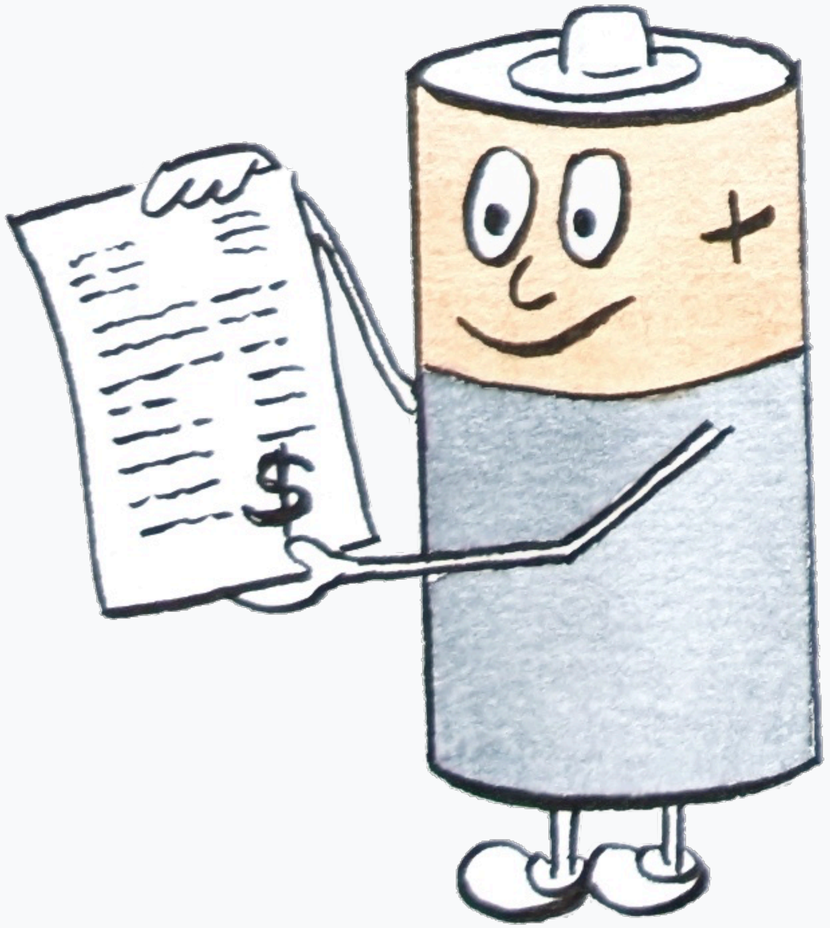
Lass uns gemeinsam ein GPT-Modell in Kotlin bauen

KI durch praktische Anwendung entmystifizieren

Michal Harakal, Deutsche Telekom AG, 2024

Wir müssen zuerst über ChatGPT&Co reden

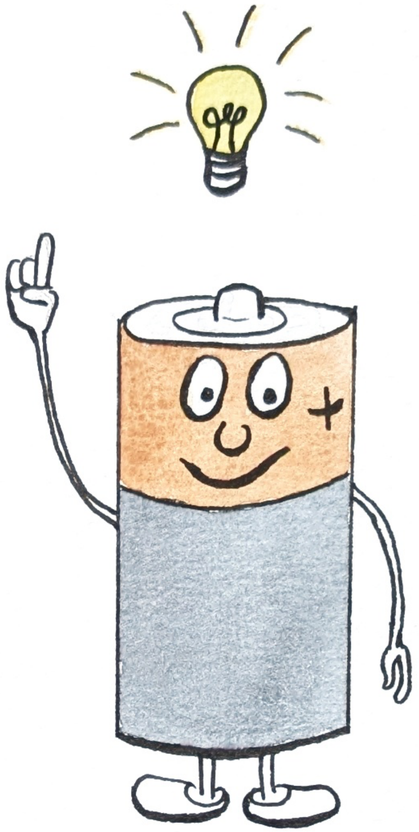
Wir müssen zuerst über ChatGPT&Co reden

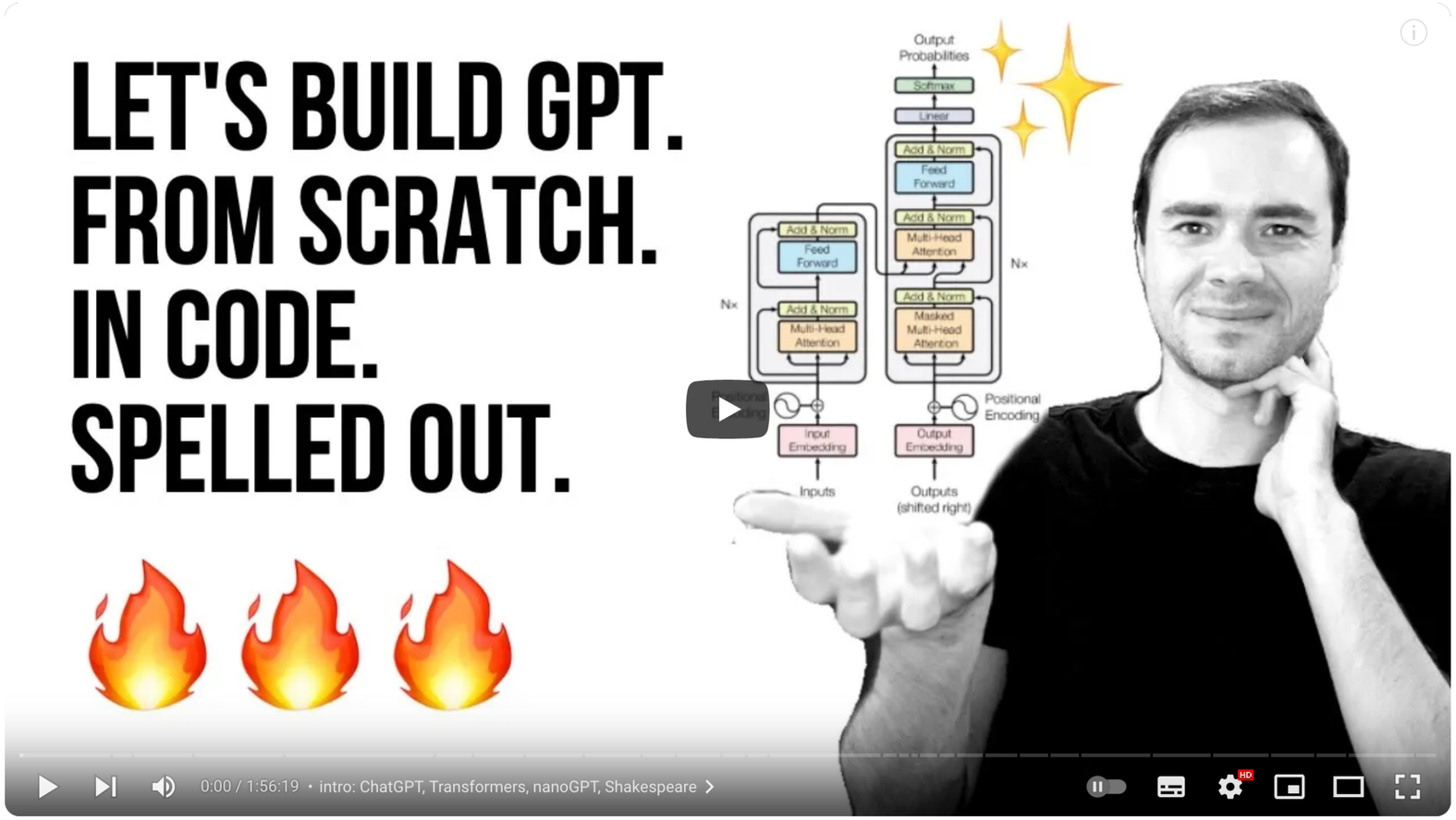


(C) Adriana Harakalova, 2024

What I cannot create,
I do not understand.
Know how to solve every problem that has been solved.

Richard Feynman, 1988





Let's build GPT: from scratch, in code, spelled out.

Andrej Karpathy
540.000 Abonnenten
Abonnieren

107.486 Teilen Speichern

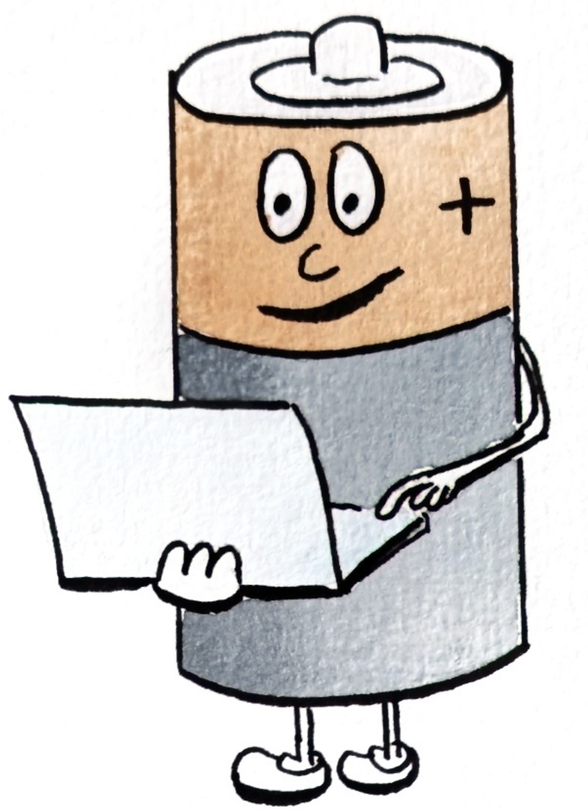
https://www.youtube.com/watch?v=kCc8FmEb1nY

Let's build GPT: from scratch, in code, spelled out.

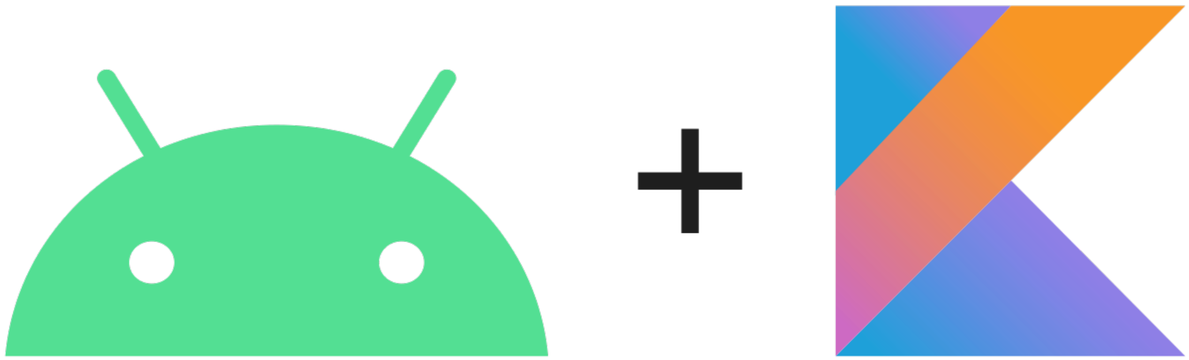
Andrej Karpathy

- GTP-2 Model mit cca 10 Millionen Parameters
- 250 Zeilen Python Quellcode
- Benutzt PyTorch

Kotlin



(C) Adriana Harakalova, 2024



Kotlin

Java und JVM

- Vollständige Interoperabilität mit Java
- Kompakterer und ausdrucksstärkerer Code
- Eingebaute Null-Sicherheit
- Unterstützung von funktionaler Programmierung
- Kompilierung auf JVM und andere Plattformen (JavaScript, Kotlin/Native)
- Verbesserte Typinferenz
- Hervorragende Unterstützung durch moderne IDEs (IntelliJ IDEA, Android Studio)
- Koroutinen für einfache asynchrone Programmierung

Kotlin Multiplatform

Quellcode Sharing

- Plattformübergreifende Entwicklung
- Leichtgewichtige und performante Parallelisierung durch Koroutinen
- Unterstützung für Native Performance
- Plattformübergreifende UI mit Compose Multiplatform

AI/ML mit Java und JVM

Aussicht

- Projekt Babylon
  - https://openjdk.org/projects/babylon/articles/auto-diff
- Projekt Panama
  - https://openjdk.org/projects/panama/
- **HAT**- Heterogeneous Accelerator Toolkit (memory structuces off the heap) using project panama for memory and functional API

Mathematik und ML mit Kotlin

Aktueller Status

- Kotlin für data analysis https://kotlinlang.org/docs/data-analysis-overview.html
- KotlinDL
- **Autodifferentiationsbibliotheken**
  - Kotlin∇ - https://github.com/breandan/kotlingrad
  - https://github.com/facebookresearch/diffkt
- Mathematik uns Statistik Bibliotheken
    - https://www.kalasim.org/
    - KMath - https://github.com/SciProgCentre/kmath

AI und Agenten

Arc - The Agent DSL

https://lmos-ai.github.io/arc/

Kombiniert Einfachheit vom Low-Code-mit der Leistungsfähigkeit eines Enterprise-Frameworks.

