Encoder Decoder Models

can code communities

Review Questions

- How does a LSTM improve on the architecture of a RNN?
- What do convolutional layers do within a CNN?
- How does a GAN’s generator model learn?

Review Take Home

What is the potential societal impact and ethical considerations of GAN technology? Find one or two examples of AI generated media that have been used to spread misinformation.

Encoder

- An encoder acts like a compressor.
- It takes a piece of information (like an image, text, or a sequence of data) and converts it into a more compact representation.
- This compressed representation, often called a code or a latent space, captures the essential features of the input data in a smaller format.
    - Image Encoders: Might compress an image into a lower-dimensional vector containing key features like edges, shapes, and colors (similar to pooling layers of a CNN).
    - Text Encoders: Can convert text into a sequence of numbers representing words or even capture semantic meaning of the text (similar to embeddings).

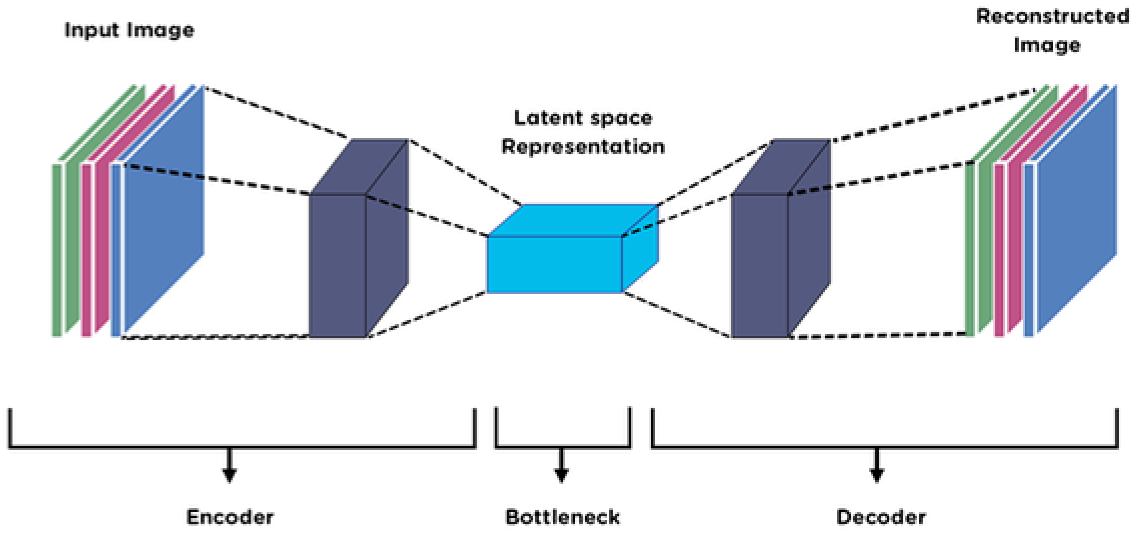
Decoder

- A decoder acts like an expander.
- It takes the compressed code generated by the encoder and attempts to reconstruct the original data or generate something similar to it.
- Ideally, the decoder leverages the code to recreate the original input as accurately as possible, or generate a new instance that shares the same characteristics as the data it was trained on.
    - Image Decoders: Can use the encoded vector to recreate the original image, or even generate a new image that maintains the encoded features (like generating a new image with a similar style but a different content).
    - Text Decoders: Might use the encoded sequence of numbers to generate text that resembles the original input, or even translate text from one language to another based on the learned encoding.

Autoencoder

- Autoencoders consist of an encoder network and a decoder network that reconstructs the input from the latent representation.
- Autoencoders learn to compress data into a lower-dimensional representation and then regenerate the original data from that compressed version (known as a bottleneck layer).
    - They can be used for dimensionality reduction or anomaly detection.
- Variational Autoencoders (VAEs) and Denoising Autoencoders are popular variations with additional constraints or noise injection for improved robustness and generative capabilities.
    - Applications: Dimensionality reduction, data denoising, generative modeling.

Autoencoder Diagram



Example

Go through the autoencoder example to better understand the encoder and decoder utilization and autoencoder tasks like denoising.

Attention

Attention mechanisms enable neural networks to focus on specific parts of the input data while processing sequences, allowing for more effective information processing. They have been widely adopted in various neural network architectures, enhancing model performance in tasks such as machine translation, text summarization, and image captioning.

How Attention Works

There are different ways to implement attention mechanisms, but here's a general concept:

- Representation: Each element in the input sequence is transformed into a numerical representation (vector).
- Scoring: A score is calculated for each pair of elements in the sequence. This score reflects how relevant one element is to another in the context of the current task.
- Weighting: Based on these scores, weights are assigned to each element. Higher weights indicate a stronger focus on that element's contribution.
- Context Vector: A context vector is created by combining the element representations weighted by their attention scores. This context vector captures the most relevant information from the entire sequence for the current step.

