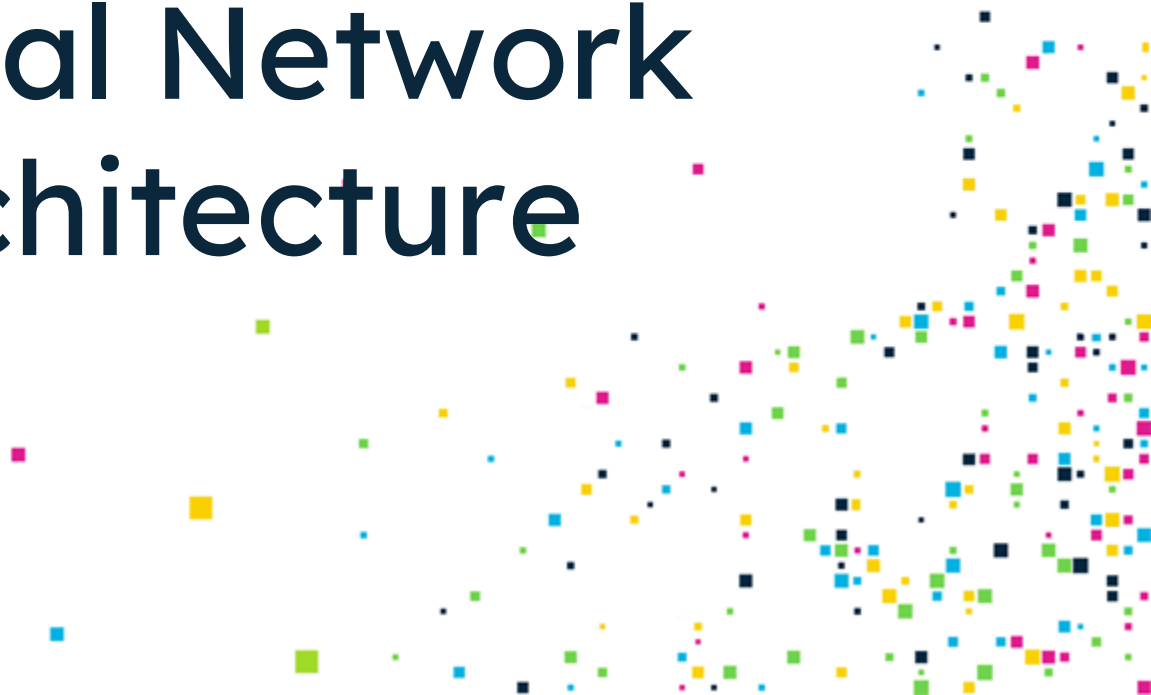
Neural Network Architecture



can code communities

Review Questions

- What are the three points at which we can calculate the cost for training?
- How do we minimize the cost of a model?
- What are some pros and cons of a feed forward neural network?

Review Take Home

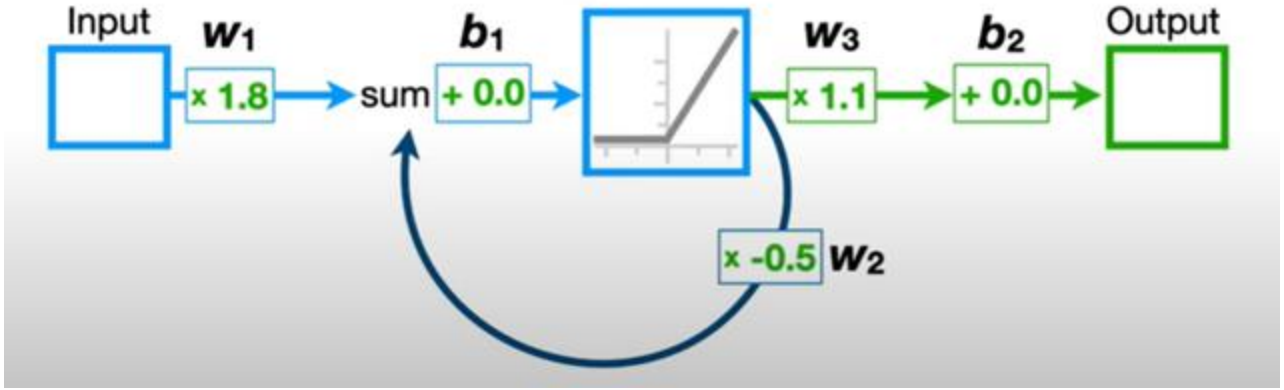
Look into stochastic gradient descent with momentum and write up an explanation of how momentum makes this optimization function more efficient than normal stochastic gradient descent.

