PREBEN THORÖ

Code Evolution Day 2024

Opening talk: Demystifying LLMs

Preben Thorö

- Professional and spare time geek
- CTO Trifork Group
- Leading the GOTO and YOW! Conferences programme work
- Senior Developer

- One of your hosts today

9.15 - 10.00: Introduction + LLMs Are Not Black Magic After All
- Preben Thorö, Trifork

10.15 - 11.00: Introduction to Github Copilot and Github Advanced Security Features
- Karl Krukow, Github

11.15 - 12.00: How to Lead Your Organisation’s AI Transformation: Strategies, Skills, Structure - or How to Skip the Platform Trap and Deliver Business With AI
- Rasmus Lystrøm, Microsoft

12.45 - 13.30: JetBrains IDE Developer Productivity and Code Generation Support
- Garth Gilmour, JetBrains

13.45 - 14.30: Refactoring vs. Refuctoring: Code Quality in the AI Age
- Peter Anderberg & Enys Mones, CodeScene

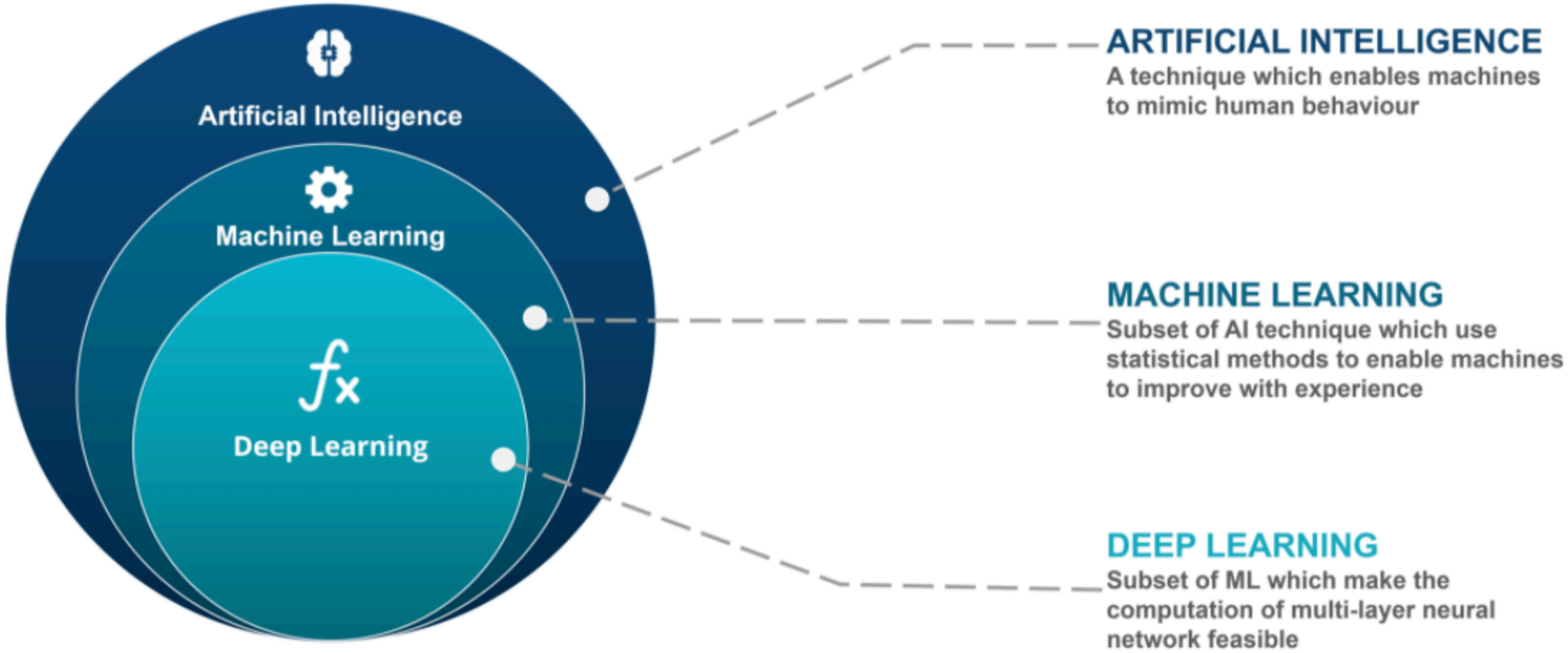
15.00 - 15.30: Considerations About the Governance Impact
- Chresten Plinius, Trifork

15.30 - 16:00: Where To Go From Here
- All

The next 30 mins or so

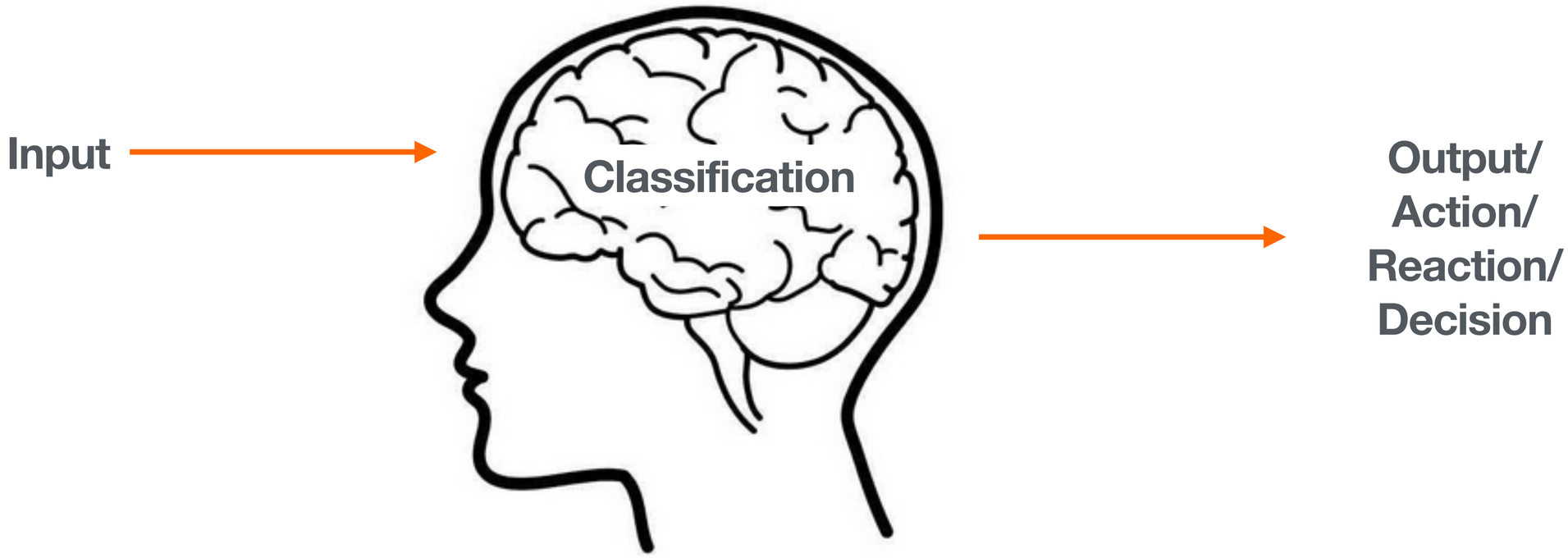
- Basics of neural networks
- Recognising features in images
- Learning text rules
- GPT-x (and ChatGPT)
- Summing up

AI and neural networks/deep learning

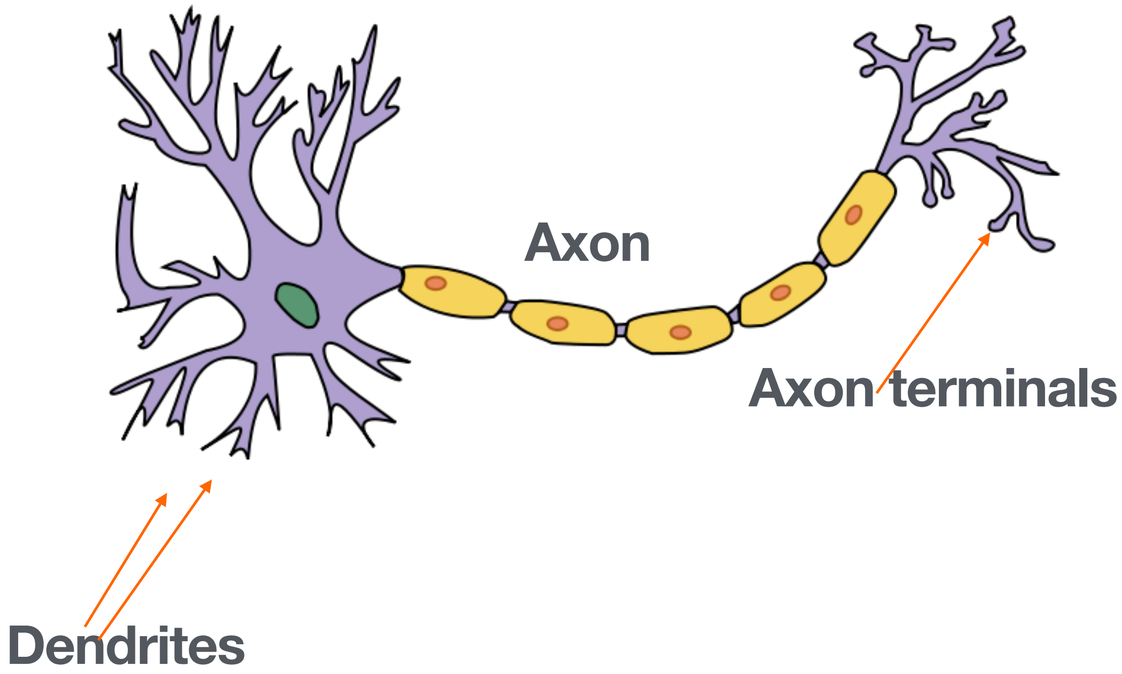


(from https:// www.edureka.co/ blog/ ai-vs-machine-learning-vs-deep-learning/)

