Transformer Networks: How They Work and Why They Matter

Rakshit Agrawal

Principal AI Scientist

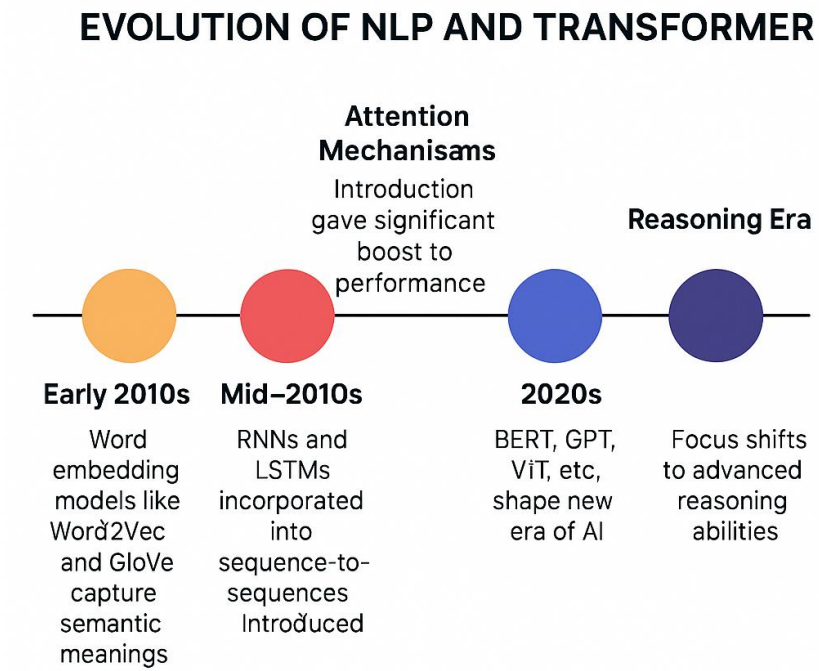
Synthpop AI

Synthpop

Introduction to AI Evolution

- In Natural Language Processing (NLP), in the early 2010s, word embedding models like Word2Vec and GloVe started capturing semantic meanings.
- In the mid 2010s, RNNs and LSTMs incorporated this into sequence-to-sequence models making it possible to generate continuous text sequences with deep learning.
- Introduction of attention mechanisms gave a significant boost to the performance of these models.
- In 2017, transformers proposed an architecture entirely based on attention, removing the limitations of recurrent neural networks
- Transformer based models such as BERT, GPT, ViT, etc., are shaping the new era of AI.

