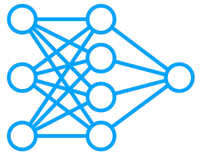
Deploying and Optimizing Neural Networks

Pratheerth Padman
Data Scientist

Training a Neural Network



Feeding data into the network and adjusting the weights and biases to minimize errors

Forward pass – Input data is fed through the network to produce an output

Backward pass - Network adjusts its parameters based on the error between its output and the actual data

This cycle repeats over epochs

A key aspect of training a neural network is the selection and tuning of hyperparameters

Hyperparameters

Hyperparameters are the settings that govern the training process. Unlike weights and biases, they are not learned from the data but set before training begins.

Essential Hyperparameters

Learning rate

Determines how much we adjust the weights with respect to the loss gradient

Number of epochs

Determines how many times the entire dataset is passed through the network

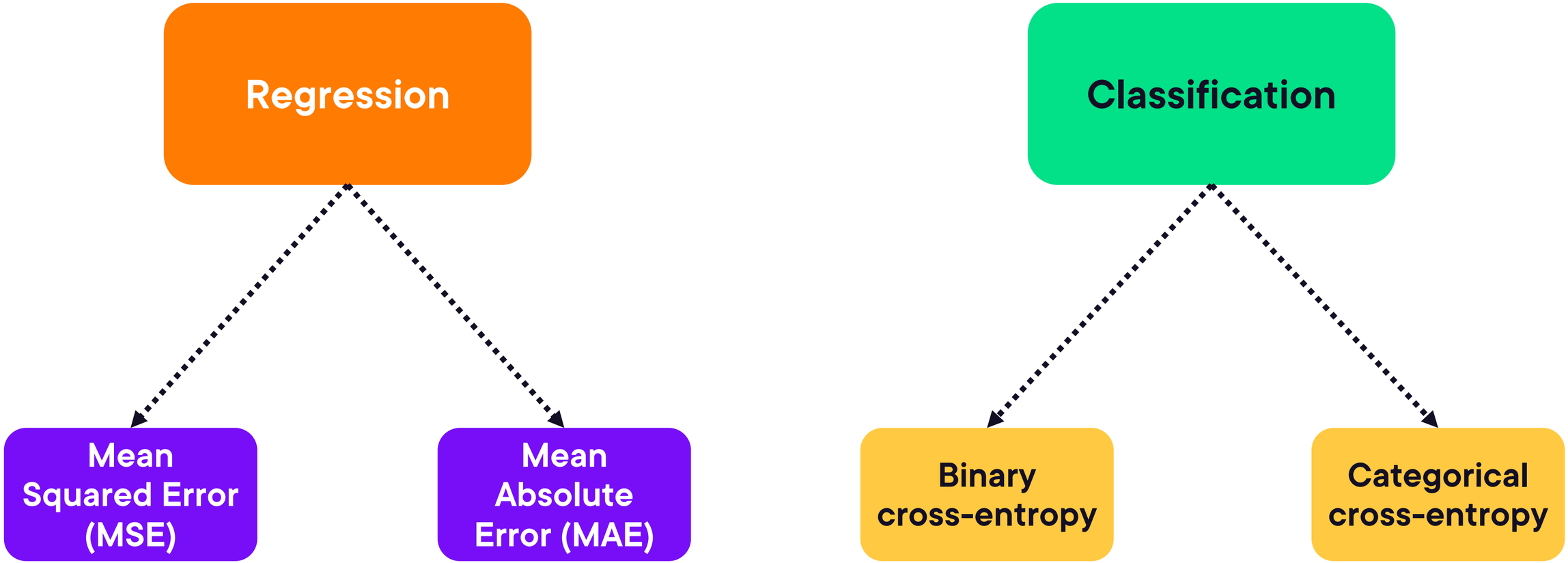
Batch size

Determines the number of samples processed before the model is updated

Loss Functions

A loss function measures the difference between the network's predictions and the actual target values. It's a crucial component because the goal of training a neural network is to minimize this loss.