AI is an attempt to replicate the brain



The Neurone



(www.verywellmind.com)

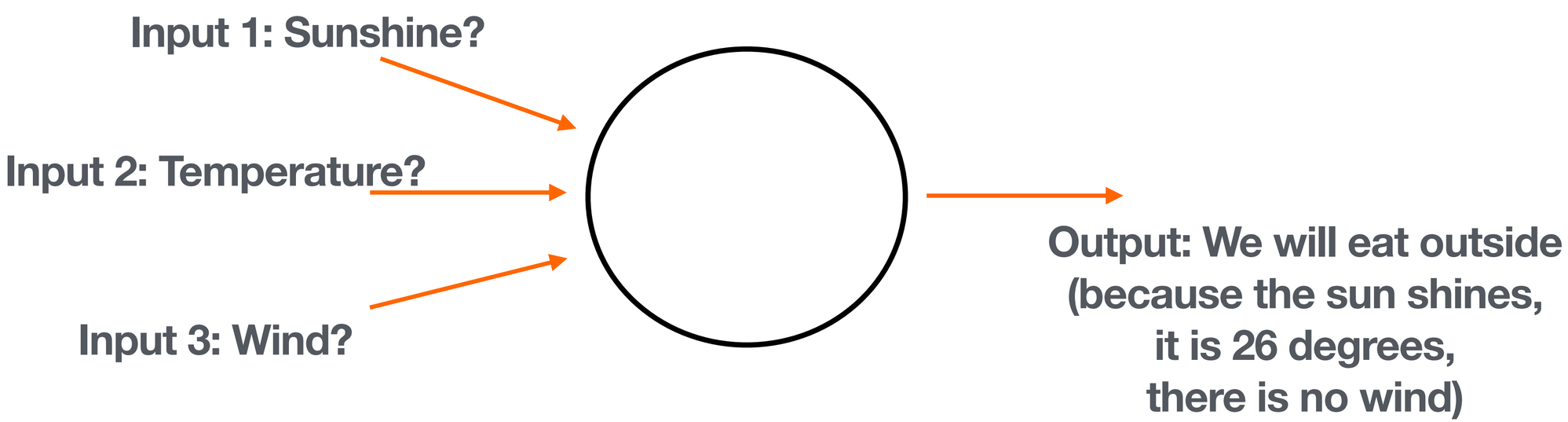
When the inputs are stimulated enough, the outputs will fire

(from wikipedia)

Exercise: Eat outside?

Classification

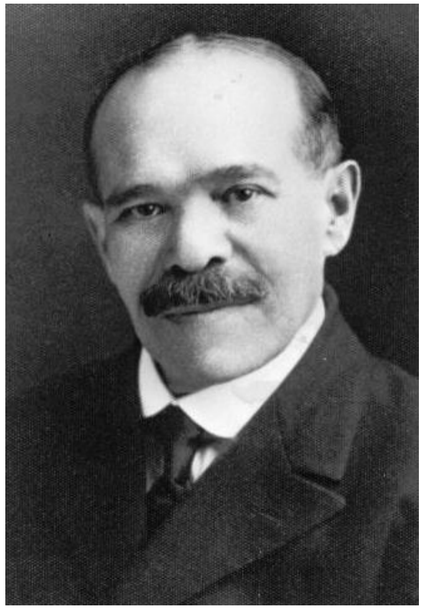
Is the weather so good that we can eat outside?



Please note, the output was based on some kind of interpretation (weights, bias)

Before we continue:
How we perceive the world around us.

