Introduction

Problem + Research Question
- Data is being generated at a rapid rate. How can we more efficiently compress text data?
- Can transformers be leveraged to more efficiently compress text?

Background
- Existing RNN + encoding approaches for character-based text compression
- Existing NN approaches for image compression

Methods

Model:
- Pretrained GPT2 model, finetuned on text generation task on the wikitext2 dataset.
- Hyperparameters: learning rate of 2e-05, training and evaluation batch size of 8, Adam optimizer

Variable-Length Integer Encoding
- Compresses fixed-length integers into variable-length integers by storing smaller numbers with fewer bits

Arithmetic Encoding
- Frequently used characters are stored with fewer bits and less frequent characters stored with more bits

Experiments & Results

Compression Ratio Histogram (Uncompressed/Compressed)

<table>
<thead>
<tr>
<th>Compression Ratio</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip_Baseline</td>
<td>2.02</td>
<td>1.51</td>
<td>2.61</td>
<td>0.16</td>
</tr>
<tr>
<td>Arith_Baseline</td>
<td>2.20</td>
<td>1.64</td>
<td>2.66</td>
<td>0.15</td>
</tr>
<tr>
<td>Neural_Variant</td>
<td>4.44</td>
<td>3.22</td>
<td>5.49</td>
<td>0.32</td>
</tr>
<tr>
<td>Neural_Arith</td>
<td>4.52</td>
<td>3.05</td>
<td>5.63</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Window Size vs Compression Ratio

Window Size vs Runtime

Discussion

Analysis
- Our results demonstrate the inherent trade-off between compression ratio and compression speed.
- While neural compression approaches are superior in compression ratio, they are inferior in compression speed.
- Smaller windows lead to far better runtime with only negligible reduction in compression ratio
- Compression ratio consistent across file sizes

Future Improvements
- Runtime:
  - Synthesizer attention
  - Fewer layers
  - Improved batching
- Compression ratio
  - Fine-tune with alternative loss term
  - Use 4-bit var-int

References