Recurrent Neural Networks

- RNNs are designed to handle sequential data (where order matters) like text, speech, or time series data.
- RNNs focus on the sequential data piece by piece instead of the whole set all at once.
    - There is no defined limit to the amount of sequential data passed through the input.
- Applications include machine translation, speech recognition, text summarization, stock price prediction.

Recurrent Neural Networks

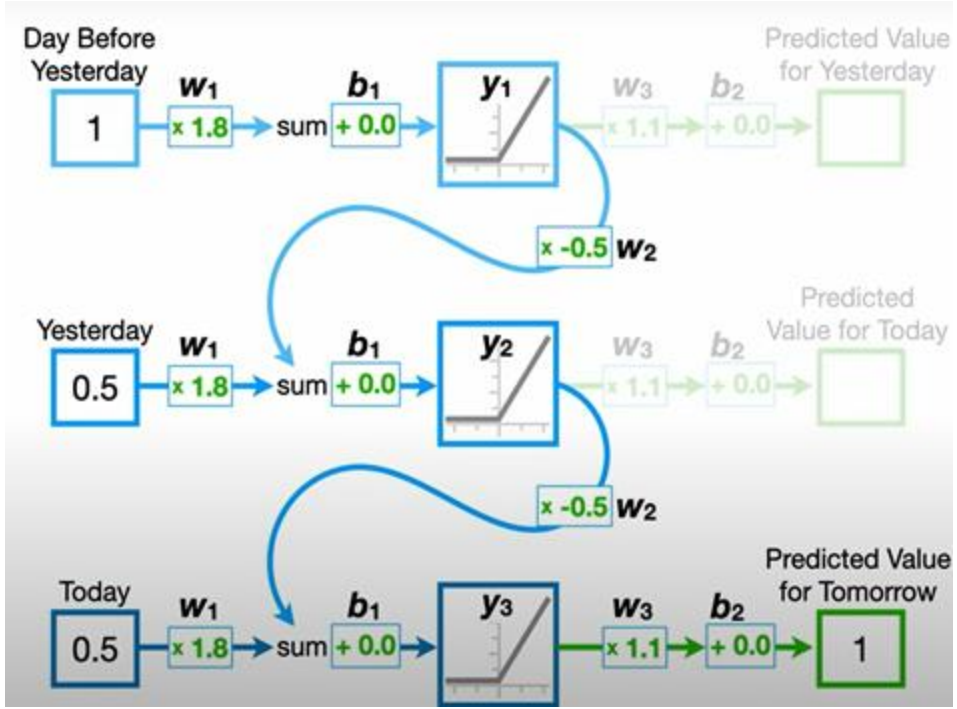
- The basic RNN unit consists of a single recurrent neuron with connections that loop back on themselves, allowing information to persist across the network, enabling them to "remember" what they've processed previously.



Recurrent Neural Networks

- We can visualize the RNN as separate networks at two or more instances where the input data is following the sequential order.
- This displays the concept of feeding the prior activation function back into the current activation function.
- Another thing to point out when looking at this visualization is that the weights and biases are the same within each pass of the sequential data.

Recurrent Neural Networks



Vanishing/Exploding Gradient Problem

- There are some issues within the architecture that make RNNs harder to train and thus makes them less commonly used.
- This problem is called the vanishing or exploding gradient problem.
  - RNNs have no limit to how many inputs of sequential data that can be passed through the model
  - The weights and biases are shared across every input
  - This makes the optimization of these weights and biases hard to manage (especially the recurrent weight).

