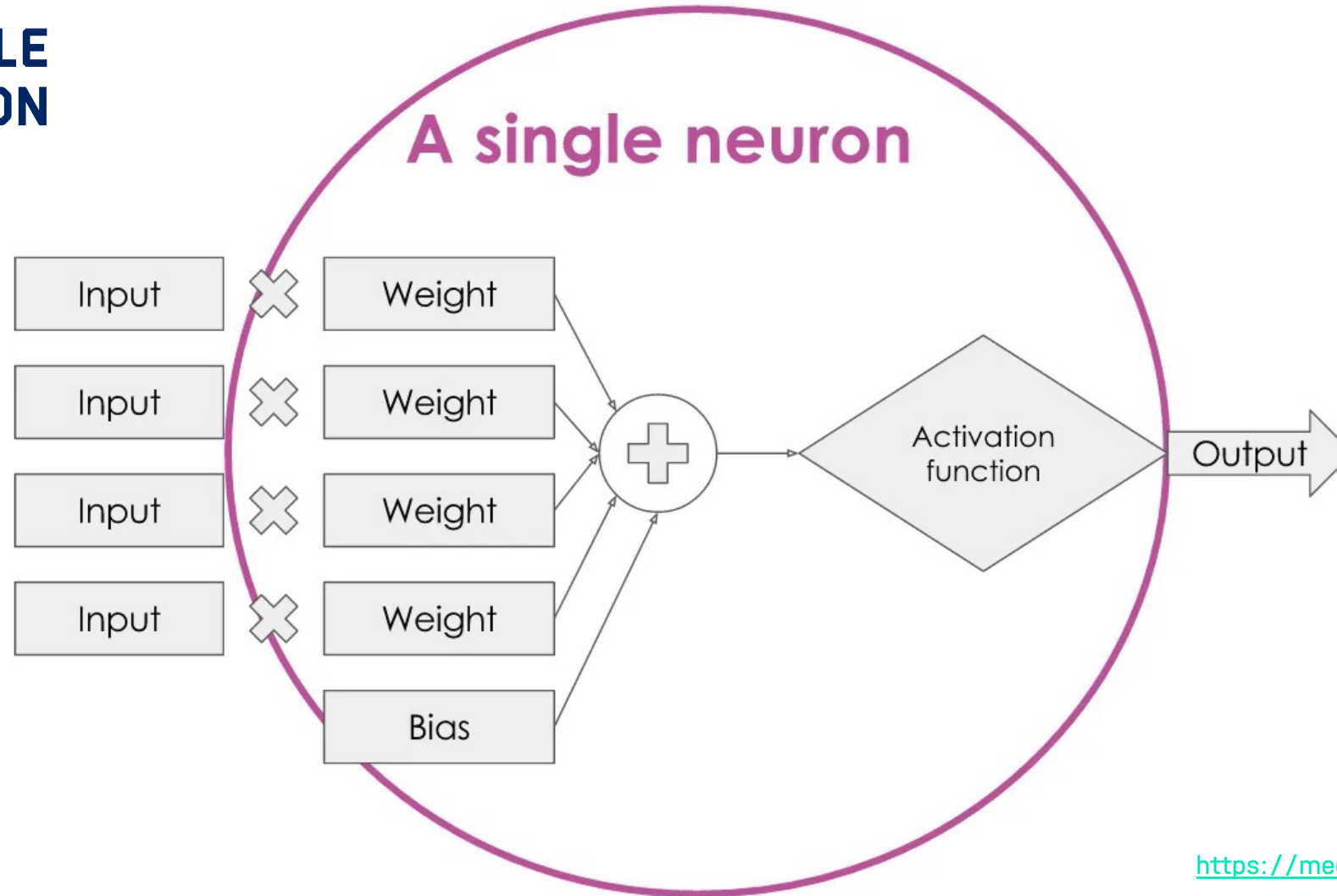


RECAP

I2DL 28/10/2025

A SINGLE NEURON



<https://medium.com/@RosieCampbell/>

ACTIVATION FUNCTIONS

why not just use
identity?

