Data Science

Artificial Intelligence

Tinglong Dai, PhD

Bernard T. Ferrari Professor

Johns Hopkins Carey Business School


LOVE IS ALL YOU NEED

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

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

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

https://bit.ly/attention17

Agenda

1

Week 1: Foundations

2

Week 2: Tools & Techniques

Machine learning, deep learning, Keras, Python, and no-code tools.

3

Week 3: CNNs & Business Models

Convolutional neural networks, using AI to create human value. Assignment #1 due.

4

Week 4: MLOps, RNNs, & Transformers

MLOps, RNNs, transformers, evaluation of AI. AI Lab proposal due.

5

Week 5: LLMs & GenAI

Large language models, generative AI, using AI to create business value. Assignment #2 due.

6

Week 6: Human-AI Interaction

Human-AI interaction, reinforcement learning, system design.

7

Week 7: Responsible AI

Responsible AI, future of AI. Lab final deliverable due.

8

Week 8: Individual Final Exam; Peer Reviews of AI Lab

Third TA Tutorial:
LLM API Programming and Creating AI Agents

🗓️ **Friday, 10/21, 12:00–1:00 PM**

📍 **Join via Zoom:** https://bit.ly/jhuaita25

Attendance is optional. Materials are on Canvas

AI Lab Project:

Feedback is on Canvas

Please reach out to Suhas for one-on-one meetings

Review of Group Assignment #2

Today 10 a.m.:

A Conversation with Dr. Michael Abramoff

Overfitting vs. Underfitting

Optimization vs. Generalization

- Machine learning has two tasks:
  - Optimization: adjust the model to best fit the training data
  - Generalization: ensure the model performs well on data it has *never* seen before

- Example: What's the best way to prepare for the U.S. medical licensing exam?
  - Memorizing answers to the sample tests ≠ doing well in the real test

- **Overfitting** occurs when the algorithm has learned too much from the training data

- **Underfitting** occurs when the algorithm hasn't modeled all relevant patterns in the training data

The Central Problem of Deep Learning: **Overfitting**

How to fight overfitting?

Key Strategies to Fight Overfitting

- Get more training data
- Reduce the size of the neural network (“pruning”)
- Add weight regularization
- Add dropout

Feedforward Neural Networks via Keras

Core Code

- Step 1. Defining a feedforward neural network:

```
model=keras.Sequential([
    layers.Dense(512,  activation = "relu"),
    layers.Dense(10, activation = "softmax")
])
```

- Step 2. Specifying the optimizer:

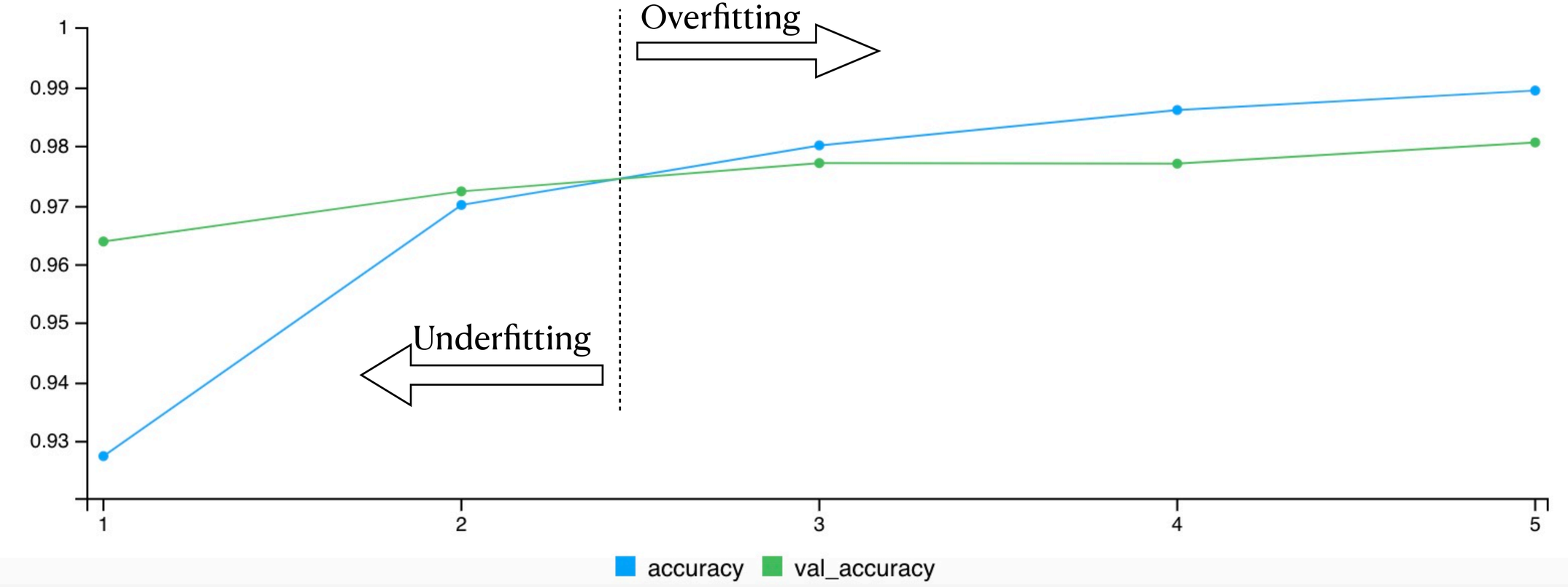
```
model.compile(optimizer="rmsprop",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
```

- Step 3. Train the model:

```
model.fit(
    train_images,train_labels,
    epochs=5,
    batch_size=128,
    validation_data=(test_images,test_labels),
    )
```

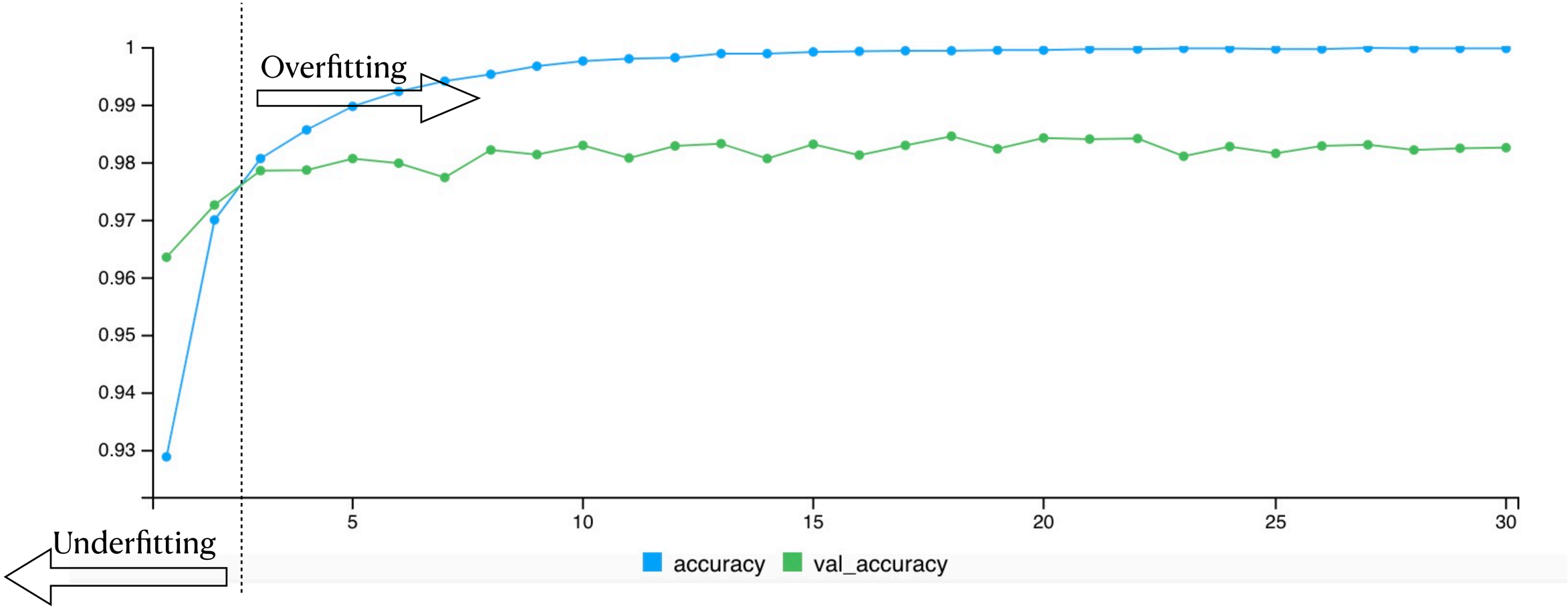
https://colab.research.google.com/drive/1An8WZ2pj7NOFChDouMkOWukoGtjFoDvp?usp=sharing

Is There an Overfitting Problem?



After 5 epochs, the accuracy is **98.94%** on the training data and **98.06%** on the test data

What If We Run 30 Epochs



After 30 epochs, the accuracy is **99.99%** on the training data and **98.26%** on the test data

Fighting Overfitting Using Dropout

- Instead of

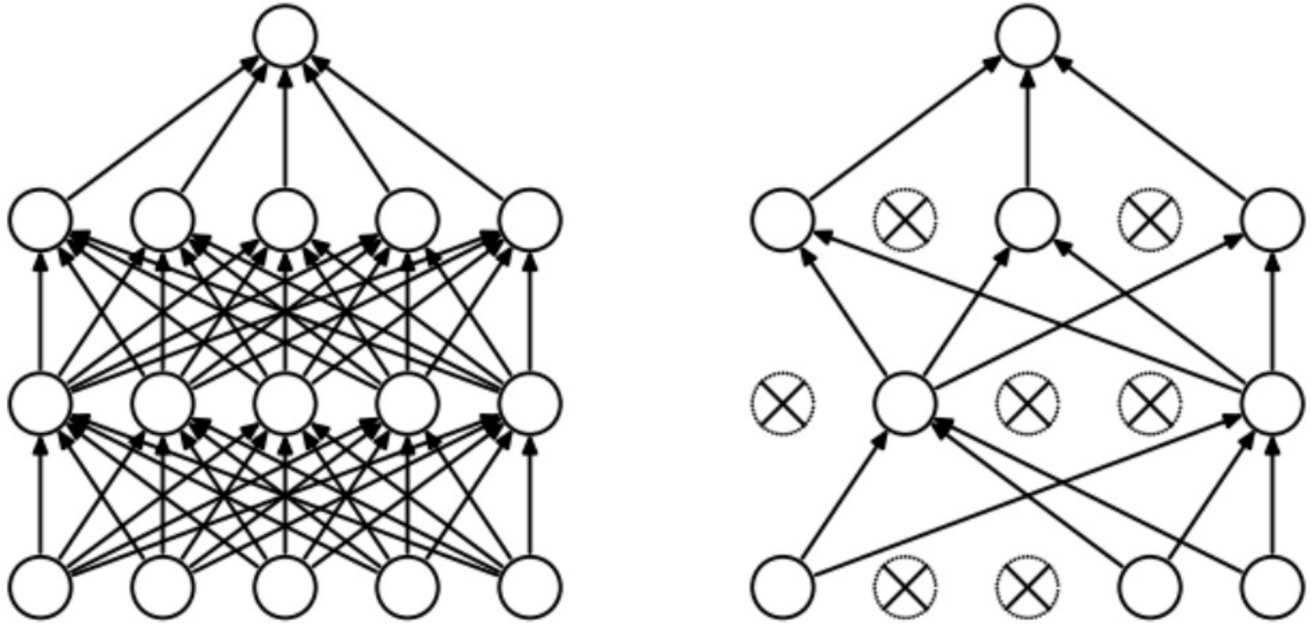
```
model=keras.Sequential([
    layers.Dense(512,  activation = "relu"),
    layers.Dense(10, activation = "softmax")
])
```

- Let's add a line of code to implement the dropout technique:

```
model=keras.Sequential([
    layers.Dense(512,  activation = "relu"),
    layers.Dropout(0.5),
    layers.Dense(10, activation = "softmax")
])
```

Dropout

Randomly dropping out (setting to zero) a number of output features



(a) Standard Neural Net

(b) After applying dropout.

Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

“I went to my bank. The tellers keep changing and I asked one of them why. He said he didn’t know but they got moved around a lot. I figured it must be because it would require cooperation between employees to successfully defraud the bank. This made me realize that randomly removing a different subset of neurons on each example would prevent conspiracies and thus reduce overfitting.”

