Coen's AI Notes and Links

diverse

2025-07-30

Table of contents

1 Introduction **1**

2 AI Overview **3**

2.1 AI, Machine Learning, Deep Learning, Generative AI 3

I GenAI **5**

3 GenAI **7**

3.1 What is GenAI? 7
3.2 GPT - Generative Pre-Trained Transformer 7
3.3 Prompting 7
3.4 Hallucinating 7
3.5 RAG - Retrieval Augmented Generation 8
3.6 Active Inference 8
3.7 Running LLM's locally 8
3.8 Coding with GenAI 8
3.9 MCP - Model Context Protocol 8
3.10 GenAI 8
3.11 Some more sites, nice to play around with 9

4 AI versus education **11**

4.1 Media & Opinions 11
4.2 Tools, Best Practices & lesson material 11

5 Jobs and AI **13**

6 Resources and References GenAI **15**

6.1 Blogs and articles 15
6.2 Online Platforms 15
6.3 (Short) Courses 16
6.4 Code Repositories 16
6.5 Community Resources 16
6.6 Academic Papers: Modern Breakthroughs 16

7 No-Code / Low Code 17

II AI Act Europe 19

8 AI Act Resources and References 21

III Train, Fine Tune, RAG 23

9 Train, Fine Tune, RAG 25

10 RAG: Retrieval Augmented Generation 27

11 Finetune 29

12 Training 31

IV Data 33

13 Finding and Preparing Data 35
13.1 The Importance of Data 35
13.2 Popular Data Sources 35

V Related subjects 37

14 Agents 39
14.1 Agent Development Kit 39
14.2 Open Agent Platform 39

15 MCP - Model Context Protocol 41

16 ACP - Agent Communication Protocol 43

17 Journalism and AI 45

VI Neuron & Network 47

18 Understanding the Perceptron 49
18.1 The Biological Inspiration: From Brain Neurons to Artificial Intelligence 49
18.2 From Biology to Machine: Implementing a Perceptron 50
18.3 Network 52
18.4 First Implementation of Perceptron algorithm 52
18.5 Reference 52

19 The Learning Perceptron **55**
19.1 The Learning Algorithm 55
19.1.1 Mathematical Foundation 55
19.1.2 Training Process 56
19.2 Visualizing the Learning Process 56
19.3 Practical Considerations 56

20 Understanding Perceptron Limitations **57**
20.1 The XOR Problem: A Classic Challenge 57
20.1.1 What is XOR? 57
20.1.2 Why Can't a Single Perceptron Solve XOR? 58
20.2 The Solution: Multiple Layers 59
20.3 Key Takeaways 61

21 Introduction to Neural Networks **63**
21.1 Beyond Single Perceptrons: Building Neural Networks 63
21.2 Understanding Network Architecture 64
21.2.1 Key Components 64
21.3 How Information Flows 64
21.4 Creating a Simple Network 65

22 Practical Example: Classifying Iris Flowers **67**
22.1 A Real-World Machine Learning Challenge 67
22.2 The Dataset 68
22.3 Key Learning Points 68

23 The Mathematics Behind Neural Networks **69**
23.1 Understanding the Magic 69
23.2 The Building Blocks 69
23.2.1 1. Neurons and Weights 69
23.2.2 2. Activation Functions 69
23.3 The Learning Process 70
23.3.1 1. Forward Propagation 70
23.3.2 2. Loss Calculation 70

24 Exploring Neural Network Architectures **71**
24.1 The Rich Landscape of Neural Networks 71
24.2 Feedforward Neural Networks (FNN) 71
24.3 Convolutional Neural Networks (CNN) 71
24.4 Recurrent Neural Networks (RNN) 71
24.5 Long Short-Term Memory (LSTM) 72
24.6 Autoencoders 72
24.7 Generative Adversarial Networks (GAN) 72
24.8 Choosing the Right Architecture 72
24.9 Future Directions 72

25 Resources and References AI **75**

25.1 Books 75
25.2 Video Courses and Tutorials 75
25.2.1 1. Foundational Series 75
25.2.2 2. Programming Tutorials 76
25.2.3 3. Advanced Topics 76
25.3 Online Platforms 76
25.3.1 1. Interactive Learning 76
25.3.2 2. Research Papers 76
25.3.3 3. Code Repositories 76
25.4 Community Resources 77
25.4.1 1. Forums and Discussion 77
25.4.2 2. Blogs and Newsletters 77
25.4.3 3. Tools and Libraries 77
25.5 Academic Papers 77
25.5.1 Foundational Papers 77

VII Experiments 79

26 Experiments 81

27 MCP hands-on 83

28 to look at still: 85

29 MCP Client 87

Chapter 1

Introduction

Welcome! This is a Work-in-Progress, a collection of notes on AI I am collecting and which I use in my workshops about AI and GenAI. The newest version of this pdf can be downloaded from
here

It is not complete nor self-describing, but when you attended one of my workshops you will probably find familiar stuff in one or more chapters.