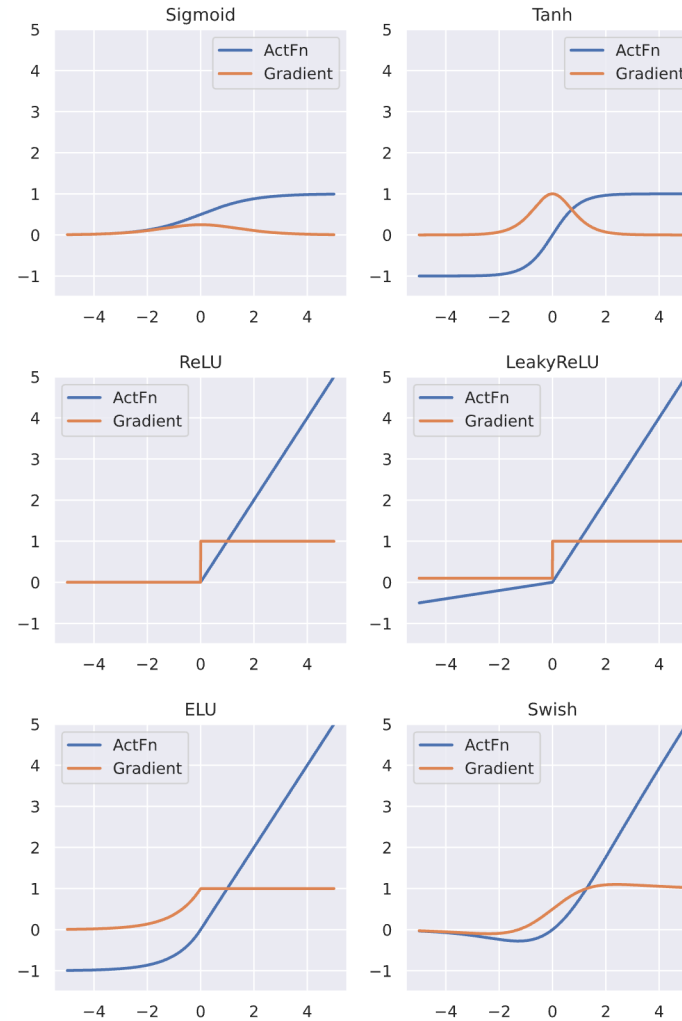
need a non-linear
function

sigmoid might cause
vanishing gradients

it's nice if the gradient is
easy to calculate

sometimes we *want*
some weights to $\rightarrow 0$

last three ones keep
some negative values



[https://lightning.ai/docs/pytorch/stable/notebooks/
course_UvA-DL/02-activation-functions.html](https://lightning.ai/docs/pytorch/stable/notebooks/course_UvA-DL/02-activation-functions.html)

FEEDFORWARD NEURAL NETWORK

A feedforward neural network implements the *perceptron* algorithm – this was intended to be a machine with 400 photocells!

stack layers of neurons,
connect them all with
activation functions
→ feedforward neural net

