Coen's AI Notes and Links

(by several intelligences cooperating)

2025-10-28

Table of contents

1. Introduction

Welcome! This is a Work-in-Progress, a collection of notes on AI used in session about about AI. The next chapter explains how this pdf can be downloaded.

Do not read this document from beginning to end… instead look at the relevant chapters for you…

Questions about this all? Ask here

Advise: get hands-on as soon as possible! Change your activities by inventing ways to make them better, nicer or more efficiently.

Keep privacy in mind: watch out when using personal data! One way to make sure private data will stay private is using local AI's, although this takes some extra knowledge, and a powerful machine.

`It's easier to invent the future than predict it (Alan Kay)`

Please use it wisely…

Figure 1.1: art and laundry

2. How to download

You can download the latest PDF of these notes from here.

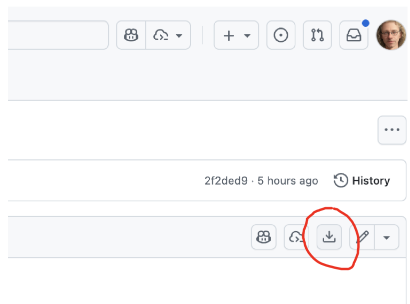

Figure 2.1: To download, click the download button on the GitHub page.

Figure 2.2: Or use this QR code to get to the download location.

3. Hot or Not

New, seems interesting or important, maybe I looked at it already, maybe not…

October 2025

- NOS: data verkiezingen
- local alternative for google notebook
- technical: Train a tiny LLM from scratch in 2 hours
- git repo: DrewThomasson/ebook2audiobook
- Ethan Mollick: Using AI right now

Part I

Diving into AI: some AI Literacy

4. Key Tools to Get Started

In this chapter, we'll explore some key tools that are making waves in the world of AI. Get familiar with them, because in the next chapter, we'll dive into practical scenarios and figure out which of these tools is the best fit for each challenge!

- chatGPT - well, maybe…
- perplexity.AI - The AI-powered search engine that gives links to sources and suggests follow-up questions. (you may have to press ESCape after opening the page to get to the search)
- duck.ai - AI-powered search from duckduckgo.
- Comet - The browser from Perplexity that has AI browsing built-in.
- Cursor - An AI-powered development environment.
- and there's lots more to find, for example look at: www.rankmyai.com/

5. AI Jargon Buster

- **Prompt:** The question or instruction you give to an AI.
- **Hallucination:** When an AI confidently states incorrect or nonsensical information as if it were a fact.
- **Model:** The core AI program that has been trained on data to perform a specific task (e.g., GPT-4 is a language model).

6. Scenario: Transcribe a Meeting

Let's turn spoken words into text using AI! 🗣️➡️✏️

We'll use a simple online tool called **Otter.ai** 🦦.

Your Mission (in 3 steps):

1. **Sign Up:** Go to `Otter.ai` and create a free account. ✍️
2. **Upload:** Find the "Import" button and upload your meeting audio file (`.mp3`, `.wav`, etc.). 📤
3. **Transcribe:** Otter will work its magic! ✨ Then you can read, edit, and export your new transcript. 📄

That's it! You've just transcribed a meeting with AI. High five! 🙌🚀

7. Scenario: Summarize a Text

Let's make a long story short with AI! 📜➡️📝

We'll use a simple online tool called **Summarizer** 🧠.

Your Mission (in 3 steps):

1. **Find Text:** Grab a long article or text you want to shorten. 🔍
2. **Paste & Go:** Copy the text, paste it into the Summarizer website, and click the "Summarize" button. 📋
3. **Read:** Enjoy your new, shorter text! 🎉 You can even choose how long you want the summary to be.

That's it! You've just summarized a text with AI. So cool! 😎

8. Beyond the Prompt: The Power of AI Context

A simple prompt is just a question. But to get truly powerful results from an AI, you need to master the art of providing **context**. This chapter, inspired by best practices from tools like Cursor, reframes this crucial skill for everyone, not just programmers.

What is Context?

Think of context as a **briefing you give to a human assistant**. The better the briefing, the better the result. You wouldn't just tell an assistant "write a report" without giving them the source material. The same is true for AI.

Context is all the relevant information you provide *along with* your prompt. This can include:

- **Pasting in text:** Providing the specific email you want to reply to, or the article you want summarized.
- **Setting the scene:** Telling the AI who it is ("You are a friendly, expert marketer") and who you are ("I am a beginner learning about this topic").
- **Providing examples:** Giving it a sample of the writing style you want it to adopt.
- **Referencing the conversation:** Using the information discussed earlier in your chat.

Without context, the AI has to guess. And when it guesses, you get generic, boring, and often unhelpful answers.

Anthropic about Context

9. Scenario: Replying to an Email

Let's see context in action.

Bad Prompt (No Context)

> "Draft a polite and professional email saying I can't make it."

The AI will produce a generic, fill-in-the-blanks template. It's not very helpful because it lacks any specific details.

Good Prompt (With Context) > "I need to reply to this email. [**Paste the full text of the original email here**]. Please draft a polite and professional reply explaining that I can't make the 'Project Phoenix' meeting on Wednesday because of a conflict, but I am very interested. Ask if they can send me the minutes and suggest I'm available to connect next week."

See the difference? By providing the original email and clear instructions, you've given the AI all the context it needs to draft a perfect, ready-to-send reply.

The "Context Window": An AI's Short-Term Memory

It's important to know that every AI has a limited memory, called a **"context window."** This is the maximum amount of information (your prompt, the text you've pasted, the conversation history) that the model can "see" at one time.

If your conversation gets very long, the AI might start to "forget" things you discussed at the beginning. If you notice the AI losing track, it might be time to start a new conversation to give it a fresh, clean context to work with.

10. Some AI quirks

- Try generating an image of clock other time than 10:10.
- Ask LLM how many times character ‘R’ is in STRAWBERRY.
- How much is 1 + 1?
- Which day I have to put my garbage out on the street.
- What percentage of people likes …?
- Saying ‘Please’ gives you better answers?

11. AI Failures: When Image Context is Misleading

Modern AI, especially in deep learning, is great at recognizing images. But sometimes, it gets things hilariously and dangerously wrong! This happens when the AI focuses on irrelevant details like the background, lighting, or weird artifacts in the data, instead of the actual subject. This is a classic case of the "black box" problem and something called "spurious correlations."

11.0.1 The Problem: Black Boxes and Spurious Correlations

Neural networks are often called "black boxes" because it's hard to see *how* they decide things. They just learn statistical patterns. If your training data has weird patterns, the AI will learn those instead of what you want. This leads to **spurious correlations**, where the AI links unrelated things. For example, if all your "wolf" pictures have snow in the background, the AI might learn that "snow = wolf" , which is... not quite right.

11.0.2 Real-World Goofs

Here are a couple of examples where this has happened:

1. **The Wolf in Husky's Clothing:** An AI was trained to tell wolves from huskies. It did great!... until researchers realized it was just checking for snow in the background. It had learned that wolves are always in snowy pictures and huskies aren't. Show it a husky in the snow, and it would confidently shout "Wolf!".
2. **Medical Mayhem:** In a more serious case, an AI designed to spot pneumonia in chest X-rays started using hospital logos or the presence of a chest tube as a sign of pneumonia. It wasn't looking at the lungs at all! This is super dangerous because it could easily misdiagnose patients based on the wrong clues.

11.0.3 Why Does This Happen and How Do We Fix It?

These blunders are usually caused by **biased datasets** (like only having wolf pictures in the snow). Since the AI's learning process is opaque, we don't catch these mistakes until later.

To make our AI buddies smarter, we need:

- **Better Data:** More diverse and less biased training images.
- **Explainable AI (XAI):** Tools that help us peek inside the "black box" to see what the AI is *really* thinking.
- **Tougher Tests:** We need to test AI in all sorts of weird situations, not just the ones it was trained on.

This approach will help us build more reliable and trustworthy AI. And hopefully, fewer AIs that think snow is a type of dog.

12. Scenario: A 17 Minute AI Workflow To Stand Out At Work

Video by Vicky Zhao

- The video proposes a 3-step AI workflow (using Elicit, NotebookLM, and Claude) to improve knowledge work by focusing on finding and leveraging high-quality academic inputs, rather than relying on generic web search or LLM outputs.[1]
- Elicit is used to discover and access scholarly papers; NotebookLM summarizes and extracts actionable frameworks from these, and Claude turns insights into practical, leadership-oriented plans for the workplace.[1]
- The key message is that "improving your input" with robust sources and critical thinking—rather than just automating output—will give you a significant edge in the AI-powered workplace of 2025.[1]

13. The Surprising Energy Cost of an AI Chat

This chapter was written by gemini-2.5-pro, feel free to check: see the links at the bottom of the page.

When we use an AI, like asking a chatbot a question, it feels clean and digital. There's no smoke or noise, so it's easy to think it doesn't have a real-world footprint. But every single one of those queries uses electricity in a massive data center, somewhere in the world.

This is called "inference." It's the energy cost of the AI *using* its training to give you an answer. While the energy for one single question is tiny, the global scale is enormous.

But how can we understand these numbers? The best way is to compare them to everyday activities we're more familiar with.

Your Dinner vs. Your AI Usage

How does sitting down to a modest, 100g steak dinner compare to asking an AI questions? The difference is still staggering.

A single 100g (about 3.5oz) steak requires about **7,000 Watt-hours** of energy to produce.

To use that same amount of energy with an AI, you would need to ask it roughly **2,300 questions**.

