HAT: Hardware-Aware Transformers for Efficient Natural Language Processing

Hanrui Wang¹, Zhanghao Wu¹, Zhijian Liu¹, Han Cai¹, Ligeng Zhu¹, Chuang Gan² and Song Han¹
¹Massachusetts Institute of Technology
²MIT-IBM Watson AI Lab
Transformers are Inefficient

- Raspberry Pi takes 20 seconds to translate a 30-token sentence with Transformer-Big model.
Searching for an Efficient Transformers is Inefficient

We need **Green AI** and **Tiny AI**

---

**Table 1**

<table>
<thead>
<tr>
<th></th>
<th>Design cost measured in CO₂ emission (lbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Life (Avg. 1 year)</td>
<td>11,023</td>
</tr>
<tr>
<td>American Life (Avg. 1 year)</td>
<td>36,156</td>
</tr>
<tr>
<td>US Car w/ Fuel (Avg. 1 lifetime)</td>
<td>126,000</td>
</tr>
<tr>
<td>Evolved Transformer</td>
<td>626,155</td>
</tr>
<tr>
<td>HAT (Ours)</td>
<td>52</td>
</tr>
</tbody>
</table>

- Previous AutoML has huge searching cost and CO₂ emission*

---

*Strubell, E., Ganesh, A., & McCallum, A. Energy and policy considerations for deep learning in NLP. ACL 2019

HAT: Hardware-Aware Transformers, ACL 2020
Hardware-Aware Transformers

Efficiently search for efficient Transformer architectures

- 250 GPU Years
  - $5,500,000
  - 626,000 lbs CO₂

- 184 GPU Hours
  - $456
  - 52 lbs CO₂

Search in a weight-sharing supernet “SuperTransformer”

- 737ms on Xeon
  - 176M model size
  - 10.6 GFLOPs

- 329ms on Xeon
  - 44M model size
  - 2.7 GFLOPs

(1) Hardware latency feedback in the search loop
(2) Novel design space
Two Common Pitfalls in Efficiency Evaluation

- FLOPs cannot reflect real latency

Nvidia GPU Latency

- Hidden Dimension Scaling
- Layer Number Scaling

Similar FLOPs, Different Latency
Two Common Pitfalls in Efficiency Evaluation

- Latency is influenced by different factors on different hardware
- Different hardware has different strategies for efficient model design
From General to Specialized Transformers

Previous Paradigm:
A general Transformer for all hardware

Low Efficiency

HAT: Hardware-Aware Transformers, ACL 2020

HAT:
One specialized Transformer model for each hardware

High Efficiency

HAT Pi Model

HAT GPU Model

HAT CPU Model

HAT Mobile Model
Three Steps in HAT

- Train a **SuperTransformer**
- **Evolutionary search** with a hardware latency constraint to find a SubTransformer
- Train the **SubTransformer** from scratch
SuperTransformer Design Space
Breaking Two Conventional Design Rules

Convention

Information bottleneck

Encoder Layer 3
Encoder Layer 2
Encoder Layer 1

Decoder Layer 3
Decoder Layer 2
Decoder Layer 1

HAT

Arbitrary Encode-Decoder Attention

Encoder Layer 3
Encoder Layer 2
Encoder Layer 1

Decoder Layer 3
Decoder Layer 2
Decoder Layer 1

• Each decoder layer can attend to arbitrary encoder layers
SuperTransformer Design Space
Breaking Two Conventional Design Rules

Convention

All layers are identical

Encoder Layer 1
Encoder Layer 2
Encoder Layer 3

Decoder Layer 1
Decoder Layer 2
Decoder Layer 3

HAT

Heterogeneous Layers

Encoder Layer 1
Encoder Layer 2
Encoder Layer 3

Decoder Layer 1
Decoder Layer 2
Decoder Layer 3

- Each layer has different architecture
Elastic SuperTransformer with Weight-Sharing

- Elastic embedding dimension, share the front part of embedding
Elastic SuperTransformer with Weight-Sharing

- Elastic head number in all self-attention and cross-attention, share Q,K,V
- Elastic embedding dimension, share the front part of embedding
Elastic SuperTransformer with Weight-Sharing

- Elastic hidden dimension, share the front part of weights
- Elastic head number in all self-attention and cross-attention, share Q,K,V
- Elastic embedding dimension, share the front part of embedding
Elastic SuperTransformer with Weight-Sharing

- Elastic number of layers, share the front several layers
- Elastic hidden dimension, share the front part of weights
- Elastic head number in all self-attention and cross-attention, share Q,K,V
- Elastic embedding dimension, share the front part of embedding
Elastic SuperTransformer with Weight-Sharing

- Elastic number of layers, share the front several layers
- Elastic hidden dimension, share the front part of weights
- Elastic head number in all self-attention and cross-attention, share Q,K,V
- Elastic embedding dimension, share the front part of embedding
Train SuperTransformer by Uniform Sampling

**Elastic Layer Num in Encoder**
- Encoder Layer $m$
- Encoder Layer 2
- Encoder Layer 1

**Elastic Layer Num in Decoder**
- Decoder Layer $m$
- Decoder Layer 1

**Elastic Hidden Dim in FFN**

**Elastic Head Num**
- (Self Attention)
- (En-De Attention)

**Elastic Embedding Dim**

**Arbitrary Encoder-Decoder Attention**

**Concat**
Train SuperTransformer by Uniform Sampling

Elastic Layer Num in Encoder
- Encoder Layer 4
- Encoder Layer 3
- Encoder Layer 2

