



D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

Advanced AI



Dr. Vaishnav G. Kale
Associate Professor

Department of Computer Science
Engineering & Applications

D.Y. Patil International University, Pune

About the Course

Name of the subject:Advanced AI

Course Code:MCA 301

Programme:Master in Computer Applications

Year of study:Second Year

Semester :III

Specialization:Artificial Intelligence and Data Science

Course Type:Specialization Course

Academic Year:2023-24

Syllabus

Module-IV:Advanced AI Models

Transformer-based Language Models:GPT-3 (Generative Pre-trained Transformer 3)

Generative Models: GAN ,HMM, Auto regressive model, Applications, Evaluation and challenges of generative models

Deep Generative Models :Deep Learning, Advanced DNN, Recurrent Neural Networks (RNNs) ,CNN,LSTM, GRU, GAN

Books

Sr.No.	Text Books	Name of the Author
1	“Artificial Intelligence: A Modern Approach”	Stuart Russell and Peter Norvig
2	“A First Course in Artificial Intelligence”	Deepak Khemani
3	“Artificial Intelligence”	Elaine Rich, Kevin Knight and Nair
4	“Deep Learning”	Ian Goodfellow The MIT Press
Sr.No.	Reference Books	Name of the Author
1	“Artificial Intelligence: A new Synthesis”	Nilsson Nils J
2	“Artificial Intelligence”	Patrick Henry Winston
3	“Computational Intelligence: An Introduction”	Andries P. Engelbrecht
4	“Artificial Intelligence: Concepts and Applications”	Dr. Lavika Goel

Advanced Artificial Intelligence

Module 04. Advanced AI Models

Transformer based Language Models

Transformer

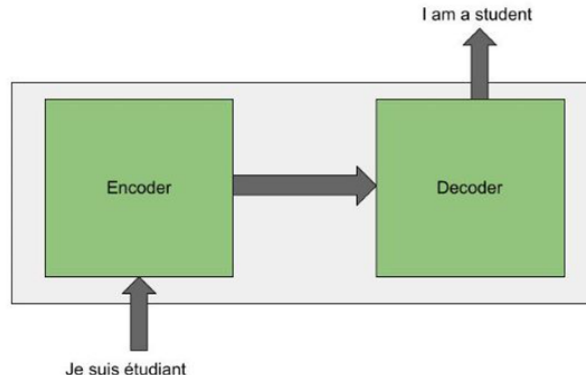
Ex: There is a word - ' **Point** ', and we use it in two different contexts given below

- The needle has a sharp **point**.
- It is not polite to **point** at people.
- Here, the word '**Point**' has two different contexts in both of the sentences, but when embedding is done the context is not taken into consideration.
- Therefore, there was a need for a different architecture — Transformer.
- Neural networks have been classified into mainly two groups, that are — **Feedforward** and **Feedback**.
- Transformers are based upon feedforward networks, which means that the information moves from the input to the output and it does not contain a feedback loop.

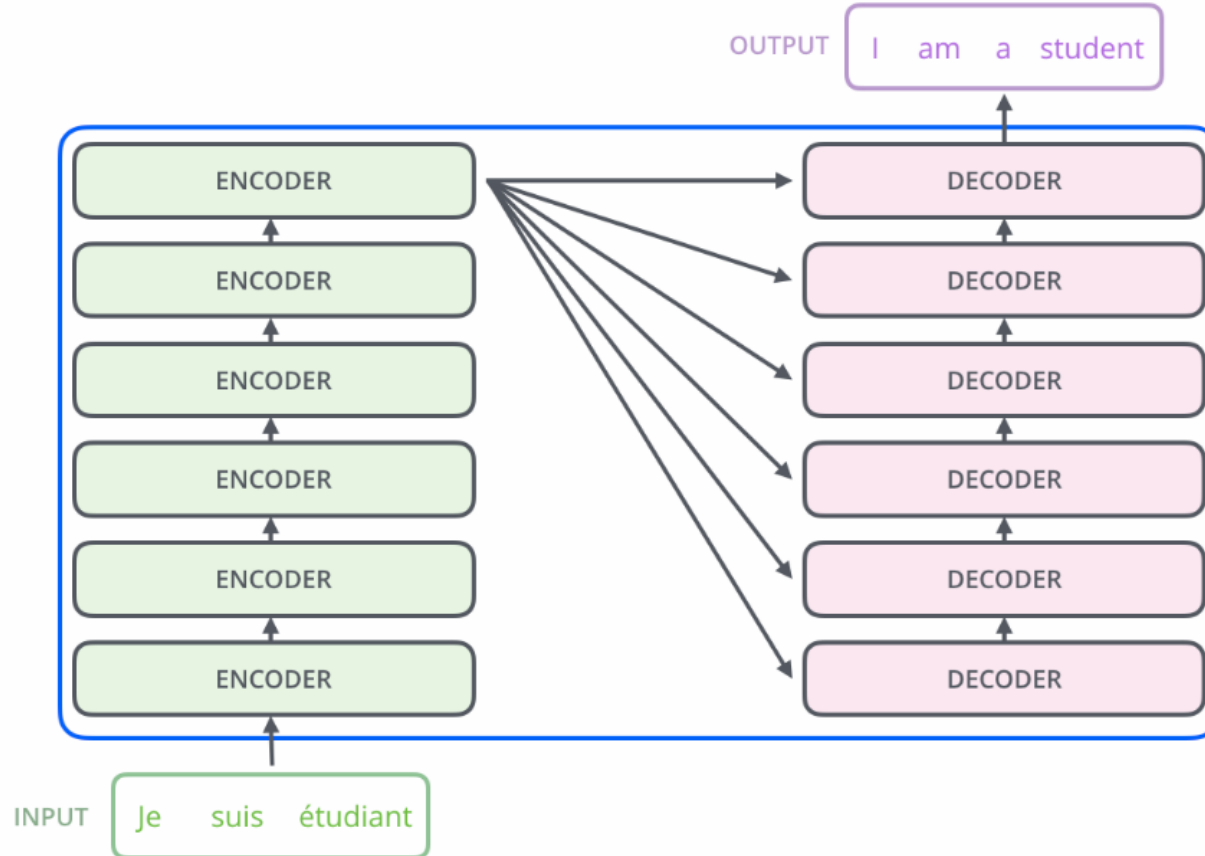
Transformer based Language Models

Transformer

- In contrast to it, feedback networks, which means that the information can pass to both directions and it consists of a feedback path i.e we can again make use of the memory for new predictions.
- Now, coming to the architecture of the Transformer. **Encoder** and **Decoder** are building blocks of a Transformer.
- The encoder block turns the sequence of input words into a vector and a Decoder converts a vector into a sequence

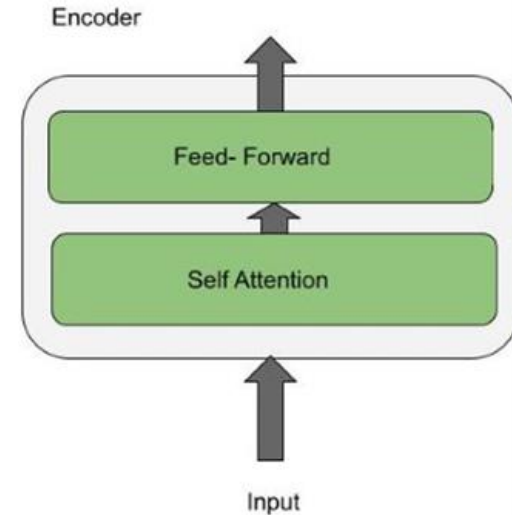


Transformer based Language Models



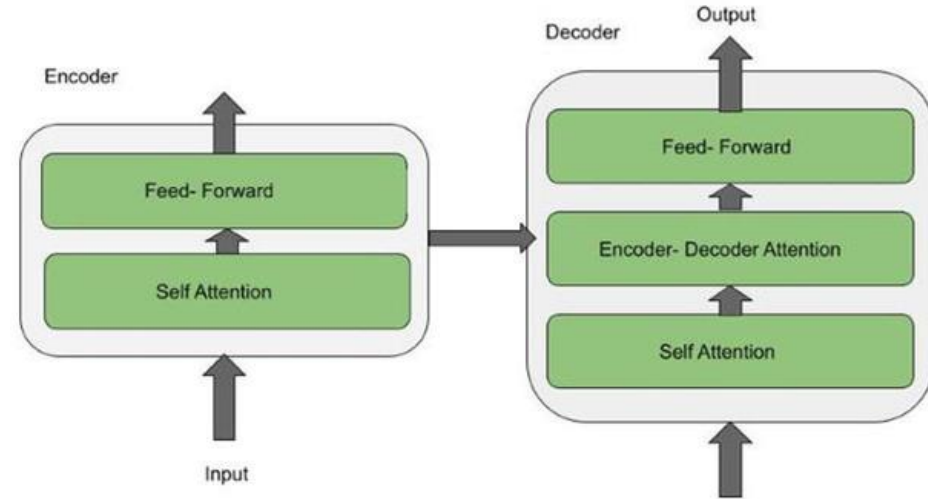
Transformer based Language Models

- The encoder architecture has two layers: **Self Attention** and **Feed Forward**.
- The encoder's inputs first pass by a self-attention layer and then the outputs of the self-attention layer are fed to a feed-forward neural network.
- Sequential data has temporal characteristics.
- It signifies that each word holds some position concerning the other.
- For example, let's take a sentence- 'The cat didn't chase the mouse, because it was not hungry'. Here, we can easily tell that 'it' is referring to the cat, but it is not as simple for an algorithm.
- When the model is processing the word 'it', self-attention allows it to associate 'it' with 'cat'.
- Self-attention is the method to reformulate the representation based on all other words of the sentence.



