Machine Learning and Neural Networks

An Introduction

Dr. Stefan Siegert
Mathematics Department
University of Exeter

University *of* Exeter

Virtual Afternoon Training Event, 21 October 2022

Slides and other material are available at:

`github.com/sieste/virtual-training-event-2022`

Goals of this session

- Understand fundamental concepts and ideas through a few concrete examples
- Demystify neural networks and explain the underlying maths
- Provide specific code examples for further learning

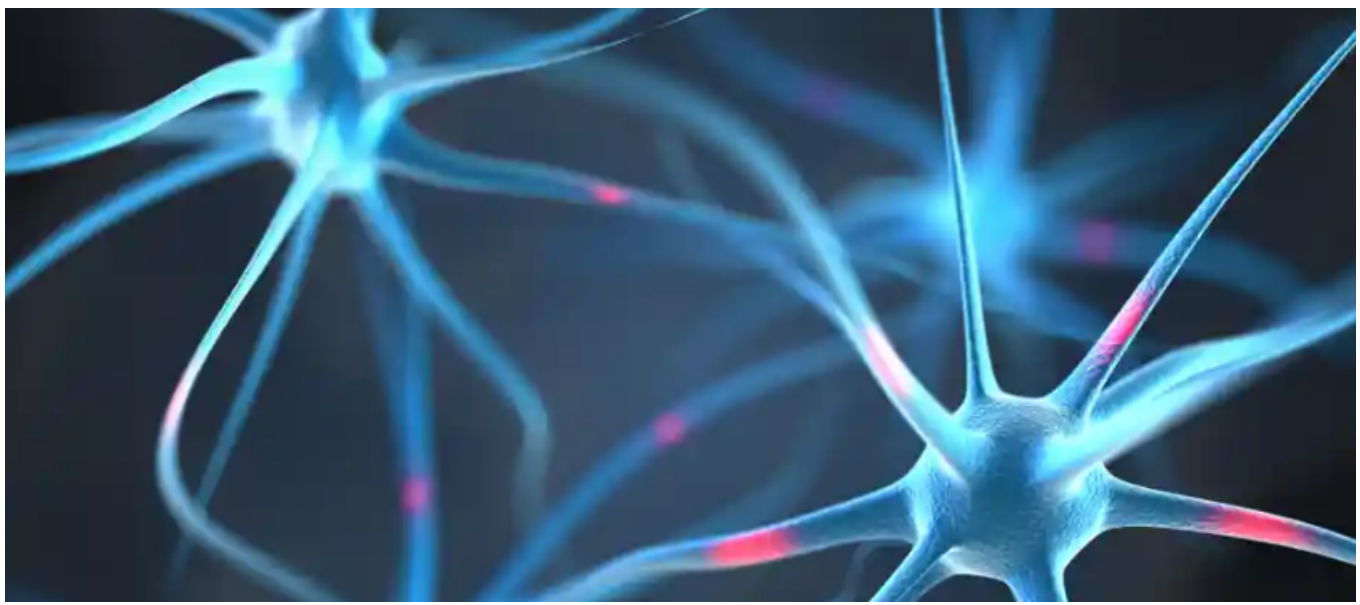
Structure

1. Early approaches, Hopfield Network
2. Unsupervised Learning, Hierarchical Clustering
3. Supervised Learning, Regression
4. Modern technology, Convolutional Neural Network

Hebbian Learning

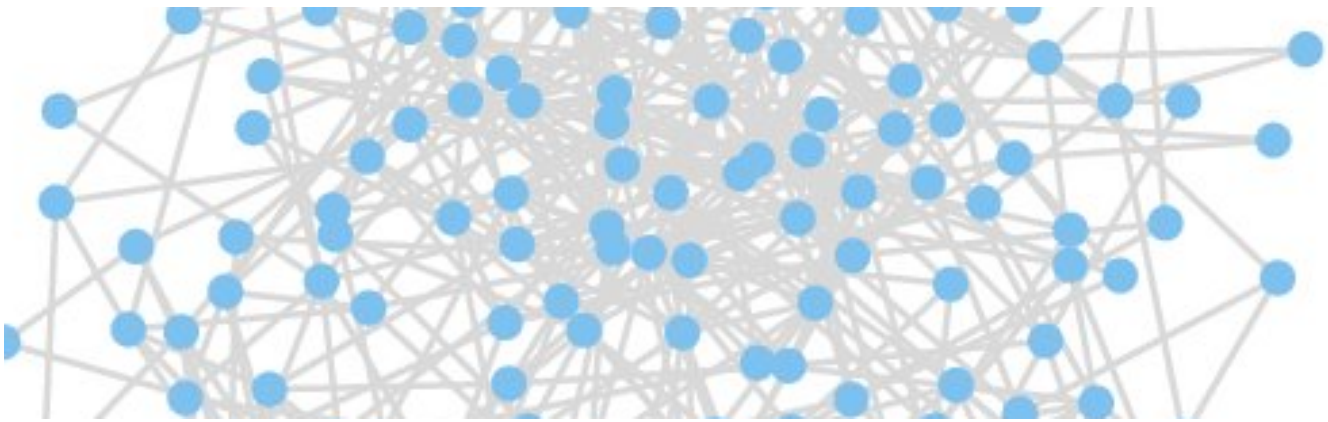
Donald Hebb (1949) in the book "The Organization of Behavior" formulated a reductionist model of how learning occurs in the brain, which is often summarised as

Cells that fire together wire together.



Hopfield network

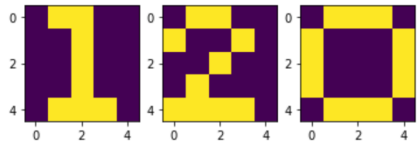
Hopfield (1984) translated Hebbian learning into a **simple mathematical model** of a network that exhibits **learning and associative memory**.



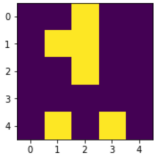
Hopfield network capabilities

The Hopfield network consists of N neurons $V_1, V_2, \ldots, V_N$, each of which can fire ($V_i = +1$) or not fire ($V_i = -1$).