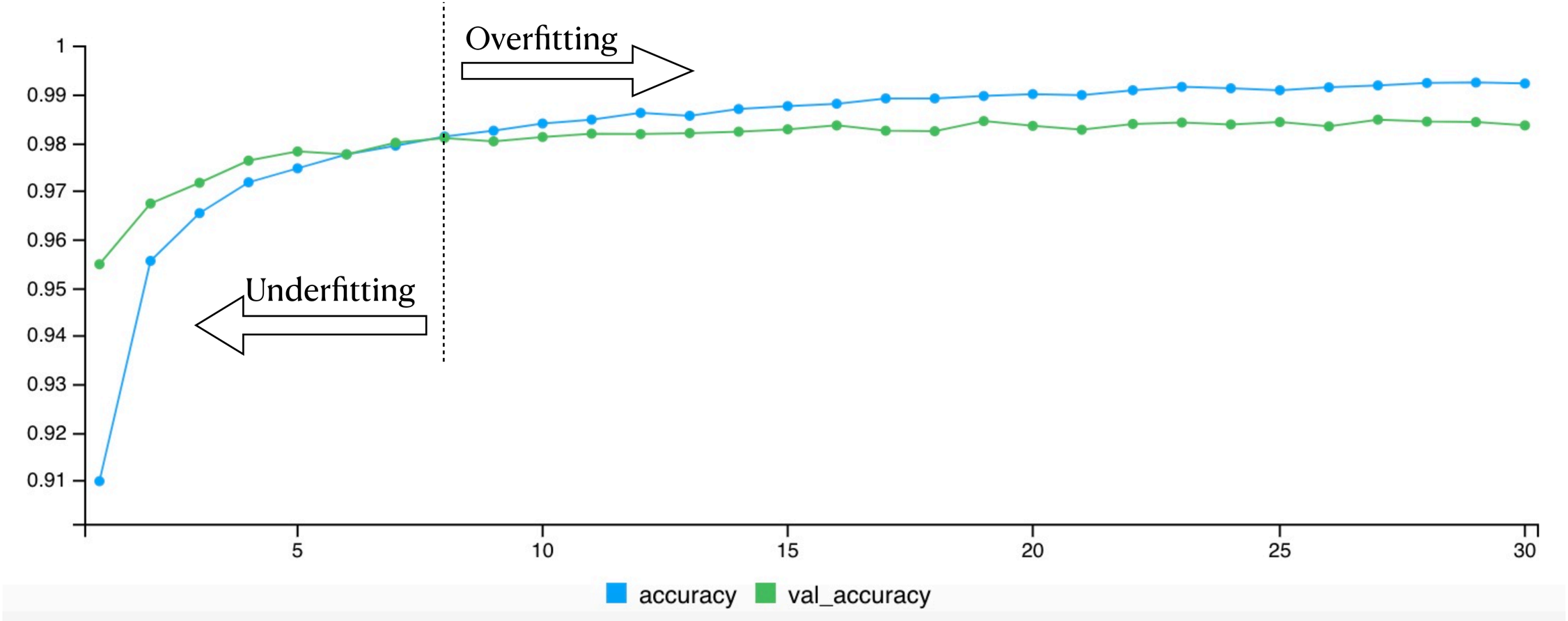
Geoffrey Hinton, inventor of the dropout technique

Why Dropout Works

- If a hidden unit knows which other hidden units are present, it can co-adapt to them on the training data
  - But complex co-adaptations are likely to go wrong on new test data
  - Big, complex conspiracies are not robust

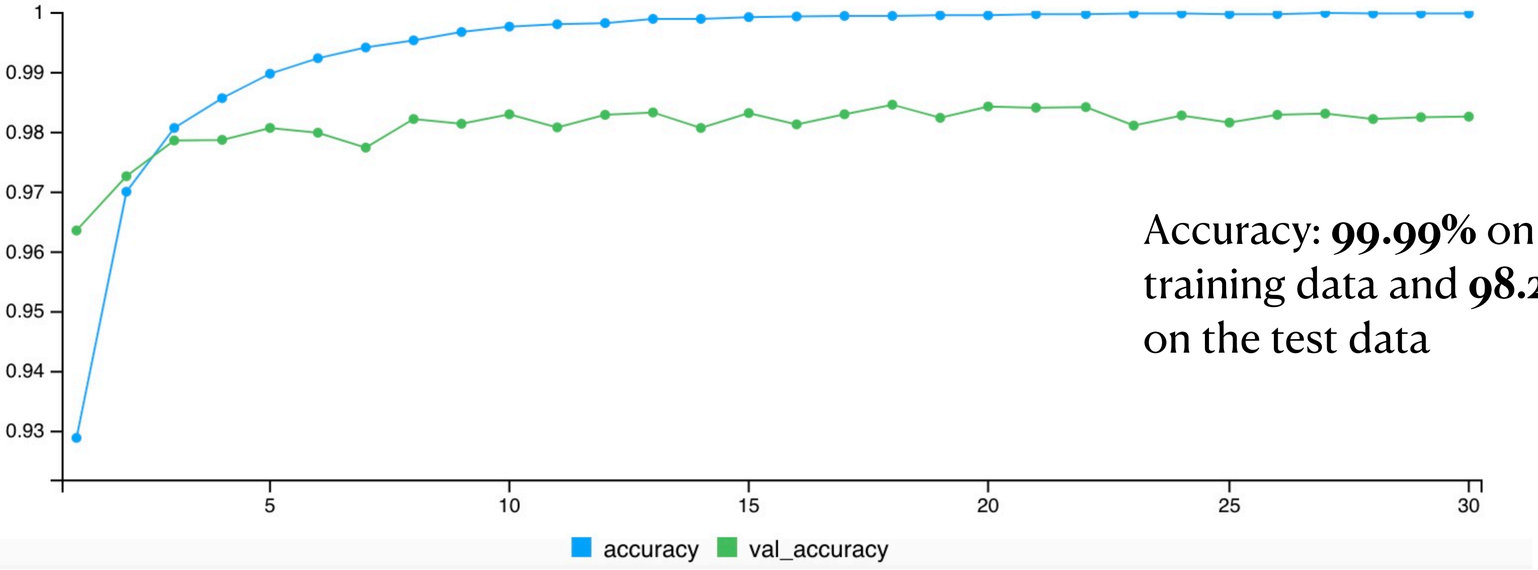
- If a hidden unit has to work well with combinatorial many sets of co-workers, it is more likely to do something that is individually useful
  - But it will also tend to do something that is marginally useful given what its co-workers achieve

Let's Training Our New Model with Dropout



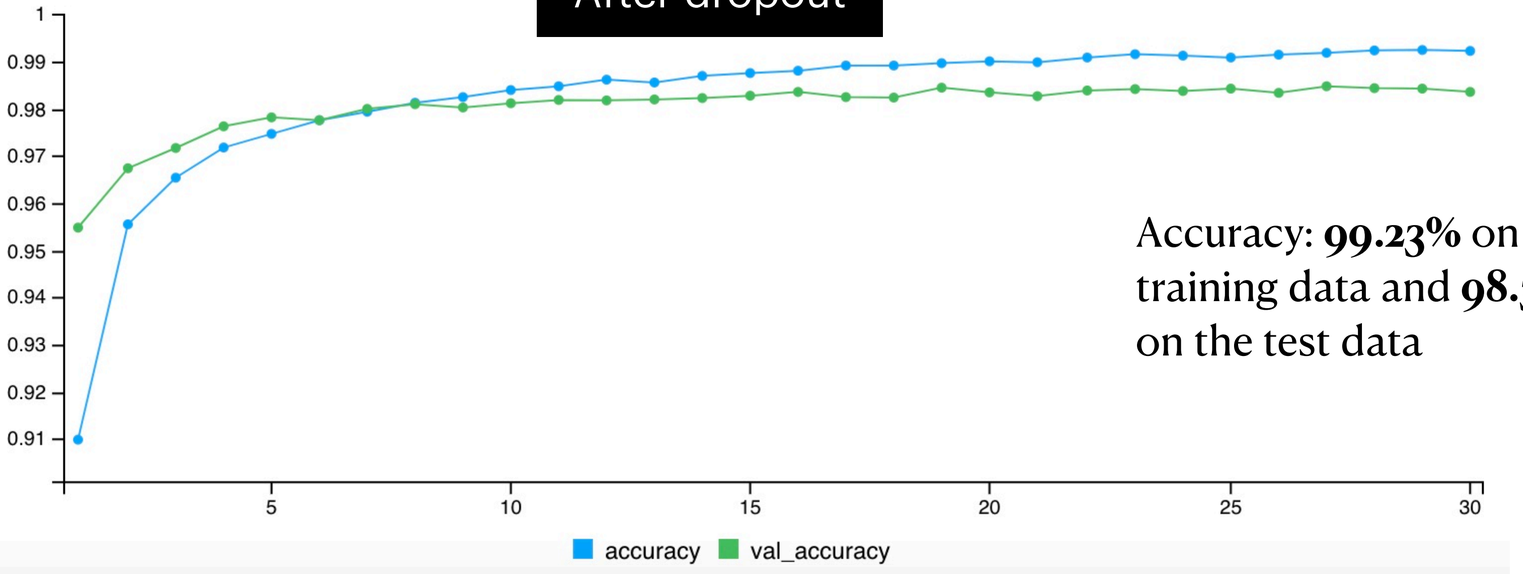
With the new model, the accuracy is **99.23%** on the training data and **98.36%** on the test data

Before dropout



Accuracy: **99.99%** on the training data and **98.26%** on the test data

After dropout



Accuracy: **99.23%** on the training data and **98.36%** on the test data

Can We Do Even Better?

- Yes! We can try a more complex neural network model with dropout
  - Use more hidden layers
  - Use more neurons per hidden layer
- What about using a CNN to train the same dataset?