Gestalt Psychology

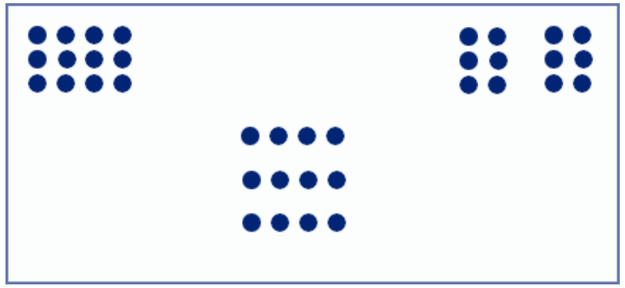


Max
Wertheimer

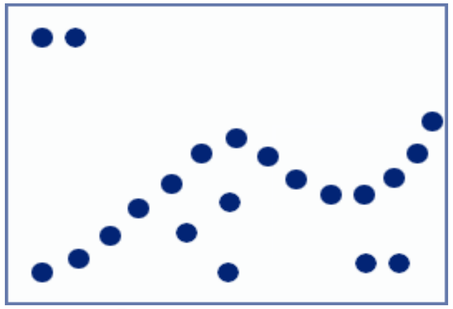
Kurt Kofta

Wolfgang
Kohler

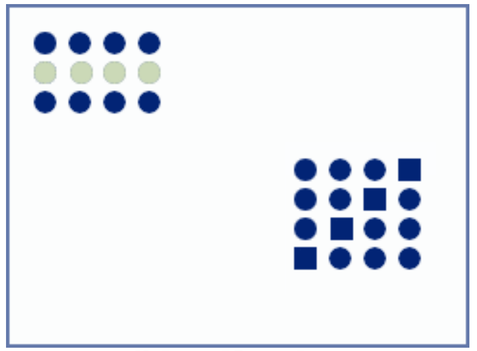
Gestalt Psychology



Proximity



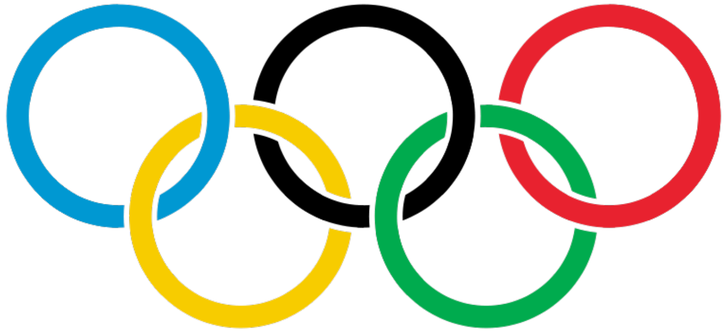
Continuity

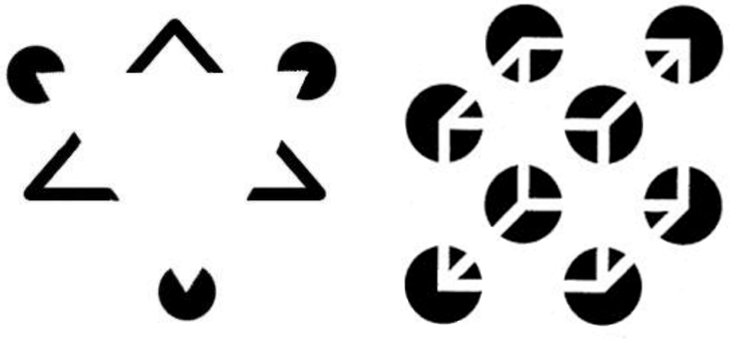


Similarity

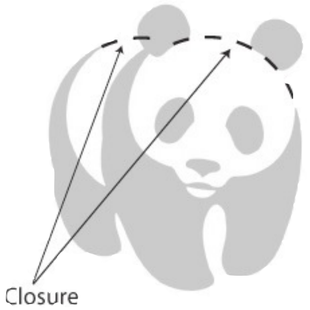
Closure

Gestalt Psychology



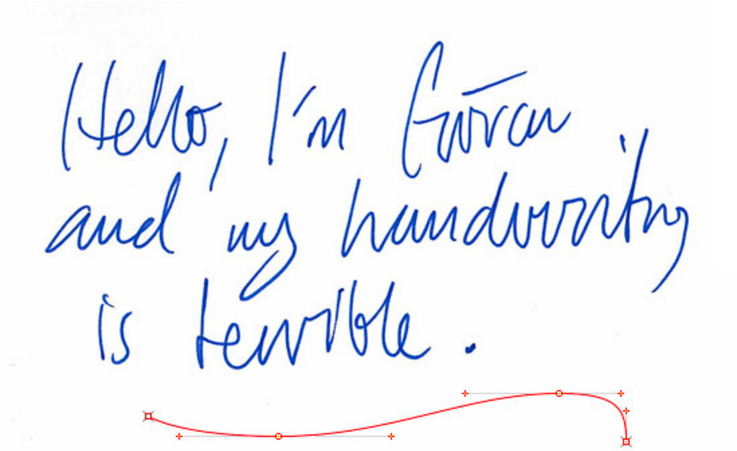


Continuity, closure, proximity and similarity



Gestalt Psychology

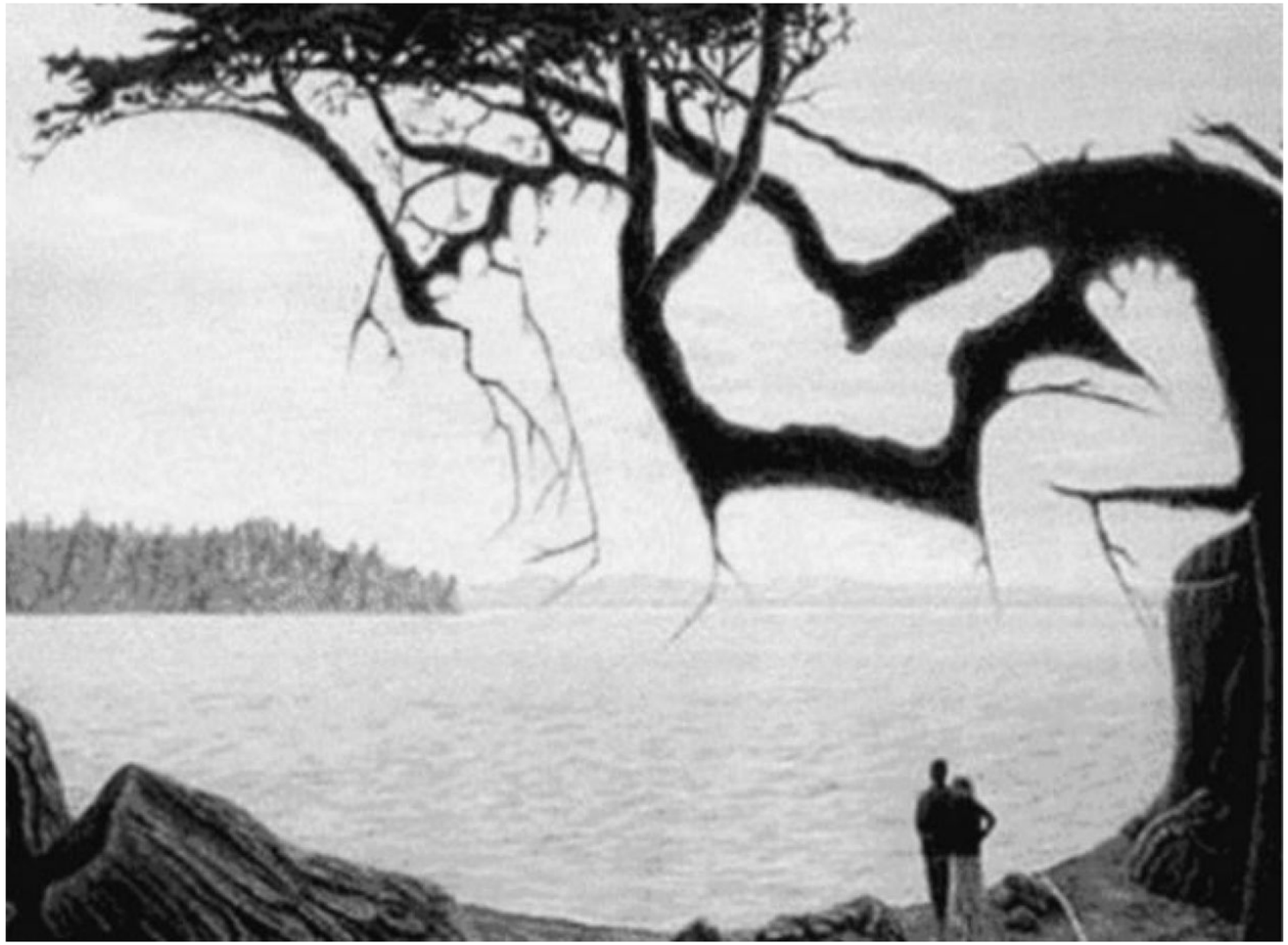
(https://internet.com/website-building)



(http://www.themaninblue.com/)

Gestalt Psychology

Gestalt Psychology



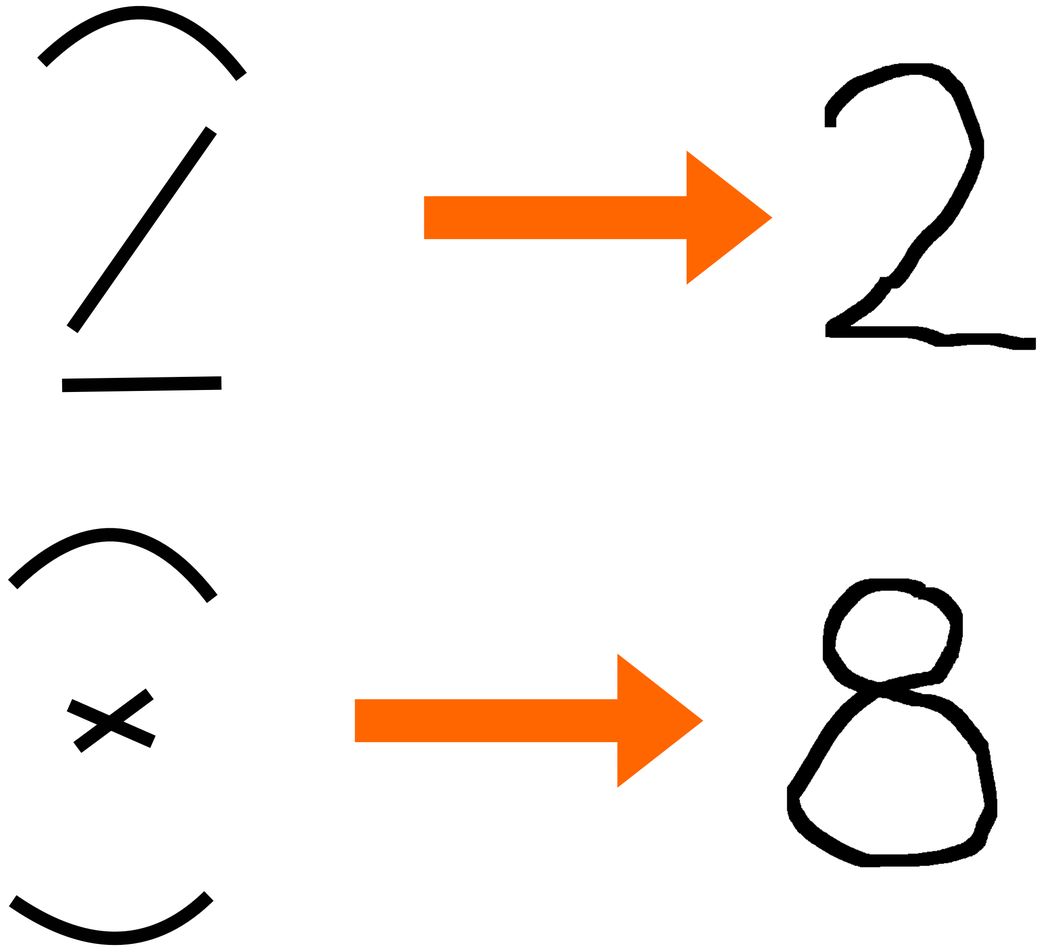
“The whole is other than the sum of the elements”

– Kurt Kofta

(which is not restricted to seeing)

