1. Applications for Different RNN Architectures

Sequence-to-Sequence RNN:
Takes a sequence as input and produces a sequence as output.
Examples:

- **Machine Translation (e.g., English to French)**
- **Text Summarization**
- **Dialogue Systems**
- **Speech Recognition**

Sequence-to-Vector RNN:
Takes a sequence input and produces a fixed-size vector.
Examples:

- **Sentiment Analysis**
- **Document Classification**
- **Video encoding before classification**

Vector-to-Sequence RNN:
Takes a fixed-size vector input and produces a sequence.
Examples:

- **Image Captioning**
- **Text Generation from features (e.g., chatbot replies)**
- **Speech synthesis from text**

---

2. Why Use Encoder–Decoder RNNs Instead of Plain Sequence-to-Sequence RNNs for Translation?

Encoder–decoder RNNs separate the model into two parts:

- **Encoder compresses the input sequence into a fixed-size context vector.**
- **Decoder generates the output sequence from this context.**

Advantages:

- **More flexible than a single sequence-to-sequence model.**
- **Handles variable-length inputs and outputs.**
- **Easily extendable with attention mechanisms for better context handling.**

---

3. Combining CNN and RNN for Video Classification

- **Use a CNN to extract features from each video frame.**
- **Feed the sequence of feature vectors into an RNN to capture temporal information.**
- **Output could be a class label or sequence of actions.**

Example Flow:
Video → CNN (frame features) → RNN → Class Label (e.g., "walking", "running")

---

4. Advantages of Using `dynamic_rnn()` Over `static_rnn()`

- **Memory Efficient: Dynamic RNN builds the computation graph at runtime, saving memory.**
- **Faster: Especially for variable-length sequences.**
- **Easier to Handle Variable Sequence Lengths: Automatically stops unrolling when sequence ends.**

---

5. Dealing with Variable-Length Sequences

Input Sequences:

- **Padding: Pad shorter sequences to the max length in the batch.**
- **Masking: Use masks or specify sequence lengths so the model ignores padding during training.**

Output Sequences:

- **Use decoder with attention or teacher forcing.**
- **Stop generation using a special end-of-sequence (EOS) token.**

---

6. Common Way to Distribute RNN Training Across Multiple GPUs

- **Data Parallelism: Split each batch of data across GPUs. Each GPU computes gradients on its subset. Then, gradients are averaged and weights updated.**
- **Use frameworks like TensorFlow's `MirroredStrategy` or PyTorch's `DataParallel` or `DistributedDataParallel`.**
- **You can also split model layers across GPUs for model parallelism, though it's less common for RNNs.**