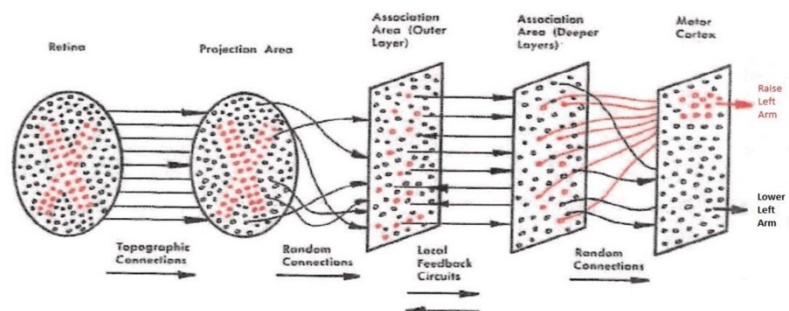


FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

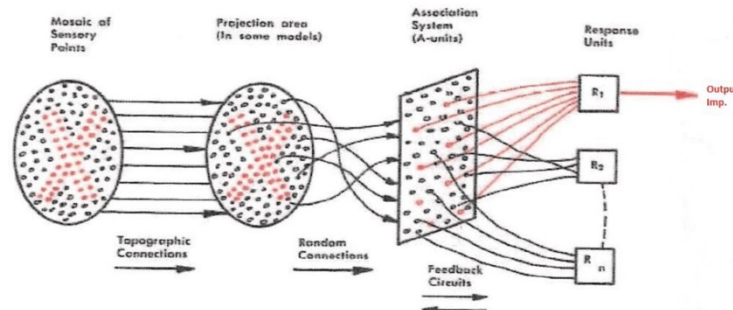
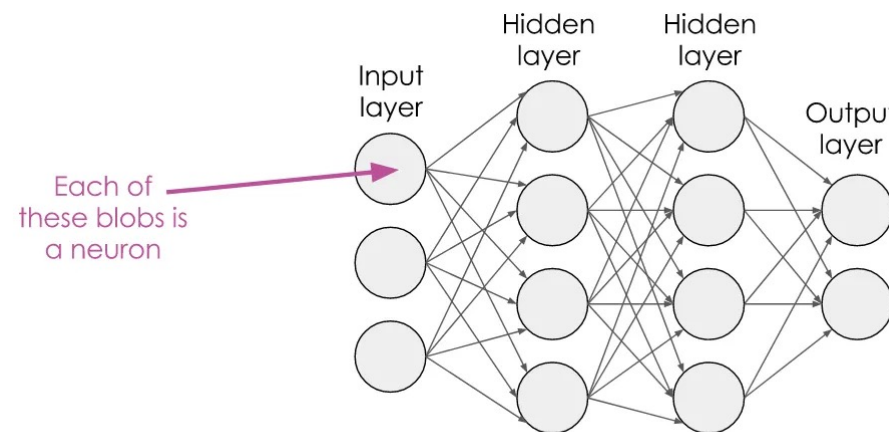


FIG. 2 — Organization of a perceptron.



<https://medium.com/@RosieCampbell>
<https://en.wikipedia.org/wiki/Perceptron#History>

FEEDFORWARD NEURAL NETWORK

These all mean the same:

- fully connected layer
- (keras) dense layer
- (pytorch) linear layer

also:
FFN = Multi-Layer
Perceptron

→
scikit-learn has probably
the most accessible
(i.e. easy to use)
MLP model

slide #5



Class [MLPClassifier](#) implements a multi-layer perceptron (MLP) algorithm that trains using [Backpropagation](#).

MLP trains on two arrays: array X of size (n_samples, n_features), which holds the training samples represented as floating point feature vectors; and array y of size (n_samples, 1), which holds the target values (class labels) for the training samples:

```
>>> from sklearn.neural_network import MLPClassifier
>>> X = [[0., 0.], [1., 1.]]
>>> y = [0, 1]
>>> clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
...                     hidden_layer_sizes=(5, 2), random_state=1)
>>> clf.fit(X, y)
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(5, 2), random_state=1,
              solver='lbfgs')
```



v2.9.0 (stable)

Linear

`class torch.nn.Linear(in_features, out_features, bias=True, device=None, dtype=None)` [\[source\]](#)

Applies an affine linear transformation to the incoming data:
 $y = xA^T + b$.

This module supports [TensorFloat32](#).

On certain ROCm devices, when using float16 inputs this module will use [different precision](#) for backward.

Parameters:

- **in_features** (*int*) – size of each input sample
- **out_features** (*int*) – size of each output sample

the layer will not learn an

ns any number of
nd $H_{in} = \text{in_features}$.

but the last dimension are
and $H_{out} = \text{out_features}$.



► [Keras 3 API documentation](#) / [Layers API](#) / [Core layers](#) / [Dense layer](#)

Dense layer

Dense class

[\[source\]](#)

```
keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    lora_rank=None,
    lora_alpha=None,
    **kwargs
)
```

Just your regular densely-connected NN layer.