Classification - a slightly more serious example

Inspired by the Gestalt Psychology principles

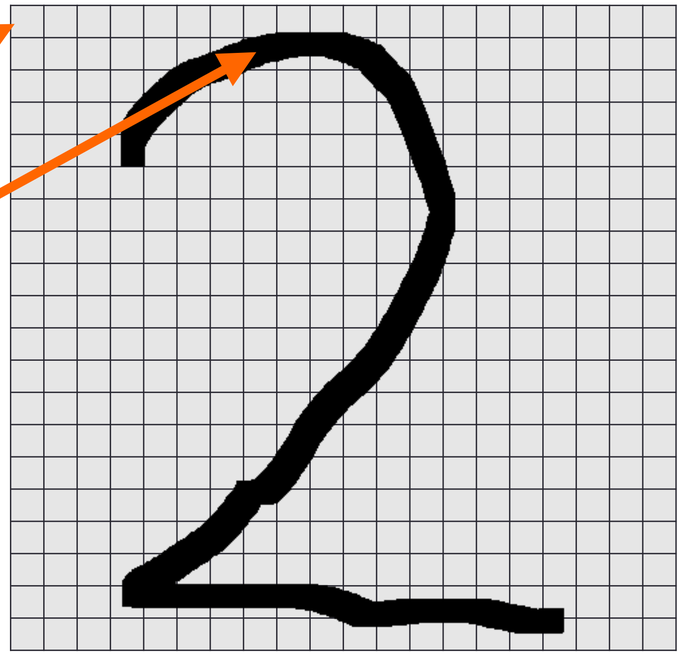


Classification - a slightly more serious example

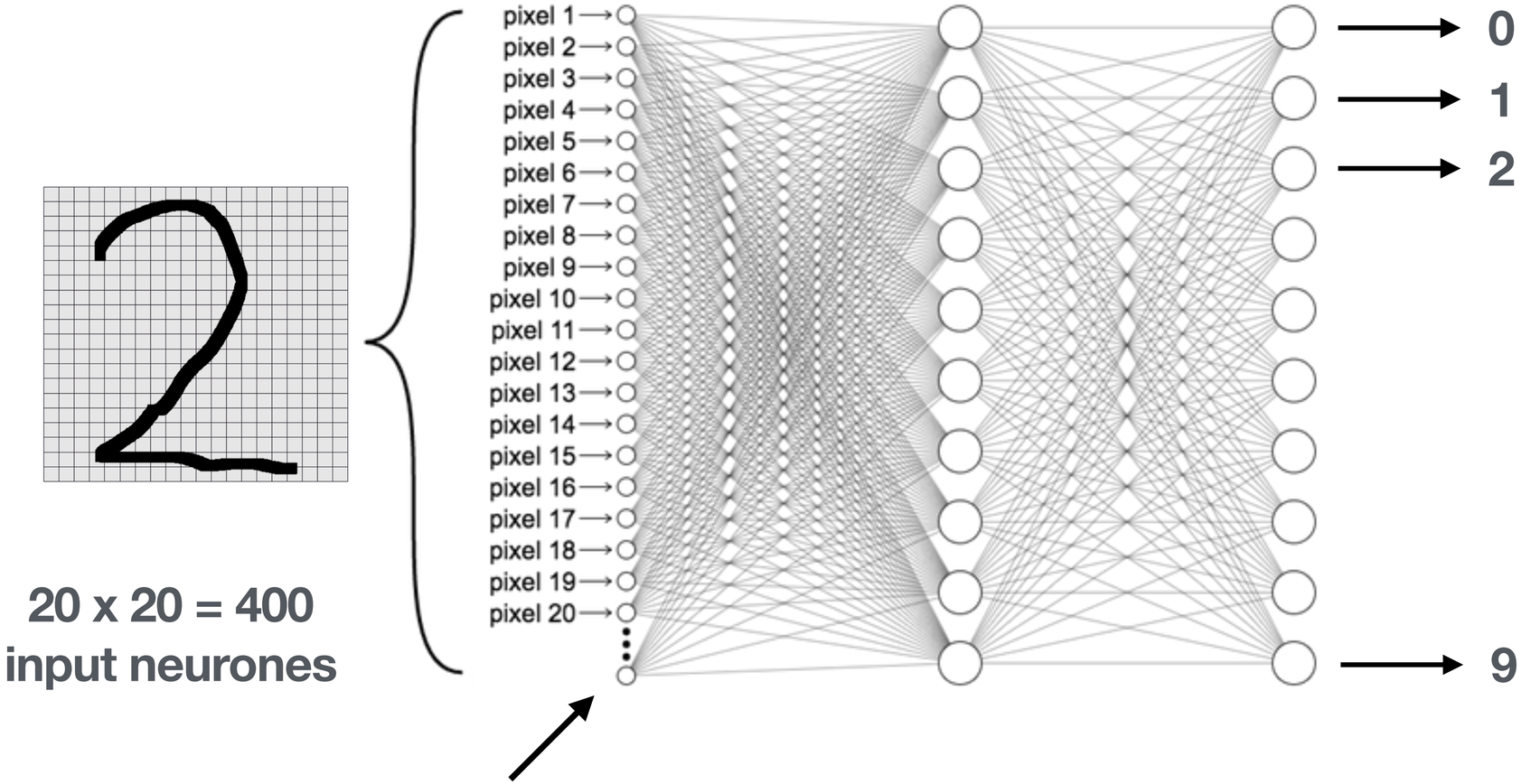
20 x 20 matrix

(A,0) = 0

(H,1) = 0.90

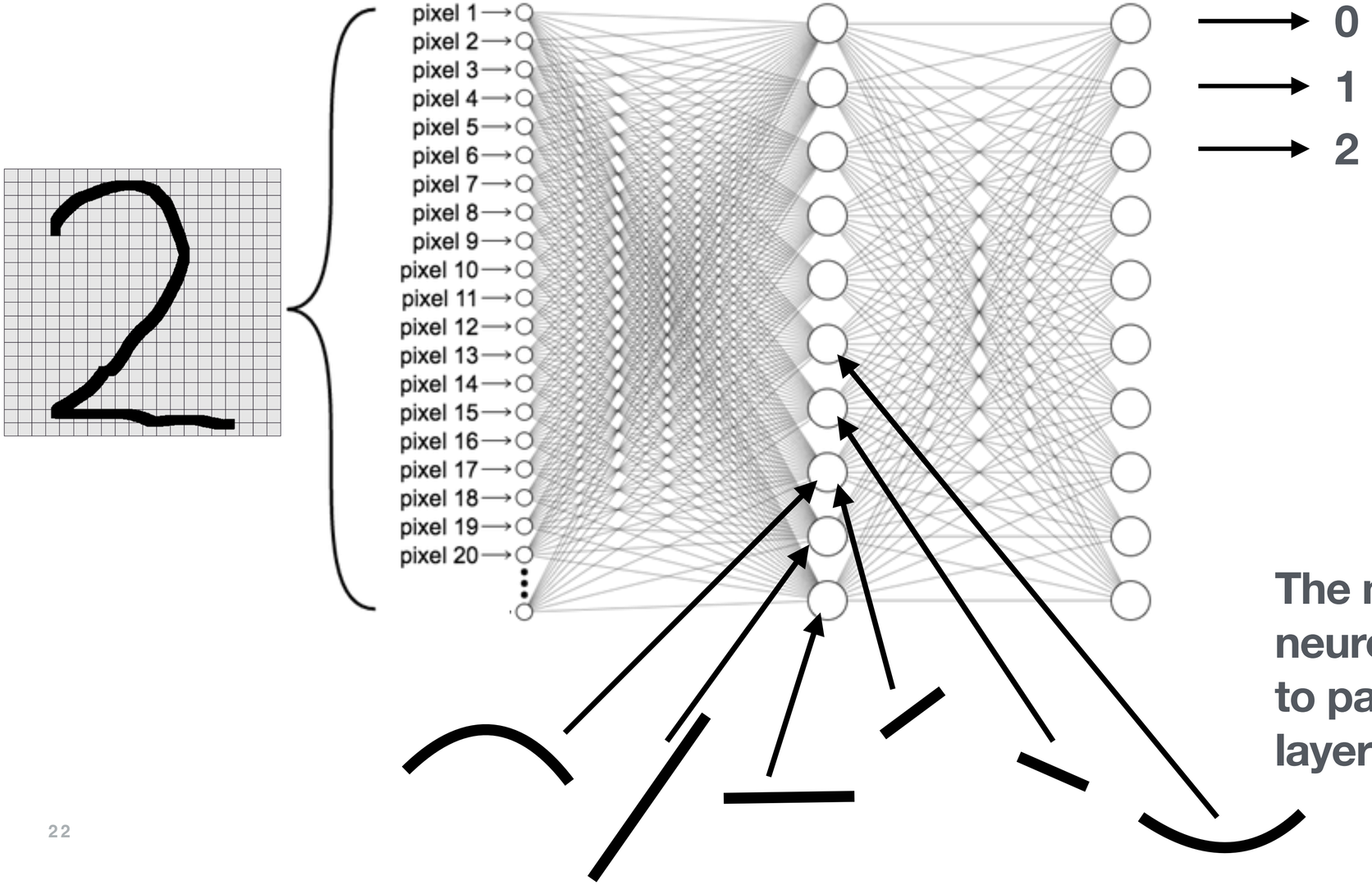


Classification - a slightly more serious example



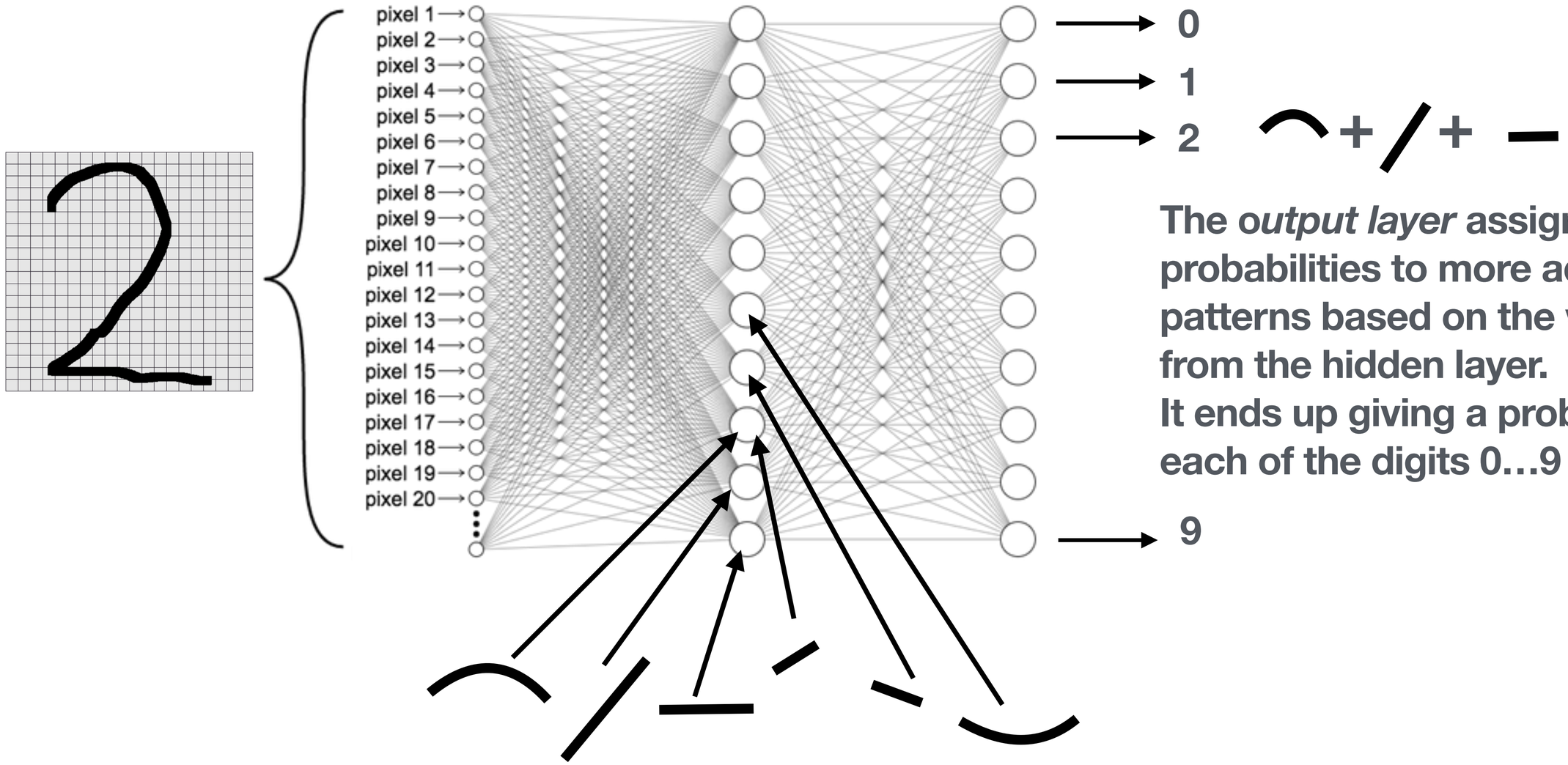
The *input layer* identifies how much of each pixel has been filled out

Classification - a slightly more serious example



The middle layer or *hidden layer* neurones assigns a probability to patterns defined by the input layer values

Classification - a slightly more serious example



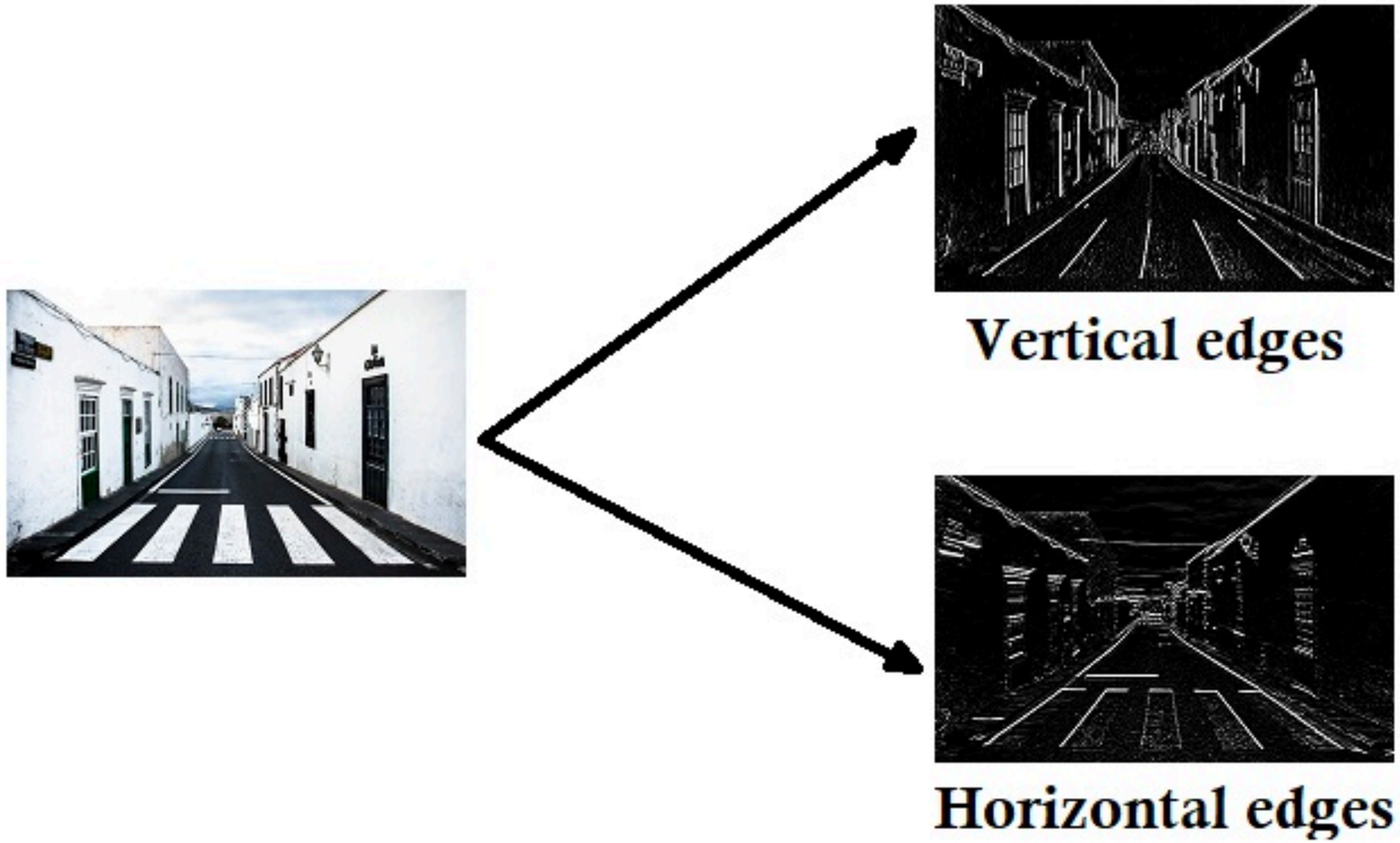
