



Digital Applications & Data Management

WS25/26

Dr. Jens Kohl



Roadmap Vorlesung

1. Einführung und Übersicht
2. Grundlagen Data Science
3. Vorgehen Data Science Use Case
4. Case Study Data Science
5. Grundlagen unüberwachtes Lernen
6. Grundlagen überwachtes Lernen
(tabellarische Daten)
7. Case Study überwachtes Lernen
(tabellarische Daten)
8. Grundlagen überwachtes Lernen (Bilddaten)
9. Case Study überwachtes Lernen und Transfer Learning (Bilddaten)
10. Grundlagen Generative AI
11. Generative AI mit Texten und Prompt Engineering
12. Agentic AI
13. Ausblick: Machine Learning in der Cloud und Reinforcement Learning

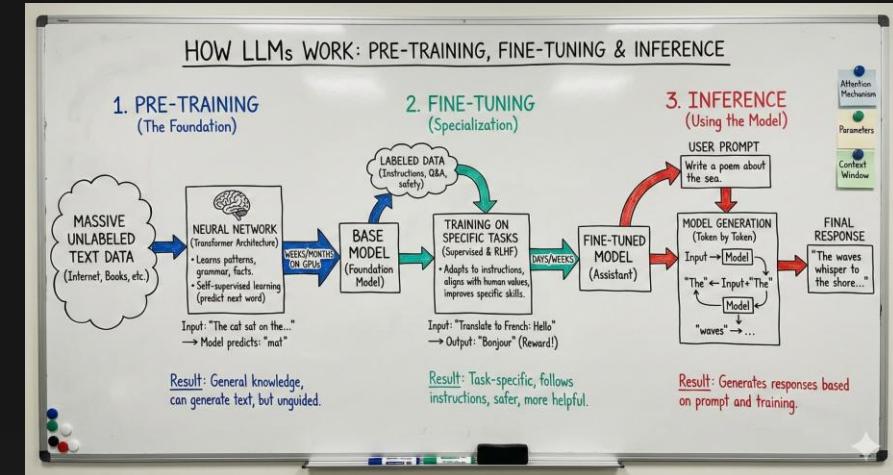
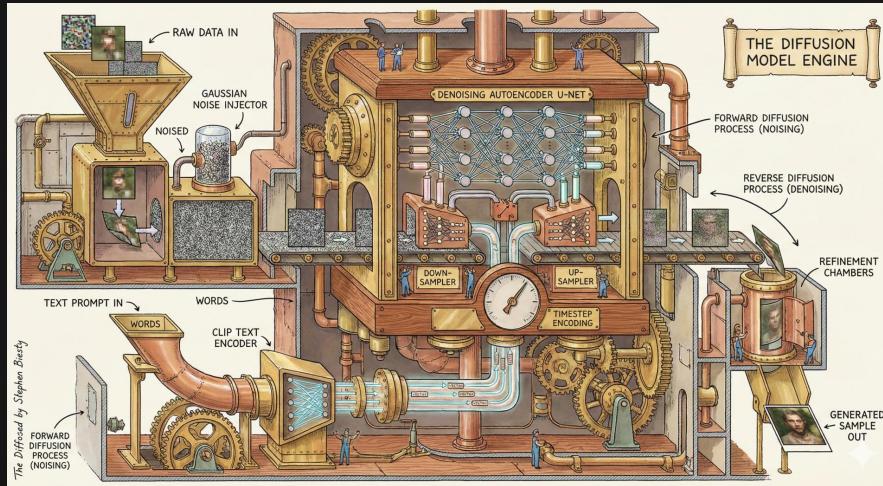


Vorlesung 10: Grundlagen Generative AI



Was machen wir heute?

Motivation



Funktionsweise Generative AI Modelle und Veranschaulichung anhand Generieren von Bildern



Sequentielle Daten



Sequentielle Daten

- Bei sequentiellen Daten können die vorherigen Daten Einfluß auf die aktuellen und nachgelagerten Daten haben.
- Für eine Verarbeitung solcher Daten mittels Machine Learning muß das Modell zeitliche Bedingungen abbilden können.
- Das Modell muß also auf aktuellen Daten, aber auch auf vorherige Daten, zugreifen.
- Die bisher betrachteten Modelle bilden dies nicht ab, wir brauchen also Modelle mit Rückkopplung oder „Gedächtnis“.

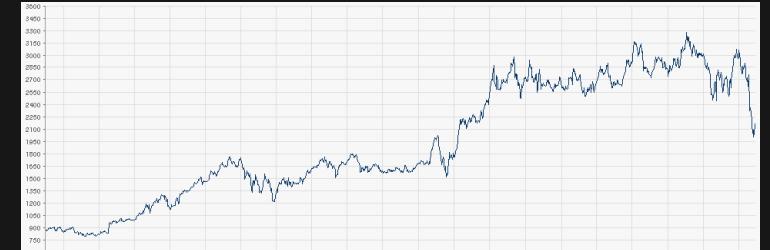
Bei sequentiellen Daten haben vorherige Daten Einfluß auf die aktuellen und nachfolgenden Daten



Sequentielle Daten

Beispiele

- Zeitreihen: Aktienkurse, Messungen, Temperaturkurven, ...
- Video: Folge von einzelnen Bildern (Frames per Second)
- Texte: Folge von einzelnen geschriebenen Wörtern
- Sprache: Folge von einzelnen gesprochenen Wörtern





Generative AI für Bilder/ Video



GENERATIVE AI

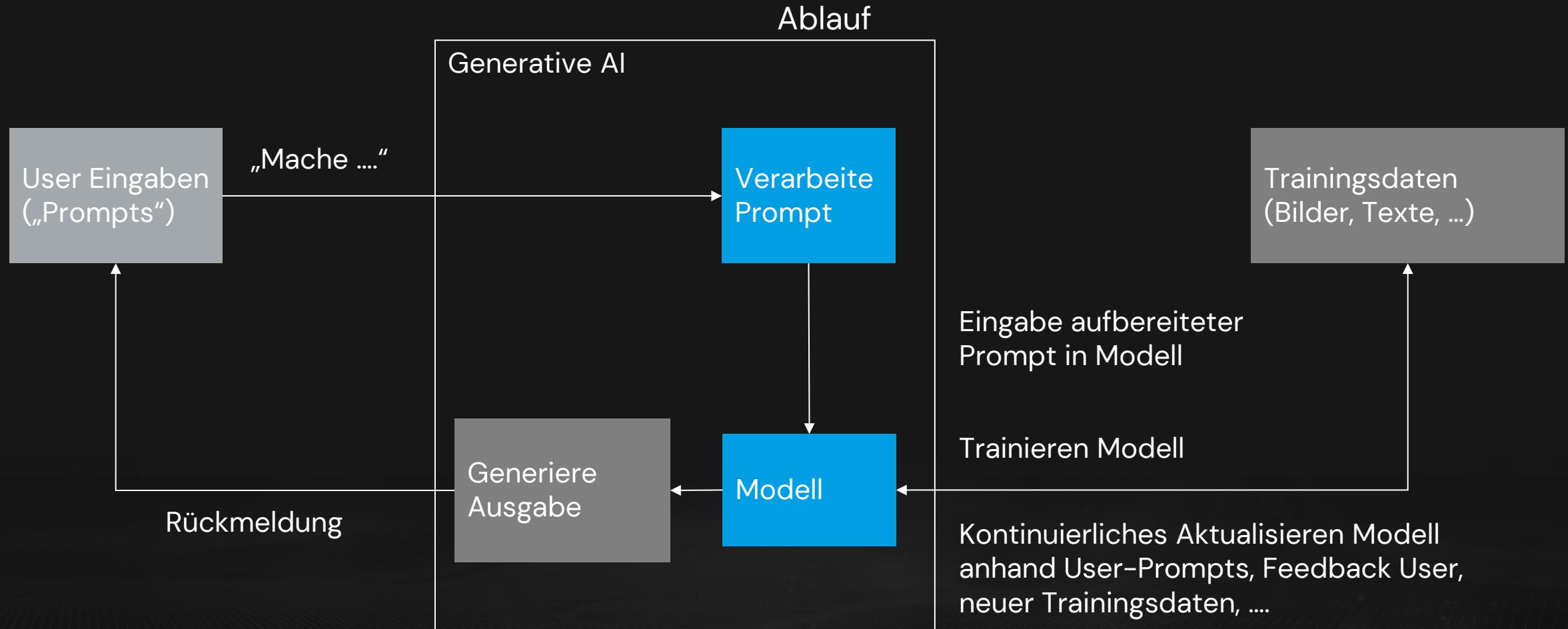
Motivation

- Bei den bisher betrachteten Verfahren erhalten wir durch den Einsatz von Machine Learning einen Informationsmehrwert.
- Wir lernen neue Zusammenhänge, erstellen aber keine neuen Daten/ Content bzw. ändern die vorhandenen Daten nur leicht.

Ziel Generative AI: Schaffen Mehrwert oder Entlasten Anwender durch Erstellen hochwertiger Ausgaben



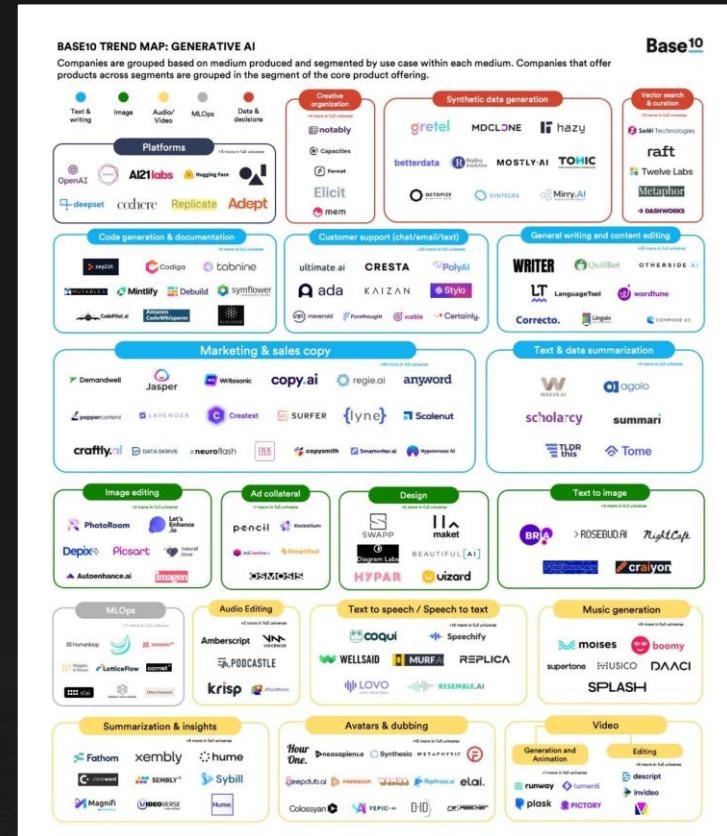
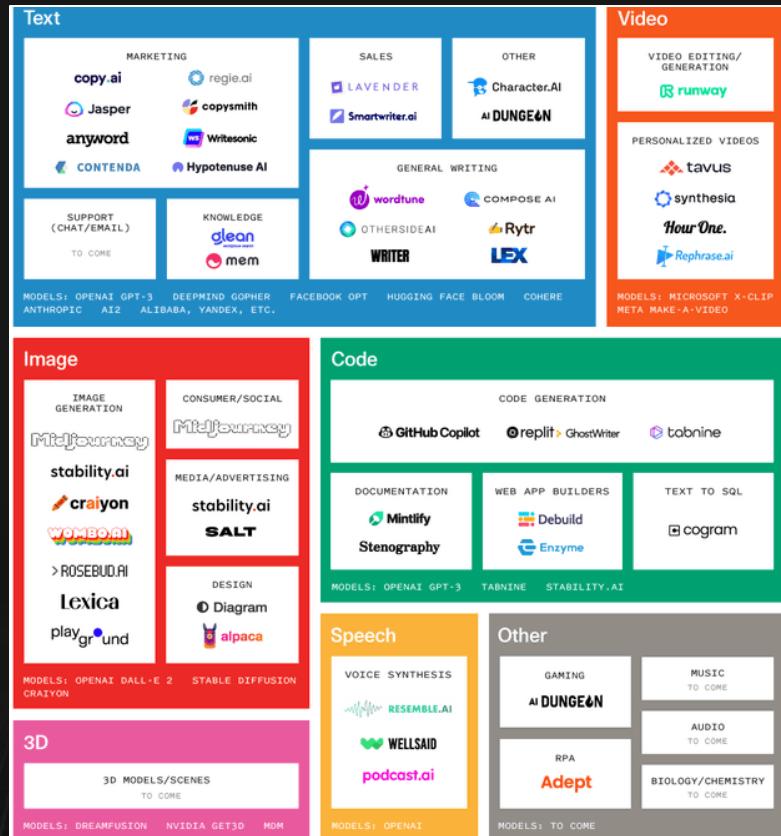
Generative AI





Generative AI

Übersicht bekannte Tools und Apps (Stand 2023)



The Generative AI Landscape			foundation capital
		Example Use Cases	Example Startups
Business Function	Marketing	Copywriting, SEO optimization; true personalization	 Jasper  copy.ai  WRITER
	Sales	SDR automation, sales coaching	 Oliv  regie.ai  PROLAI
	Customer Success	User insights, answering tickets	 Forethought  CRESTA  symbl.ai
	HR	Job description writing, interviewing, performance reviews, AI coach, training	 MANAGE BETTER  onicop  CONVERZAI
	Legal	Document drafting, synthesis, legal to non-legal translation	 casetext
	Ops	Research, search, synthesis, knowledge retrieval and management, manual tasks	 YOU  mem  hello
	Finance	Data entry, data summarization	
	Internal Tooling	Natural language to code generation; truly custom tools that can be built by business users	 Adept  māyā
	Asset Creation	Next gen Wikipedia, gaming studios, movie studios, news channels	 Midjourney  runway  Semberush
Developer	Frontend Development	AI can generate unlimited designs and iterate until it finds "best"	 beamagine  Debuild
	Coding	Generate test cases and create test automation	 replit  Modeme  GitHub Copilot
	Training	Model Training	 Replicate  Hugging Face  Lambda



