# Post-Read Document

## Foundations of Generative AI and Advanced Prompting

This document is designed for participants who want to deepen their understanding beyond the classroom sessions. It expands on neural network foundations, encoding methods, attention mechanisms, transformer architectures, Mixture of Experts (MoE), and advanced prompting strategies.

---

# Part I — Foundations of Modern Generative AI

Before Transformers and large language models, several neural architectures laid the groundwork for today's systems. Understanding these helps in grasping why modern architectures evolved the way they did.

---

## 1. Deep Neural Networks (DNN)

A Deep Neural Network is a feedforward neural network with multiple hidden layers.

**Structure:**

- Input layer
- One or more hidden layers
- Output layer

Each layer performs:

Output = Activation(Weights × Input + Bias)

### Key Characteristics:

- Learns hierarchical feature representations
- Used in classification, regression, and structured prediction
- Forms the backbone of CNNs, RNNs, and Transformers

Limitations:

- Cannot handle sequential dependencies directly
- No memory of previous inputs

---

## 2. Recurrent Neural Networks (RNN)

RNNs were introduced to process sequential data.

### Core Idea:

Each output depends not only on the current input but also on the previous hidden state.

h_t = f(Wx_t + Uh_{t-1})

### Applications:

- Language modeling
- Speech recognition
- Time-series forecasting

Limitations:

- Vanishing gradient problem
- Difficulty learning long-range dependencies
- Sequential processing (not parallelizable)

Extensions:

- LSTM (Long Short-Term Memory)
- GRU (Gated Recurrent Unit)

These improved memory handling but still struggled with very long contexts.

---

## 3. Convolutional Neural Networks (CNN)

CNNs are designed primarily for spatial data (images).

### Key Concepts:

- Convolution filters
- Feature maps
- Pooling
- Local receptive fields

In NLP, CNNs were also used for:

- Sentence classification
- Character-level modeling

Limitation:

- Limited global context modeling
- Not ideal for long sequential dependencies

---

# Part II — Data Representation and Encoding

Neural networks require numerical inputs. Text must be converted into numeric representations.

---

## 4. One-Hot Encoding (OHE)

One-Hot Encoding represents each word as a sparse vector.

Example vocabulary:

["cat", "dog", "bird"]

Representation:

- cat → [1, 0, 0]
- dog → [0, 1, 0]
- bird → [0, 0, 1]

Limitations:

- High dimensional
- Sparse
- No semantic similarity

---

## 5. Word Embeddings

Embeddings convert words into dense vectors where semantic similarity is captured geometrically.

Example:

- “king” − “man” + “woman” ≈ “queen”

Embedding models:

- Word2Vec
- GloVe
- FastText

Modern LLMs learn contextual embeddings dynamically.

---

# Part III — Attention Mechanism

RNN limitations led to the introduction of attention.

## 6. Why Attention?

Instead of compressing an entire sentence into a single vector, attention allows the model to focus on relevant words when generating each output.

---

## 7. Core Attention Formula

Given Query (Q), Key (K), and Value (V):

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

This computes similarity between tokens and weighs their importance dynamically.

Benefits:

- Captures long-range dependencies
- Enables parallel processing
- Improves translation and reasoning tasks

---

# Part IV — Transformers

Transformers replaced recurrence with attention.

## 8. Transformer Architecture Overview

Core Components:

- Input embeddings
- Positional encoding
- Multi-head self-attention

- Feedforward layers
- Layer normalization
- Residual connections

Key Idea:

Every token attends to every other token.

---

## 9. Self-Attention

Self-attention allows tokens within a sequence to interact with each other.

Example:

In the sentence:

"The animal didn't cross the road because it was tired."

Self-attention helps determine that "it" refers to "animal".

---

## 10. Multi-Head Attention

Instead of one attention mechanism, multiple heads learn different relationships simultaneously.

Benefits:

- Captures syntax and semantics
- Learns multiple relational patterns

---

## 11. Sparse Attention

Standard attention is quadratic in complexity ($O(n^2)$). Sparse attention reduces computation by:

- Limiting attention to local windows
- Using block patterns

- Applying global tokens selectively

This enables longer context handling.

---

# Part V — Mixture of Experts (MoE)

MoE improves scalability and efficiency.

## 12. What is MoE?

Instead of activating all parameters for every input:

- Multiple expert networks exist
- A gating network selects which experts to use
- Only selected experts are activated

Benefits:

- Larger models with lower compute per token
- Efficient scaling
- Improved specialization

Used in many modern large language models.

---

# Part VI — Multimodal Models

Modern GenAI systems process:

- Text
- Images
- Audio
- Video

Multimodal models use shared embedding spaces to connect modalities.

Example:

- Text-to-image generation
- Image captioning
- Visual question answering

---

# Part VII — Advanced Prompt Engineering Concepts

Beyond basic prompting, advanced strategies improve reasoning and reliability.

---

## 13. Prompt Components

A structured prompt may include:

- Role definition
- Task instruction
- Constraints
- Examples
- Output format

---

## 14. Zero, One, and Few-Shot Prompting

Zero-shot:

No examples provided.

One-shot:

One example included.

Few-shot:

Multiple examples included.

---

## 15. Chain-of-Thought (CoT)

Encourages step-by-step reasoning before producing a final answer.

Improves:

- Arithmetic reasoning
- Logical tasks
- Structured problem solving

---

## 16. Auto-CoT

Automatically generates reasoning examples before solving tasks.

---

## 17. Least-to-Most Prompting

Breaks complex tasks into simpler sub-problems.

Solves them sequentially.

---

## 18. ReAct Prompting

Combines reasoning and acting:

- Think
- Take action (e.g., retrieve information)
- Observe
- Continue reasoning

Often used in agent-based systems.

---

## 19. Self-Consistency

Instead of generating one reasoning path:

- Generate multiple reasoning paths
- Select the most consistent answer

Improves reliability.

---

## 20. Generate Knowledge Prompting

Model first generates background knowledge, then solves the task.

Improves performance in knowledge-heavy queries.

---

## 21. Tree of Thought

Explores multiple reasoning branches before selecting the best path.

Used for complex decision-making problems.

---

## 22. Reflection Prompting

The model:

- Generates an answer
- Critiques it
- Refines it

Improves accuracy and reduces hallucinations.

---

# Suggested Deeper Study Path

1. Study gradient descent and backpropagation mathematically
2. Implement a basic DNN from scratch
3. Train a small RNN for sequence prediction
4. Implement attention in PyTorch
5. Build a minimal Transformer encoder
6. Experiment with different prompting strategies
7. Analyze trade-offs between dense and MoE architectures

---

# Closing Perspective

Modern Generative AI did not emerge suddenly. It evolved through:

DNN → CNN/RNN → Attention → Transformers → MoE → Multimodal Systems

Understanding this progression enables better architectural decisions, more effective prompting, and deeper system-level thinking.

Mastery comes from combining theory, experimentation, and architectural analysis.