Vanishing/Exploding Gradient Problem

- The recurrent weight is exponentially tied to the number of sequential inputs there are.
  - A weight with a value below one can diminish to almost zero
  - A weight with a bigger value can skyrocket depending on the amount of sequential inputs.
- This weight is being used to calculate the gradient descent (the size of the step to take to minimize the cost), and super large or small steps can ruin the optimization process.

Long Short Term Memory

- LSTMs are equipped with a more sophisticated memory cell compared to traditional RNNs.
  - Effective in tasks such as speech recognition and natural language processing.
- They are capable of learning long-range dependencies in sequential data.
  - They have mechanisms to selectively remember or forget information over time.
- LSTMs consist of a cell state, which serves as the memory, and three gates.

Cell State

The cell state is the core component of an LSTM

- Acts as a long-term memory.
- Unlike traditional RNNs where information degrades over time, the cell state allows LSTMs to persist information relevant for future predictions.
- It avoids the vanishing/exploding gradient problem by:
  - Not having any consistent weights along its path
  - Designing computations within the gates to not let the values get out of a usable range.
- The short term memory does deal with consistent weights, but it gets redefined as the output of the model which is more directly influenced by the long term memory.

LSTM Gates

The three gates within a LSTM model are the forget gate, input gate, and output gate.

- We start with the forget gate instead of the input gate because that is how this model is read from left to right.

1. Forget gate: Determines the percentage of information in the cell state, allowing the LSTM to forget irrelevant information from the past.
2. Input gate: Controls the flow of information into the cell state, determining which information to store.
    a. This is done by summing the current long term memory value with the newly computed long term memory value.
3. Output gate: Regulates the information flow from the cell state to the output, ensuring that the relevant information is used for predictions.

Sigmoid vs. Tanh

In the LSTM architecture, we use two functions for different purposes. The two functions are the sigmoid and tanh functions. Both have similar curves, but the range of y values affect the use within this structure.

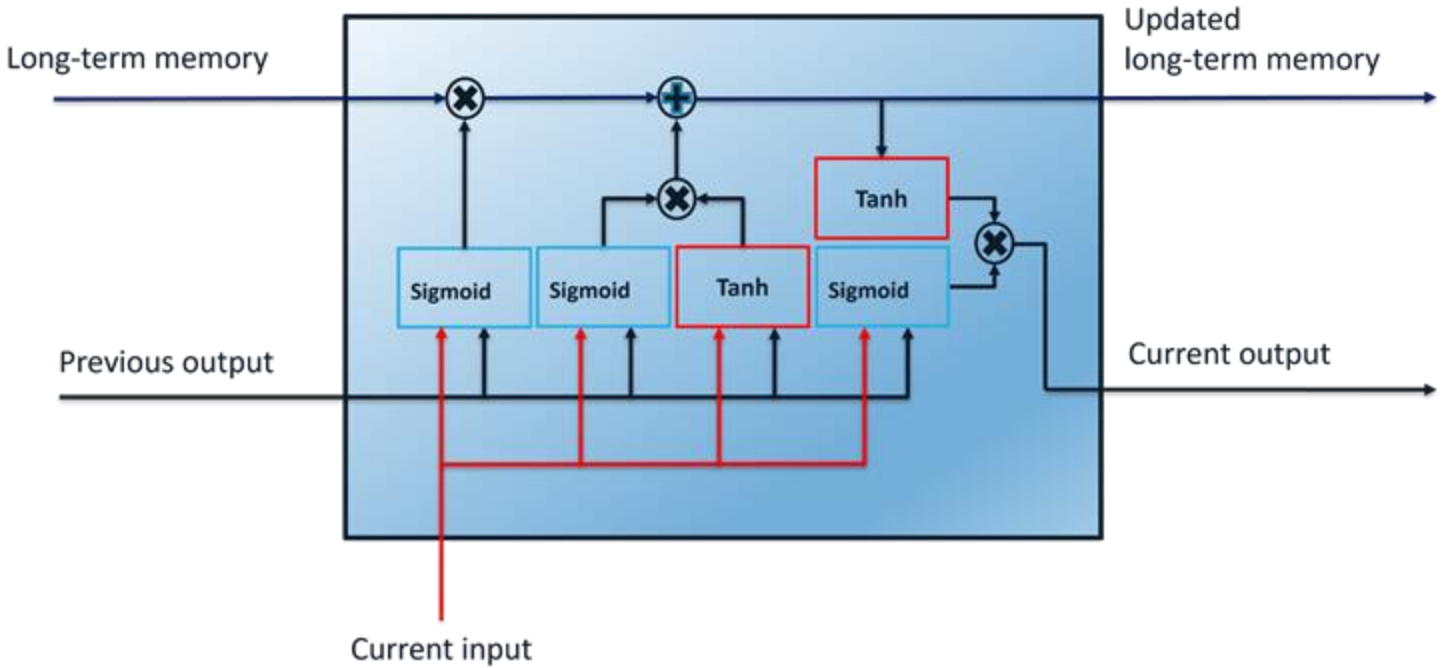
- Sigmoid ranges from 0 to 1.
  - This is used in LSTMs to determine the percentage of memory to store for the next step.
- Tanh ranges from -1 to 1.
  - This is used in LSTMs to calculate the value of memory to be stored. The value calculated will then be affected by the percentage of memory that is allowed to be stored.