Our world is changing rapidly through AI and GenAI. One can ignore it or decide to not use it, but that does not stop it… One can also decide to dive in and help 'invent' the future, or at least learn about all the new stuff.

These notes started out as visualizations of Perceptrons and Neural Networks in the Glamorous Toolkit, which helped me give students insights in Neural Networks.

I advise to get hands-on with the tools around.

When using online AI tools, please keep the privacy in mind when using personal data! One way to make sure private data will stay private is using local AI's.

Please also keep the Societal impact in mind! We can use AI to help us all, but there is of course also a dark side:
People getting fired, it's easier to create fake news, a few people getting rich at the expense of others,
some nice activities (I like programming for example) will never be the same. Please use it wisely…

Figure 1.1: art and laundry

Chapter 2

AI Overview

2.1 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

Part I

GenAI

Chapter 3

GenAI

Recent:

- Amy Webb SxSW 2025 - Emerging Tech Trend

3.1 What is GenAI?

- Why not ask perplexity.ai ?

3.2 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)

3.3 Prompting

New 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

3.4 Hallucinating

- 'Which day do I have to put the garbage can out on the street?'
- 'Can you help me find my lost keys?'

- ‘Can you create an image of a watch that says it is 3 o’ clock?’

3.5 RAG - Retrieval Augmented Generation

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

3.6 Active Inference

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

3.7 Running LLM’s locally

On your laptop/desktop or on a company server:

- ollama
- LM-studio
- Open Web AI

3.8 Coding with GenAI

- 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)

3.9 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

3.10 GenAI

- awesome GenAI guide
- huggingface

3.11 Some more sites, nice to play around with

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

Chapter 4

AI versus education

4.1 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...

4.2 Tools, Best Practices & lesson material

- 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

Chapter 5

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

Chapter 6

Resources and References GenAI

6.1 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).

6.2 Online Platforms

- spacy.io : NLP

6.3 (Short) Courses

- Short courses at Deeplearning.ai
  - Implementations of papers
  - Benchmarks
  - State-of-the-art tracking

6.4 Code Repositories

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

6.5 Community Resources

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

6.6 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)

Chapter 7

No-Code / Low Code

Worth looking at:

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

Part II

AI Act Europe

Chapter 8

AI Act Resources and References

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

Part III

Train, Fine Tune, RAG

Chapter 9

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.

Chapter 10

RAG: Retrieval Augmented Generation

Chapter 11

Finetune

Chapter 12

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

Part IV

Data

Chapter 13

Finding and Preparing Data

13.1 The Importance of Data

Data is the foundation of most machine learning projects. The quality and quantity of your data often matter more than the sophistication of your model.

13.2 Popular Data Sources

- Kaggle
  - Competitions and datasets
  - Active community
  - Detailed documentation
- Eindhoven open data
  - lots of data about Eindhoven

Part V

Related subjects

Chapter 14

Agents

14.1 Agent Development Kit

- https://google.github.io/adk-docs/https://google.github.io/adk-docs/

14.2 Open Agent Platform

- docs.oap.langchain.com

Chapter 15

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
- Servers
- Greg Isenberg/Ras Mic explaining MCP
- short course MCP at deeplearning.ai
- Ruud mijn-nieuwe-mcp-server-laat-ai-zichzelf-actief tegenspreken

Chapter 16

ACP - Agent Communication Protocol

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

- agentcommunicationprotocol.dev
- deeplearning.ai on ACP

Chapter 17

Journalism and AI

- Hey Aftonbladet (chatbot): What do YOU want to know?

Part VI

Neuron & Network

Chapter 18

Understanding the Perceptron

18.1 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.