The *output layer* assigns probabilities to more advanced patterns based on the values from the hidden layer.
It ends up giving a probability for each of the digits 0…9

Each individual little building block outputs a certain probability based on the specific inputs it had. These building blocks have all been ordered in ways that allow the next group(s) of blocks to combine the probabilities to an aggregated probability for a more complicated pattern.

This is exactly what we did in the initial weather exercise.
We could have done the handwriting exercise as a similar group exercise too.

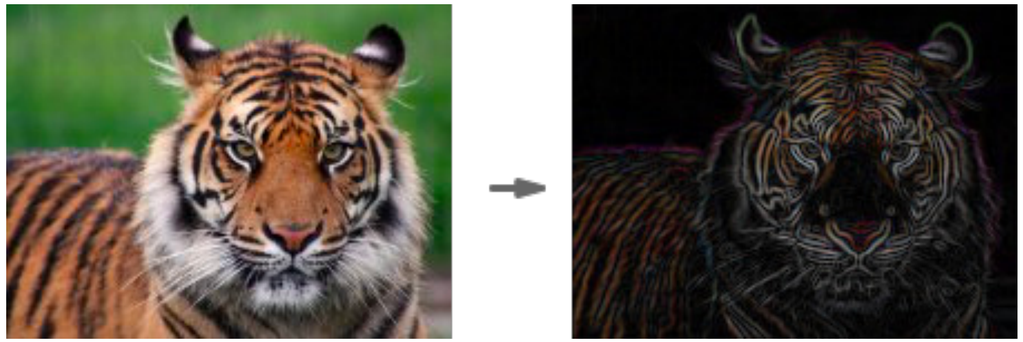
But please note: The structure, number of neurones, groups, etc. is strongly defined by the exact use case.

