Additional Notes

1. **Input Parameters**

Input parameters are essential components of prompt templates that allow for customization based on specific needs. They act as placeholders that can be filled with relevant information to tailor the prompt to the user's query. Here are the common input parameters with detailed explanations and examples:

- User query: The specific task the user wants to address.
- Context: Background information that helps the LLM understand the query better.
- Tone: The desired tone of the response (e.g., formal, casual, empathetic).
- Format: The structure in which the response should be delivered (e.g., list, paragraph, bullet points).

Example

Let's use an industrial example to illustrate the combined use of input parameters in a prompt template.

Template: "Provide a detailed explanation of [User Query] suitable for [Context]. Respond in a [Tone] manner and structure the information in [Format]."

Filled Template: "Provide a detailed explanation of 'the process of quality control in manufacturing' suitable for 'a training session for new employees'. Respond in a 'formal' manner and structure the information in 'bullet points'."

Explanation:

- User Query: 'the process of quality control in manufacturing'
- Context: 'a training session for new employees'
- Tone: 'formal'
- Format: 'bullet points'

2. **Expected Output Format**

In a Prompt Template, the Expected Output Format section guides the Large Language Model (LLM) on how to structure its response. This ensures that outputs are consistent, clear, and aligned with user expectations—especially critical for tasks involving automation, documentation, or standardized communication. While the specific format may vary depending on the task, a well-structured response typically includes the following components:

- **Introduction**: A brief overview or acknowledgment of the user’s query, setting the context for the response.

- **Body**: A detailed explanation, analysis, or solution that directly addresses the query. This section should be comprehensive and tailored to the user's intent.
- **Conclusion**: A concise summary or closing statement that reinforces the key points or next steps.
- **Formatting Guidelines**: Utilize Markdown formatting to enhance readability and structure. This may include bold text, italics, bullets, or code blocks.

A consistent output format not only improves the user experience but also helps in better understanding and actionability of the LLM's responses.

3. Example/Scenario of Usage

Prompt templates are flexible tools that can be adapted for use in different domains such as customer service, education, healthcare, software development, marketing, and more.

- **Scenario**: Responding to a customer complaint.
- **Instruction**: Compose a polite and professional apology.
- **Context**: The customer received a defective product and reported it via email.
- **Input Data**: Customer name is John; the product is a Bluetooth speaker ordered on April 10.
- **Output Format**: Apology message in paragraph form, offering a replacement and discount code.

4. Limitations or Known Issues

Identifies any limitations or common issues (technical issues) that may arise when using the template. Documenting these known issues helps avoiding or be cautious using the template in situation where it might fail.

- **Context Sensitivity**: May not capture the full context of complex queries.
- **Overfitting**: Templates might become too rigid, limiting creativity.
- **Maintenance**: Requires regular updates to stay relevant.
- **Bias**: Templates can inadvertently introduce bias if not carefully designed.

5. Version History

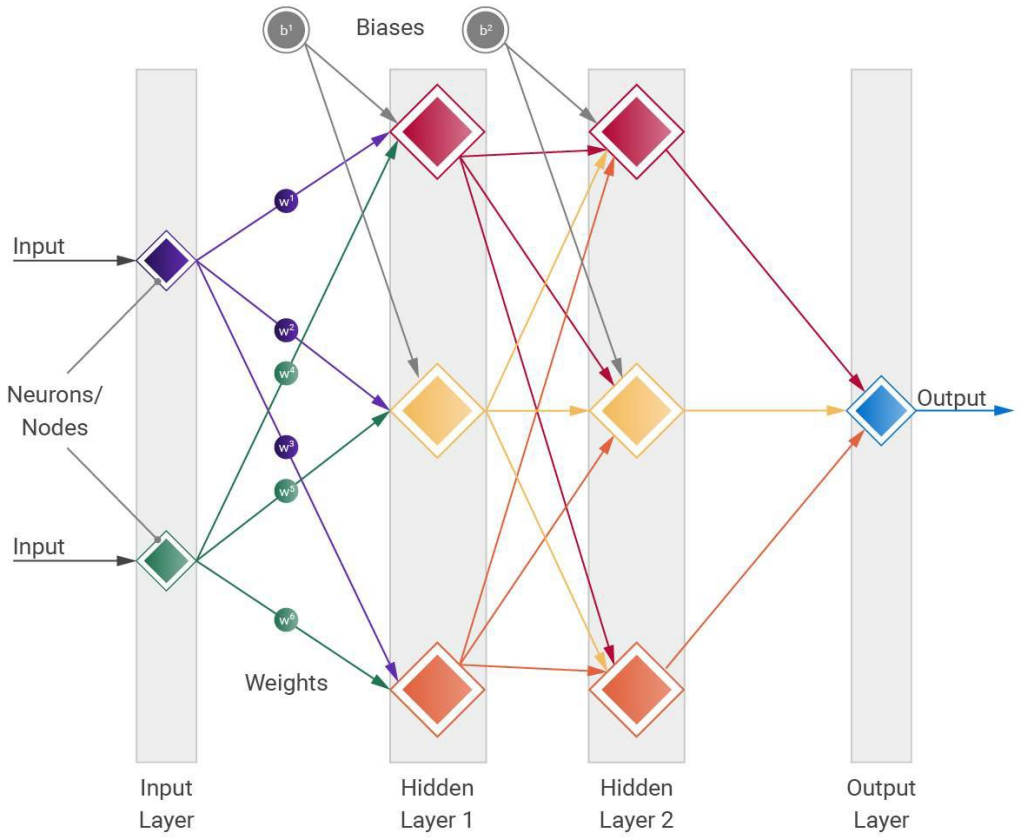
Maintaining a version history of prompt templates ensures traceability and continuous improvement. Tracking changes helps maintain the template's relevance and effectiveness