Transformer based Language Models

- The decoder architecture has three layers: Self Attention, Encoder-decoder attention, and Feed Forward.
- The decoder has both the self-attention and feed-forward layer which is also present in the encoder, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence.

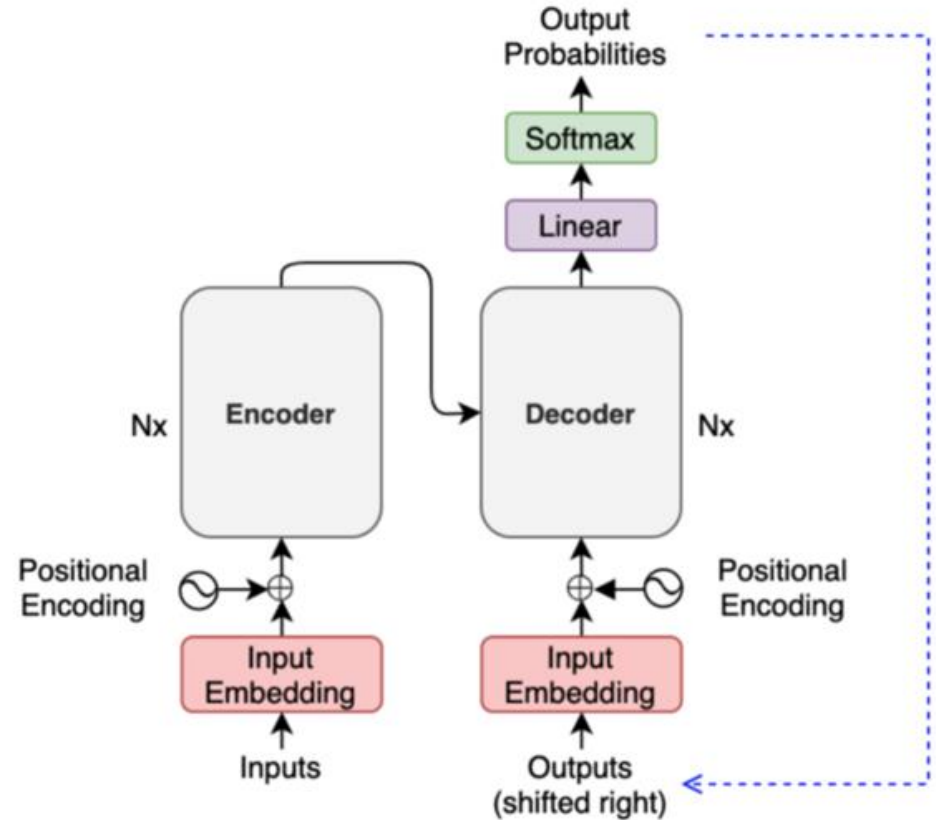




Transformer based Language Models

Complete Block diagram of Transformer

<http://jalammar.github.io/illustrated-transformer/>



Transformer based Language Models

Advantages of Transformer

1. They hold the potential to understand the relationship between sequential elements that are far from each other.
2. They are way more accurate.
3. They pay equal attention to all the elements in the sequence.
4. Transformers can process and train more data in lesser time.
5. They could work with virtually any kind of sequential data.
6. Transformers serve to be helpful in anomaly detection.

Transformer based Language Models

Applications of Transformer

- Translating from one language to another.
- Programming more engaging and useful chatbots.
- Summarizing long documents.
- Generating a long document from a brief prompt.
- Generating drug chemical structures based on a particular prompt.
- Generating images from a text prompt.
- Creating captions for an image.
- Creating a robotic process automation ([RPA](#)) script from a brief description.
- Providing code completion suggestions based on existing code.

Transformer based Language Models

Transformer Model Implementations

- Google's Bidirectional Encoder Representations from Transformers was one of the first LLMs based on transformers.
- **OpenAI's** GPT followed suit and underwent several iterations, including GPT-2, GPT-3, GPT-3.5, **GPT-4** and Chat GPT.
- Meta's Llama achieves comparable performance with models 10 times its size.
- Google's **Pathways Language Model** generalizes and performs tasks across multiple domains, including text, images and robotic controls.
- Open AI's Dall-E **creates images from a short text description**.

GPT3

- GPT-3 is the third version of OpenAI's family of Generative Pre-Trained models.
- GPT-1 and GPT-2 laid the foundations for GPT-3, proving the success of two key hypotheses:
- Transformers+unsupervised pre-training works fine (GPT-1) and language models can multitask (GPT-2).
- GPT-3 is a language model based in the transformer architecture, pre-trained in a generative, unsupervised manner that shows decent performance in zero/one/few-shot multitask settings.
- It works by predicting the next token given a sequence of tokens and can do so for NLP tasks it hasn't been trained on.
- GPT-3 was trained with huge Internet text datasets — 570GB in total.
- When it was released, it was the largest neural network with 175 billion parameters (100x GPT-2), 96 attention layers, and a 3.2 M batch size.

GPT3

- The most impressive feature of GPT-3 is that it's a meta-learner; it has learned to learn.
- You can ask it in natural language to perform a new task and it “understands” what it has to do
- specialized version of GPT-3 would be different than the GPT-3 of any other user.
- That's the power of prompting+meta-learning; without changing the original model, users can make GPT-3 a specialist for different tasks.
- Baseline GPT-3 doesn't know how to perform any task, it knows how to learn to do it, which makes it more powerful and versatile.
- GPT-3 is a neural network **machine learning** model trained using internet data to generate any type of text.
- It requires a small amount of input text to generate large volumes of relevant and sophisticated machine-generated text.

How GPT works

- GPT (Generative Pre-trained Transformer) is a type of language model that utilizes a deep learning architecture known as a Transformer to generate human-like text.

1) Data Collection and Preprocessing:

- GPT is trained on a massive amount of text data from the internet, books, articles, and more.
- The data is preprocessed to remove noise, tokenize the text into smaller units (usually words or subwords), and create a dataset for training.

2) Architecture:

- GPT employs a Transformer architecture, which consists of an encoder and a decoder.
- However, in the case of GPT, only the decoder portion is used, as GPT is focused on generating text rather than translating it.

How GPT works

3) Self-Attention Mechanism:

- The heart of the Transformer architecture is the self-attention mechanism.
- It allows the model to weigh the importance of different words in a sentence based on their relationships.
- Self-attention helps the model understand context and long-range dependencies.

4) Positional Encoding:

- Since Transformers don't have an inherent sense of word order, positional encodings are added to the input embeddings.
- These encodings provide information about the position of each word in the sequence.

How GPT works

5) Layer Stacking:

- The decoder consists of multiple layers, each containing a self-attention sub-layer and feedforward neural network sub-layers.
- The model uses multiple stacked layers to capture increasingly complex patterns in the data.

6) Training:

- GPT is pre-trained in an unsupervised manner.
- It learns to predict the next word in a sentence by masking out some words and having the model predict them based on the context.
- The model aims to minimize the difference between its predictions and the actual next words.

How GPT works

7) Fine-Tuning:

- After pre-training, GPT can be fine-tuned on specific tasks.
- For example, it can be fine-tuned for language translation, text completion, question answering, and more.
- During fine-tuning, the model is trained on task-specific data with labeled examples to adapt its knowledge.

8) Text Generation:

- When generating text, you provide a prompt to the trained GPT model.
- The model takes the prompt and generates a continuation of the text one word at a time.
- It samples from its learned probability distribution of words to predict the next word based on the context of the prompt and the generated text so far.

How GPT works

9) Output Sampling:

- GPT often uses techniques like temperature control and nucleus sampling to adjust the randomness of generated text.
- Temperature control influences the randomness of word selection, and nucleus sampling focuses on selecting from a subset of the most likely words based on their cumulative probabilities.

10) Repetition and Coherence:

- While GPT is excellent at generating coherent and contextually relevant text, it can sometimes produce repetitive or nonsensical outputs.
- Post-processing techniques and careful prompting can be used to improve the quality of generated text.

How GPT works

- However, in a decoder-only transformer like GPT, the attention mechanism is “masked” to prevent it from looking at future parts of the input when generating each part of the output.
- This is necessary because GPT models are trained to predict the next word in a sentence, so they should not have access to future words.
- The decoder-only architecture simplifies the model and makes it more efficient for certain tasks, like language modeling.
- By removing the encoder, GPT models can process input data more directly and generate output more quickly.
- This architecture also allows GPT models to be trained on a large amount of unlabeled data, which is a significant advantage in the field of NLP where labeled data is often scarce.

How GPT works

- GPT's decoder-only architecture is a powerful and efficient alternative to the traditional encoder-decoder model.
- It simplifies the model, makes it more efficient, and allows it to be trained on a large amount of unlabeled data.
- Despite not having an encoder, GPT models are still capable of performing tasks typically associated with encoder-decoder models, due to the power of the transformer's decoder and the training method used.

Language Models

- Language modeling (LM) is the use of various statistical and probabilistic techniques to determine the probability of a given sequence of words occurring in a sentence.
- Language models analyze bodies of text data to provide a basis for their word predictions.
- So simply put, a Language Model predicts the next word(s) in a sequence.
- Language models have many applications like:
 - Part of Speech (PoS) Tagging
 - Machine Translation
 - Text Classification
 - Speech Recognition
 - Information Retrieval
 - News Article Generation
 - Question Answering, etc.