AI Evolution Visualized by Transformers

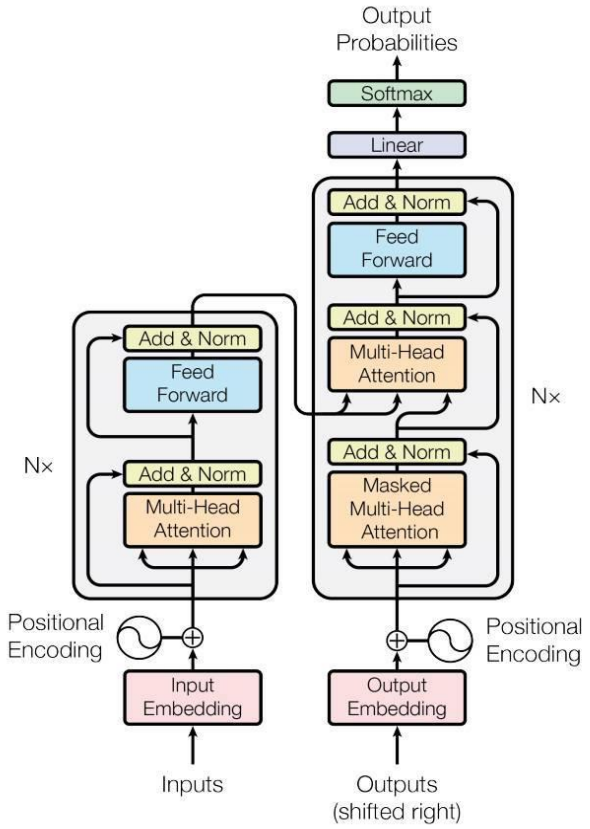


Importance of Transformers in Modern AI Research and Applications

- **Multi-Domain Integration -** Transformers enable unified learning across text, images, and audio for cross-modal reasoning.
- **Flexible and Scalable Architecture -** One design handles diverse data types, replacing the need for CNNs or RNNs.
- **Superior Performance in Vision -** Vision Transformers capture long-range patterns and reduce biases, outperforming CNNs.
- **Deep Contextual Understanding -** Self-attention considers all input at once for coherent understanding and generation.
- **Core of Modern AI Applications -** Transformers power LLMs and multi-modal systems through simple prompting.
- **Pathway to General AI -** Their ability to learn across domains supports better generalization and adaptability.

Understanding Transformers

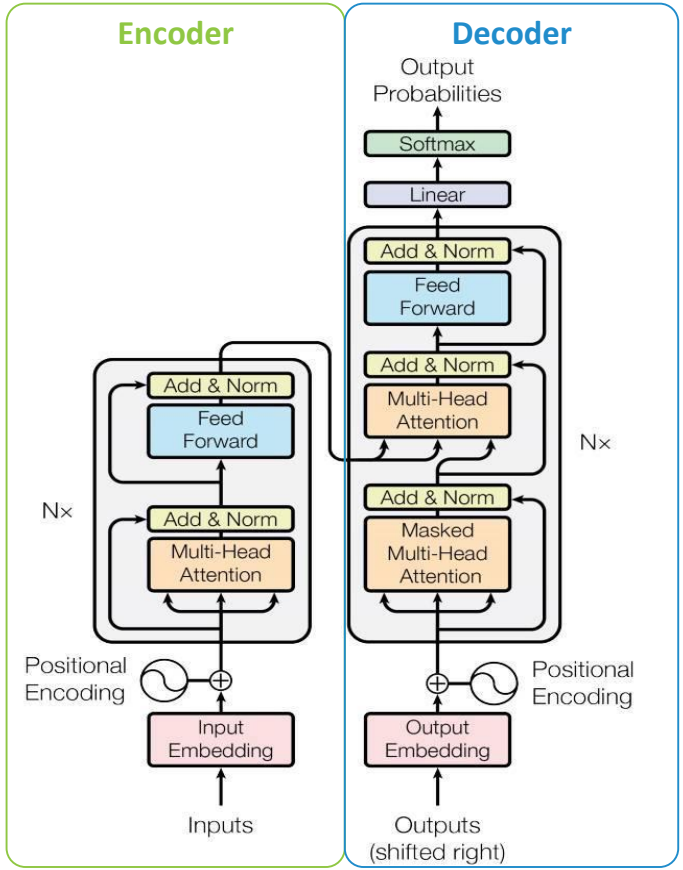
- Neural networks based entirely on attention, replacing recurrent layers.
- Processes entire sequences in parallel, boosting speed and efficiency.
- Highly scalable with increased computational power and data size.
- Versatile across multiple domains, including text, vision, and speech.



The Core of Transformers

Transformer Architecture

- Consists of encoder and decoder blocks
- Main components of a block:
  - Self-attention
  - Layer normalization
  - Feed-forward neural network
- Uses positional encodings



Encoders-Decoders

	Encoder	Decoder
Function	Input → Context	Context → Output
Process	Generates a representation encoding the entire input sequence.	Combines encoder output and previous decoder outputs to predict next token in sequence.
Components	• Self-attention • Layer normalization • Feed-forward neural network	• Self-attention • Layer normalization • Feed-forward neural network • Additional layer for outputs of encoder

Tokenization

What is Tokenization?

- Converts text into smaller units: words, subwords, or characters.
- Tokens are mapped to numerical IDs.

Why Tokenization Matters:

- Enables transformers to process consistent numerical inputs.