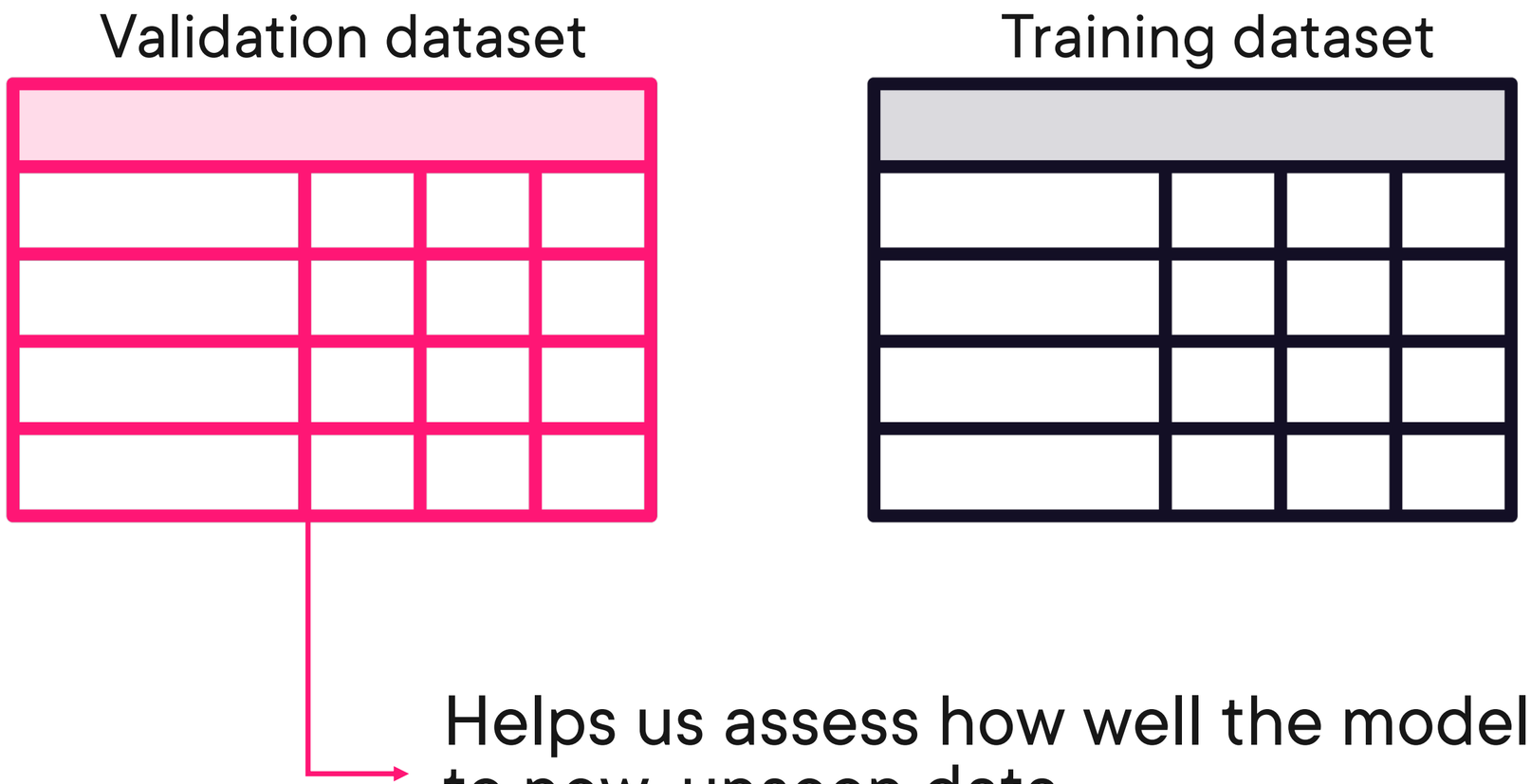
Common Loss Functions



The choice of loss function can also depend on specific characteristics of your data and model