Regular Attention vs. Self-Attention

- Regular Attention focuses on the relationship between elements in an input sequence and an external query.
  - For example, in machine translation, the attention mechanism might focus on relevant parts of a source language sentence (input sequence) for translating a specific word in the target language (query).
  - It helps the model understand how different parts of the input sequence contribute to the current prediction.
- Self-Attention focuses solely on the relationships within a single input sequence.
  - The model essentially "attends" to different parts of its own input to understand the context and relationships between them.
  - This is particularly useful for tasks like text summarization or sentiment analysis, where understanding the relationships between words within a sentence is crucial.

Self-Attention Mechanism

In self-attention, each input position is associated with three vectors: Query, Key, and Value.

- Query Vector (Q): This vector represents the "current focus" of the model.
    - It can be thought of as the question the model is asking about a specific word in the sequence.
- Key Vector (K): This vector captures the "informational content" of each word.
    - It acts like a summary of the information a word holds.
- Value Vector (V): This vector holds the actual content or representation of each word.
    - It's the information the model might want to "pay attention to" based on the query and key vectors.

The Self-Attention Process

- Generate Query, Key, and Value Vectors: The model transforms each word in the input sequence into three vectors: query (Q), key (K), and value (V).
- Calculate Attention Scores: The model computes a score for each pair of words in the sequence.
    - This score represents how relevant one word (value vector) is to the current focus (query vector) based on the information it contains (key vector).
- Apply Softmax: A softmax function is applied to the scores, converting them into probabilities.
    - These probabilities indicate how much "attention" each word should receive in the context of the current focus.
- Weighted Sum: The value vectors of all words are weighted by their corresponding attention scores and summed up.
    - This creates a context vector that captures the most relevant information from the entire sequence based on the current focus.

Multi-Head Attention

- While self-attention offers significant benefits, relying on a single set of query, key, and value vectors might limit the model's ability to capture diverse aspects of the relationships between words in a sequence.
    - The single attention head might focus on a specific type of relationship or pattern, potentially missing other crucial information.
- Multi-head attention addresses this limitation by creating multiple "heads" of self-attention, each with its own independent set of Q, K, and V vectors.
    - In essence, we stack multiple self-attention layers on top of each other, where each layer performs self-attention with its own unique set of vectors.

Multi-Head Attention Process

- Linear Transformations: Each word in the input sequence is projected into multiple sets of query (Q), key (K), and value (V) vectors, one set for each attention head.
- Independent Self-Attention: Each head performs self-attention independently using its own Q, K, and V vectors.
  - This allows each head to focus on potentially different aspects of the relationships within the sequence.
- Concatenation: The outputs (context vectors) from all the heads are concatenated into a single vector.
  - This combined vector captures a richer representation of the sequence by incorporating information learned from various perspectives (different heads).

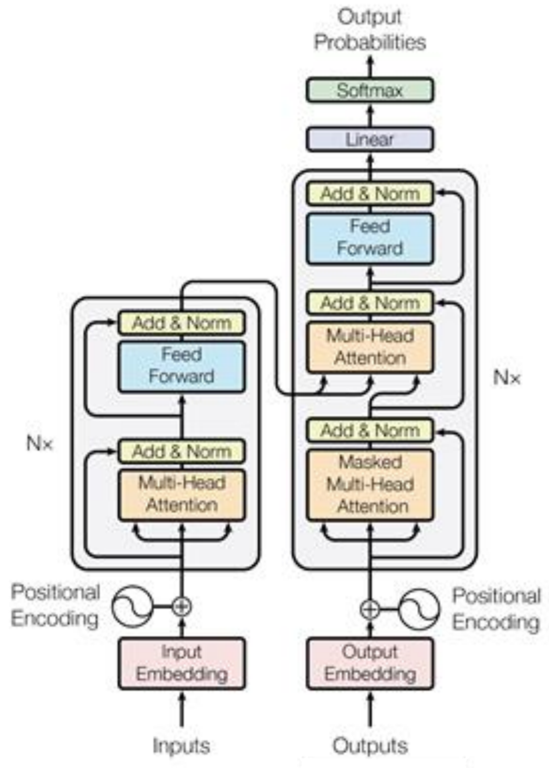
Benefits of Multi-Head Attention

- Improved Representation: By learning from multiple heads, the model captures a more comprehensive understanding of the relationships within the sequence.
- Attention to Diverse Patterns: Different heads can focus on various aspects of the relationships, like grammatical structure, word co-occurrence, or semantic similarity.
- State-of-the-Art Performance: Multi-head attention is a core component of transformers, which have achieved remarkable results in various NLP tasks.