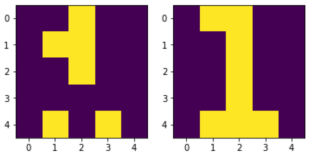
The network is "trained" on a small number of examples, here $N = 25$ pixels of an image:

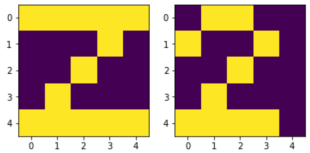


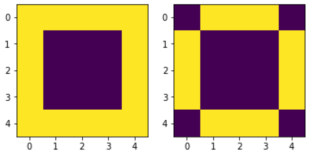
After training, the network can be presented with a new, yet unseen example



and "recalls" the similar looking image it has seen previously







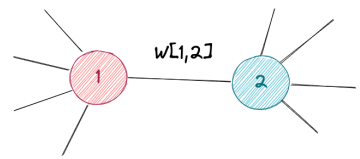
Hopfield network details: Training

The connection strength between neuron i and neuron j is denoted by $W_{i,j}$. Initially (before training) all connection strengths are zero:

$$W_{i,j} = 0 \text{ for all } i, j.$$

An image is "shown" and creates a firing pattern in the neurons, e.g.

$$(V_1, V_2, \ldots, V_{25}) = (1, -1, \ldots, 1).$$



As a result, the $W_{i,j}$ are updated as follows

$$W_{i,j} \to W_{i,j} + V_i \times V_j$$

If V_i and V_j are doing the same thing, either $(1, 1)$ or $(-1, -1)$, the corresponding $W_{i,j}$ is increased by one. Otherwise it is decreased by one.

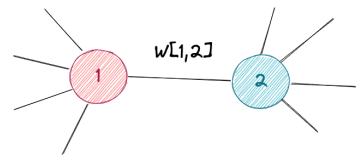
The process is repeated with more input images, leading to further increases/decreases of the $W_{i,j}$'s.

Hopfield network details: Recall

Let's now expose the network to a new "stimulus", given by an input sequence

$$(V_1^*, V_2^*, \ldots, V_N^*) = (1, 1, \ldots, -1).$$

Assume neuron two fires. Should neuron one fire as well?



If, in training, neurons 1 and 2 always fired together, $W_{1,2}$ is large. Neuron one should fire.

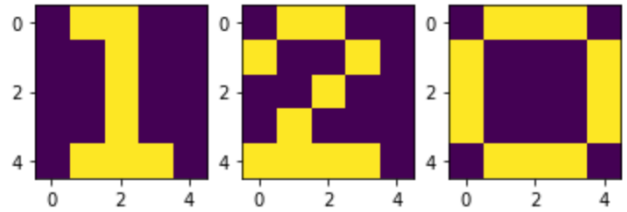
So the excitation of neuron one from neuron two can be calculated by $W_{1,2} \times V_2^*$. The total excitation of neuron one from all the inputs combined is

$$W_{1,2} V_2^* + W_{1,3} V_3^* + \cdots + W_{1,25} V_{25}^*$$

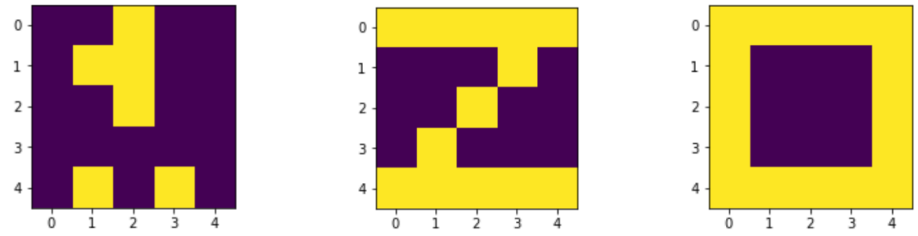
We repeat that process for all other neurons.

Hopfield network recall in action

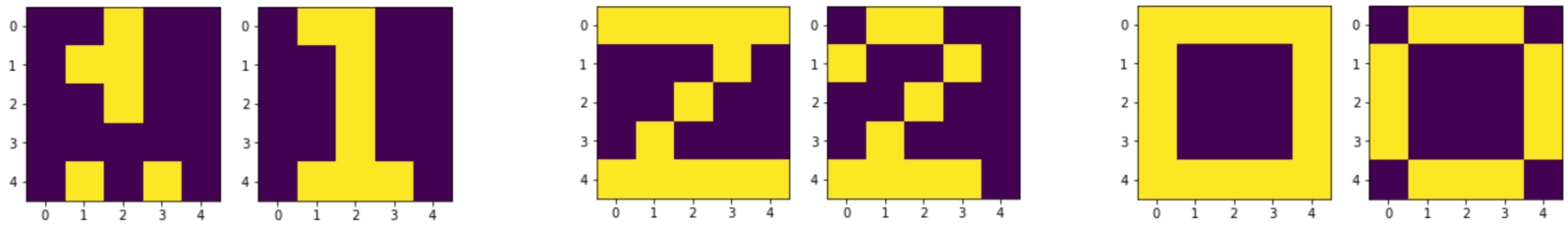
Training:



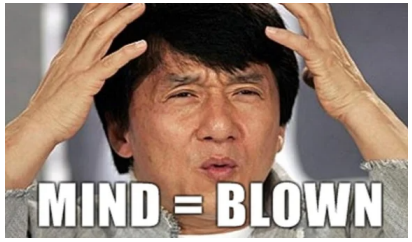
Recall input patterns:



Recall output patterns:



Hopfield network: Bigger implications



A biological principle is translated into a simple mathematical model, which mimics behavior of complex organisms (learning and associative recall).

The Hopfield network exhibits learning by strengthening network weights.

Successful learning is demonstrated during recall, where arbitrary input patterns are successfully matched to previously seen patterns.

Unsupervised learning

Unsupervised Learning is a class of machine learning algorithms that learn patterns in unlabelled data, without being told what's correct or wrong.

The learnt patterns are used to group, classify, associate new inputs, or generate ("imagine") entirely new data.