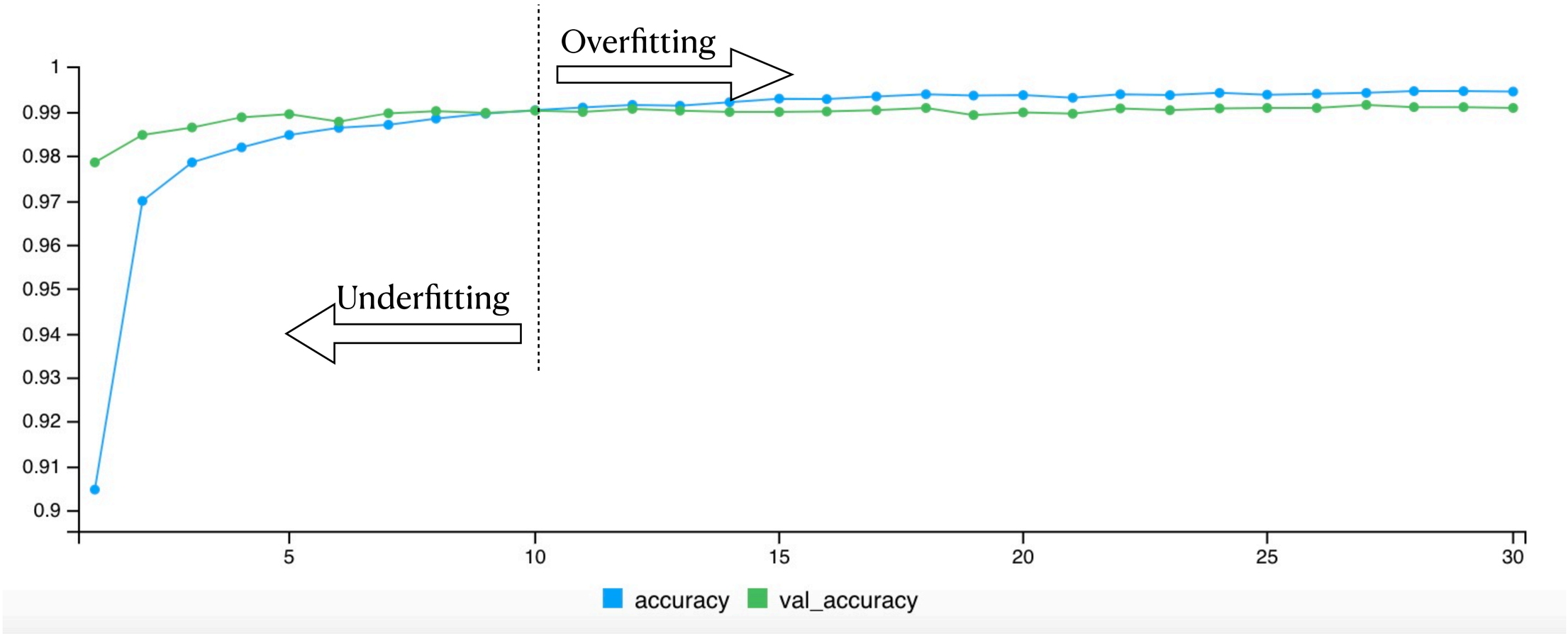
```
from tensorflow import keras
from tensorflow.keras import layers
inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(inputs)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dense(512,  activation = "relu")(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

Convolution, ReLu, and Maxpooling

- Running 30 epochs will take ~15 minutes

https://bit.ly/cnnsimu

A CNN Model for the MNIST Dataset



With the new model, the accuracy is **99.45%** on the training data and **99.08%** on the test data

CNN Operations

Convolution

Convolutional filter

1	0
0	1

Input feature map

0	1	1	1	0
0	-1	0	-1	1
0	0	-1	0	1
0	-1	0	-1	1
0	0	0	0	0

Output feature map

-1	1	0	2
0	-2	0	0
-1	0	-2	1
0	-1	0	-1

CNN Operations

ReLU

Input feature map

-1	1	0	2
0	-2	0	0
-1	0	-2	1
0	-1	0	-1

$$\text{ReLU} = \max\{x, 0\}$$

Output feature map

0	1	0	2
0	0	0	0
0	0	0	1
0	0	0	0

CNN Operations

Max Pool

Input feature map

0	1	0	2
0	0	0	0
0	0	0	1
0	0	0	0

Max pool with 2×2 filters and stride 2

Output feature map

1	2
0	1

The Transformer

Large Language Models (LLMs)

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

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

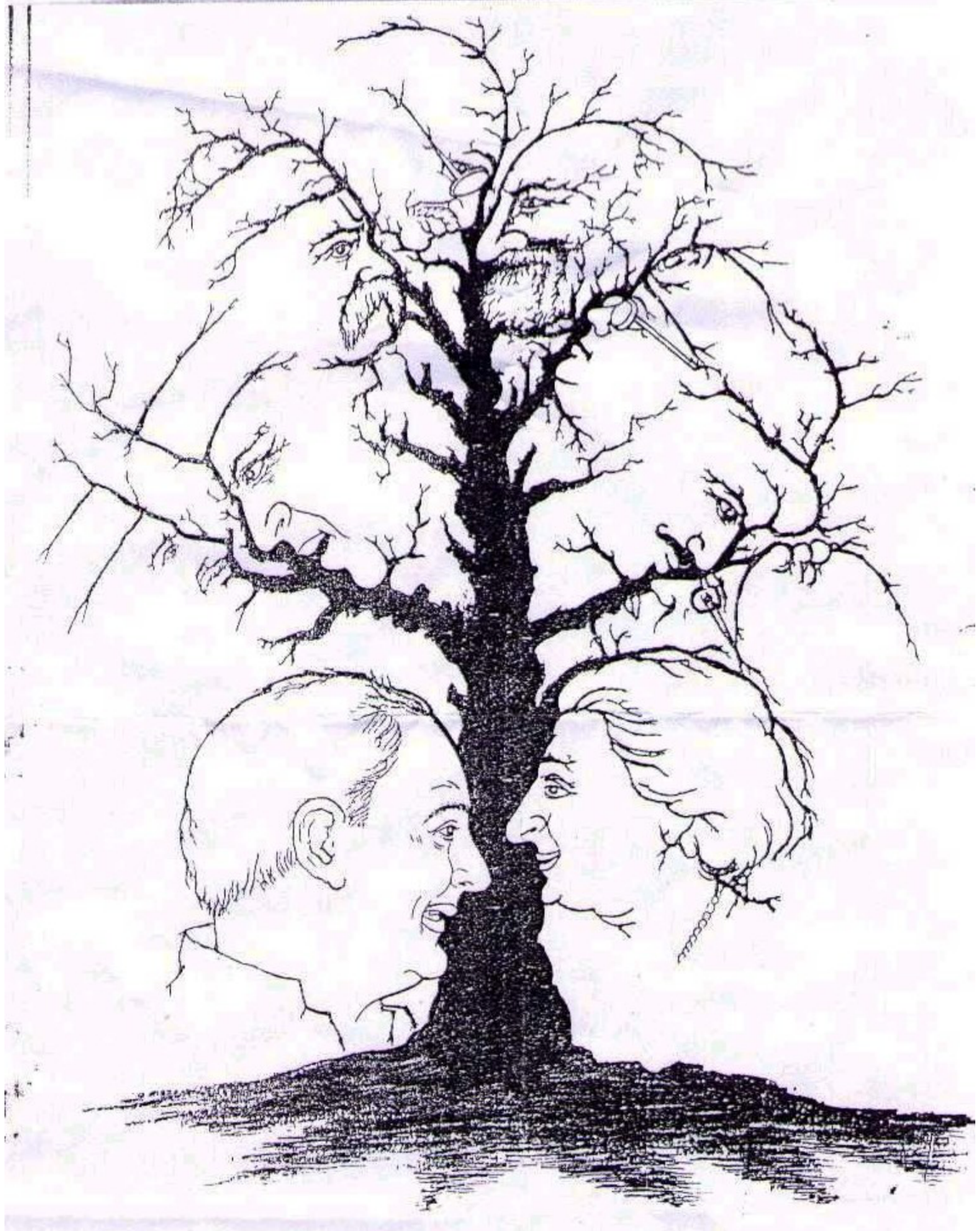
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

https://bit.ly/attention17



How many faces are in the picture?

"National Leaders Tree"

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

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

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

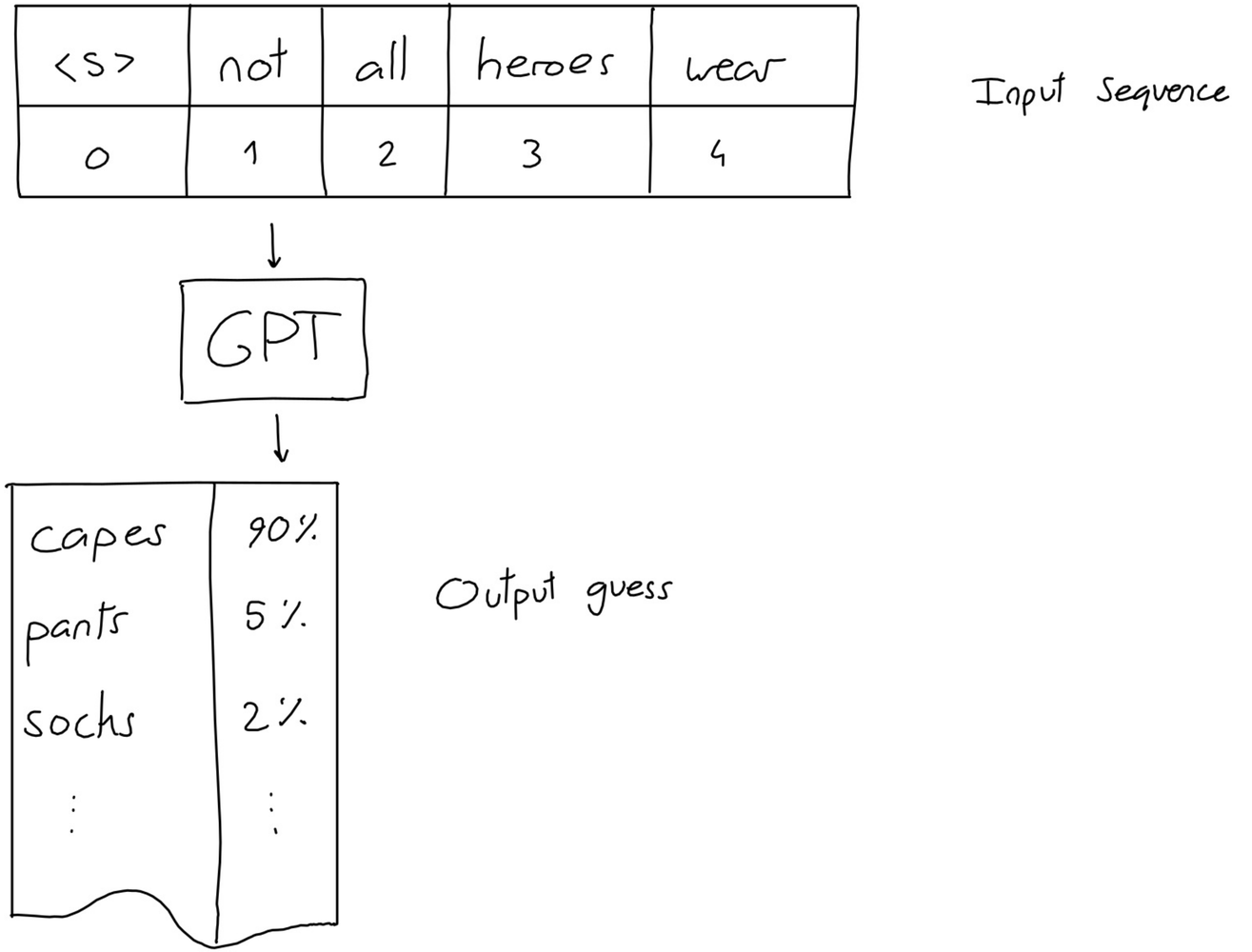
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

https://bit.ly/attention17

GPT = Generative Pre-trained Transformer

- **Generative Pretraining (“GP”)**, developed decades ago, seeks to train a language model unsupervised on a large corpus of text data
- The goal of GP: Generate text that closely resembles human-written text by **predicting the next word** in a given sequence
- The **Transformer (“T”)** architecture was developed in **2017**, in a paper written by Google employees (Ashish Vaswani et al.) called "Attention is All You Need"
- **OpenAI**, founded in 2015 by Sam Altman, Greg Brockman, Reid Hoffman, Jessica Livingston, Peter Thiel, and Elon Musk, was the first to combine “GP” and “T” to invent GPT in **2018**
- **ChatGPT (GPT-3.5)** was lunched on **November 30, 2022**

What GPT Does = Predicting the Next Word



https://bit.ly/gpt-napkin

What GPT Does = Predicting the Next Word

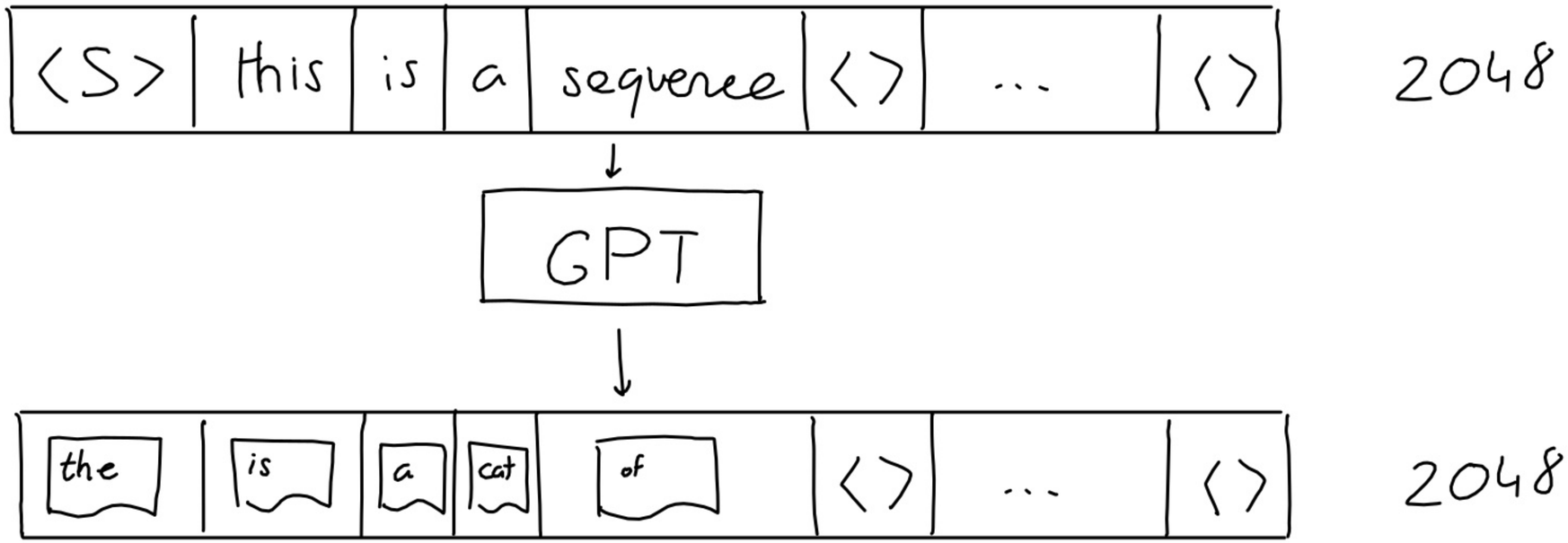
- Not all heroes wear → *capes*
- Not all heroes wear capes ...
- Not all heroes wear capes → *but*
- Not all heroes wear capes but ...
- Not all heroes wear capes but → all
- Not all heroes wear capes but all → *villains*
- Not all heroes wear capes but all villains ...
- Not all heroes wear capes but all villains → *do*

https://bit.ly/gpt-napkin

What GPT Does = Predicting the Next Word

In the case of GPT-3:

- **Input:** fixed to 2,048 words; for shorter inputs, extra positions are filled with empty values
