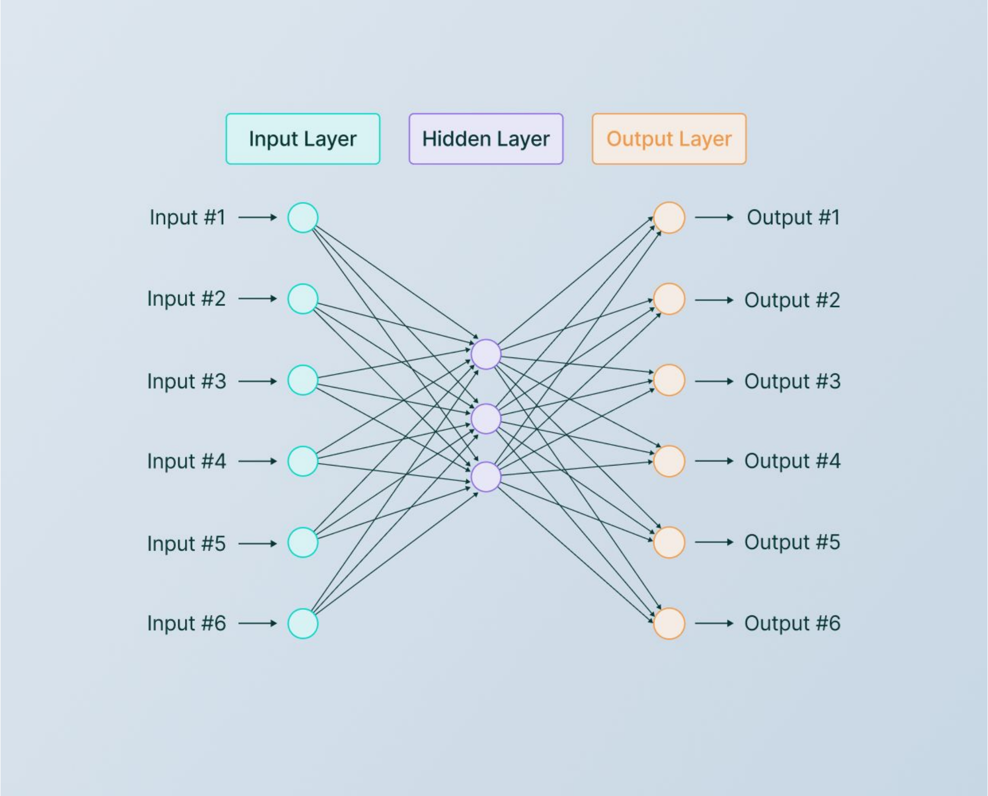
TA8

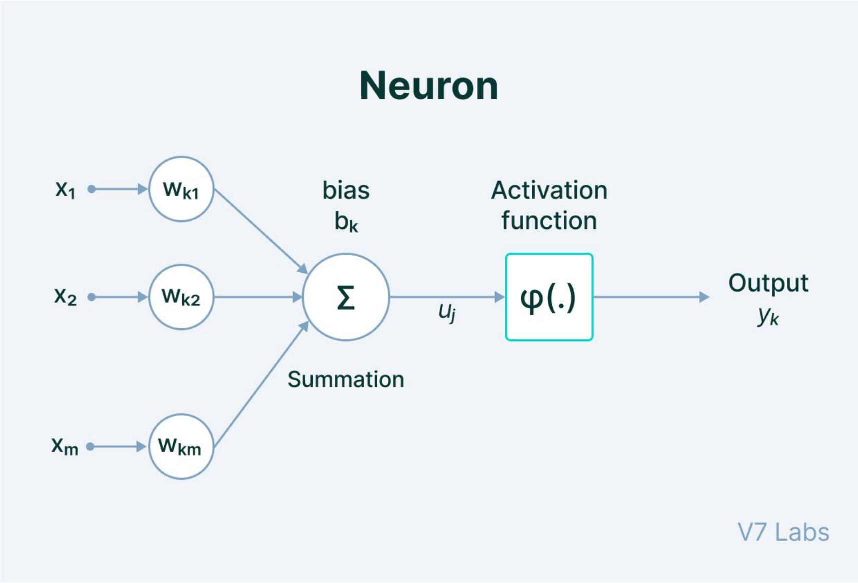
Arun Kumar Rajasekaran

Let’s start with NNs



Neurons

The Neural Network architecture is made of individual units called *neurons* that mimic the biological behavior of the brain.