Applications of unsupervised learning:

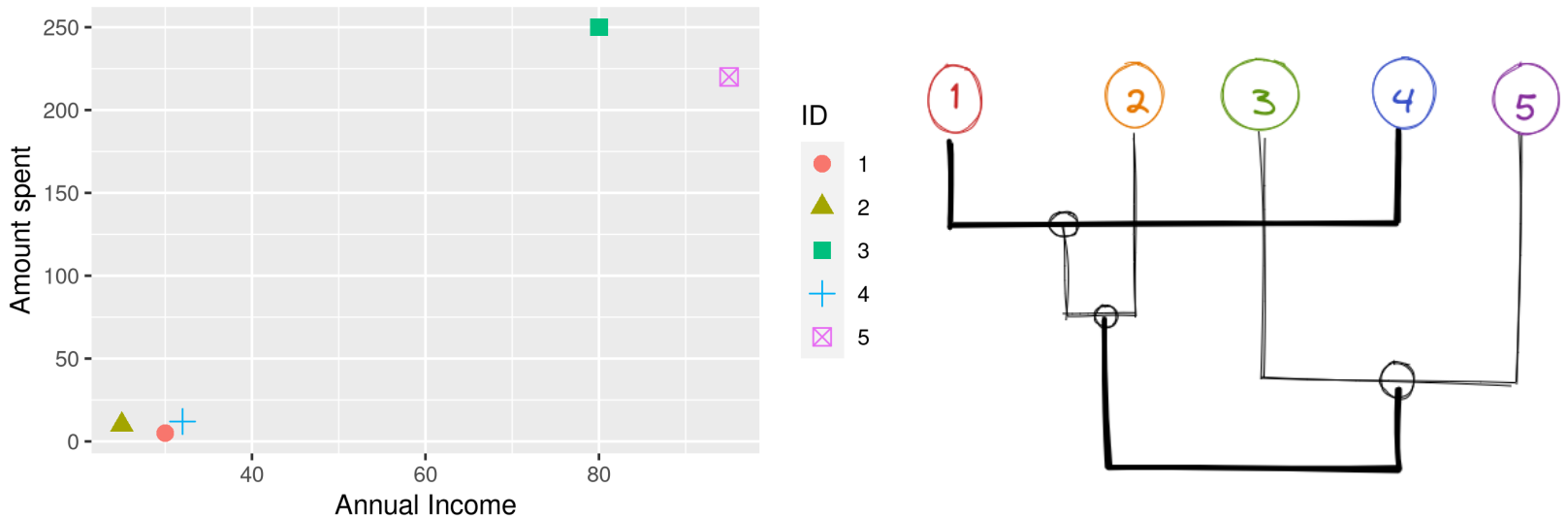
- Recommendation system

- Outlier detection ("That credit card transaction is different from anything we've seen before ... Investigate!")
- Image Generation (Style transfer)

- Customer segmentation

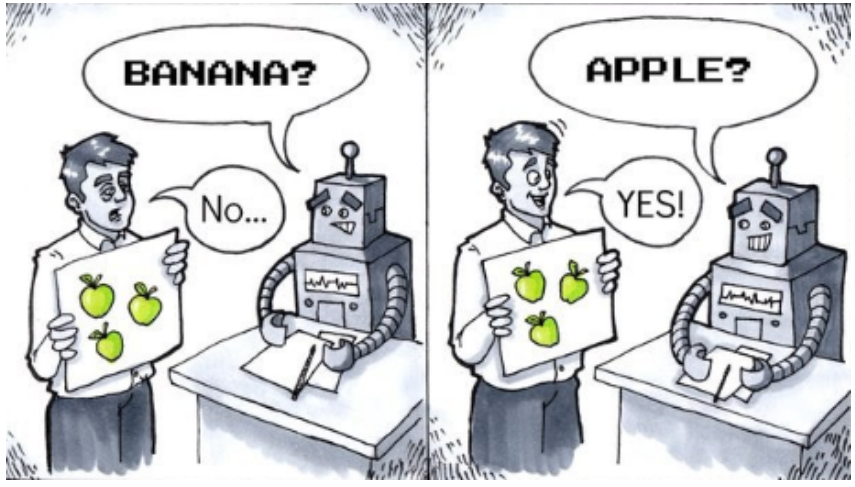
Customer segmentation by hierarchical clustering

ID	Annual Income (k£)	Amount spent (£)	...
1	30	5	...
2	25	10	...
3	80	250	...
4	32	12	...
5	110	150	...
6	...	...	...



Hierarchical clustering: Each unit starts as their own "cluster", and the nearest (most similar) units are successively combined into bigger and bigger clusters.

Supervised Learning



In contrast to unsupervised learning, SL works on labelled input-output pairs. A number of examples of inputs is available for learning, in combination with their expected outputs.