- **Output:** a sequence (2,048 words) of guesses, one for each “next” position in the sequence



Predicting the Next Word = A Sequence Learning Problem

- Key challenge: The balance between long- and short-term memory
- Recurrent Neural Networks (RNNs) learn the texts **sequentially**, word by word, so the memory problem is severe
- The Transformer learns all the texts **simultaneously**, using the concept of **attention**

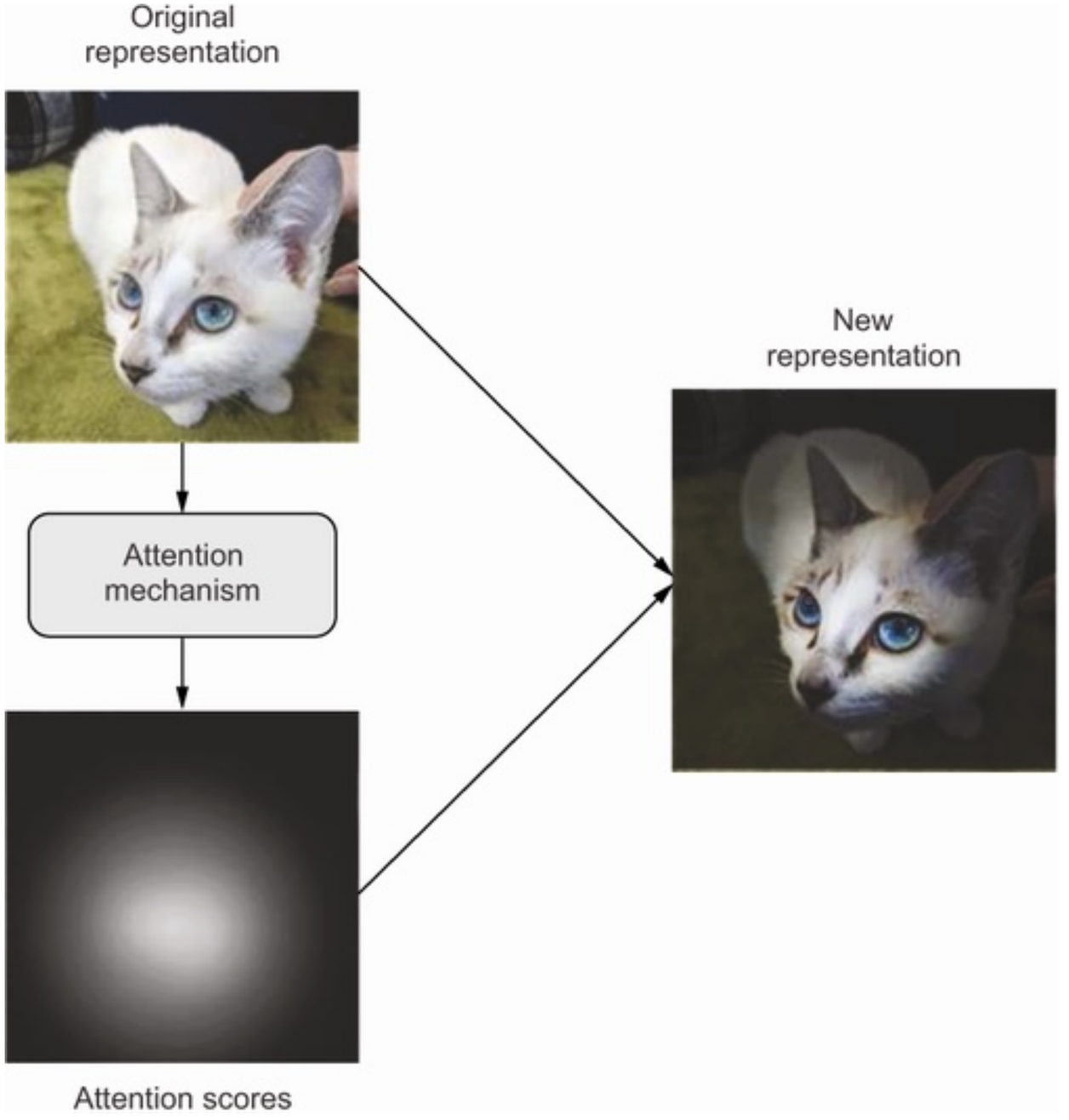
Attention

A mechanism in deep learning to focus on specific input parts

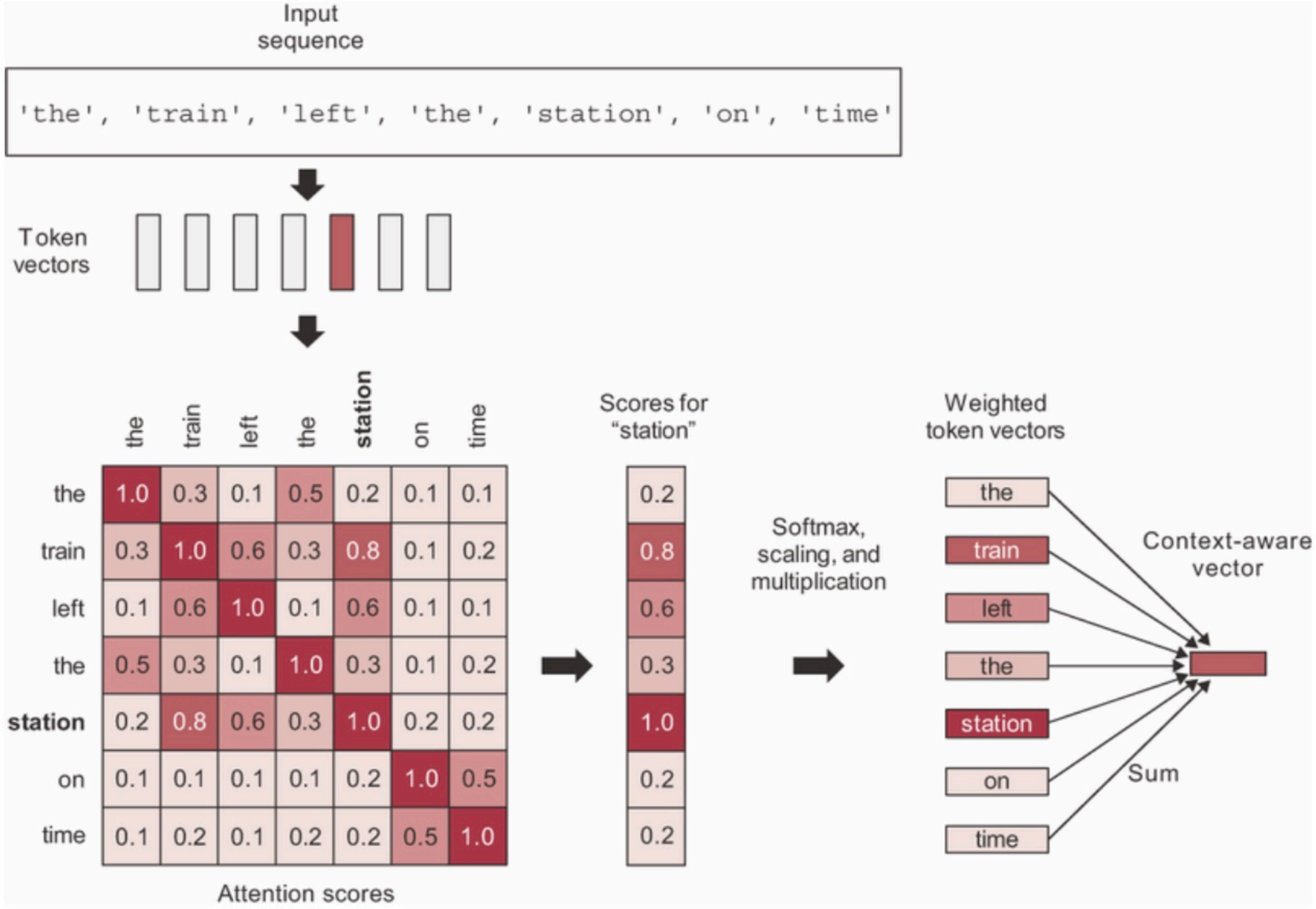
- Just like you skim through books, focusing on relevant parts, deep learning models can do the same

- Key Idea: Not all information is equally crucial. Models should focus on important features

- We've already seen a similar concept — MaxPooling in CNNs
  - Selects the most important feature in a region, ignores the rest
  - Essentially an "All or nothing" approach to attention

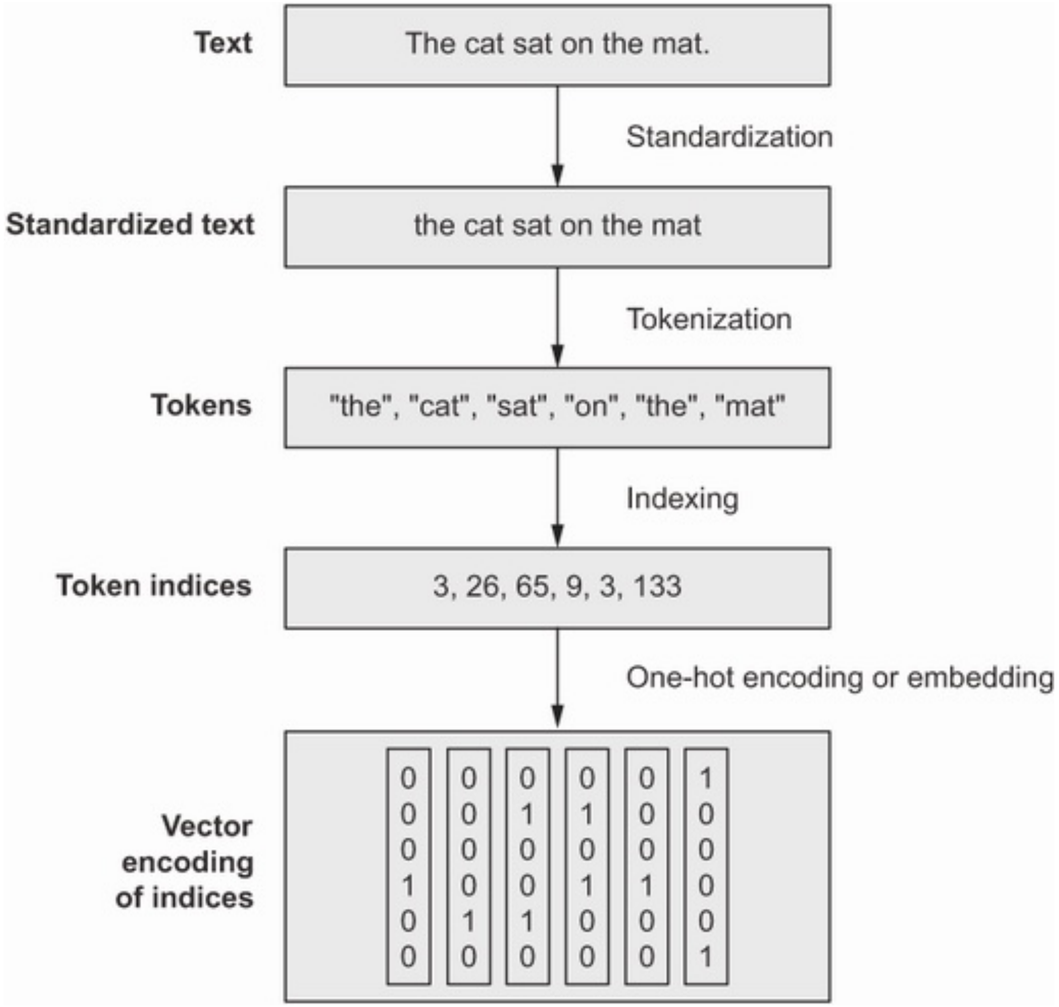

Original representation
Attention mechanism
Attention scores
New representation

Attention Is All You Need for Predicting the Next Word



Preparing the Text Data: *tokenization* and *vectorization*

Convert Text Data Into Numbers



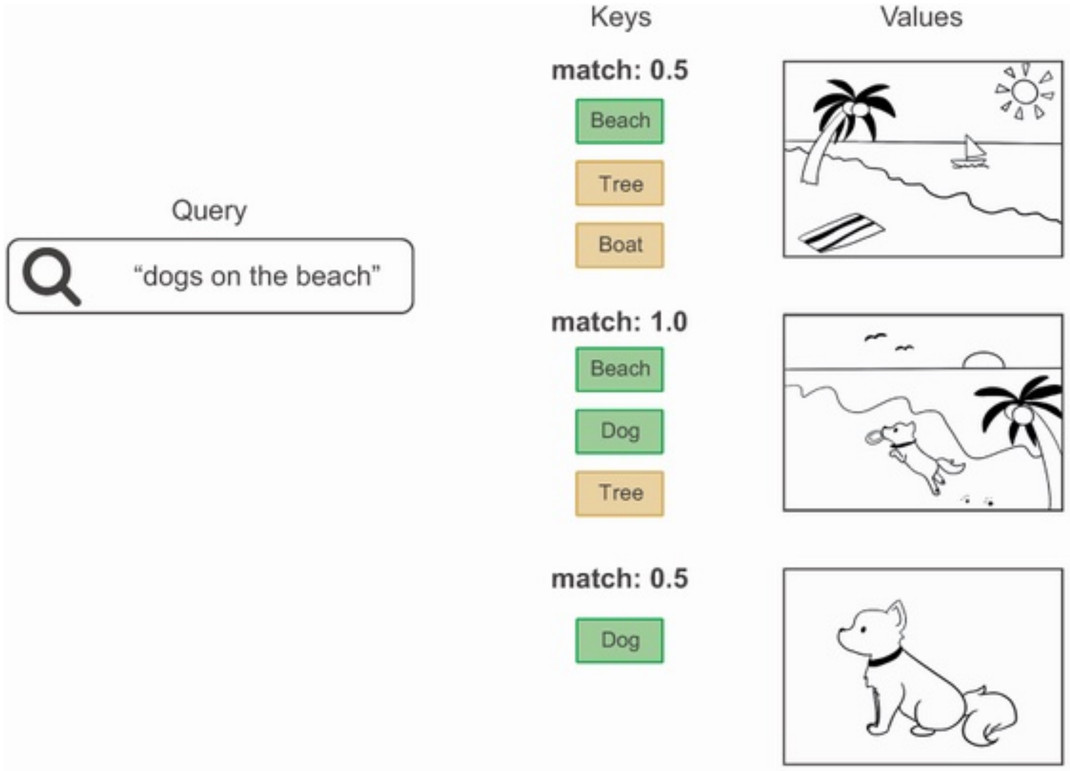
OpenAI’s tokenizer tool:

https://platform.openai.com/tokenizer

Rule of thumb: For typical English text, one token represents ~4 characters.
This equals about ¾ a word (100 tokens ≅ 75 words)

Where Do the Attention Scores Come From?

The Query-Key-Value Model



Retrieving images from a database: the "query" is compared to a set of "keys," and the match scores are used to rank "values" (images)

Where Do the Attention Scores Come From?

The Query-Key-Value Model

- Core Components
  - *Query*: What you're looking for
  - *Keys*: Descriptors for the values
  - *Values*: Information you want to extract
- Matching Process: Match the query to keys to find relevant values, and return a weighted sum of these values
- In Practice: query, keys, and values are all the same

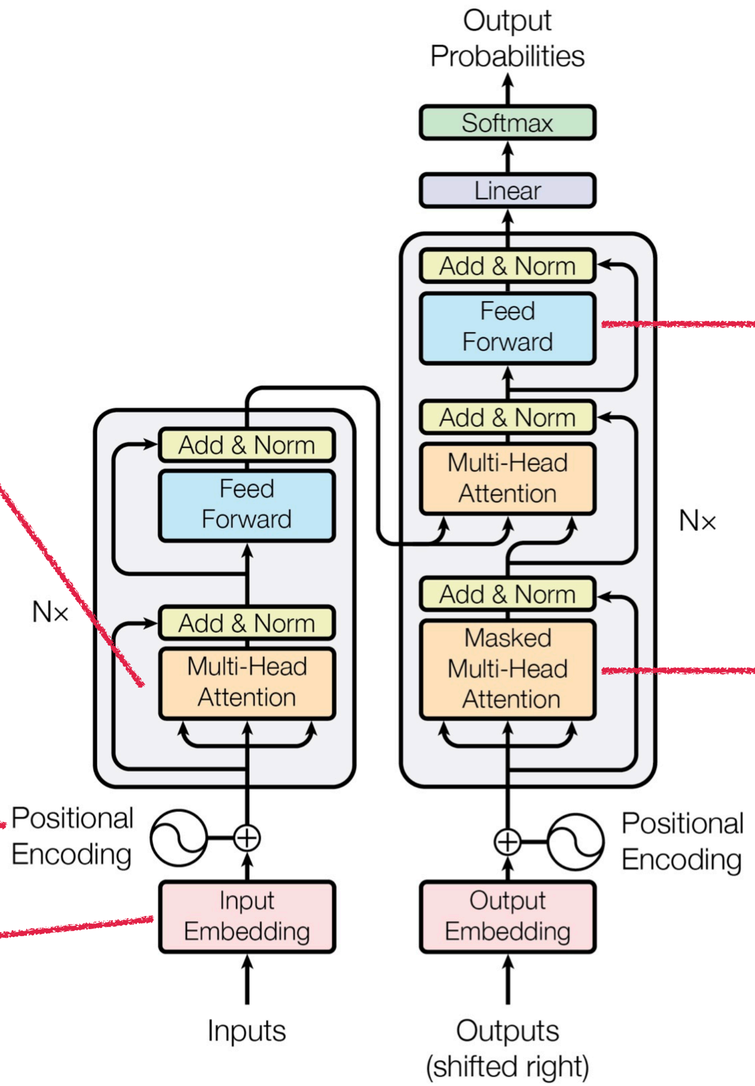
High-Level Structure of Transformer

Focuses on different parts of the input using attention scores to improve context

(Multi-head attention = multiple independent attention mechanisms)

Adds word order info to help the model process words simultaneously

Converts words into vectors that the computer can process



Standard neural network layers that further process information to refine predictions.

Masked attention stops the model from looking ahead at future words, ensuring it learns to predict one word at a time based on what it has already seen

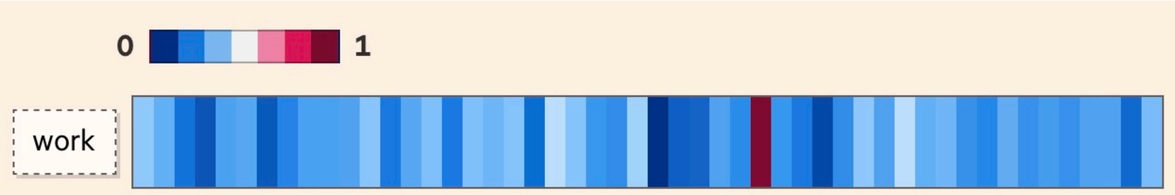
Transformer processes input sequences by focusing on different parts (using attention) and predicting output step by step (with masking)

The Training Process of a Large Language Model:

https://bit.ly/nytllm

If you haven't done so, make sure you activate your *New York Times* subscription via https://bit.ly/jhunyt