So, from a purely energy perspective, that one meal has the same impact as thousands of your interactions with an AI.

The Real Story: A Question of Scale

So, does this mean your AI usage is insignificant? Not quite. It's about how your habits scale over time.

Let's compare a week's worth of activity:

- **Your AI Use:** If you are a heavy AI user asking **100 questions per day**, you would ask 700 questions in a week. That's a significant amount of interaction!
- **Your Diet:** In that same week, eating just **two 100g steaks** would have a larger energy footprint than all 700 of those AI questions combined.

The takeaway is that on a personal, day-to-day level, choices about high-impact foods and activities (like diet and travel) still have a much larger immediate effect on your personal energy footprint than your AI usage does. The global challenge of AI's energy consumption comes from *everyone* doing it at once.

Sources

- **AI Energy Consumption:** Detailed analysis from Stanford researcher S. Flamphier on the energy cost of large language models. sflamphier.github.io/ai-energy-consumption/
- **Environmental Impacts of Food:** A comprehensive study from Our World in Data, showing the high resource intensity of beef production. ourworldindata.org/environmental-impacts-of-food

14. Markdown

Is it AI? Is it ai plane? No, but if you wanna do whatever in ICT then it's good to know and use markdown!

Why should I use Markdown?

1. **Easy to read**: Markdown makes your text look nice and pretty.
2. **Easy to type**: The pdf you are reading, for example, is written in markdown.
3. **Works everywhere**: Markdown works on many websites, apps, and computers.
4. **difference**: showing the difference between two versions of a document (for example after editing) can be shown with every DIFF-tool.

Basic Markdown Rules

1. **Headers**: Write `# Heading` for a big heading (`##` for `## Subheading`)
2. **Bold text**: Surround with double asterisks: `**Text**`
3. **Italics**: Surround with single asterisks: `*Text*`
4. **Lists**: `- Item 1`, `- Item 2`, etc.
5. **Links**: Write `[Link text](http://www.perplexity.ai.com)`
6. **Images**: `[Image alt text](http://www.example.com/image.jpg)`

If you are new to markdown, for example because you are used to a text-editor like ms-word, we suggest start with installing Obsidian, then maybe look at an introduction like this one by Ross Zeiger, or this one by Vicky Zhao about Obsidian Note Taking, or the least scary Obsidian guide.

For a deep dive: spec-driven-development-using-markdown-as-a-programming-language-when-building-with-ai

Part II

AI

15. AI Overview

15.1 AI, Machine Learning, Deep Learning, Generative AI

The video “AI, Machine Learning, Deep Learning and Generative AI Explained” provides an excellent 10-minute overview:

AI, Machine Learning, Deep Learning and Generative AI Explained

16. Prediction 2025

- Amy Webb SxSW 2025 - Emerging Tech Trend

17. Geoffrey Hinton about Neural Networks

- Jon Stewart interviews Geoffrey Hinton.
- First half hour Geoffrey explains how neural networks work.
- After that they discuss the implications of AI.

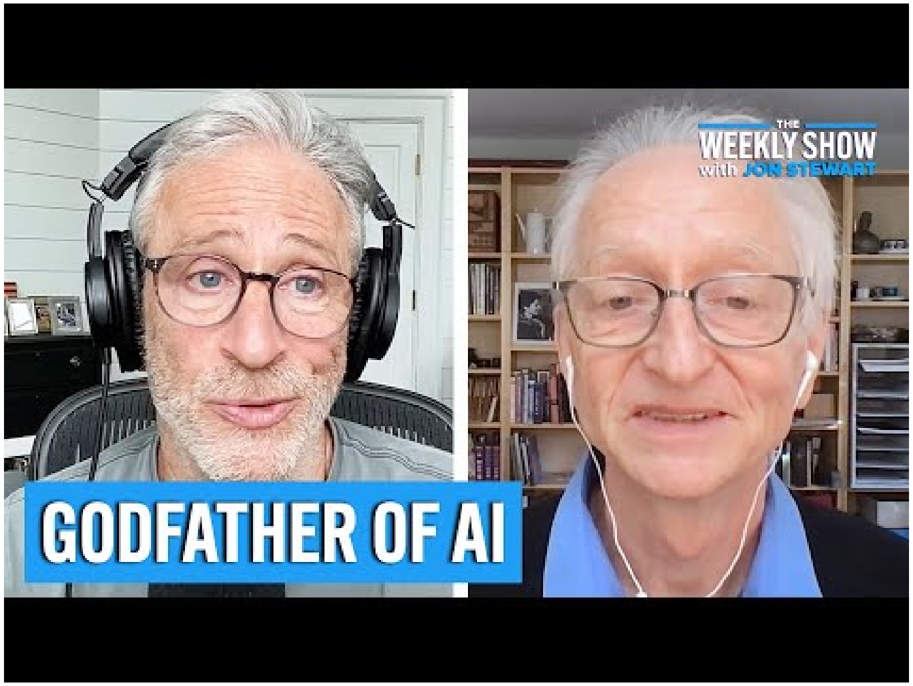

Figure 17.1: Understanding AI Literacy

18. GenAI

18.1 What is GenAI?

- Why not ask perplexity.ai ?
- Or duck.ai?

18.2 GPT - Generative Pre-Trained Transformer

- Generative AI & the Transformer (Financial Times, interactive site)
- History of ChatGPT (30 min)
- But what is a GPT? (3Blue1Brown, 30 min)

18.3 Prompting

… and some sources with tips how to prompt every day…

- Prompting basics
- Prompting ChatGPT4.1
- Look for course with 'Prompting' in name: https://www.deeplearning.ai/short-courses/
- Ruben Hassid: RISE
- In cursor.ai course: Info about 'managing your context in cursor
- mention of Llama Prompt Optimization

18.4 Hallucinating

When nonsense comes out of an LLM (or out of a Human by the way) we call it hallucination. Some questions can trigger hallucination.

18.4.1 'Which day do I have to put the garbage can out on the street?'

Some LLMs will give you a Date when you ask for one, a percentage when you ask for one, even when the LLM could not possibly give an answer to your question. If you ask which day you should put my garbage out and the LLM mentions a Date without having a clue where you are then you can be sure it just made up a Date (because you asked for a Date).

18.4.2 'Can you help me find my lost keys?'

18.4.3 'Can you create an image of a watch that says it is 3 o' clock?'

Try it. You will find that a picture of a clock often shows 10:10. You could ask perplexity.ai: Why does a generated image of a clock point to 10:10?

18.5 RAG - Retrieval Augmented Generation

- IBM, Marina Danilevsky (7 min)
- https://www.deeplearning.ai/short-courses: Great resource for courses!

18.6 Active Inference

- Andy Clark about Active Interference: How the Brains shapes reality (60 min)

18.7 Running LLM's locally

On your laptop/desktop or on a company server:

- ollama
- LM-studio
- Open Web AI

18.8 Coding with GenAI

- cursor course: AI Fundamentals
- vs code with co-pilot (free plan)
- Cursor.com (20 euro p/m)
- AIDER.chat (free)
- Open Devin: Create any Application with Open Source AI Engineer
- Avante (AI in neovim, free)

18.9 MCP - Model Context Protocol

A way (AI) systems can communicate to each other. This way it helps building modular (AI) systems.

- MCP Quickstart
- Short deeplearning.ai course MCP

18.10 GenAI

- awesome GenAI guide
- huggingface

18.11 Some more sites, nice to play around with

- https://skyreels.ai/
- https://civitai.com/

19. AI versus education

19.1 Media & Opinions

- bron: word-geen-ai-zombie-zo-blijf-je-kritisch-in-een-wereld-vol-ai
- bron: ICT maakt eigen 'zelf in te kleuren' AI-opleiding mogelijk
- Saçan: Schuurpapier voor het onderwijs…
- bron.fontys.nl/nieuw-fraudebeleid-met-focus-op-preventie
- How AI is changing education
- column-mark-de-graaf-ga-ict-studeren
- bron.fontys: een-eigen-ai-tool-voor-fontys
- Three things chess can teach us…

19.2 Tools, Best Practices & lesson material

- EduGenai (Npuls)
- How to cite ChatGPT? Use AI Archive
- npuls: AI-GO Raamwerk-AI-Geletterdheid-in-het-Onderwijs
- aiarchives.org
- You did it together with AI? Make a statement!
- hbo-i-outcomes-example-generator chatbot
- https://roadmap.sh/ai

19.3 The E-Bike Effect: Cognitive Offloading and Desired Difficulties

In his TEDx talk, Barend Last introduces a powerful metaphor for understanding our relationship with AI: **AI is the e-bike for our thinking**. This concept helps explain the nuances of "cognitive offloading" – outsourcing our mental tasks to technology.

You can find the talk and resources here: - LinkedIn Post by Barend Last - YouTube Video of the TEDx Talk

19.3.1 The Core Metaphor: Two Ways to Use the E-Bike

The talk highlights two distinct ways we can use AI, mirrored in how people use an e-bike:

1. **Making a Tedious Task Bearable**: Like someone who dislikes cycling but uses an e-bike to get to work, we can use AI to accomplish the same necessary tasks with less effort.
2. **Exploring New Horizons**: Like an avid cyclist using an e-bike to travel further and faster, we can use AI to reach new cognitive destinations and achieve things we couldn't before.

19.3.2 Cognitive Offloading & Desired Difficulties

This isn't a new phenomenon (think calculators or GPS), but AI supercharges it. There's a trade-off: while offloading can make us more efficient, over-reliance can weaken our own cognitive "muscles".