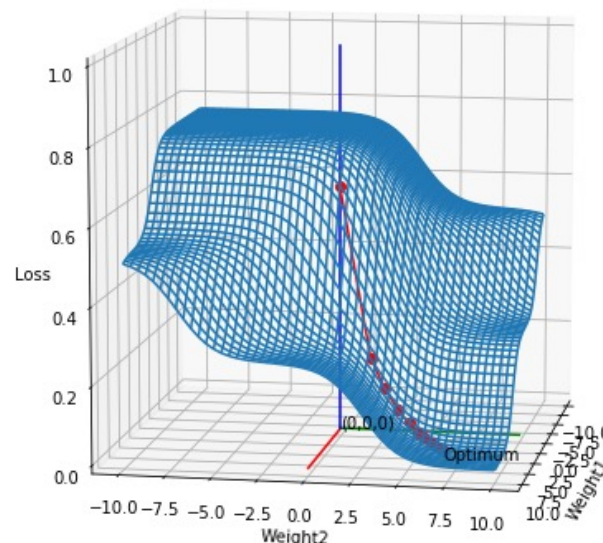
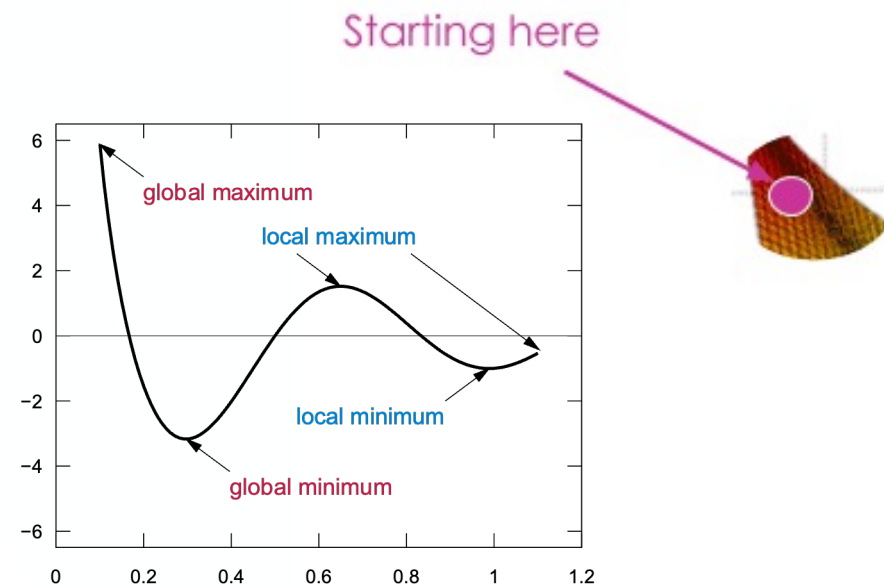
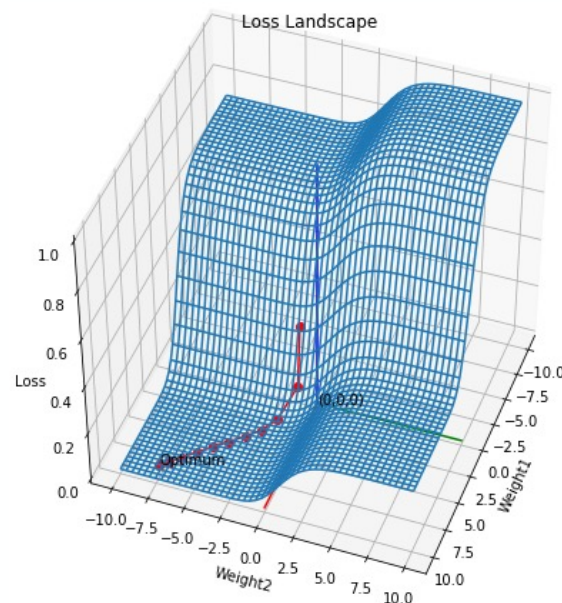
https://scikit-learn.org/stable/modules/neural_networks_supervised.html

GRADIENT DESCENT

we formulate learning as minimizing an error function, usually called the loss.