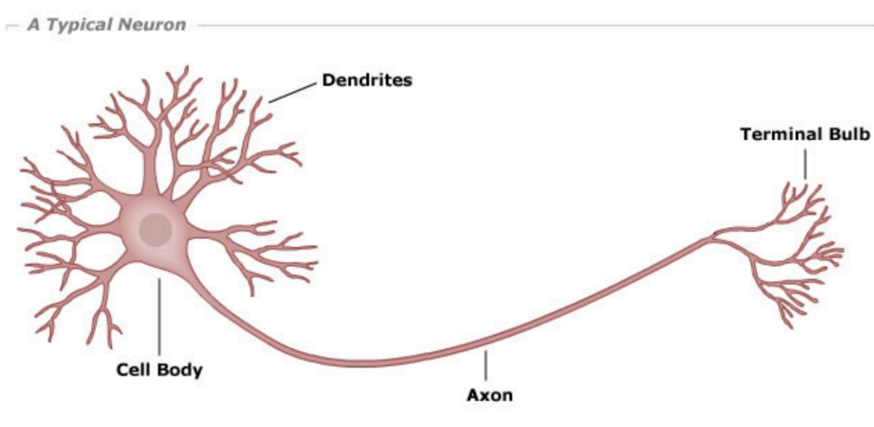

Figure 18.1: A typical biological neuron structure

18.2 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.

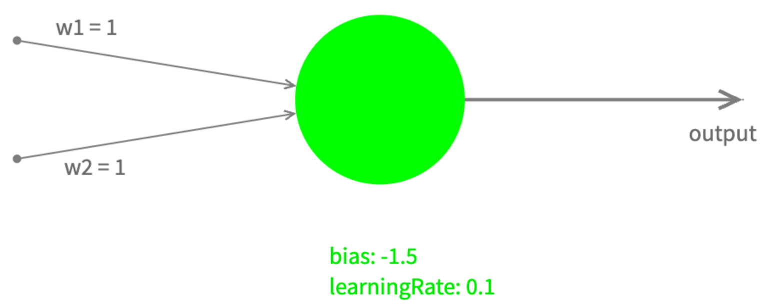

Figure 18.2: Perceptron's architectural diagram

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:

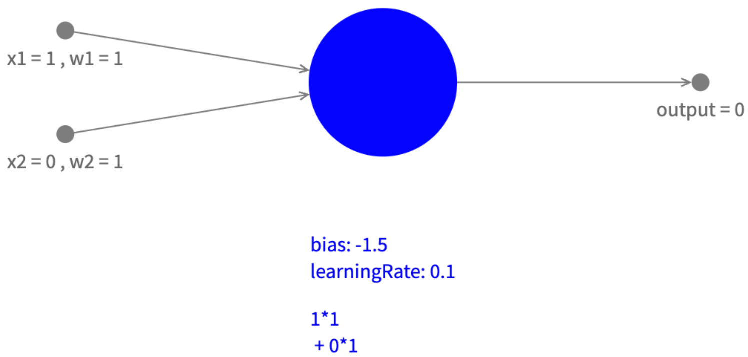

Figure 18.3: Perceptron's architectural diagram

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

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
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?

18.3 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).

18.4 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.

