wie
Batch Normalisierung, oder ähnliche Ansätze.

Grund: ohne diese haben die Netze schlecht trainiert.

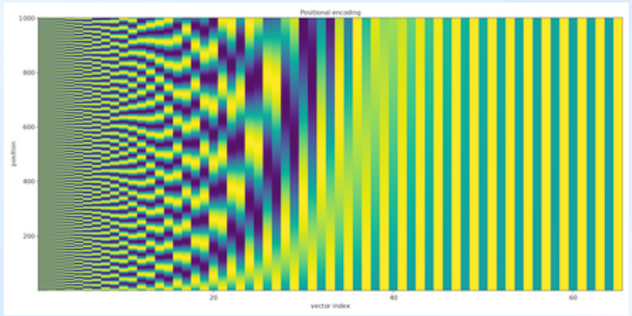


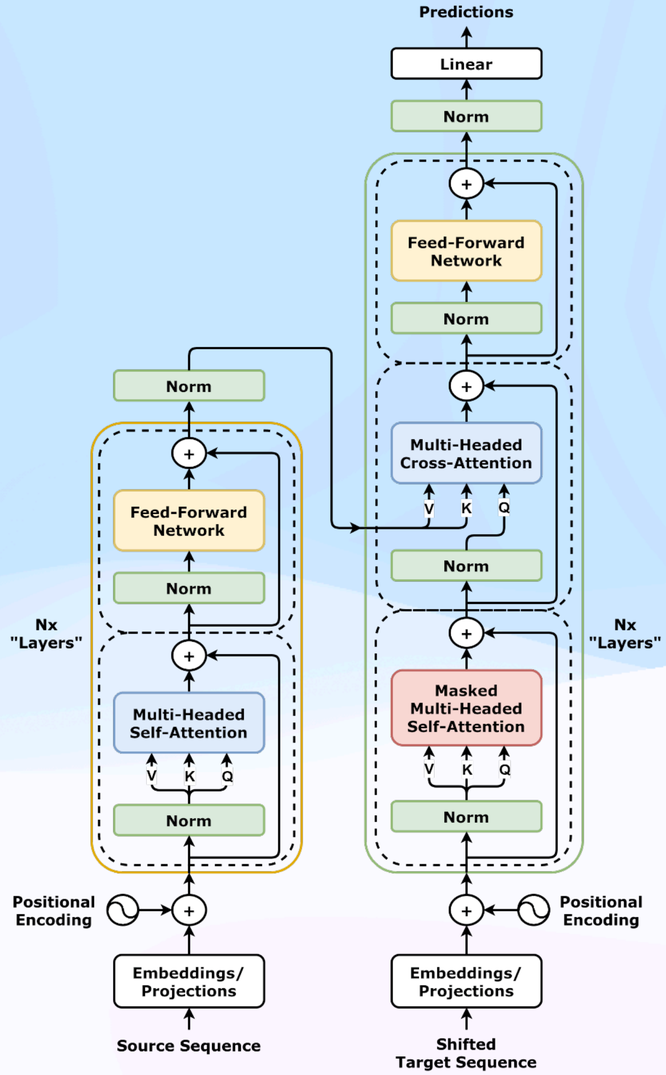
Transformer-Architektur

Position Encoding:

Damit das Netzwerk Werk die einkommenden Worte nicht einfach als Sack "Bag of Words" wahrnimmt, wird zusätzlich die Position im Satz mit hinzu codiert.

$$f(t) = \left(e^{it/r^k}\right)_{k=0,1,\ldots,\frac{d}{2}-1}$$ **(sin(θ), cos(θ)) trick**





Later, one found that causal masking itself provides enough signal to a Transformer decoder that it can learn to implicitly perform absolute positional encoding without the positional encoding module.

Transformer-Architektur

Position Encoding:

Warum nicht als Haufen von Tensoren? Dann muss Netz sich merken welcher Vektor an welcher Stelle kam.

Positional embedding:

(1, 0, 0, 1) + (.1, .2, …)

Naiver Ansatz: Füge einfach Wort Position an den embedding Vektor an:

(1, 0, 0, 1) + (2) = Hund an Position 2 regression Wort 1 2 3 unstabil

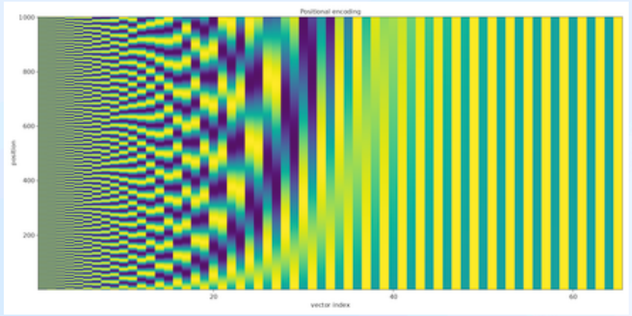
(1, 0, 0, 1) + (0, 1, 0, 0) = Hund an Position 2

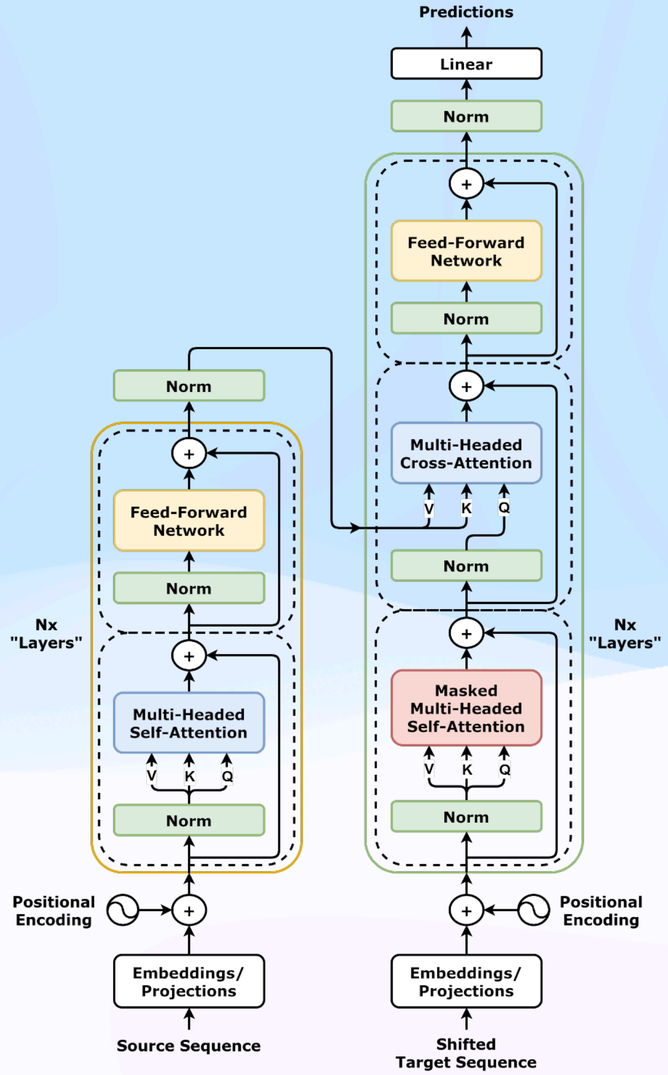
Problem bei variabler Satzlänge, verschwenderisch

Damit das Netzwerk Werk die einkommenden Worte nicht einfach als Sack "Bag of Words" wahrnimmt, wird zusätzlich die Position im Satz mit hinzu codiert als *elementweise addition!*

$$f(t) = \left(e^{it/r^k}\right)_{k=0,1,\ldots,\frac{d}{2}-1}$$

(sin(θ), cos(θ)) trick





Later, one found that causal masking itself provides enough signal to a Transformer decoder that it can learn to implicitly perform absolute positional encoding without the positional encoding module.