Model Evaluation and Deployment

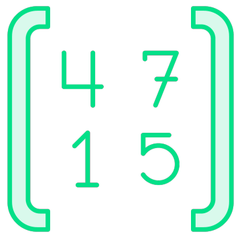
Model Evaluation Techniques



Model Performance

Accuracy, precision, recall, and F1 score

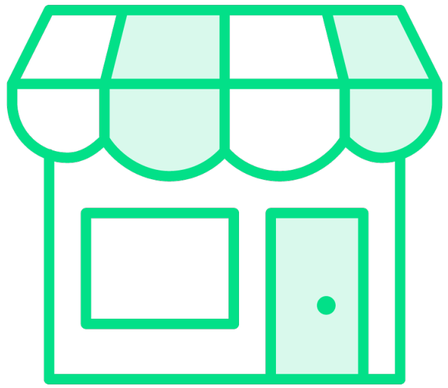
For regression models, metrics like mean squared error or mean absolute error are often used



Confusion matrices help visualize the model's performance across different classes

Model Deployment

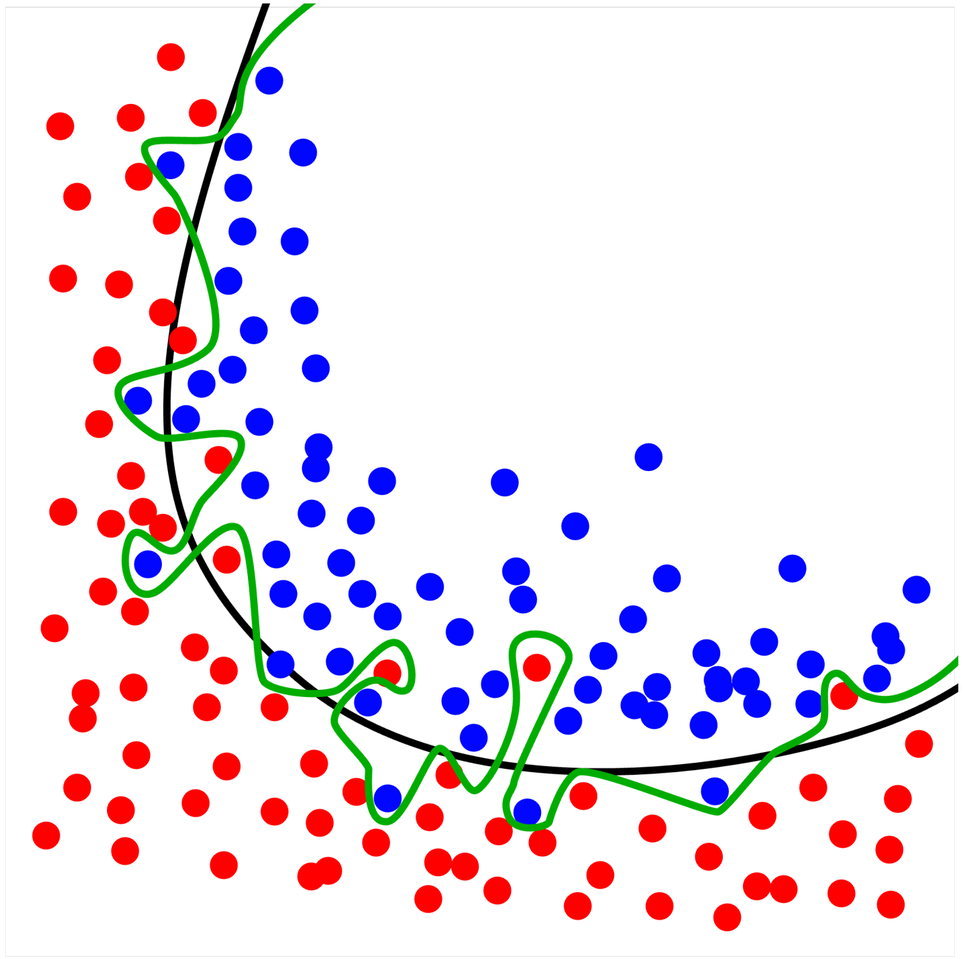
Deploying a neural network model involves integrating it into an existing production environment.