Generative AI

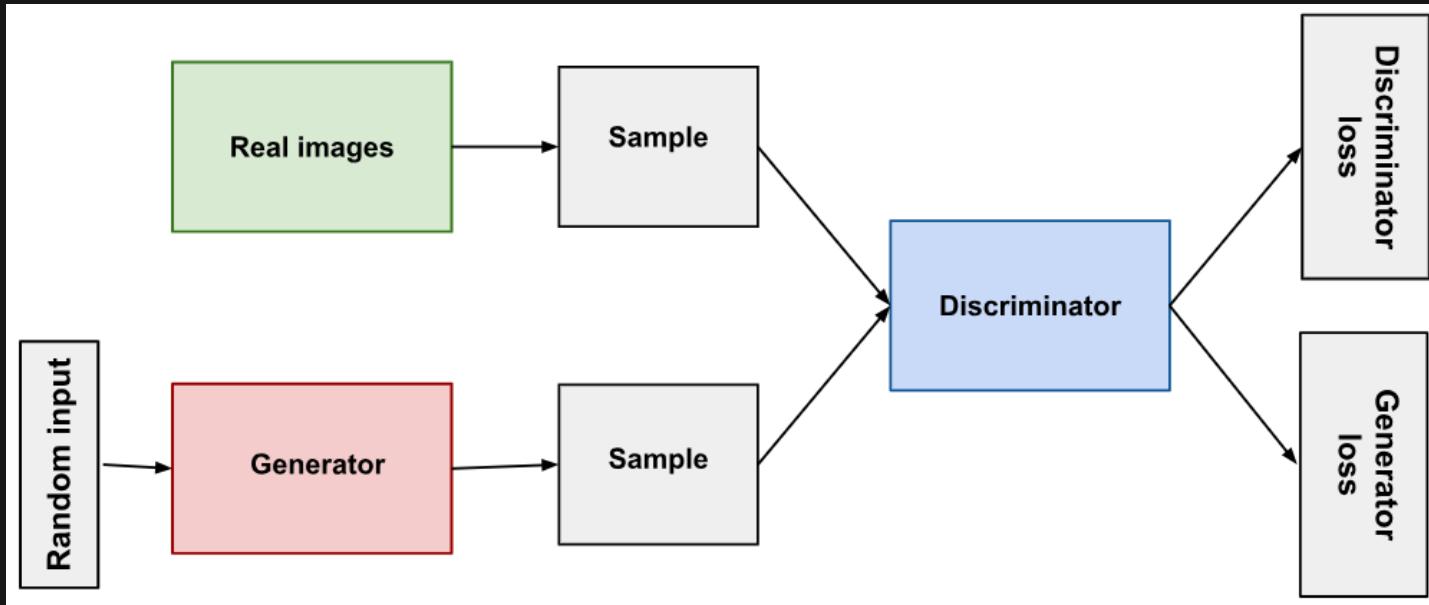
Übersicht bekannte Verfahren für Bilder

- 2014: Generative Adversarial Networks
- 2015: Neural Style Transfer
- Diffusion Models
 - 2020 Denoising Diffusion Probabilistic Model
 - 2021: Open Ai: Dall-E
 - 2022: Midjourney, Stable Diffusion
 - 2024: Flux sowie viele weitere Tools
 - 2025: Nano Banana



Generative AI

GENERAL ADVERSIAL NETWORKS: ÜBERSICHT.



Paper: Goodfellow, Ian et al (2014). Generative Adversarial Nets. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014)

Vorgehensweise: Generator und Discriminator sind jeweils neuronale Netzwerke.

- Generator erstellt Bilder, die Input für den Discriminator sind. Dabei lernt er, möglichst genaue, plausible Bilder zu erstellen.
- Discriminator lernt, zwischen realen Bildern und den „falschen“ vom Generator zu unterscheiden. Die Rückmeldung bei der Erkennung nutzen beide Netzwerke zur Verbesserung Erkennung und Generierung per Backpropagation.



Generative AI

GENERAL ADVERSIAL NETWORKS: Anwendungsbeispiele

Text-to-Image

The small bird has a red head with feathers that fade from red to gray from head to tail

Stage-I images



Stage-II images

This bird is black with green and has a very short beak

Stage-I images



Stage-II images

Han Zhang et al., „StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks”, 2016. [Link](#)

Modeindustrie: Virtual try on



Bergmann: „Generative Models in e-Commerce“, 2020. Vortrag an LMU, [Link](#)

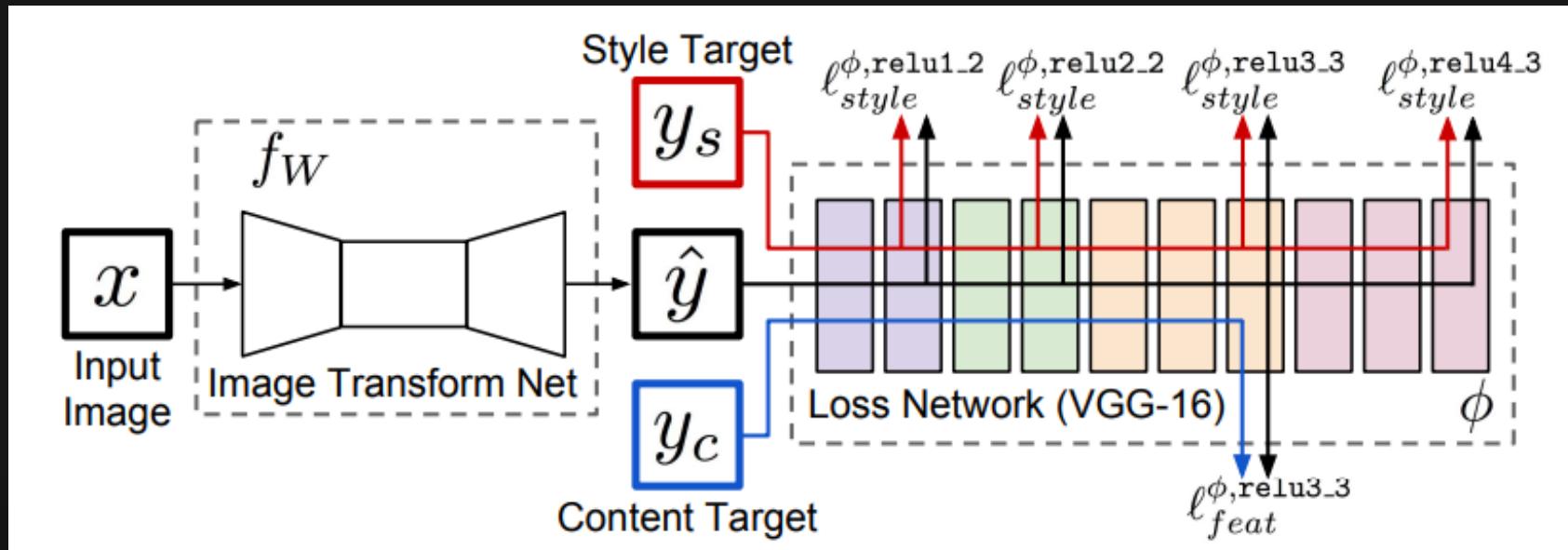
Weitere Anwendungsbeispiele: [Link](#), [Link](#)

- Erstellen/ Vorhersage nächster Frames für Videos
- Generieren von 3D-Objekten aus 2D Bildern



Neural style transfer

Übersicht



Papers:

- Gatys, Leon A et al. "A Neural Algorithm of Artistic Style", [arXiv:1508.06576](https://arxiv.org/abs/1508.06576), 2015,
- Johnson et al., „Perceptual Losses for Real-Time Style Transfer and Super-Resolution“, 2016. [Link](#)

Code-Notebook: [Hier](#), [Hier](#)

Vorgehensweise: „We train an image transformation network to transform input images into output images. We use a loss network pretrained for image classification to define perceptual loss functions that measure perceptual differences in content and style between images. The loss network remains fixed during the training process“ (Modifikation Transfer Learning)



Neural style transfer

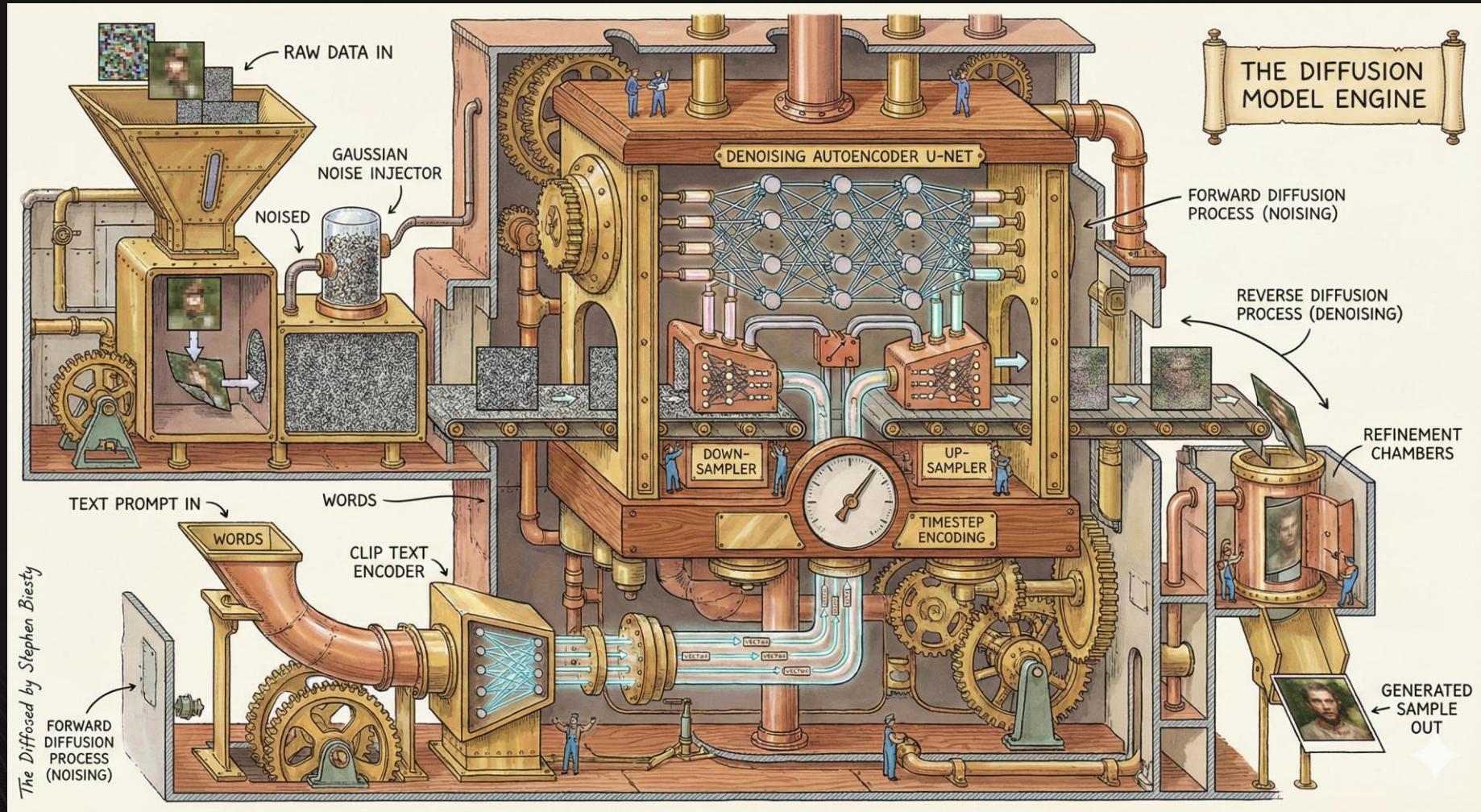
Anwendungsbeispiele



Modifikation Transfer Learning: Nehme Stil eines bekannten Bildes und zeichne ein anderes Bild in diesem Stil