Transformer-Architektur

1. **Parameterwachstum**

- Concatenation verdoppelt (oder zumindest vergrößert) die Dimension des Eingabevektors. Das zwingt jedes nachfolgende Layer, in höherer Dimension zu operieren → mehr Parameter, quadratisch steigende Kosten in Attention-Matrizen.

2. **Skalen**problem

- Wenn man [x, p] (Konkatenation) nimmt, kann das Netz Position und Inhalt leicht trennen. Aber: es muss lernen, diese Informationen zu mischen. Bei Addition (x+p) ist die Fusion erzwungen: jedes Feature hängt gleichzeitig von semantischem und positionalem Anteil ab.

3. **Symmetrie**argument

- Attention basiert auf Skalarprodukten QK^\top. Bei Addition fließen Positionsanteile direkt in diese Produkte ein, d.h. Positionsinformation wirkt auf dieselbe Weise wie Inhaltsinformation. Bei Konkatenation müsste man lernen, wie man die Position überhaupt in das gleiche Rechenformat überführt.

4. Generalisation auf **variable Längen**

- Concatenation erfordert, dass das Netz eine Art “zweiten Kanal” für Position separat versteht. Bei Addition (insbesondere sinusoidal) entstehen Translationseigenschaften: relative Abstände werden direkt linear kodiert.

5. **Empirie**

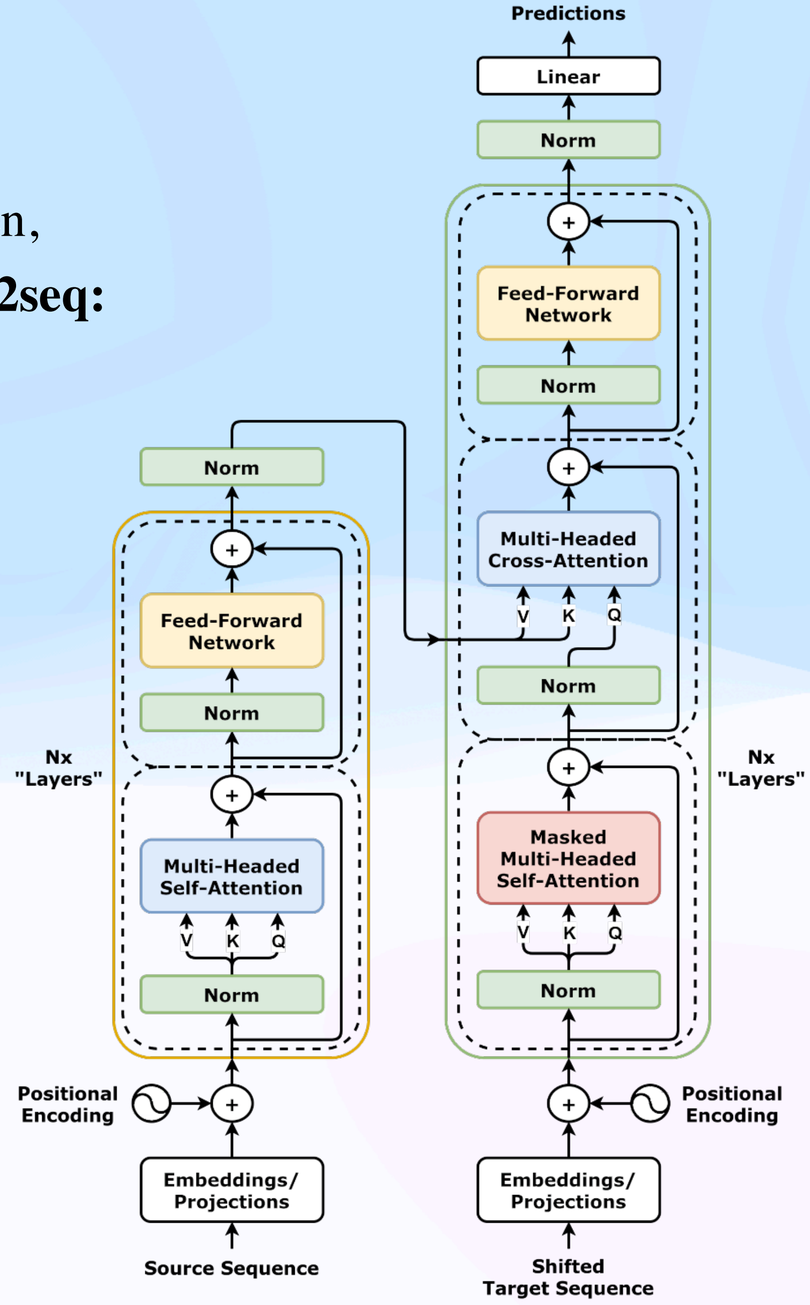
- Arbeiten wie Vaswani et al. 2017 berichten, dass Concatenation getestet wurde, aber schlechtere Stabilität zeigte. Späteres Research (z.B. Shaw et al. 2018 mit relative positional encoding) zeigt, dass explizite relative Information in Attention-Matrix deutlich besser funktioniert als jede Form von simplen Concatenation.

Transformer-Architektur

Die **original Transformer-Architektur** enthielt einige Komponenten,
Welche die Architektur leicht **komplizierter** erscheinen lässt als **seq2seq:**

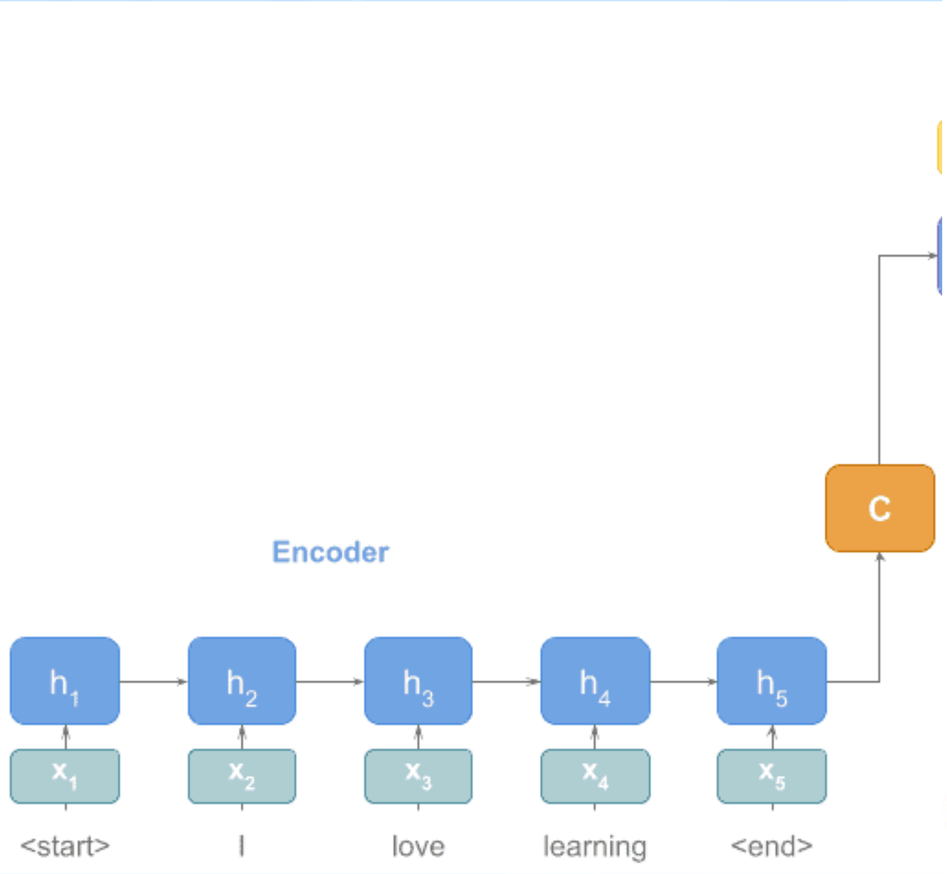
Hinter jedem Multihead Self-Attention Block
kommt ein Multiplayer Perceptron (Synonyme?)