Word Embeddings



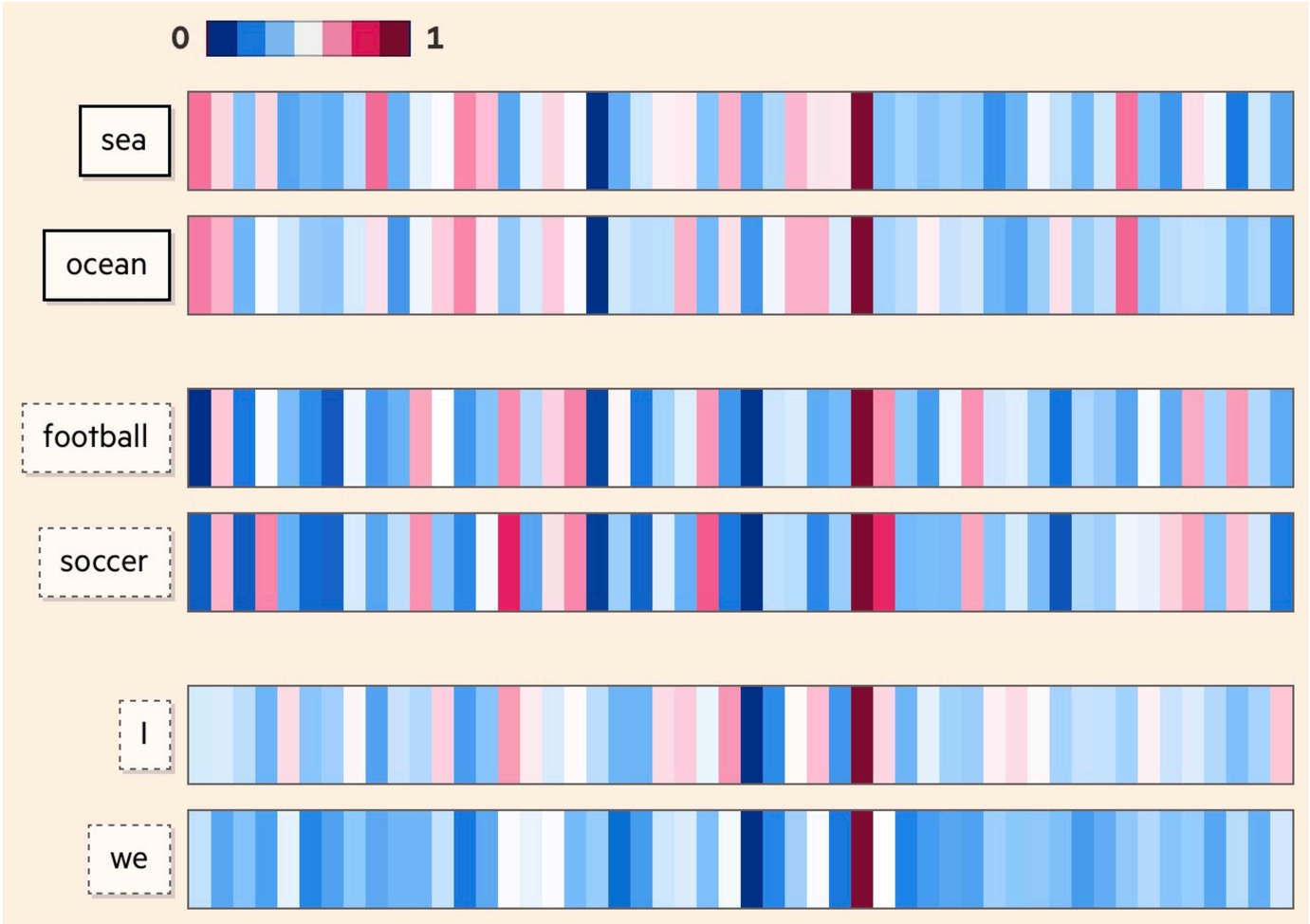
- A **word embedding** can contain hundreds of values, each representing a different aspect of a word's meaning
- Values in embedding quantify linguistic features of a word
- E.g., “work” can mean “job” or “writings”

https://ig.ft.com/generative-ai/

Embeddings Projector:

https://projector.tensorflow.org/

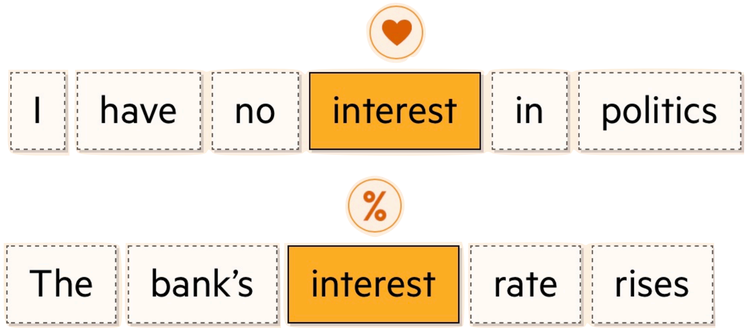
Similar Words Have Similar Embeddings



https://ig.ft.com/generative-ai/

Why Self-Attention is Useful

- Consider the following two sentences:

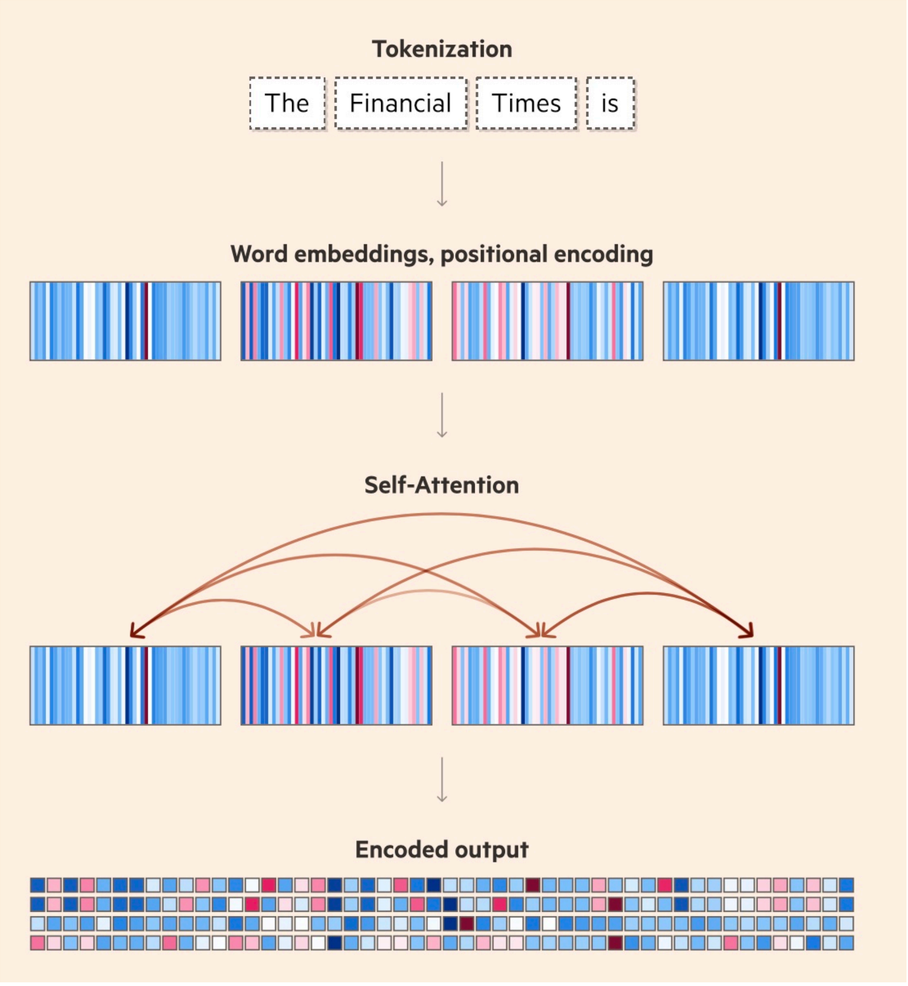


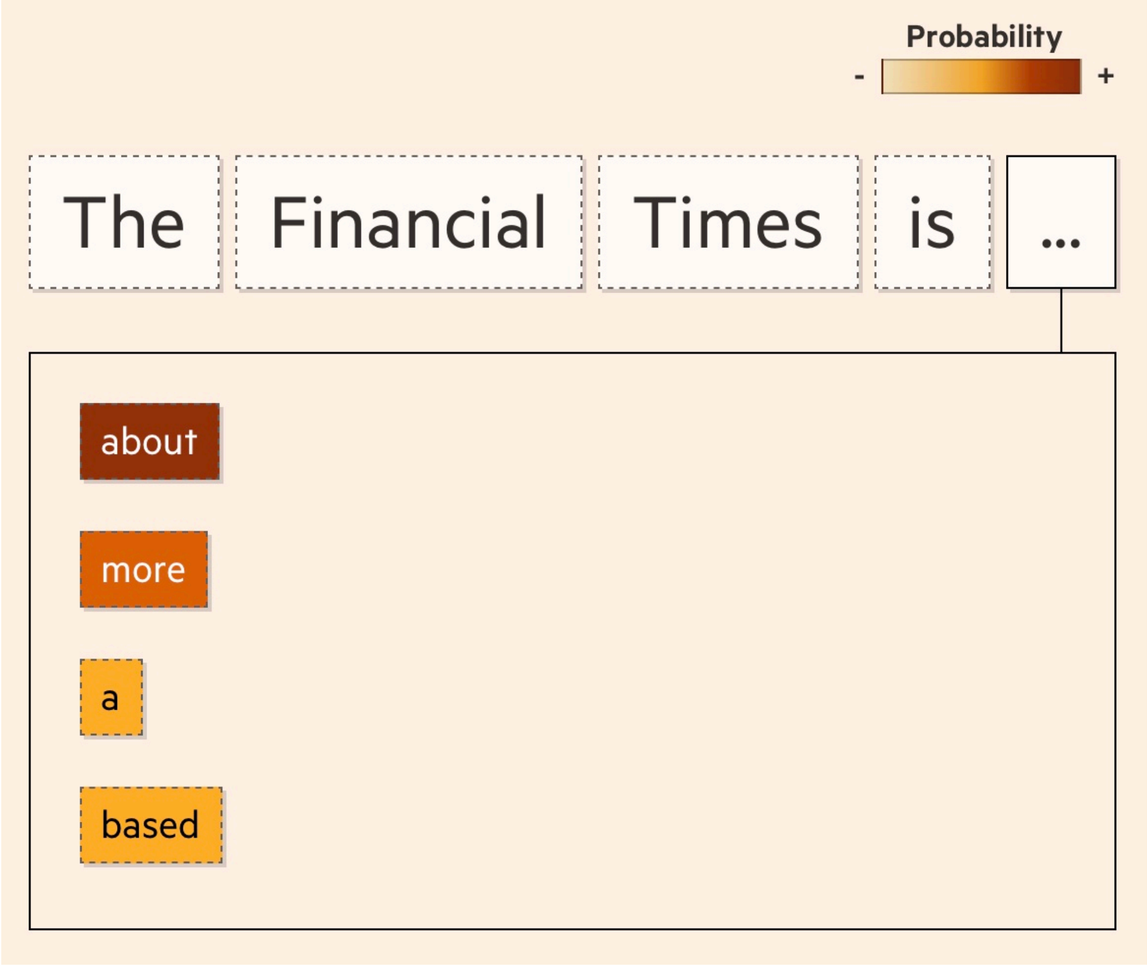
How do we figure out the meaning of "interest"?

- Pre-trained "self-attention" scores helps the model decide each word's meaning

https://ig.ft.com/generative-ai/

Applying Self-Attention





https://ig.ft.com/generative-ai/

A Conversation with Dr. Michael Abramoff

LLM API Programming

From ChatGPT User to Agent Builder

Why API Instead of Just ChatGPT?

ChatGPT UI

The ChatGPT web interface is optimized for **one-off, interactive conversations**: great for drafting, exploring ideas, and quick Q&A. It's your personal assistant for immediate needs.

While excellent for individual productivity, UI doesn't scale to organizational needs or integrate with your existing business systems.

LLM API

The **API** is optimized for **systems**: you can automate workflows, connect to databases, integrate with CRM/ERP, schedule jobs, and build products that run 24/7. Through the API, you can get larger context windows and direct file inputs, which enable tasks like full-report analysis, contract review, and multi-document synthesis that may exceed practical limits of the UI.

APIs also give you clear **versioning and cost control**: each call is logged and billable, which is essential for budgeting in a business context.

ChatGPT UI

Human has a conversation

Your App + API

Automated, repeatable systems

Customers & Employees

Scalable business value

Six Steps in Building AI Agents

01

Secure Your API Key

Authentication and environment setup

02

Install SDK & Make Text Calls

Your first "Hello, world" with LLMs

03

Analyze Images & Files

Multi-modal business applications

04

Extend with Tools

Web search, databases, custom APIs

05

Stream Responses

Real-time user experiences

06

Build Agents

Orchestrate complex workflows

Step 1 – Secure Your API Key

The API key is like a password for your application and must **never** be shared or hard-coded into public repositories. Think of it as the keys to your organization's AI infrastructure—losing control of this key could mean unauthorized usage and unexpected costs. The key should be stored as an environment variable, which the SDK will automatically read, keeping it separate from your codebase and version control systems.

On macOS and Linux systems, you can export the key directly in your terminal session. Similar can be done for Windows systems and for Colab notebooks.