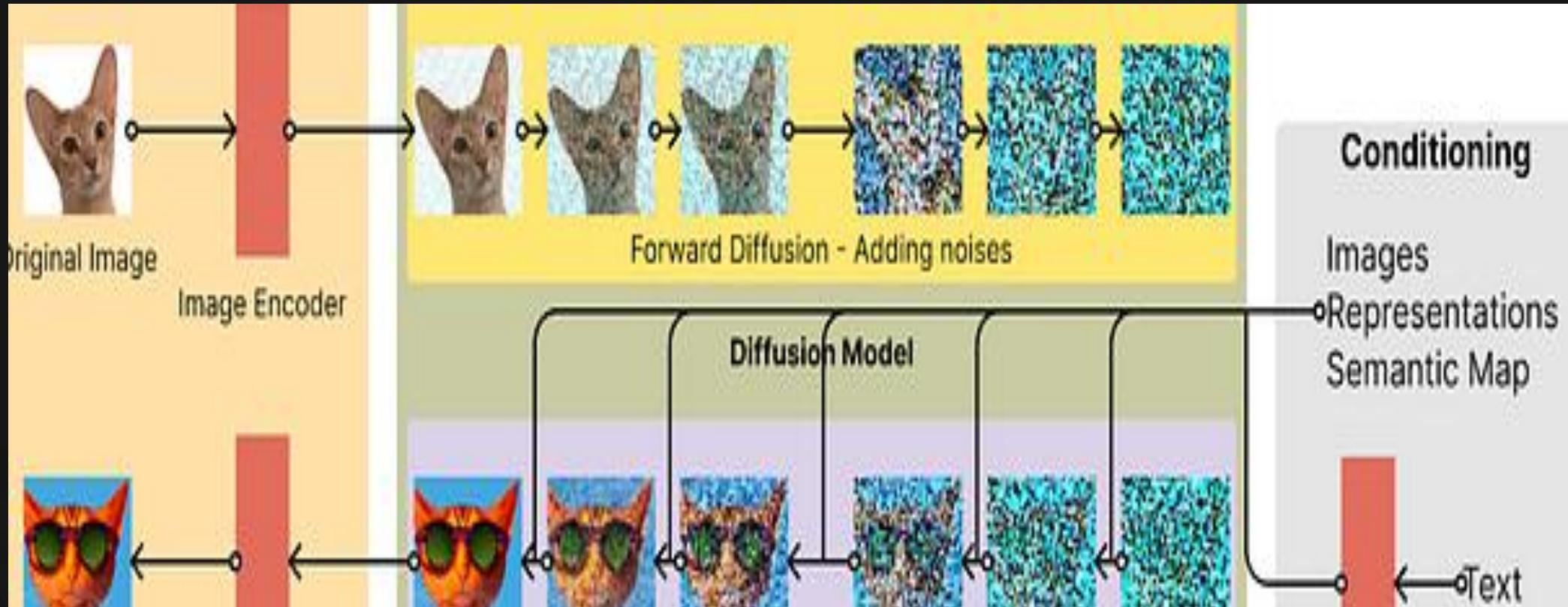
Diffusion model





Stable diffusion/ Flux

Übersicht



Paper: Rombach et al., „High-Resolution Image Synthesis with Latent Diffusion Models“, [Link](#), 2021.

Source Code: [Github](#) (Stable Diffusion), [Github Flux](#)

Online aufrufbar unter: [StableDiffusionWeb](#), [StableBoost.AI](#), [Flux](#)



<https://poloclub.github.io/diffusion-explainer/>



Stable Diffusion/ Flux

Beispiele



(masterpiece, digital art:1.3) , highly intricate double exposure art inspired by Yuumei, (close-up of a person's eye:1.2) superimposed on a bustling city street, cyberpunk aesthetic, trending on CGSociety, multiple exposure technique, overlaying textures with glowing holographic elements, technicolor palette, bright city lights illuminating the scene, reminiscent of "The Matrix" film's Pinterest color scheme, high definition rendering, colorful dream-like atmosphere, elaborate details in the digital art, futuristic and vibrant, captivating visual narrative.



Stable Diffusion/ Flux

Beispiele

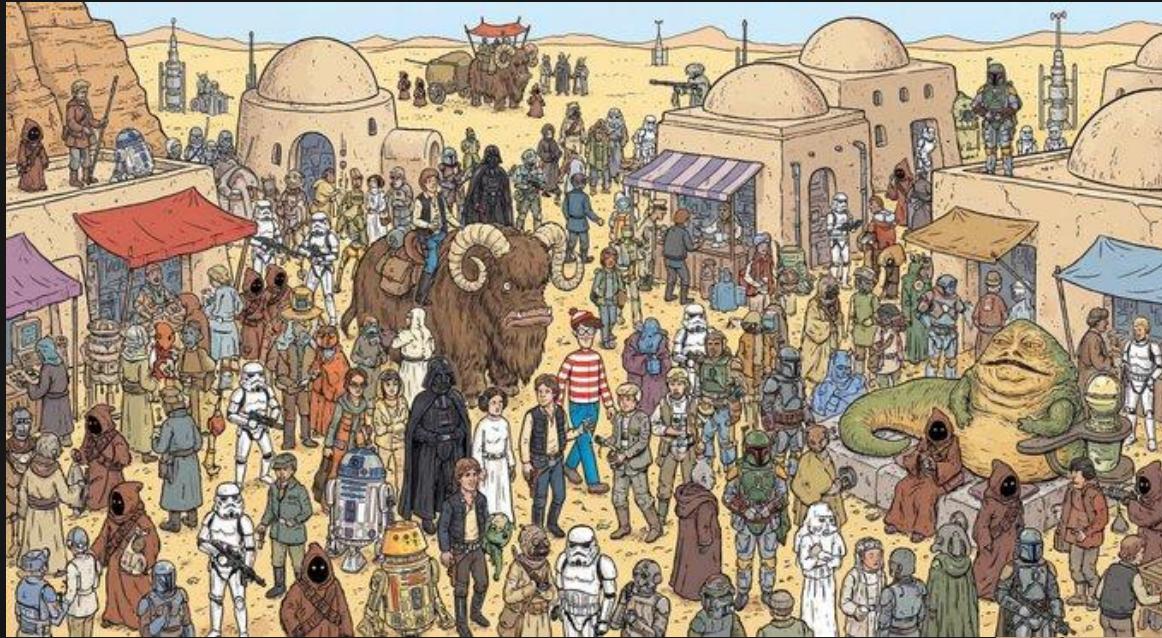


A photograph from [the opening of the Eiffel Tower in Paris in 1889] captures the monumental occasion with a grand, historic ambiance. The image shows [the towering iron structure, still under construction, against a backdrop of a bustling Parisian skyline]. Crowds of people, dressed in late 19th-century attire, are gathered around, marveling at the iconic new landmark. The photograph exudes a sense of excitement and wonder as [Parisians celebrate the unveiling of this groundbreaking architectural feat].



Nano Banana

Some examples



"A where is waldo image showing all Star Wars characters on Tatooine"

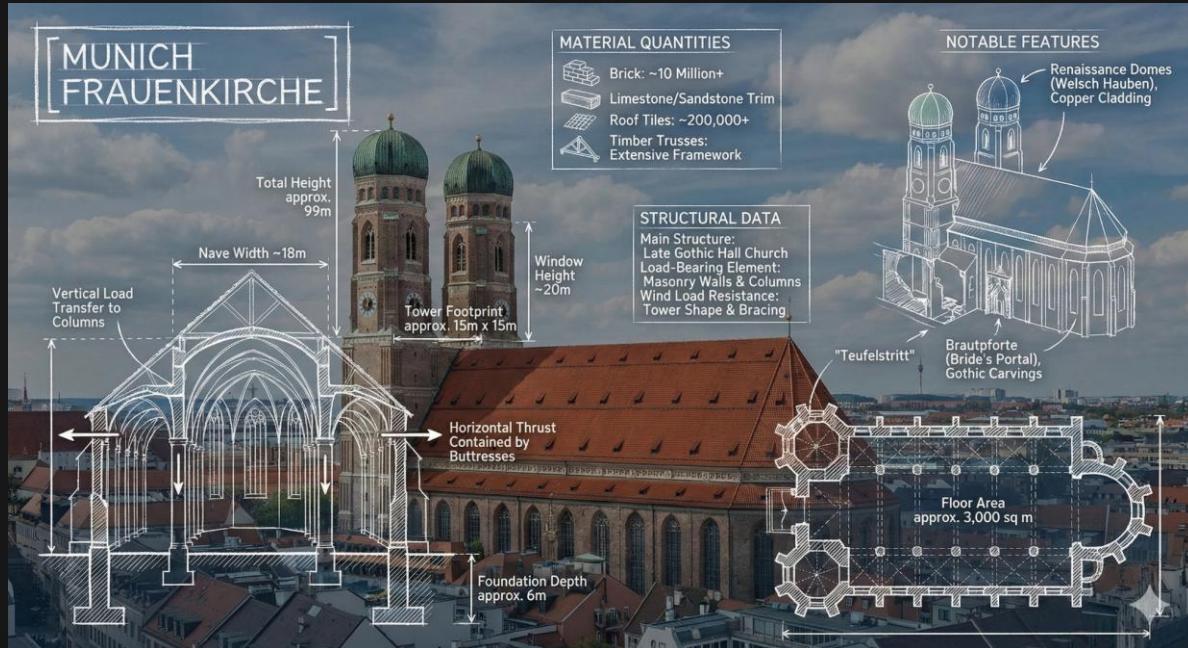
"A where is waldo image showing all Donald Duck characters before uncle scrooge's money bin"



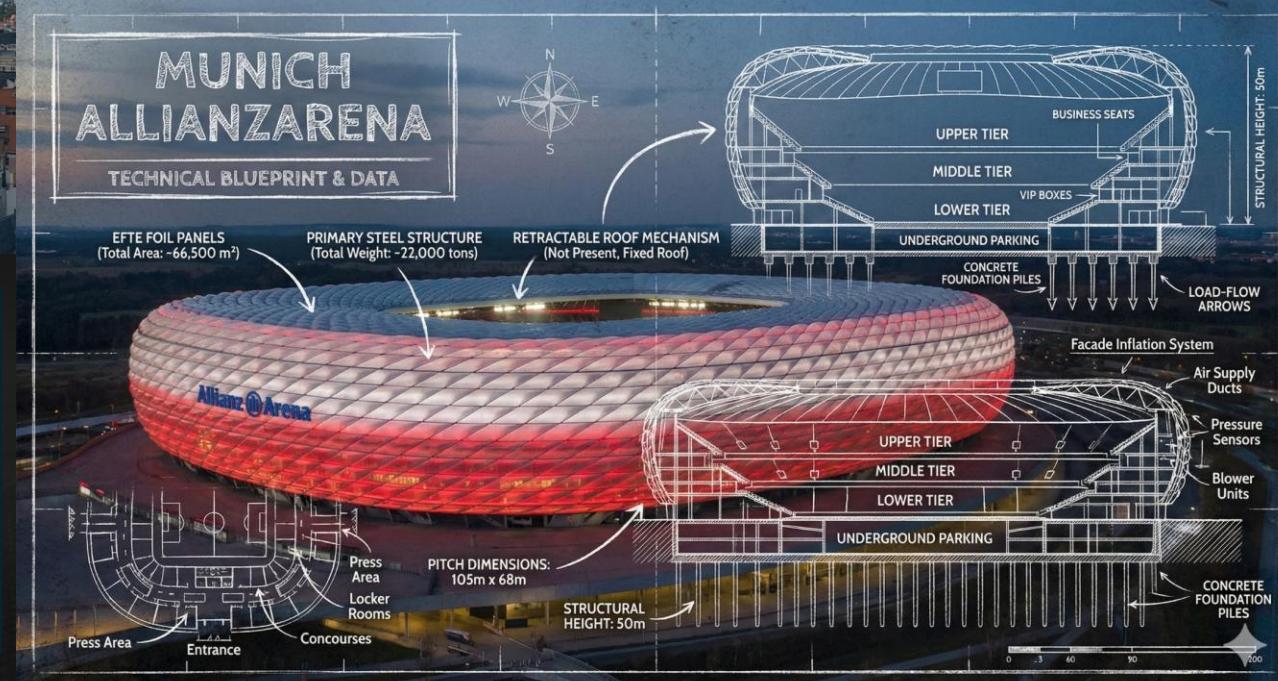


Nano Banana

Some examples



Create an infographic image of [LANDMARK], combining a real photograph of the landmark with blueprint-style technical annotations and diagrams overlaid on the image. Include the title “[LANDMARK]” in a hand-drawn box in the corner. Add white chalk-style sketches showing key structural data, important measurements, material quantities, internal diagrams, load-flow arrows, cross-sections, floor plans, and notable architectural or engineering features. Style: blueprint aesthetic with white line drawings on the photograph, technical/architectural annotation style, educational infographic feel, with the real environment visible behind the annotations”





Nano Banana

Using the uploaded AWS Q3 FY2026 earnings PDF as the data source, create a clean, premium, and modern infographic in the style of Nano Banana image prompt

DESIGN REQUIREMENTS:

- X-optimized aspect ratio: 1920x1080 (16:9) or 1500x2000 (3:4)
- High-end background aesthetic:
 - soft light gray or off-white
 - subtle gradient
 - gentle noise texture
 - smooth layered shadows
 - avoid harsh dark themes
- Clean, modern finance/tech design (Apple Keynote x Nano Banana)
- Rounded floating cards with depth, soft shadows, and gentle glow
- Premium sans-serif fonts (SF Pro, Inter, Helvetica Neue)
- Clear visual hierarchy with bold section headers and balanced spacing
- Use NVIDIA-inspired accents: #76B900 green, mint, teal, soft blue
- Include modern iconography: revenue, income, GPU, cloud, robotics
- Add small arrows, percentage badges, and microcharts for visual context
- CONTENT TO AUTO-EXTRACT FROM THE PDF (do NOT include)
 - Total revenue, YoY growth, QoQ growth
 - Operating income, net income, EPS
 - GAAP and non-GAAP gross margins
 - Segment-level performance:
 - Data Center
 - Gaming & AI PC
 - Professional Visualization
 - Automotive & Robotics
 - Key growth drivers (Blackwell, cloud GPU demand, AI ecosystem)
 - Capital allocation (buybacks, dividends, cash, remaining authority)
 - Critical risks and headwinds
 - Q4 FY2026 guidance (revenue outlook, margin outlook)
- STYLE NOTES:
 - Premium, minimalist, futuristic finance aesthetic
 - Keep the design uncluttered and highly readable on mobile
 - Use soft lighting, layered cards, and subtle neon accents
 - Ensure the final image looks scroll-stopping and share-worthy
- OUTPUT:
 - A high-resolution, polished, and visually striking X/Twitter-ready financial infographic that uses ONLY the metrics extracted dynamically from the uploaded PDF."