Components of a neuron

Input - It is the set of features that are fed into the model for the learning process. For example, the input in object detection can be an array of pixel values pertaining to an image.

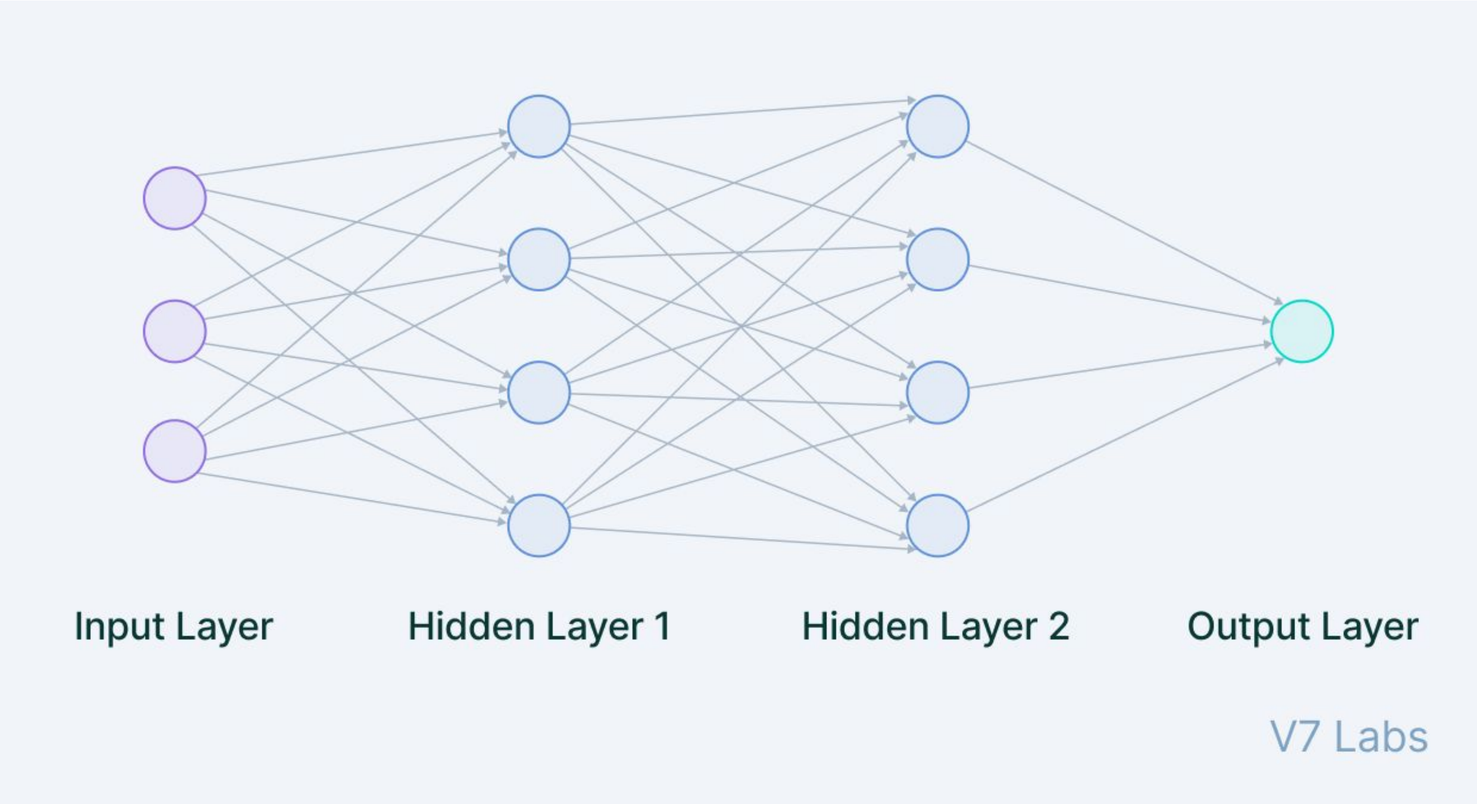
Weight - Its main function is to give importance to those features that contribute more towards the learning. It does so by introducing scalar multiplication between the input value and the weight matrix. For example, a negative word would impact the decision of the sentiment analysis model more than a pair of neutral words.

