Report on Transformer Model Implementation

Harshit Kumar

Introduction

The project aimed to implement a Transformer model from scratch for a machine translation task. The Transformer architecture, introduced by Vaswani et al. in 2017, represents a significant departure from previous sequence-to-sequence models by relying entirely on self-attention mechanisms to process input data in parallel, improving training speed and performance on translation tasks.

Implementation Process

The implementation followed the original Transformer model structure, comprising an encoder and decoder, each with multiple layers of self-attention and position-wise feedforward networks.

1. Model Architecture: Defined classes for the Transformer model, including the encoder, decoder, multi-head attention mechanism, position-wise feed-forward networks, positional encoding, and embedding layers. The `Encoder` and `Decoder` were built with stacks of `EncoderLayer` and `DecoderLayer`, each incorporating normalization and dropout.

2. Training Setup: Utilized the Adam optimizer alongside a CrossEntropyLoss function to train the model on a machine translation task from German to English on the Multi30k dataset.

3. Data Preparation: Involved tokenizing text data into source and target languages, mapping tokens to numerical indices, and batching for training.

4. Hyperparameters: Set initial hyperparameters based on recommendations from the original paper, including the number of layers (N=6), model dimension (d_model=512), number of heads (num_heads=8), feed-forward dimension (d_ff=2048), and dropout rate (dropout=0.1). Performed hyperparameter tuning for the hyperparameters.

5. Beam Search Implementation: Modified the greedy decoding strategy to include beam search for better translation quality, allowing the model to consider multiple translation hypotheses at each step.

Hyperparameter tuning and Greedy search vs Beam Search Results

Hyperparameter set 1
```
src_vocab_size = len(vocab_src)  # Size of source vocabulary
tgt_vocab_size = len(vocab_tgt)  # Size of target vocabulary
d_model = 512  # Embedding dimension
N = 6          # Number of encoder and decoder layers
num_heads = 8  # Number of attention heads
d_ff = 2048    # Dimension of feed forward networks
max_seq_length = 5000 # Maximum sequence length
dropout = 0.1  # Dropout rate
batch_size = 128
grad_clip = 1
Optimizer = Adam
Learning rate=0.0001
Adam betas=(0.9, 0.98)
```

Epoch: 20
    Train Loss: 2.852
    Val Loss: 3.141

Greedy Decoding vs Beam Search
German: `Ein kleiner Junge spielt draußen mit einem Ball.`
English: `A little boy playing outside with a ball.`
Greedy decoding: A young boy is playing in a pool.
Beam search: A young boy is playing in a pool.

German: `Ein Mann in einem Anzug steht auf einer Bühne und spricht in ein Mikrofon.`
English: `A man in a suit stands on a stage and speaks into a microphone.`
Greedy decoding: A man in a hat and a woman in a black dress are sitting on a bench.
Beam search: A man with glasses and a beard is sitting on a bench in front of a microphone.

German: `Ich liebe indisches Essen.`
English: `I love Indian food.`
Greedy decoding: A crowd of people are working together.
Beam search: A crowd of people are dancing.

German: `Wer übernachtet in der Snell-Bibliothek?`
English: `Who stays at night in Snell library?`
Greedy decoding: Construction workers are working on a street.
Beam search: Construction workers are working on a street.

Hyperparameter set 2
```
# Changed hyperparameters
```

```
d_model = 768  # Embedding dimension
N = 7          # Number of encoder and decoder layers
```

Epoch: 20
Train Loss: 4.586
Val Loss: 14.489
Translation results were bad; clearly, validation loss increased a lot due to overfitting. Received the recurring 'A' token for every input sentence.
Translated sentence: A A A A A A A A A A A A A

Hyperparameter set 3: Replicating GPT2
```
# Changed hyperparameters
d_model = 768    # Embedding dimension to match GPT-2 small
N = 12           # Number of layers; GPT-2 is decoder-only, but for a seq2seq model, this could apply to each part
num_heads = 12   # Number of attention heads to match GPT-2 small
d_ff = 3072      # Dimension of feed-forward networks, approximated to GPT-2's configuration
max_seq_length = 1024 # Adjusted maximum sequence length to match GPT-2's capability
dropout = 0.3  # Dropout rate increased to handle overfitting
learning rate = 0.00001
batch_size = 512
```

Epoch: 20
Train Loss: 4.946
Val Loss: 5.289
The model seems to be overfitting in this case as well.
Translated sentence: A man in in a a a a a a a . .
Translated sentence: A man in in a a a a a a a a a a a a a a .
Translated sentence: A are are are the .
Translated sentence: A are are are a . .

Hyperparameter set 4: Increasing attention heads
```
# Changed hyperparameters
d_model = 528  # Embedding dimension
num_heads = 12  # Number of attention heads
```

Epoch: 20
Train Loss: 2.858
Val Loss: 3.172
Translated sentence: A young boy is playing in a pool.
Translated sentence: A man and a woman are sitting in front of a woman in a white shirt.
Translated sentence: Construction workers are working at a party.
Translated sentence: Construction workers are working on the street.

Observations:

1. Increasing number of encoder and decoder layers resulted in overfitting.
2. Replicating GPT2 parameters resulted in overfitting of the model even after increasing the dropout and reducing learning rate.
3. Increasing the number of attention heads require more number of epochs for the model to reach optimum.

Challenges Faced

1. Loading the spacy tokenizer for the Multi30k dataset, required reloading the Colab session after installing portalocker package.
2. Understanding the Self-Attention Mechanism: Implementing the forward method required understanding the correct dimensions of the intermediate tensors.
3. Beam Search implementation: Spent quite some time fixing the correct way to use tokenization.
4. Memory Constraints: Larger batch sizes and model dimensions significantly increased GPU memory requirements, limiting experimentation with larger models.
5. Hyperparameter Tuning: Finding the optimal set of hyperparameters required extensive experimentation and was computationally expensive.

Lessons Learned

1. Understood the self-attention mechanism by implementing it from scratch.
2. The Transformer's ability to process data in parallel significantly speeds up training compared to RNNs.
3. Learned how positional encodings inject sequence order information, enabling the model to consider the order of tokens despite the absence of recurrence or convolution.
4. Implementing beam search gave better quality translation results compared to greedy decoding.
5. Transformer training is quite sensitive to hyperparameters used. Doubling embedding layers and the number of attention layers without carefully changing the other parameters worsened validation loss.