AMAZON Q3 FY2025 EARNINGS: Strong Growth & AI Momentum, Despite Special Charges

TOTAL NET SALES

\$180.2B

+13%
YoY

[cite: 4]

OPERATING INCOME

\$17.4B

Flat
YoY

[cite: 8]

NET INCOME

\$21.2B

Includes \$9.5B gain from Anthropic investment

+38%
YoY

[cite: 14, 15]

DILUTED EPS

\$1.95

+37%
YoY

[cite: 14]

SEGMENT PERFORMANCE

NORTH AMERICA

Sales: **\$106.3B** (+11% YoY)

Op. Inc: **\$4.8B**

INTERNATIONAL

Sales: **\$40.9B** (+14% YoY)

Op. Inc: **\$1.2B**

AWS

Sales: **\$33.0B** (+20% YoY)

Op. Inc: **\$11.4B**

Growth accelerated to 20.2% YoY

KEY GROWTH DRIVERS



AI & CHIPS: Strong momentum. Trainium2 grew 150% QoQ. Project Rainier launched (500k chips)

[cite: 19, 23, 24]



AWS ACCELERATION: Strong AI demand. Added 3.8 GW power capacity

[cite: 20, 26]



DELIVERY SPEED: Fastest Prime speeds. Same-Day grocery to 1,000+ cities

[cite: 20, 42]

CAPITAL ALLOCATION & CASH FLOW

OPERATING CASH FLOW (TTM):

\$130.7B
(+16% YoY)

[cite: 16]

FREE CASH FLOW (TTM):

-\$14.8B
(Decreased from \$47.7B)

[cite: 17, 18]

CRITICAL RISKS & HEADWINDS



ECONOMIC & GEOPOLITICAL:

FX, global conditions, trade policies



OPERATIONAL CHALLENGES:

Labor constraints, competition, fulfillment & data center risks

[cite: 65]

[cite: 65, 75, 76]

Q4 FY2025 GUIDANCE

NET SALES OUTLOOK:

\$206.0B - \$213.0B
(+10% to +13% YoY)

[cite: 67]

OPERATING INCOME OUTLOOK:

\$21.0B - \$26.0B
(vs. \$21.2B in Q4 2024)

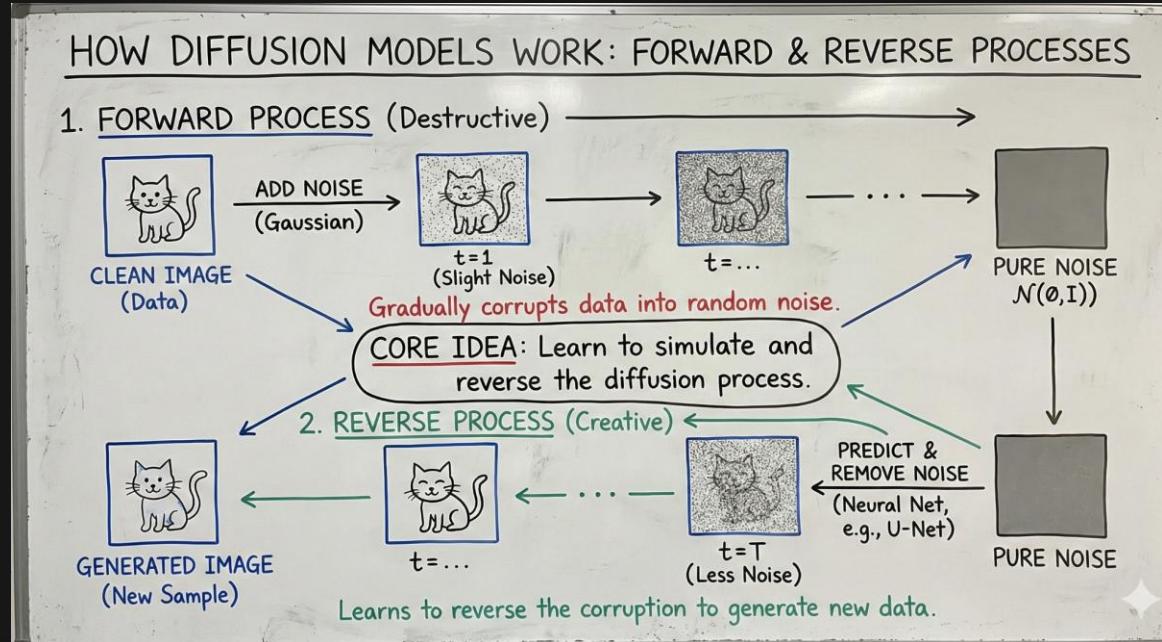
[cite: 68]

Source: Amazon Q3 FY2025 Earnings Release

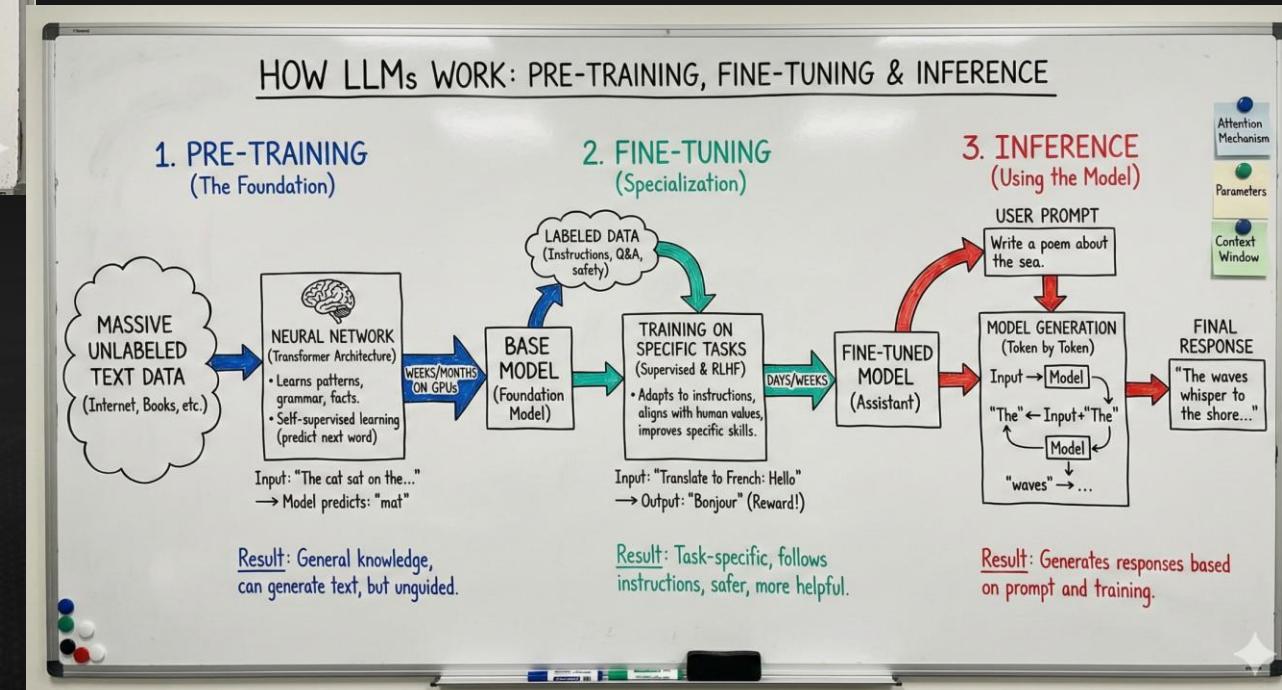


Nano Banana

Some examples



create an image of a whiteboard that summarizes how diffusion models work using diagrams, arrows, colors, and captions explaining the core idea





Nano Banana

A fashion photograph of a woman with long dark hair, wearing a black sleeveless ribbed knit top and a matching black ribbed knit mini skirt. She is leaning her back against a dark, ornate metal lamppost on a city street. The background is softly blurred, showing classical architecture and other people in the distance. The lighting is soft and diffused, creating a cinematic feel. The overall aesthetic is hyperrealistic and photorealistic.





Nano Banana



generate a detailed football scene as an image and label every object with english words in one line, german in the next line and spanish in the third line



Nano Banana



create an image about the movie "Casablanca" retold through a series of at least 8 Polaroid photos pinned to a cork board. Each photo captures a key moment, with simple captions below. arrange the photos in a loosely chronological path across the board, using colored strings to connect events and characters. Light the scene warmly to evoke nostalgia. Include incidental details, coffee cup rings, paper clips, handwritten notes, for authenticity.





Nano Banana

Create a hyper-realistic detailed recipe infographic showcasing a single, perfect Tequila Sunrise cocktail, detailing the density-driven distinct layers: the deep ruby red grenadine settled at the bottom, the vibrant transition into fresh orange juice in the middle, and the translucent tequila float at the top. Include high-definition textures of condensation on the highball glass, suspended ice cubes, and a fresh orange slice garnish.





More examples:

https://colab.research.google.com/github/google-gemini/cookbook/blob/main/quickstarts/Get_Started_Nano_Banana.ipynb



Making movies

- <https://x.com/maxescu/status/1992320390031651018>



Exkurs: wir bauen einen digitalen Avatar





Exkurs: wir bauen einen digitalen Avatar

<https://twitter.com/CharaspowerAI/status/1854222578849579222>



Literatur und weitere Quellen

LSTM:

- <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Hochreiter, Schmidhuber: Long short-term memory, 1997 (meistzitierte AI-Paper!).

Attention:

- <https://distill.pub/2016/augmented-rnns/>

Transformers:

- <https://e2eml.school/transformers.html>
- <https://nlp.seas.harvard.edu/annotated-transformer/>
- <https://www.tensorflow.org/text/tutorials/transformer>
- <https://jalammar.github.io/illustrated-transformer/>
- <https://levelup.gitconnected.com/understanding-transformers-from-start-to-end-a-step-by-step-math-example-16d4e64e6eb1>

Large Language Models:

- <https://mark-riedl.medium.com/a-very-gentle-introduction-to-large-language-models-without-the-hype-5f67941fa59e>

Stable Diffusion:

- <https://bootcamp.uxdesign.cc/how-stable-diffusion-works-explained-for-non-technical-people-be6aa674fa1d>
- <https://medium.com/@steinsfu/diffusion-model-clearly-explained-cd331bd41166>
- <https://analyticsindiamag.com/10-awesome-google-colab-notebooks-on-stable-diffusion/>
- <https://jalammar.github.io/illustrated-stable-diffusion/>

GPT:

- Karpathy: Entwicklung GPT2-Clone in 600 Zeilen Python:
<https://github.com/karpathy/nanoGPT> und
<https://www.youtube.com/watch?v=kCc8FmEb1nY>
- <https://www.youtube.com/watch?v=kCc8FmEb1nY>
- Jay Mody: <https://jaykmody.com/blog/gpt-from-scratch/#putting-it-all-together> und
<https://github.com/jaymody/picoGPT/blob/main/gpt2.py>
- <https://bootcamp.uxdesign.cc/how-chatgpt-really-works-explained-for-non-technical-people-71efb078a5c9>
- <https://medium.com/@fareedkhandev/create-gpt-from-scratch-using-python-part-1-bd89ccf6206a>
- Visualisierung verschiedener Modelle unter: [Link](#)
- Llama: Source Code unter [Link](#), Tutorials unter [Link](#), [Link](#)

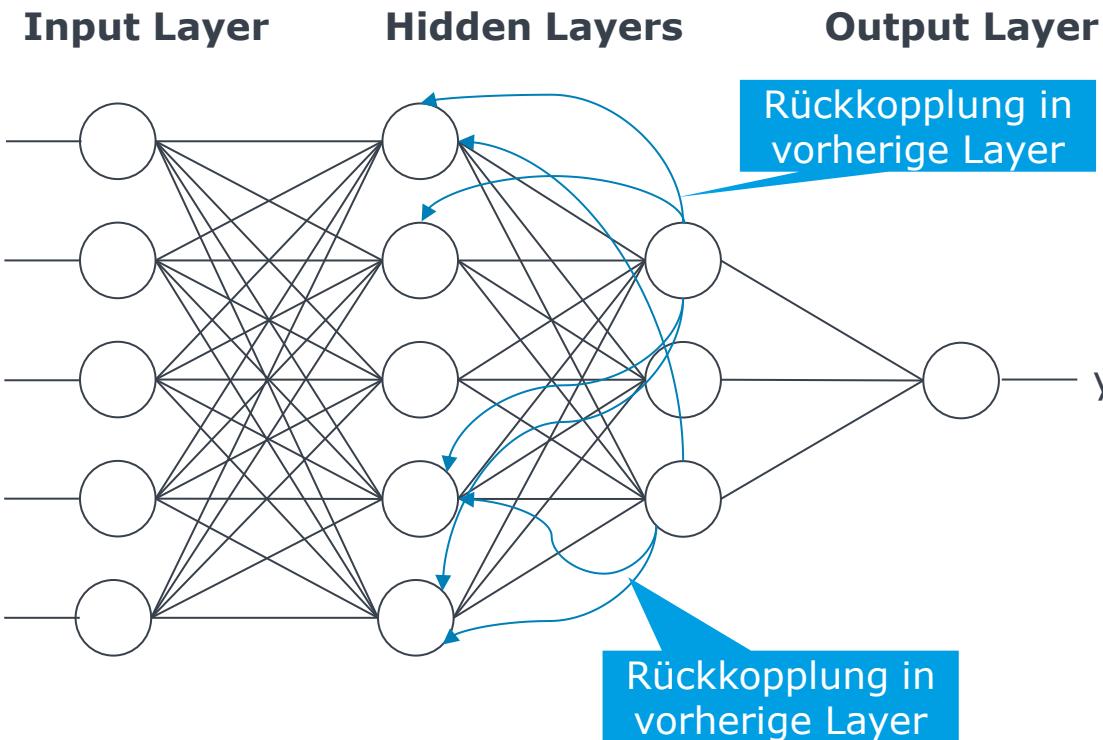


Backup

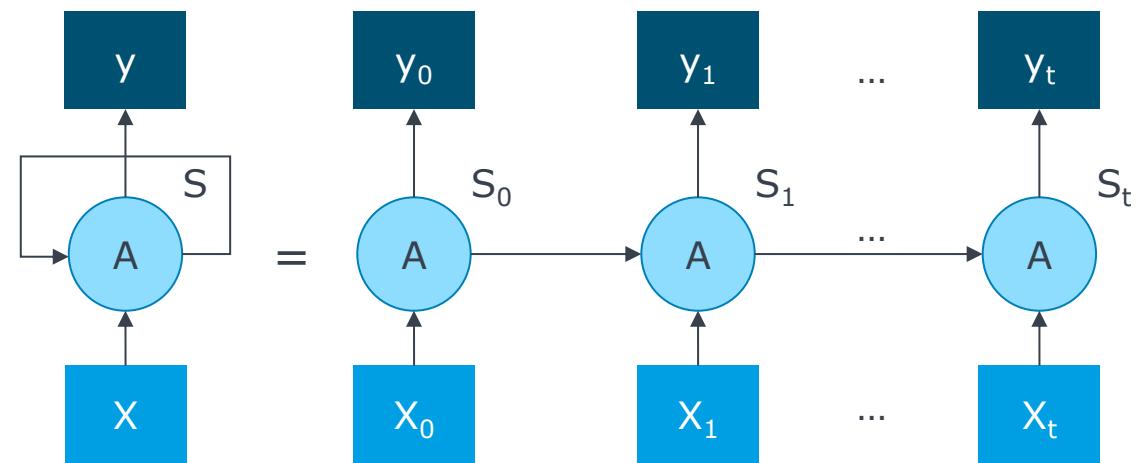


DETAILLIERUNG REKURRENTE NEURONALE NETZE

STRUKTUR REKURRENTE NEURONALE NETZE.



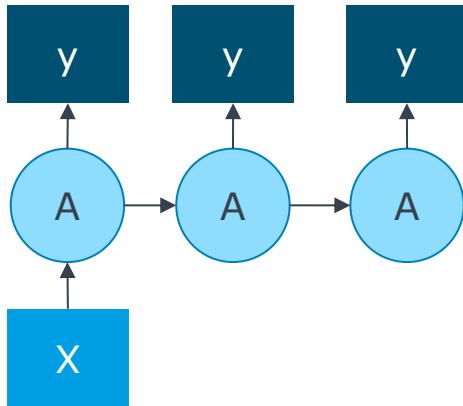
Detaillierung Struktur RNN über Zeitschritte t „Querschnittsbild“



- Eingaben X_i : Sequenz wird auf Inputs aufgeteilt.
- Hidden State S_i : „Kurzzeitgedächtnis“. Abhängig vom Input x_i sowie vorherigen Zustände S_i . Überträgt auch (gewichtet) Informationen zum nächsten Schritt.
- A: Neurales Netzwerk inklusive Rückkopplung Ergebnisse.
- Ausgaben y_i : Ausgaben/ Prädiktion des Netzwerks.

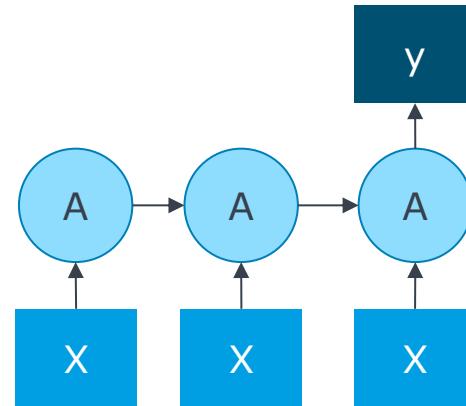
(GROB-)STRUKTUR RNN ABHÄNGIG VOM EINSATZGEBIET.

Ein Input, viele Outputs



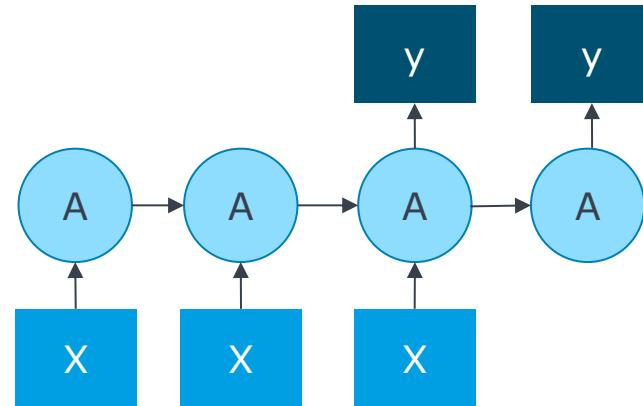
Sequence Output:
 Bspw. automatisierte
Bilderkennung und -beschreibung
 (1 Bild mit n Wörtern beschreiben)

Viele Inputs, ein Output



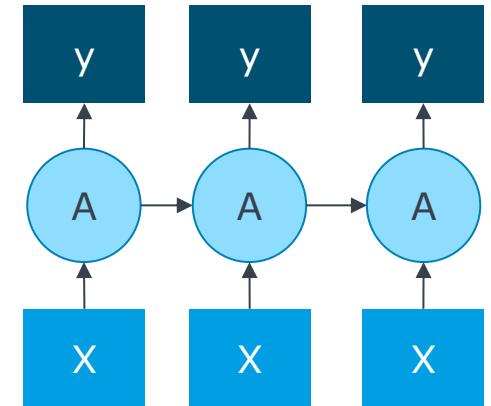
Sequence Input:
 Bspw. automatisierte
Sentiment Analysis/ Gefühlsanalyse
 (1 Satz wird analysiert ob positive/ negative Aussagen)

Viele Inputs, viele Outputs



Sequence input and output:
 Bspw. Automatisiertes **Übersetzen** **Texte** (Deutsch – Englisch), aber Input und Output können verschiedene Längen haben.

Viele Inputs, viele Outputs



Gleich lange Sequence input and output:
 Bspw. Videoklassifikation, jeder Frame wird gelabelt

- Frage: welche Struktur wird eingesetzt für**
- Bewerten ob positives/ negatives Kundenfeedback
 - Automatisiertes Übersetzen eines Satzes
 - Labeln Videoframes (je Frame)
 - Beschreiben eines Bildes mit Wörtern



BEKANNTE MACHINE LEARNING VERFAHREN FÜR TEXTE

ÜBERSICHT BEKANNTE MACHINE LEARNING VERFAHREN.

- LSTM (1997)¹:

- Rekurrentes Neuronales Netzwerk, entwickelt von Sepp Hochreiter und Jürgen Schmidhuber an der TU München.
- aktuell (noch?) eines der häufigst eingesetzte Verfahren für sequentielle Daten, bspw. Spracherkennung in Handys ([Link](#)).
- Kann viel längere Zeitsequenzen verarbeiten als normale RNN (keine Probleme mit Vanishing Gradient²)
- Ressourcenaufwendig: hoher Speicherbedarf und (sehr) aufwendige Trainingszeit.
- Und neigt zum Overfitting (das man durch Tunen Hyperparameter und Struktur LSTM aber reduzieren kann).

Wiederholungsfrage: was ist Overfitting?

- Transformer (2017)³:

- Entwickelt von Google, basierend auf **Attention⁴-Verfahren** aus 2015 (basierend auf Schmidhuber 1992).
- Kein RNN, sondern Kombination verschiedener sequentieller Verfahren ohne Rückkopplung (inkl. Transfer Learning).
- Schnelleres Training als LSTM aufgrund Aufteilen Lernen auf mehrere Rechner (Parallelisieren).
- Hat LSTM in vielen Gebieten abgelöst, bspw. für maschinelles Übersetzen, aufgrund höherer Flexibilität.
- Einsatz analog Transfer Learning für State-of-the-Art Ansätze wie BERT (Bidirectional Encoder Representations from Transformers)⁵ und GPT-3 (Generative Pre-trained Transformer 3)⁶.

Quellen: 1 Hochreiter, Schmidhuber: "Long Short-Term Memory ", 1997. [Link](#)

2 Vanishing Gradient: beim Trainieren von Netzen mit vielen Schichten und Nutzen Exponentialfunktion als Aktivierungsfunktion kann bei Backpropagation Fall entstehen, daß Gradienten sehr, sehr klein werden und damit die Gewichte und Bias des Modells, v.a. der ersten Schichten, nicht mehr geändert/ trainiert werden. Vgl. Hochreiter: „Untersuchungen zu dynamischen neuronalen Netzen“, 1991.

3 Vaswani et al.: "Attention is all you need ", 2017. [Link](#)

4 Bahdanau et al.: "Neural Machine Translation by Jointly Learning to Align and Translate", 2015. [Link](#). Basierend auf Schmidhuber, "Learning to control fast-weight memories.", 1992.

5 Devlin et al.: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2019. [Link](#)

6 Brown et al.: "Language Models are Few-Shot Learners", 2020

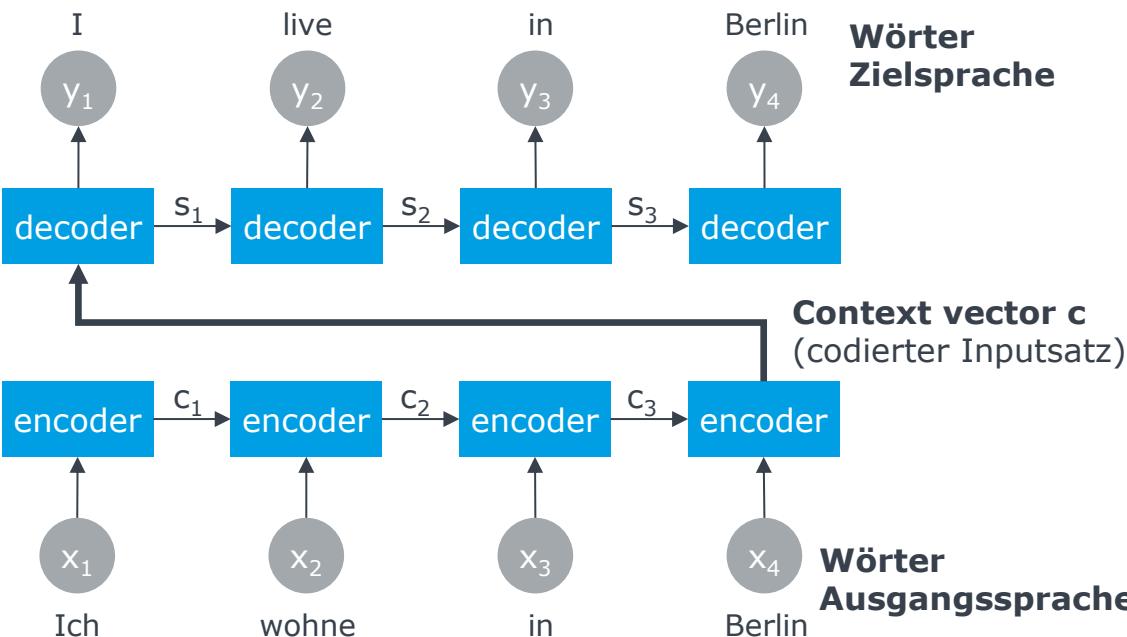
DETAILLIERUNG BEISPIELHAFTE NLP-TECHNIKEN:

- **Sentence breaking:** Aufspalten Text in einzelne Sätze, bspw. anhand Punkt oder Komma.
 - **Lemmatization:** Vereinheitlichen ähnlicher Wörter/ Reduktion Varianz mit Wörterbuch. {Eats, eating, ate, eaten} → eat {ride, taking a ride} → ride
 - **Stemming:** Vereinheitlichen eines Wortes/ Reduktion Varianten durch Abschneiden Endungen. {played, plays, playing, ...} → play
 - **Stop Word Removal:** Entfernen Füllwörter (Pronomen, Artikel, Präpositionen, Fragewörter,....) [i, my, her, hers, has, had, hadnt, no, nor, not, only, doesnt, ...]
 - **Text/Word Embedding:** Umwandeln einzelner Buchstaben/ ganzer Wörter in eindeutige Zahlen. (Rechner können ja nur mit Zahlen arbeiten....)
- | | |
|-------|-----------|
| 1 | → <start> |
| Bus | → 5 |
| kommt | → 144 |
| heute | → 114 |
| 2 | → <end> |

NLP ermöglicht das effiziente Verarbeiten von Texten für Machine Learning

(VEREINFACHTES) ATTENTION – MOTIVATION.