Types of Tokenizers:

- WordPiece (BERT), BPE (GPT), Unigram (SentencePiece)

Example:

“Unbelievable” → ["un", "believ", "able"] → [1034, 785, 9012]

Embeddings: Turning Tokens into Vectors

- **Next Step After Tokenization –** Once text is tokenized into IDs, each token is mapped to a high-dimensional vector called an embedding.
- **Why It Matters?** Embeddings give numerical IDs semantic meaning – capturing relationships, similarity, and context.
- **Token vs. Embedding**
  - **Token ID** = discrete integer (e.g. 1034)
  - **Embedding** = dense vector (e.g. [0.12, -0.07, ..., 0.45])
- **Transformer Input Starts Here –** These vectors are passed to the transformer (along with positional encodings) to begin learning from context.
- **Example**

  "un" → 1034 →[0.12, -0.07, ..., 0.45]

Positional Encodings

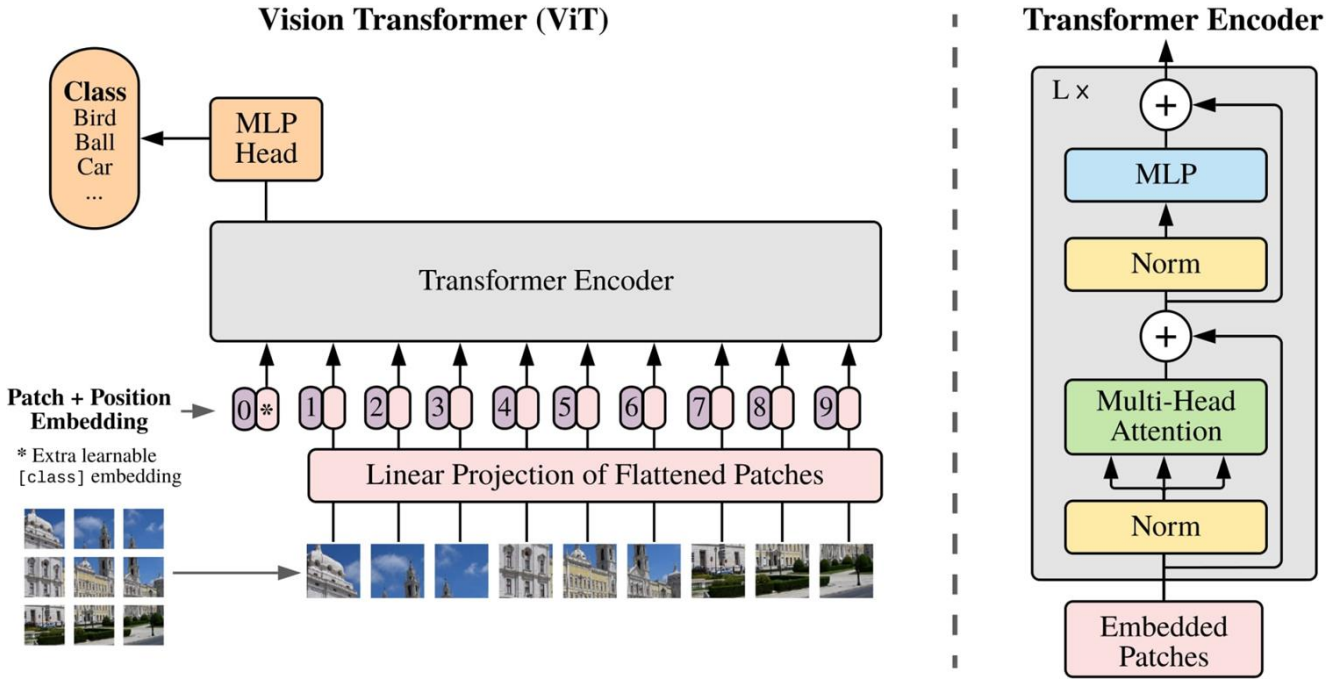
- **Role**: Enables sequence recognition by providing unique positional signals.
- **Types**: Mainly sinusoidal or trainable learned encodings.
- **Integration**: Added to input embeddings before self-attention layers.
- **Purpose**: Maintains position information throughout the transformer.
- **Impact**: Enhances handling of sequence-dependent tasks effectively.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

where *pos* is the position and *i* is the dimension from d_{model} dimensions of the model