18.5 Reference

- wikipedia: perceptron

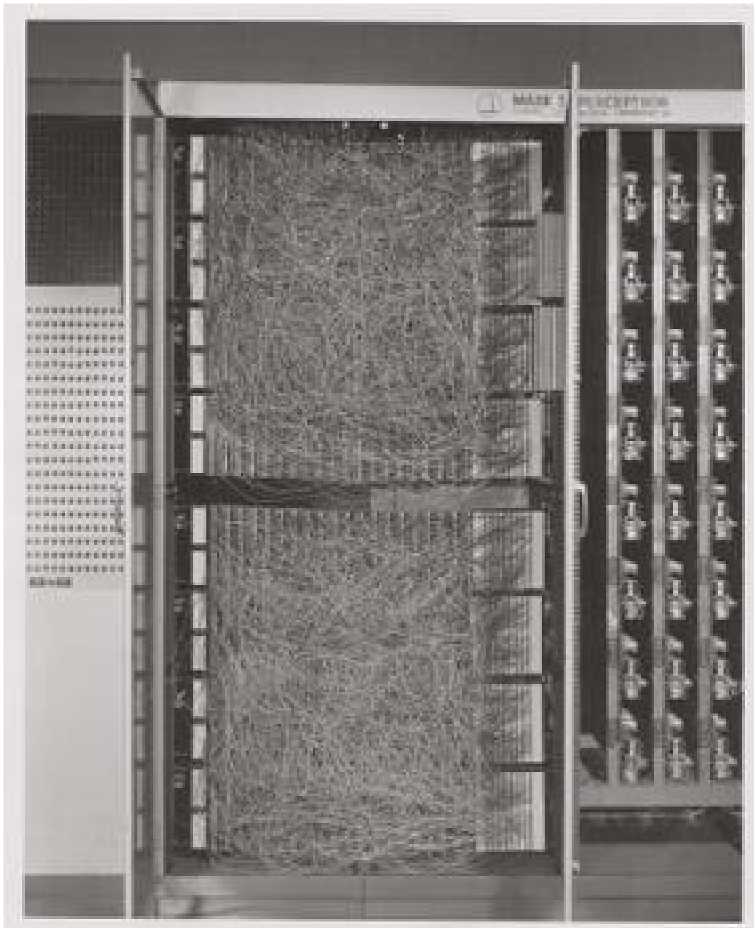

Figure 18.4: The Mark I Perceptron machine, the first implementation of the perceptron algorithm (source: wikipedia)

Chapter 19

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.