Von der Prototypenerstellung bis zur Produktion mit derselben Technologie.

Arc ist eine Kotlin-DSL, die entwickelt wurde, um LLM-gesteuerte Agents schnell und präzise zu erstellen.

- **Framework-Agnostisch**
  Agents können auf jeder Plattform laufen, wie Spring Boot, Quarkus, Ktor… sogar auf mobilen Geräten.

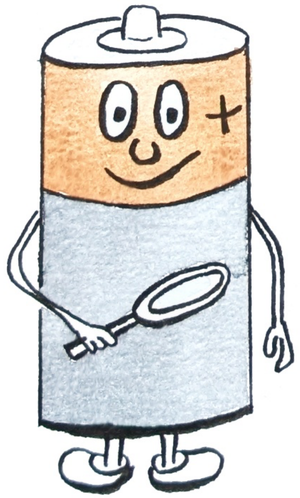
- **Erweiterbar**
  Die DSL ist leicht erweiterbar. Neue Funktionen und Features können zur Sprache hinzugefügt werden.

```
agent { this: AgentDefinition
    name = "contract-service-agent"
    model = { "GPT-4o" }
    tools = listOf(
        "get_credits_information",
        "list_contracts",
        "contract_details",
        "contract_lock_unblock_numbers",
    )
    description = """Agent that helps with account settings
        (e.g. contact details) and contract information."""
    filterOutput { this: OutputFilterContext
         +HallucinationDetectedFilter::class
    }
    systemPrompt = { this: DSLContext
        val customerProfile = get<CustomerProfile>()
        """
### Role and Responsibilities ###
You are a customer care agent for Deutsche Telekom and your task
All incoming inquiries are related to
and landline accounts which can includ

### Access and Privacy ###
Privacy and Data Requirement: You do n
This helps maintain a level of privacy
```

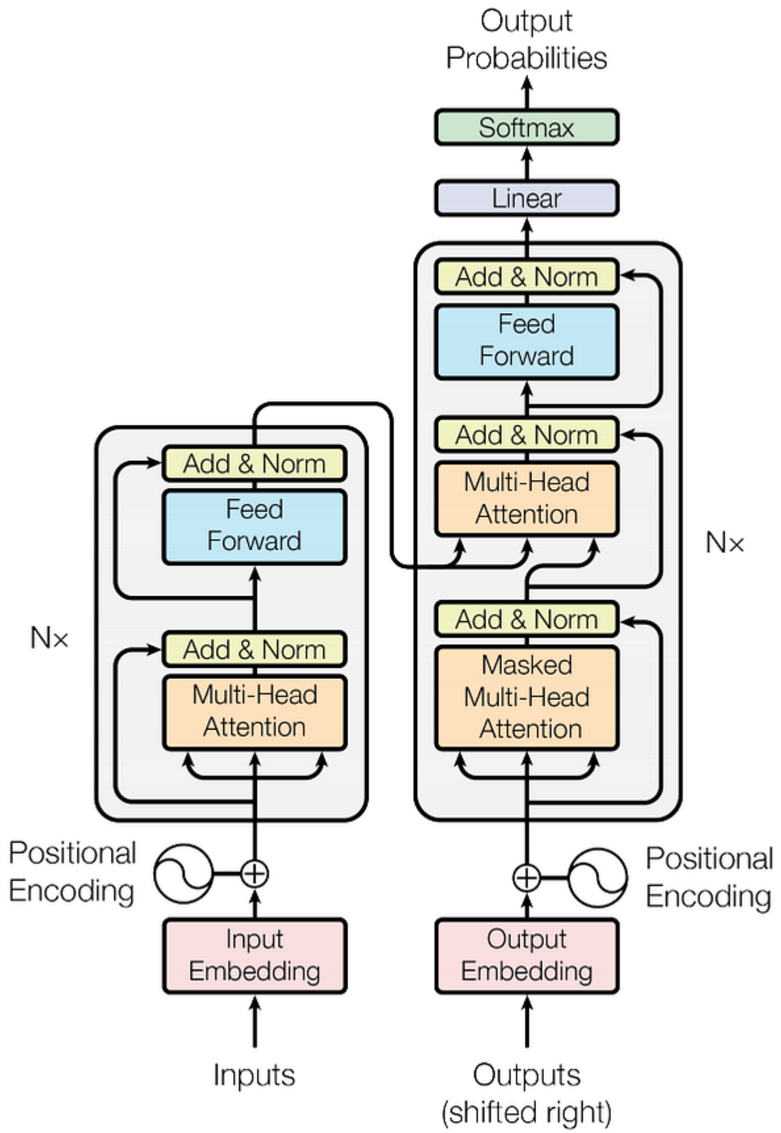
Generative Pre-trained Transformers

Transformers



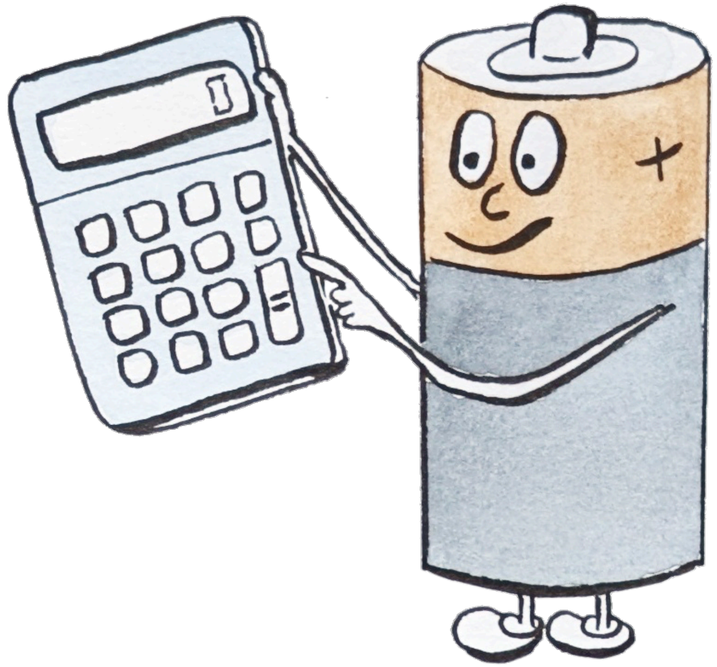
Attention Is All You Need

- Veröffentlicht 2017
- Google Brain, Google Research und andere
- Transformer model Architektur
- Übersetzungen