Both of these functions working in relation to each other is a big part of how LSTMs handle the vanishing/exploding gradient problem.

Example

Take a look at the diagram of a LSTM model and follow the flow of the diagram to better understand the utility of the cell state and the sigmoid and tanh functions.

LSTM Diagram



Convolutional Neural Network (CNN)

- CNNs excel at image recognition and analysis.
- They leverage a special architecture with convolutional layers that can extract features from images.
- Applications include image classification (object detection, facial recognition), medical image analysis, self-driving cars.
- CNNs consist of three types of layers:
  - Convolutional layers
  - Pooling layers
  - Fully connected layers

Convolution

- A mathematical combination.
- Simply put, it is the dot product of two functions to produce a third function.
- We are focusing on the convolution of two vectors
  - The image and the filter that we use to detect patterns
- The convolution is calculated by reversing (flip horizontally) the second vector and sliding it across the first vector.
  - The dot product of the two vectors is calculated at each point as the second vector is slid across the first vector.
  - The dot products are collected in a third vector which is the convolution of the two vectors.

Convolutional Layers

- The building blocks of CNNs
- Responsible for extracting features from the input data.
- Convolutional layers apply a set of learnable filters (kernels) to the input data through convolution operations.
    - At the start, the types of kernels we use are quite simple and more geometric, detecting things such as edges, corners, and simple shapes and patterns (like a circle).
    - In later layers of the network, more complex kernels are incorporated that detect shapes, objects, and other complex structures.

Convolutional Layers

- The output of a convolutional layer is a feature map that highlights the presence of different features in the input.
- Each convolutional layer can have multiple kernels that produce multiple feature maps of their own.

Pooling Layers

- Pooling layers are used to downsample our feature maps
  - Keeping the most important parts and discarding the rest.
- Downsampling is done by reducing the dimension size of the image and using pooling operations to compress the feature map information to the smaller sized image.
  - This speeds up calculations done in later layers due to the reduced spatial size.
- These pooling layers also help control the model from overfitting to the training data.
  - Which helps the model focus on the general features instead of the finer characteristics that can change between images.

Pooling Operations

- Pooling layers operate similarly to convolutional layers with a sliding window where computation is done, but the computation of a pooling operation is simpler.
- Common pooling operations include max pooling and average pooling.
    - Max pooling retains the maximum value within a pooling window, preserving the most notable features.
    - Average pooling calculates the average value within a pooling window, providing a smoother downsampling of features.
- Pooling layers help in summarizing the feature map for more efficient performance down the line.