Example:

Version	Date	Changes Made	Author
1.0	2025-01-10	Initial template creation	A. Sharma
1.1	2025-02-15	Added context parameter for clarity	B. Mehta
1.2	2025-03-20	Updated output format to include JSON option	C. Gupta

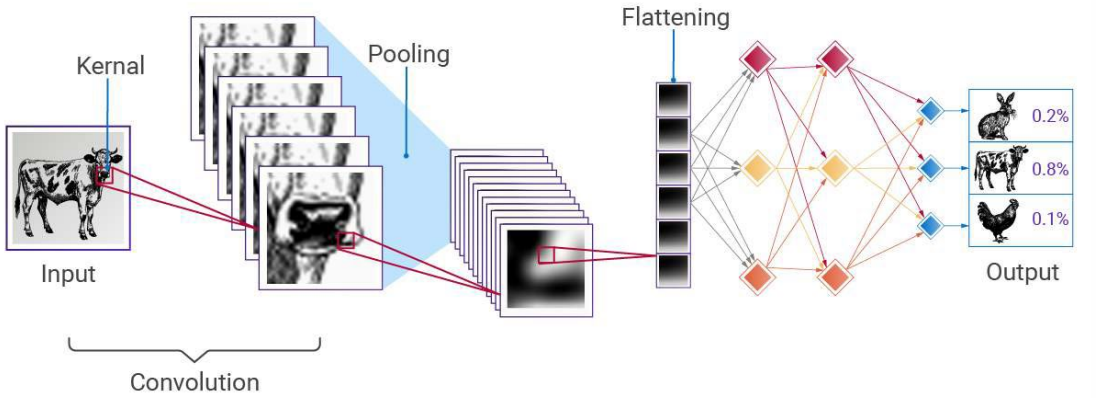
Insights into Key Neural Network Architectures

1. Artificial Neural Networks/Feedforward Networks



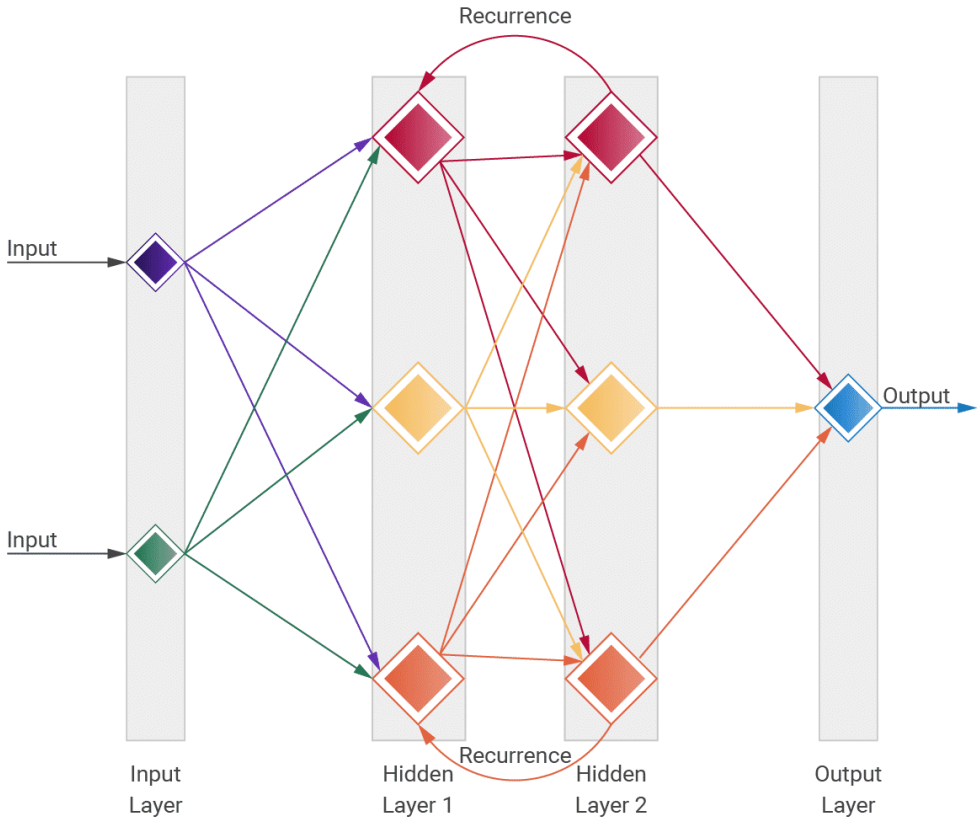
- Straight Path: Data moves in one direction, from input to output, without looping back.
- Layers: Consists of an input layer, one or more hidden layers, and an output layer. Each layer has neurons that process the data.
- Activation Functions: Uses functions like ReLU (Rectified Linear Unit) or sigmoid to introduce non-linearity, helping the network learn complex patterns.
- Learning: Adjusts weights using backpropagation, which calculates the error between predicted and actual outputs and updates the weights to minimize this error.