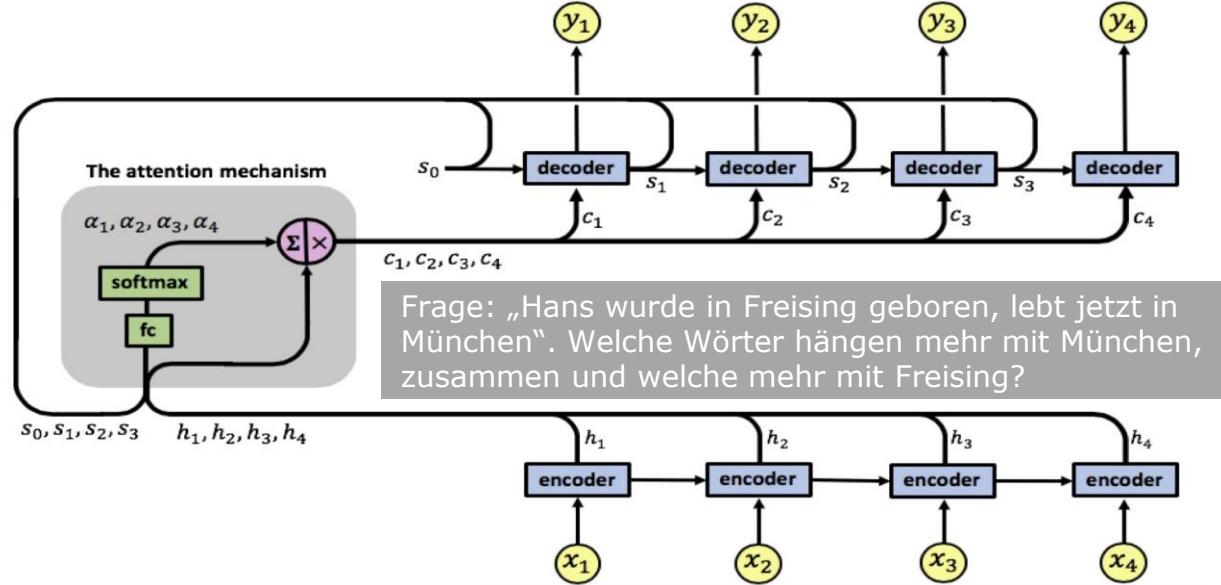
Übersetzen Ausgangs- in Zielsprache bei RNN



Herausforderungen/ Probleme bisheriger Verfahren:

- letztes Wort Ausgangssprache ist Input erstes Wort für Zielsprache; dabei sind oft erste Wörter ähnlich.
- „Normales“ RNN: Länge Decoder (Satz einer Zielsprache) festgelegt, was zu Problemen beim Lernen von langen Sätzen führt (umgangssprachlich: Modell „vergisst“ Kontext).
- RNN/ LSTM: sehr hoher Rechenaufwand & nicht parallelisierbar.

Optimierung durch Attention Mechanismus

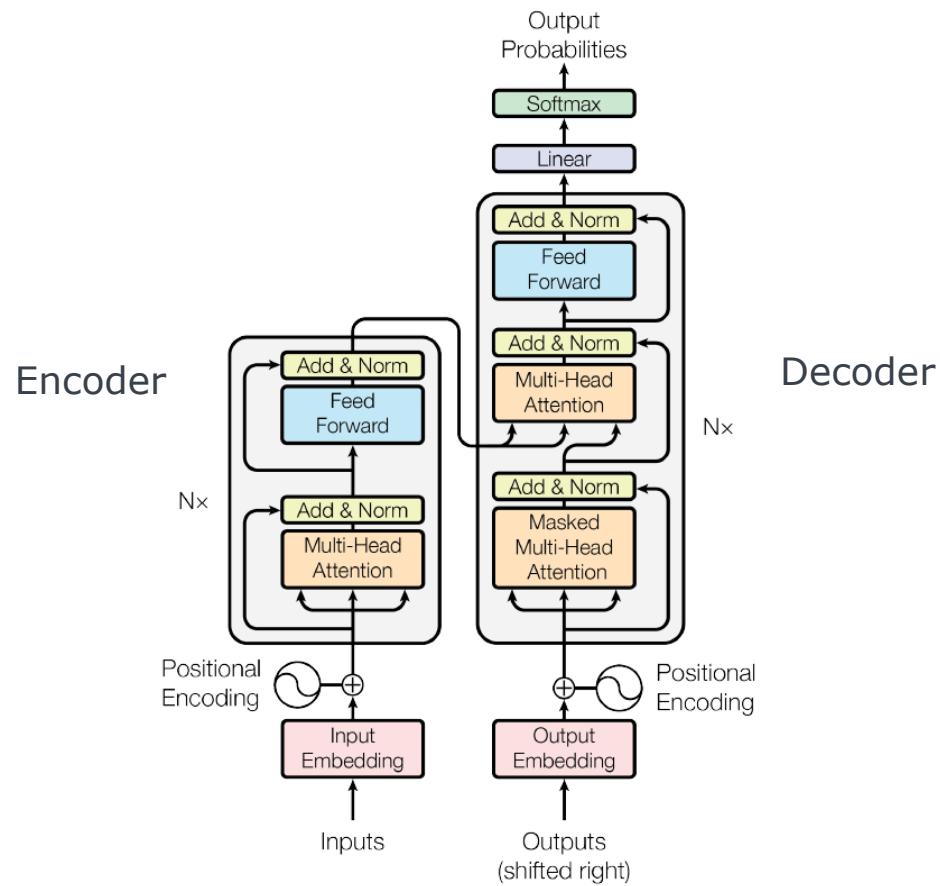


Attention löst die erwähnten Herausforderungen:

- Attention ermöglicht dem Decoder, bestimmte Teile des Inputs stärker zu gewichten bei der Vorhersage des Outputs (hier: über $c_1 - c_4$). Lernen dieser Attention-Gewichte ($a_1 - a_4$) per neuronalem Netz.
- Das Problem der Satzlänge wird durch individuelle Gewichte für jeden Output gelöst. Von dieser unterschiedlichen Fokussierung stammt der Name Attention.

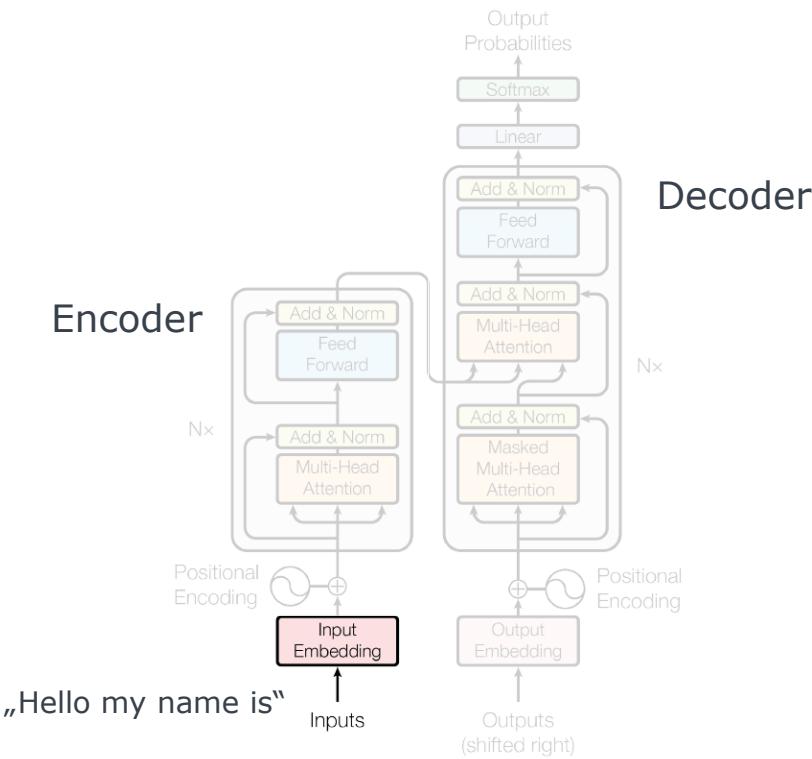
DETAILLIERUNG TRANSFORMER

ÜBERSICHT TRANSFORMER.



Aufgabe Encoder: speichere Eingabe in sequentieller Form inkl. Attention (welche Input-Wörter hängen mit welchen zusammen?)
Aufgabe Decoder: nutze diese Form für Fokus auf passende Input-Wörter und erstelle Wahrscheinlichkeitsverteilung über gesamtes Vokabular für möglichst genaue Vorhersage des nächsten Ausgabeworts/ nächsten Ausgabewörter.

TRANSFORMER IM DETAIL. ENCODER – INPUT EMBEDDING.

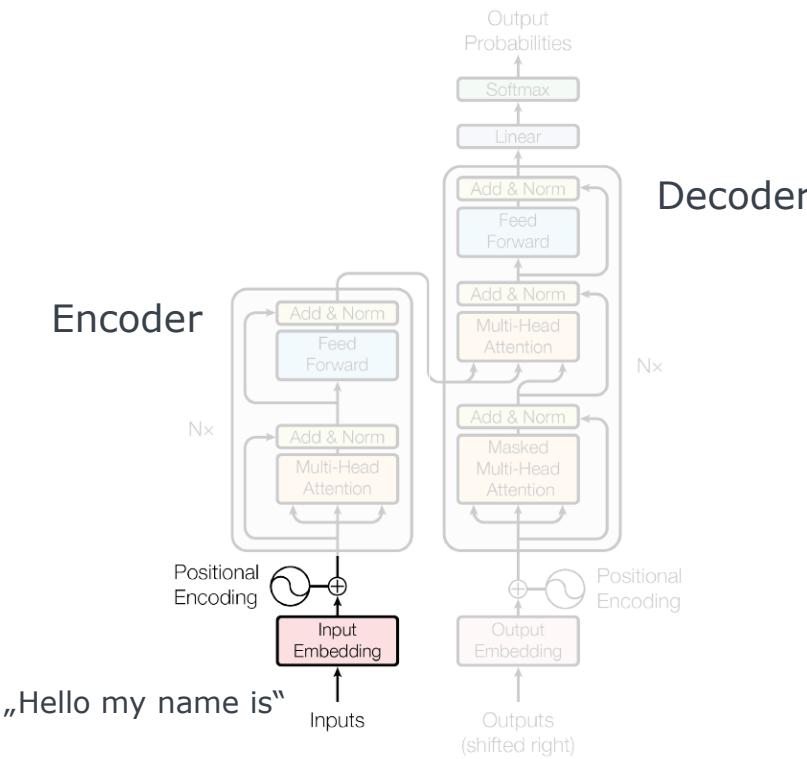


Input Embedding:

- Eingabetext wird aufgeteilt in einzelne Wörter („Tokens“). Kann auf mehreren Rechnern parallel erfolgen.
„Hello my name is“ → [Hello] [my] [name] [is]
- Hinzufügen Tokens zu einem Vokabular mit eindeutiger Nummer.
[Hello] [my] [name] [is] -> [3 721 68 1337]
- Jeder Token-Eintrag hat eine Verbindung zu dem (bisher) gelernten Modell und dessen n-dimensionalen Vektor. Zusätzlich werden ähnliche Wörter gruppiert.
[3 721 68 1337] -> [[0.123, -5.234, ...], [...], [...], [...]]

Computer verstehen keine Wörter, sondern Zahlen. Deshalb bauen wir eine Art Wörterbuch mit eindeutiger Nummer je Wort. Um die Qualität des Wörterbuchs zu verbessern, gruppieren wir von der Bedeutung ähnliche Wörter.

TRANSFORMER IM DETAIL. ENCODER – POSITIONAL ENCODING

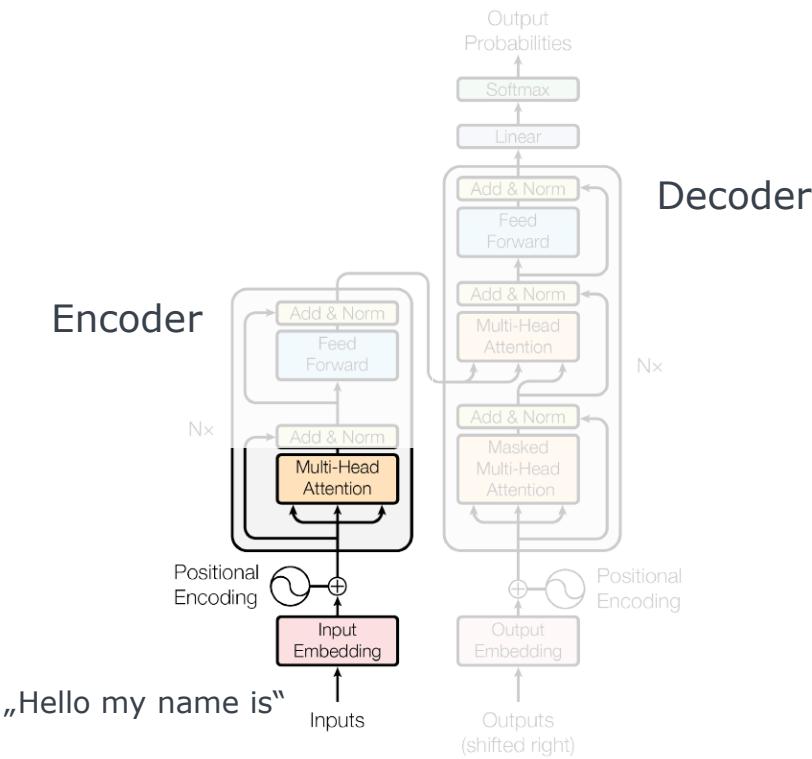


Positional Encoding:

- Speichere Position, die die einzelnen Wörter im Input-Satz hatten.
- Addiere die Position zu dem Embedding aus dem vorigen Schritt.

Durch Positional Encoding speichern wir die Reihenfolge der Wörter und können so bspw. „A liebt B“ von „B liebt A“ unterscheiden.

TRANSFORMER IM DETAIL. ENCODER – MULTI-HEAD ATTENTION.



Self-headed attention:

- Modell soll die Verbindungen zwischen den einzelnen Input-Wörtern untereinander lernen (bspw. „wie“ mit „geht“ und „dir“).