```
export OPENAI_API_KEY="your_api_key_here"
```

Security Best Practice

Never commit API keys to Git repositories. Use environment variables, secret management services (like AWS Secrets Manager or Azure Key Vault), or configuration files that are explicitly excluded from version control via .gitignore.

Cost Management

Each API key can have usage limits and monitoring enabled through the OpenAI dashboard. Set up alerts to prevent unexpected charges and track usage patterns across different projects or teams.

Step 2 – Install SDK and Call the Model

To use the OpenAI API in Python, you can use the official OpenAI SDK for Python. Get started by installing the SDK using pip:

```
pip install openai
```

Once the SDK is installed, you create a client and send your first text request. This is the "Hello, world" of LLM APIs: one function call that turns a natural-language input into a natural-language output.

```
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input="Write a one-sentence bedtime story about a unicorn."
)

print(response.output_text)
```

1

Configure Client

Initialize with your API key from environment

2

Specify Model

Choose the right model for your use case

3

Provide Input

Send your natural language prompt

4

Read Response

Extract output_text from the result

Step 3 – Using Image Inputs

API can analyze images, combining text and visual information in a single request. This multi-modal capability opens up entirely new categories of business applications.

```
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input=[
        {
            "role": "user",
            "content": [
                {
                    "type": "input_text",
                    "text": "What teams are playing in this image?",
                },
                {
                    "type": "input_image",
                    "image_url": "https://upload.wikimedia.org/wikipedia/commons/3/3b/LeBron_James_Layup_%28Cleveland_vs_Brooklyn_2018%29.jpg"
                }
            ]
        }
    ]
)

print(response.output_text)
```

Step 4 – Extending the Model with Tools

The tools concept is where the API becomes truly powerful for business applications. Tools give the model "buttons it can press" to fetch data or trigger actions.

```
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    tools=[{"type": "web_search"}],
    input="What was a positive news story from today?"
)

print(response.output_text)
```

Web Search

Access current information, news, and real-time data beyond the model's training cutoff

Database Queries

Retrieve customer records, inventory data, or business metrics from your systems

Custom APIs

Connect to internal tools, CRM platforms, or third-party services for seamless workflows

File Search

Search through uploaded documents, knowledge bases, or document repositories

Step 5 – Streaming Responses

Streaming lets you show partial results as they are generated, dramatically improving user experience. This is critical for chatbots, dashboards, and interactive tools where users expect fast, incremental feedback rather than waiting for a complete response. Think about the difference between watching text appear word-by-word versus staring at a loading spinner for 30 seconds; streaming transforms the perceived responsiveness of your application.