Elastic Hidden Dim in FFN
- Encoder Layer 1

Elastic Embedding Dim

Elastic Layer Num in Decoder
- Decoder Layer 3
- Decoder Layer 2

Elastic Hidden Dim in FFN
- Decoder Layer 1

Elastic Head Num (Self Attention)
- Encoder Layer 4
- Encoder Layer 3
- Encoder Layer 2

Elastic Head Num (En-De Attention)
- Decoder Layer 1

Arbitrary Encoder-Decoder Attention
- Encoder Layer 1

Elastic Embedding Dim
Train SuperTransformer by Uniform Sampling

Elastic Layer Num in Encoder

Encoder Layer 3
Encoder Layer 2
Encoder Layer 1

Elastic Hidden Dim in FFN
Elastic Head Num (Self Attention)
Elastic Embedding Dim

Elastic Layer Num in Decoder

Decoder Layer 2
Decoder Layer 1

Elastic Hidden Dim in FFN
Elastic Head Num (En-De Attention)
Elastic Head Num (Self Attention)
Elastic Embedding Dim

Arbitrary Encoder-Decoder Attention
Specialized SubTransformer for Different Hardware
Specialized SubTransformer for Different Hardware
Specialized SubTransformer for Different Hardware
SuperTransformer Provides Accurate Performance Proxy

- SuperTransformer provides fast and accurate proxy of SubTransformer performance
  - The smaller the validation loss of a inherited-weight SubTransformers, the better the final performance trained from-scratch
Three Steps in HAT

- Train a SuperTransformer
- Evolutionary search with a hardware latency constraint to find a SubTransformer
- Finally, train the searched SubTransformer from scratch
Evolutionary Search with Hardware Latency Feedback

- Search for a model with low loss and satisfies latency constraint
Latency Predictor

- Train a latency predictor to provide fast latency feedback
- With a dataset of [SubTransformer architecture, measured latency]
- Accurate: RMSR is 0.1s

Predicted Latency (s) vs. Real Latency on Raspberry Pi ARM CPU (s)

- Predicted latency is very close to real latency.
- \(-\rightarrow y = x\)
Three Steps in HAT

- Train a SuperTransformer
- Evolutionary search with a hardware latency constraint to find a SubTransformer
- Finally, train the searched SubTransformer from scratch
HAT (Ours)

- Layer Number Scaling of Baseline Transformer
- Dimension Scaling of Baseline Transformer

The higher the better
The lower the better

- HAT achieves 3x faster and 3.7x smaller size over baseline Transformer
HAT (Ours)
Layer Number Scaling of Baseline Transformer
Dimension Scaling of Baseline Transformer

Latency (ms)
BLEU Score

2.7× Faster
Transformer-Big
Transformer-Base
Dimension Scaling cannot reduce latency of Baseline Transformer on GPU

• HAT achieves 2.7x speedup
Comparison with Baseline Transformer

- HAT outperforms baseline Transformer on diverse hardware platforms
HAT Searches Specialized Models
On WMT’14 En-De Task

- Searched SubTransformers for ARM CPU and Nvidia GPU, both have 28.1 BLEU
HAT Searches Specialized Models
On WMT’14 En-De Task

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
<th>GPU Latency</th>
<th>ARM CPU Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU-Efficient Model</td>
<td>28.1</td>
<td>147ms</td>
<td>6491ms</td>
</tr>
<tr>
<td>ARM-Efficient Model</td>
<td>28.1</td>
<td>184ms</td>
<td>6042ms</td>
</tr>
</tbody>
</table>

- The efficient model for GPU is not efficient for ARM CPU and vice versa
- **Specialized** model is necessary
Comparison with the Evolved Transformer

WMT'14 En-Fr running on a Raspberry Pi

- For WMT'14 En-Fr running on Raspberry Pi, HAT achieves 0.1 higher BLEU, 2.7x faster, 3.7x smaller model size, 3.2x fewer FLOPs, and 10,148x less search cost.
• For WMT’14 En-De, the search cost of HAT is $12,000\times$ less than the Evolved Transformer.
• HAT is cost-efficient because in evolutionary search, it leverages the performance proxy instead of performance trained to the end.
Comparison with Other Models
On WMT’14 En-De Task

- HAT has lowest latency and highest BLEU
- HAT is orthogonal to new operations

**Diagram:**

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Latency (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>25.9</td>
<td>4.3</td>
</tr>
<tr>
<td>Evolved</td>
<td>25.4</td>
<td>3.7</td>
</tr>
<tr>
<td>Levenshtein</td>
<td>25.2</td>
<td>6.5</td>
</tr>
<tr>
<td>Lite</td>
<td>25.8</td>
<td>3.4</td>
</tr>
<tr>
<td>HAT (Ours)</td>
<td>25.9</td>
<td>3.4</td>
</tr>
</tbody>
</table>
Further Compress HAT by 25x
On WMT’14 En-Fr Task

- HAT is orthogonal to general model compression techniques
• Code and 50 pre-trained models are released
• Latency, BLEU, model size, FLOPs are provided
• Push-the-button to run models

https://github.com/mit-han-lab/hardware-aware-transformers
HAT: Hardware-Aware Transformers

Pushing the frontier of Green AI and Tiny AI

- A specialized transformer for each hardware
- Arbitrary encoder-decoder attention and heterogeneous layers improve performance
- 3x speedup, 3.7x smaller size, 12,000x less cost over baselines

Live Q&A:

Wed. July 8, 13:00 UTC+0 13B Machine Translation-15
Wed. July 8, 21:00 UTC+0 15B Machine Translation-18

Paper ID: 148