- Ensuring model's scalability
- Managing resource requirements
- Monitoring systems for performance tracking
- Model updation and maintenance

Optimization Techniques

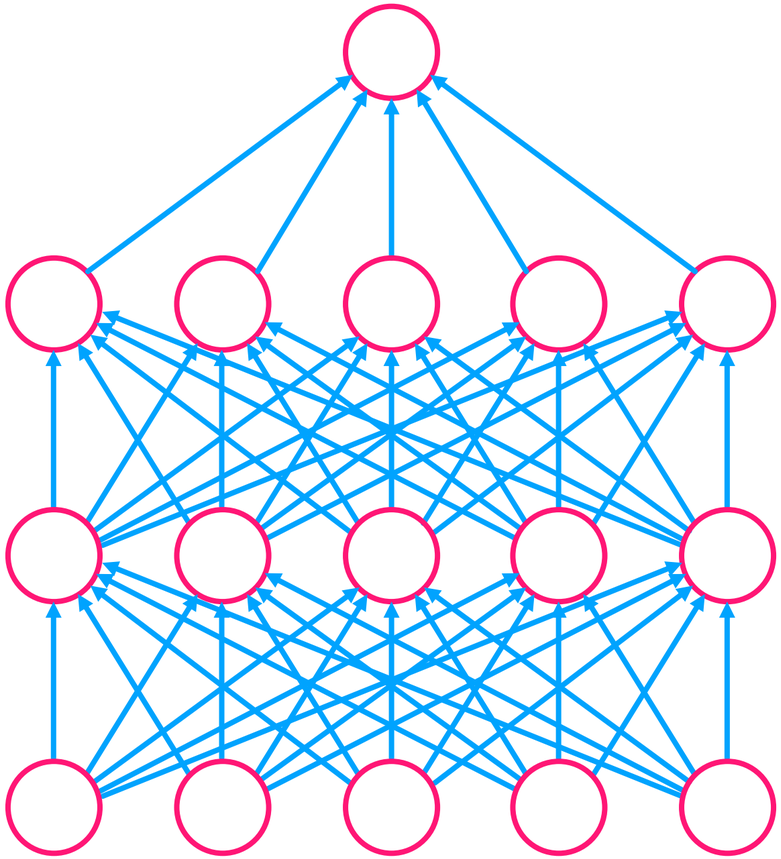
Overfitting



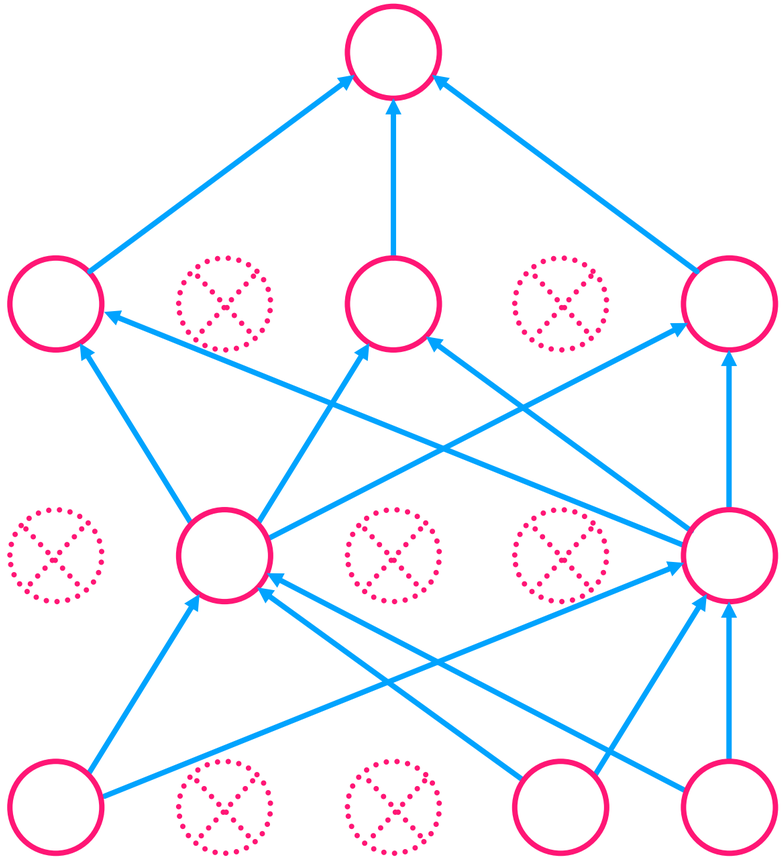
Regularization

Techniques like L1 and L2 regularization work by adding a penalty to the loss function for large weights, encouraging the model to keep the weights small and, in turn, simpler. This simplicity helps the model generalize better to new data.