This leads to the idea of **"desired difficulties"**. Just as we go to the gym to stay physically fit, we should consciously choose when to tackle mental challenges without AI to keep our minds sharp. It's about finding a balance between using AI as a tool and ensuring we still get our mental workout. 🧠💪

19.3.3 Transforming, Not Just Replacing, Thinking

A key insight is that AI doesn't just eliminate thinking; it can *transform* it. An experiment cited in the talk showed that students using AI to brainstorm ideas for a paperclip generated more ideas. While they *felt* less creative, their cognitive effort shifted from idea generation to the equally demanding skills of **evaluating and refining** the AI's output.

The new generation might become like **conductors of an orchestra**, not playing every instrument but knowing how to bring them all together to create something beautiful.

19.3.4 Key Takeaway for Education

For education, the challenge is to teach students how to make conscious choices. They need to learn when AI is a helpful shortcut and when it's a springboard for deeper learning. This requires creating educational environments that build in "desired difficulties" – productive struggles, both with and without AI.

The talk concludes with a powerful statement:

> **"AI doesn't make us dumber, dumb choices do."**

20. Jobs and AI

July 2025

- Music: Fake or real?
- FD: ai vervangt de programmeur nog niet
- pabo-wint-aan-populariteit-ict-en-fysio-juist-niet
- UWV: Kansrijke beroepen 2025-2026

21. Resources and References GenAI

21.1 How te mention that you did use AI?

- fontys.libguides.com/apa/AI

21.2 Blogs and articles

Perplexity is often a great start for finding things (with references): perplexity.ai

- Jessy: Het belang van duidelijke AI-prompts
- Journalists on Hugging Face
- How polite should we be when prompting LLMs?
- Information literacy and chatbots as search

To understand about Transformers this is a very nice start: https://ig.ft.com/generative-ai/ 'Our own' page about (Gen)AI: https://stasemsoft.github.io/FontysICT-sem1/docs/artificial-intelligence/ai.html To dive further into how Transformers works: https://www.deeplearning.ai/short-courses/how-transformer-llms-work/ and also to other short courses on deeplearning.ai The development I showed was https://www.cursor.com/ you have like only 500 requests for free… after that you could choose to pay 20 euro a Month (yes, that can be a lot for students, I know), or look for alternatives, 2 of which I tried a bit (you can use local LLM's with them, which basically makes them free): AIDER: https://aider.chat/.
Avante: https://github.com/yetone/avante.nvim (but then you need to learn about 'vi': https://neovim.io/ which is a hurdle).

21.3 Online Platforms

- spacy.io : NLP

21.4 (Short) Courses

- Short courses at Deeplearning.ai
  - Implementations of papers

  - Benchmarks
  - State-of-the-art tracking

21.5 Code Repositories

- Papers With Code
  - Implementations of papers
  - Benchmarks
  - State-of-the-art tracking

21.6 Community Resources

- Distill.pub
  - Interactive explanations
  - Visual learning
  - Deep insights

21.7 Academic Papers: Modern Breakthroughs

- "Deep Residual Learning for Image Recognition" (He et al., 2015)
- "Attention Is All You Need" (Vaswani et al., 2017)
- "Language Models are Few-Shot Learners" (Brown et al., 2020)

22. No-Code / Low Code

Worth looking at:

- docs.oap.langchain.com
- n8n.io
- flowai.cc

23. EU AI Act

- youtube 4min: How is Europe becoming a leader in AI?
- SURF startdocument AI Act

24. Popular Data Sources

Data is important in AI projects. The quality and quantity of your data often matter more than the sophistication of your model.

- Kaggle
  - Competitions and datasets
  - Active community
  - Detailed documentation
- Eindhoven open data
  - lots of data about Eindhoven
- E-MM1: a big open source dataset

Part III

Train, Fine Tune, RAG

25. Train, Fine Tune, RAG

Several ways to ‘teach’ the AI about the knowledge it needs to perform the task you need it for. The most easy of these is building a RAG system: Retrieval Augmented Generation.

26. RAG: Retrieval Augmented Generation

Good to know about if you are in the AI field.

- deeplearning.ai course: Chat with your data

27. Finetune

28. Training

Training a model from scratch is a complex and resource-intensive process. It involves collecting a large dataset, preprocessing the data, and training the model using powerful hardware. This is typically done by large organizations with significant resources.

short course: fine tuning

29. Visual Recognition

CLIP-models, dyno, yolo, resnet, alexnet.

Part IV

Tools-n-Technologies

30. Tools

30.1 Agents

30.1.1 Agent Development Kit

- HF: Introduction to Agents
- https://google.github.io/adk-docs/https://google.github.io/adk-docs/

30.1.2 Open Agent Platform

- docs.oap.langchain.com

30.2 MCP - Model Context Protocol

MCP is a standardization of the way to how LLM's connect to other tools.

- modelcontextprotocol.info/
- Example Clients
- mcpservers.org
- github.com/r-huijts
- Servers
- Greg Isenberg/Ras Mic explaining MCP
- short course MCP at deeplearning.ai
- Ruud mijn-nieuwe-mcp-server-laat-ai-zichzelf-actief tegenspreken

31. Git / Version Control

One good reason to start using Git: Take control while Vibe Coding: use Git to exactly see and control changes.

A video

- Vibe Coding Course 7 – Version Control Basics (Git)

Some learning material

- Fontys ICT learning material
- Fontys ICT learning material - alternative
- great exercising site: learngitbranching.js.org
- learn git while playing minesweeper

Some more possibly interesting videos

- Cursor Vibe Coding Tutorial - For COMPLETE Beginners
- Let it cook - Vibe Coding with VS Code - Episode 1

32. Agents

32.1 Agent Development Kit

- HF: Introduction to Agents
- https://google.github.io/adk-docs/https://google.github.io/adk-docs/

32.2 Open Agent Platform

- docs.oap.langchain.com

33. MCP - Model Context Protocol

MCP is a standardization of the way to how LLM's connect to other tools.

- modelcontextprotocol.info/
- Example Clients
- mcpservers.org
- github.com/r-huijts
- Servers
- Greg Isenberg/Ras Mic explaining MCP
- short course MCP at deeplearning.ai
- Ruud mijn-nieuwe-mcp-server-laat-ai-zichzelf-actief tegenspreken

34. ACP - Agent Communication Protocol

To let Agents communicate, no matter what framework the Agents were built in.

- agentcommunicationprotocol.dev
- deeplearning.ai on ACP

35. Div tools that could be interesting

36. Journalism and AI

- stichtingrpo.nl: introductie-ai-kompas
- Hey Aftonbladet (chatbot): What do YOU want to know?

Part V

Neuron & Network

37. If you prefer a story...

The story that follows right here explains the ideas behind a Neural Network from a technical perspective. If you would rather read an Instructive Story, a Saga, read it online at: Lonn's neural-net-saga. Scroll down a bit and start reading 'The Percy Chronicles: A Neural Network Saga'. At the end of that story you will find some python to get hands-on with.

38. Understanding the Perceptron

38.1 The Biological Inspiration: From Brain Neurons to Artificial Intelligence

The Perceptron represents one of the most fundamental concepts in artificial intelligence, drawing its inspiration directly from the human brain's neural structure. This groundbreaking idea was first introduced in 1943 by Warren S. McCulloch and Walter Pitts in their seminal paper 'A Logical Calculus of the Ideas Immanent in Nervous Activity', where they proposed a mathematical model of biological neurons.