Max Pooling Example

Input

3	0	1	5	1	3
5	7	3	4	4	6
7	7	1	8	3	5
6	1	7	0	0	5
0	4	5	5	7	2
3	2	0	2	0	2

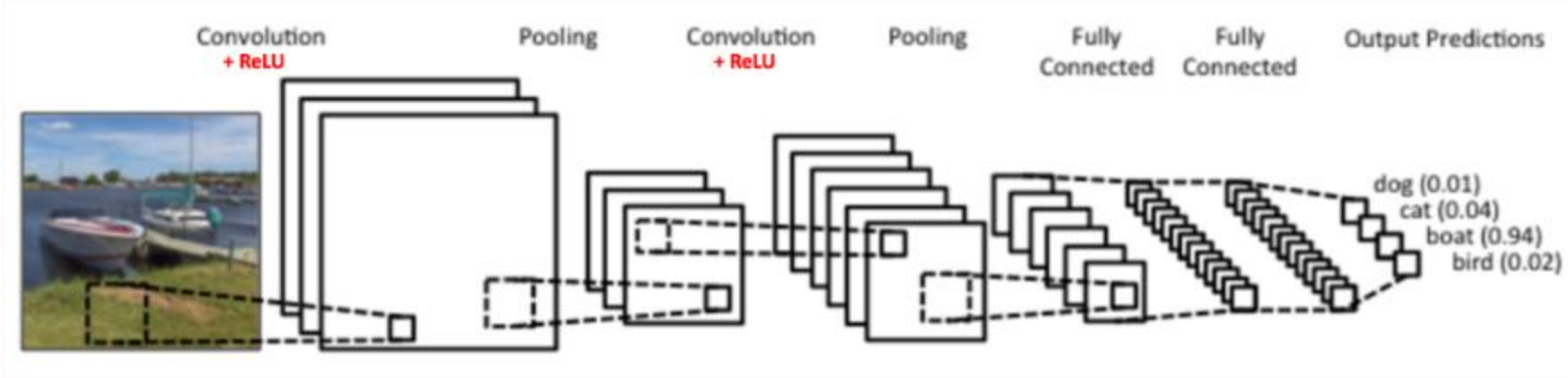
Output

7	5	6
7	8	5
4	5	7

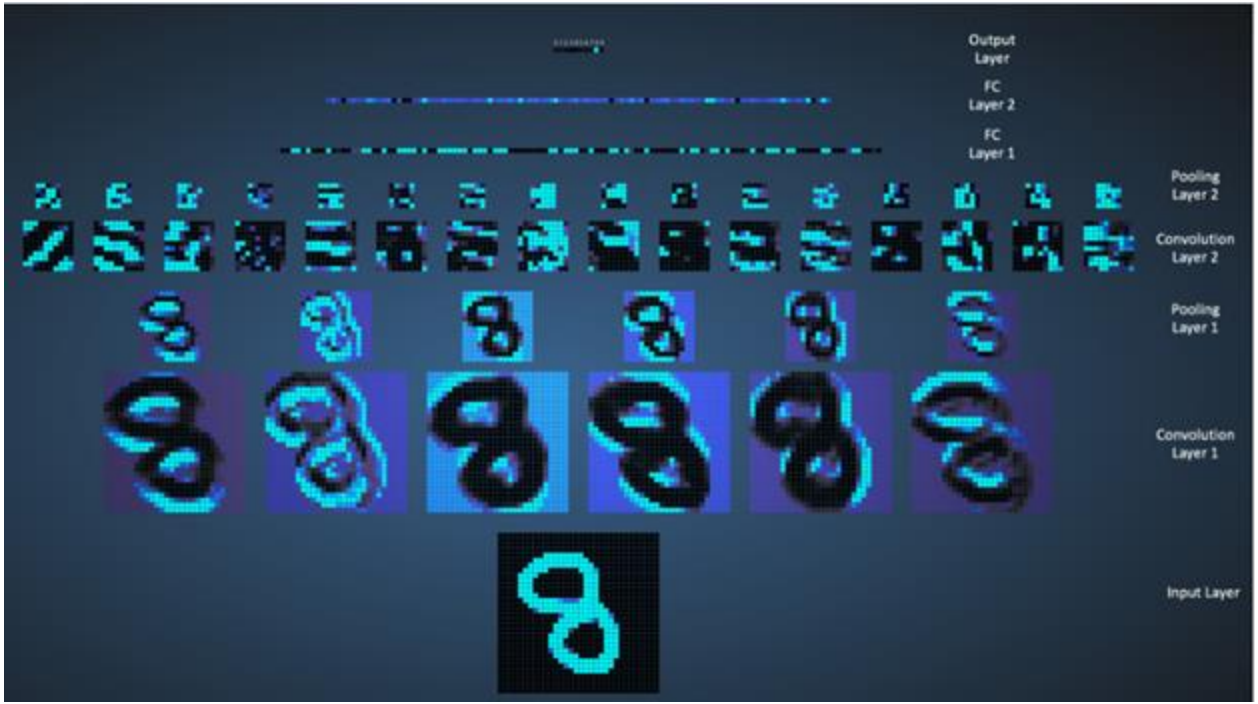
Fully Connected Layers

- Integrate the high-level features extracted by convolutional and pooling layers for classification or regression tasks.
- Fully connected form essentially a feed forward network
  - Layers connect every neuron in one layer to every neuron in the next layer.
  - The output of the fully connected layers undergoes activation functions such as softmax (for classification) or sigmoid (for binary classification).
- Fully connected layers enable CNNs to make predictions based on the learned representations of the input data.

CNN Diagram


Convolution
+ ReLU
Pooling
Convolution
+ ReLU
Pooling
Fully
Connected
Fully
Connected
Output Predictions
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)

CNN Diagram


Output Layer
FC Layer 2
FC Layer 1
Pooling Layer 2
Convolution Layer 2
Pooling Layer 1
Convolution Layer 1
Input Layer

Hyperparameters

While giving an overview of CNNs, I haven't covered a few factors that can affect the performance of these models. So, let's briefly cover them here.

- **Kernel size** refers to the dimensions (width and height) of the kernel.
  - Larger kernels can capture larger and more complex features in the image, while smaller ones are better at detecting smaller and more specific features.
- **Stride** controls how much the kernel moves after each convolution operation.
  - For a stride of 1, the kernel moves one pixel at a time. This results in a full convolution where every pixel is included in the operation.
  - For a stride of 2, the kernel jumps two pixels at a time. This leads to a downsampled output (fewer feature maps), which can be useful for dimensionality reduction.

Hyperparameters

- **Dilation rate** is a way to control the "density" of feature extraction without changing the kernel size.