Transfer function - The job of the transfer function is to combine multiple inputs into one output value so that the activation function can be applied. It is done by a simple summation of all the inputs to the transfer function.

Activation Function—It introduces non-linearity in the working of perceptrons to consider varying linearity with the inputs. Without this, the output would just be a linear combination of input values and would not be able to introduce non-linearity in the network.

Bias - The role of bias is to shift the value produced by the activation function. Its role is similar to the role of a constant in a linear function.

Lets stack up ….



Lets stack up ….

Input Layer

The data that we feed to the model is loaded into the input layer from external sources like a CSV file or a web service. It is the only visible layer in the complete Neural Network architecture that passes the complete information from the outside world without any computation.

Hidden Layers

The hidden layers are what makes deep learning what it is today. They are intermediate layers that do all the computations and extract the features from the data. There can be multiple interconnected hidden layers that account for searching different hidden features in the data. For example, in image processing, the first hidden layers are responsible for higher-level features like edges, shapes, or boundaries. On the other hand, the later hidden layers perform more complicated tasks like identifying complete objects (a car, a building, a person).

Output Layer

The output layer takes input from preceding hidden layers and comes to a final prediction based on the model's learnings. It is the most important layer where we get the final result.

In the case of classification/regression models, the output layer generally has a single node. However, it is completely problem-specific and dependent on the way the model was built.

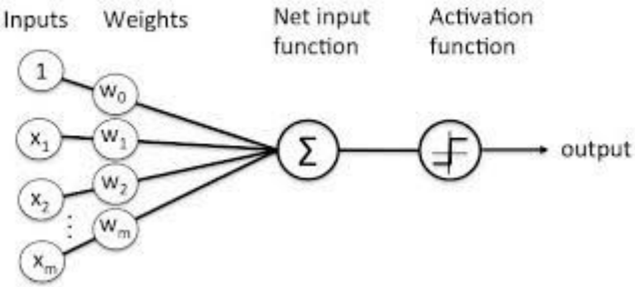
The perceptron

Perceptron is the simplest Neural Network architecture.

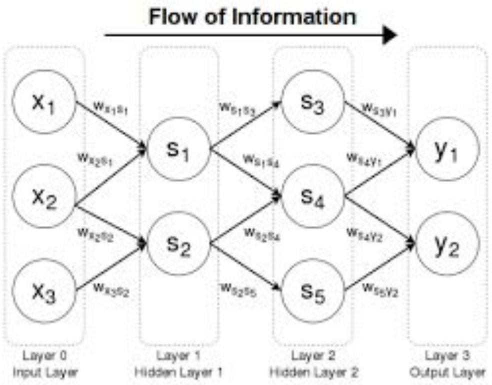
It is a type of Neural Network that takes a number of inputs, applies certain mathematical operations on these inputs, and produces an output. It takes a vector of real values inputs, performs a linear combination of each attribute with the corresponding weight assigned to each of them.

The weighted input is summed into a single value and passed through an activation function.

These perceptron units are combined to form a bigger Artificial Neural Network architecture.



Feed forward NN



Perceptron represents how a single neuron works.

But—

What about a series of perceptrons stacked in a row and piled in different layers? How does the model learn then?

It is a multi-layer Neural Network, and, as the name suggests, the information is passed in the forward direction—from left to right.

In the forward pass, the information comes inside the model through the input layer, passes through the series of hidden layers, and finally goes to the output layer. This Neural Networks architecture is forward in nature—the information does not loop with two hidden layers.

The later layers give no feedback to the previous layers. The basic learning process of Feed-Forward Networks remain the same as the perceptron.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks is a type of Feed-Forward Neural Networks used in tasks like image analysis, natural language processing, and other complex image classification problems.

A CNN has hidden layers of convolutional layers that form the base of ConvNets.

Features refer to minute details in the image data like edges, borders, shapes, textures, objects, circles, etc.

At a higher level, convolutional layers detect these patterns in the image data with the help of filters. The higher-level details are taken care of by the first few convolutional layers.

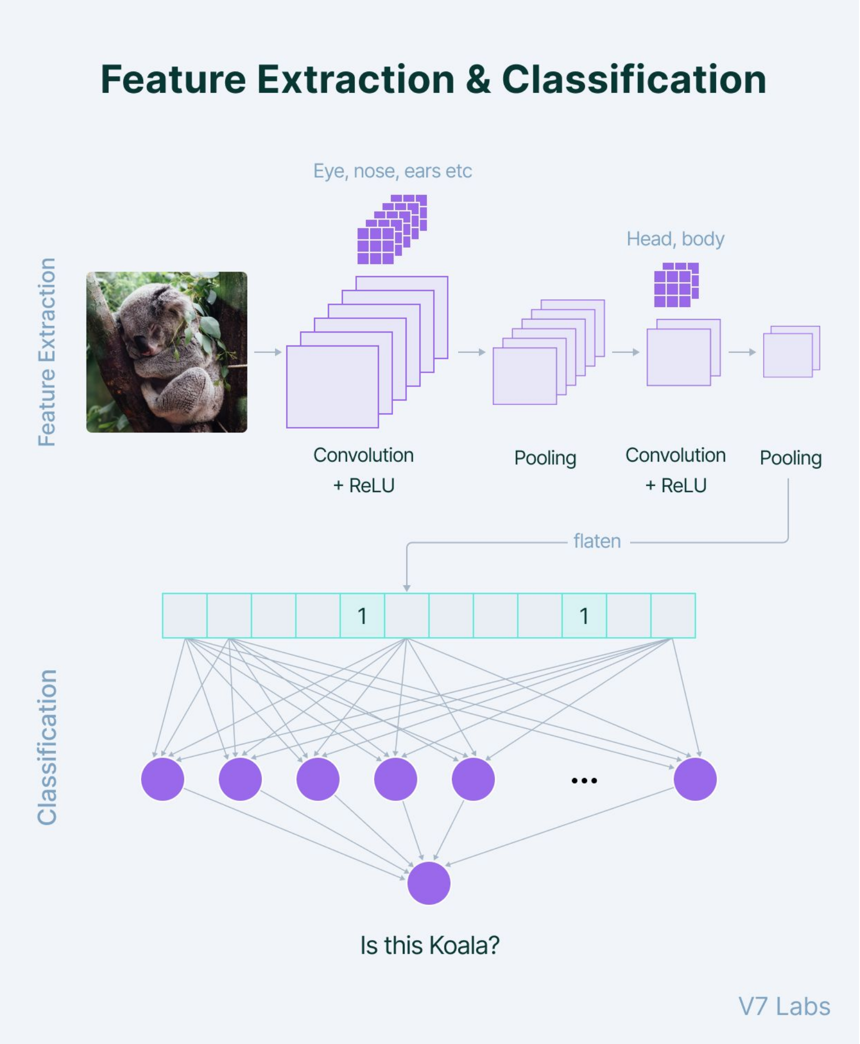
The deeper the network goes, the more sophisticated the pattern searching becomes.

A *filter* can be thought of as a relatively small matrix for which we decide the number of rows and columns this matrix has.

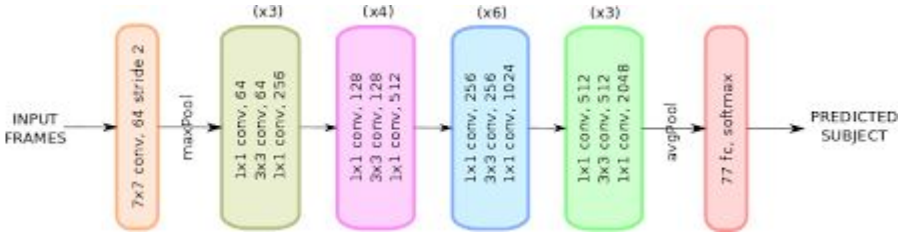
The value of this feature matrix is initialized with random numbers. When this convolutional layer receives pixel values of input data, the filter will convolve over each patch of the input matrix.

The output of the convolutional layer is usually passed through the ReLU activation function to bring non-linearity to the model. It takes the feature map and replaces all the negative values with zero.

Pooling is a very important step in the ConvNets as it reduces the computation and makes the model tolerant towards distortions and variations. A Fully Connected Dense Neural Networks would use a flattened feature matrix and predict according to the use case.



Residual Networks (ResNets)



How to decide on the number of layers in our neural network architecture?

A naive answer would be: *The greater the number of hidden layers, the better is the learning process.*

More layers enrich the levels of features.But—

Very deep Neural Networks are extremely difficult to train due to vanishing and exploding gradient problems.

ResNets provide an alternate pathway for data to flow to make the training process much faster and easier.

This is different from the feed-forward approach of earlier Neural Networks architectures.

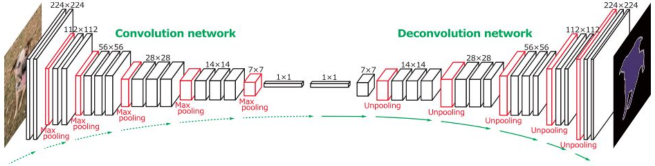
The core idea behind ResNet is that a deeper network can be made from a shallow network by copying weight from the shallow counterparts using identity mapping.

The data from previous layers is fast-forwarded and copied much forward in the Neural Networks. This is what we call *skip connections* first introduced in Residual Networks to resolve vanishing gradients.

The Deconvolutional Neural Networks (DNN)

Deconvolutional Neural Networks are CNNs that work in a reverse manner.

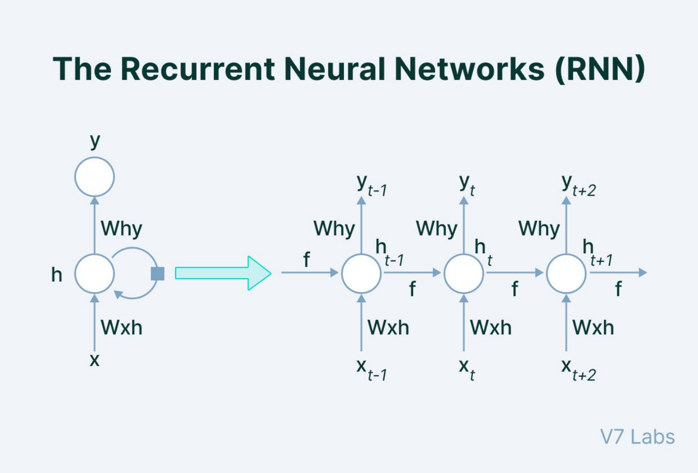
When we use convolutional layers and max-pooling, the size of the image is reduced. To go to the original size, we use upsampling and transpose convolutional layers. Upsampling does not have trainable parameters—it just repeats the rows and columns of the image data by its corresponding sizes.



Transpose Convolutional layer means applying convolutional operation and upsampling at the same time. It is represented as Conv2DTranspose (number of filters, filter size, stride). If we set stride=1, we do not have any upsampling and receive an output of the same input size.

The Recurrent Neural Networks (RNN)

Recurrent Neural Networks have the power to remember what it has learned in the past and apply it in future predictions.



V7 Labs

LSTM

In RNN each of our predictions looked only one timestamp bac

doesn't use any information from further back.

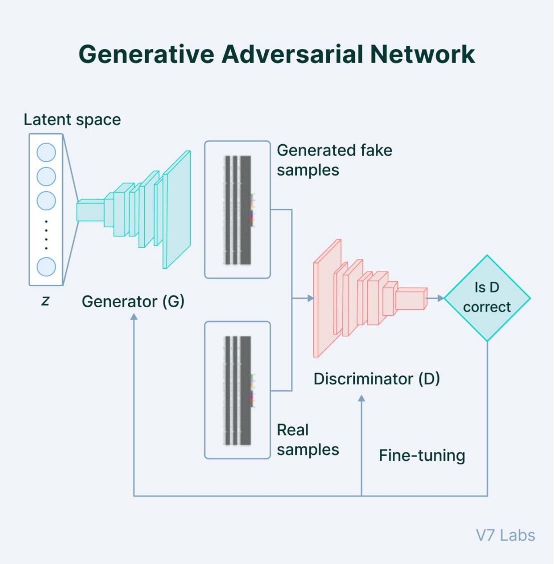
To rectify this, we can take our Recurrent Neural Networks stru

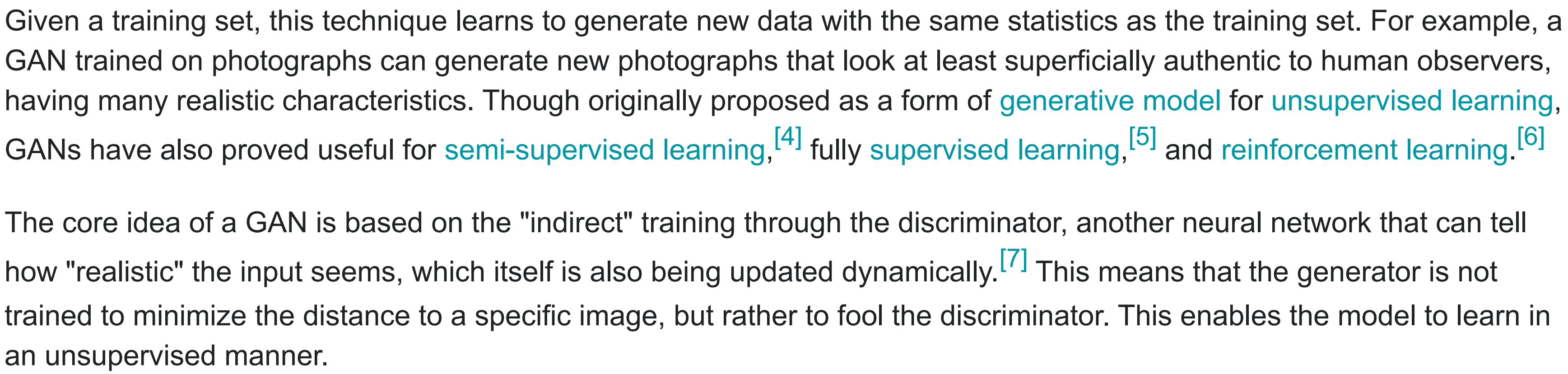
pieces to it.

Generative Adversarial Network (GAN)

Generative modeling comes under the umbrella of unsupervised learning, where new/synthetic data is generated based on the patterns discovered from the input set of data.

GAN is a generative model and is used to generate entirely new synthetic data by learning the pattern and hence is an active area of AI research.





Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Though originally proposed as a form of generative model for unsupervised learning, GANs have also proved useful for semi-supervised learning,[4] fully supervised learning,[5] and reinforcement learning.[6]

The core idea of a GAN is based on the "indirect" training through the discriminator, another neural network that can tell how "realistic" the input seems, which itself is also being updated dynamically.[7] This means that the generator is not trained to minimize the distance to a specific image, but rather to fool the discriminator. This enables the model to learn in an unsupervised manner.

Generator/discriminator

The generator’s job is to create synthetic data based on the model’s features during its learning phase. It takes in random data as input and returns a generated image after performing certain transformations.

The discriminator acts as a critic and has an overall idea of the problem domain with a clear understanding of generated images.

The generator network produces samples based on its learning.

Its adversary, the discriminator, strives to distinguish between samples from the training data and samples produced from the generator. There is feedback from the discriminator fed to the generator to improve the performance.

It is used in scenarios like predicting the next frame in a video, text to image generation, image to image translation like style transfer, denoising of the image, etc.

Breakout

- What are some ideal things you would like to do in the next two summer/winter breaks?
- What would be your ideal college/ college major/college experience would look like?
- (^_^) How will your ideal career look like…

Transformers ………….

Why?

RNNs are slow and take too much time in training.

They are not very good with large sequenced data and lead to vanishing gradients. LSTMs that were introduced to bring memory in the RNN became even slower to train.

For both RNN and LSTM, we need to feed the data sequentially or serially. This does not make use of GPUs.

How to parallelize the training on sequential data?

The answer is Transformers.

These networks employ an encoder-decoder structure with a difference that the input data can be passed parallelly.

In RNN structure, one word at a time was passed through the input layer. But in Transformers, there is no concept of timestamps for passing the input. We feed the complete sentence together and get the embeddings for all the words together.

“RNNs had a drawback of not using parallel computing and loss of critical information through the sequenced time stamped data. In contrast, Transformers are based on Attention that require a single step to feed all the sequential data and have a self-attention mechanism working in the core architecture to preserve important information.”

BERT (*Bidirectional Encoder Representations from Transformer*s)

These models are faster as the words can be processed simultaneously. The context of words is better learned as they can learn from both directions simultaneously. If we stack the encoders, we get the BERT model.

Masked Language Modeling: BERT takes input sentences and replaces some random words with [MASK] tokens. The goal of the model is to predict the original word of the masked words based on the context provided by the other, non-masked, words in the sequence.

Next Sentence Prediction: In this case, BERT take the input of two sentences, and it determines if the second sentence follows the first sentence.

When training the BERT model, Masked LM and Next Sentence Prediction are trained together to minimize the combined loss function of the two strategies and get a good understanding of the language.

GPT; GPT2; GPT3

GPT (Generative PreTraining) is a language model used to predict the probability of the sequence of words.

Language models involving generative training do not require human-labeled data.

GPT-1 has two steps of training—unsupervised pre-training using unlabeled data with language model objective function followed by supervised fine-tuning of the model without a task-specific model. GPT uses transformer decoder architecture.

With GPT2, the purpose of the model shifted more to the text generation side. It is an autoregressive language model. It is trained on an input sequence and its target is predicting the next token at each point of the sequence.

It consists of a single stack of transformer blocks with an attention mechanism. It has slightly lower dimensionality than BERT, with more transformer blocks(48 blocks) and a larger sequence length.

The basic structure of GPT3 is similar to that of GPT2, with the only difference of more transformer blocks(96 blocks) and is trained on more data. The sequence size of input sentences also doubled as compared to GPT2. It is by far the largest neural network architecture containing the most number of parameters.

Coming back to back propagation NNs

Backpropagation Artificial Neural Network (ANN) is a supervised learning algorithm used to train a feedforward neural network. It is the most popular and widely used algorithm for training artificial neural networks.

Backpropagation is a learning algorithm that adjusts the weights and biases of the neural network based on the error between the predicted output and the actual output. The algorithm works by propagating the error backwards through the network, from the output layer to the input layer, and adjusting the weights and biases of each neuron along the way.

Simplified working

The backpropagation algorithm consists of two phases: forward propagation and backward propagation. During forward propagation, the input is fed into the neural network, and the network calculates the output. During backward propagation, the error between the predicted output and the actual output is calculated, and the weights and biases of each neuron are adjusted to reduce the error.

The backpropagation algorithm uses the gradient descent optimization method to update the weights and biases of the network. The gradient descent method calculates the gradient of the error function with respect to the weights and biases, and then updates the weights and biases in the direction of the negative gradient, which reduces the error.

To do

Compile eng design/optimization examples that uses back propagation network? And why?