How Language Models work

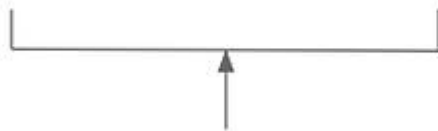
- For training a language model, a number of probabilistic approaches are used.
- These approaches vary on the basis of the purpose for which a language model is created.
- The amount of text data to be analyzed and the math applied for analysis make a difference in the approach followed for creating and training a language model.
- Consider an arbitrary language L .
- In this case, English will be utilized to simplify the arbitrary language.
- A language model assigns probabilities to sequences of arbitrary symbols such that the more likely a sequence (w_1, w_2, \dots, w_n) is to exist in that language, the higher the probability.
- A symbol can be a character, a word, or a sub-word (e.g. the word 'going' can be divided into two sub-words: 'go' and 'ing').



How Language Models works

$$\begin{aligned} P(w_1, w_2, \dots, w_n) &= p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)\dots p(w_n|w_1, w_2, \dots, w_{n-1}) \\ &= \prod_{i=1}^n p(w_i|w_1, \dots, w_{i-1}) \end{aligned} \quad (1)$$

S = Where are we going



Previous words
(Context)



Word being
predicted

$$P(S) = P(\text{Where}) \times P(\text{are} \mid \text{Where}) \times P(\text{we} \mid \text{Where are}) \times P(\text{going} \mid \text{Where are we})$$

GPT3

Limitations

- **GPT-3 lacks long-term memory** — the model does not learn anything from long-term interactions like humans.
- **Lack of interpretability** — this is a problem that affects extremely large and complex in general. GPT-3 is so large that it is difficult to interpret or explain the output that it produces.
- **Limited input size** — transformers have a fixed maximum input size and this means that prompts that GPT-3 can deal with cannot be longer than a few sentences.
- **Slow inference time** — because GPT-3 is so large, it takes more time for the model to produce predictions.
- **GPT-3 suffers from bias**

Generative Models

- Generative models are considered a class of statistical models that can generate new data instances.
- A generative model could generate new photos of animals that look like real animals
- GANs(Generative Adversarial Network") is just one kind of generative model

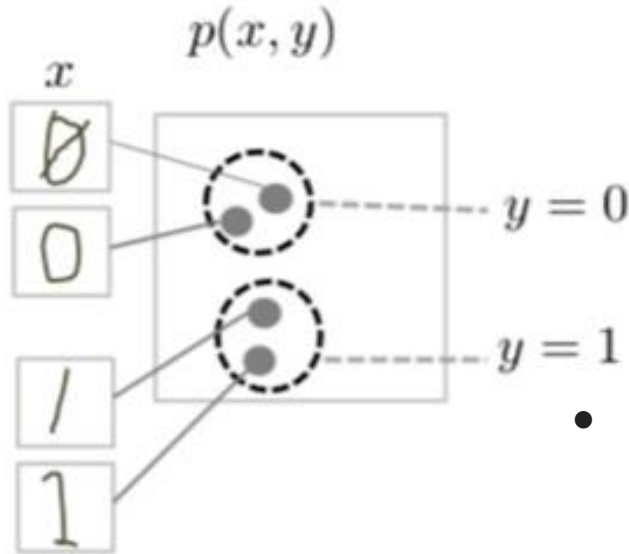
More formally, given a set of data instances X and a set of labels Y :

- Generative models capture the joint probability $p(X, Y)$, or just $p(X)$ if there are no labels.
- A generative model includes the distribution of the data itself, and tells you how likely a given example is.
- For example, models that predict the next word in a sequence are typically generative models (usually much simpler than GANs) because they can assign a probability to a sequence of words.



Generative Models

- Generative Model



- The generative model tries to produce convincing 1's and 0's by generating digits that fall close to their real counterparts in the data space.
- It has to model the distribution throughout the data space.
- These models use the concept of joint probability and create instances where a given feature (x) or input and the desired output or label (y) exist simultaneously.
- These models use probability estimates and likelihood to model data points and differentiate between different class labels present in a dataset.

Generative Models

Examples of Generative Models

- Naïve Bayes
- Bayesian networks
- Markov random fields
- Hidden Markov Models (HMMs)
- Latent Dirichlet Allocation (LDA)
- Generative Adversarial Networks (GANs)
- Autoregressive Model

Generative Adversarial Models(GAN's)

GAN Structure

A generative adversarial network (GAN) has two parts:

- The **generator** learns to generate plausible data. The generated instances become negative training examples for the discriminator.
- The **discriminator** learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake:

Generative Adversarial Models(GAN's)



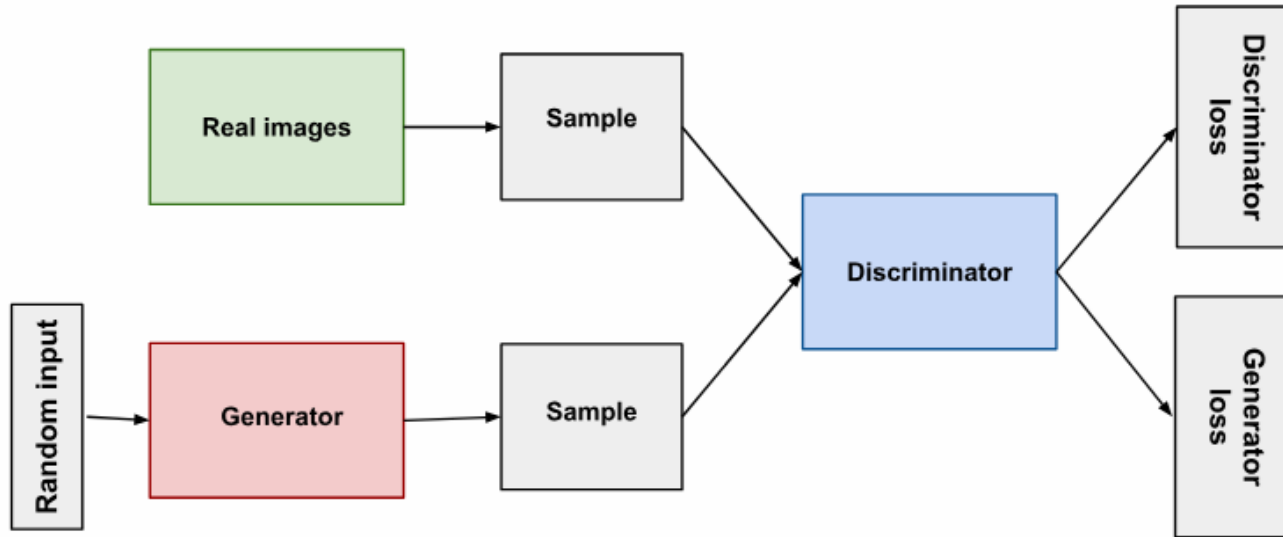
As training progresses, the generator gets closer to producing output that can fool the discriminator:



Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases.



Generative Adversarial Models(GAN's)

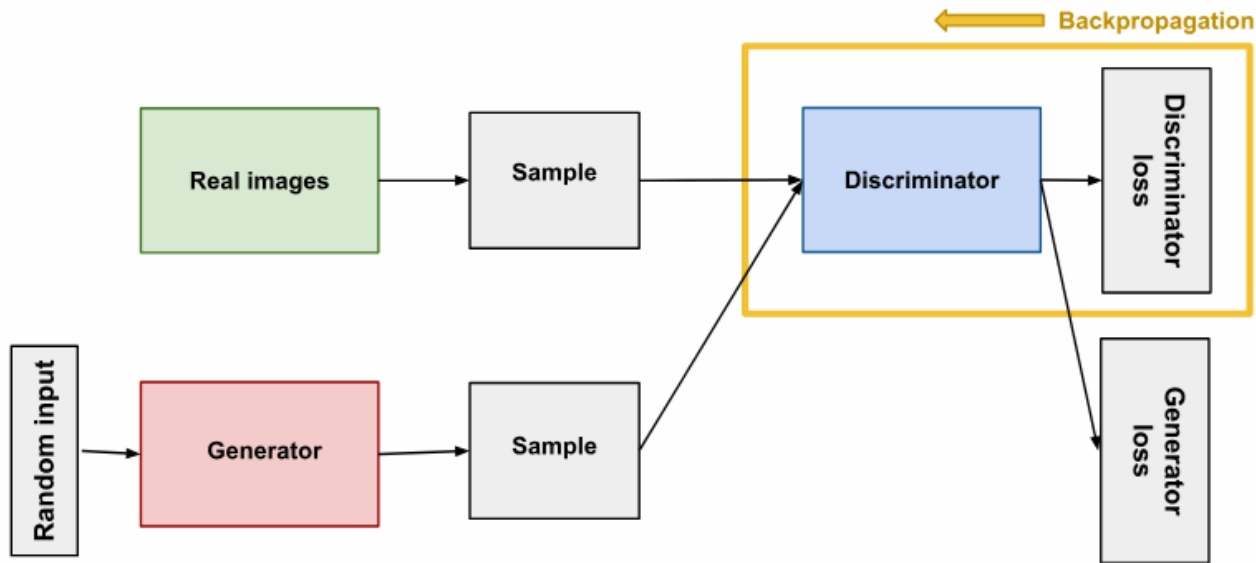


- Both the generator and the discriminator are neural networks.
- The generator output is connected directly to the discriminator input.
- Through backpropagation, the discriminator classification provides a signal that the generator uses to update its weights.