Because the error functions of artificial neural networks are non-convex, we will almost certainly only find a local minimum.

The loss value is a number, the "height" above ground in a multi-dimensional loss landscape.



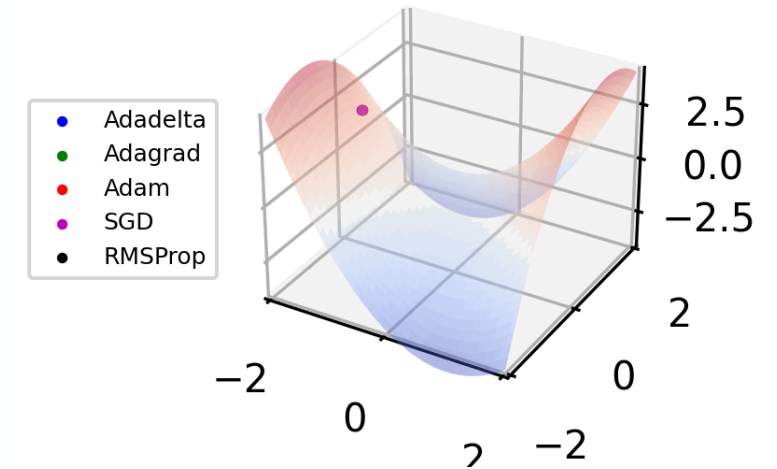
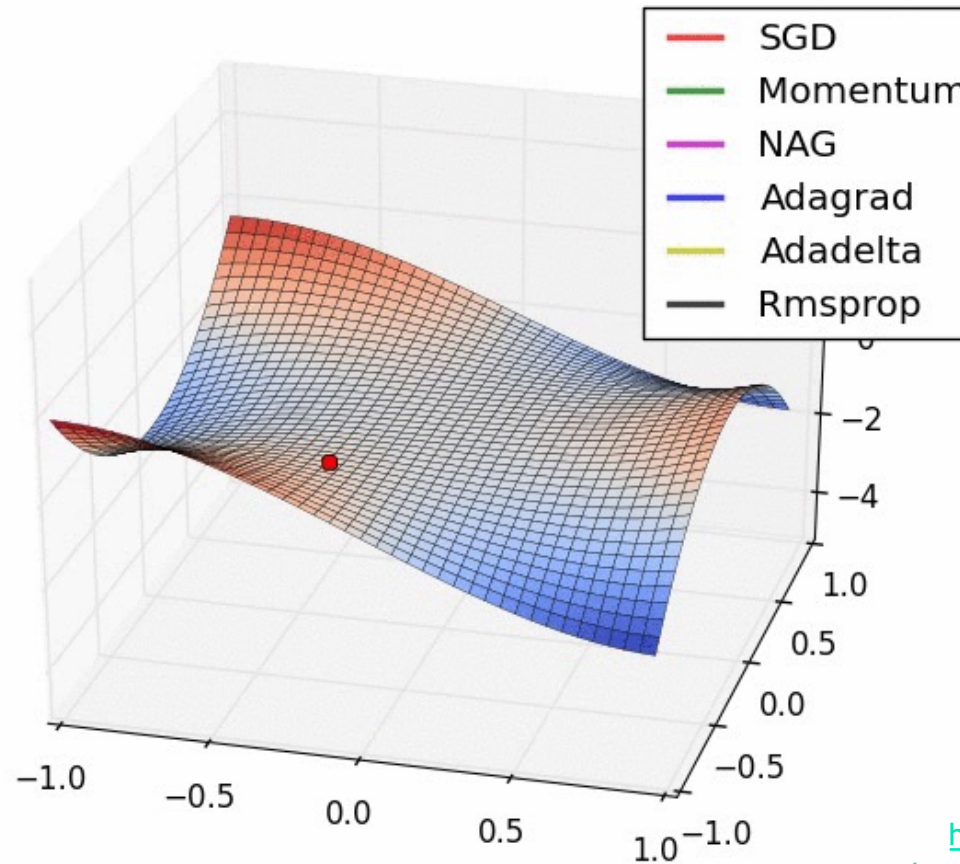
<https://ai.stackexchange.com/users/2844/dan-d>
<https://medium.com/@RosieCampbell>
<https://en.wikipedia.org/wiki/Backpropagation>

OPTIMIZERS

At the core, all common optimizers use variations of gradient descent.

Actually *stochastic* gradient descent: ∇ is calculated in many little batches instead of for the entire data set.

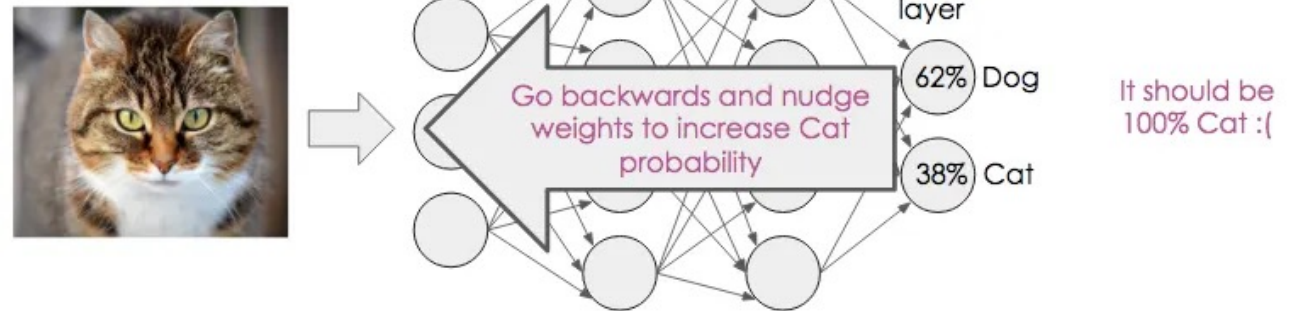
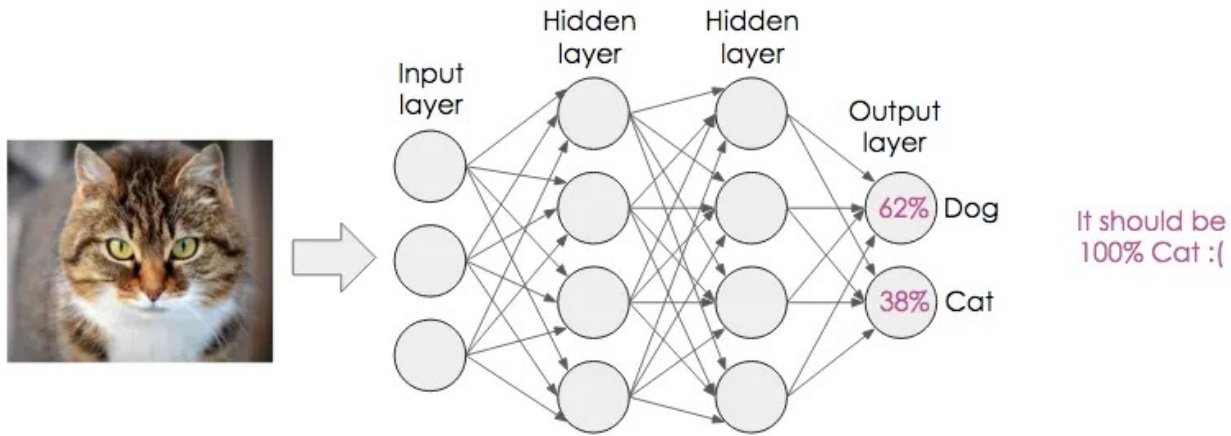
They all aim to speed up learning along stable directions by gathering momentum, while slowing oscillations.



LEARNING RATE:
 😊 OPTIMIZER
 👍 SCHEDULE
 😐 CONSTANT LR

https://en.wikipedia.org/wiki/Stochastic_gradient_descent
<https://optimization.cbe.cornell.edu/index.php?title=RMSProp>

BACKPROPAGATION



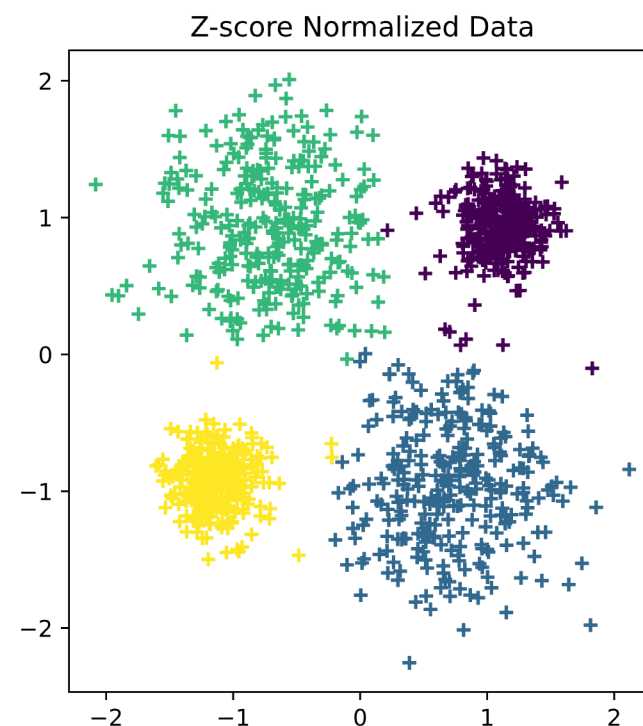
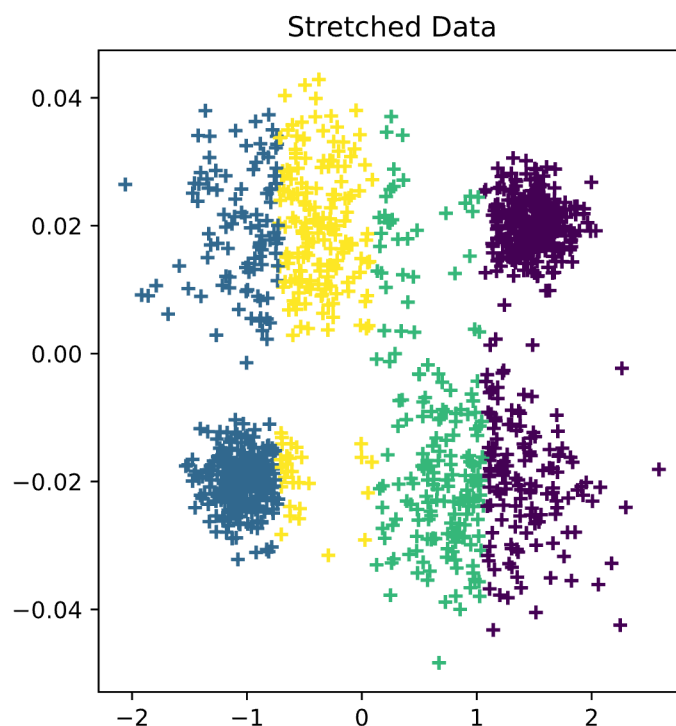
<https://medium.com/@RosieCampbell>

FEATURE SCALING

Feedforward neural
networks can be
susceptible to features
"living" on different
numerical scales
→ use feature scaling

min-max scaling, median
scaling or z-scaling
are common

→
the y-axes are different!



https://en.wikipedia.org/wiki/Feature_scaling

RECAP CORE CONCEPTS

I2DL 29/10/2025

CORE CONCEPTS RECAP

Learning Rate

Hyperparameter Tuning

Epoch

what is one-hot-encoding

name a non-linear
activation function

CORE CONCEPTS RECAP

when to use categorical cross entropy
vs binary cross entropy

what are optimizers?

explain the idea of
backpropagation

what is a batch?