More Complicated Pictures



(from https://datahacker.rs)

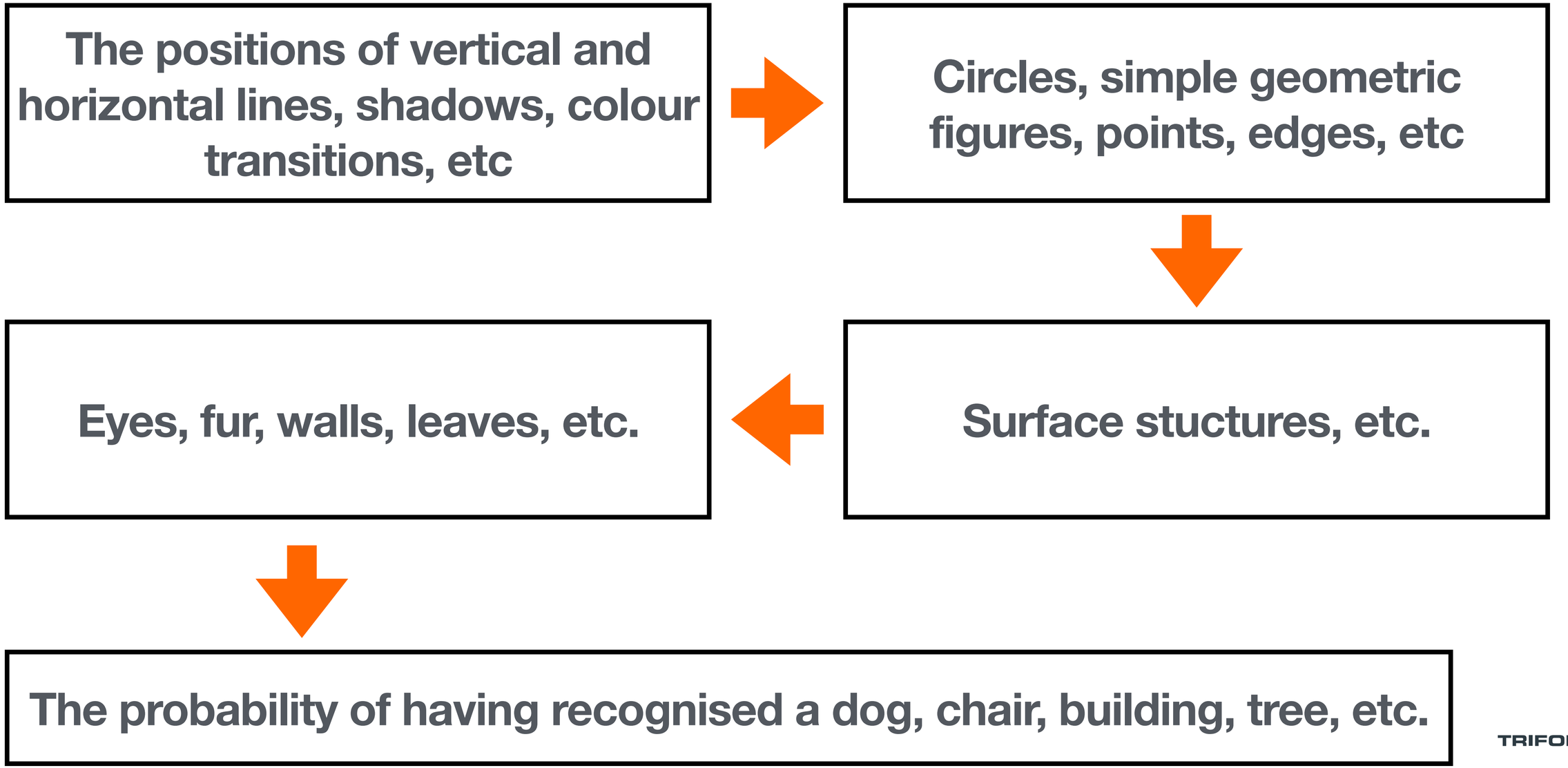
More Complicated Pictures



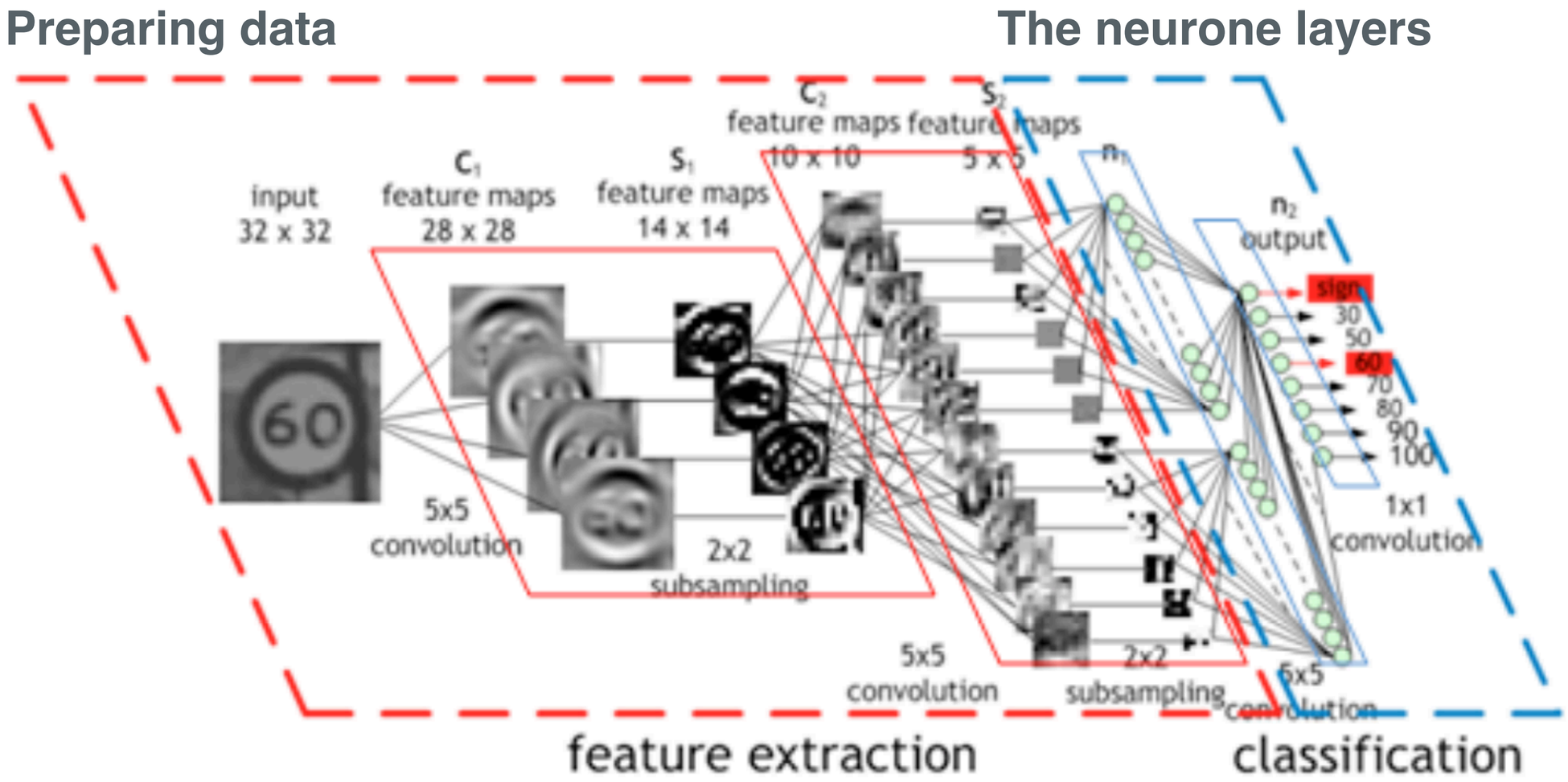
(https://shafeentejani.github.io/)

(https://cambridgespark.com)

More Complicated Pictures



More Complicated Pictures



(from https://devblogs.nvidia.com)