Generative Adversarial Models(GAN's)

Discriminator

- The discriminator in a GAN is simply a classifier.
- It tries to distinguish real data from the data created by the generator.
- It could use any network architecture appropriate to the type of data it's classifying.



Generative Adversarial Models(GAN's)

- The discriminator training data comes from two sources:
 - 1) **Real data** instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
 - 2) **Fake data** instances created by the generator. The discriminator uses these instances as negative examples during training.
- From figure, the two "Sample" boxes represent these two data sources feeding into the discriminator.
- During discriminator training the generator does not train.
- Its weights remain constant while it produces examples for the discriminator to train on.
- The discriminator connects to two loss functions.
- During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss.

Generative Adversarial Models(GAN's)

During discriminator training:

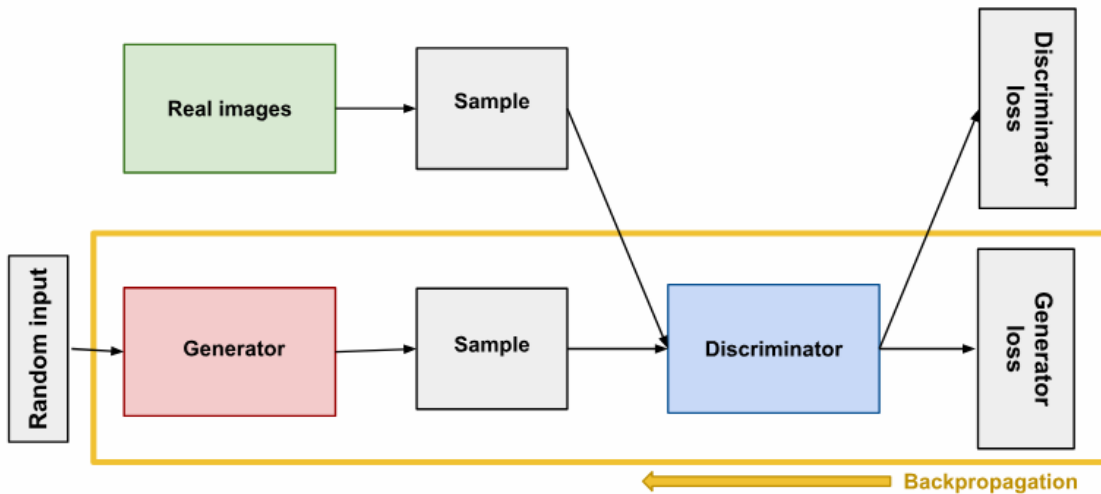
1. The discriminator classifies both real data and fake data from the generator.
2. The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.
3. The discriminator updates its weights through back propagation from the discriminator loss through the discriminator network.

Generative Adversarial Models(GAN's)

Generator

- The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator.
- It learns to make the discriminator classify its output as real.
- Generator training requires tighter integration between the generator and the discriminator than discriminator training requires.
- The portion of the GAN that trains the generator includes:
 - ❖ random input
 - ❖ generator network, which transforms the random input into a data instance
 - ❖ discriminator network, which classifies the generated data
 - ❖ discriminator output
 - ❖ generator loss, which penalizes the generator for failing to fool the discriminator

Generative Adversarial Models(GAN's)



- In its most basic form, a GAN takes random noise as its input.
- The generator then transforms this noise into a meaningful output.
- By introducing noise, we can get the GAN to produce a wide variety of data, sampling from different places in the target distribution.
- For convenience the space from which the noise is sampled is usually of smaller dimension than the dimensionality of the output space.

Generative Adversarial Models(GAN's)

So we train the generator with the following procedure:

1. Sample random noise.
2. Produce generator output from sampled random noise.
3. Get discriminator "Real" or "Fake" classification for generator output.
4. Calculate loss from discriminator classification.
5. Backpropagate through both the discriminator and generator to obtain gradients.
6. Use gradients to change only the generator weights.

Because a GAN contains two separately trained networks, its training algorithm must address two complications:

- GANs must juggle two different kinds of training (generator and discriminator).
- GAN convergence is hard to identify.

Generative Adversarial Models(GAN's)

- The generator and the discriminator have different training processes. So how do we train the GAN as a whole?
- GAN training proceeds in alternating periods:
 1. The discriminator trains for one or more epochs.
 2. The generator trains for one or more epochs.
 3. Repeat steps 1 and 2 to continue to train the generator and discriminator networks.

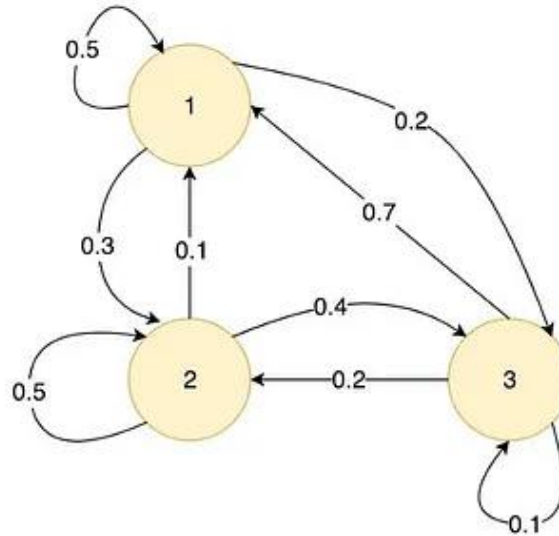
Two losses

1)Minimax

2)Wasserstein loss

Hidden Markov Model(HMM)

A Markov chain contains all the possible states of a system and the probability of transiting from one state to another.



Markov process

	S ₁	S ₂	S ₃
S ₁	0.5	0.1	0.7
S ₂	0.3	0.5	0.2
S ₃	0.2	0.4	0.1

Transition matrix

Hidden Markov Model(HMM)

- A first-order Markov chain assumes that the next state depends on the current state only.
- For simplicity, we often call it a Markov chain.

$$P(X_{n+1} = x \mid \underbrace{X_1 = x_1, X_2 = x_2, \dots, X_n = x_n}_{\text{can be ignored}}) = P(X_{n+1} = x \mid X_n = x_n)$$

An HMM is a statistical model that deals with sequences. It consists of two main components:

- **Hidden States (S):** These are the underlying, unobservable states that the model transitions between. In many applications, these states represent some underlying phenomenon or process.
- **Observations (O):** These are the observable outcomes that are associated with each hidden state. Observations are generated based on the current hidden state.

Hidden Markov Model(HMM)

- This model will be much easier to handle. However, in many ML systems, not all states are observable and we call these states hidden states or internal states.
- For example, it may not be easy to know whether I am happy or sad. My internal state will be {H or S}. But we can get some hints from what we observe.
- For example, when I am happy I have a 0.2 chance that I watch a movie, but when I am sad, that chance goes up to 0.4.
- The probability of observing an observable given an internal state is called the **emission probability**.
- The probability of transiting from one internal state to another is called the **transition probability**.

Hidden Markov Model(HMM)

State Transition Probability (A):

- This represents the probability of transitioning from one hidden state to another.
- It's usually represented as a matrix where element $A[i][j]$ is the probability of transitioning from state i to state j .
- Example: If we have two weather states, "Sunny" (S1) and "Rainy" (S2), our transition matrix might look like:

$$A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$

- This means there's a 70% chance of staying sunny and a 30% chance of transitioning from sunny to rainy.

Hidden Markov Model(HMM)

Observation Probability (B):

- This represents the probability of observing a particular outcome given the hidden state.
- It's often represented as a matrix where element $B[i][j]$ is the probability of observing outcome j in hidden state i .
- Example: Let's say our weather states correspond to "Sunny" (S1) and "Rainy" (S2) again, and our observations are "No umbrella" (O1) and "Umbrella" (O2):

$$\mathbf{B} = \begin{bmatrix} 0.9 & 0.1 \end{bmatrix}$$

$$\begin{bmatrix} 0.2 & 0.8 \end{bmatrix}$$

- This means there's a 90% chance of not needing an umbrella when it's sunny and an 80% chance of needing an umbrella when it's rainy.

Hidden Markov Model(HMM)

Initial State Probability (π):

- This represents the probability distribution of starting in each hidden state.
- It's usually represented as a vector where element $\pi[i]$ is the probability of starting in state i .
- Example: Initial probabilities for our weather example:

$$\pi = [0.6 \quad$$

$$0.4]$$

- This means there's a 60% chance of starting in the sunny state and a 40% chance of starting in the rainy state.

Hidden Markov Model(HMM)

Given the above components, you can use the following steps to work with an HMM:

- 1) Initialize the HMM with the state transition matrix A , observation matrix B , and initial state probabilities π .
- 2) Given a sequence of observations, you can use algorithms like the Forward-Backward algorithm or the Viterbi algorithm to compute the probabilities of the observations given the model, and to find the most likely sequence of hidden states.
- 3) You can also train an HMM using methods like the Baum-Welch algorithm (a type of Expectation-Maximization algorithm) to adjust the model parameters based on observed sequences.

Auto Regressive Model

- An autoregressive (AR) model forecasts future behavior based on past behavior data.
- This type of analysis is used when there is a correlation between the time series values and their preceding and succeeding values.
- Autoregressive modeling uses only past data to predict future behavior.
- Linear regression is carried out on the data from the current series based on one or more past values of the same series.
- AR models are linear regression models where the outcome variable (Y) at some point of time is directly related to the predictor variable (X).
- In AR models, Y depends on X and previous values for Y, which is different from simple linear regression.