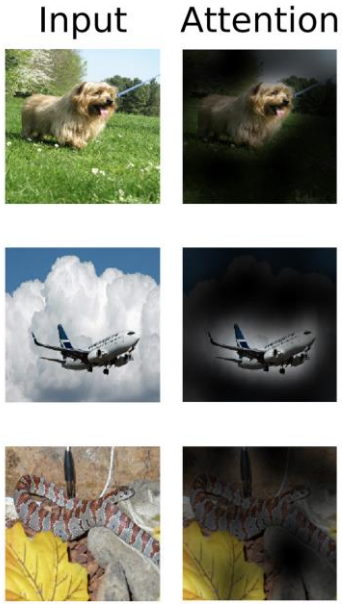
Positional Encodings in Vision Transformer (ViT)



Source: [An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale]

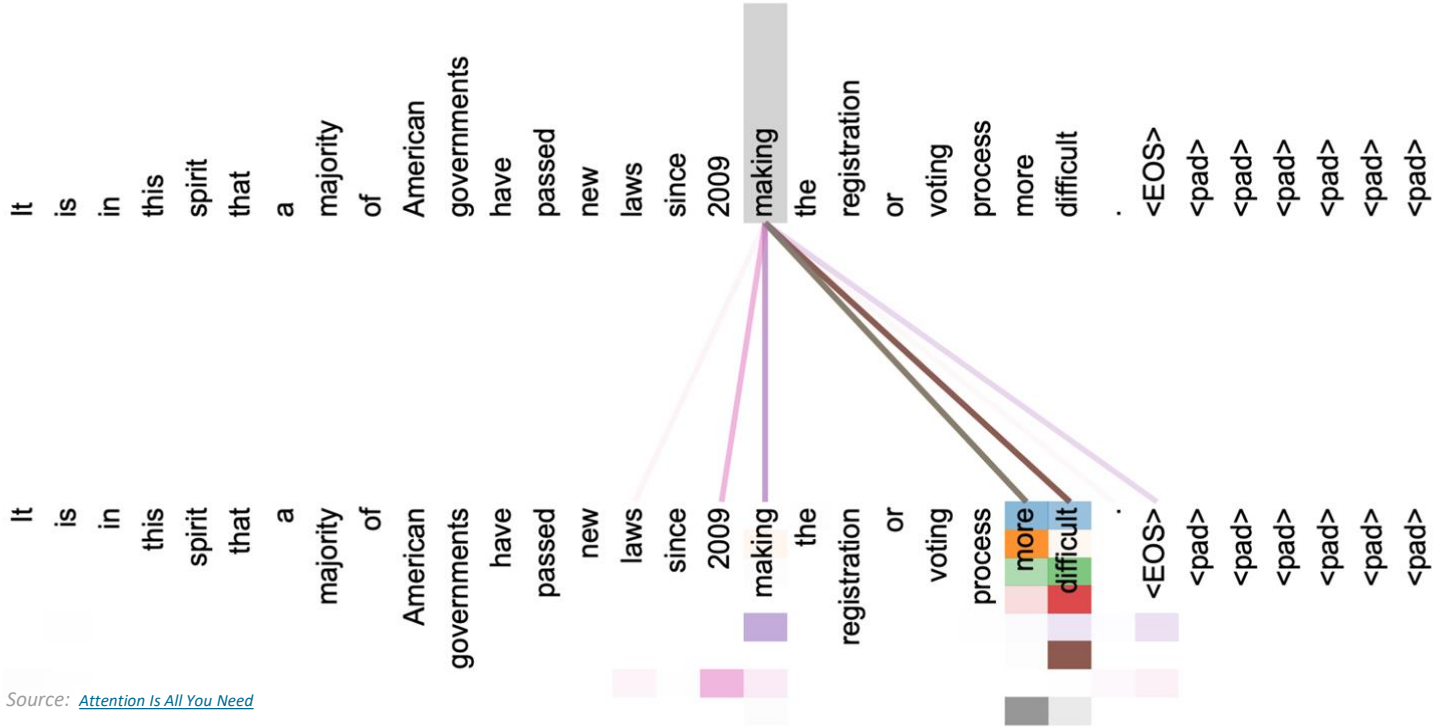
Self-Attention Mechanism

- A mechanism that allows each position in the decoder to attend to all positions in the encoder of the previous layer.
- Self-attention computes a weighted sum of all input representations with respect to their relevance.
- Benefits of self-attention
  - Allows the model to dynamically focus on different parts of the sequence.
  - Provides a more nuanced understanding and representation of the sequence.
  - Facilitates parallel processing, unlike RNNs which process data sequentially.