  - In standard convolution, the filter elements touch each other (dilation rate of 1).
  - With dilation rates bigger than 1, spaces are inserted between filter elements, allowing for a sparse convolution.
  - This can be helpful for capturing features at different resolutions or avoiding checkerboard artifacts that can sometimes occur with striding.
- **Padding** refers to adding extra pixels (usually zeros) around the borders of the input image.
  - This can be useful to control the output size of the convolution operation.
  - Without padding, striding might shrink the output significantly.
  - With padding, you can control the output size to be the same as the input size.

Example

Walk through the CNN example using the MNIST dataset and compare the results to the Feed Forward model example.

Exercise

Come up with three examples of complex feature maps that would be used in later convolutional layers that would help detect specific objects in images. What would its real world application be used for?

Supervised vs. Unsupervised Learning

There are two common approaches to how a neural network learns. Most models typically are designed for either supervised or unsupervised learning.

We've been covering models that are designed for **supervised learning**

- Relies on labeled data.
  - Paired input and output data that is used during training to calculate the cost.
- This is kind of like learning with a teacher who provides guidance and correct answers to help you understand concepts and solve problems.

Unsupervised Learning

- Deals with unlabeled data.
  - The data points don't have predefined labels or categories.
- The goal is to uncover hidden patterns or structures within the data itself.
- This is kind of like exploring a new environment on your own, identifying patterns, relationships, and interesting features without a predefined goal or map.