2. Convolutional Neural Networks (CNN)



- Filters: Uses convolutional layers with filters (kernels) to scan the input image and extract features like edges, textures, and shapes.
- Pooling: Applies pooling layers (e.g., max pooling) to reduce the size of the feature maps while retaining important information, making the network more efficient.
- Connections: Combines features from convolutional and pooling layers in fully connected layers to make final predictions.
- Training: Uses techniques like data augmentation (flipping, rotating images) to improve model performance and generalization.

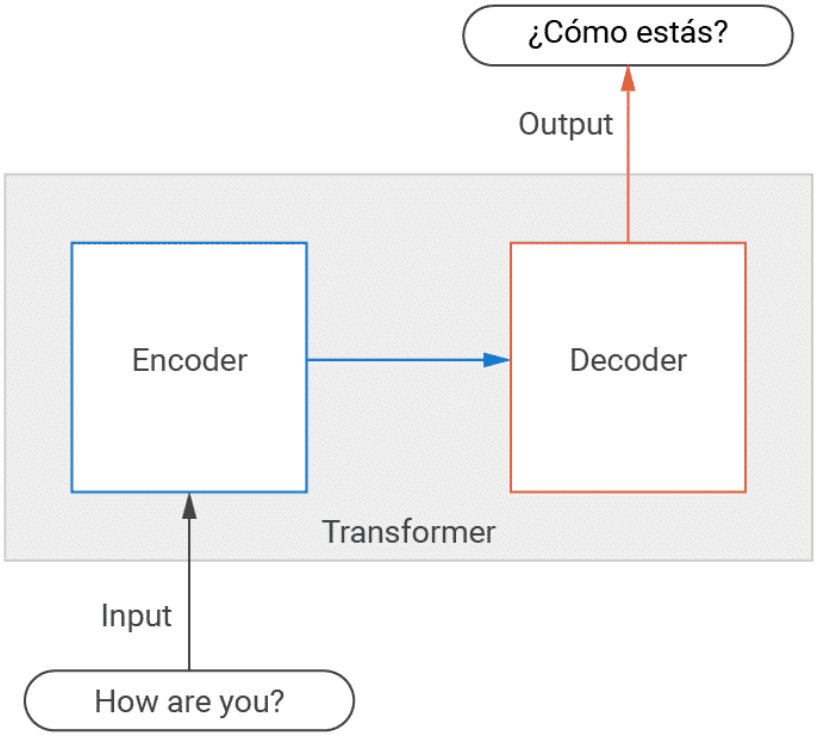
3. Recurrent Neural Networks (RNN)



- Sequences: Designed to handle sequential data, such as time series, text, or speech, by processing one element at a time.
- Memory: Maintains hidden states that capture information from previous time steps, allowing the network to remember past inputs.

- Loops: Data can loop back, enabling the network to use information from previous steps to influence current processing.
- Learning: Uses backpropagation through time (BPTT) to adjust weights based on errors across the entire sequence.

4. **Transformer-based Neural Networks**



- Attention: Uses an attention mechanism to focus on different parts of the input data, allowing the model to weigh the importance of each part.
- Self-Attention: Enables the model to consider the entire input sequence simultaneously, improving context understanding and capturing relationships between distant elements.
- Layers: Consists of encoder and decoder layers, each with multiple attention heads that process the data in parallel.
- Training: Learns from large datasets and uses techniques like transfer learning to achieve high performance on tasks like language translation, text generation, and more.