Auto Regressive Model

- A simple autoregressive model, also known as an AR(1), would look like this, for example, if X is a time-series variable:
- $X_t = C + \phi_1 X_{t-1} + \epsilon_t$
- In the first place, X_{t-1} represents the previous period's value of X .
- “ t ” represents today, while “ $t-1$ ” represents last week. As a result, X_{t-1} reflects last week's value.
- The coefficient ϕ_1 represents the numeric constant multiplied by the lagged variable (X_{t-1}). In other words, it represents the future portion of the previous value.
- Try to maintain these coefficients between -1 and 1. The reason for this is that when the absolute value of the coefficient exceeds 1, it will explode exponentially over time.
- ϵ_t -value is known as residual, representing the difference between our period t prediction and the correct value ($\epsilon_t = y_t - \hat{y}_t$).

Auto Regressive Model

- It is a common belief that past values influence current values in autoregressive models. Hence, this is why the statistical technique is widely getting used to analyze natural phenomena, economic processes, and other processes that change over time.
- Many regression models use linear combinations of predictors to forecast a variable.
- In contrast, autoregressive models use the variable's past values to determine the future value.
- AR(1) autoregressive processes depend on the value immediately preceding the current value. Alternatively, AR(2) uses the previous two values to calculate the current value while AR(0) processes white noise, which does not depend on terms.
- The least squares method gets used to calculate coefficients with these variations.

Auto Regressive Model

Advantages

- Advantage of this model is that you can tell if there is a lack of randomness by using the autocorrelation function.
- Additionally, it is capable of forecasting recurring patterns in data.
- It is also possible to predict outcomes with less information using self-variable series.

Limitations

- The autocorrelation coefficient must be at least 0.5 in this case for it to be appropriate.
- It is usually used while predicting things associated with economics based on historical data. Something that is significantly affected by social factors.
- It is highly recommended to use the vector autoregressive model instead. The reason being a single model can be used to predict multiple time series variables at the same time.

Evaluating of Generative Models

1) Inception Score (IS)

- The Inception Score (IS) is a popular evaluation metric for generative models, particularly for image generation tasks.
- It is based on the idea that a good generative model should produce diverse and realistic samples.
- The IS is calculated by using a pre-trained classifier (typically the Inception network) to classify the generated samples and compute the entropy of the predicted class probabilities.
- A high IS indicates that the generated samples are both diverse (high entropy) and realistic (low conditional entropy).
- A large number of generated images are classified using the model.
- Specifically, the probability of the image belonging to each class is predicted.
- These predictions are then summarized into the inception score.

Evaluating of Generative Models

- The entropy is calculated as the negative sum of each observed probability multiplied by the log of the probability.
- The intuition here is that large probabilities have less information than small probabilities.
- Entropy = $-\sum(p_i * \log(p_i))$
- The conditional probability captures our interest in image quality.
- To capture our interest in a variety of images, we use the marginal probability.
- This is the probability distribution of all generated images. We, therefore, would prefer the integral of the marginal probability distribution to have a high entropy.
- These elements are combined by calculating the **Kullback-Leibler divergence**, or KL divergence (relative entropy), between the conditional and marginal probability distributions $KL(C \parallel M)$
- The KL divergence is then calculated for each image as the conditional probability multiplied by the log of the conditional probability minus the log of the marginal probability.
- KL divergence = $p(y|x) * (\log(p(y|x)) - \log(p(y)))$

Evaluating of Generative Models

2) Frechet Inception Distance (FID)

- The Frechet Inception Distance (FID) is another evaluation metric for generative models that addresses some of the limitations of the IS.
- The FID measures the similarity between the distributions of the generated samples and the real data in the feature space of a pre-trained classifier (again, typically the Inception network).
- The FID is calculated by computing the Frechet distance between the two distributions, which takes into account both the mean and covariance of the feature vectors.
- A lower FID indicates that the generated samples are more similar to the real data.

The FID score is then calculated using the following equation

$$d^2 = \|\mu_1 - \mu_2\|^2 + \text{Tr}(\mathbf{C}_1 + \mathbf{C}_2 - 2\sqrt{\mathbf{C}_1\mathbf{C}_2})$$

Evaluating of Generative Models

- The FID score is then calculated using the following equation
- $d^2 = \|\mu_1 - \mu_2\|^2 + \text{Tr}(C_1 + C_2 - 2\sqrt{C_1 C_2})$
- d^2 :it is a distance and has squared units.
- “ μ_1 ” and “ μ_2 ” refer to the feature-wise mean of the real and generated images
- C_1 and C_2 are the **covariance matrix** for the real and generated feature vectors
- The $\|\mu_1 - \mu_2\|^2$ refers to the sum squared difference between the two mean vectors.
- Tr refers to the **trace linear algebra operation**, e.g. the sum of the elements along the main diagonal of the square matrix .
- The sqrt is the **square root of the square matrix**, given as the product between the two covariance matrices.

Evaluating of Generative Models

3) Perceptual Path Length (PPL)

- Perceptual Path Length (PPL) is an evaluation metric for generative models that focuses on the smoothness of the latent space(The latent space is simply a representation of compressed data in which similar data points are closer together in space).
- The idea is that a good generative model should have a smooth and continuous latent space, where small changes in the input result in small changes in the output.
- The PPL is calculated by measuring the average perceptual distance between pairs of generated samples that are interpolated in the latent space.
- A lower PPL indicates a smoother and more continuous latent space.
- Drastic changes mean that multiple features have changed together and that they might be entangled.

Evaluating of Generative Models

4) Precision, Recall, and F1 Score

- Precision, recall, and F1 score are evaluation metrics borrowed from the field of information retrieval and adapted for generative models.
- These metrics are based on the idea that a good generative model should generate samples that cover the diversity of the real data (high recall) while avoiding generating unrealistic samples (high precision).
- The F1 score is the harmonic mean of precision and recall, providing a single metric that balances the trade-off between the two.
- These metrics can be computed using nearest-neighbor matching in the feature space of a pre-trained classifier.

Evaluating of Generative Models

5) Log Likelihood

- Log-likelihood is a fundamental evaluation metric for generative models that measures the probability of the real data given the model.
- A higher log-likelihood indicates that the model assigns a higher probability to the real data, suggesting a better fit.
- However, log-likelihood can be difficult to compute for some generative models, such as **Generative Adversarial Networks** (GANs), due to the lack of an explicit likelihood function.

Likelihood $L(\theta; \xi)$



Some observed data



a set of probability distributions that could have generated the data; each distribution is identified by a parameter.



Evaluating of Generative Models

5) Log Likelihood

The log-likelihood function is typically used to derive the maximum likelihood estimator of the parameter θ .

The estimator $\hat{\theta}$ is obtained by solving

$$\hat{\theta} = \arg \max_{\theta \in \Theta} l(\theta; \xi)$$

that is, by finding the parameter $\hat{\theta}$ that maximizes the log-likelihood of the observed sample ξ .

This is the same as maximizing the likelihood function $L(\theta; \xi)$ because the natural logarithm is a strictly increasing function.

The logarithm transforms a product of densities into a sum

Challenges of Generative Models

1) Data Quality and Quantity

- Generative models require high-quality data to learn from and generate realistic and diverse content.
- However, data collection and preparation can be costly, time-consuming, and prone to errors and biases.
- Moreover, data privacy and security issues may limit the availability and accessibility of data for generative models.
- Therefore, generative models must ensure that they use reliable and representative data sources, apply proper data cleaning and augmentation techniques, and comply with ethical and legal standards for data protection.

Challenges of Generative Models

2) Model Performance and Evaluation

- Generative models are often complex and computationally intensive to train and serve.
- They may require specialized hardware and software infrastructure, such as GPUs, TPUs, cloud computing, and distributed systems.
- Moreover, generative models are difficult to evaluate and compare, as there is no clear and objective metric to measure their quality and diversity.
- Therefore, generative models need to optimize their resource utilization, scalability, and efficiency and develop appropriate evaluation methods and benchmarks that reflect the desired outcomes and user feedback.

Challenges of Generative Models

3) Model Explainability and Trustworthiness

- Generative models are often opaque and unpredictable in their behavior and outputs.
- They may generate inaccurate, inappropriate, or harmful content to users or society.
- They may also infringe on the original content creators' or consumers' intellectual property rights or moral values.
- Therefore, generative models need to ensure that they are transparent and accountable for their actions and decisions and respect the rights and preferences of the stakeholders involved.
- They also need to provide explanations and justifications for their outputs and mechanisms for correction and control.

Challenges of Generative Models

4) Balancing between quality and diversity:

- One of the main challenges of generative models is balancing between the quality of the generated data and its diversity.
- Some models may prioritize the quality of the data at the cost of diversity and vice versa.
- Finding a balance between these two factors is crucial for the success of the model.

Applications of Generative Models

1) Data Augmentation

- In cases when it is difficult or expensive to annotate a large amount of training data, we can use GANs to generate synthetic data and increase the size of the dataset.
- For example, **StyleGAN** is a generative model proposed by **Nvidia** that is able to generate very realistic images of human faces that don't exist.