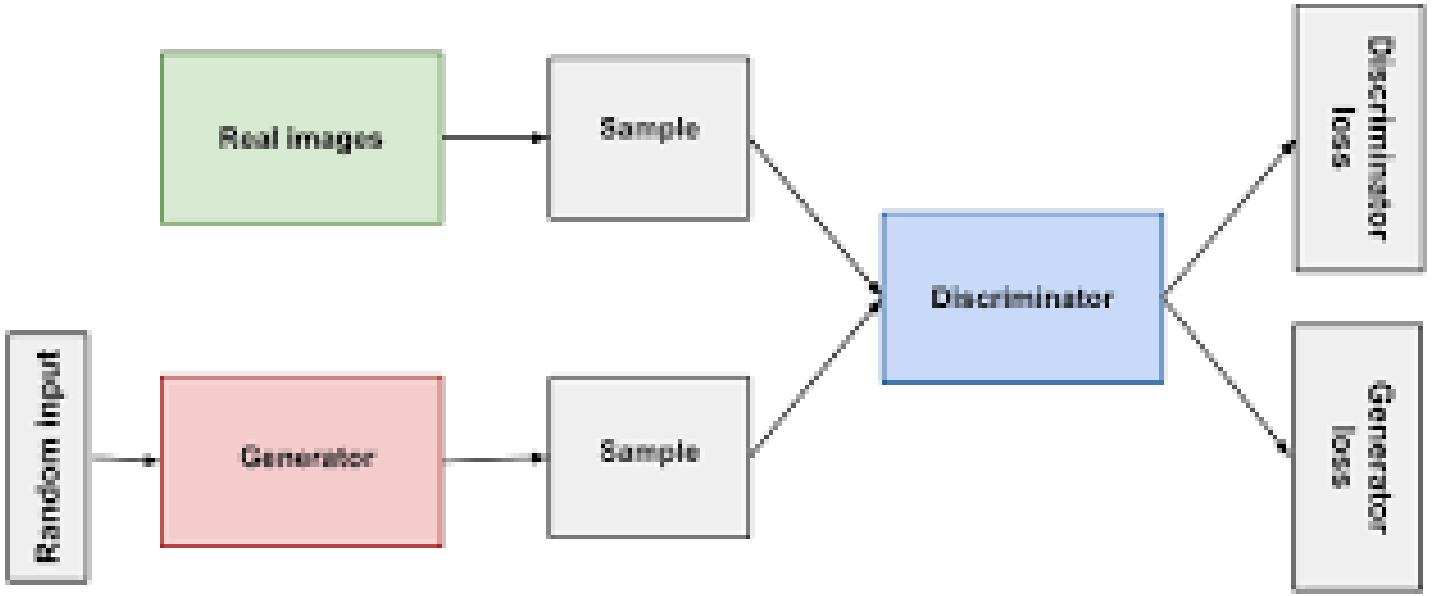
Generative Adversarial Networks (GAN)

- GANs are an example of unsupervised learning.
- Applications include generating realistic images (e.g., creating new faces), creating artistic styles, text generation.
- They are a fascinating concept where two neural networks compete with each other.
    - These two models are described as the generator and the discriminator.
    - GAN training involves a minimax game, where the generator and discriminator are trained simultaneously.
    - A minimax game is essentially where both opponents are trying to minimize the other opponent's maximum payoff.

GAN's Minimax Training Game

- The generator network creates new data samples, like images or music, that mimic the real world data distribution.
- The discriminator network tries to distinguish between real data (from the training dataset) and the generated samples.
    - The discriminator deals with labeled data at this point (the label being whether the data is generated or real).
    - This allows the discriminator model to produce a cost function which is used to optimize both the generator and the discriminator to perform better.
- Through this competition, the generator learns to create increasingly realistic data that can fool the discriminator, while the discriminator improves its ability to distinguish real from generated samples.

GAN Diagram


Real images
Sample
Random input
Generator
Sample
Discriminator
Discriminator loss
Generator loss

Take Home Exercise

What is the potential societal impact and ethical considerations of GAN technology? Find one or two examples of AI generated media that have been used to spread misinformation.