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

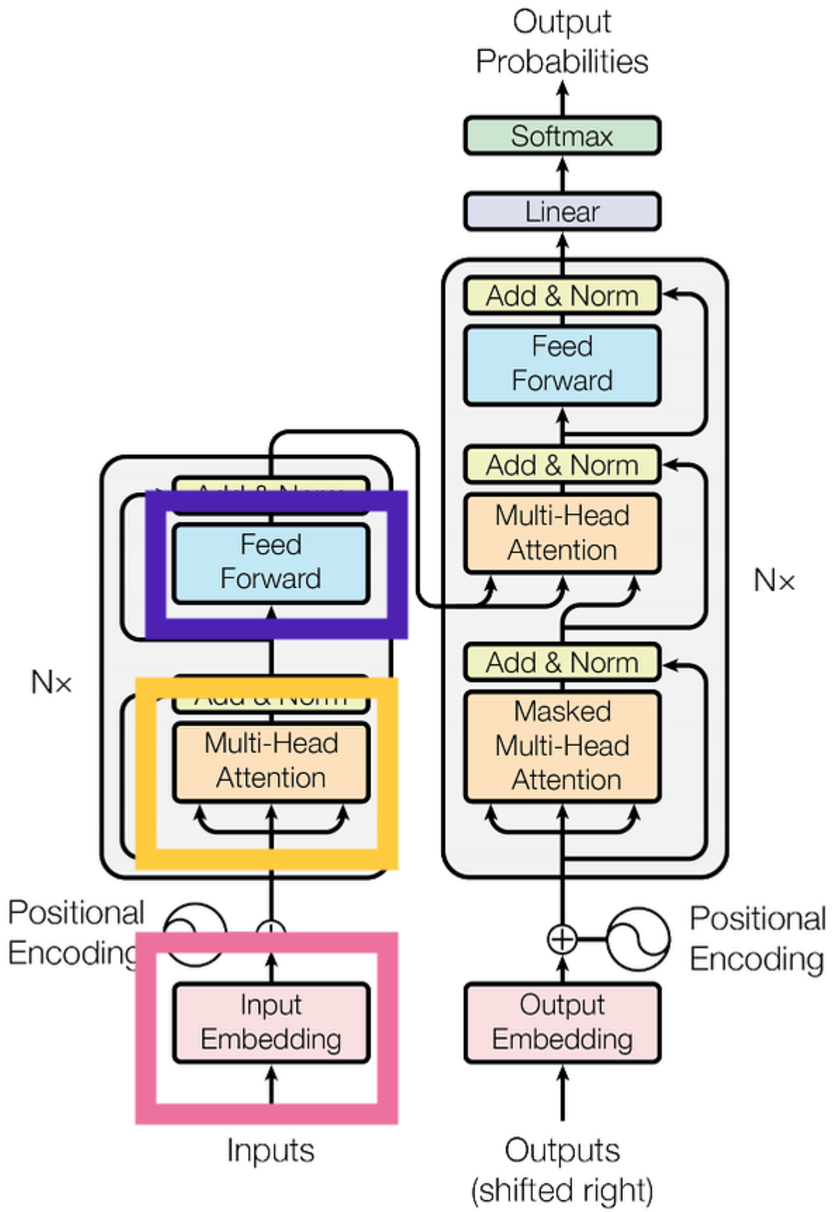
Pre-trained



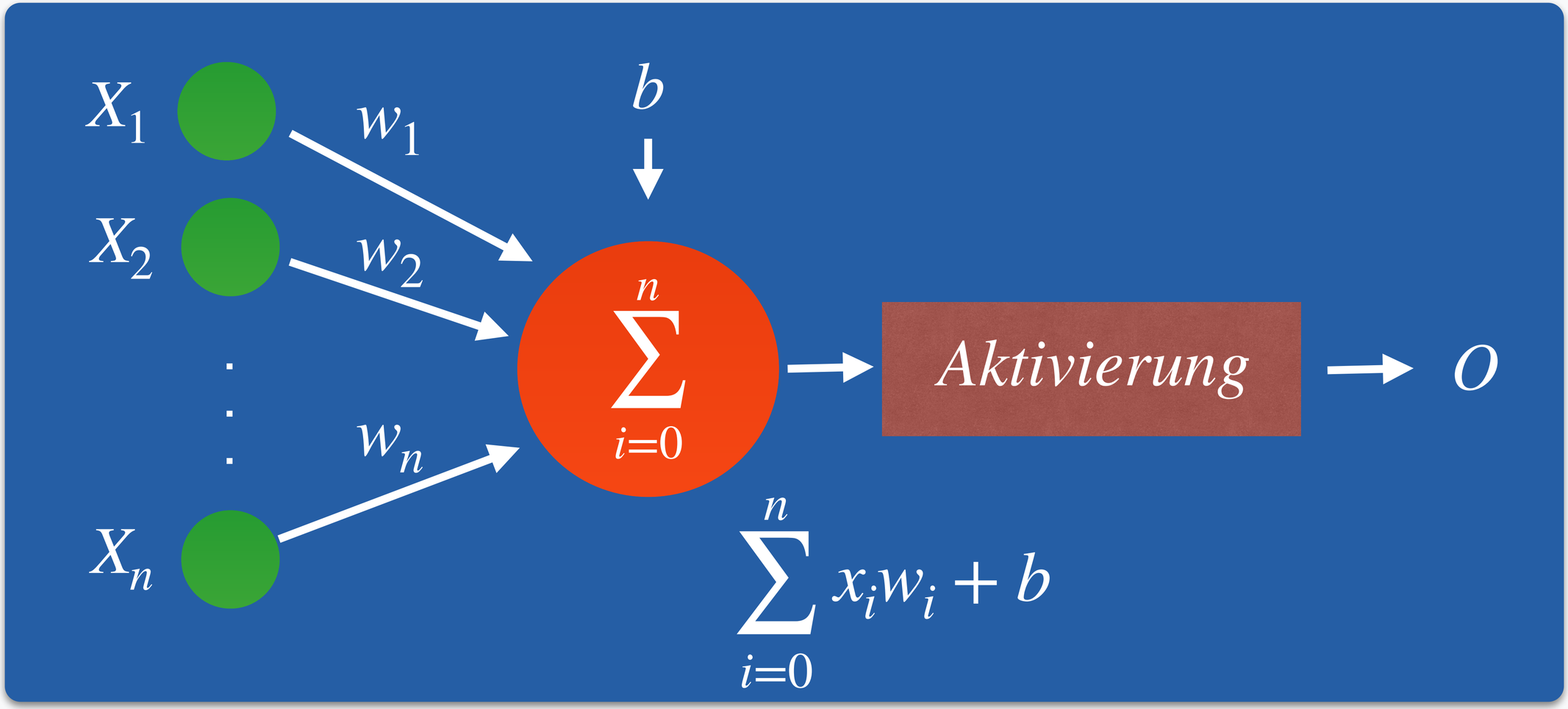
(C) Adriana Harakalova, 2024

Transformers

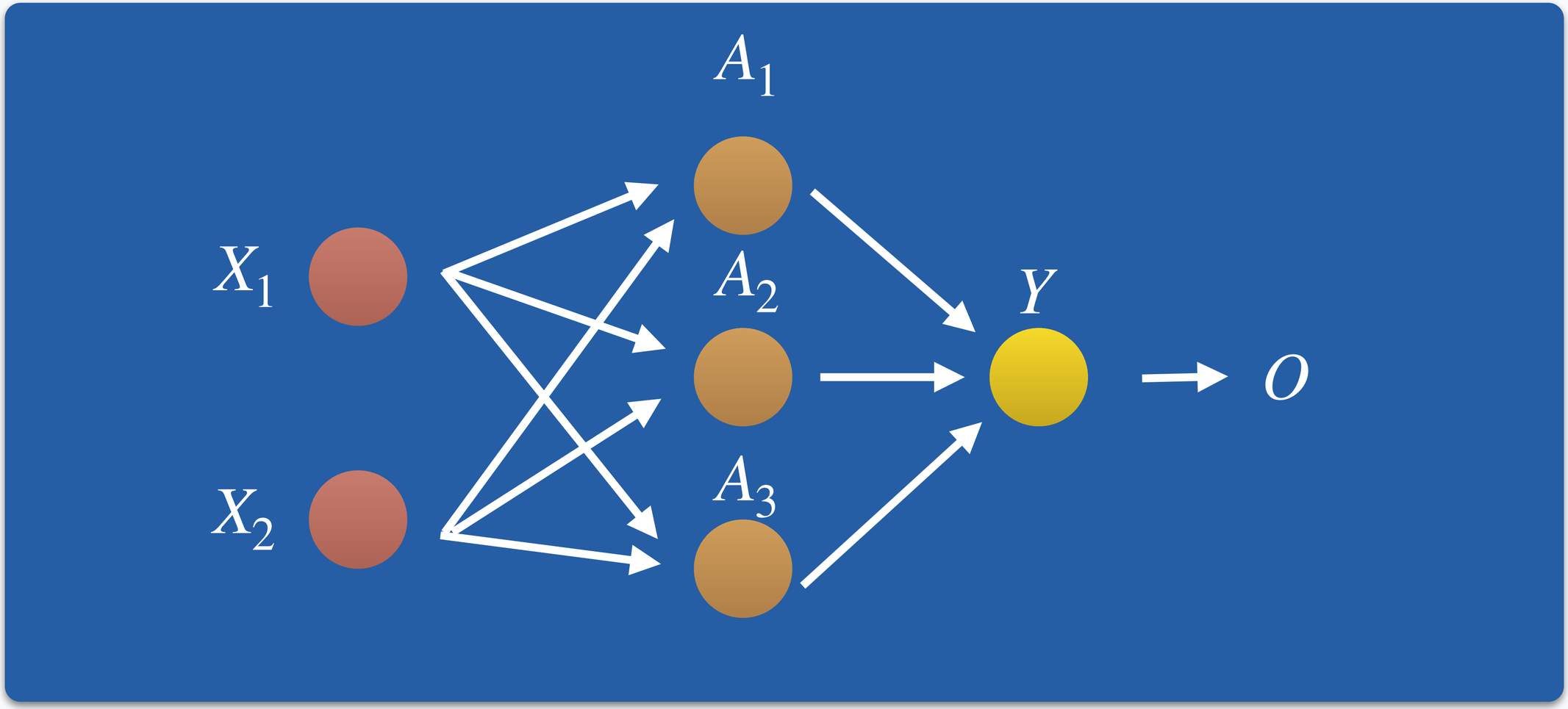
- Embedding Layer
- Multi-head attention
- Feedforward Neural Network
- Layer Normalization & Softmax