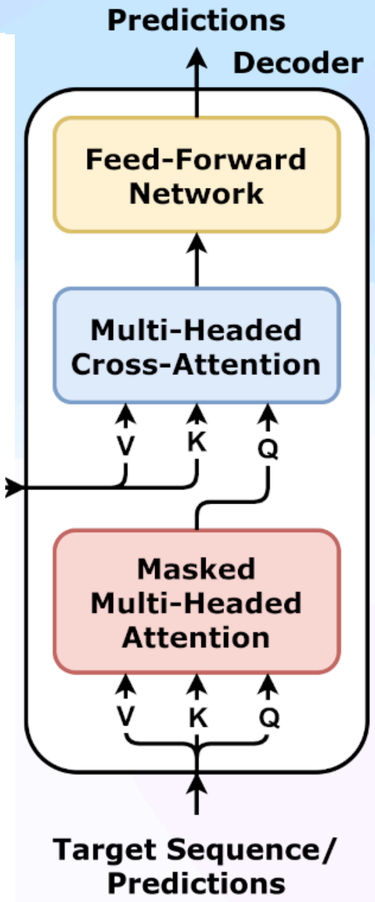
Damit das Modell nicht in die *Zukunft* gucken kann,
werden spätere Inputs "**maskiert**" "**causal**"
Dies geschieht in dem vor dem softmax **-∞ addiert** wird,
denn softmax(**-∞**) **= 0**, die Aktivierung wird also unterbunden
Maskierungs Matrix M wird vor dem softmax hinzu**addiert**
[[0, -∞, -∞],
[0,0,-∞],
[0,0,0]]
softmax(K(i)*V(i)+M)



Transformers

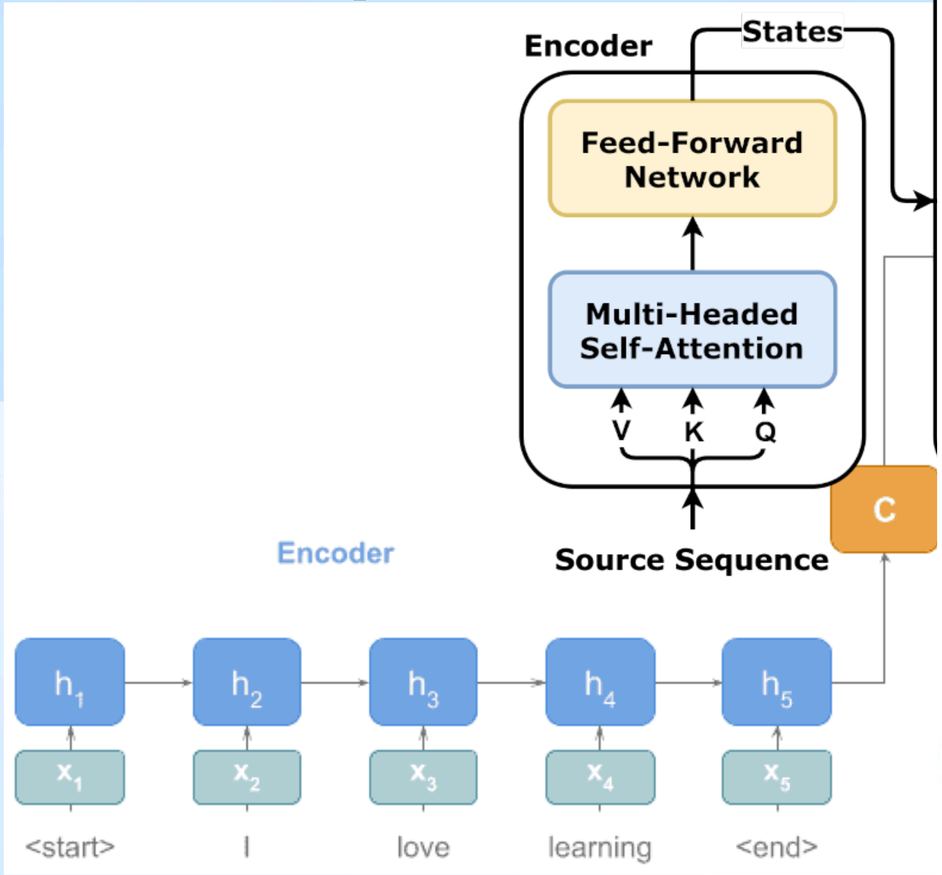
Spezielle Transformers:
"Encoder Only" Elmo & BERT, … ***Embedding***
"Decoder Only" GPT, …





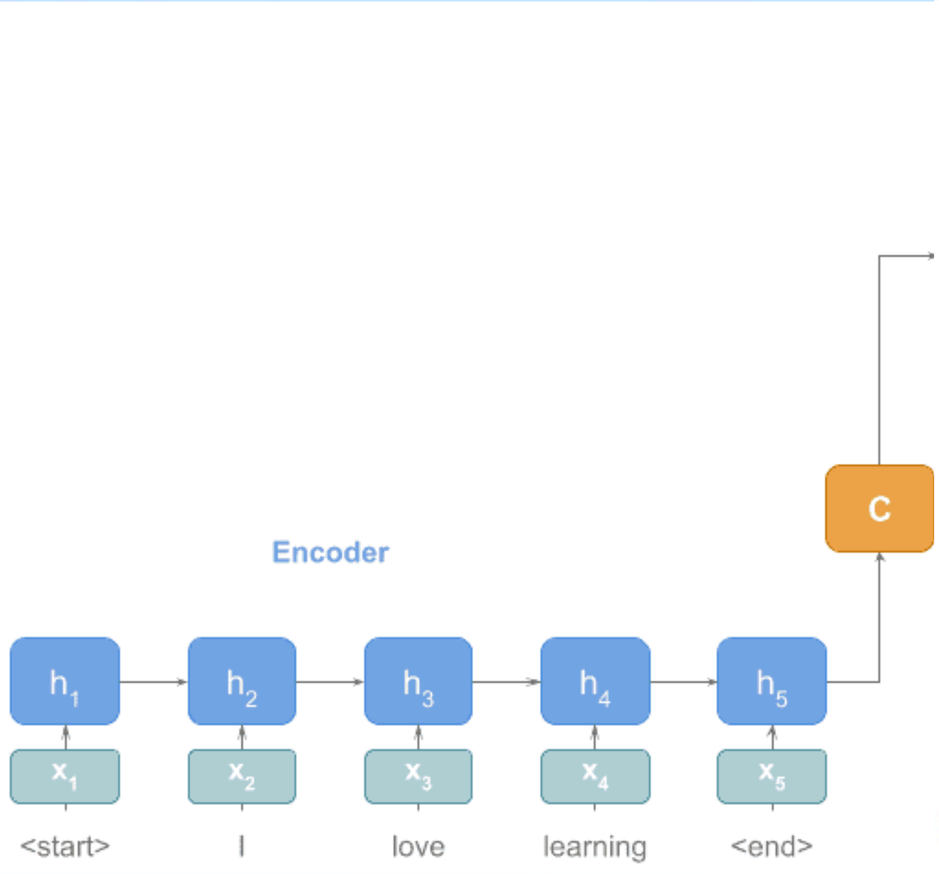
Transformers

Spezielle Transformers:
"Encoder Only" Elmo & BERT
"Decoder Only" GPT, …



BERT

"Encoder Only" Elmo & BERT, …
Statt des Decoders lernt man hinter den Context C spezielle Task Heads



BERT

"Encoder Only" BERT:
Bidirectional **E**ncoder latentent **R**epresentations from **T**ransformers

Statt des Decoders lernt man hinter den Context C spezielle Task Heads:

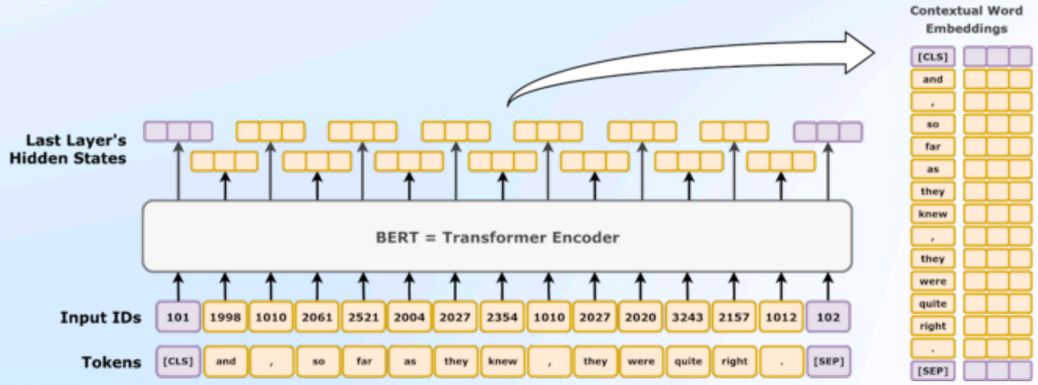
Masked Language Modeling

- **MLM Task Head**: A fully connected layer over the token hidden states to predict masked tokens.

Suche wahrscheinlichste Tokens per softmax Ich ___ gerne Berliner. => **esse / bin masking**
Berliner Token => Vektor (Quantenzustand Mensch / Gebäck) => Attention: kombiniert mit essen im INPUT => Vektor wird **spezifiziert** => Gebäck Berliner!

Next Sentence Prediction

- **NSP Task Head**: A fully connected layer over the [CLS] token to predict whether two sentences are consecutive.
  Ist der folgende Satz plausibel bit **ja: nein?**



bidirectional model that understands context from both directions

GPT Generative Pre-trained Transformer family

"Decoder Only" Seq2seq Architektur

ohne Cross Attention

Also einfacher als ein vollständiger Transformer

Attention is all you need! ≈ Autoencoder mit Attention statt RNN

GPT Generative *Pre-trained* Transformer family

Post training z.B. bei GPT-3.5 Alignment

1) finetuning auf chat flow (also Frage und Antwort)
2) Verbot aller rassistischen und politisch inkorrekter Äusserungen
3) generelle Stil Beeinflussung (sachlich, nicht widersprechend ausser fachlich)
4) gute Quellen, trustworthiness

GPT Generative *Pre-trained* Transformer family

Generative: Weil es als **Autoencoder** neue Texte erzeugen kann GAN!

***P**re-trained:* Weil es als **Autoencoder** das Internet gelesen hat (unsupervised)

Transformer: Architektur kennen wir jetzt (fast)

GPT Generative *Pre-trained* Transformer family

Generative: Weil es als **Autoencoder** neue Texte erzeugen kann GAN!

***P**re-trained:* Weil es als **Autoencoder** das Internet gelesen hat (unsupervised)

Transformer: Architektur kennen wir jetzt (fast)

=> Nazi mit schlechten Manieren

Post Training:

RLHF Reinforcement Learning with Human Feedback

Extra Trainingssatz mit "Gutem Verhalten"

(Amazon Mechanical Turks)

GPT RLHF

Post Training:

RLHF Reinforcement Learning with Human Feedback

Komplettes Finetuning auf allen Gewichten

Extra Trainingssatz mit "Gutem Verhalten"

Chat artiges Verhalten. => ChatGPT

Anhand von Beispiel Trainingsdaten.

(Amazon Mechanical Turks)

≠ Prompt: Man beeinflusst Ausgabe durch "Präambel"

Am Anfang wurde Prompt einfach vor die User Eingabe eingefügt.

Feinsteuerung auf zwei Ebenen! Mittlerweile mit META tokens

<system>You are a chatbot</system> <user>Wie wird das Wetter</user>

Slop

Generative: Weil es als **Autoencoder** neue Texte erzeugen kann

***P**re-trained:* Weil es als **Autoencoder** das Internet gelesen hat (unsupervised)

Slop: Liest eigene Inhalte =>

Vokabular verschiebt sich,

Halluzianationen / Fehlinformationen verstärken sich!

GPT Generative *Pre-trained* Transformer family

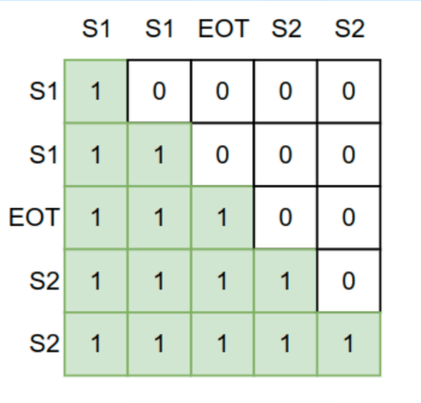
Training so wie seq2seq nur dass Eingabe = Ausgabe (Target) Satz ist

Blind unsuperwised vortrainieren:
Lerne bei beliebigen Daten das nächste Wort vorherzusagen.

Das ganze Internet steht als Lernbasis zur Verfügung
(Keine Satz Paare wie bei Translation notwendig)

Masking

teacher forcing training mit Ausblenden der Zukunft (Maskierung mit 0en in diagonal Matrix / -inf vor softmax)



	S1	S1	EOT	S2	S2
S1	1	0	0	0	0
S1	1	1	0	0	0
EOT	1	1	1	0	0
S2	1	1	1	1	0
S2	1	1	1	1	1

Kausalität: Aktuelles Wort soll nur von Vergangenheit beeinflusst werden

GPT

gpt from scratch:

https://www.youtube.com/watch?v=kCc8FmEb1nY

https://github.com/Smit6/GPT-from-scratch

Nano GPT

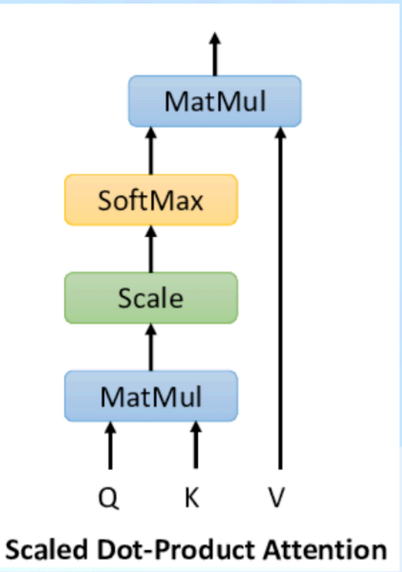
Vollständige Implementierung in 311 Zeilen!