Dropout



(a) Standard neural net



(b) After applying dropout

Batch Normalization

Imagine each layer of a neural network as a processing station

Data comes in, gets processed, and moves to the next layer

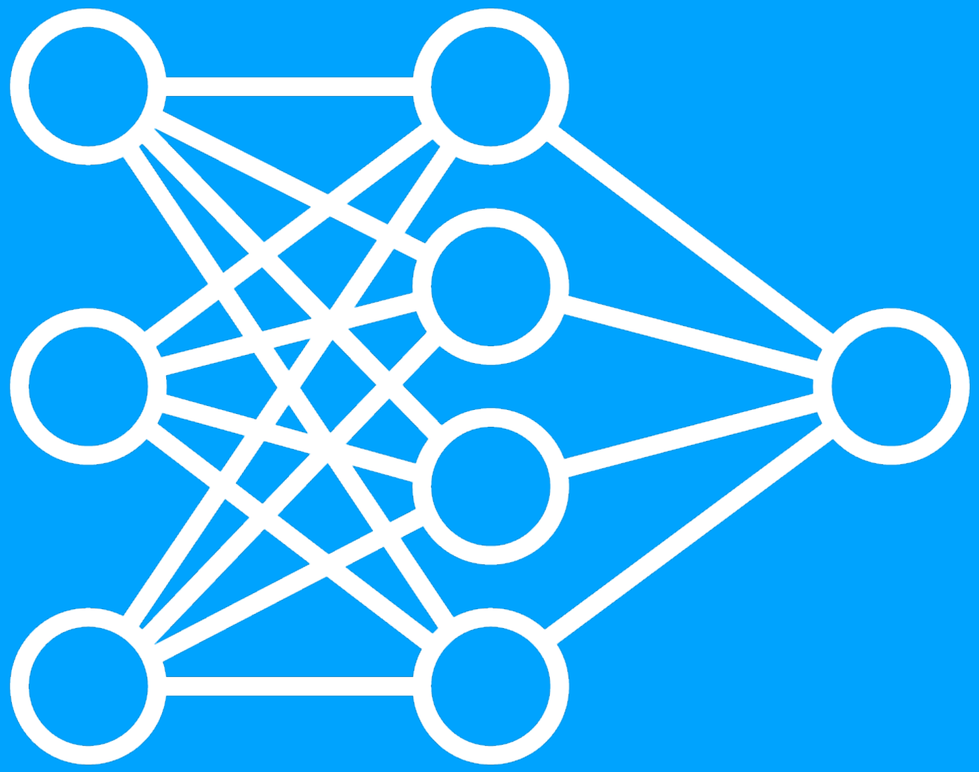
If the incoming data is too high or too low, it makes processing difficult and slow

Batch normalization helps by making this data more uniform or normalized before it goes into each layer