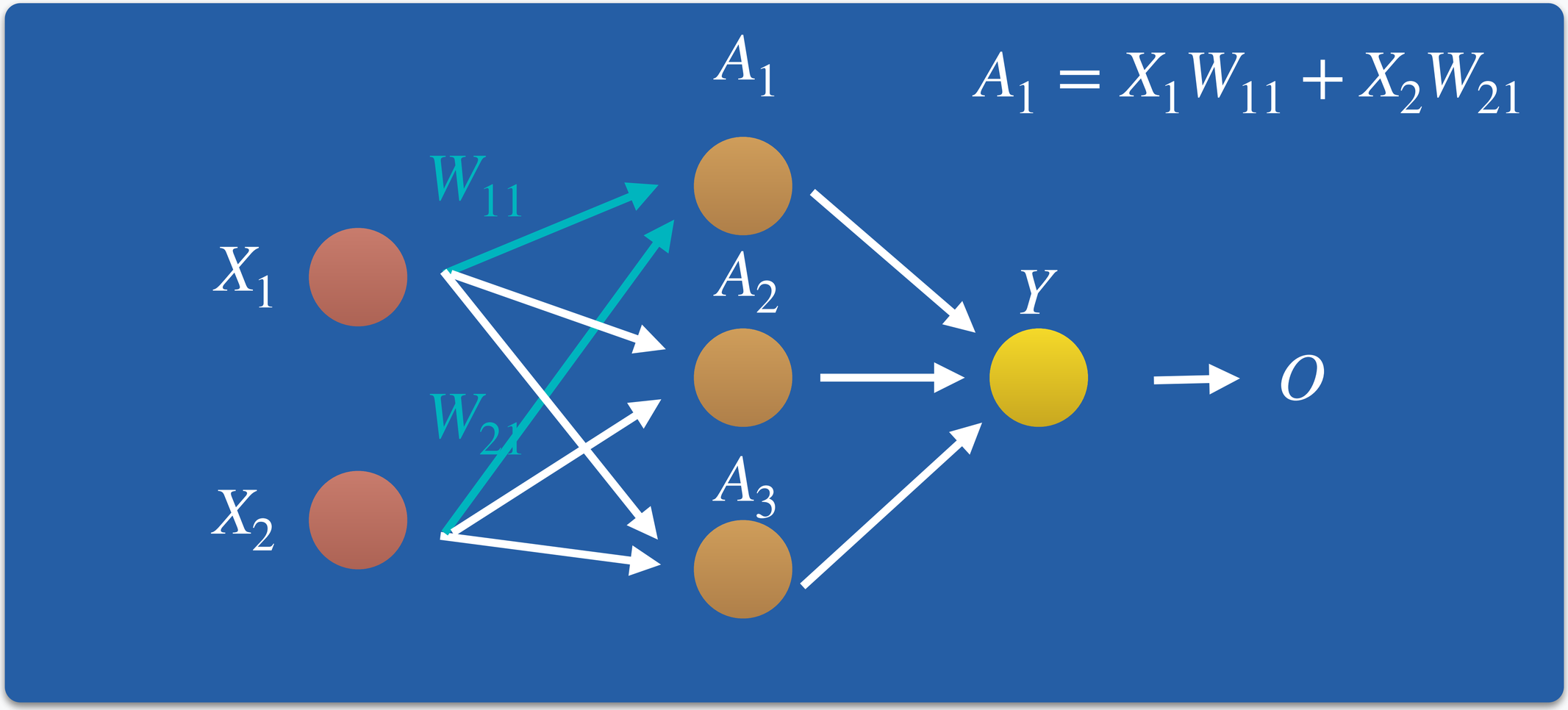
Neuron



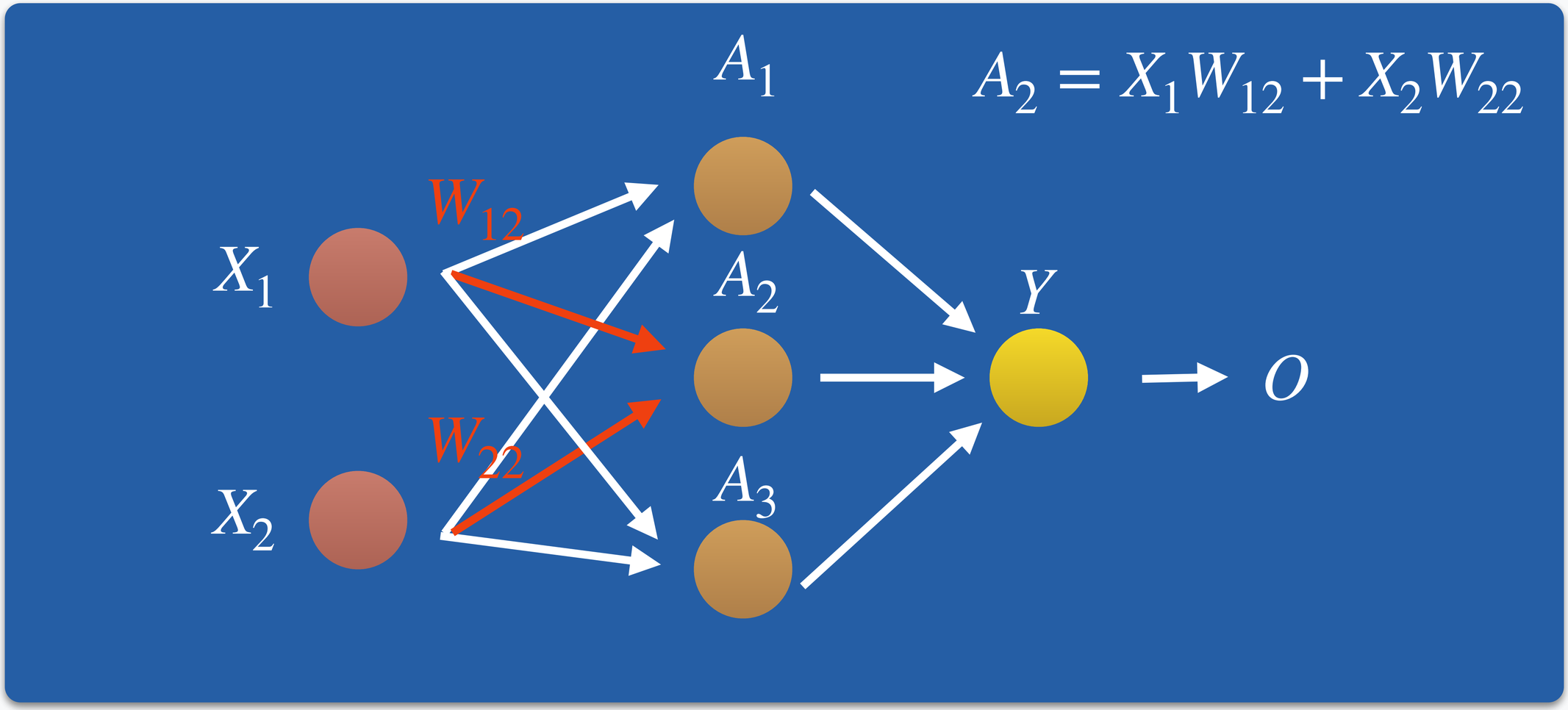
Deep neural network



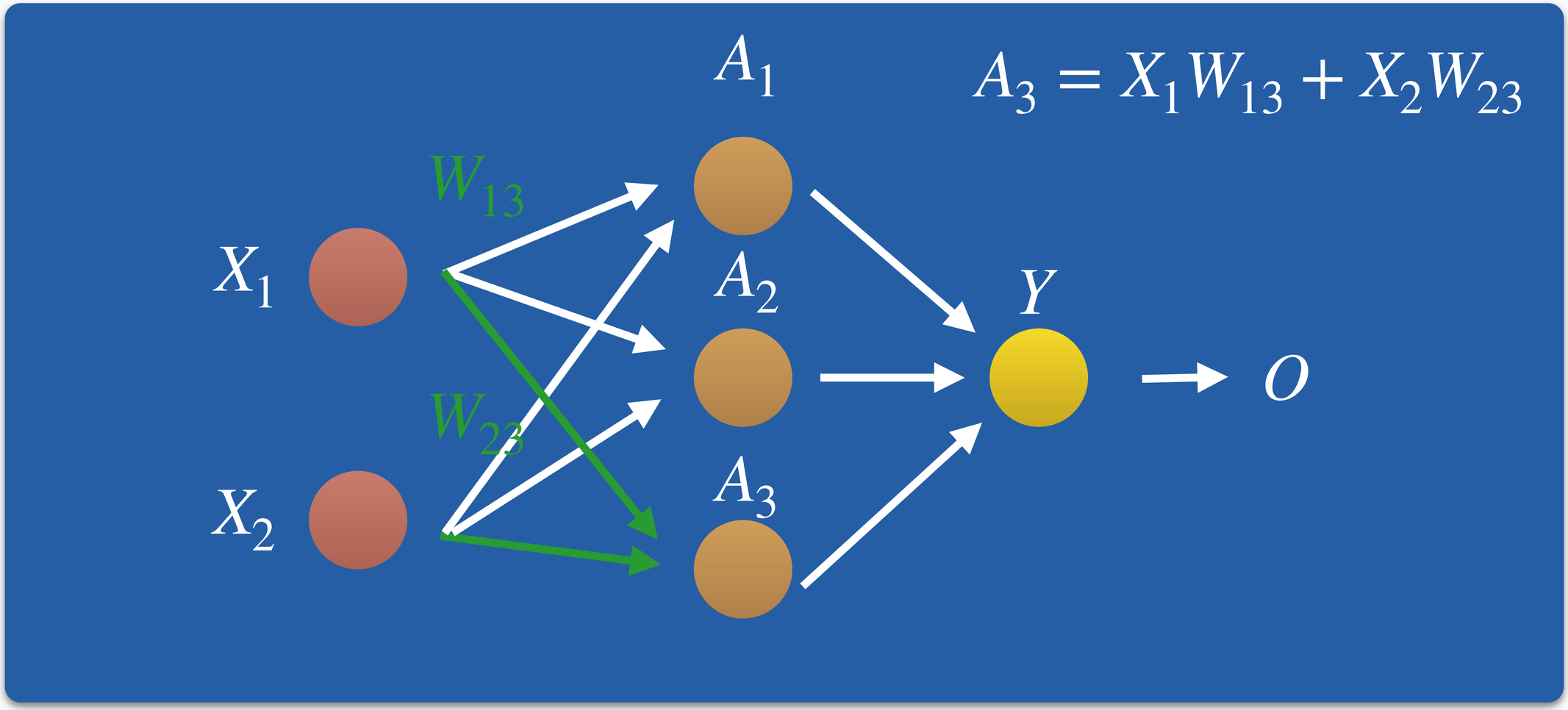
Forward propagation



Forward propagation



Forward propagation



Matrizen Multiplizieren

$$A_{11} = X_1 W_{11} + X_2 W_{21}$$

$$A_{12} = X_1 W_{12} + X_2 W_{22}$$

$$A_{13} = X_1 W_{13} + X_2 W_{23}$$

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \end{bmatrix}$$

$$X = \begin{bmatrix} X_1 & X_2 \end{bmatrix} \quad A = \begin{bmatrix} A_{11} & A_{12} & A_{12} \end{bmatrix}$$

Matrizen Multiplizieren

In Kotlin als Tensor

```
fun matmul(other: Tensor): Tensor {
    // Ensure that matrices can be multiplied (cols of first matrix == rows of second matrix)
    if (this.cols != other.rows) {
        throw IllegalArgumentException("Cannot multiply matrices: incompatible dimensions.")
    }

    // Create a result matrix with the appropriate size
    val result = Tensor(Array(this.rows) { Array(other.cols) { 0.0 } })

    // Perform the multiplication
    for (i in 0 until this.rows) {
        for (j in 0 until other.cols) {
            var sum = 0.0
            for (k in 0 until this.cols) {
                sum += this[i, k] * other[k, j]
            }
            result[i, j] = sum
        }
    }
    return result
}
```

Softmax

Aktivierungsfunktion

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

```
class Tensor(val values: List<Float>) {

    fun softmax(): Tensor {
        // Exponentielle Transformation
        val expValues = values.map { Math.exp(it.toDouble()).toFloat() }
        val sumExpValues = expValues.sum()

        // Normierung
        val softmaxValues = expValues.map { it / sumExpValues }

        // Einen neuen Tensor mit den Softmax-Werten zurückgeben
        return Tensor(softmaxValues)
    }
}
```

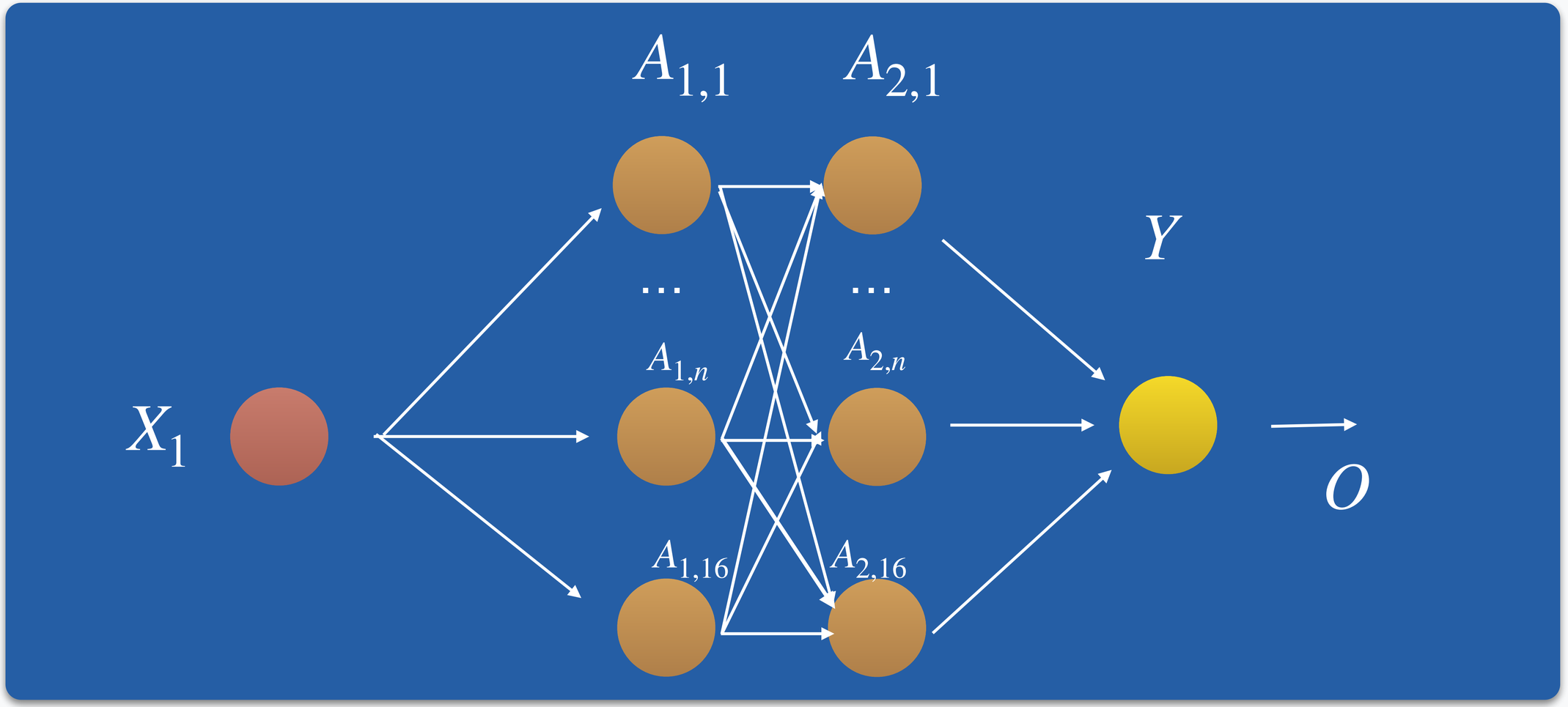
Forward Propagation in Kotlin

Versteckte Schicht

```
class Linear() {
    val weights:Tensor
    val bias:Tensor

     fun forward(input: Tensor): Tensor {
       val output = input.matmul(weight.value.t()) + bias.value
       return output
     }
}
```

Sinus als neuronales Netzt



Forward Propagation in Kotlin

Tiefes Neuronales Netz

```
class SineNN(override val name: String="sin") : Module() {

    private val sineModule =  sequential{
        input(1)
        linear(16, "layer1") {
            activation = relu
        }
        linear(16, "layer2") {
            activation = relu
        }
        linear(1, "output_layer")
    }
    override val modules: List<Module>
        get() = sineModule.modules

    override fun forward(input: Tensor): Tensor =
        sineModule.forward(input)
}
```

Forward Propagation in Kotlin

1 Eingang, 2 Verstecke Schicken mit 16 Neuronen,

Schicht	Gewichte	Biases	Zusammen
Eingabe Schicht	16	16	32
1.Schicht => 2.Schicht	256	16	272
2.Schicht => Ausgang	16	1	17
Summe	288	33	321