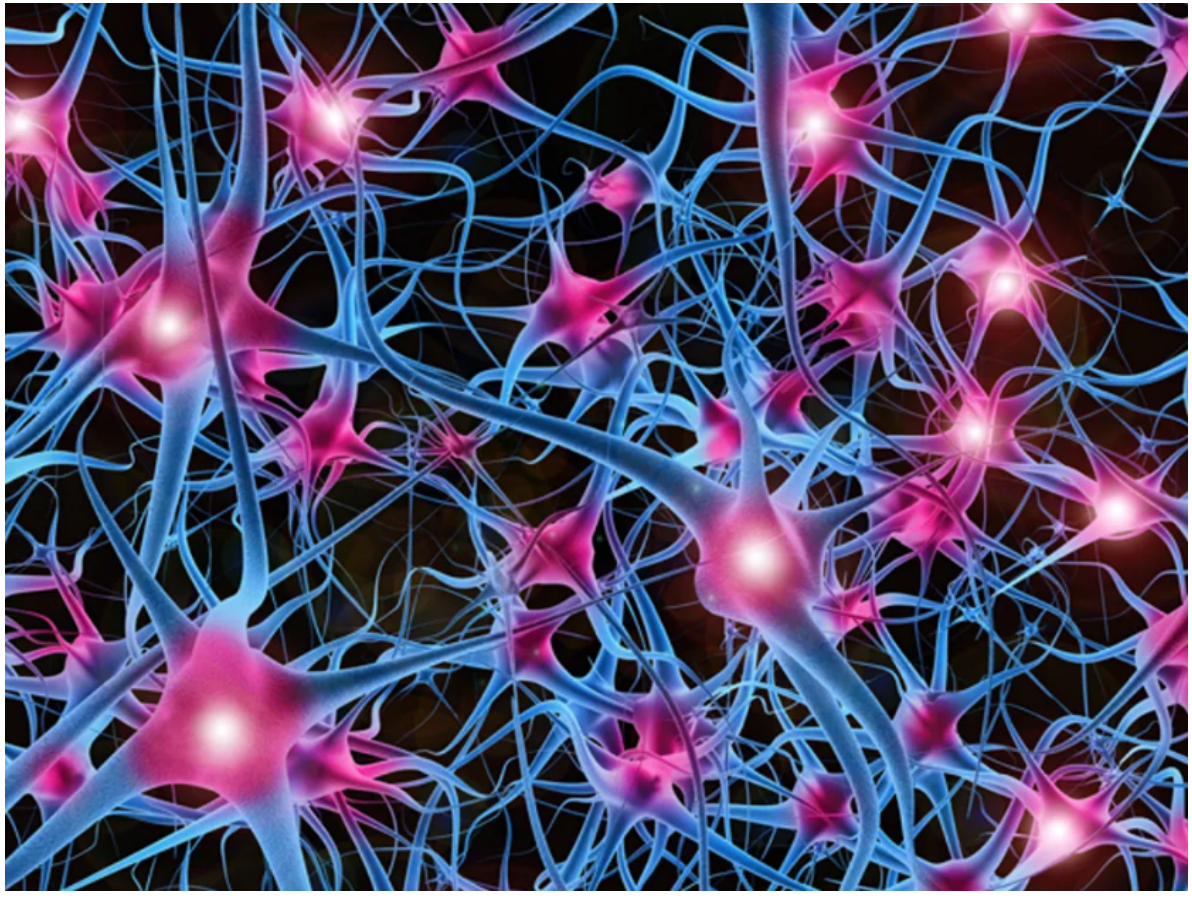
Linear regression

A classic and simple example of supervised learning is linear regression, where inputs and outputs are related by a straight line.

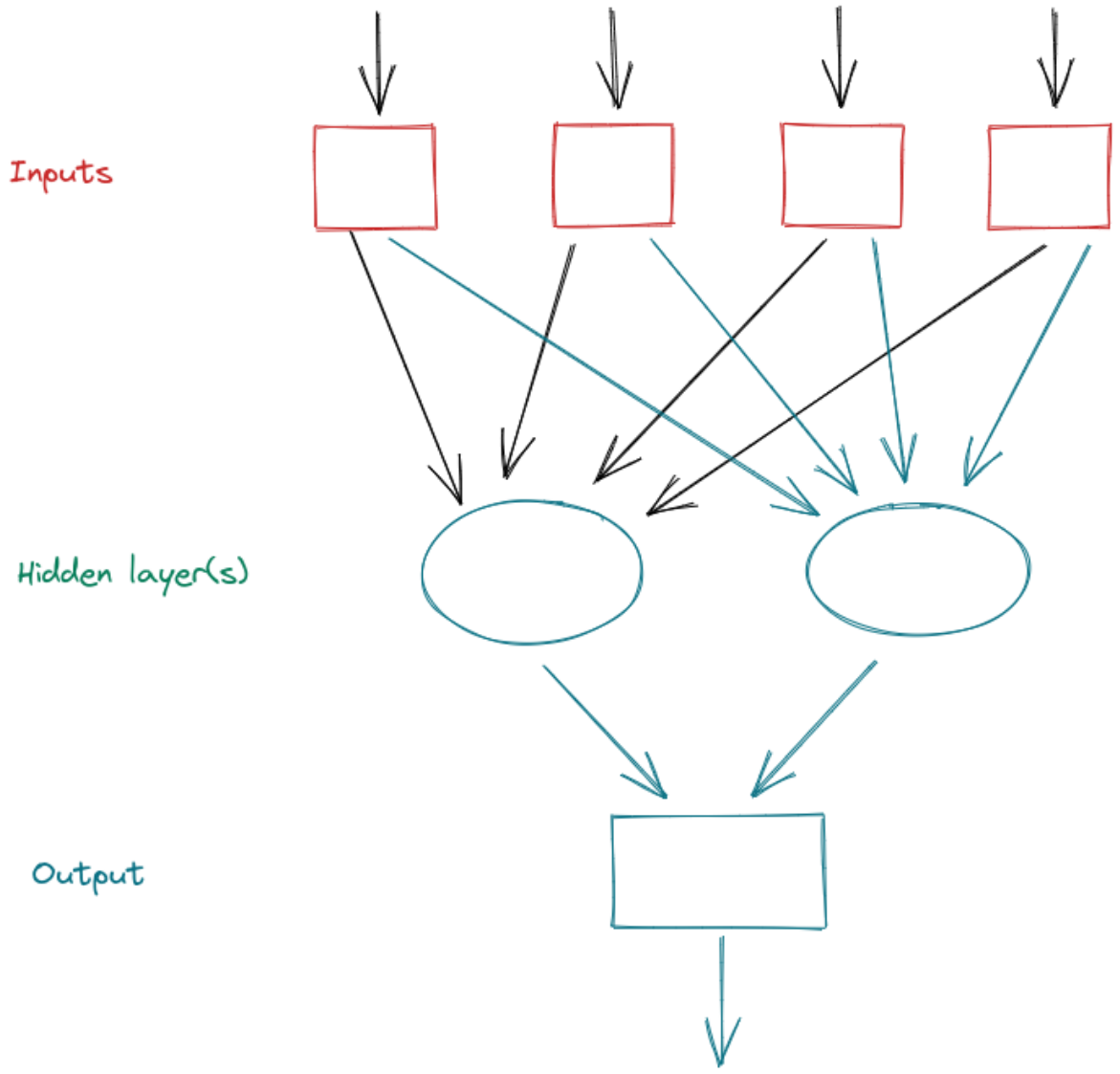
For example to predict the number of sales of a new product based on its price we might look at past sales numbers of similar products:

Supervised learning methods learn input-output relationships from training data.