```
from openai import OpenAI
client = OpenAI()

stream = client.responses.create(
    model="gpt-5",
    input=[
        {
            "role": "user",
            "content": "Say 'double bubble bath' ten times fast.",
        },
    ],
    stream=True,
)

for event in stream:
    print(event)
```

Without Streaming: User waits 20-30 seconds staring at a loading indicator, then sees the complete response all at once. Higher perceived latency and potential for user abandonment.

With Streaming: User sees text appearing within 1-2 seconds and can start reading immediately. Lower perceived latency and better engagement, even if total time is similar.

Step 6 – Building Agents That Take Action

Agents are higher-level constructs that can decide what to do next: which tools to call, which sub-agent to hand off to, and when to stop.

Agents are workflow engines powered by language models that can route work, draft responses, and interact with software on behalf of users. This represents a fundamental shift from simple request-response patterns to autonomous systems that can handle complex, multi-step business processes.

```
from agents import Agent, Runner
import asyncio

spanish_agent = Agent(
    name="Spanish agent",
    instructions="You only speak Spanish.",
)

english_agent = Agent(
    name="English agent",
    instructions="You only speak English",
)

triage_agent = Agent(
    name="Triage agent",
    instructions="Handoff to the appropriate agent based on the language of the request.",
    handoffs=[spanish_agent, english_agent],
)