19.1 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

19.1.1 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

19.1.2 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)
```

19.2 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

19.3 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.

Chapter 20

Understanding Perceptron Limitations

20.1 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.

20.1.1 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

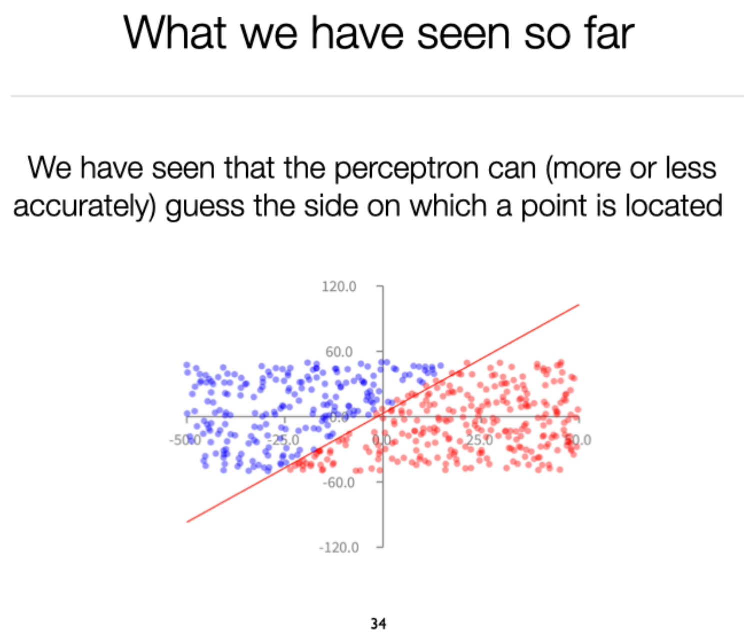

Figure 20.1: Visual representation of XOR problem

20.1.2 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.

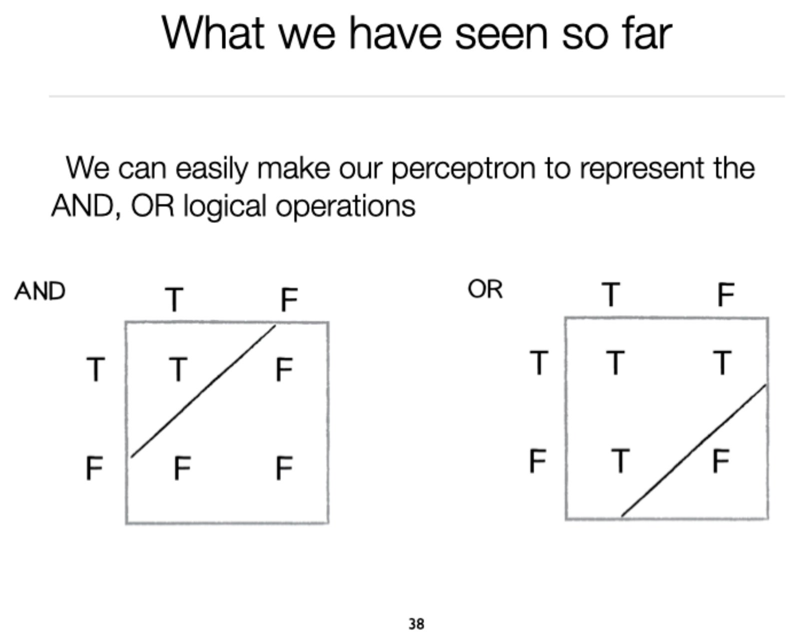

Figure 20.2: Attempted linear separation of XOR

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).

20.2 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!

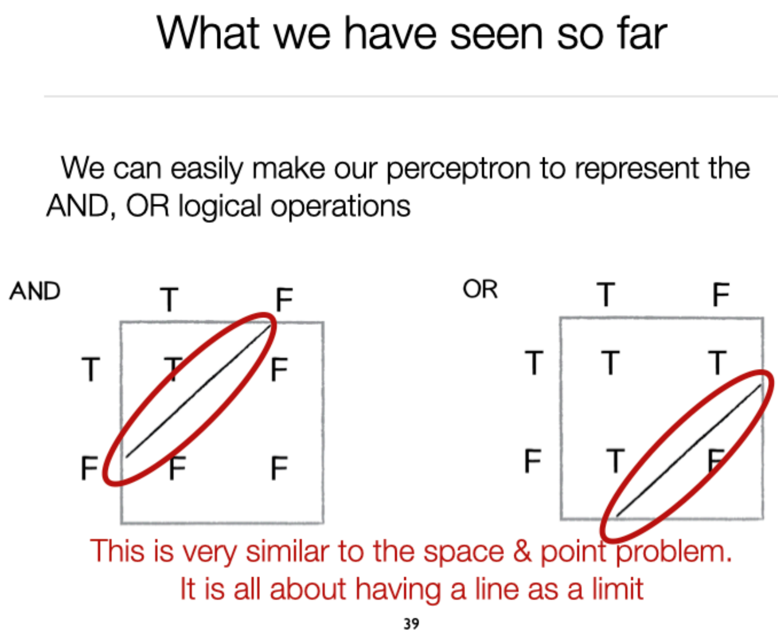

Figure 20.3: Multi-layer solution

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

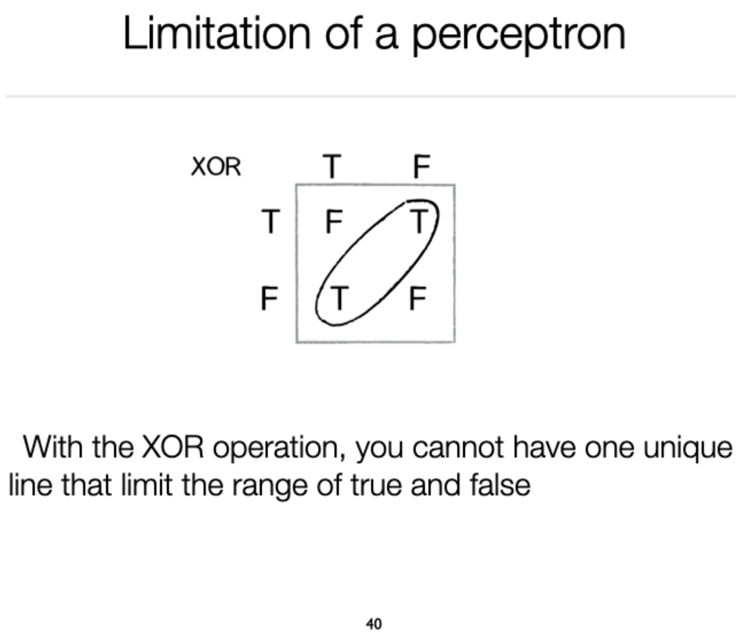

Figure 20.4: Complete neural network solution

20.3 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.

Chapter 21

Introduction to Neural Networks

21.1 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.

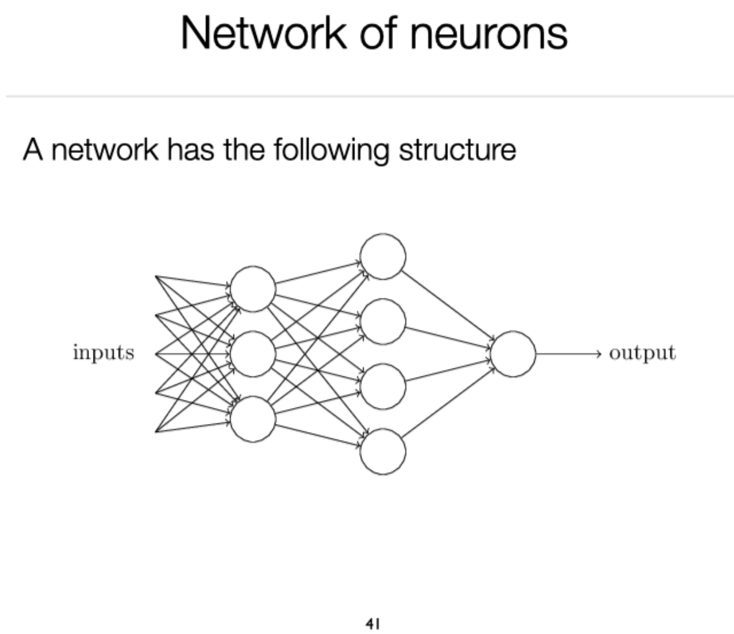

Figure 21.1: Basic neural network architecture

21.2 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

21.2.1 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

21.3 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

21.4 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.

Chapter 22

Practical Example: Classifying Iris Flowers

22.1 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.

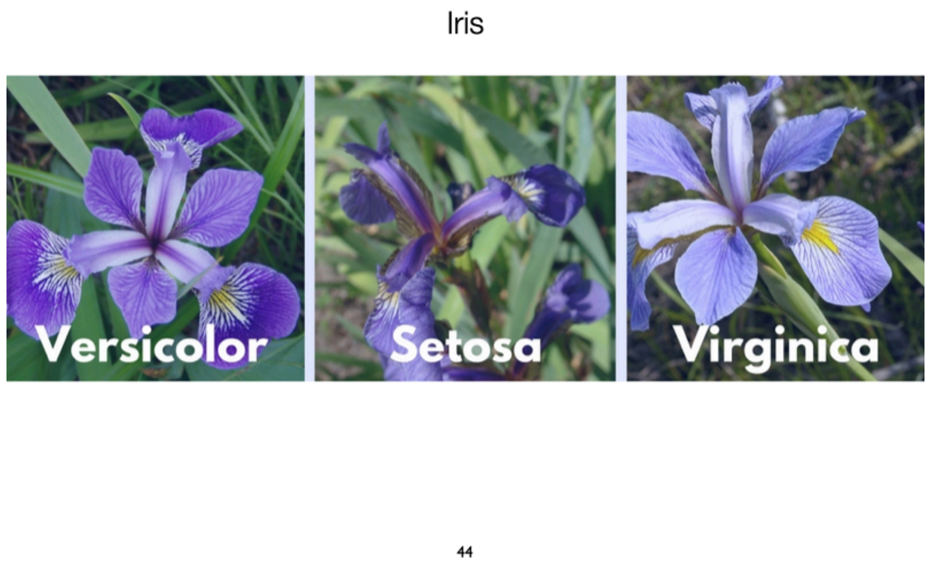

Figure 22.1: Different types of Iris flowers

22.2 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`.

22.3 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.

Chapter 23

The Mathematics Behind Neural Networks

23.1 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.

23.2 The Building Blocks

23.2.1 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: $a = f(z)$

23.2.2 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

23.3 The Learning Process