Neural networks



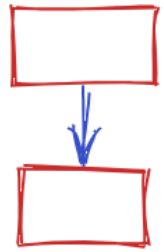
Neural networks



Neural networks: Terminology

Nodes

Mathematical computations (sums, products, ...) of the inputs



Edges

results from one node are passed to another node as inputs

Linear combinations of inputs

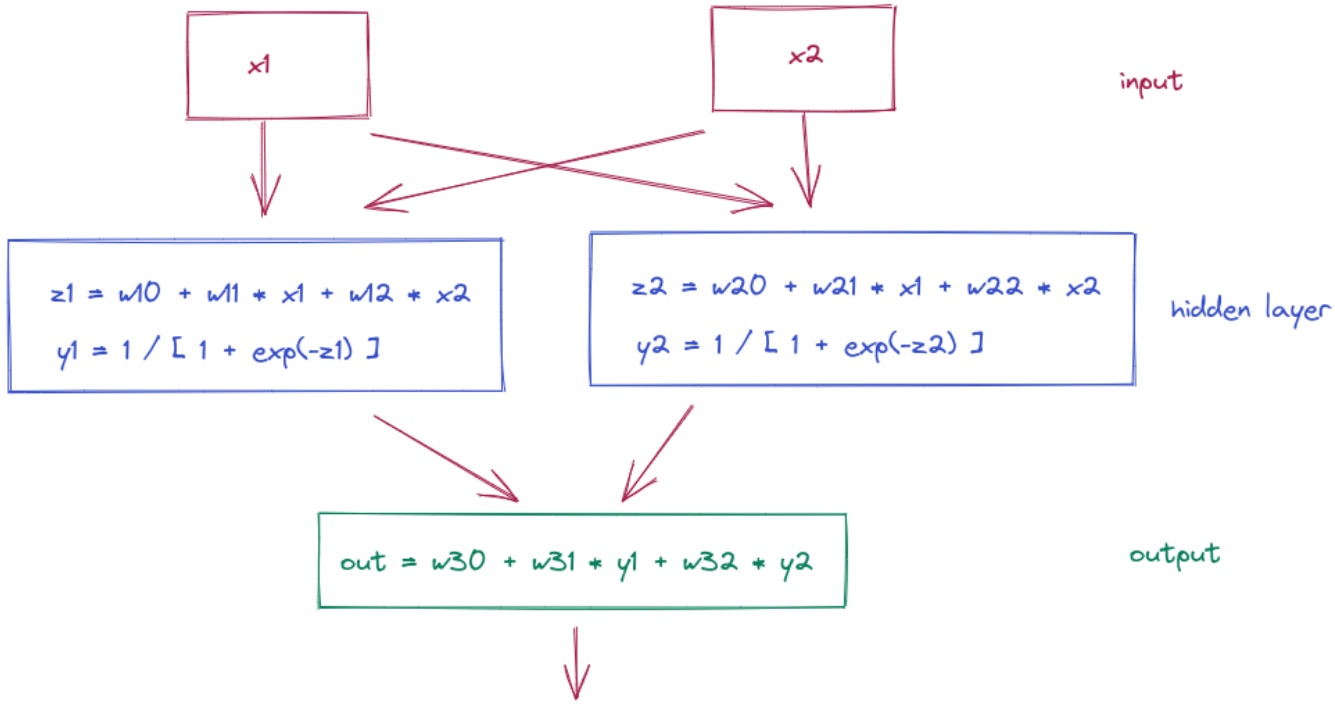
$$w_0 + w_1 \times input_1 + \cdots + w_p \times input_p$$

Activation function

$$output = f(w_0 + w_1 \times input_1 + \cdots + w_p \times input_p)$$

- ReLU $f(x) = max(0, x)$
- Sigmoid $f(x) = 1/[1 + e^{-x}]$

A "simple" neural network in full detail



- 2 inputs, 1 **hidden layer** with 2 nodes, and an output layer with one node
- layers are **densely connected**
- existence of hidden layer makes this a **deep neural network**
- 9 trainable weights $(w_{10}, w_{11}, \ldots, w_{32})$

Tensorflow

- `www.tensorflow.org`
- open-source machine learning platform
- provides tools and libraries for constructing, training, and applying neural network models
- tutorials, interactive examples, community support

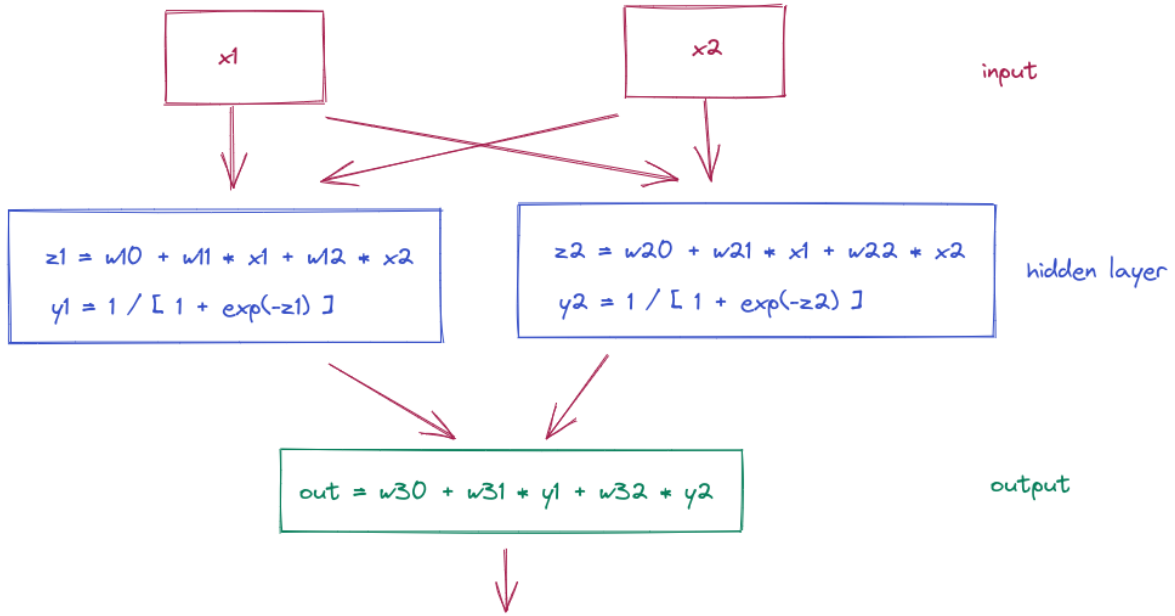
```
import tensorflow as tf
```

Keras

- `www.keras.io`
- high-level API built on top of tensorflow
- easy construction, training, debugging of standard types of neural networks
- less flexible but more beginner-friendly than tensorflow
- **We are using keras in this workshop.**

```
import keras
```