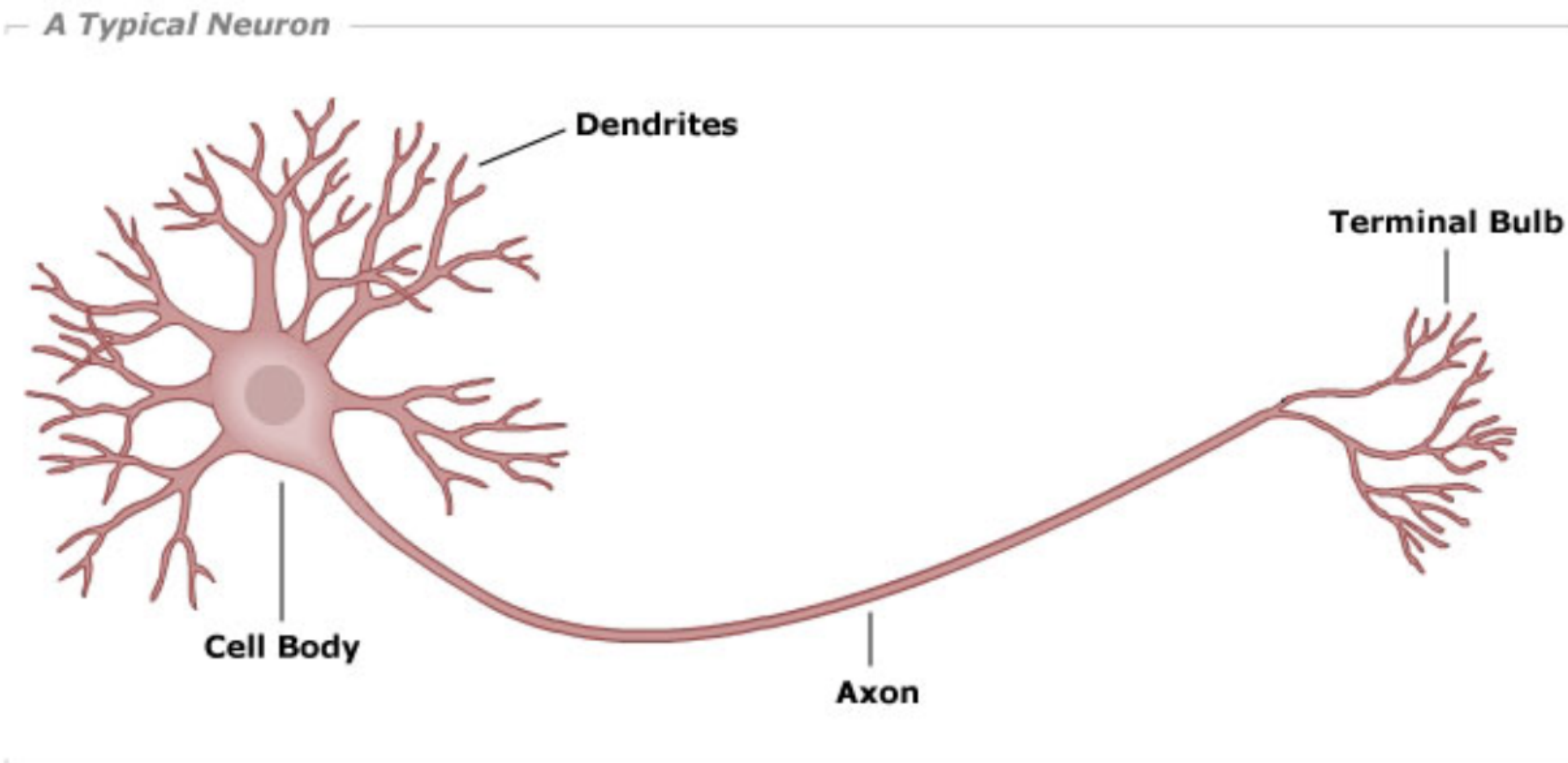

Figure 38.1: A typical biological neuron structure

38.2 From Biology to Machine: Implementing a Perceptron

A Perceptron's architecture mirrors its biological counterpart through three key components: `inputs`, `weights`, and a `bias`. Each input connection has an associated weight that determines its relative importance, while the bias helps adjust the Perceptron's overall sensitivity to activation.

Let's look at a simple yet useful perceptron with 2 inputs.

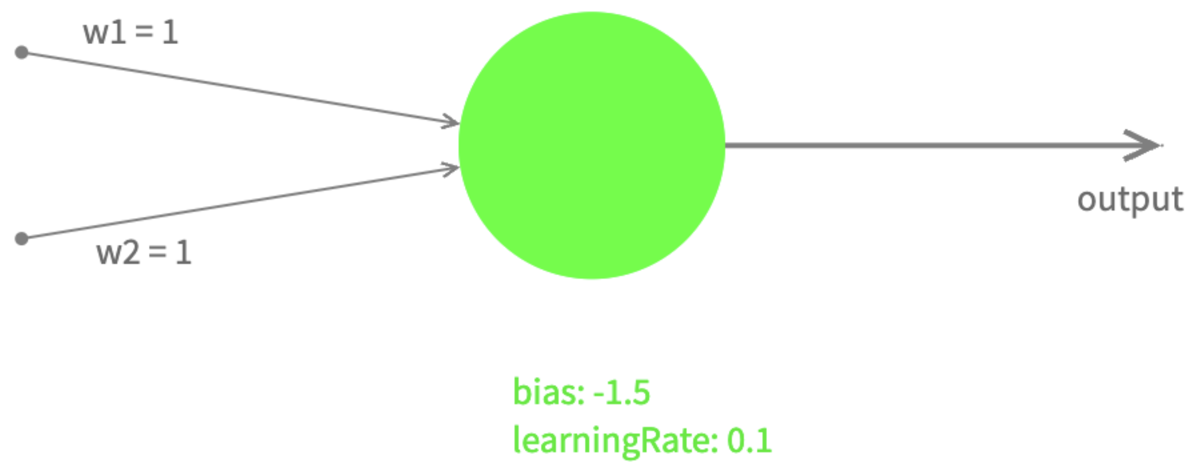

Figure 38.2: Perceptron's architectural diagram

We'll call our input values `x1`, `x2` with their corresponding weights `w1`, `w2`. The Perceptron processes these inputs in two steps:

1. First, it calculates a `weighted sum` and adds the bias: `z := w1*x1 + w2*x2 + bias`
2. Then, it applies what we call an **activation function** to produce the final output: let's use a very simpel activation function, called a Step function:

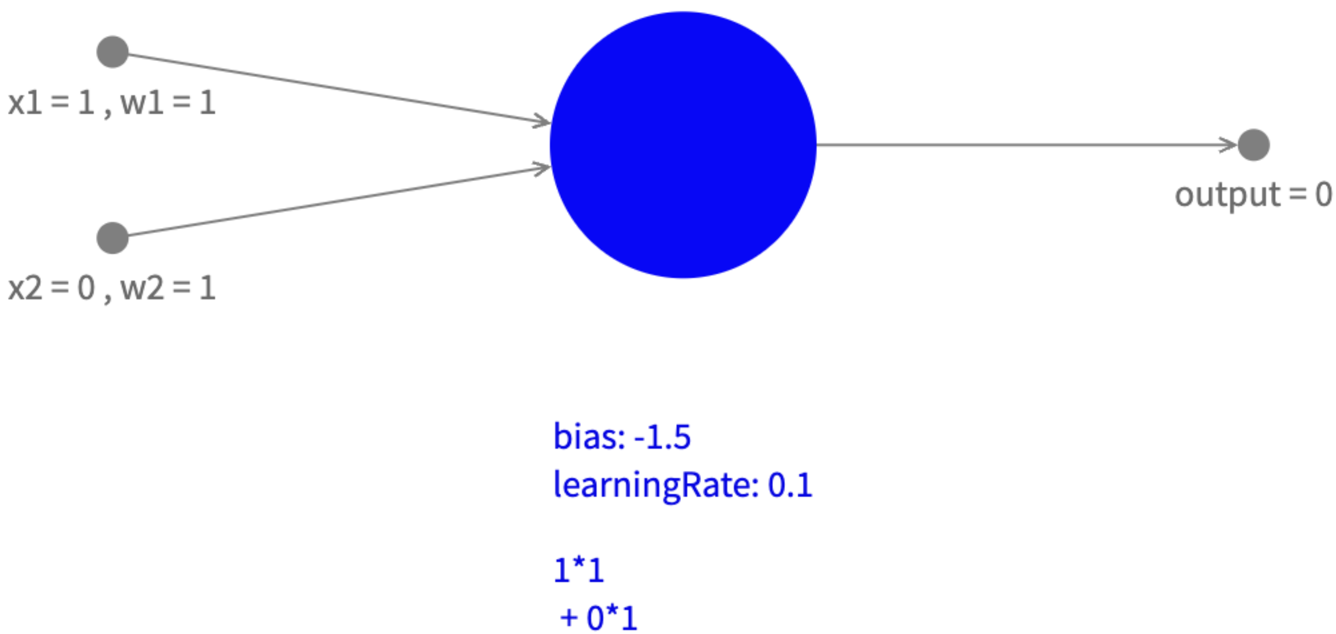

Figure 38.3: Perceptron's architectural diagram

$$\begin{cases} \text{Output is 1 if } z > 0 \\ \text{Output is 0 if } z \leq 0 \end{cases}$$

which determines the final output.

Let's restrict ourselves for now to possible input values 0 and 1: If we look at all possibilities combinations of input and the corresponding output we can create a table:

Input 1	Input 2	Output
0	0	0

Input 1	Input 2	Output
0	1	0
1	0	0
1	1	1

A close look will tell us that the output is only 1 when inputs are 1, and 0 in all other cases, which you could recognize as a logical AND. So with these weights and bias this Perceptron can be used to act as a logical AND.

For different values it will behave like a logical OR (and more). Can you come up with those values?

38.3 Network

By combining several Perceptrons (sending the output of a perceptron to the input of another one) you can probably imagine that it is possible to create Networks of Perceptrons. By changing the values of weights and biases of the connected Perceptrons it is possible to build complex electronic circuits.

When we generalize this concept to other values, not only 0 and 1, and different activation functions, the Perceptron becomes an incredibly versatile tool. This generalization opens up possibilities for pattern recognition, classification tasks, regression problems, and complex decision-making systems. This is where the true power of neural networks begins to emerge, as they can learn to handle continuous data and make sophisticated decisions based on multiple inputs.

Up until now we didn't look at how a perceptron can learn and become smarter. That will be subject of next chapter chapters. The concept of a Perceptron was generalized to what we now call an (artificial) Neuron.

Search terms: Perceptron, Artificial Neuron, `Multi Layered Perceptron` (MLP), `(Artificial) Neural Network` (ANN).

38.4 First Implementation of Perceptron algorithm

According to Wikipedia:

> The artificial neuron network was invented in 1943 by Warren McCulloch and Walter Pitts in 'A logical calculus of the ideas immanent in nervous activity'. the Perceptron Machine was first implemented in hardware in the Mark I, which was demonstrated in 1960.
>
> It was connected to a camera with 20×20 cadmium sulfide photocells to make a 400-pixel image. The main visible feature is the sensory-to-association plugboard, which sets different combinations of input features. To the right are arrays of potentiometers that implemented the adaptive weights.

38.5 Reference

- wikipedia: perceptron

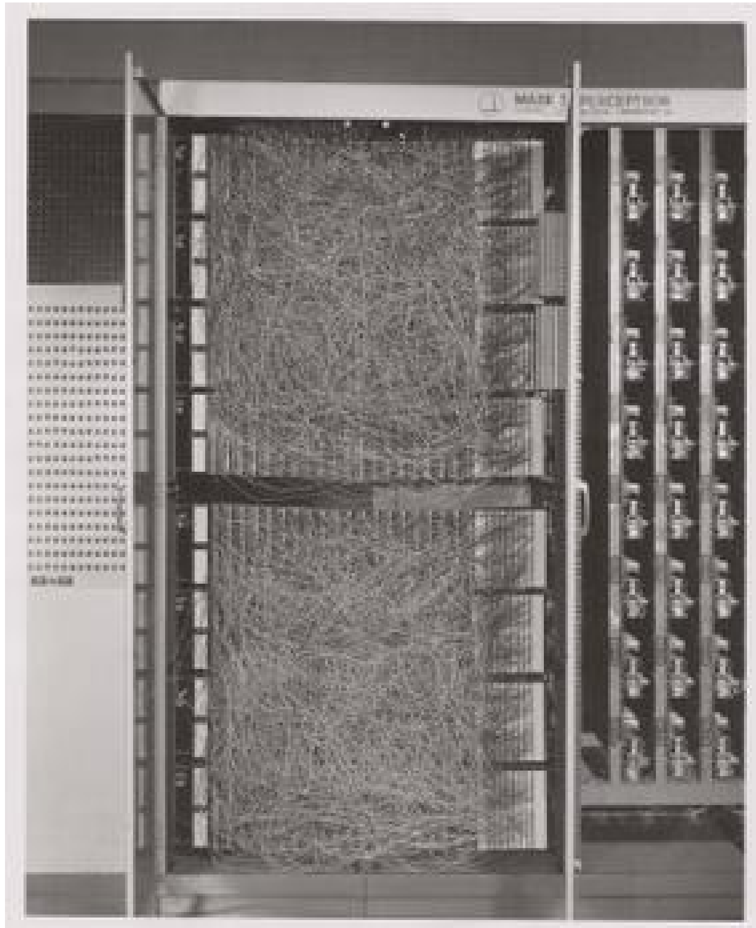

Figure 38.4: The Mark I Perceptron machine, the first implementation of the perceptron algorithm (source: wikipedia)

39. The Learning Perceptron

One of the most fascinating aspects of Perceptrons is their ability to learn from examples. Instead of manually setting weights and bias, we can train a Perceptron to discover the optimal parameters through a process called supervised learning.

39.1 The Learning Algorithm

The learning process follows these key steps:

1. Start with random weights and bias
2. Present a training example
3. Compare the Perceptron's output with the desired output
4. Adjust the weights and bias based on the error
5. Repeat with more examples until performance is satisfactory

39.1.1 Mathematical Foundation

The weight update rule is elegantly simple:

```
new_weight := current_weight + learning_rate * error * input
```

Where:

- `learning_rate` is a small number (like 0.1) that controls how big each adjustment is
- `error` is the difference between desired and actual output (1 or -1)
- `input` is the input value for that weight

39.1.2 Training Process

To train the Perceptron, we have to have `labeled data` (ie. input data combined with the desired output for those values)

So for training AND gate behavior we have to list all combinations of 2 bits that are possible as input, and also the desired output value:

```
| Input  | Desired Output |
```

```
|--------|----------------|
| (0, 0) | 0              |
| (0, 1) | 0              |
| (1, 0) | 0              |
| (1, 1) | 1              |
```

and training (1 epoc) means calling the train function with each of these examples:

```
foreach dataItem in trainingData do:
    inputs := dataItem[0]
    desiredOutput := dataItem[1]
    learningPerceptron train(inputs, desiredOutput)
```

39.2 Visualizing the Learning Process

As the Perceptron learns, its decision boundary gradually moves to the correct position. You can monitor this progress by:

1. Tracking the error rate over time
2. Visualizing the decision boundary's movement
3. Testing the Perceptron with new examples

39.3 Practical Considerations

For successful learning: - Ensure your training data is representative - Consider using multiple training epochs (complete passes through the data) - Monitor for convergence (when the weights stabilize) - Be aware that not all problems are linearly separable

In the next chapter, we'll explore the limitations of what a single Perceptron can learn, which will lead us naturally to the need for more complex neural networks.

40. Understanding Perceptron Limitations

40.1 The XOR Problem: A Classic Challenge

While Perceptrons are powerful tools for many classification tasks, they face a fundamental limitation: they can only solve linearly separable problems. The classic example of this limitation is the XOR (exclusive OR) function.

40.1.1 What is XOR?

The XOR function outputs 1 only when exactly one of its inputs is 1: - Input (0,0) → Output: 0 - Input (0,1) → Output: 1 - Input (1,0) → Output: 1 - Input (1,1) → Output: 0

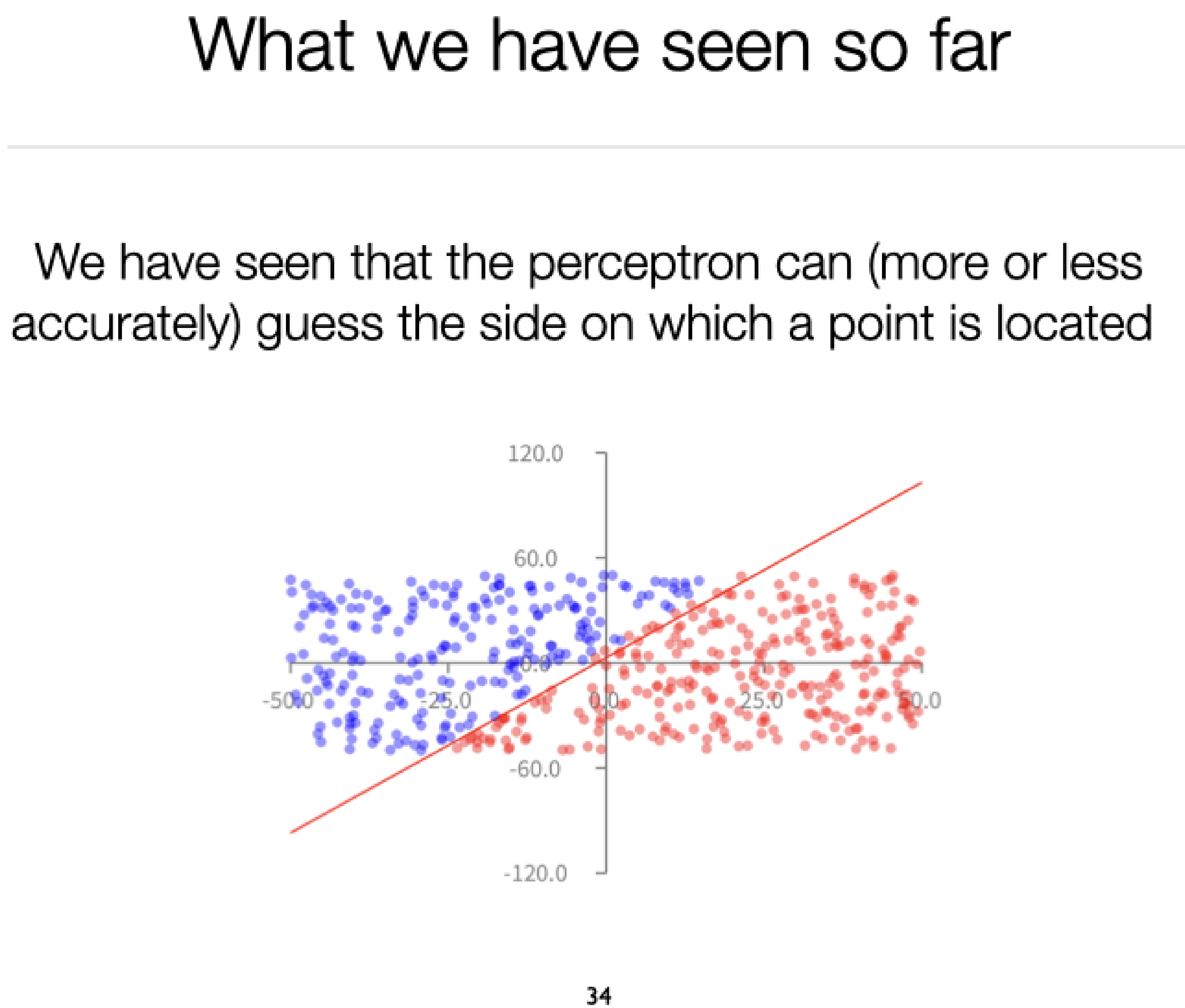

Figure 40.1: Visual representation of XOR problem

40.1.2 Why Can't a Single Perceptron Solve XOR?

A Perceptron creates a single straight line (or hyperplane in higher dimensions) to separate its outputs. However, the XOR problem requires two separate lines to correctly classify all points.

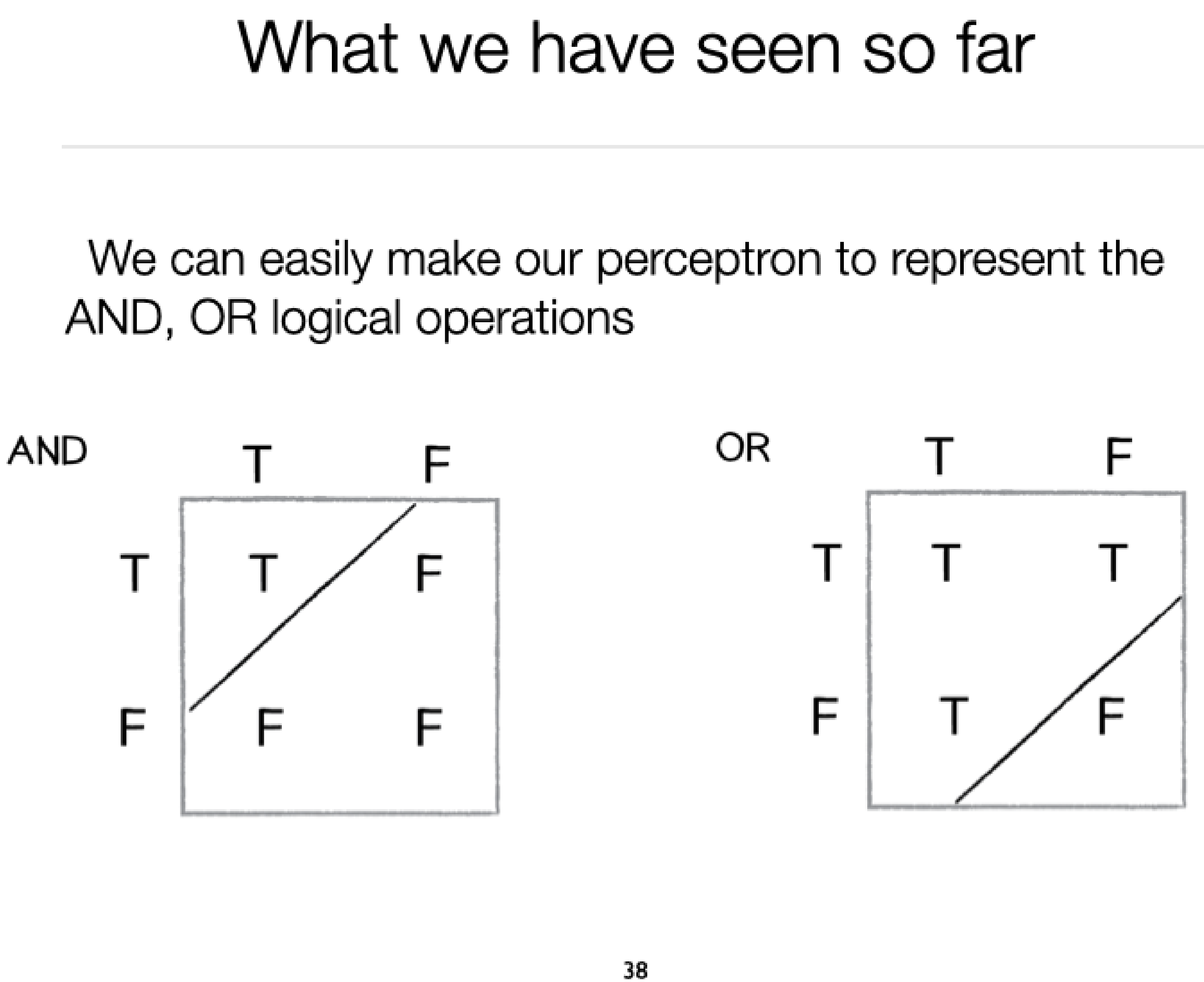

Figure 40.2: Attempted linear separation of XOR

As you can see, no single straight line can separate the points where output should be 1 (blue) from points where output should be 0 (red).

40.2 The Solution: Multiple Layers

To solve the XOR problem, we need to combine multiple Perceptrons in layers. This is our first glimpse at why we need neural networks!

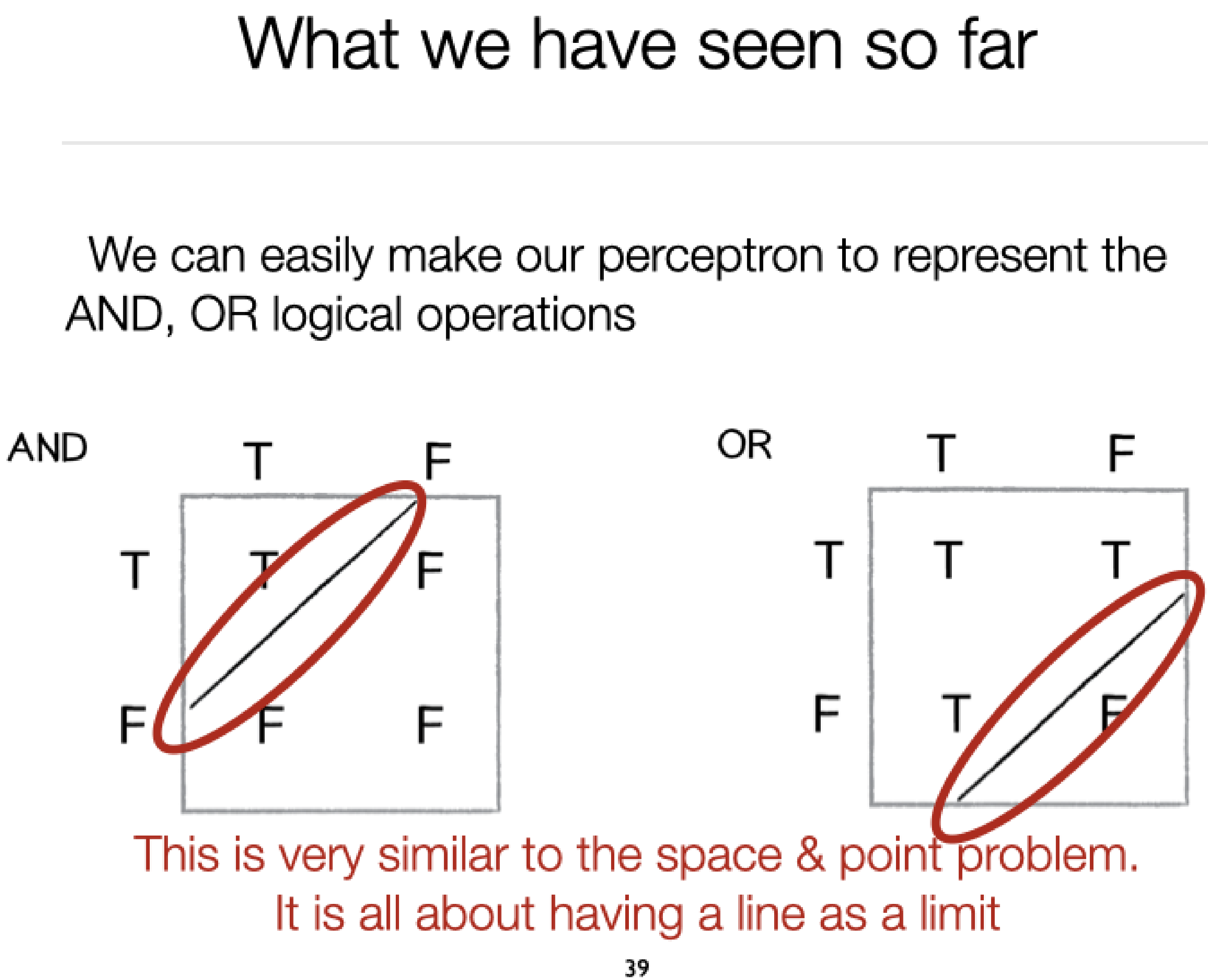

Figure 40.3: Multi-layer solution

By using multiple Perceptrons, we can: 1. First create separate regions with individual Perceptrons 2. Then combine these regions to form more complex decision boundaries

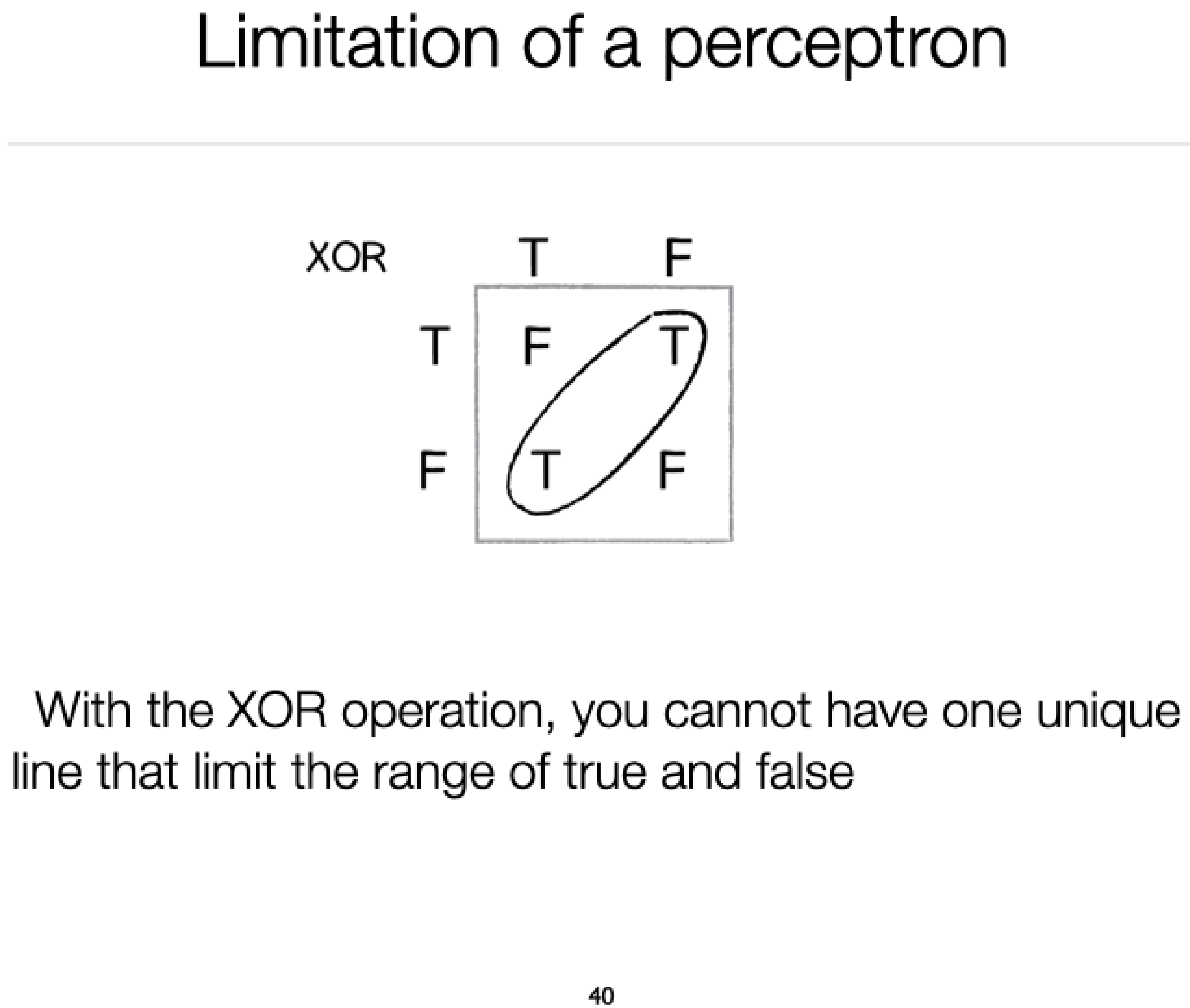

Figure 40.4: Complete neural network solution

40.3 Key Takeaways

1. Single Perceptrons can only solve linearly separable problems
2. Many real-world problems (like XOR) are not linearly separable
3. Combining Perceptrons into networks overcomes this limitation
4. This limitation led to the development of multi-layer neural networks

In the next section, we'll explore how to build and train these more powerful multi-layer networks.

41. Introduction to Neural Networks

41.1 Beyond Single Perceptrons: Building Neural Networks

Having seen the limitations of single Perceptrons, we now venture into the fascinating world of neural networks. These powerful structures combine multiple Perceptrons in layers to solve complex problems that single Perceptrons cannot handle.

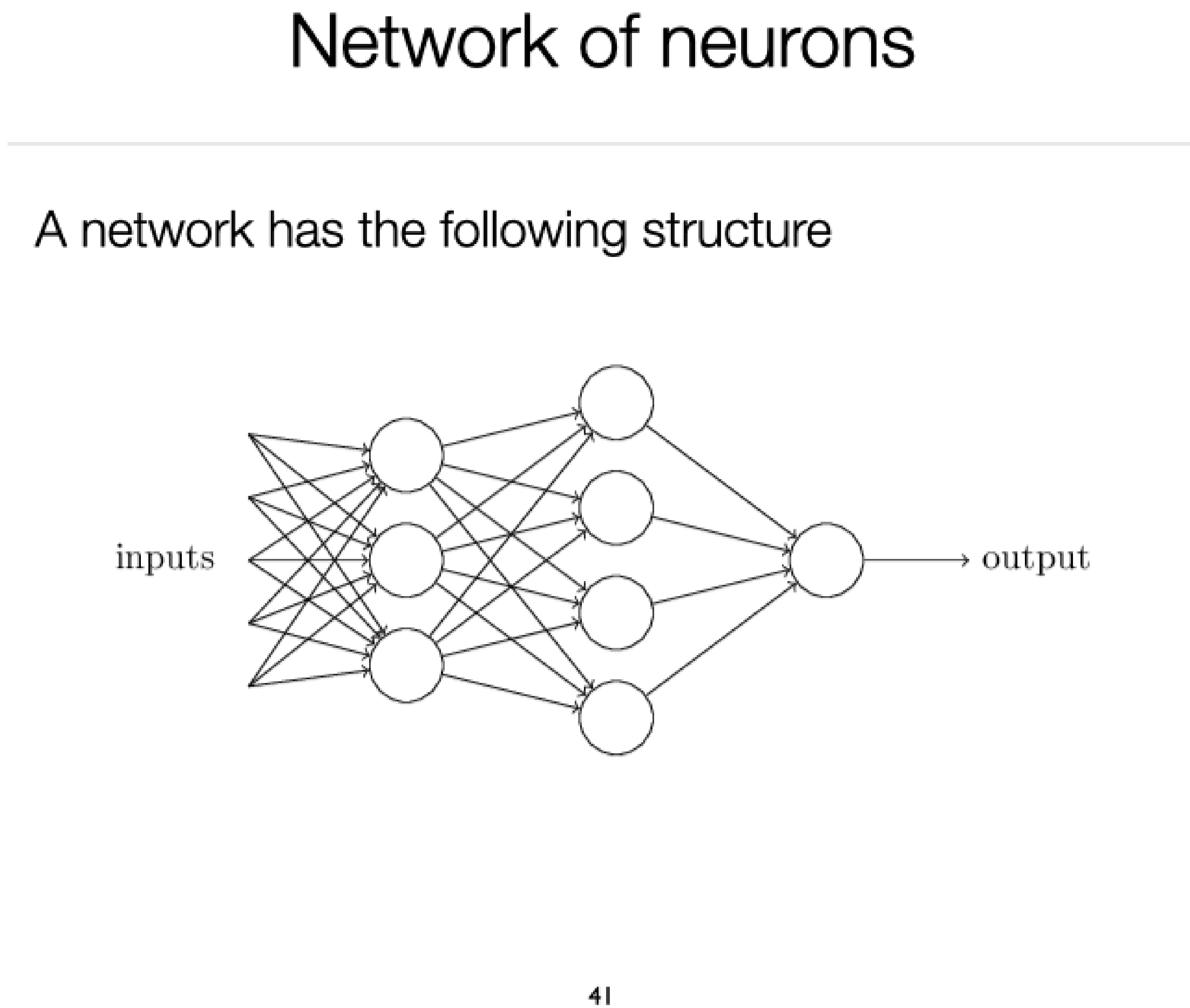

Figure 41.1: Basic neural network architecture

41.2 Understanding Network Architecture

A typical neural network consists of three main components:

1. **Input Layer**: Receives the raw data
2. **Hidden Layer(s)**: Processes the information through multiple Perceptrons
3. **Output Layer**: Produces the final result

41.2.1 Key Components

Each connection in the network has: - A weight that determines its strength - A direction of information flow (forward only) - An associated neuron that processes the incoming signals

41.3 How Information Flows

The network processes information in these steps:

1. Input values are presented to the input layer
2. Each neuron in subsequent layers:
    - Receives weighted inputs from the previous layer
    - Applies its activation function
    - Passes the result to the next layer
3. The output layer produces the final result

41.4 Creating a Simple Network

You probably have seen a picture of a neural network before.

Neural Networks can

1. solve problems that are more difficult.
2. Handle complex pattern recognition
3. Learn hierarchical features automatically
4. Scale well to large problems

In the next sections, we'll explore practical applications and see how to train networks on real-world data.

42. Practical Example: Classifying Iris Flowers

42.1 A Real-World Machine Learning Challenge

The Iris flower classification problem is a classic example in machine learning. It involves predicting the species of an Iris flower based on measurements of its physical characteristics. This problem perfectly illustrates how neural networks can solve real-world classification tasks.

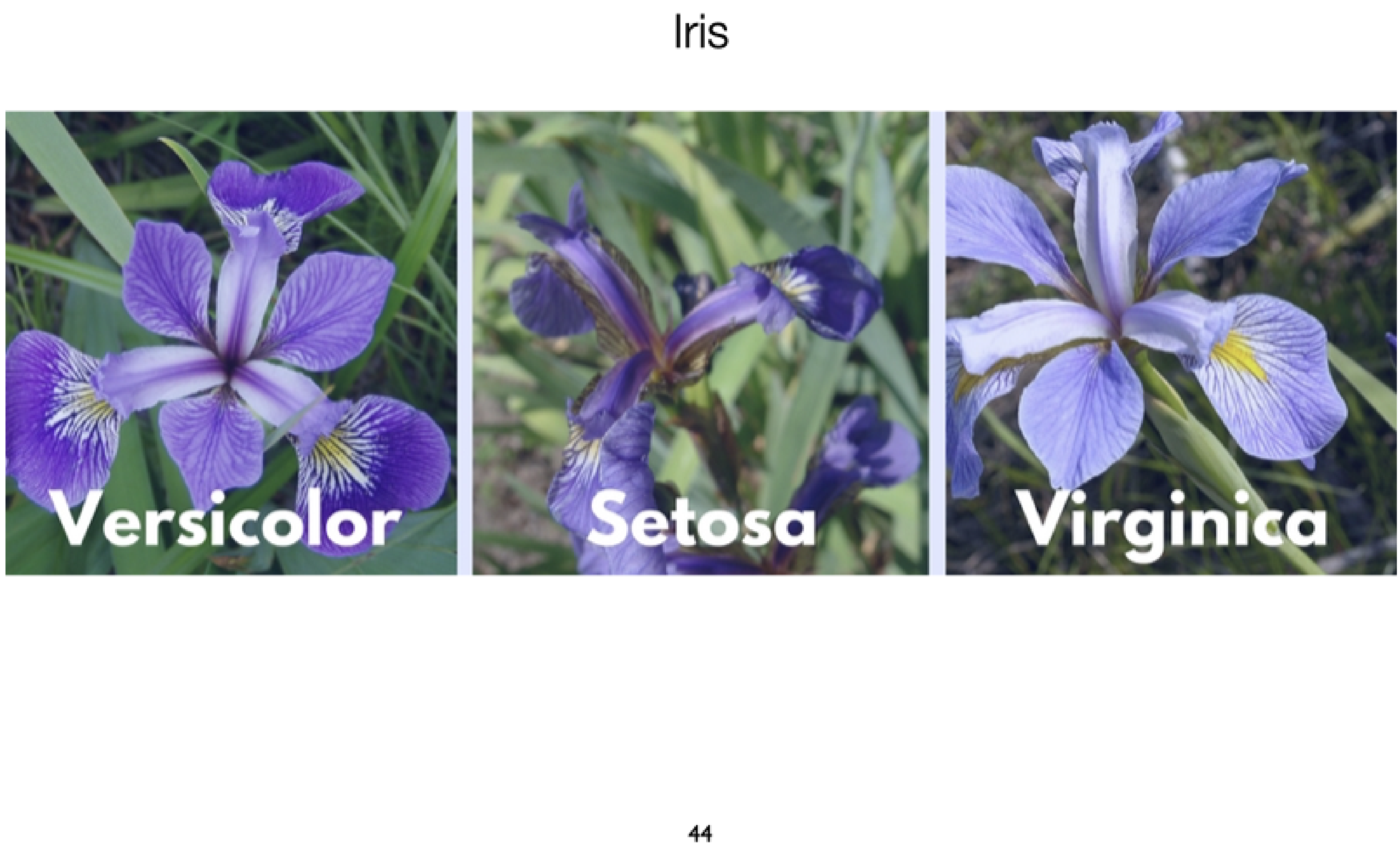

Figure 42.1: Different types of Iris flowers

42.2 The Dataset

The Iris dataset includes measurements of three different Iris species: - Iris Setosa - Iris Versicolor - Iris Virginica

For each flower, we have four measurements: 1. Sepal length 2. Sepal width 3. Petal length 4. Petal width

Building a network that can do this is really outside of the scope of these notes, but a lot of info can be found on the internet on `Iris Classification`.

42.3 Key Learning Points

1. Neural networks can handle multi-class classification
2. Real-world data often needs preprocessing
3. We can measure success with accuracy metrics
4. The same principles apply to many similar problems

This practical example demonstrates how neural networks can solve real classification problems. In the next section, we'll explore the mathematics behind how these networks learn.

43. The Mathematics Behind Neural Networks

43.1 Understanding the Magic

While neural networks might seem magical, they're built on solid mathematical foundations. Let's demystify (a bit of) how they actually work under the hood.

43.2 The Building Blocks

43.2.1 1. Neurons and Weights

To be formally correct we should say `artificial neuron` to distinguish them from `biological neurons` like we have in our brain. A neuron normally has inputs: 1, or 2, or $\cdots$

Each neuron performs two key operations: 1. Weighted sum of inputs. 2. Activation function: $a = f(z)$

43.2.2 2. Activation Functions

Common activation functions include:

1. Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
    - Outputs between 0 and 1
    - Useful for probability predictions
2. ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$
    - Simple and efficient
    - Helps prevent vanishing gradients
3. Tanh: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
    - Outputs between -1 and 1
    - Often better than sigmoid for hidden layers

43.3 The Learning Process

43.3.1 1. Forward Propagation

Information flows through the network.

43.3.2 2. Loss Calculation

Measure the network's Error and Backpropagation

- What is the output?
- What would be my desired output?

The smaller the difference between the output I got and the output I desired, the better the output of my model is. This difference is calculated with a so-called Loss function. Backpropagation is an algorithm that helps make that difference small. When backpropagation is performed we call that Training the AI model.

44. Exploring Neural Network Architectures

44.1 The Rich Landscape of Neural Networks

Neural networks come in many shapes and sizes, each designed to excel at specific types of tasks. Let's explore some of the most important architectures and their applications.

44.2 Feedforward Neural Networks (FNN)

The classic architecture: Information flows in one direction:

- Input layer → Hidden layer(s) → Output layer
- Perfect for classification and regression tasks
- Examples: Our Iris classifier, handwriting recognition

44.3 Convolutional Neural Networks (CNN)

Inspired by the human visual cortex:

- Specialized for processing grid-like data (images, video)
- Uses convolution operations to detect patterns
- Excellent at feature extraction
- Applications: Image recognition, computer vision, medical imaging

44.4 Recurrent Neural Networks (RNN)

Networks with memory:

- Can process sequences of data
- Information cycles through the network
- Great for time-series data and natural language

- Applications: Language translation, speech recognition, stock prediction

44.5 Long Short-Term Memory (LSTM)

A sophisticated type of RNN:

- Better at remembering long-term dependencies
- Controls information flow with gates
- Solves the vanishing gradient problem
- Applications: Text generation, music composition

44.6 Autoencoders

Self-learning networks:

- Learn to compress and reconstruct data
- Useful for dimensionality reduction
- Can detect anomalies
- Applications: Data compression, noise reduction, feature learning

44.7 Generative Adversarial Networks (GAN)

Two networks competing with each other:

- Generator creates fake data
- Discriminator tries to spot fakes
- Through competition, both improve
- Applications: Creating realistic images, style transfer, data augmentation

44.8 Choosing the Right Architecture

The choice of architecture depends on:

1. Type of data (images, text, time-series)
2. Task requirements (classification, generation, prediction)
3. Available computational resources
4. Need for real-time processing

44.9 Future Directions

Neural network architectures continue to evolve:

- Hybrid architectures combining multiple types
- More efficient training methods
- Better handling of uncertainty
- Integration with other AI techniques

In the next section, we'll dive deeper into training these networks effectively.

45. Resources and References AI

45.1 Books

1. **Neural Networks and Deep Learning**
   - Author: Michael Nielsen
   - Free Online Book
   - Perfect for beginners and intermediate learners
   - Clear explanations with interactive examples
2. **Deep Learning**
   - Authors: Ian Goodfellow, Yoshua Bengio, Aaron Courville
   - Available Online
   - Comprehensive coverage of deep learning
   - Industry standard reference
3. **Agile AI in Pharo**
   - Author: Alexandre Bergel
   - Practical implementation in Pharo
   - Hands-on examples and exercises
   - Book Link

45.2 Video Courses and Tutorials

45.2.1 1. Foundational Series

- 3Blue1Brown Neural Networks
  - Visual explanations
  - Mathematical intuition
  - Clear animations

https://www.youtube.com/watch?v=O5xeyoRL95U

45.2.2 2. Programming Tutorials

- Fast.ai Deep Learning Course
  - Practical approach
  - Top-down learning
  - Real-world applications

45.2.3 3. Advanced Topics

- Stanford CS231n
  - Computer Vision
  - Deep Learning
  - State-of-the-art techniques

45.3 Online Platforms

45.3.1 1. Interactive Learning

- Kaggle Learn
  - Hands-on exercises
  - Real datasets
  - Community support

45.3.2 2. Research Papers

- arXiv Machine Learning
  - Latest research
  - Open access
  - Preprint server

45.3.3 3. Code Repositories

- Papers With Code
  - Implementations of papers
  - Benchmarks
  - State-of-the-art tracking

45.4 Community Resources

45.4.1 1. Forums and Discussion

- r/MachineLearning
- Cross Validated
- AI Stack Exchange

45.4.2 2. Blogs and Newsletters

- Distill.pub
  - Interactive explanations
  - Visual learning
  - Deep insights

45.4.3 3. Tools and Libraries

- TensorFlow
- PyTorch
- Scikit-learn

45.5 Academic Papers

45.5.1 Foundational Papers

- "A Logical Calculus of Ideas Immanent in Nervous Activity" (McCulloch & Pitts, 1943)
- "Learning Internal Representations by Error Propagation" (Rumelhart et al., 1986)
- "Gradient-Based Learning Applied to Document Recognition" (LeCun et al., 1998)

45.5.2 Podcasts

- MLST: Machine Learning Street Talk
- Brainport/Iman interviews Sepp Hochreiter: XLSTM
- other podcasts from this series: 'Deep Dives with Iman'.
- Fontys AI Garage

45.5.3 Other

- email news letter: alphasignal.ai

Part VI

Experiments

46. Experiments

Experiments, maybe incomplete... never finished, the whole reutemeteut!

47. MCP hands-on

Diving in…

So MCP standardizes the way I can combine sources of info (like RAG?) with an LLM.

Duckduckgoing for 'MCP vs ollama hands-on' (adding CLI afterwards) gives me some links to look at, and after a closer look these still seem interesting:

- agentic-rag-and-mcp
- Ollama MCP bridge
- ollama-mcp
- https://modelcontextprotocol.io/introduction
- lazy terminal
- https://apidog.com/blog/neovim-mcp-server/

48. to look at still:

- agentic-rag-and-mcp
- Ollama MCP bridge
- ollama-mcp
- https://modelcontextprotocol.io/introduction
- lazy terminal
- https://apidog.com/blog/neovim-mcp-server/

  First I need an MCP client and an MCP server.

49. MCP Client

- 5ire looks nice.
- oterm