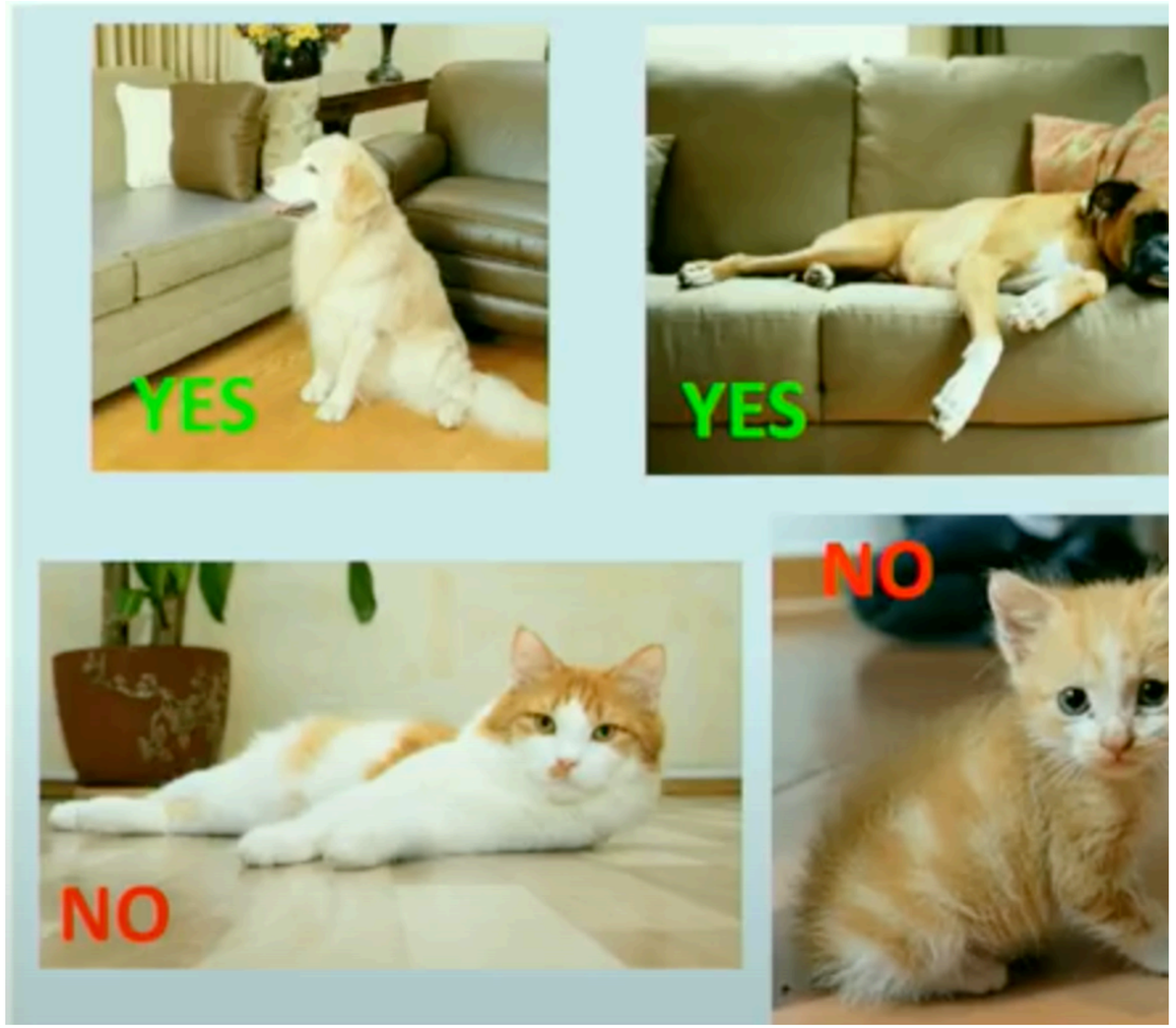
More Complicated Pictures

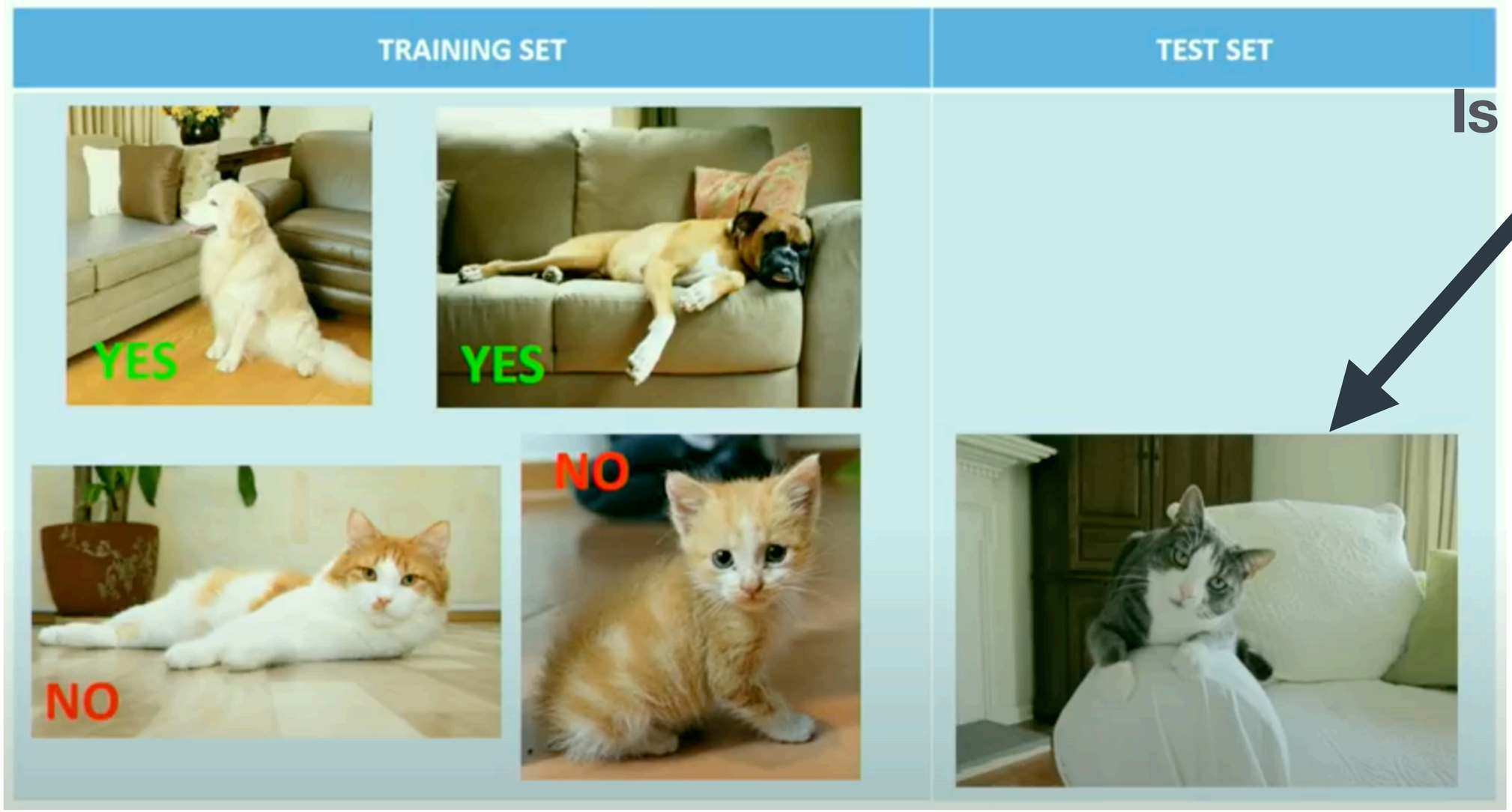
But please help me…there must be an enormous number of neurones and layers.
How do we program the rules?
Remember the weights and biases?
How do we adjust it all?

Answer:

With training. Lots of training!

Doing the correct training - this is a dog




TRAINING SET
TEST SET
YES
YES
NO
NO
Is this a dog?

How do we recognise objects we haven’t seen during the training?

How do we recognise object we haven’t seen during the training?

We don’t! This is not intelligence. The ability to draw associations is 100% absent.

We recognise patterns and nothing else!

The right training…

The Pentagon Tank Case

Preconditions: 100 positives, 100 negaitves.
50 positives + 50 negatives used for training.

The remaining 50 positives and negatives used for verification with a 100% success rate!

Applied into real life the success rate was 0%.

The right training…

The Pentagon Tank Case

Preconditions: 100 positives, 100 negaitves.
50 positives + 50 negatives used for training.

The remaining 50 positives and negatives used for verification with a 100% success rate!

Applied into real life the success rate was 0%.

The system could with almost 100% confidentiality recognise the shadows of the trees in the landscape

But wait…

…if we can recognise patterns in pictures, we might also do so in text…

…which would lead to rules for the order of the words

Which one is correct?

- The quick brown fox jumped over the lazy dog
- The brown quick fox jumped over the lazy dog

Only one of them is syntactically/grammatically correct

Which one is correct?

- The quick brown fox jumped over the lazy dog
- ~~The brown quick fox jumped over the lazy dog~~

Only one of them is syntactically/grammatically correct

Which one is correct?

The correct order of adjectives:

- Quantity or order
- Quality or opinion ← quick
- Size
- Age
- Shape
- Colour ← brown
- Proper adjective (often nationality, other place of origin, or material)
- Purpose or qualifier

Which one is correct?

The correct order of adjectives:

- Quantity or order
- Quality or opinion
- Size
- Age
- Shape
- Colour
- Proper adjective (often nationality, other place of origin, or material)
- Purpose or qualifier

How would you ever solve this using conventional programming?

We assign each word a number/value

1 2 3 4 5 6 1 7 8
The quick brown fox jumped over the lazy dog

9 10 1 11 3
We painted the house brown

- which we use to setup probabilities for the order of the words…given that we use the words again and again (large training set)

So now…

…we should be able to predict the probability of a given word given the context so far…

…which would allow us to construct our own text and sentences

Constructing sentences

He drives

his	0.3
red	0.1
iphone	0.01
walks	0.002
car	0.05

Constructing sentences

He drives his

red	0.3
paper	0.05
tv	0.01
runs	0.002

Constructing sentences

He drives his red

car	0.7
xxx	0.05
xxx	0.01
xxx	0.002

Constructing sentences

He drives his red car

Generative Pre-Trained Transformers

A machine learning (AI) model trained with incredibly huge data sets allowing it to generate relevant and both syntactically and semantically correct sentences.

Generative Pre-Trained Transformers

2018: GPT-1
Training set: Lots of web pages + 11000 books

117 mill parameters

Too limited and not really capable of producing meaningful sentences

Generative Pre-Trained Transformers

2019: GPT-2
Trained on even larger data sets.

1.5 bill parameters.

Much more advanced and realistic sentences but still unable to “understand” the context

Generative Pre-Trained Transformers

2020: GPT-3
Even more training data consisting of web pages, books, wikipedia, articles.
175 bill parameters (100 x GPT-1, 10 x GPT-2)
96 layers of neurones

More advanced and sophisticated language
ChatGPT was based on this

Generative Pre-Trained Transformers

2023: GPT-4

An improvement of GPT-3, more exact details not known

Most likely several 100s of billions, maybe even +1000 billions of parameters

ChatGPT

Advanced software based on GPT-3.5 or 4.0

With a set of filters to prevent biased and politically incorrect output, cannot encourage to violent behaviour, etc.

ChatGPT

The training set is still a significant limitation

No updated knowledge since the training

Still fundamental rules defined by humans (example: Biden - Trump)

It is frightening how realistic it seems.