1) Super Resolution

- we take as input a low-resolution image (like) and we want to increase its resolution (to and even more) and keep its quality as high as possible.
- **SRGAN** is a generative model that can successfully recover photo-realistic high-resolution images from low-resolution images.
- The model comprises a deep network in combination with an adversary network like in most GAN architectures.

Applications of Generative Models

3) Impainting

- In image inpainting, our task is to reconstruct the missing regions in an image.
- In particular, we want to fill the missing pixels of the input image in a way that the new image is still realistic and the context is consistent.
- The applications of this task are numerous like image rendering, editing, or unwanted object removal.
- [Deepfill](#) is open-source framework for the image inpainting task that uses a generative model-based approach.
- Its novelty lies in the Contextual Attention layer which allows the generator to make use of the information given by distant spatial locations for the reconstruction of local missing pixels.

Applications of Generative Models

4) Denoising

- Nowadays, thanks to modern digital cameras we are able to take high-quality photos.
- However, there are still cases where an image contains a lot of noise and its quality is low.
- Removing the noise from an image without losing image features is a very crucial task and researchers have been working on denoising methods for many years.
- A very popular generative model for image denoising is [Autoencoder](#) that is trained to reconstruct its input image after removing the noise.
- During training, the network is given the original image and its noisy version.
- Then, the network tries to reconstruct its output to be as close as possible to the original image.

5) Image Colorization

- CycleGAN for automatic image colorization which is very useful in areas like restoration of aged or degraded images. CycleGAN converts a grayscale image of a flower to its colorful RGB form

Applications of Generative Models

6) Translation

- Generative models are also used in image translation where our goal is to learn the mapping between two image domains.
- Then, the model is able to generate a synthetic version of the input image with a specific modification like translating a winter landscape to summer.
- **CycleGAN** is a very famous GAN-based model for image translation.
- The model is trained in an unsupervised way using a dataset of images from the source and the target domain.

7) Object Transfiguration

- Another exciting application of StyleGAN is object transfiguration where the model translates one object class to another like translating a horse to a zebra, a winter landscape to a summer one, and apples to oranges.

Deep Generative Models

- A subset of generative modeling, deep generative modeling uses deep neural networks to learn the underlying distribution of data.
- These models can develop novel samples that have never been seen before by producing new samples that are similar to the input data but not exactly the same.
- Deep generative models are multi-layer nonlinear neural networks that are used to simulate data dependency.
- Deep generative models have sparked a considerable interest because of their ability to provide a very efficient approach to evaluate and understand unlabeled data.
- They do not suffer from the capacity limitation and can learn to generate high-level representations solely from data.
- More crucially, when back-propagation is allowed, deep generative model training becomes extremely efficient, resulting in much superior performance than shallow models.

Deep Generative Models

- A subset of generative modeling, deep generative modeling uses deep neural networks to learn the underlying distribution of data.
- These models can develop novel samples that have never been seen before by producing new samples that are similar to the input data but not exactly the same.
- Deep generative models are multi-layer nonlinear neural networks that are used to simulate data dependency.
- Deep generative models have sparked a considerable interest because of their ability to provide a very efficient approach to evaluate and understand unlabeled data.
- They do not suffer from the capacity limitation and can learn to generate high-level representations solely from data.
- More crucially, when back-propagation is allowed, deep generative model training becomes extremely efficient, resulting in much superior performance than shallow models.

Deep Learning/DNN

- Deep learning is a branch of machine learning which is based on artificial neural networks.
- It is capable of learning complex patterns and relationships within data.
- In deep learning, we don't need to explicitly program everything.
- It is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs)
- These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.
- In a fully connected Deep neural network, there is an input layer and one or more hidden layers connected one after the other.
- Each neuron receives input from the previous layer neurons or the input layer.
- The layers of the neural network transform the input data through a series of nonlinear transformations, allowing the network to learn complex representations of the input data.

Advanced DNN

- Advanced Deep Neural Networks (DNNs) refer to sophisticated and complex architectures that have been developed to tackle more challenging tasks and learn intricate patterns from data.
- These architectures often build upon the foundation of basic neural networks (feedforward networks) by incorporating advanced techniques and structures to enhance their performance.

1)CNN(Convolutional Neural Network)

2)RNN(Recurrent Neural Network)

3)LSTM(Long Short Term Memory)

4)GRU(Gated Recurrent Units)

5)VAE(Autoencoders and Variational Autoencoders)



CNN

- A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision.
- Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.
- Neural Networks are used in various datasets like images, audio, and text.
- Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use [Recurrent Neural Networks](#) more precisely an [LSTM](#), similarly for image classification we use Convolution Neural networks.
- In a regular Neural Network there are three types of layers: Input Layer, Hidden Layer and Output Layer

1) Input Layer:

- It's the layer in which we give input to our model.
- The number of neurons in this layer is equal to the total number of features in our data

2) Hidden Layer:

- The input from the Input layer is then feed into the hidden layer.
- There can be many hidden layers depending upon our model and data size.
- Each hidden layer can have different numbers of neurons which are generally greater than the number of features.
- The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear

3)Output Layer:

- The output from the hidden layer is then fed into a activation function which converts the output of each class into the probability score of each class.
- Activation Functions :The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks. Linear, Sigmoid,Tanh,RELU & Softmax

Next.....

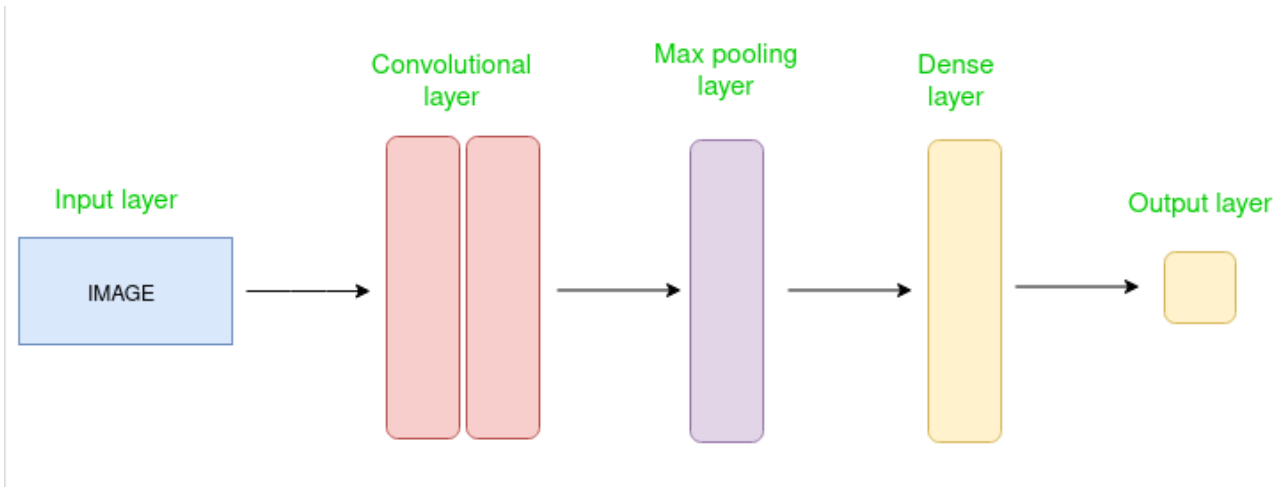
- The data is fed into the model and output from each layer is obtained from the above step is called **feedforward**, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc.
- The error function measures how well the network is performing.
- After that, we backpropagate into the model by calculating the derivatives.
- This step is called **Backpropagation** which basically is used to minimize the loss.

CNN

- Convolutional Neural Network (CNN) is the extended version of **artificial neural networks (ANN)** which is predominantly used to extract the feature from the grid-like matrix dataset.

CNN architecture

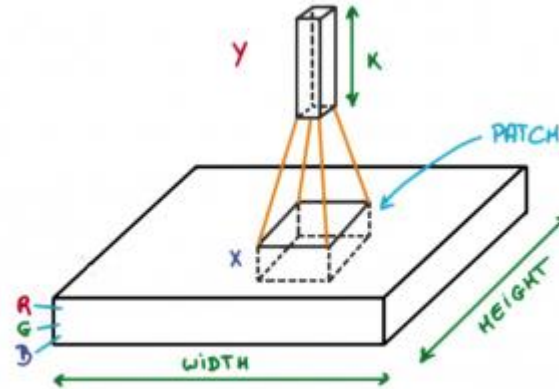
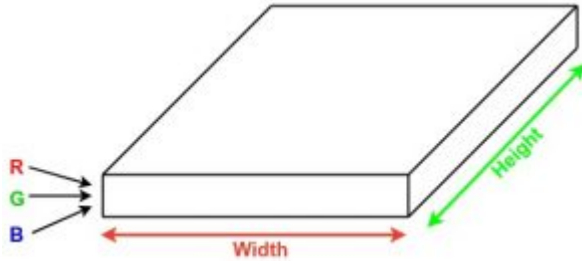
- Convolutional Neural Network consists of multiple layers like the Input layer, Convolutional layer, Pooling layer, and Dense, fully connected layers.





CNN

- The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction.
- The network learns the optimal filters through backpropagation and gradient descent.



CNN

- Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).
- Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically.
- Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height.
- This operation is called **Convolution**.
- If the patch size is the same as that of the image it will be a regular neural network.

CNN

Now let's talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimensions $34 \times 34 \times 3$. The possible size of filters can be $a \times a \times 3$, where 'a' can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
- During the forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.
- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

CNN

Layers used to build ConvNets(CNN Architecture)

A convnets is a sequence of layers, every layer transforms one volume to another through a differentiable function.