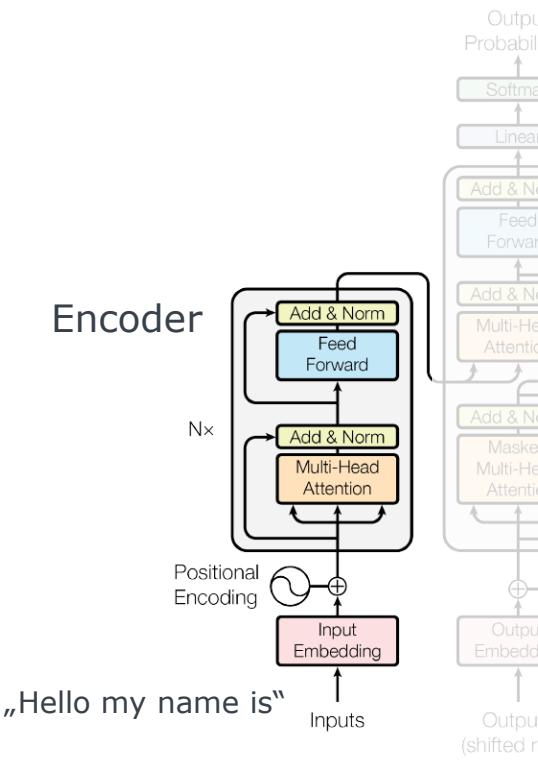
	Hello	my	name	is
Hello	0,75	0,15	0,05	0,05
my	0,1	0,6	0,2	0,1
name	0,05	0,2	0,65	0,1
is	0,2	0,1	0,1	0,6

Multi-headed attention:

- vermeidet, daß self-headed attention den eigenen Input stärker gewichtet als die restlichen Wörter.
- Gewichten der Ergebnisse einzelner self-headed Attentions und Kombination zu finalem Vektor je Token. Dieser Vektor sagt, wie stark das Token auf die anderen Wörter schauen sollte.

Multi-headed Attention erlaubt das Finden und Lernen von Korrelationen zwischen den einzelnen Bestandteilen eines Satzes.

TRANSFORMER IM DETAIL. ENCODER – RESIDUAL, NORMALISIERUNG UND FEED FORWARD



Residual:

- Der vorherige Ausgabevektor wird zum Positions-Encoding addiert. Dies ermöglicht ein starkes Verbessern der Trainierbarkeit.

Normalisierung:

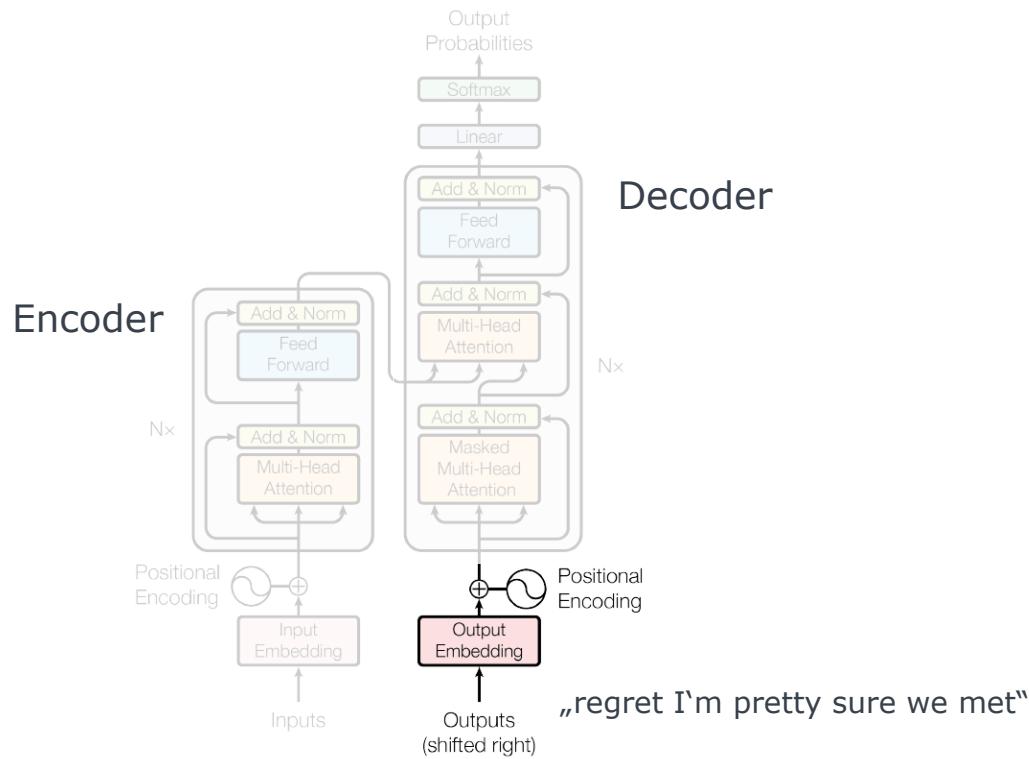
- Stabilisieren Netzwerk und deutliche Reduktion Trainingszeiten.

Feed forward:

- Einsatz eines einfachen, nicht-rekurrenten neuronalen Netzwerks für Verarbeitung Ergebnisse (analog bisherige Vorlesungen...).
- Speichern der Ergebnisse in einem Format für die spätere Berechnung Wahrscheinlichkeit (analog Bilderkennung!)

Einsatz dieser Techniken ermöglichte Verbesserung der Ergebnisse sowie Reduzierung Ressourceneinsatz.

TRANSFORMER IM DETAIL. DECODER – INPUT EMBEDDING UND POSITIONS ENCODING



Input Embedding:

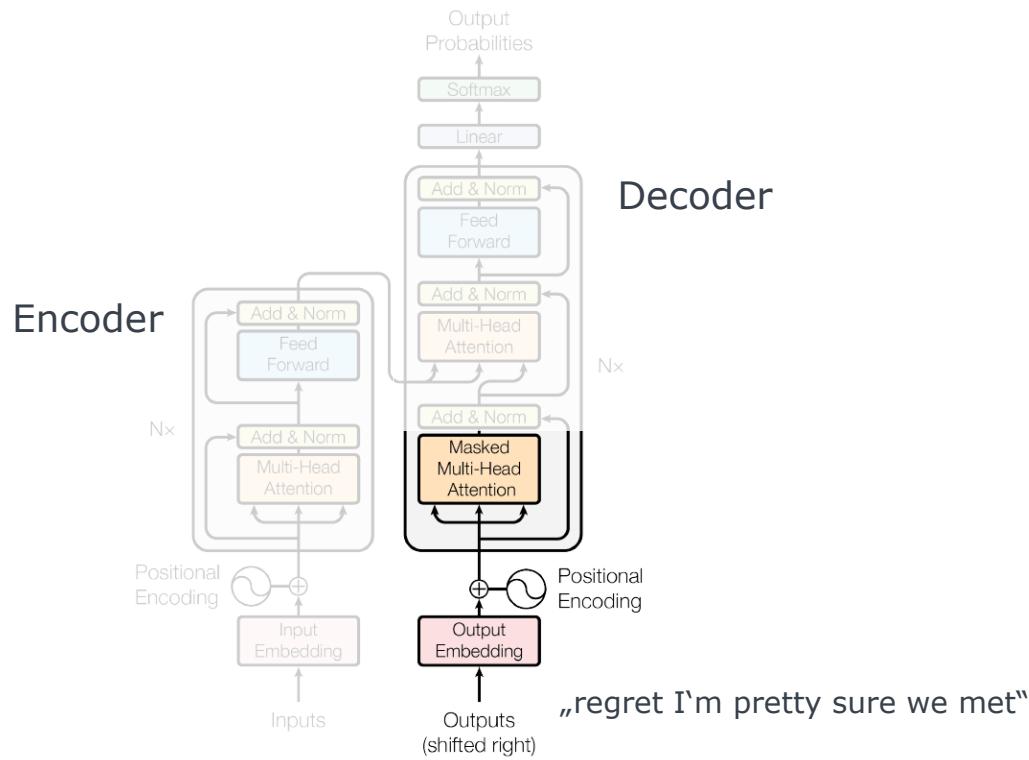
- Gleiches Vorgehen für Output-Satz wie bei Input

Positional Encoding:

- Analog zu Encoding.

Gleiches Vorgehen wie beim Encoding, nur für Decoding Satz. Dieser ist beim Trainieren bekannt, beim späteren Einsatz jedoch nicht.

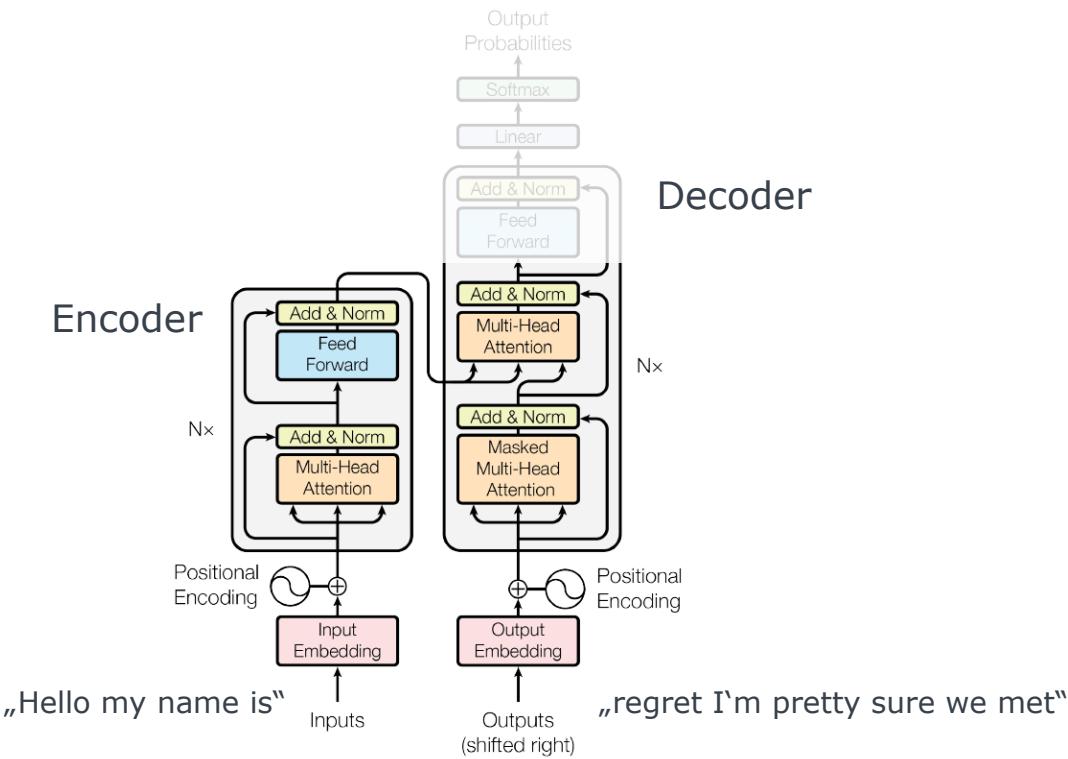
TRANSFORMER IM DETAIL. DECODER – MASKED MULTI-HEAD ATTENTION.



Masked Multi-head attention

- Ähnlicher Schritt wie Multi-Head Attention beim Encoding.
- Aber: beim Trainieren haben wir jeweils einen Satz für Input als auch für Output.
- Damit das Modell lernt und nicht einfach das nächste Wort "nachschaut", werden die restlichen Wörter des Vergleichssatzes ausmaskiert (auf 0 gesetzt), d.h. das Modell kann diese nicht nachsehen.
- Dadurch kann auch parallel trainiert werden und das Training somit beschleunigt werden.

TRANSFORMER IM DETAIL. DECODER –MULTI-HEAD ATTENTION.

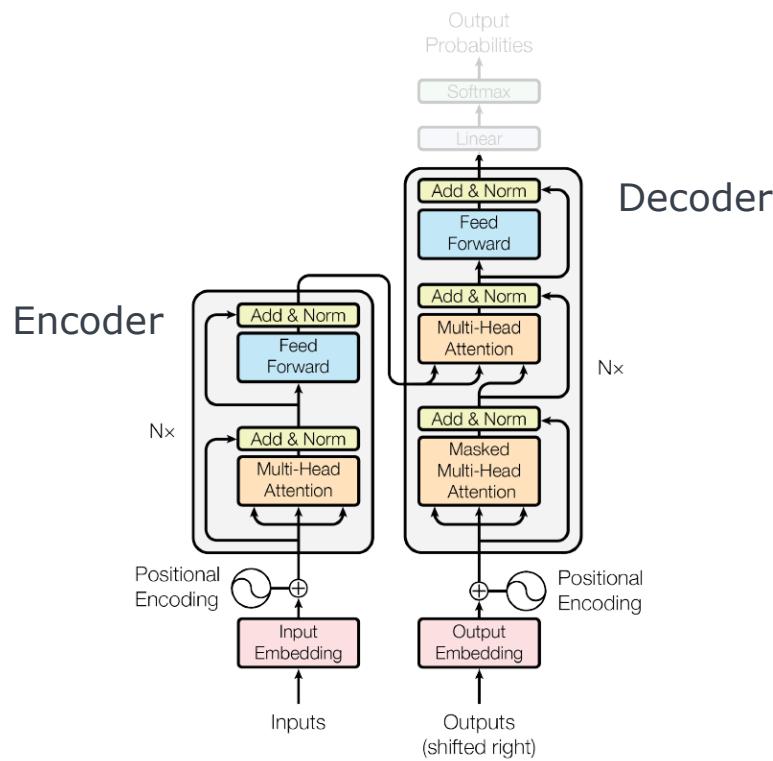


Multi-head attention (oder Encoder-Decoder Attention)

- In diesem Block kommen die Tokens vom Encoder und die maskierten Eingaben vom Decoder (inkl. Residual) zum ersten Mal zusammen.
- Dadurch kann das Modell für jeden Token vom Decoder lernen, welcher Input vom Encoder relevant ist.