Source: *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*

Self-Attention Mechanism: Visualizing Attention



Source: Attention Is All You Need

Query, Key, Value

What Are Q, K, V?

- Each token is transformed into three vectors:
  - Query (Q) – what this token is looking for.
  - Key (K) – how relevant this token is to others.
  - Value (V) – the information this token carries.

How They Work Together:

- Each token's Query is matched with all Keys via a dot product.
- Scores are scaled and passed through softmax → attention weights.
- These weights are applied to the Values → weighted sum is the output for that token.

Why This Matters:

- Allows each word to attend to others dynamically, capturing context across positions.

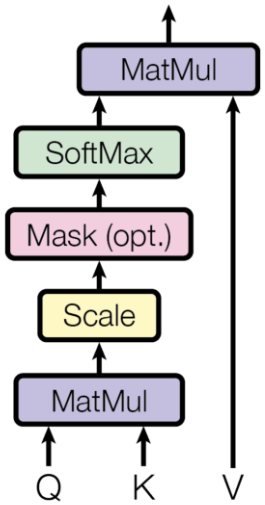
Query, Key, Value: Example

- *Input sentence*: 'The cat sat on the mat.'
- Let's say we're focusing on the word ‘sat’ (this becomes our **Query**).
- Each word in the sentence is transformed into a **Key** and a **Value**.
- 'sat' (Query) is compared with Keys of all words:
  - Similarity with 'cat' → High
  - Similarity with 'mat' → Medium
  - Others → Lower
- The model computes attention scores based on these similarities.
- It then combines the Values of the other words, weighted by attention scores.
- *Output*: A new representation of 'sat' that captures its relationship to 'cat' and 'mat'.

Attention in Transformers

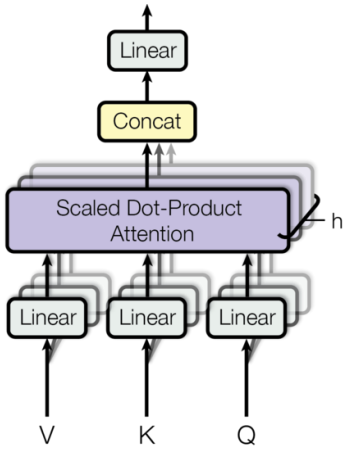
Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Multi-Head Attention

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Components of Self-Attention in Transformers

Component	Description	Function
Query, Key, Value Vectors	Each input token is transformed into Q, K, V vectors.	Enables calculation of attention scores and retrieval of information.
Attention Score Calculation	Scores calculated via dot product of Q and K vectors.	Determines focus on different parts of the sequence.
Softmax Layer	Applies softmax to attention scores.	Normalizes scores into a probability distribution.
Weighted Sum	Weighted sum of V vectors using softmax probabilities.	Aggregates information from the sequence based on attention.
Output	Resultant vector from weighted sum.	Serves as input for the next layer, represents aggregated information.