https://github.com/karpathy/minGPT =>

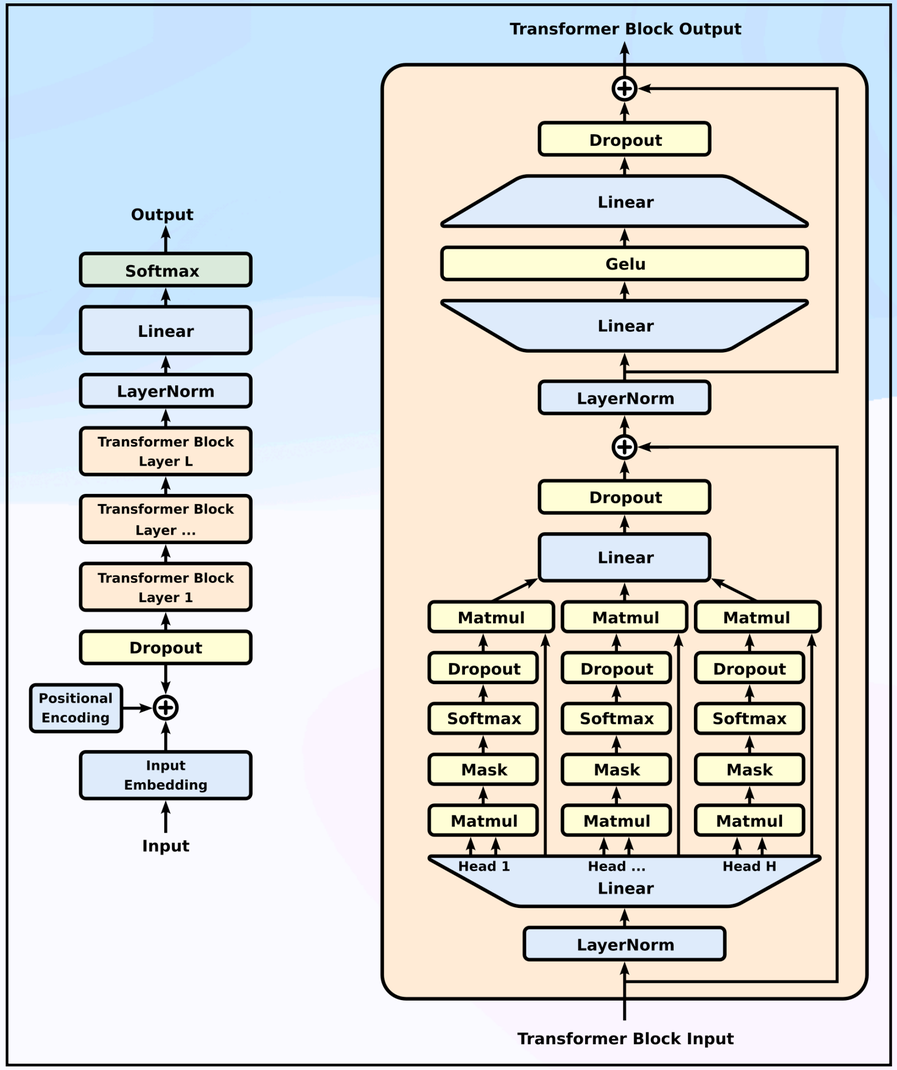
https://github.com/karpathy/nanoGPT

tag15/nano-gpt.py << auf Shakespeare loslassen!

GPT Generative Pre-trained Transformer family



👑 FINALES DIAGRAMM 👑



Ihr könnt nun (hoffentlich) zu jedem Detail sagen:

"verstehe ich" (im Prinzip, etwas)

Aufgabe: Lest Euch nano-gpt.py durch und seht ob etwas fragwürdig ist

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Random im DL

Wie kommen zufällige unterschiedliche Predictions zustande?

Bei Inferenz:

a) **Temperatur** bei Auswahl der Softmax predictions

b) **GPU randomness (a + b) + c ≠ a + (b + c)**

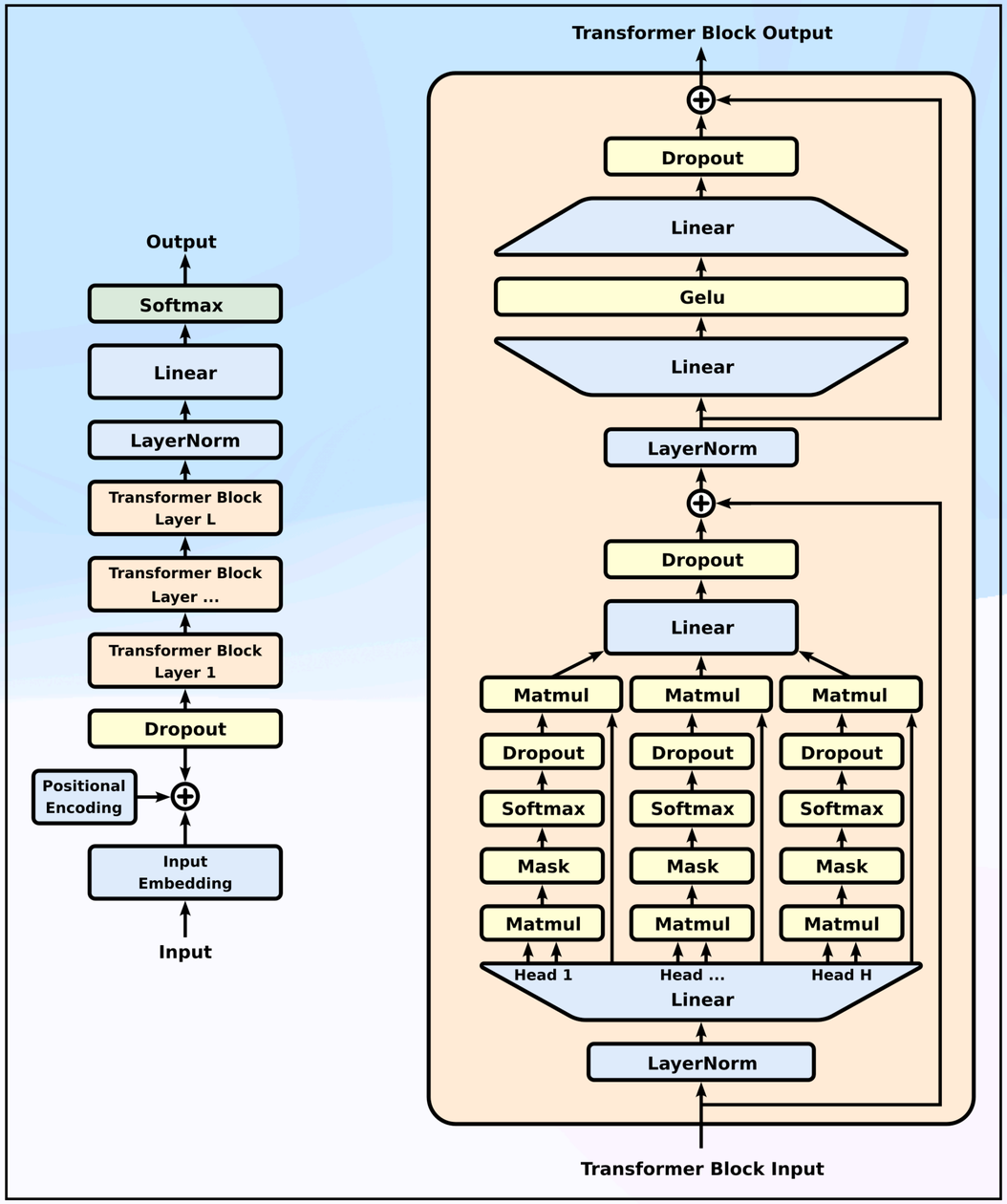
gilt NICHT bei float numbers!

je nachdem welcher GPU kernel eher fertig ist

ändert sich die Reihenfolge von Berechnungen!

seed()

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Man experimentiert mit verschiedenen Aktivierungen:

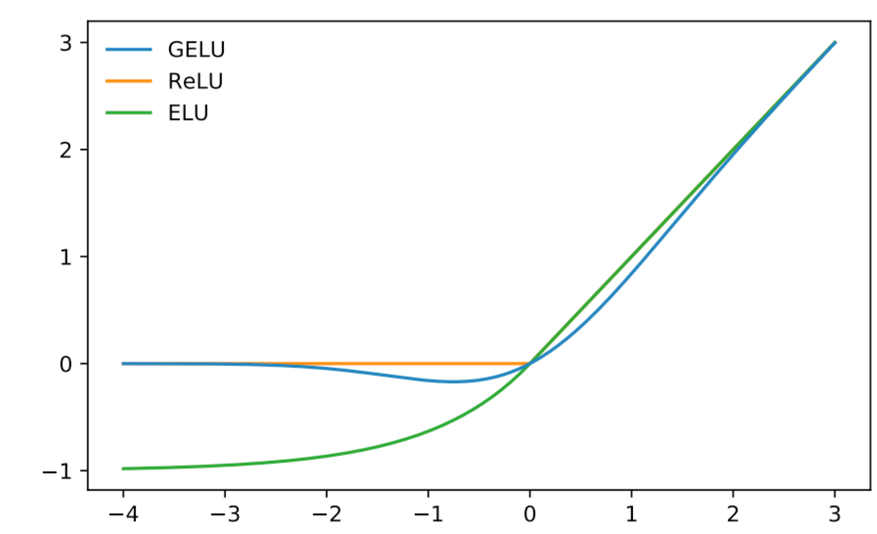
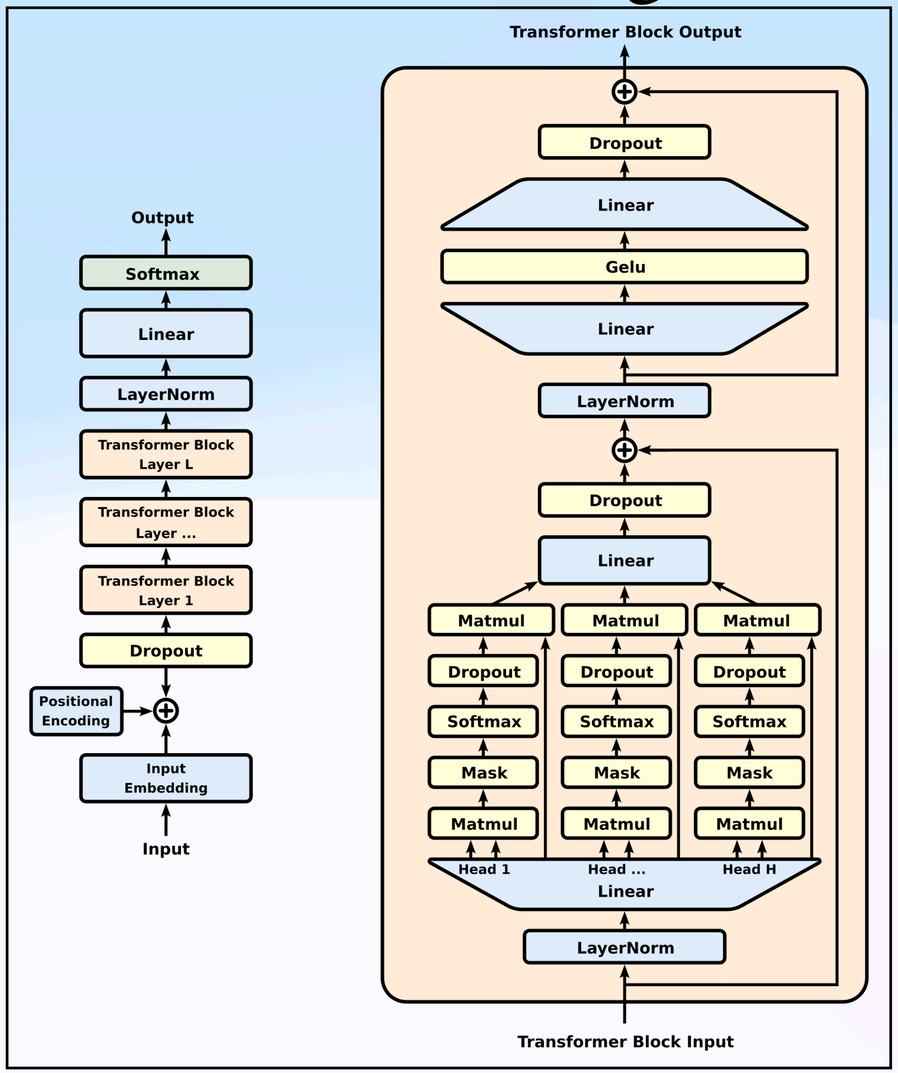
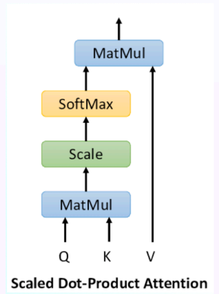


Figure 1: The GELU ($\mu = 0, \sigma = 1$), ReLU, and ELU ($\alpha = 1$).

...

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$





Transformers

Haben klassische Encoder-Decoder Transformer Architekturen noch eine Daseinsberechtigung?

• Nützlich für eigene Daten mit mapping.

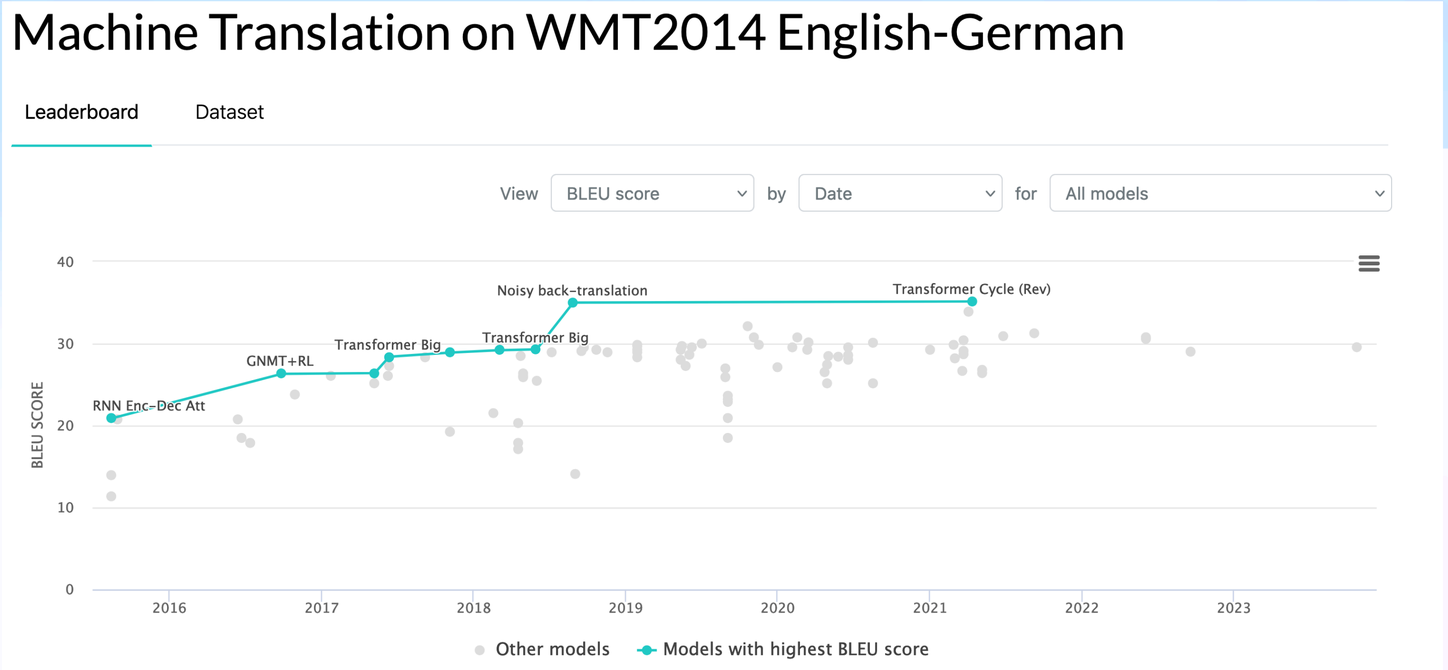
ROUGE summarization score

BLEU machine translation score **BART / T5**

BLEU: A Misunderstood Metric from Another Age

Frage: Wie schneidet GPT bei diesen benchmarks ab?

Elu Benchmarks: Blind user preference.



Attention-Schichten

12288 embedding length (#features) * context length 2048 (#tokens)   unembedding Matrix

Query Embedding
Query Weight Matrix * Embedding i = Query i vector (128) "Any Adjectives in Front (before pos 4)?"

W(Q) * E(i) = Q(i) same for K(i)
128-dim*12000-dim * 12000*1 => 128*1

Key Weight Matrix * Embedding i = Key i vector (128) "Yes, I'm an adjective (at pos 2)"

dot product *alignment matching => large positive numbers 'score'*
condition c = softmax(K dot Q) ≈ 1 "attending" Key muss zur Query passen

masking -∞ => 0 softmax: einfach nicht berechnen! "Zukunft darf nicht vorhergehendes beeinflussen"

ΔE = ∑c*Values = Embedding update conditionally applied (true if attending)
QTK vs KTQ!? softmax row/col wise

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

fluffy & blue modify creature but none else!
W(Value) map 12288*12288? #WV = #WQ + #WK V: how to update embedding
rank of Value up/down 12288-> 128 -> 12288, dimension von query space low rank transform
later tokens cosy (?) GPT *96 heads* only? 96 proposed changes Value down maps grouped in 'output' matrix 96 layers are COPIES? no
T=0, still random selection of lower prob prediction? no, explicit selection of argmax! "Attention is 1/3 of what you need" ;) Chris Olah VcubeX Brit Cruise Art of the Problem

"Next Token Prediction"

"GPT is *just* Next Token Prediction"

Ja, aber:

Kriminalroman, 120 Seiten mit hinwiesen zur Tat.

letzter Satz: "Und der Mörder *ist:* ___"

Was ist das nächste Wort hinter 'ist'? Das hat nicht mehr viel mit einfachen Wort-Verteilungen zu tun, sondern das ganze Buch wird für die Vorhersage 'verstanden'.

'just' some Billion neurons and computations

AUSBLICK

Viel Forschung und viele Optimierung Möglichkeiten.

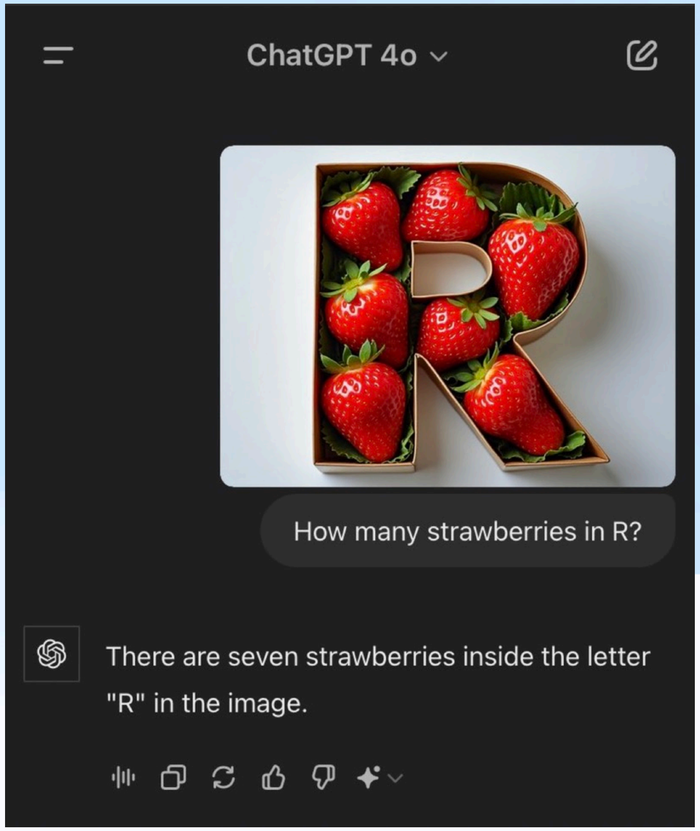
Momentan ist Anzahl Parameter quadratisch zur Kontext Länge.

Kontext Länge GPT 3: 512 … 2048 Tokens

Kontext Länge GPT 4: 8.192 Tokens

GPT-4-32k Version: 32.768 Tokens Kontext
Aktuelle Versionen (GPT-4 Turbo und GPT-4o): 128.000 Tokens
GPT-5 (Input) bis zu etwa 272.000 Tokens, Ausgabe (Output) ≈ 128.000 Tokens
Kombinieren mit RNN? Streaming Context window…

Kontext Länge Gemini 1 - 10 Millionen !! 1ct / 10000 tokens kann teuer werden!




```
ollama run llama3.1  # 8B => 4/16 GB Grafikkarte?
```

```
Generell: kleine LLMs sind für 'dumb tasks' nützlich!
```

MultiQueryAttention: share Query weights between layers
This has a neutral effect on model quality and training speed, but sharply increases inference speed.

learnable positional embeddings instead of fixed, sinusoidal encodings

…
…

AUSBLICK

Named Tensors:

Statt einen Tensor mit verschiedensten Komponenten zu haben

$t_{i\ j\ k\ l\ m\ n}$ **batch**, sequence, layer, heads … breite

man kann und sollte diese Komponenten explizit benamen!

Wie in der **Physik** können die Operationen dann automatisch 'aligned' werden:

Im Grunde wollen wir immer nur in einer **Dimension** das Dot Produkt ausführen, nur halt parallel,

über verschiedene "Stapel"

momentan: **Dimension entfernen, wenn es noch nicht implementiert ist**

Schlechte alternative:

Einstein Summe Tensor Notation (noch kryptischer, aber kürzer: summiere über die 'freien Indizes', hier 'd')

```
torch.einsum("nqhd,nkhd->nhqk", [queries, keys]) # gleich mit umsortiert
```

$t_{i\ j\ k}\ s^{j\ k\ l} := \sum(j,k)\ t_{i\ j\ k}\ s^{j\ k\ l} = z_{i\ l}$

Projektarbeit?

AUSBLICK

Allgemeine Themen

Reinforcement Learning

Reinforcement Learning und LLMs "**reasoning**" Vortrainieren noch wichtiger?

Model lernt **autark** Probleme zu lösen, wir geben nur Belohnung für richtig/falsch
Z.B. 35352*3425325 => "<think>"..."</think>" => answer = "…"

LLMs und Robotik?

Language Diffusion Models

Was ist die logische Essenz vom LLM?
50 Millionen Parameter genug für viele IQ Tests. (HRM paper)

Ausblick

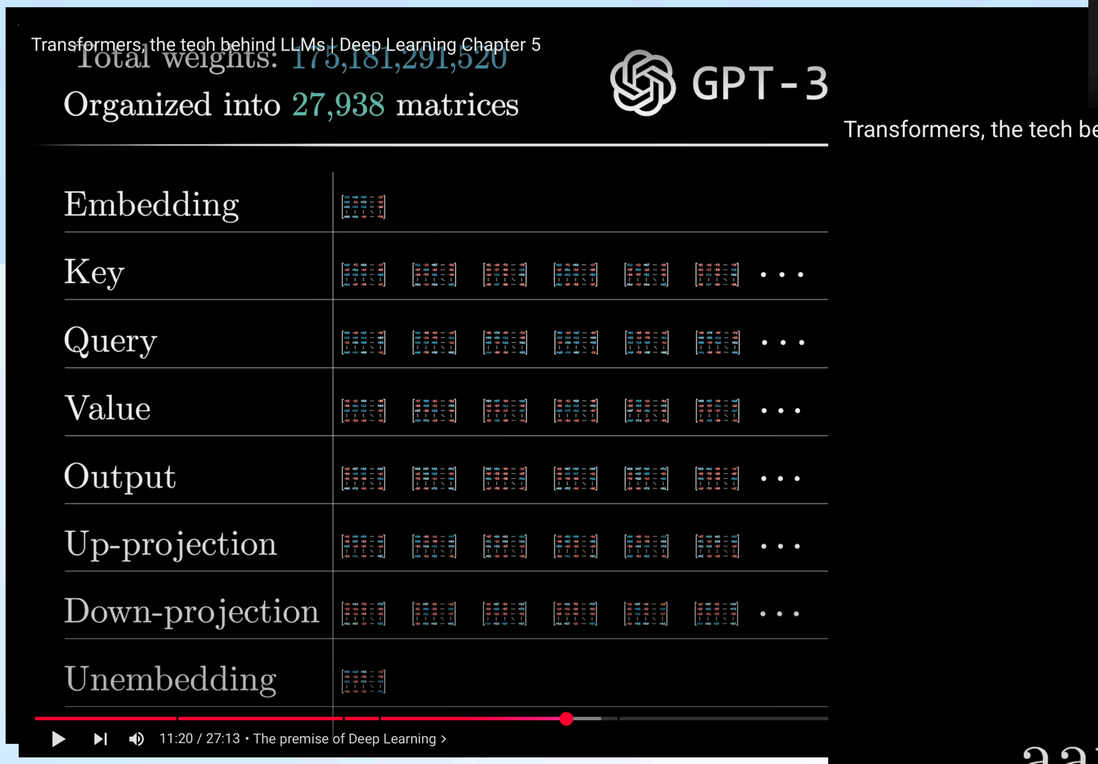
Ausblick:

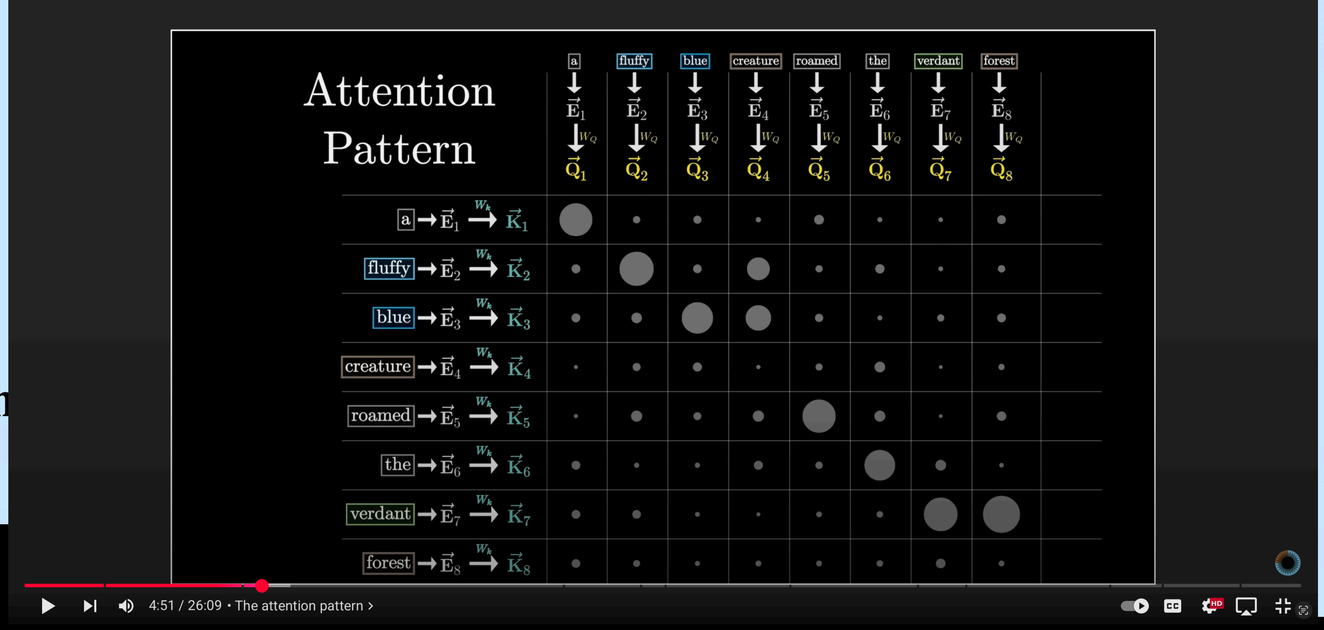
ONYX / safetensors als Austauschformat

AUSBLICK

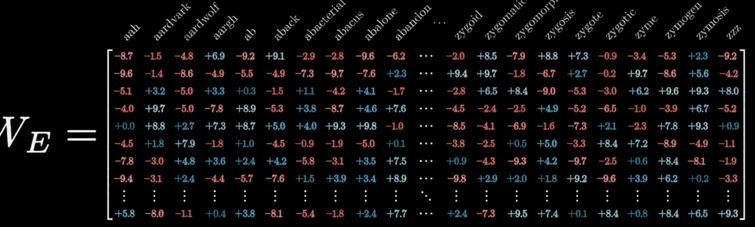
Gaussian Splatting

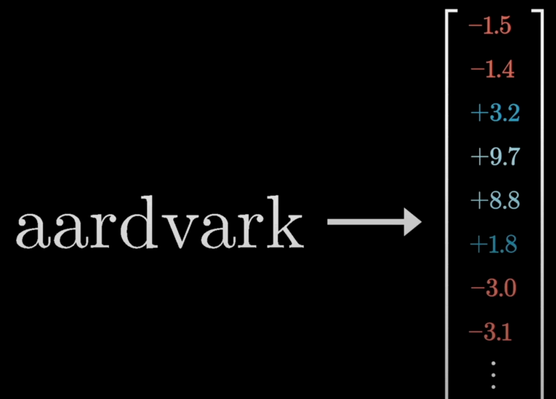
https://www.youtube.com/@TwoMinutePapers (Ach

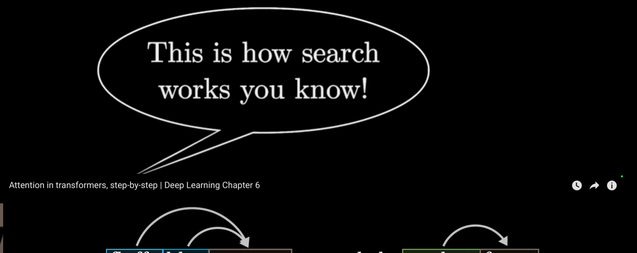




Transformers, the tech behind LLMs | Deep Learning Chapter 5







Deep Learning Kurs - Tag 15

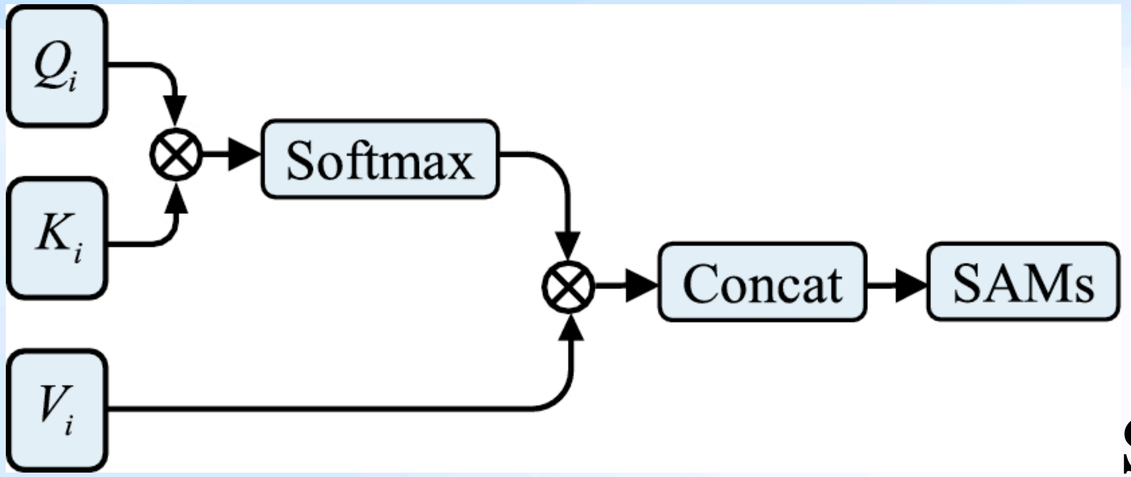
Perfekte alternative Erklärung

3Blue1Brown Animated Math YouTube Channel

https://www.youtube.com/watch?v=wjZofJX0v4M Transformers

https://www.youtube.com/watch?v=eMlx5fFNoYc Attention

https://www.youtube.com/watch?v=9-Jl0dxWQs8 Facts in LLMs



SAM = Self-Attention Mechanisms