23.3.1 1. Forward Propagation

Information flows through the network.

23.3.2 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.

Chapter 24

Exploring Neural Network Architectures

24.1 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.

24.2 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

24.3 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

24.4 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

24.5 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

24.6 Autoencoders

Self-learning networks:

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

24.7 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

24.8 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

24.9 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.

Chapter 25

Resources and References AI

25.1 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

25.2 Video Courses and Tutorials

25.2.1 1. Foundational Series

- 3Blue1Brown Neural Networks

  - Visual explanations
  - Mathematical intuition
  - Clear animations

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

25.2.2 2. Programming Tutorials

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

25.2.3 3. Advanced Topics

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

25.3 Online Platforms

25.3.1 1. Interactive Learning

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

25.3.2 2. Research Papers

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

25.3.3 3. Code Repositories

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

25.4 Community Resources

25.4.1 1. Forums and Discussion

- r/MachineLearning
- Cross Validated
- AI Stack Exchange

25.4.2 2. Blogs and Newsletters

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

25.4.3 3. Tools and Libraries

- TensorFlow
- PyTorch
- Scikit-learn

25.5 Academic Papers

25.5.1 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)

Part VII

Experiments

Chapter 26

Experiments

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

Chapter 27

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/

Chapter 28

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.

Chapter 29

MCP Client

- 5ire looks nice.
- oterm