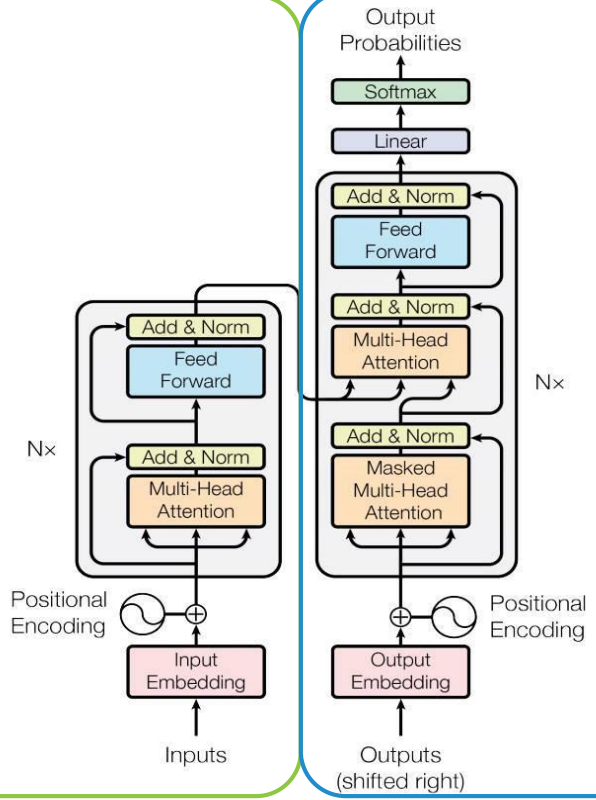
Challenges with Transformers

- **Training Cost & Complexity -** Requires massive compute, long training times, and specialized hardware (e.g., TPUs, GPUs).
- **Interpretability Limitations -** Attention maps offer some insights, but true decision-making processes largely remain opaque.
- **Resource Demands -** Large models consume extensive memory and power, making deployment difficult on edge or mobile devices.
- **Latency & Scalability -** Inference can be slow for large models, limiting real-time use cases like chat or vision on-device, unless distilled to much smaller models.
- **Environmental Impact -** Training large-scale models contributes to significant energy consumption and carbon emissions.

Transformers in Modern Research and Applications

Transformers for Representation & Generation

Representation



Generation

Transformers for Representation & Generation

Attributes	Encoder	Decoder
Role	Representation	Generation
Input	Entire input sequence	Encoded embeddings + partially generated sequence
Output	Contextual embeddings of input	Next token prediction or entire output sequence
Token Visibility	All input tokens visible to each other	Restricted to previous and current tokens only
Autoregressive	No	Yes
Models	BERT - Bidirectional Encoder Representations	GPT – Generative Pretrained Transformer

Applications of Transformers

Model Type	Description	Applications
GPT (e.g., GPT-4, GPT-4o)	Autoregressive language model trained to predict the next token; optimized for generation and reasoning.	Text generation, coding assistants, chatbots, summarization, tutoring.
BERT (and derivatives like RoBERTa, DeBERTa)	Bidirectional encoder trained with masked language modeling for contextual understanding.	Text classification, semantic search, question answering, information retrieval.
ViT (Vision Transformer)	Applies transformer architecture to images by treating image patches as tokens.	Image classification, segmentation, detection, vision-language tasks.
CLIP (Contrastive Language-Image Pre-training)	Aligns image and text embeddings using contrastive learning; enables cross-modal understanding.	Image-text retrieval, zero-shot classification, visual search, content filtering.

Applications of Transformers (cont.)

Model Type	Description	Applications
LLaVA (and successors like Gemini, GPT-4V)	Multimodal LLMs trained on both vision and language inputs for general-purpose reasoning.	Visual question answering, document analysis, multimodal chat, accessibility tools.
DALL·E (e.g., DALL·E 3)	Text-to-image generation model using transformer-based architecture for creativity and synthesis.	Image generation, concept art, advertising, product design.
Code LLMs (e.g., CodeLlama, Claude)	Transformers fine-tuned on code datasets for understanding and generating software.	Code generation, code explanation, debugging, AI pair programming.
Segment Anything (SAM)	Vision transformer trained to segment any object in an image given a prompt or click input.	Image editing, object detection, medical imaging, annotation tools.

Conclusion: Why Transformers Matter

- **Revolutionary Architecture -** Transformers have redefined NLP and extended their impact to vision, speech, and multi-modal AI.
- **State-of-the-Art Performance -** They outperform legacy models in accuracy, scalability, and cross-domain generalization.
- **Built for Scale -** Their design thrives on large data and compute, powering today's most advanced AI systems.
- **Ongoing Challenges -** High resource demands, interpretability issues, and environmental costs remain key hurdles.
- **A Platform for the Future -** Continued research is making transformers faster, leaner, and more adaptable across domains.

Resources

Understanding Transformers

"Attention Is All You Need" by Vaswani et al. (2017)
arxiv.org/abs/1706.03762

Language Translation with Transformers
pytorch.org/tutorials/beginner/translation_transformer

Hugging Face - Transformers
huggingface.co/docs/transformers

Rakshit Agrawal

Principal AI Scientist

Synthpop AI

LinkedIn: linkedin.com/in/rakshit-agrawal/

Advisory | Teaching | Speaking
rakshit@pragmainnov.com