Transformer Model

- Introduced by Vaswani et al. in the paper "Attention is All You Need," revolutionized natural language processing tasks by relying solely on attention mechanisms without recurrent or convolutional layers.
- Unlike traditional RNNs that process sequences sequentially, transformers adopt an entirely different approach.
- They rely on an encoder-decoder structure.
  - Within both the encoder and decoder, transformers extensively use multi-head attention.
  - This allows the model to focus on crucial parts of the input sequence, critical for grasping context and relationships between words.

Transformer Model Diagram



Positional Encoding

- While self-attention is powerful, it doesn't inherently capture the order of words in a sequence. This is where positional encoding comes in.
- Positional encoding is a technique that injects information about the order of words into the model's inputs.
  - This allows the model to understand the relative position of words within a sequence, even though attention mechanisms themselves don't directly process information in a sequential manner.
- Various positional encoding schemes exist, but the core idea is to add positional information to the word vectors before feeding them into the self-attention layers.

Feed Forward Networks

- Transformers don't solely rely on attention mechanisms.
- They incorporate feed-forward neural networks within both the encoder and decoder after each multi-head attention layer.
  - Add non-linearity to the model.
  - This allows the model to learn more complex relationships between words and capture patterns that might not be easily captured by attention alone.

Transformer Journey (Preprocessing and Encoder)

Now that we've explored the key components of a transformer model, let's delve into the fascinating path a piece of text takes as it travels through the model.

The journey begins with the input text.

- The text is preprocessed, which might involve steps like removing punctuation, converting text to lowercase, and handling special characters.
- Next, each word in the preprocessed text is converted into a numerical vector representation using a technique called word embedding.
- The input sequence is then fed into the encoder, where positional encoding and then multi-head attention is performed.
- The output of the multi-head attention layer is then passed through a feed forward network.
    - After both the multi-head attention and the feed forward network, there are residual connections and layer normalizations applied which helps with training stability.

The Transformer Journey (Decoder)

- The decoder attends to its own previously generated outputs using multi-head attention but with a mask applied.
  - If there are no previous outputs as it is the initial step, it might be initialized with a special "start of sentence" token embedding.
  - The mask prevents the model from attending to future words in the training output sequence that haven't been generated yet.
- In addition to the masked self-attention, the decoder also attends to the encoded representation from the encoder using another multi-head attention layer (this one is not masked).
  - This allows the decoder to focus on relevant parts of the context (captured in the encoded representation) when generating each word in the output sequence.
- Similarly to the encoder, the decoder uses a feed forward network for additional processing.
  - There are also residual connections and layer normalization after the mask self-attention, the normal self-attention, and the feed forward network which help with training stability.

The Transformer Journey (Output Generation)

- After processing the entire sequence through the decoder layers, the final decoder output is fed into a softmax layer.
    - The softmax layer converts the decoder output into a probability distribution over all possible words in the vocabulary.
    - This distribution indicates the likelihood of each word appearing as the next word in the output sequence.

In machine translation: The model selects the word with the highest probability as the predicted translation for the next word in the target language. In text summarization: The model might select a subset of words with high probabilities to include in the summary.

Exercise

Explain how using attention mechanisms in a transformer can help when translating a sentence from English to Spanish.

Teacher Forcing

During training, a technique called teacher forcing can be used to improve the model's learning process.
In teacher forcing, the correct next word from the target sequence is fed back into the decoder as input during training. This helps the model learn to generate the desired output sequences.

World of Neural Networks

Remember, this is just a glimpse into the diverse world of neural networks. As research continues, new architectures and applications are constantly emerging. Here are a couple of other notable structures and a small description.

- Capsule Networks (CapsNets)
    - Capsule Networks are designed to improve upon the limitations of CNNs in handling hierarchical relationships between features.
    - Capsule Networks utilize capsules, which are groups of neurons representing ideal representation parameters of a specific type of entity, such as the pose, viewpoint, or deformation of an object.
    - Capsule Networks show promise in tasks requiring spatial hierarchies and viewpoint invariance.

World of Neural Networks

- Residual Networks (ResNets)
  - Residual Networks are a type of deep neural network architecture that facilitates training of very deep networks by introducing skip connections.
  - ResNets contain residual blocks, each comprising multiple layers, where the input to a layer is combined with the output through skip connections.
  - Skip connections allow gradients to flow more easily during training, alleviating the vanishing gradient problem and enabling the training of deeper networks.
  - ResNets have achieved state-of-the-art performance in various computer vision tasks and are widely used in image classification, object detection, and semantic segmentation.

Take Home Exercise

Now that we've covered the popular model architectures in the current AI landscape, let's practice choosing the right model for the specific task. Below are three examples of real world tasks that AI will be able to help perform. Choose which model architecture is best suited for this task and explain why.

1. Develop an image classification system to classify satellite images into different land use categories (e.g., urban, agricultural, forested).
2. Develop a text summarization system to generate concise summaries of research papers in the field of natural language processing.
3. Develop a dialogue generation system capable of generating realistic and contextually coherent conversations between virtual agents.