Hier entsteht das Mapping von Input auf Output also bspw. ein Attention Vektor für jedes Wort

TRANSFORMER IM DETAIL. DECODER –FEED FORWARD.



Normalisierung:

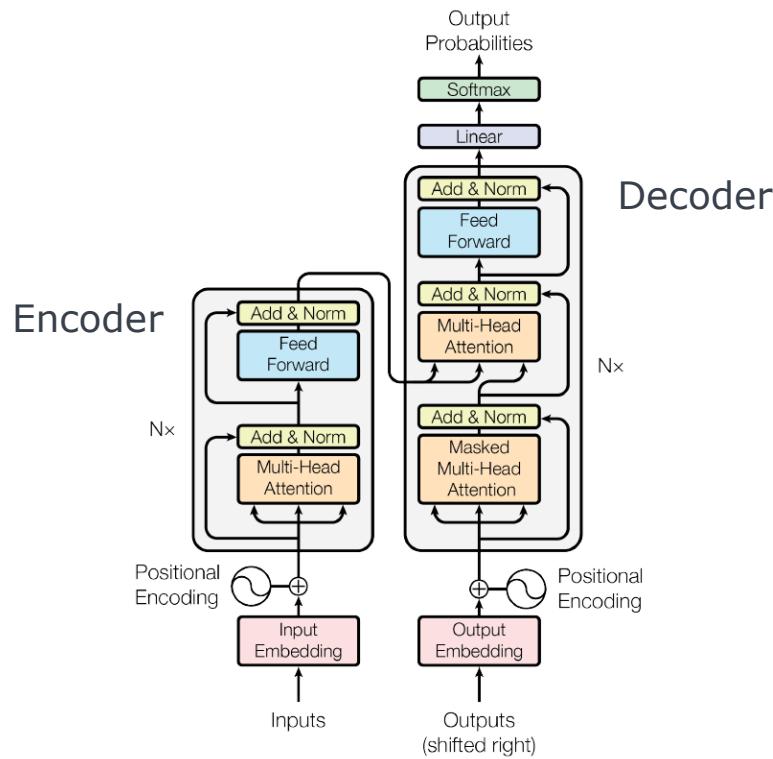
- Stabilisieren Netzwerk und deutliche Reduktion Trainingszeiten.

Feed forward:

- Einsatz eines einfachen, nicht-rekurrenten, neuronalen Netzwerks für Verarbeitung Ergebnisse.
- Speichern der Ergebnisse in einem Format für spätere Berechnung Wahrscheinlichkeit (analog Bilderkennung!)

Einsatz dieser Techniken ermöglichte Verbesserung der Ergebnisse sowie Reduzierung Ressourceneinsatz.

TRANSFORMER IM DETAIL. DECODER – OUTPUT.



Lineares Neuronales Netz

- so groß wie das gesamte Vokabular für Output, d.h. eine Zelle je Wort (gleiches Vorgehen wie Flatten-Vektor bei CNN!)

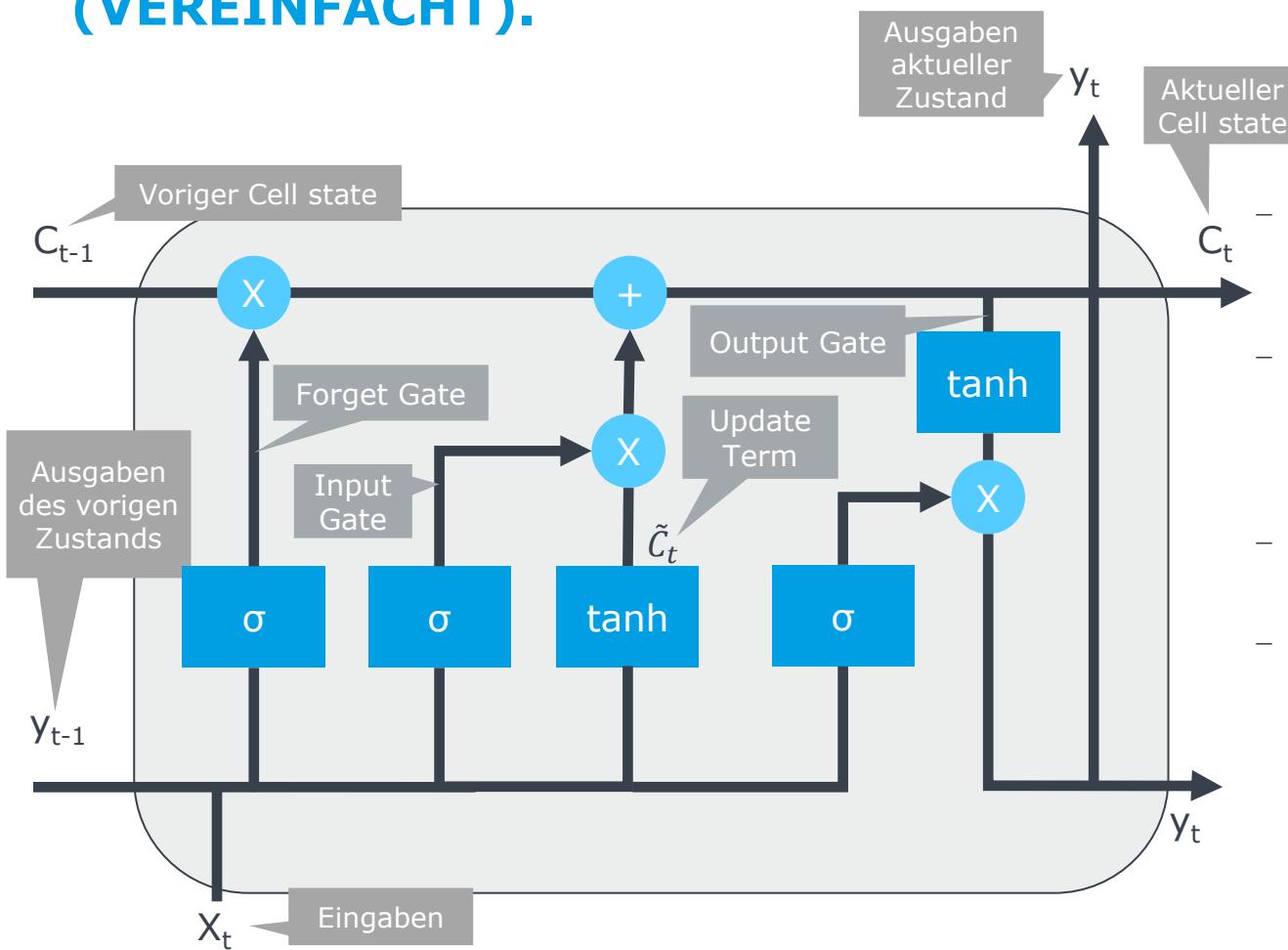
Softmax:

- Erstellt Wahrscheinlichkeitsverteilung für alle Zellen (d.h. über alle Wörter des gesamten Wörterbuchs).
- Die Zelle mit höchstem Wert für Wahrscheinlichkeit wird als vorhergesagtes nächstes Token genommen.

Zusammengefaßt ist die Aufgabe des Decoders das Sammeln aller notwendigen Informationen um mittels einer Wahrscheinlichkeitsverteilung aus dem gesamten Vokabular das nächste Ausgabewort möglichst genau vorherzusagen.

DETAILLIERUNG LSTM

LONG SHORT-TERM MEMORY: STRUKTUR UND ABLAUF (VEREINFACHT).



Wir schauen uns eine von mehreren LSTM-Zelle an, die miteinander verknüpft sind, d.h. die abgebildete Zelle hat „Nachbarzellen“.

- **Forget Gate**: entscheidet, welche Infos des Cell State C_{t-1} zu vergessen oder weiter zu erinnern sind. Geschieht per Parameter mit Wert 0 für Vergessen, 1 für Erinnern oder Wert dazwischen, falls Gate unsicher ist.
- **Input Gate**: hat zwei Schritte.
Schritt 1: Entscheidet, welche Infos für Cell State C_t zu aktualisieren sind. Erfolgt per Parameter mit Werten von 0 (irrelevant) bis 1 (relevant).
Schritt 2: Bestimmt Update-Terme, d.h. die neuen Werte für den Cell State.
- **Cell State**: Hier wird die Zelle C_t aktualisiert, basierend auf Inputs des Forget Gate und Input Gates.
- **Output Gate**: entscheidet was aus der Zelle ausgegeben wird.

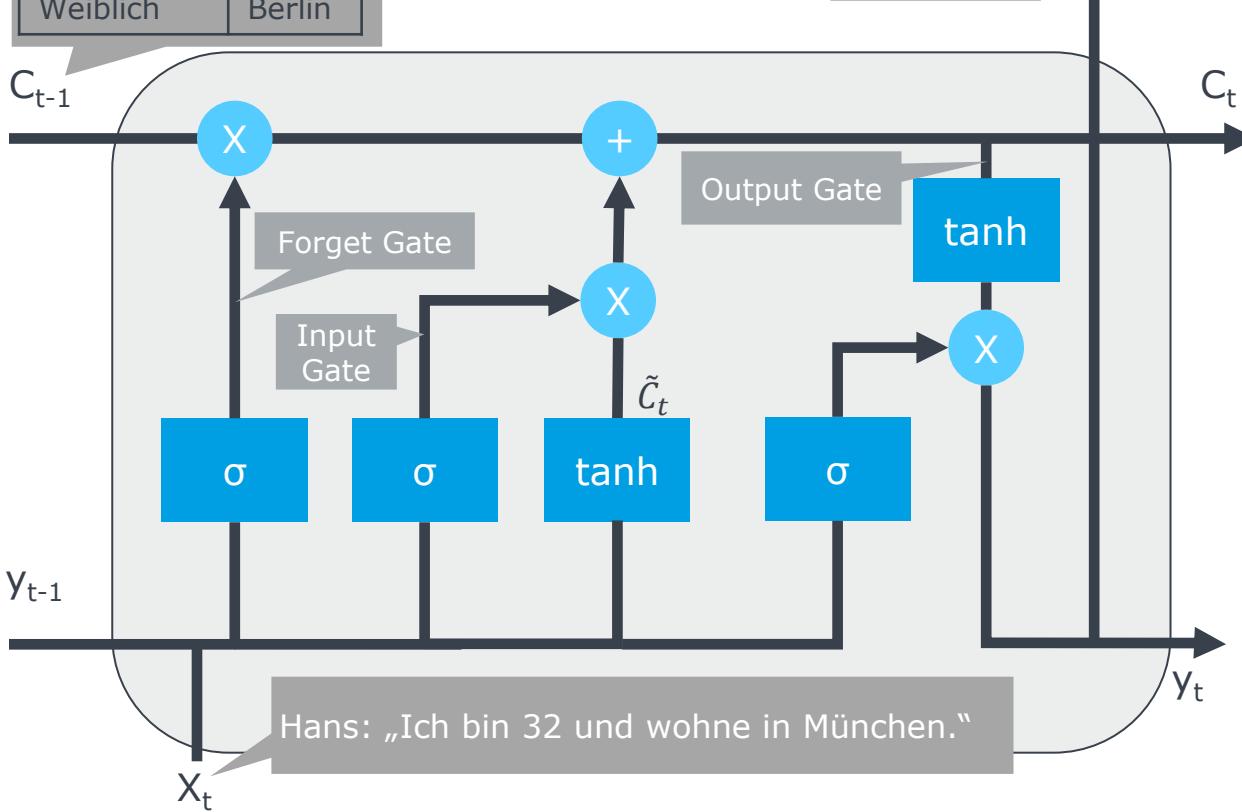
LSTM vermeidet Vanishing Gradient durch:

- Steuerung Verhalten Gradienten durch Werte Gates: Wert 0 verhindert bspw. Änderung Gradient.
- Werte des Gates für Vermeiden Explosion werden gelernt.
- Formel für Ausgabe C_t enthält Addition. Dadurch hat die den Gradienten erzeugende Ableitung ein „besseres Verhalten“ als bei bspw. Multiplikation

LONG SHORT-TERM MEMORY: FALLBEISPIEL (VEREINFACHT).

Voriger Cell state	
Geschlecht	Dort
Weiblich	Berlin

Ausgabe
Dort
München



Ziel: „Verstehen“ einer Konversation durch Algorithmus

Anna: „Ich bin 30 Jahre alt und wohne in Berlin“

Hans: „Ich bin 32 und wohne in München.“

Anna: „Wie viele Menschen leben dort?“

Frage: worauf bezieht sich dort?

Geschlecht	Dort
Weiblich	Berlin

Start mit Hans' Satz: Cell state C_{t-1} :

- **Forget Gate:** vergiss Werte für Geschlecht und Wohnort (Forget gate = 0), da mit Hans andere Person mit anderem Geschlecht und Wohnort spricht.
- **Input Gate:**
 - Schritt 1: bestimme zu aktualisierende Werte
 - Schritt 2: bestimme Änderungswerte \tilde{C}_t
- **Cell State:** schreibe die Werte in die Zelle C_t
- **Output Gate:** schreibe in Ausgabe y_t Wert für dort, da dieser Begriff für Annas nächsten Satz relevant ist. LSTM lernt also, daß Annas Frage sich auf München bezieht!

Geschlecht	Dort
Männlich	München

Geschlecht	Dort
Männlich	München

Dort
München