A simple neural network with keras

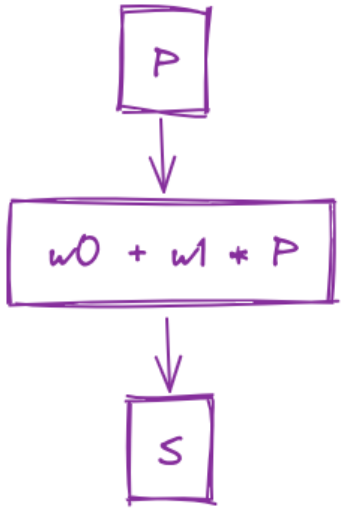


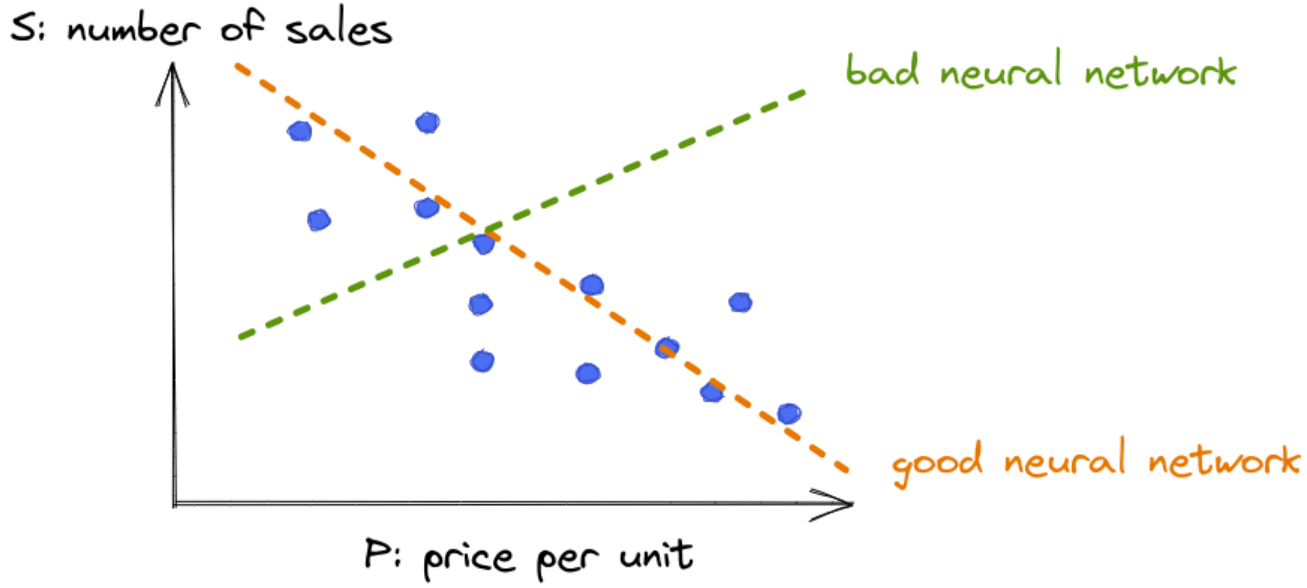

```
# construct model
model = keras.Sequential([ keras.layers.Input(shape=(2)),
                           keras.layers.Dense(2, activation='sigmoid'),
                           keras.layers.Dense(1)])

# compile and train
model.compile(optimizer='sgd', loss='mse')
model.fit(x_train, y_train)

# apply the model
prediction = model(x_new)
```

Training Neural Networks

NNs are trained by providing a **training data set** of inputs and corresponding outputs, and **systematically adjusting the network weights** so that the inputs are reliably mapped to expected outputs.

This is usually done by minimising some **loss function** that quantifies the distance between the NN predictions and the expected outputs.

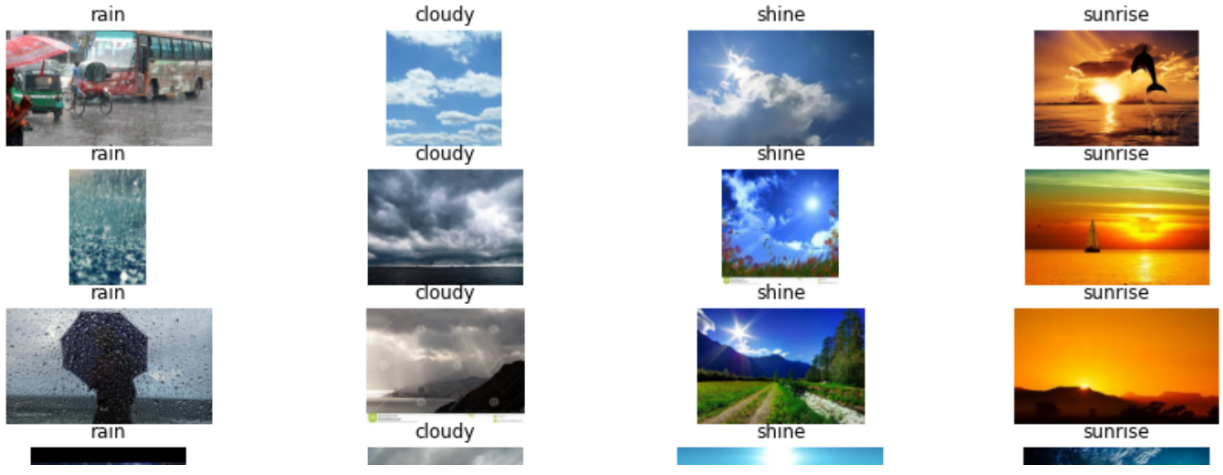




- The bad and good neural network differ only in their values of w_0 and w_1.
- The bad neural network produces outputs that differ from the expected outputs in the training data.
- The good neural network fits the training data well.
- Training a neural network means adjusting its weights to turn it from bad (poor fit) to good.
- Training algorithms: Gradient descent, backpropagation (We won't cover these today.)