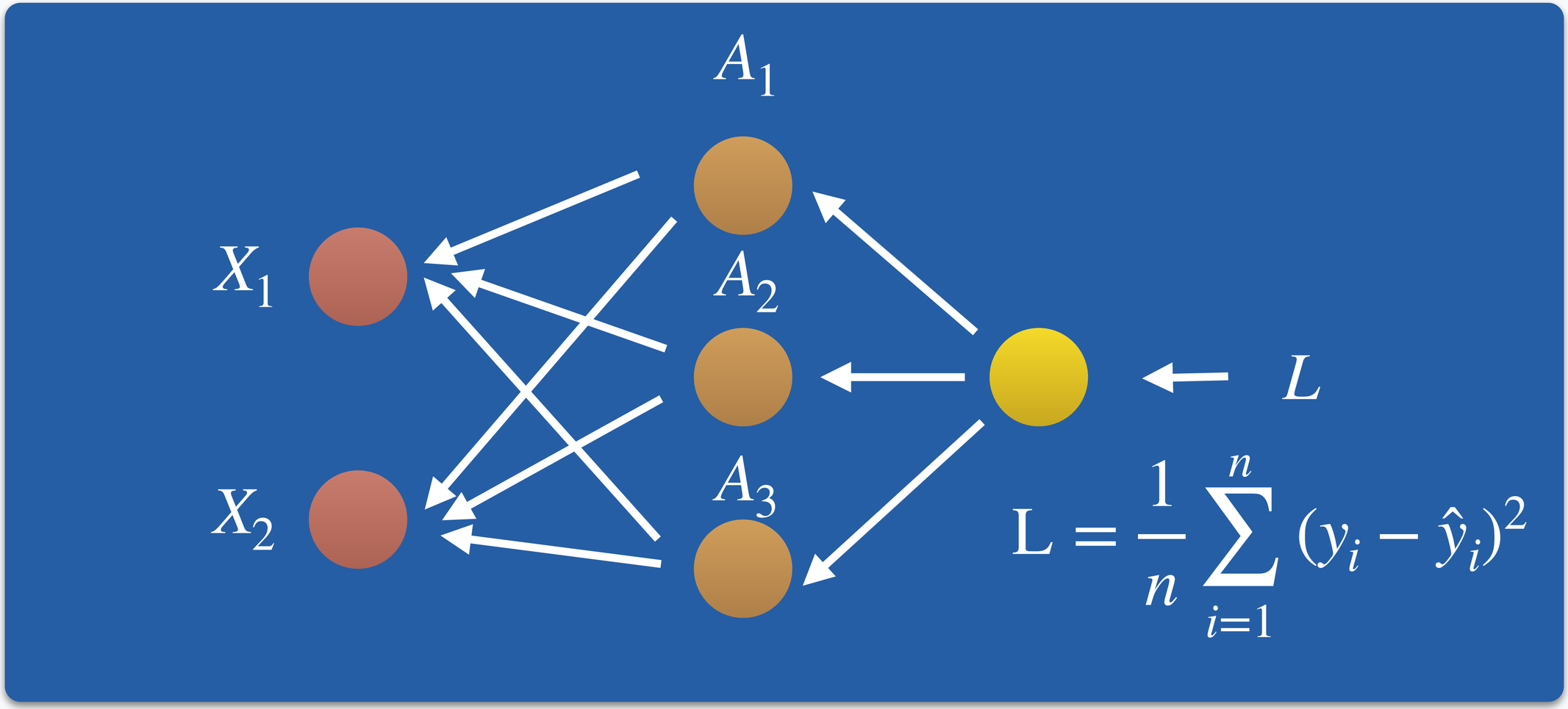
Trainieren

(C) Adriana Harakalova, 2024

Überwachtes Lernen

PARADIGMA DES MASCHINELLEN LERNENS

Rückwärtspropagation



Verlustfunktion

Mittlerer Quadratischer Fehler

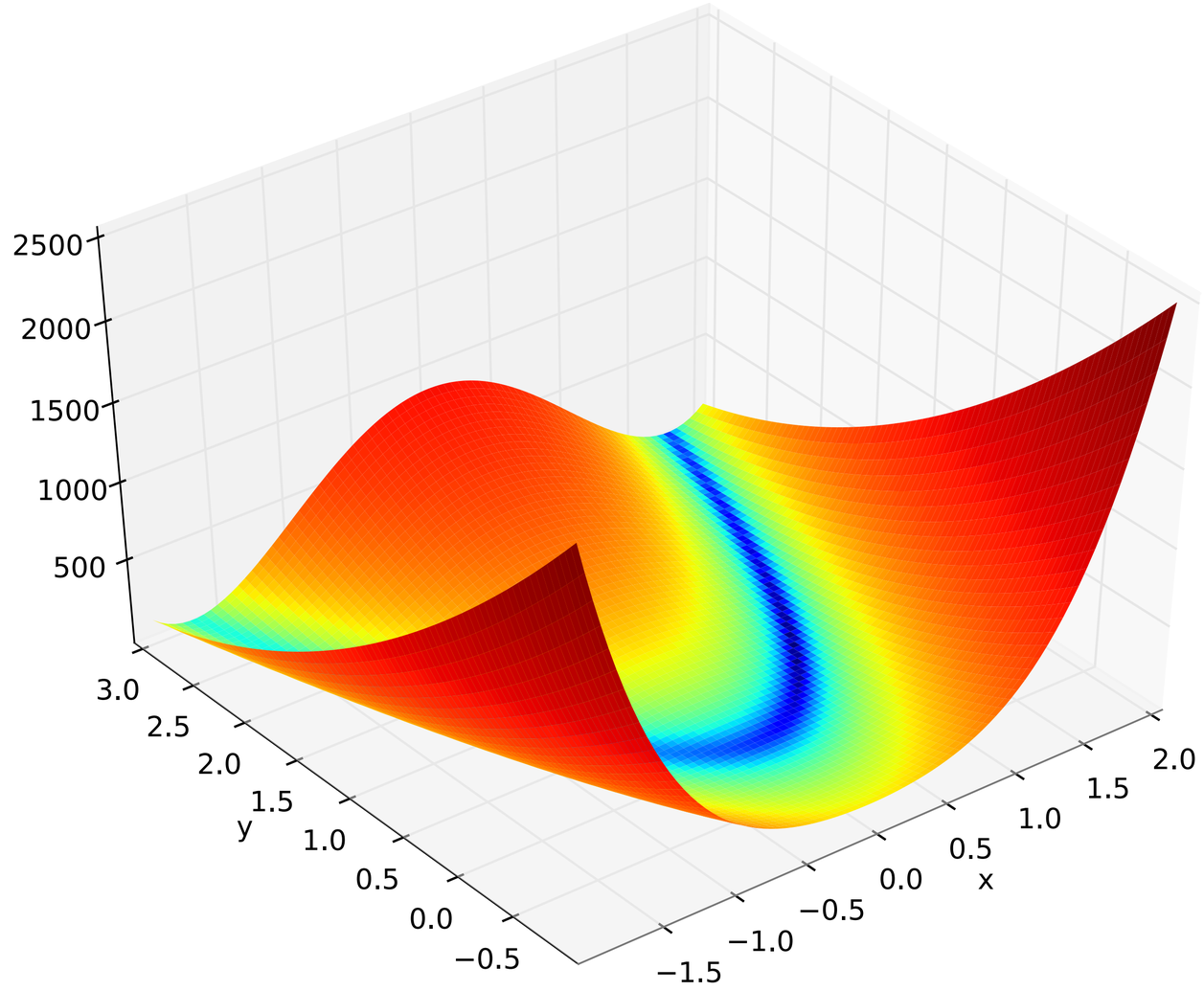
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

```kotlin
// Extension Function für die MSE-Berechnung
fun Tensor.meanSquaredError(other: Tensor): Double {
    // Überprüfen, ob die Tensoren dieselbe Größe haben
    if (this.size != other.size) {
        throw IllegalArgumentException("Tensoren müssen die gleiche Größe haben.")
    }

    // Berechnung der Fehlerquadrate und des Mittels
    var sum = 0.0
    for (i in 0 until this.size) {
        val error = this[i] - other[i]
        sum += error * error
    }
    return sum / this.size
}
```

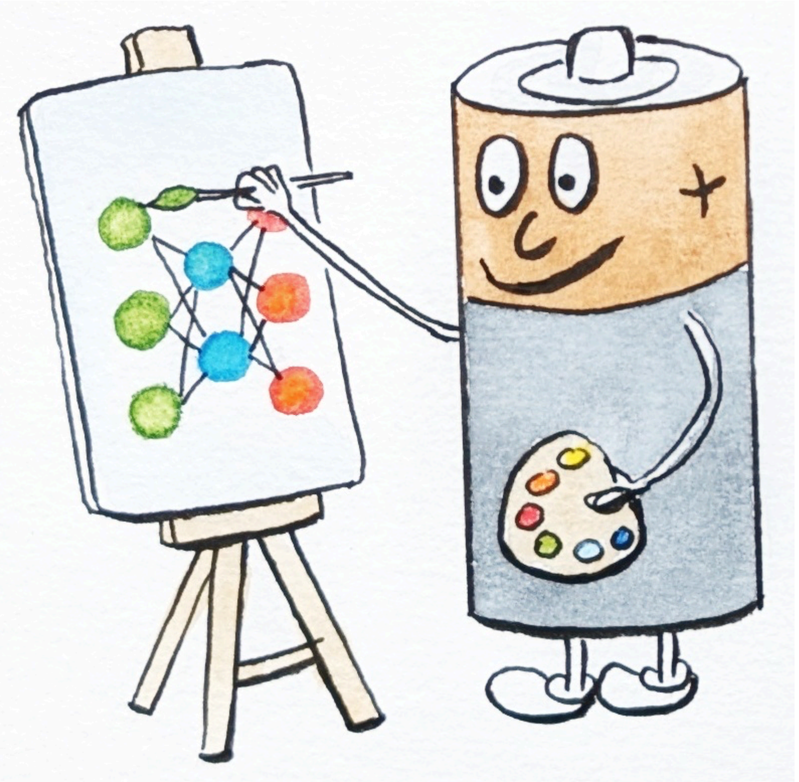
Stochastic gradient descent

Optimierungsverfahren



https://commons.wikimedia.org/wiki/File:Rosenbrock_function.svg

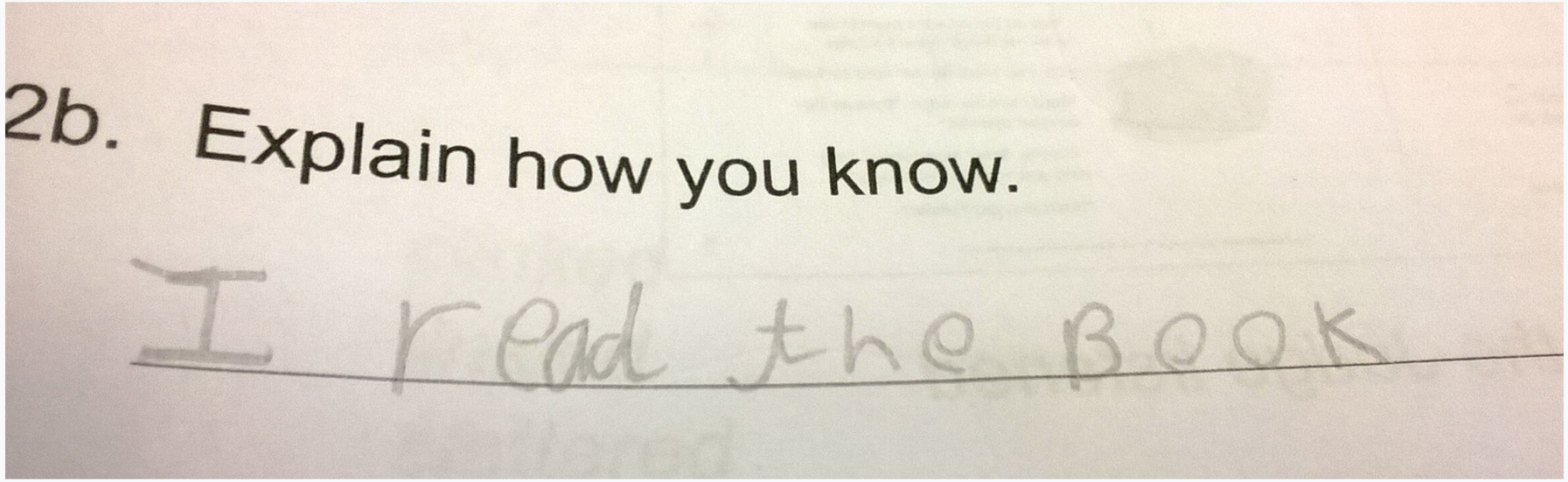
Generative



(C) Adriana Harakalova, 2024

Generative

Mustererkennung und Vorhersage



Input Text Processing

- **Tokenisierung** - ein Prozess, bei dem ein Text in kleinere Einheiten wie Wörter, Sätze oder Zeichen zerlegt wird, die als "Tokens" bezeichnet werden.
- **Encoding** - Jedem Token eine eindeutige numerische Kennung zuweisen.
- *GTP-2* has cca 50,257 tokens (**Byte Pair Encoding**)