It refers to itself as “I”

You usually call it “you” and you are tempted to add a “please” to your input.

what does this code line do?

```
my_func() ->
  L = 6,
  <<<<(rand:uniform($9 - $0 + 1) - 1 + $0):8/unsigned-integer>> ||
    _ <- lists:seq(1, L)>>.
```

This code line defines a function named `my_func` and uses list comprehension to generate a binary string of length 6.

Here's a breakdown of what the code does:

- `L = 6` assigns the value 6 to the variable `L`.
- `lists:seq(1, L)` generates a list `[1, 2, 3, 4, 5, 6]`.
- `rand:uniform($9 - $0 + 1) - 1 + $0` generates a random number between 0 and 9 (inclusive) by subtracting the ASCII value of '0' (`$0`) from the ASCII value of '9' (`$9`), adding 1, and then subtracting 1 again to get a number between 0 and 8. Finally, the ASCII value of '0' is added to the result to convert it back to the character representing the number.
- `<<(rand:uniform($9 - $0 + 1) - 1 + $0):8/unsigned-integer>>` generates an 8-bit binary value from the random number, representing an unsigned integer.
- `<<<<(rand:uniform($9 - $0 + 1) - 1 + $0):8/unsigned-integer>> || _ <- lists:seq(1, L)>>` repeats the previous step `L` times (in this case, 6 times), concatenating the generated binary values together using the `||` operator.

In summary, the code generates a binary string of length 6, where each digit is a random number between 0 and 9.

This code line defines a function named `my_func` and uses list comprehension to generate a binary string of length 6.

Here's a breakdown of what the code does:

- `L = 6` assigns the value 6 to the variable `L`.
- `lists:seq(1, L)` generates a list `[1, 2, 3, 4, 5, 6]`.
- `rand:uniform($9 - $0 + 1) - 1 + $0` generates a random number between 0 and 9 (inclusive) by subtracting the ASCII value of '0' (`$0`) from the ASCII value of '9' (`$9`), adding 1, and then subtracting 1 again to get a number between 0 and 8. Finally, the ASCII value of '0' is added to the result to convert it back to the character representing the number.
- `<<(rand:uniform($9 - $0 + 1) - 1 + $0):8/unsigned-integer>>` generates an 8-bit binary value from the random number, representing an unsigned integer.
- `<<<<(rand:uniform($9 - $0 + 1) - 1 + $0):8/unsigned-integer>> || _ <- lists:seq(1, L)>>` repeats the previous step `L` times (in this case, 6 times), concatenating the generated binary values together using the `||` operator.

PT

Can you make an elixir equivalent of it?

Certainly! Here's the equivalent code in Elixir:

elixir

Copy code

```elixir
def my_func do
  l = 6
  Enum.map(1..l, fn _ ->
    <<(Enum.random(0..9) + ?0)::size(8)>>
```

```
  end)
end
```

In Elixir, the `Enum.map/2` function is used to iterate over the range `1..1` and generate the binary strings. Inside the anonymous function, `Enum.random/1` is used to generate a random number between 0 and 9, and then `?0` is added to convert it to the character representing the number. Finally, `<<>>` is used to generate the binary string of size 8.

The resulting list of binary strings is returned as the result of the `my_func` function.

LISP

```
(defun my-func ()
  (let* ((l 6)
         (random-digit (lambda () (char (random 10) #\0))))
    (loop repeat l collect (make-array 8 :element-type 'unsigned-byte
                                         :initial-contents (loop
repeat 8 collect (funcall random-digit))))))
```

Fortran

```
program my_func
  implicit none

  integer, parameter :: l = 6
  character(8) :: binary_string
  integer :: i

  call random_seed()
  do i = 1, l
     call random_number(binary_string)
     binary_string = char(int(binary_string*10, kind=char),
kind=char)
     write(*, '(a)') binary_string
  end do

end program my_func
```

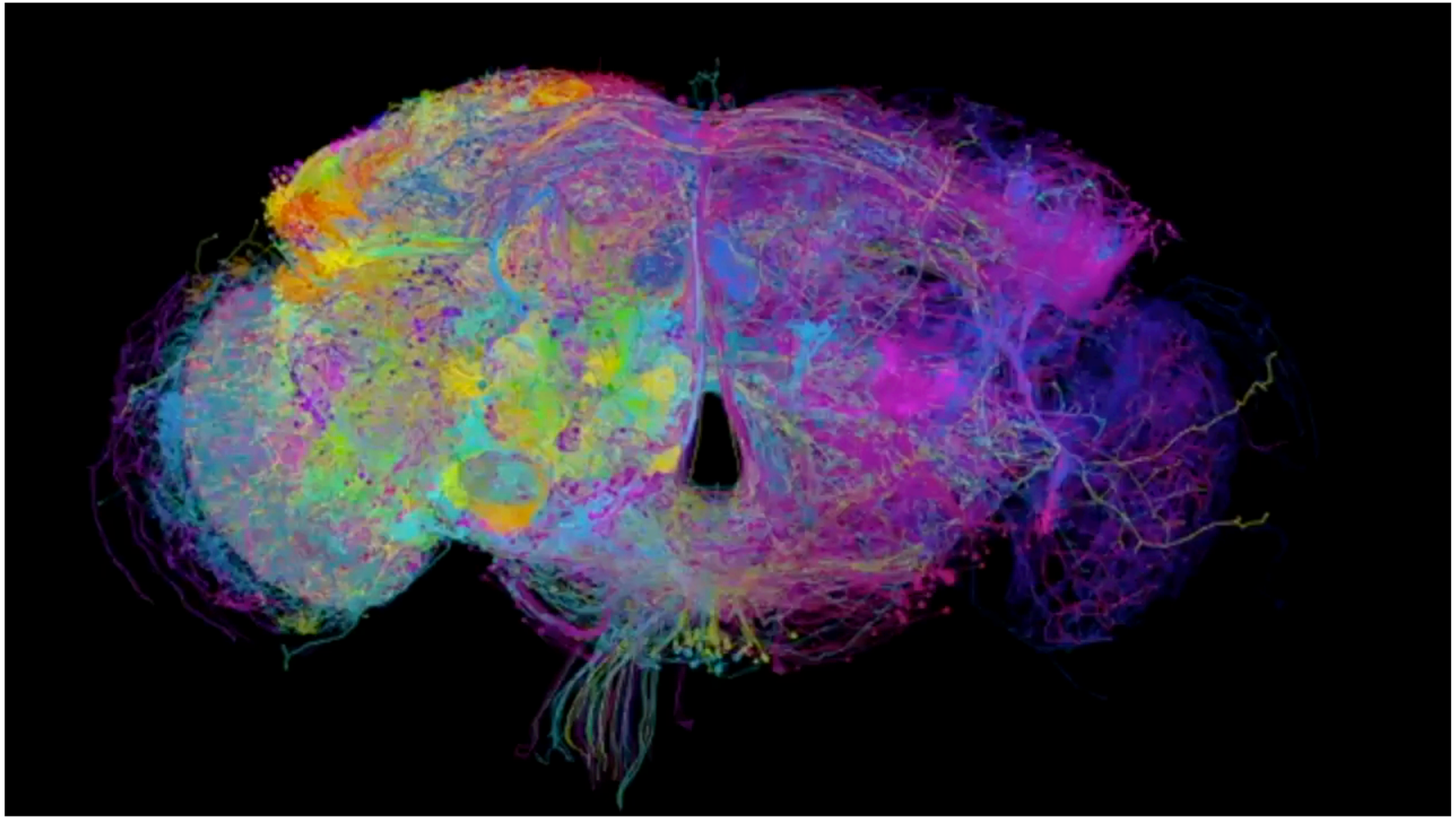
6502 Assembler

```
my_func:
    LDA #$06          ; Set L to 6
    STA L
    LDY #0            ; Initialize Y as the counter
    JSR RAND          ; Generate a random number between 0 and 9

generate_string:
    LDA RAND_RESULT ; Load the random number
    ADC #'0'          ; Add the ASCII value of '0' to convert to character
    STA (RESULT_PTR),Y  ; Store the character in the result string
    INY               ; Increment the counter
    CPY L             ; Compare the counter with L
    BNE generate_string ; If not equal, continue generating the string

    RTS

RAND:
    ; Implement your random number generation routine here
    ; Store the result in the RAND_RESULT variable
    ; Ensure that the result is between 0 and 9
```



A fruit fly brain with its 100000 neurones (from Youtube). The human brain has estimated 100 bill neurones.

AI is pattern recognition

Our brain:

- 100 bill neurones
- Each of them is connected to 100s, 1000s of other neurones
- Neurones not ordered in layers as we do in our solutions
- The neurone can fire in 26 different ways
- Connections can come and go
- The transmission time corresponds to a clock freq. of 200 Hz
- The total effect equals 20w

AI is pattern recognition

Most likely limiting the brain size

The brain is an incredible 20w computer ticking at 0,0000002 GHz, with millions of cores and transmission patterns in more dimensions

It is estimated that a computer consumes around 50 mill times more energy than the brain to solve simple tasks

Imagine

If GPT had been trained on code

Imagine

If GPT had been trained on code

Karl, please take it from here…

Thank you

TRIFORK.