Implementing and training a neural network for image classification in keras

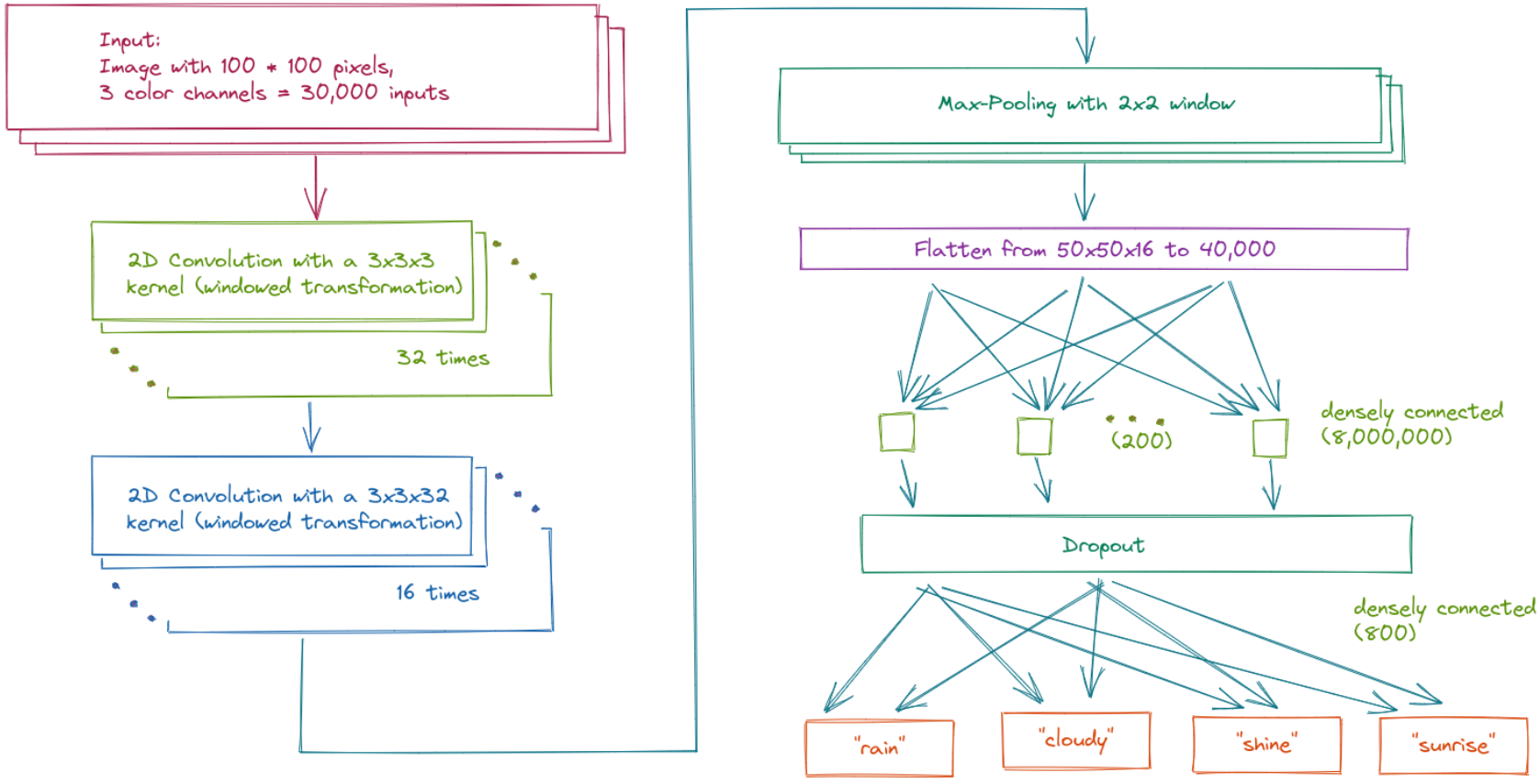
```
weather-img
├── train
│   ├── cloudy
│   │   ├── 11.jpg
│   │   └── ...
│   ├── rain
│   │   ├── 107.jpg
│   │   └── ...
│   ├── shine
│   │   ├── 131.jpg
│   │   └── ...
│   └── sunrise
│       ├── 121.jpg
│       └── ...
├── test
│   └── ...
└── validate
    └── ...
```



(1125 images in total)

Data augmentation to increase the amount of training data by randomly shearing, rotating, zooming, flipping the images in the training data

Architecture: Deep convolutional neural network



```
# build model
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(IMG_SIZE,IMG_SIZE,3)),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu',
                           padding='same'),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu',
                           padding='same'),
    tf.keras.layers.MaxPool2D(pool_size=(2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=200, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(4, activation='softmax')
])
```

```
# compile model
model.compile(optimizer='adam',
              loss='categorical_crossentropy')


# fit model (takes about 3 minutes)
model.fit(train, validation_data=val, epochs=20)


# use model to classify a new image
prediction = model(new_img)

# Output:
# <tf.Tensor: shape=(1, 4), dtype=float32, numpy=
# array([[0.05, 0.01, 0.93, 0.01]], dtype=float32)>
```

Performance

Good

labels = [cloudy rain shine sunrise]
prediction = [0.04 0.01 0.04 0.91]
truth = [0 0 0 1]

Meh

labels = [cloudy rain shine sunrise]
prediction = [0.01 0.03 0.42 0.54]
truth = [0 0 0 1]