Tokenisierung und Kodierung

Beispiel mit Zeichen basierte Kodierung

H A L L O Text

Tokenisierung und Kodierung

Beispiel mit Zeichen basierte Kodierung

H A L L O Text

‘H‘ ‘A‘ L‘ L‘ ‘O‘ Tokens

Tokenisierung und Kodierung

Beispiel mit Zeichenkodierung

H A L L O — Text

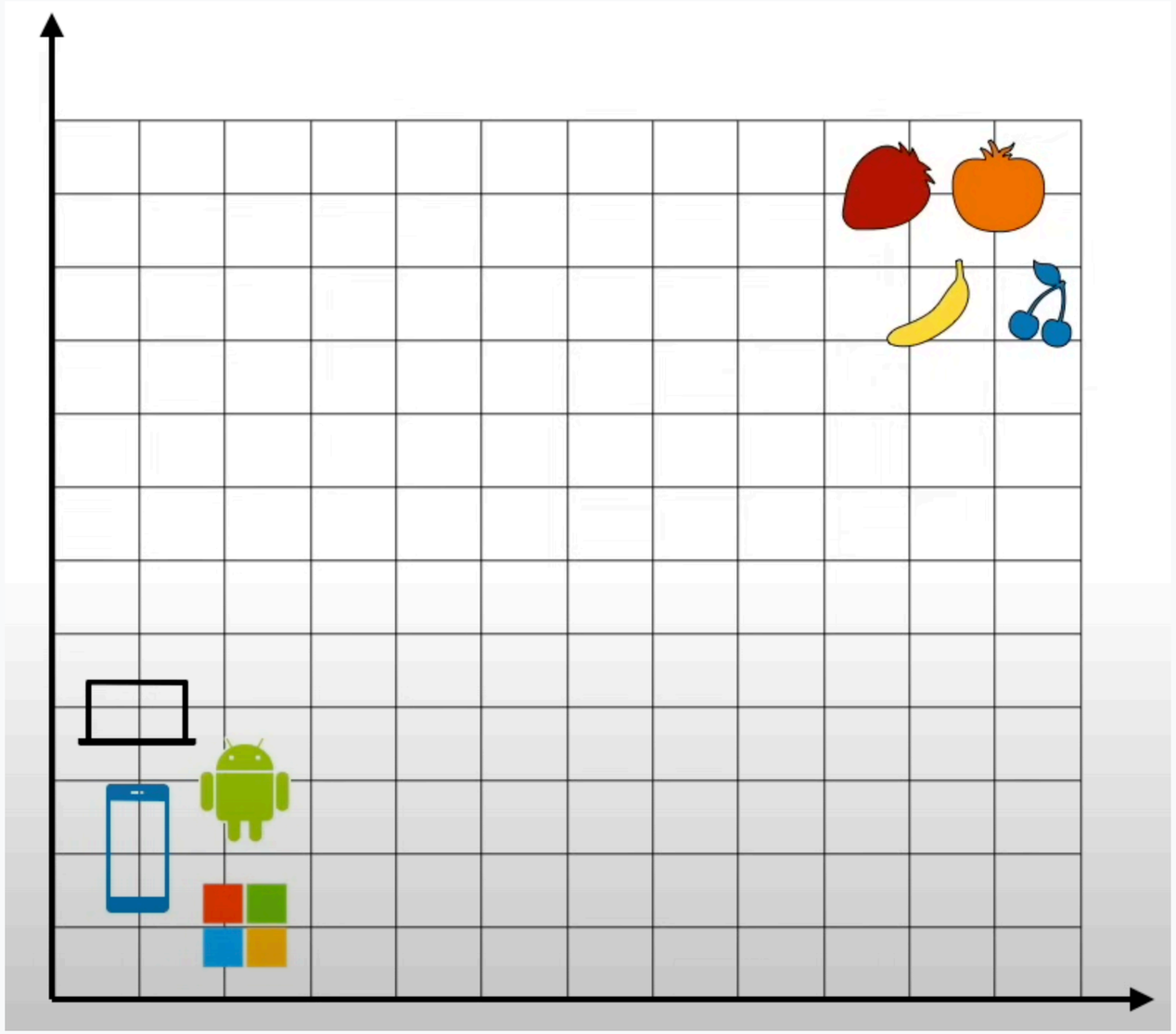
‘H‘ ‘A‘ L‘ L‘ ‘O‘ — Tokens

$\vec{v} = (8 \quad 1 \quad 12 \quad 12 \quad 15)$ — numerische Sequenz

Embedings

Text als Zahlen
in multidimensionalem Raum

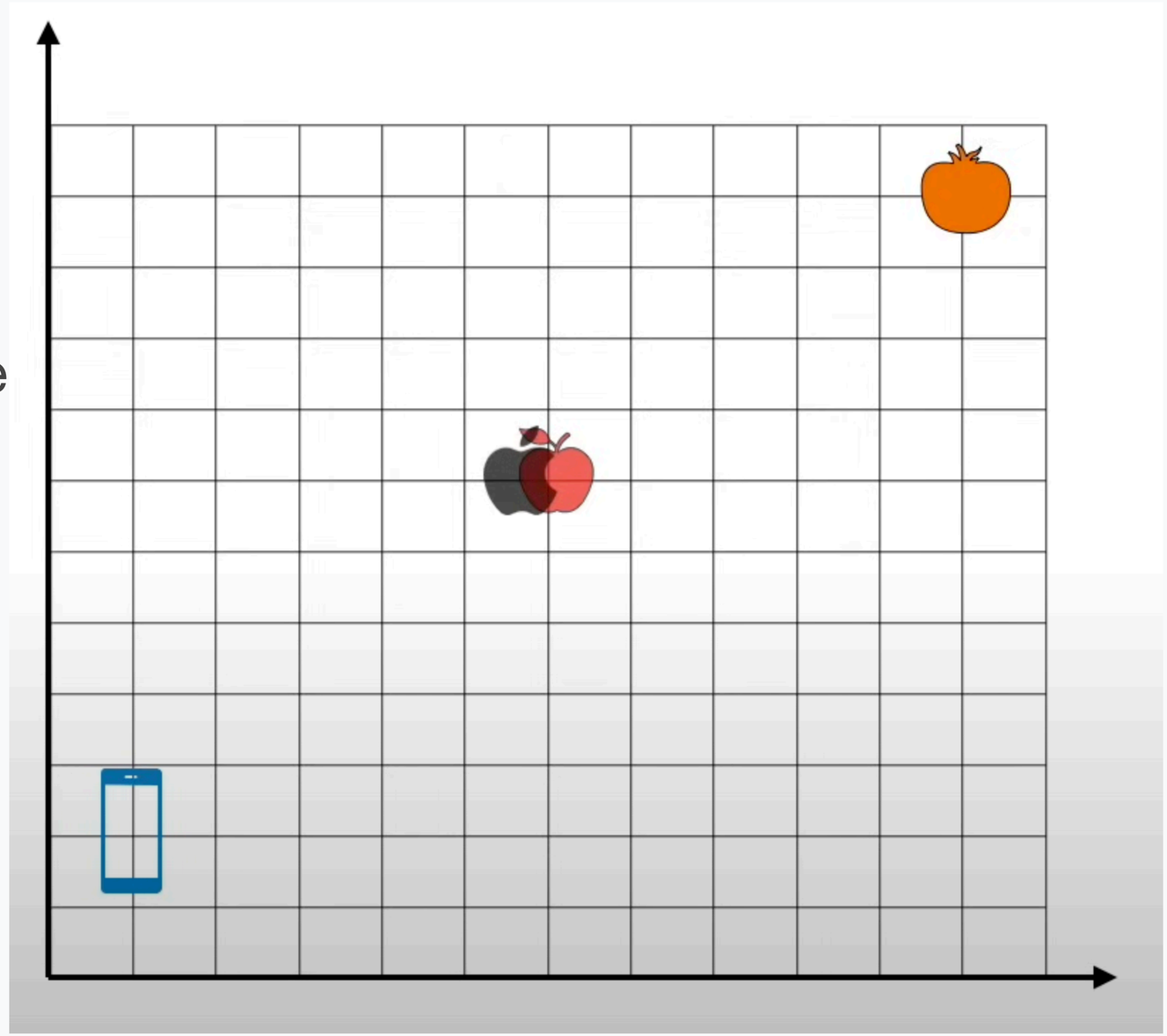
GTP2-768 Dimensionen



https://www.youtube.com/watch?v=OxCpWwDCDFQ&list=PLs8w1Cdi-zvYskDS2icIItfZgxclApVLv

Embeddings

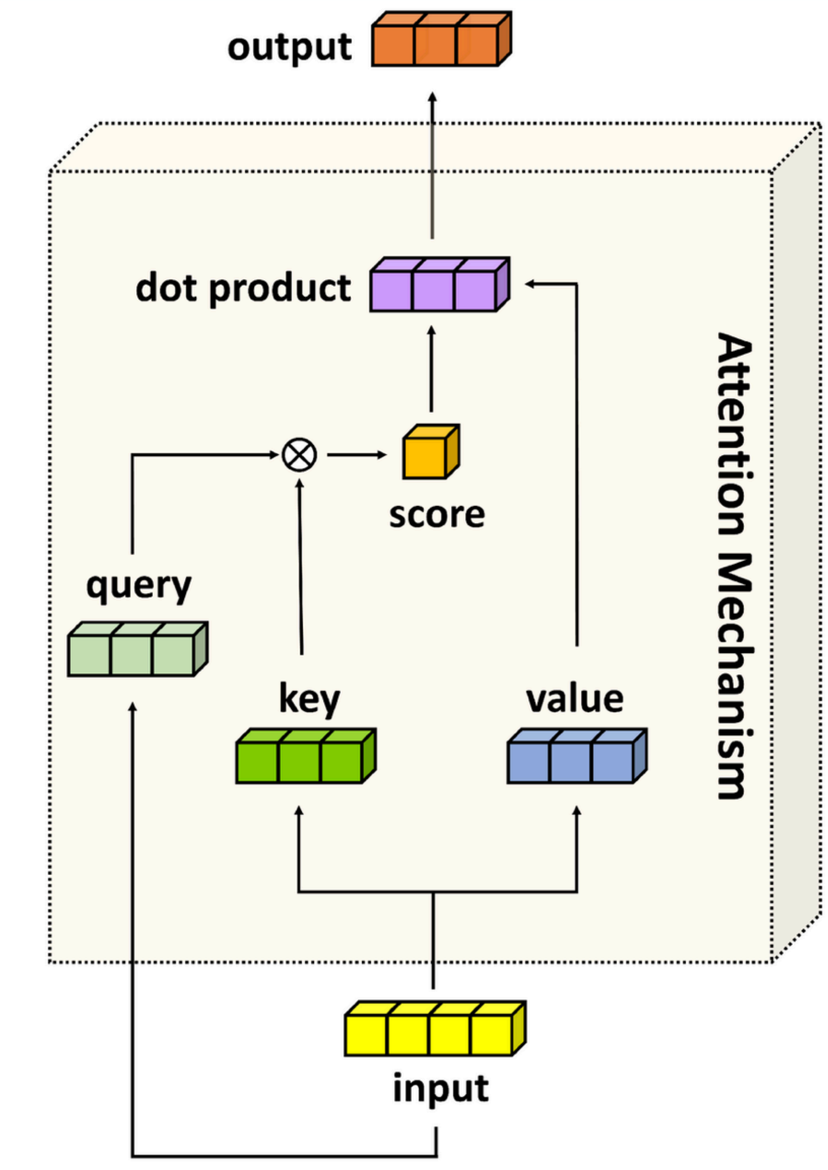
- Please buy an apple and an orange
- Apple unveiled the new phone
- CONTEXT