async def main():
    result = await Runner.run(triage_agent, input="Hola, ¿cómo estás?")
    print(result.final_output)
```

The triage agent routes the request to the right language agent, demonstrating multi-agent orchestration. This pattern scales to far more complex scenarios: routing customer inquiries to specialized support agents, coordinating between research and writing agents for report generation, or managing approval chains that involve multiple stakeholders.

Taken Together:

https://bit.ly/llmapi25

API Pricing: Thinking in Tokens, Not Requests

Model choice is fundamentally a business decision: cheaper, faster models are sufficient for many routine tasks like classification, simple summarization, or data extraction; premium models are reserved for complex, high-value work like strategic analysis, creative writing, or nuanced decision-making. The key is matching model capability to task requirements, not defaulting to the most powerful (and expensive) option for everything.

Model	Input per 1M tokens	Output per 1M tokens
GPT-4.1	$2.00	$8.00
GPT-4.1 mini	$0.40	$1.60
GPT-5.1 mini	$0.25	$2.00

The API enables fine-grained cost management via logging, usage dashboards, and model selection. You can track costs by project, team, or use case; set spending limits to prevent overruns; and optimize by testing different models for specific tasks.

$0.002

Cost to Summarize

10-page report with GPT-4.1 mini

30min

Time Saved

Analyst time freed for higher-value work

5000x

ROI Potential

When comparing AI cost to human labor

AI-Driven Business Models

The Deep Learning Workflow

Step 1. Define the problem: What data is available and what you are trying to predict? Will you need to collect more data or hire people to manually label a dataset?

Step 2. Identify your measure of success, including prediction accuracy and domain-specific metrics

Step 3. Prepare the validation process that you'll use to evaluate your models (i.e., defining a training set, a validation set, and a test set)

Step 4. Turn data into vectors that are easily approachable by neural networks

Step 5. Develop a first model that beats a trivial common-sense benchmark

Step 6. Refine your model by tuning hyper-parameters and adding regularization/dropout

Limitations of Deep Learning

- Many applications are completely out of reach for current deep learning techniques, even with vast amounts of human-annotated data

- Anything that requires **reasoning**—e.g., complex mathematical/programming tasks or applying the scientific method—long term planning and algorithmic data manipulation is out of reach for deep-learning models, no matter how much data you throw at them

- Reason: Deep learning is just a chain of simple, continuous geometric transformations mapping one vector space into another

McKinsey Analytics

Most of AI's business uses will be in two areas

An examination of more than 400 AI use cases revealed the two areas where AI can have the greatest impact.

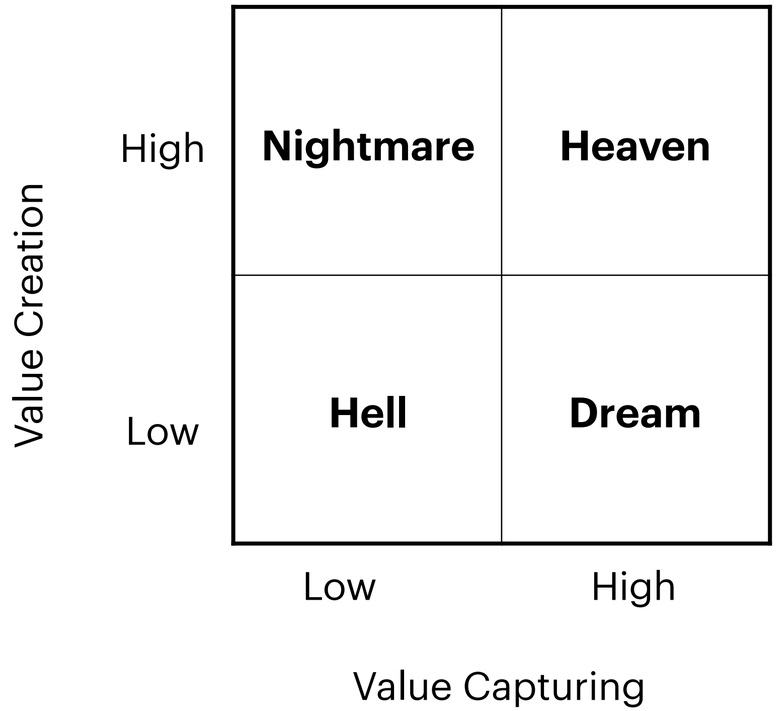
by Michael Chui, Nicolaus Henke, and Mehdi Miremadi

- "40% of all the potential value that can be created by analytics today comes from the AI techniques that fall under the umbrella "**deep learning**," which could account for between $3.5 trillion and $5.8 trillion in annual value."

- "While applications of AI cover a full range of functional areas, it is in fact in these two cross-cutting ones—supply-chain management/manufacturing and marketing and sales—where we believe AI can have the biggest impact."

What Does Business Model Mean?

- Business model = the design of value creation and value capture mechanisms
  - Value creation = perceived (or received) benefits to the customer
  - Value capture = pricing and cost structures used to generate revenue and profits



Brandenburger and Stuart (1996)

Customer- vs. Employee-Facing AI Solutions

- For customer-facing AI solutions, value is usually related to **engagement**: Number of active users, time spent, money saved, better health etc.
  - Focus: Using AI to create more engaging applications

- For employee-facing AI solutions, value is usually related to **productivity**: Time saved, tasks performed, escalation rate, enhanced safety, etc.
  - Focus: Using AI to create more productive applications

- **Small-group discussions**: Use the value creation-value capture framework to discuss your AI Lab project

Using AI to Create Value in Manufacturing

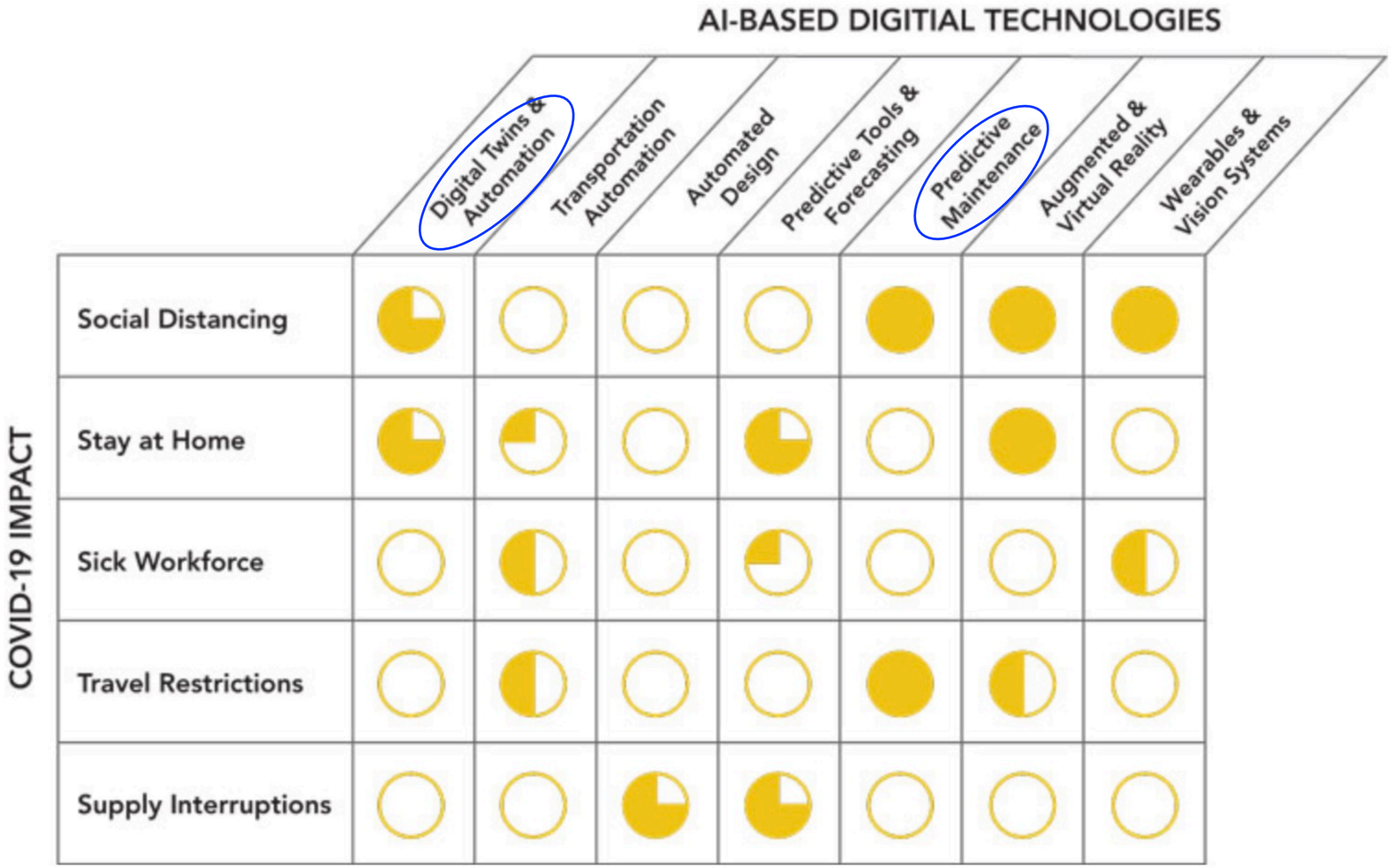
Lessons learned from hurricane recovery can improve supply chains

BY SHELDON H. JACOBSON AND TINGLONG DAI 09/27/21 03:00 PM EDT

A shortage of workers performing preventive checkups at factories has led to a dramatic increase in fire accidents and other disruptions. With the COVID-19 vaccination rates highly uneven across nations, areas experiencing the most challenging infection surges are also experiencing challenged supply chain links.

https://bit.ly/thehillsc

Using AI to Create Value in Manufacturing



Thorsten Wuest, Andrew Kusiak, Tinglong Dai, Sridhar Tayur. 2020. Impact of COVID-19: The Case for AI-Inspired Digital Transformation. *OR/MS Today*. 47 (3) 34–39.

Using AI to Create Value in Manufacturing

Predictive maintenance: Analyze the data generated by sensors in the equipment to predict when maintenance should be performed, instead of sticking to time-based maintain schedules

Classification

Regression

Quality/safety assurance: Use perception techniques (e.g., CNNs) to identify defective products and safety violations in the manufacturing line

Vision

Classification

Demand forecasting: Anticipating demand increases or decreases based on external factors to optimize production

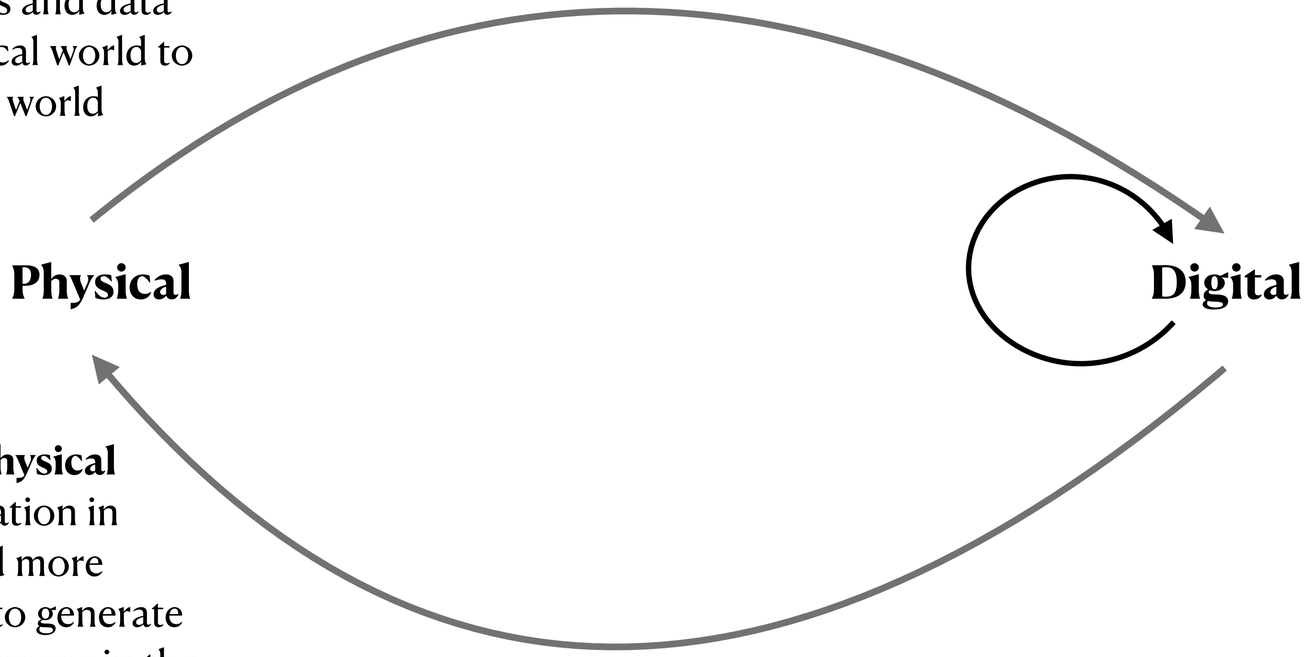
Regression

Digital twins: Create a virtual model of a production, process, or even an entire factory to monitor and analyze the system

Regression

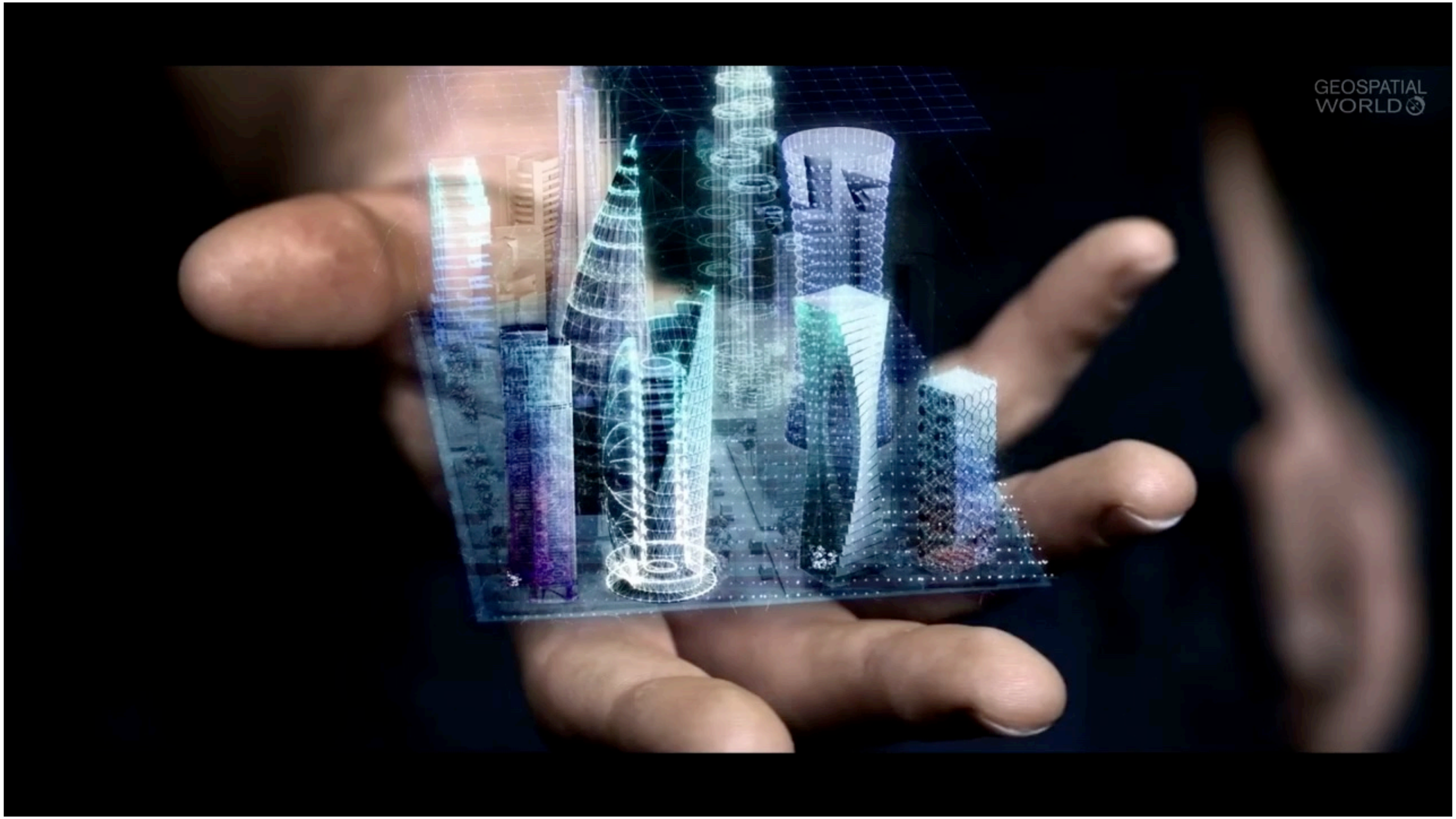
Digital Twins Create a Physical-Digital-Physical Loop

1. Physical to digital
Capture signals and data from the physical world to create a digital world



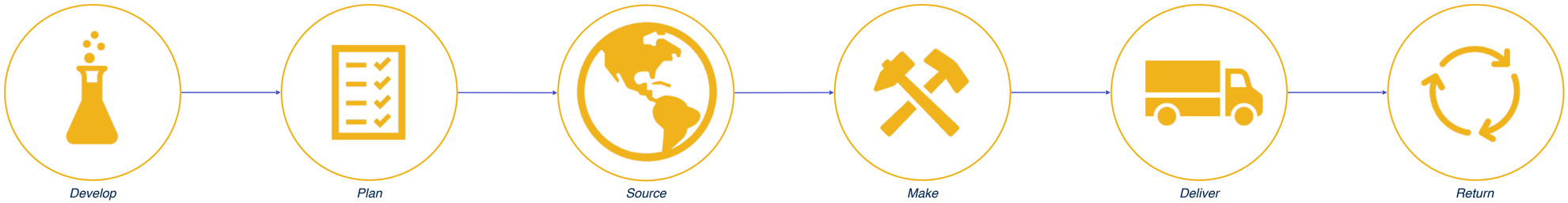
2. Digital to digital
Exchange and enrich information using AI to drive meaningful insights

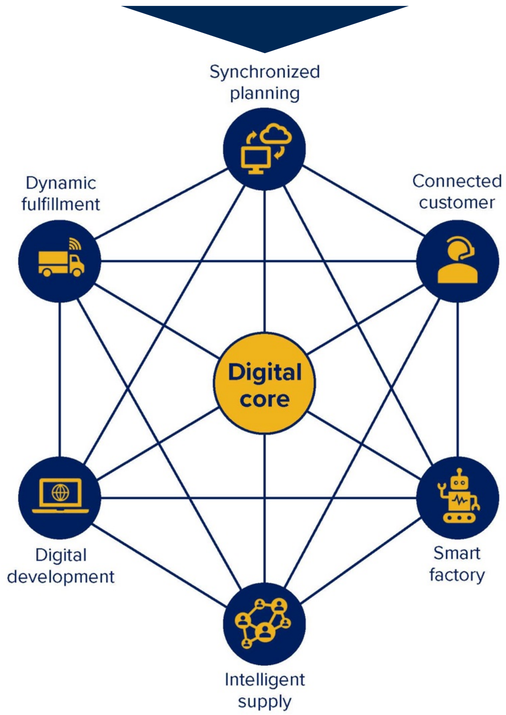
3. Digital to physical
Deliver information in automated and more effective ways to generate actions and changes in the physical world



https://bit.ly/ytdigitaltwin

Using AI to Transform Supply Chains

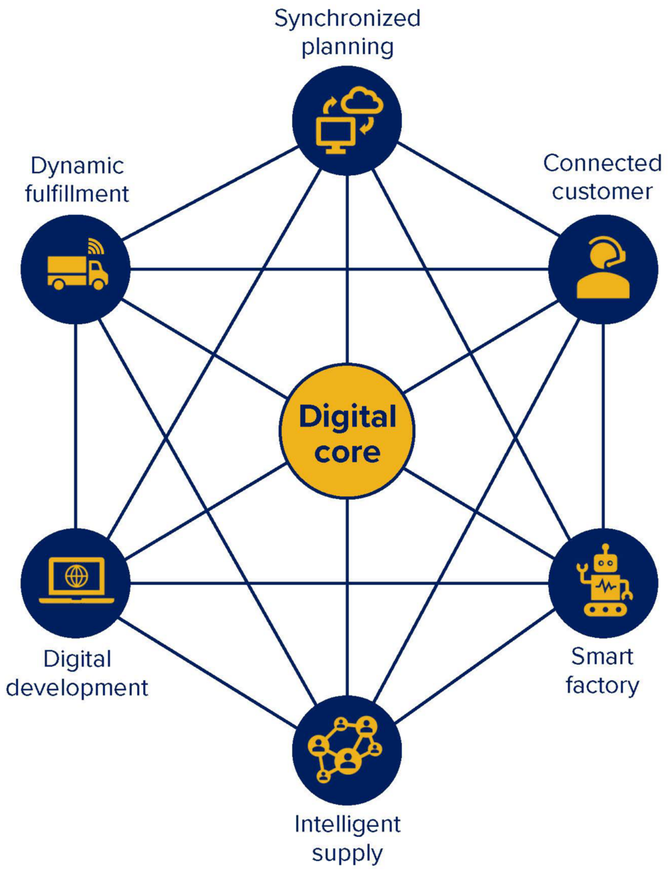




Sinha et al. (2020)

Using AI to Transform Supply Chains

- Focused on customer-fulfilling requirements and demand
- Digital core at the center to connect nodes
- Critically assessing status quo and developing a strategic vision and implementation roadmap



Sinha et al. (2020)

Using AI to Create Value in Retail

Product recommendation: Automatically “push” products based on previous purchases, viewing history, and other customer signal	Recommendation
Personalization: Customize the online experience and/or outbound marketing content based on customers’ behaviors and performances	Recommendation
Product discovery: Assist shoppers with intelligent tools such as visual search, virtual dressing rooms, or conversational AI to help them narrow down their selection	Vision
Dynamic pricing: Adjust prices and personalize discounts in real time based on factors such as competitor pricing, demand estimation, user shopping habits, and external factors (e.g., COVID restrictions)	Optimization

Using AI to Create Value in Financial Services

"Just in time" lending: Apply AI techniques to accelerate the risk assessment process for credit applications	Classification
Dynamic pricing: Apply AI to broader data sets to provide more accurate pricing and optimize underwriting (e.g., drivers' sensor data)	Optimization
Fraud detection: Use AI to spot unusual patterns by analyzing multiple data points to identify fraudulent transactions that rule-based analysis may miss	Classification
Trading and investment management: Optimize trade execution strategies and mergers/acquisitions analysis, augmenting agents and analysts with intelligent insights across multiple data sources	Optimization

AI Lab Project:

Feedback is on Canvas

Please reach out to Suhas for one-on-one meetings


We are Doomed...


BostonDynamics

Until Next Class

- No assignment due — enjoying working on your **AI Lab project**
  - Final report due by **December 11**

- Check Canvas for Readings for Session 6

- Happy Thanksgiving!


LOVE IS ALL YOU NEED


Thank You!