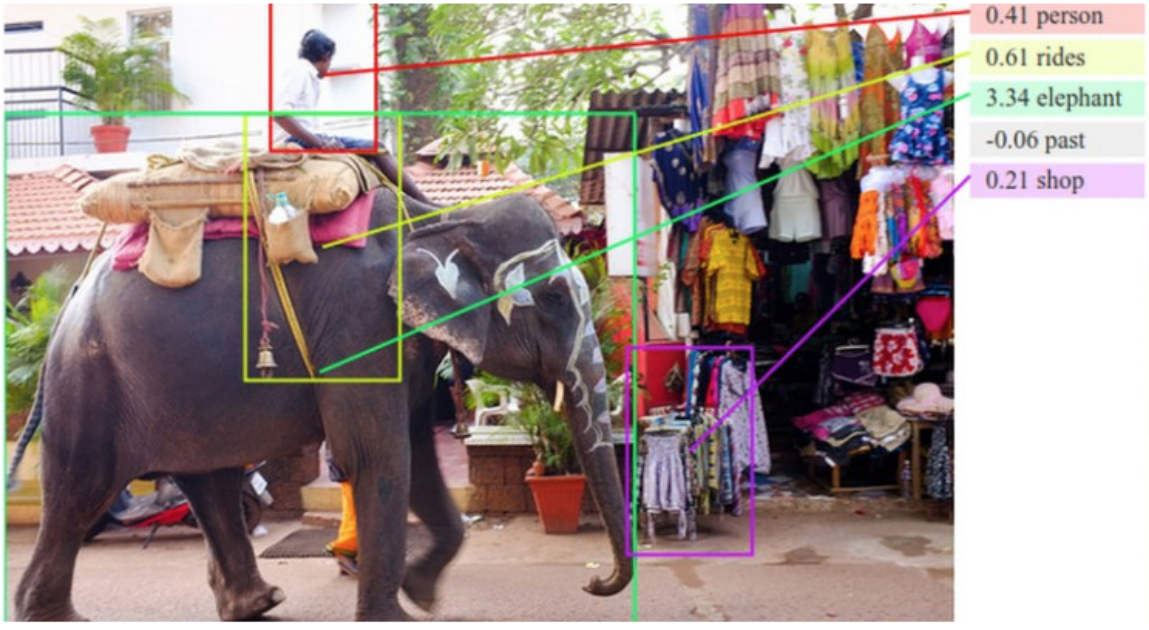
Bad

labels = [cloudy rain shine sunrise]
prediction = [0.05 0.01 0.94 0.0]
truth = [1 0 0 0]

Overall the deep convolutional neural network can classify new images correctly over 90% of the time. The accuracy increases to over 95% when training for longer.

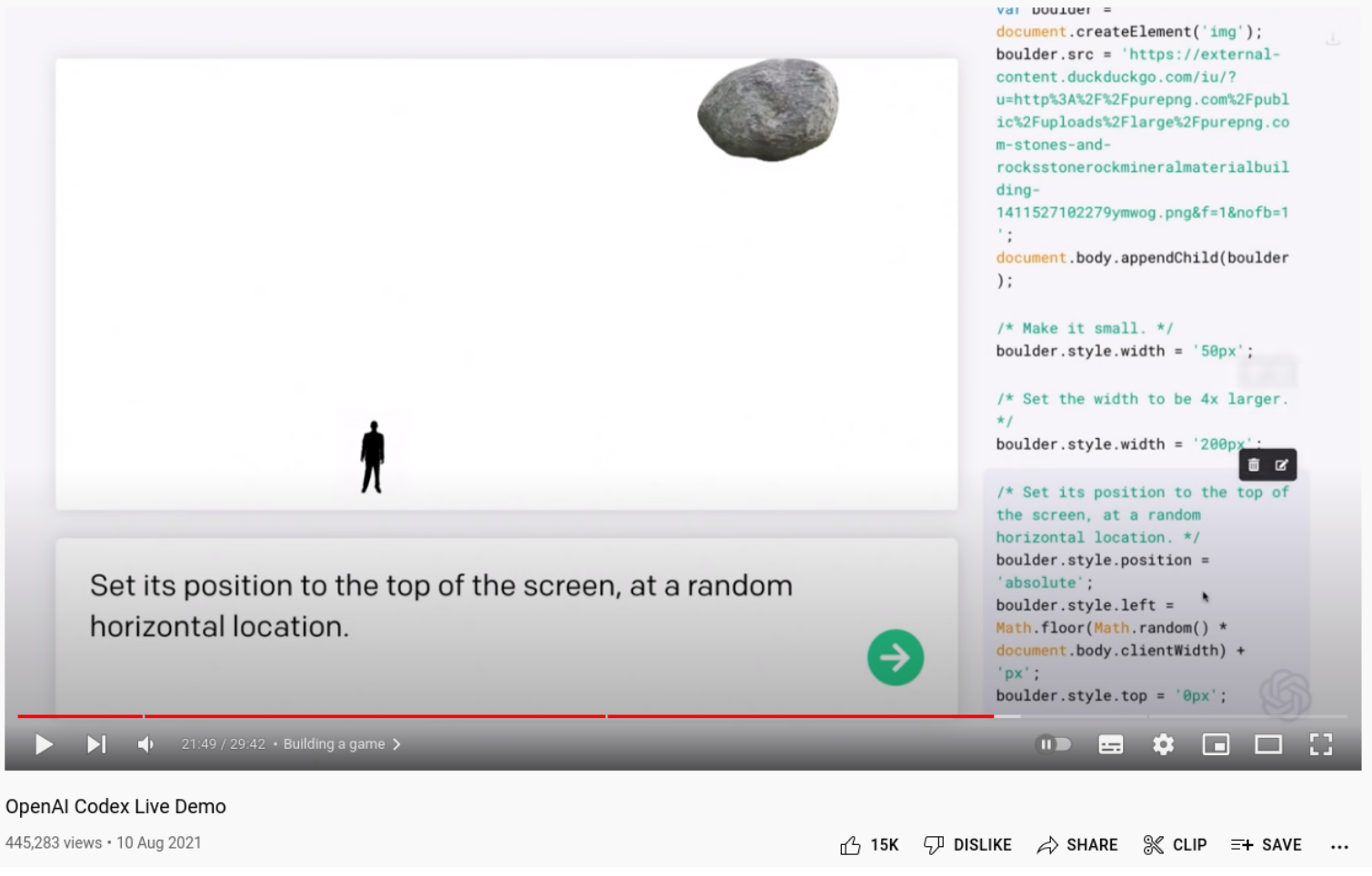
Other applications

Image description/captioning



Other applications

Automatic code generation (GPT-3)



Other applications

AI generated images (Dall-E)

TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

TEXT PROMPT a snail made of harp. a snail with the texture of a harp.

AI-GENERATED IMAGES

Some advice for learning machine learning

- Explore online tutorials, modify and play with them.
- Start with simple problems.
- Don't be afraid to dig into the mathematics.
- Work through many small projects rather than few big ones.

- Contact me if you want to learn more or have questions:
  Stefan Siegert <s.siegert@exeter.ac.uk>

- Thank you!