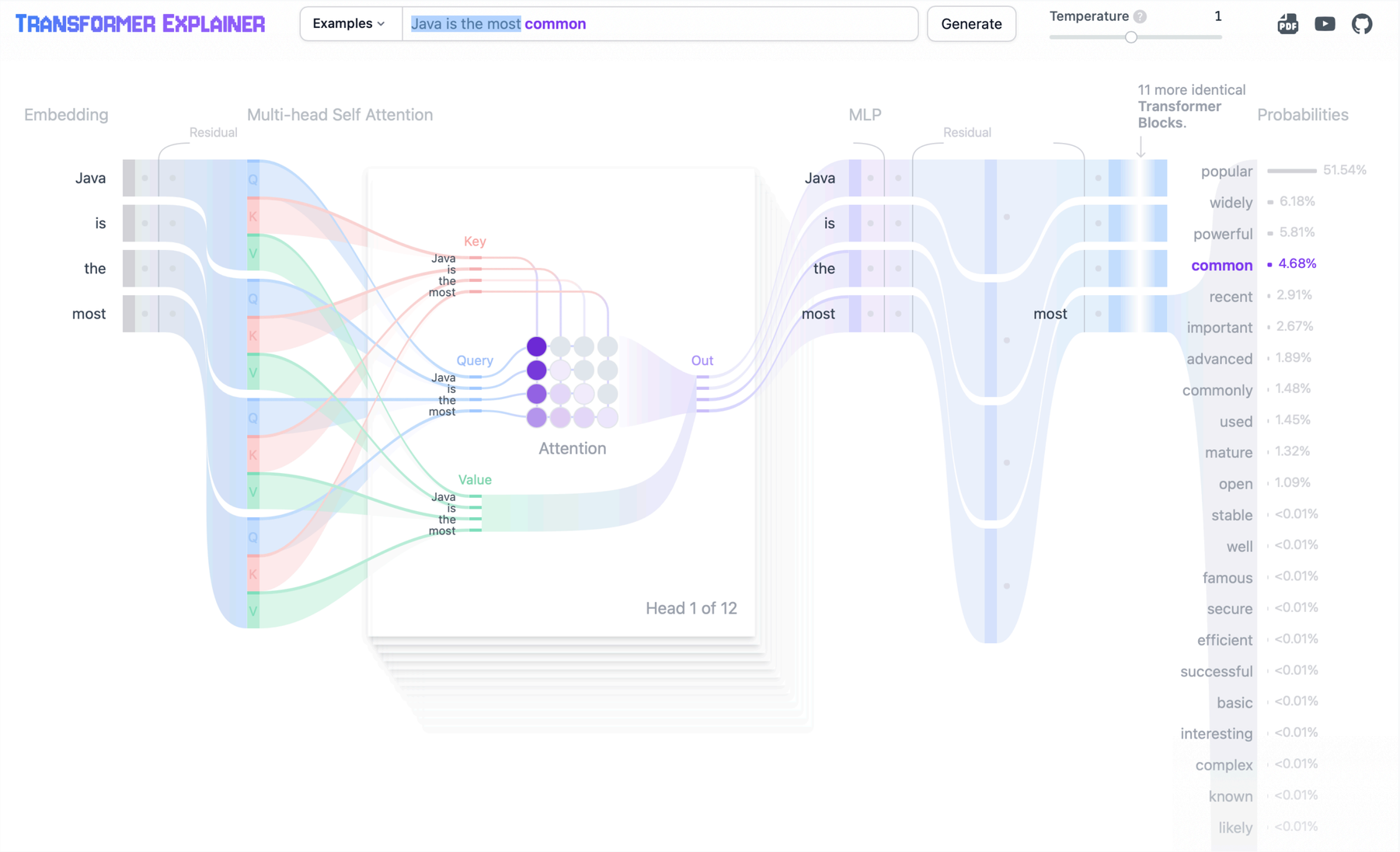
https://www.youtube.com/watch?v=OxCpWwDCDFQ&list=PLs8w1Cdi-zvYskDS2icIItfZgxclApVLv

Attention

- Query (Q)
- Key (K)
- Value (V)
- Trainierte Werte



Attention


TRANSFORMER EXPLAINER
Examples
Java is the most common
Generate
Temperature
1
Embedding
Multi-head Self Attention
MLP
11 more identical Transformer Blocks.
Probabilities
Residual
Residual
Java
is
the
most
Q
K
V
Key
Query
Value
Attention
Out
Head 1 of 12
popular 51.54%
widely 6.18%
powerful 5.81%
common 4.68%
recent 2.91%
important 2.67%
advanced 1.89%
commonly 1.48%
used 1.45%
mature 1.32%
open 1.09%
stable <0.01%
well <0.01%
famous <0.01%
secure <0.01%
efficient <0.01%
successful <0.01%
basic <0.01%
interesting <0.01%
complex <0.01%
known <0.01%
likely <0.01%