It does this by adjusting the data in each mini-batch

This process stabilizes and accelerates the training process

Types of Neural Networks



Feedforward Neural Network

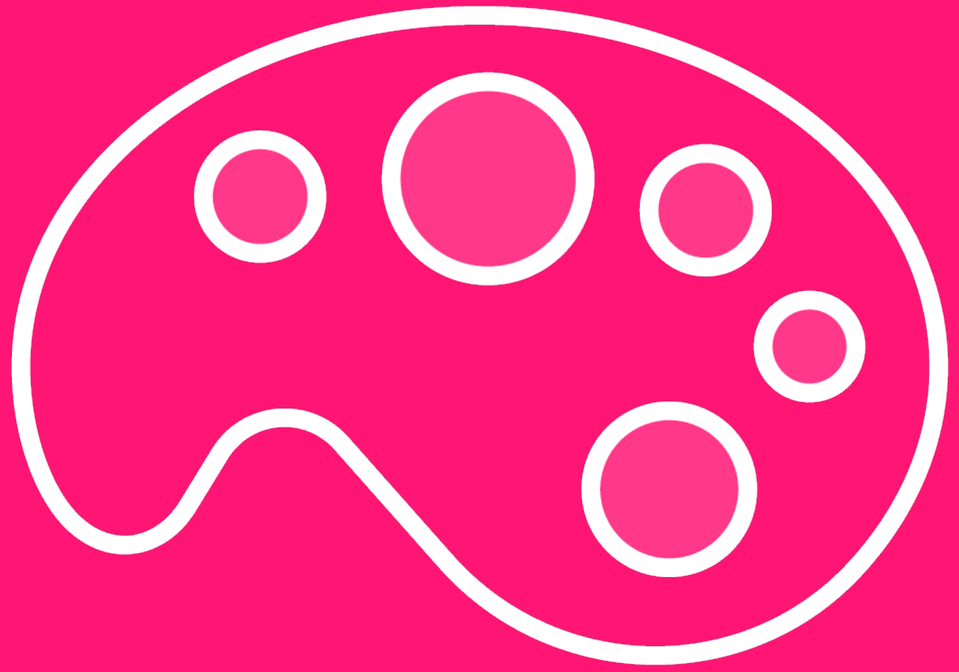
This is the simplest form of neural networks, where the information moves in only one direction—from input to output, without any loops

Recurrent Neural Network

RNNs have connections that form cycles, allowing information to persist. This makes them ideal for tasks involving sequential data, such as language modeling and text generation.

Convolutional Neural Network

CNNs are tailored for grid-patterned data like images, where the arrangement of pixels is important. They're widely used for image and video recognition tasks.



Generative Adversarial Networks

Generative Adversarial Networks, or GANs, create realistic new data similar to what they've seen. They have two main parts: a generator that makes data and a discriminator that judges it. This helps GANs get better at producing very realistic outputs, like lifelike images.

Demo

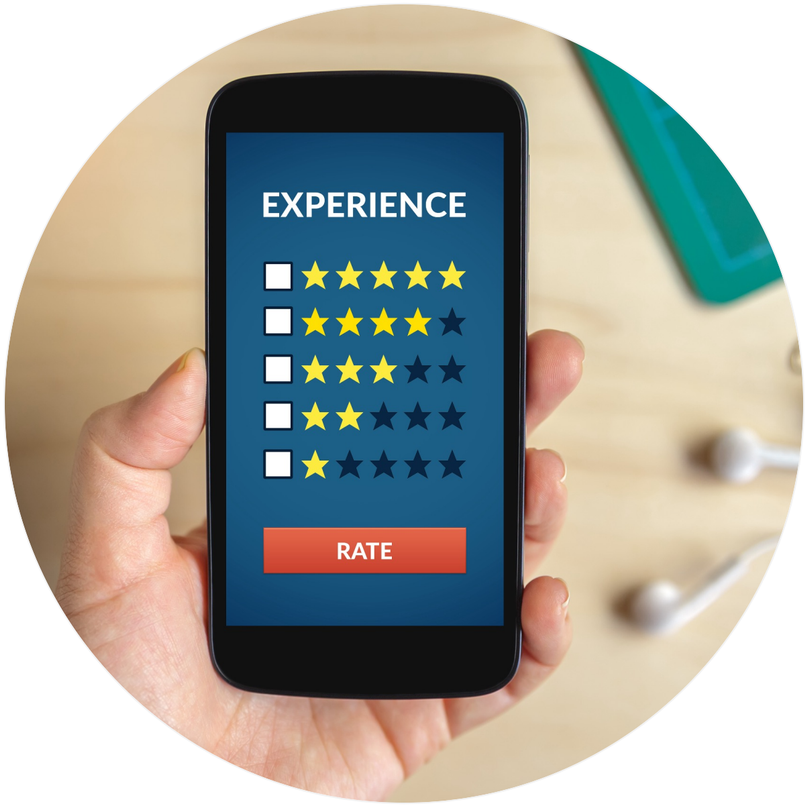
Setting up your environment

Demo

Building and training your neural network

Demo

Improving the performance of your model



Feedback

Discussion tab for any feedback or questions

Please leave a star rating!

Thank you for watching!