Let's take an example by running a convnets on of image of dimension $32 \times 32 \times 3$.

- 1) **Input Layers:** In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.
- 2) **Convolutional Layers:** This is the layer, which is used to extract the feature from the input dataset.
 - It applies a set of learnable filters known as the kernels to the input images.
 - The filters/kernels are smaller matrices usually 2×2 , 3×3 , or 5×5 shape.
 - It slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch.
 - The output of this layer is referred ad feature maps. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension $32 \times 32 \times 12$.

3) Activation Layer:

- By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network.
- It will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: $\max(0, x)$, Tanh, Leaky RELU, etc.
- The volume remains unchanged hence output volume will have dimensions $32 \times 32 \times 12$.

4) Pooling layer:

- This layer is periodically inserted in the convnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting.
- Two common types of pooling layers are max pooling and average pooling.
- If we use a max pool with 2×2 filters and stride 2, the resultant volume will be of dimension $16 \times 16 \times 12$.

CNN

5) Flattening:

- The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

6) Fully Connected Layers:

- It takes the input from the previous layer and computes the final classification or regression task.

7)Output Layer:

- The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

Advantages of Convolutional Neural Networks (CNNs):

1. Good at detecting patterns and features in images, videos, and audio signals.
2. Robust to translation, rotation, and scaling invariance.
3. End-to-end training, no need for manual feature extraction.
4. Can handle large amounts of data and achieve high accuracy.

Disadvantages of Convolutional Neural Networks (CNNs):

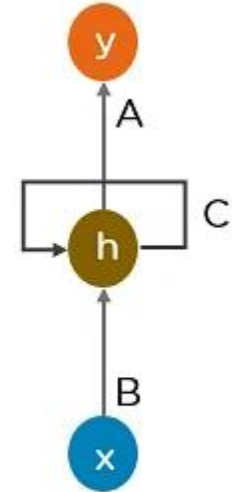
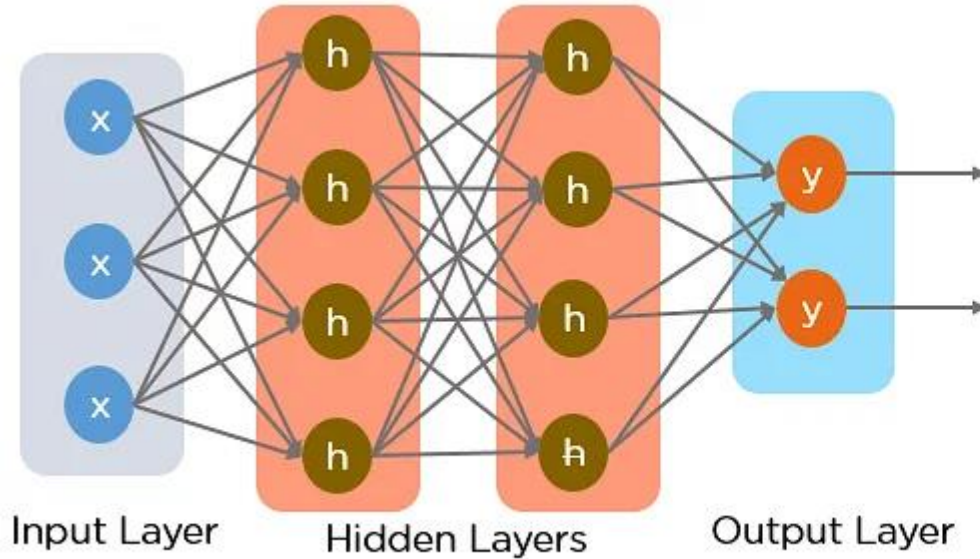
1. Computationally expensive to train and require a lot of memory.
2. Can be prone to overfitting if not enough data or proper regularization is used.
3. Requires large amounts of labeled data.
4. Interpretability is limited, it's hard to understand what the network has learned.

RNN

- **Recurrent Neural Network** also known as (**RNN**) that works better than a simple neural network when data is sequential like Time-Series data and text data.
- It is a type of **Neural Network** where the output from the previous step is fed as input to the current step.
- In traditional neural networks, all the inputs and outputs are independent of each other, but in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words.
- Thus RNN came into existence, which solved this issue with the help of a Hidden Layer.
- The main and most important feature of RNN is its **Hidden state**, which remembers some information about a sequence.
- The state is also referred to as *Memory State* since it remembers the previous input to the network.



RNN

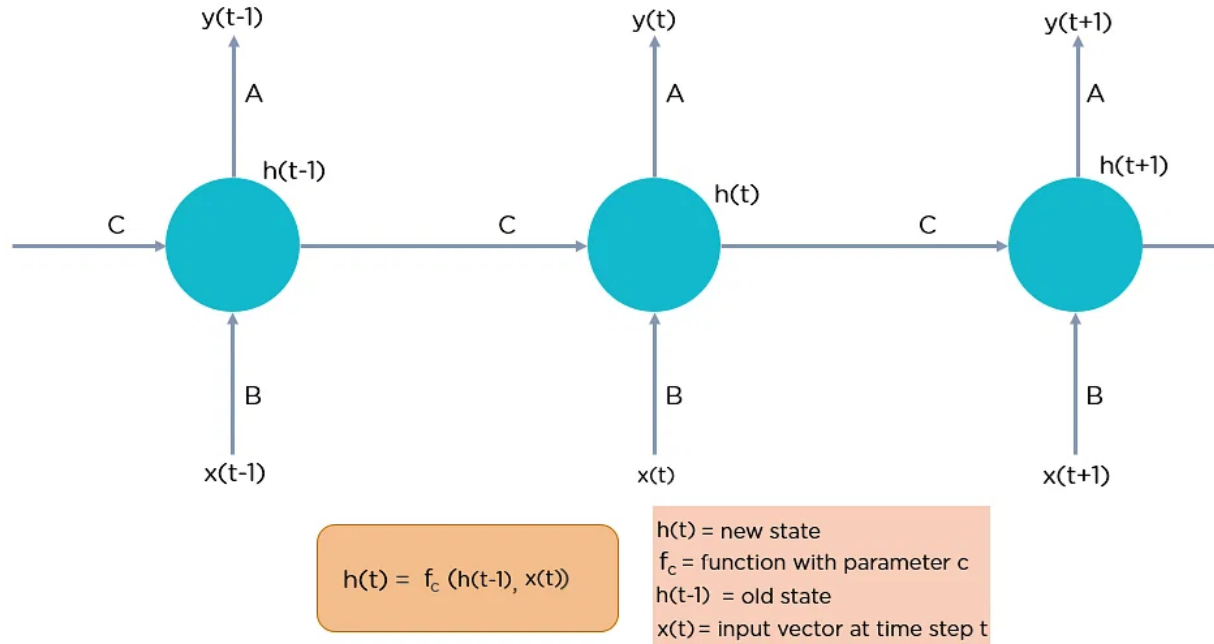


Recurrent Neural Network

- The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.



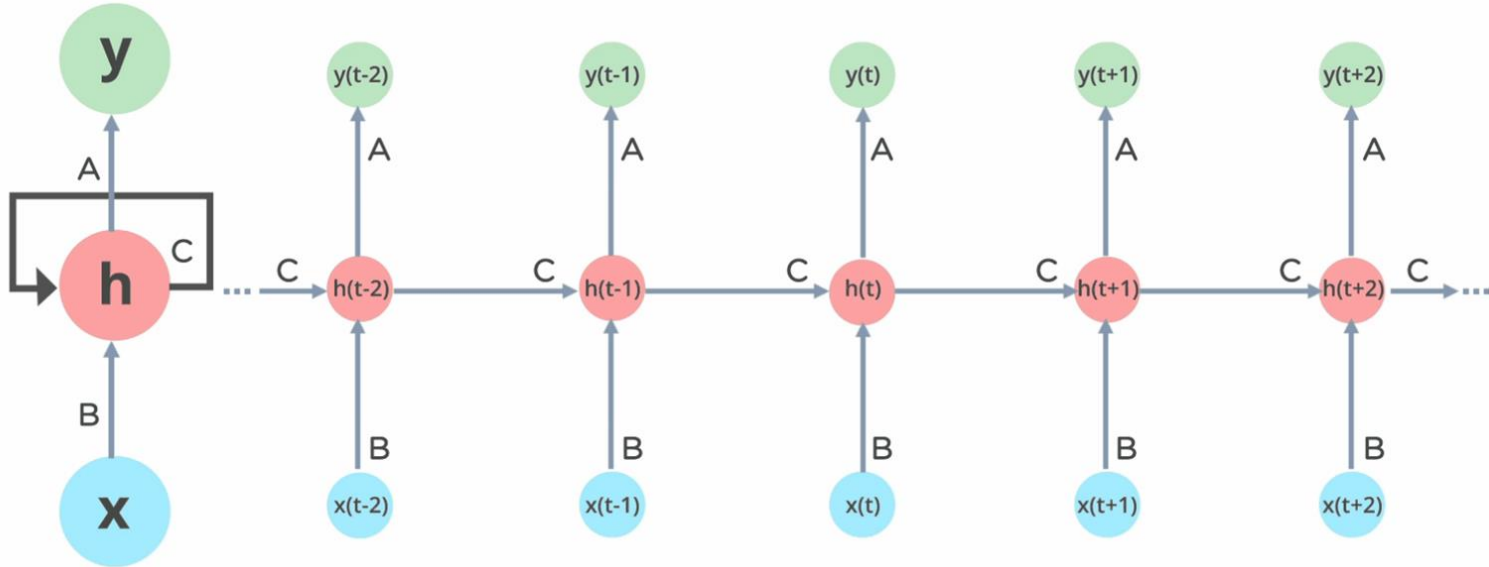
RNN



- The output at any given time is fetched back to the network to improve on the output.