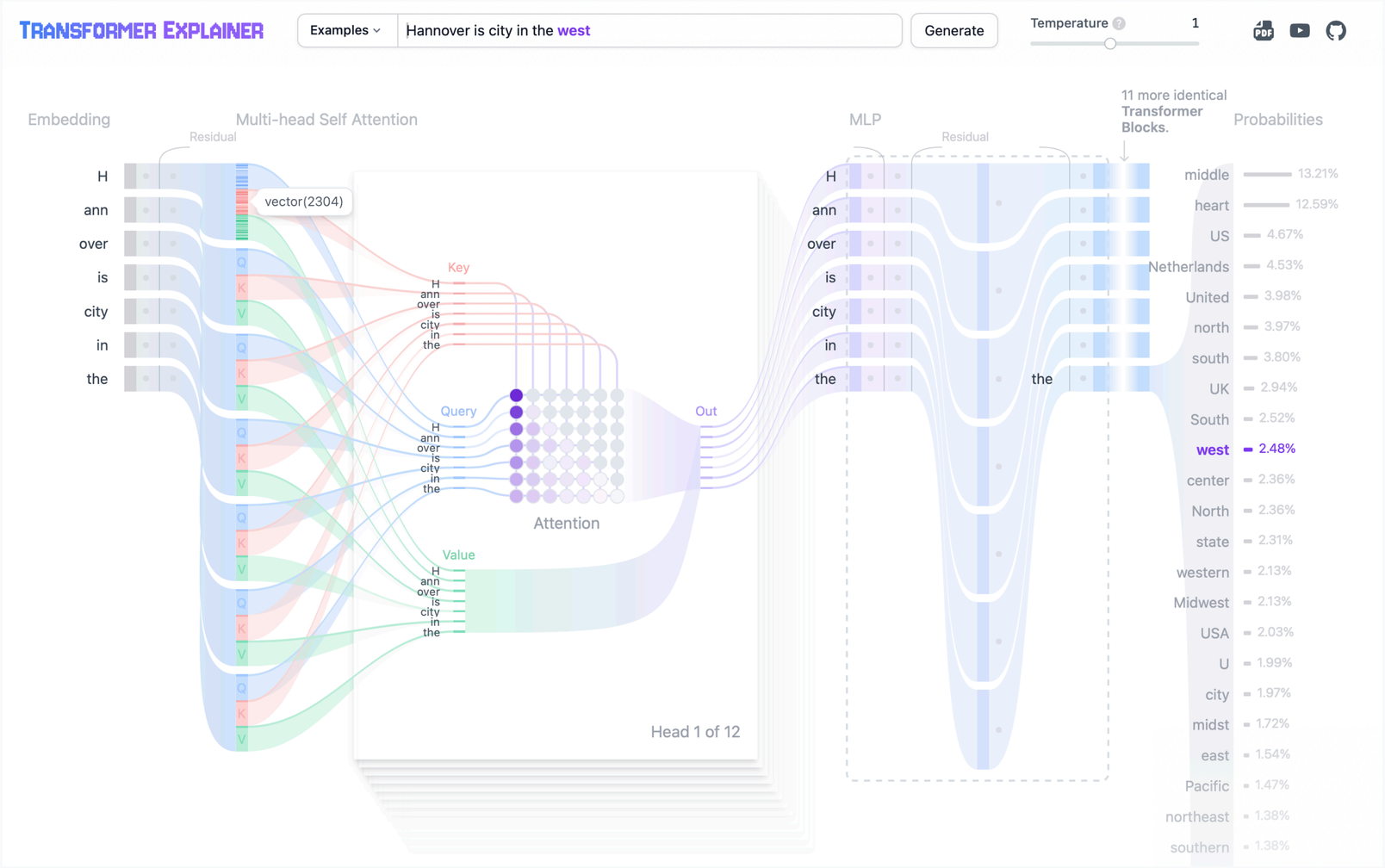
GTP-2 Visualization

2D und 3D

- https://bbycroft.net/llm
- https://poloclub.github.io/transformer-explainer/

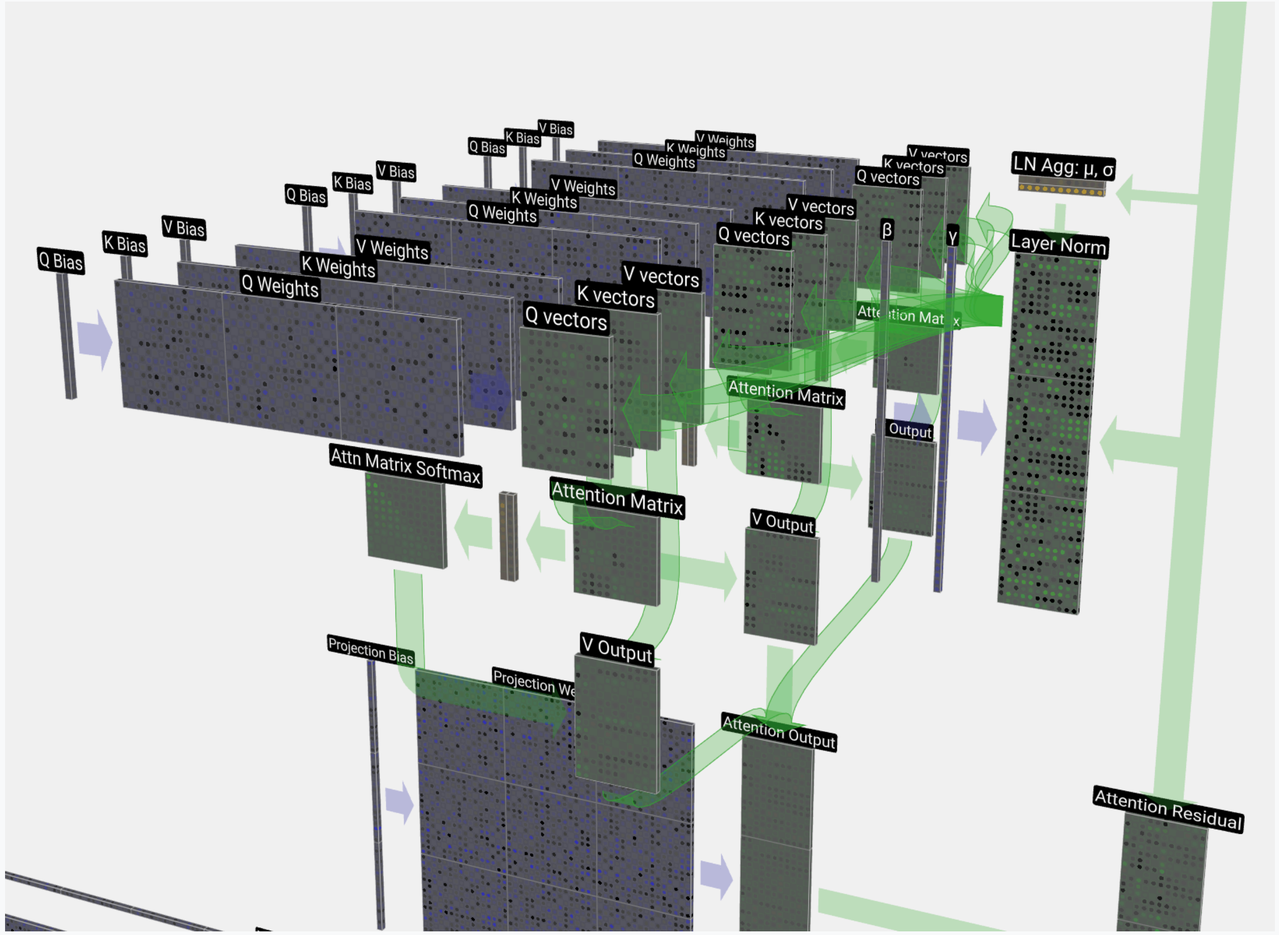
2D GTP-2 Visualization

https://poloclub.github.io/transformer-explainer/

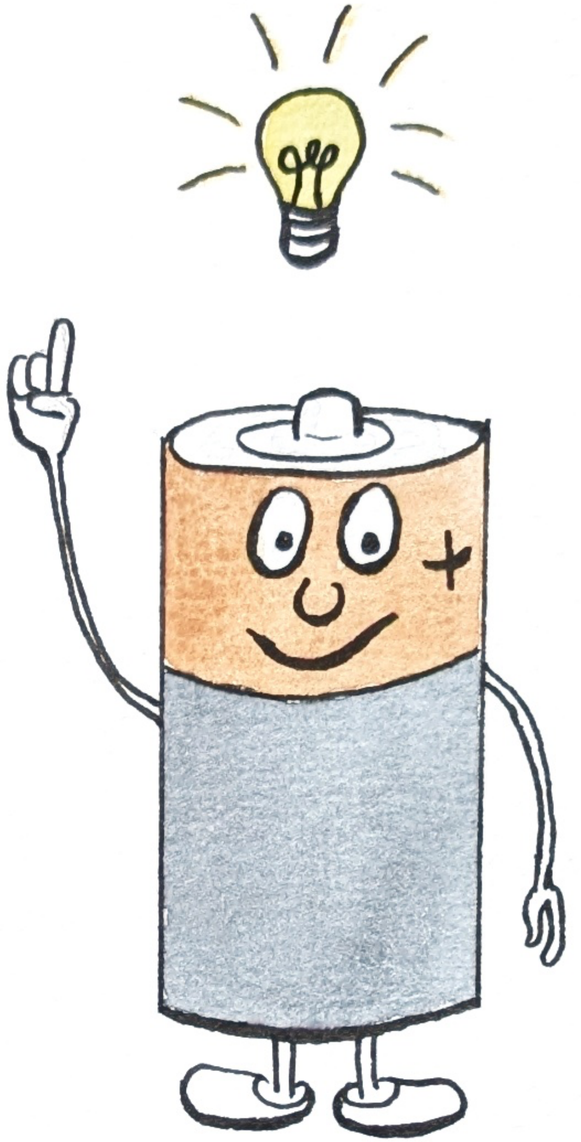


3D nanoGTP Visualization

https://bbycroft.net/llm



SKaiNET



Eigene ML Bibliothek in Kotlin

Kotlin Bibliotheken

- Auf den Schultern von Riesen stehen
- Kotlin Port von nanoGPT -> 80%
    - Jetzt als Kotlin Multiplattform
      https://github.com/michalharakal/KPTChat/tree/feature/kmp
- Kotlin Port vom mikrograd -> 90%
    - Mit eigener DSL - Sprache für *graphviz DOT* Sprache
      https://github.com/michalharakal/miKrograd
- Testing Framework für Validieren mit Pytorch Berechnungen
    - https://github.com/michalharakal/gradienttracer

miKrograd

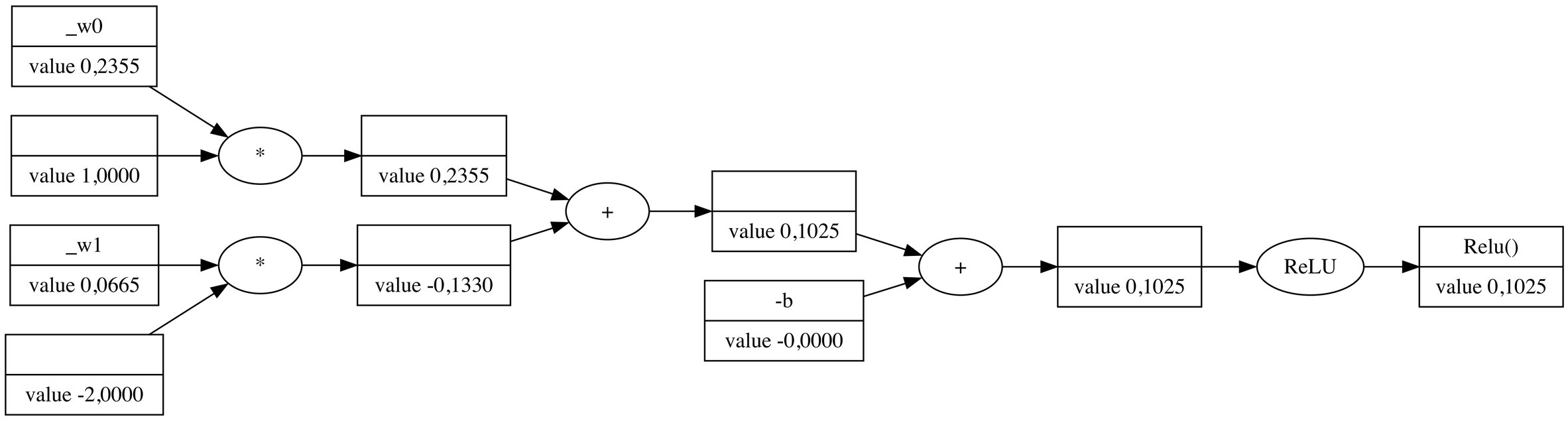
Autodifferenzierung

```
import org.mikrograd.diff.Value
import org.mikrograd.diff.div
import org.mikrograd.diff.plus

val a = Value(-4.0)
val b = Value(2.0)
var c = a + b
var d = a * b + b.pow(3.0)
c += c + 1
c += 1.0 + c + (-a)
d += d * 2 + (b + a).relu()
d += d * 3.0 + (b - a).relu()
val e = c - d
val f = e.pow(2.0)
var g = f / 2
g += 10.0 / f
println("$g")  // prints 24.7041, the outcome of this forward pass
g.backward()
println("${a.grad}") // prints 138.8338, i.e. the numerical value of dg/da
println("${b.grad}") // prints 645.5773, i.e. the numerical value of dg/db```
```

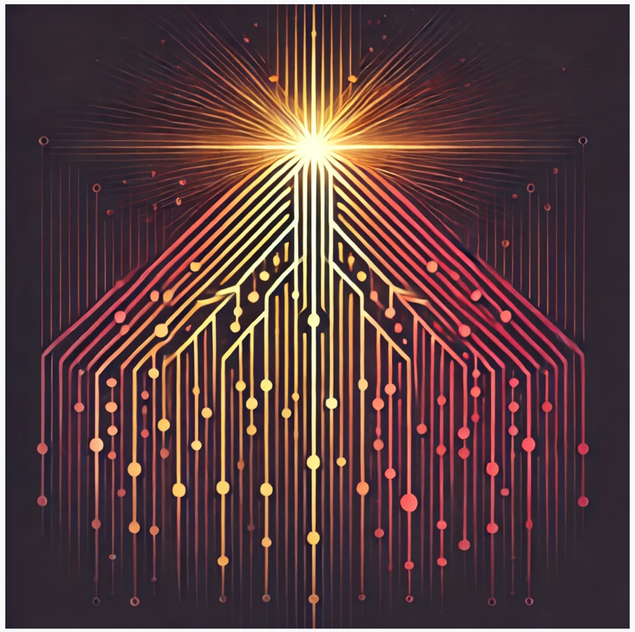
miKrograd

Visualisierung von DAG



gradienttracer

Testing Kotlin Implementierung gegen Pytorch Berechnungen



```
your_data/
├── core.py
├── TS-001/
│   └── UC-001.py
│   └── UC-002.py
├── TS-002/
│   └── UC-001.py
│   └── UC-002.py
└── TS-003/
    └── UC-001.py
    └── UC-002.py
```

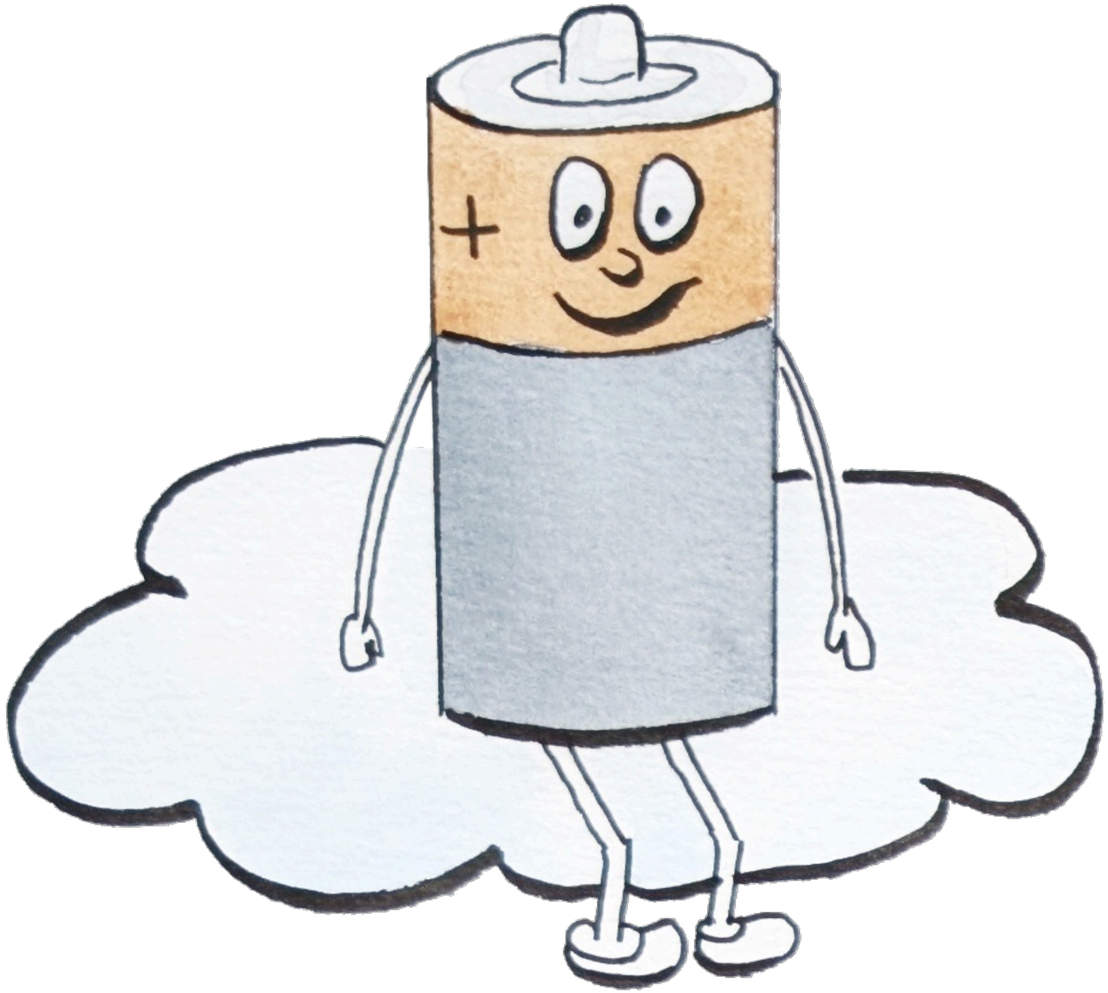
gradienttracer

Testfälle in Python

```python
from core import Executable
import torch


@Executable('simple expression with relu')
def simple_expression_with_relu():
    x = torch.Tensor([-4.0]).double()
    x.requires_grad = True
    z = 2 * x + 2 + x
    q = z.relu() + z * x
    h = (z * z).relu()
    y = h + q + q * x
    y.backward()
    return [x], y
```

Demo



(C) Adriana Harakalova, 2024


+
Batterie

Danke

Folien und Quellcode

https://github.com/michalharakal/from-zero-to-kotlin-gpt/tree/jfn2024

Links

Quellenangabe

- Attention Is All You Need - https://arxiv.org/pdf/1706.03762
- Let's build GPT: from scratch, in code, spelled out (https://www.youtube.com/watch?v=kCc8FmEb1nY)
- Pytorch logo https://commons.wikimedia.org/wiki/File:Pytorch_logo.png
- https://www.neuromedia.ca/how-many-neurons-are-in-the-human-brain/
- The math behind Attention: Keys, Queries, and Values matrices https://www.youtube.com/watch?v=UPtG_38Oq8o&list=PLs8w1Cdi-zvYskDS2icIItfZgxclApVLv&index=2
- https://commons.wikimedia.org/wiki/File:Rosenbrock_function.svg https://github.com/michalharakal/KPTChat/tree/feature/kmp
- Kotlin Port vom mikrograd -> 90% Mit eigener DSL - Sprache für *graphviz DOT* Sprache - https://github.com/michalharakal/miKrograd
- https://lmos-ai.github.io/arc/
- Testing Framework für Validieren mit Pytorch Berechnungen- https://github.com/michalharakal/gradienttracer