RNN



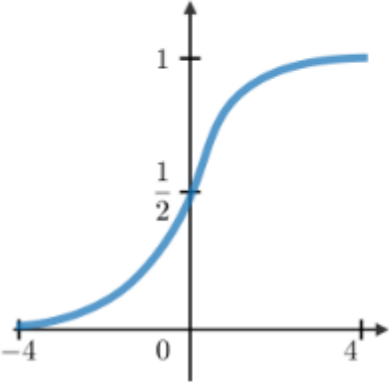
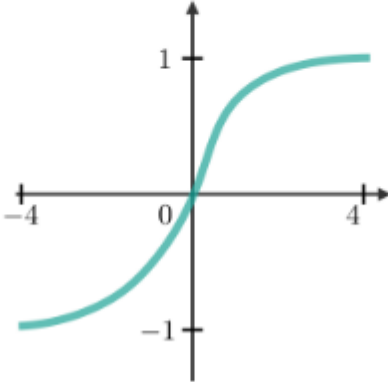
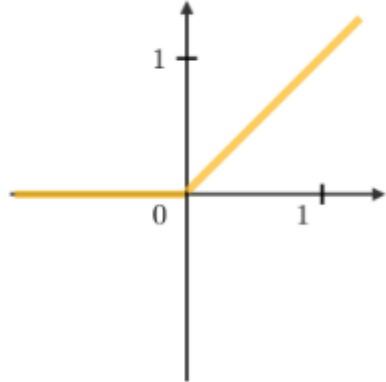
RNN

- The input layer 'x' takes in the input to the neural network and processes it and passes it onto the middle layer.
- The middle layer 'h' can consist of multiple hidden layers, each with its own activation functions and weights and biases.
- If you have a neural network where the various parameters of different hidden layers are not affected by the previous layer, ie: the neural network does not have memory, then you can use a recurrent neural network.
- The Recurrent Neural Network will standardize the different activation functions and weights and biases so that each hidden layer has the same parameters.
- Then, instead of creating multiple hidden layers, it will create one and loop over it as many times as required.



RNN

- The common types of activation functions used in RNN modules

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

RNN

Vanishing/exploding gradient

- The vanishing and exploding gradient phenomena are often encountered in the context of RNNs.
- The reason why they happen is that it is difficult to capture long term dependencies because of multiplicative gradient that can be exponentially decreasing/increasing with respect to the number of layers.

Gated Recurrent Unit (GRU) and Long Short-Term Memory units (LSTM) deal with the vanishing gradient problem encountered by traditional RNNs, with LSTM being a generalization of GRU.

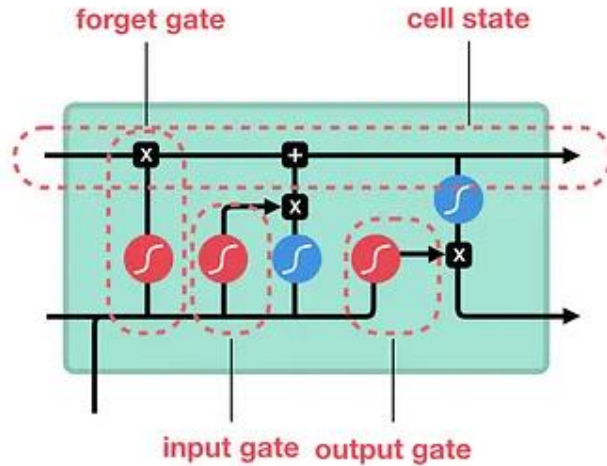
LSTM & GRU

- Recurrent Neural Networks suffer from short-term memory.
- If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones.
- So if you are trying to process a paragraph of text to do predictions, RNN's may leave out important information from the beginning.
- During back propagation, recurrent neural networks suffer from the vanishing gradient problem.
- Gradients are values used to update a neural networks weights.
- The vanishing gradient problem is when the gradient shrinks as it back propagates through time.
- If a gradient value becomes extremely small, it doesn't contribute too much learning.
- LSTM 's and GRU's were created as the solution to short-term memory.
- They have internal mechanisms called gates that can regulate the flow of information.

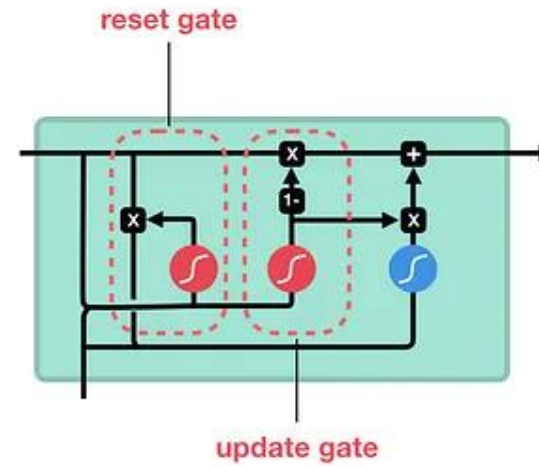


LSTM & GRU

LSTM



GRU



sigmoid



tanh



pointwise
multiplication



pointwise
addition



vector
concatenation

LSTM & GRU

- LSTM or GRU learn to keep only relevant information to make predictions, and forget non relevant data.

Review of RNN

- 1) First words get transformed into machine-readable vectors.
- 2) Then the RNN processes the sequence of vectors one by one.
- 3) While processing, it passes the previous hidden state to the next step of the sequence.
- 4) The hidden state acts as the neural networks memory.
- 5) It holds information on previous data the network has seen before.
- 6) First, the input and previous hidden state are combined to form a vector.
- 7) That vector now has information on the current input and previous inputs.
- 8) The vector goes through the tanh activation, and the output is the new hidden state, or the memory of the network.

LSTM & GRU

- The tanh activation is used to help regulate the values flowing through the network.
- The tanh function squishes values to always be between -1 and 1.
- A tanh function ensures that the values stay between -1 and 1, thus regulating the output of the neural network.
- RNN's uses a lot less computational resources than it's evolved variants, LSTM's and GRU's.

LSTM & GRU

1) LSTM

- An LSTM has a similar control flow as a recurrent neural network.
- It processes data passing on information as it propagates forward.
- The differences are the operations within the LSTM's cells.
- The cell state act as a transport highway that transfers relative information all the way down the sequence chain.
- You can think of it as the “memory” of the network. The cell state, can carry relevant information throughout the processing of the sequence.
- So even information from the earlier time steps can make it's way to later time steps, reducing the effects of short-term memory.

LSTM & GRU

- As the cell state goes on its journey, information get's added or removed to the cell state via gates.
- The gates are different neural networks that decide which information is allowed on the cell state.
- The gates can learn what information is relevant to keep or forget during training.
- Gates contains sigmoid activations.
- That is helpful to update or forget data because any number getting multiplied by 0 is 0, causing values to disappears or be “forgotten.”
- Any number multiplied by 1 is the same value therefore that value stay's the same or is “kept.”
- The network can learn which data is not important therefore can be forgotten or which data is important to keep.
- we have three different gates that regulate information flow in an LSTM cell.

Forget gate, Input gate, and Output gate.

LSTM & GRU

1)Forget Gate

- This gate decides what information should be thrown away or kept.
- Information from the previous hidden state and information from the current input is passed through the sigmoid function.
- Values come out between 0 and 1.
- The closer to 0 means to forget, and the closer to 1 means to keep.

LSTM & GRU

2) Input Gate

- To update the cell state, we have the input gate.
- First, we pass the previous hidden state and current input into a sigmoid function.
- That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important.
- You also pass the hidden state and current input into the tanh function to squish values between -1 and 1 to help regulate the network.
- Then you multiply the tanh output with the sigmoid output.
- The sigmoid output will decide which information is important to keep from the tanh output.

3) Cell State

- Now we should have enough information to calculate the cell state.
- First, the cell state gets pointwise multiplied by the forget vector.
- This has a possibility of dropping values in the cell state if it gets multiplied by values near 0.
- Then we take the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant.
- That gives us our new cell state.

4) Output Gate

- Last we have the output gate.
- The output gate decides what the next hidden state should be.
- Remember that the hidden state contains information on previous inputs.
- The hidden state is also used for predictions.
- First, we pass the previous hidden state and the current input into a sigmoid function.
- Then we pass the newly modified cell state to the tanh function.
- We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry.
- The output is the hidden state.
- The new cell state and the new hidden is then carried over to the next time step.

LSTM & GRU

- To review, the Forget gate decides what is relevant to keep from prior steps.
- The input gate decides what information is relevant to add from the current step.
- The output gate determines what the next hidden state should be.

LSTM & GRU

2) GRU

- The GRU is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM.
- GRU's got rid of the cell state and used the hidden state to transfer information.
- It also only has two gates, a reset gate and update gate.

a) Update Gate

- The update gate acts similar to the forget and input gate of an LSTM.
- It decides what information to throw away and what new information to add.

a) Reset Gate

- The reset gate is another gate is used to decide how much past information to forget.
- GRU's has fewer tensor operations; therefore, they are a little speedier to train than LSTM's.

Thank you!



D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE