Searching for Efficient Transformers using Neural Architectural Search

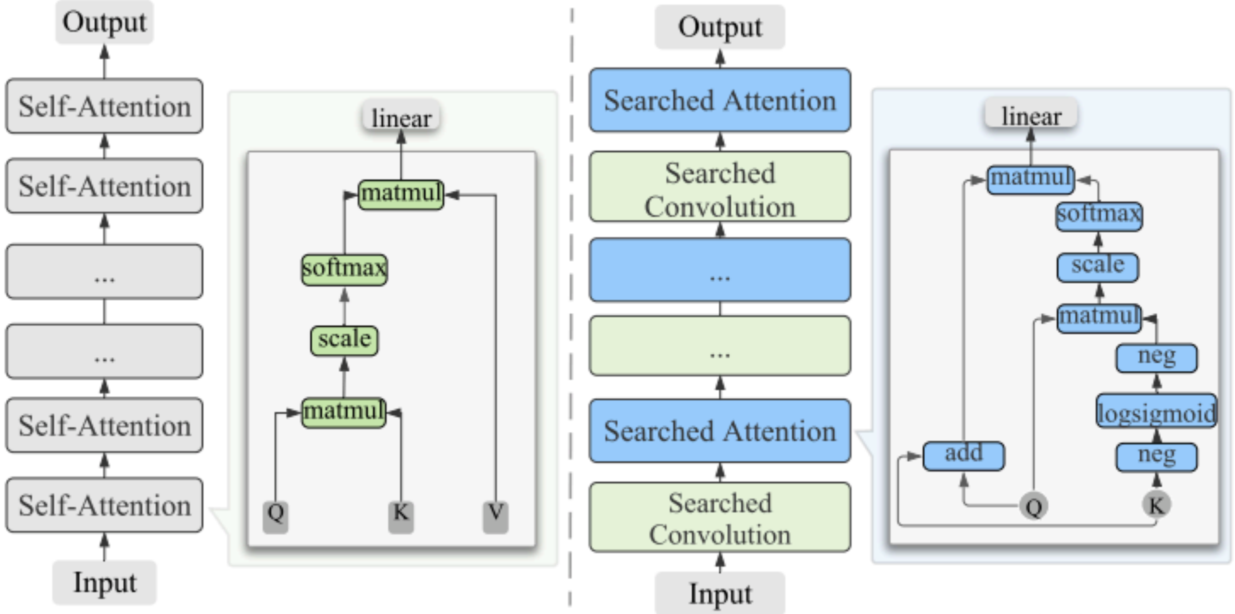
May 4, 2022

Menu

- AutoBERT-Zero: Evolving BERT Backbone from Scratch [Gao et al., AAAI’22]
- Primer: Searching for Efficient Transformers for Language Modeling [So et al., NeurIPS’21]
- LiteTransformerSearch: Training-free On-device Search for Efficient Autoregressive Language Models [Javaheripi et al., arXiv’22]
- AutoDistil: Few-shot Task-agnostic Neural Architecture Search for Distilling Large Language Models [Xu et al., arXiv’22]

AutoBERT-Zero: Evolving BERT Backbone from Scratch [Gao et al., AAAI'22]

- Does there exist more powerful and efficient attention beyond the pure Q-K-V self-attention?
- Can we boost the model performance and efficiency by flexibly combining global attention with local operations?



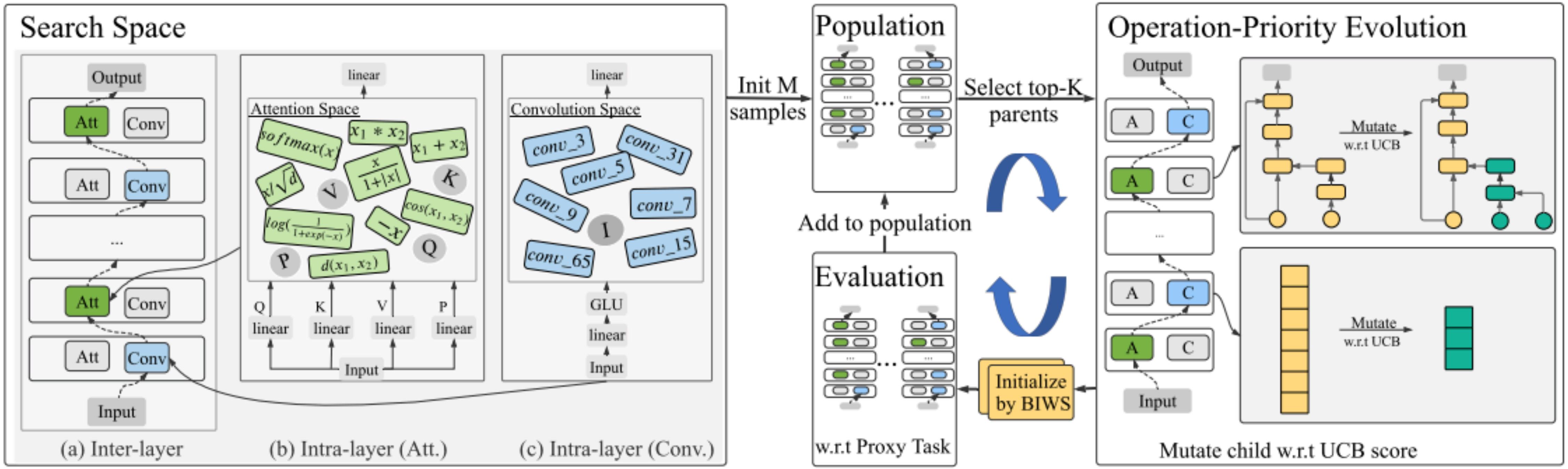
(a) BERT (b) Self-Attention (c) Searched backbone (d) Searched Attention

$$\begin{aligned} Attn(X) &= \sigma(XW_Q(XW_K)^\top/\sqrt{d_h})XW_V W_O^\top \\ &= \sigma(QK^\top/\sqrt{d_h})VW_O^\top, \end{aligned}$$

$$\hat{Attn}(X)_{L_2} = \sigma(Q\log(1+\exp(K^\top))/\sqrt{d_h})(K+Q)W_O^\top.$$

$$\hat{Attn}(X)_{L_{12}} = \sigma(Q(K/\sqrt{d_h}+V)^\top/\sqrt{d_h})VW_O^\top.$$

AutoBERT-Zero: Evolving BERT Backbone from Scratch [Gao et al., AAAI'22]

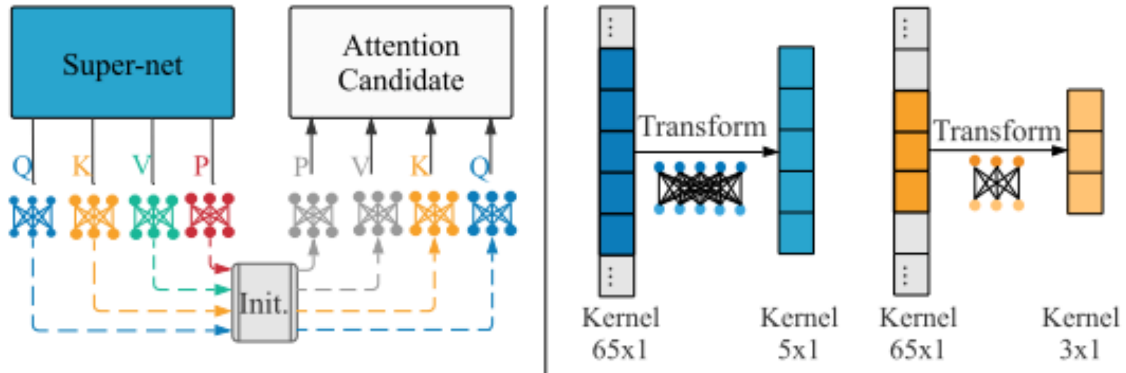


AutoBERT-Zero: Evolving BERT Backbone from Scratch [Gao et al., AAAI'22]

Algorithm 1: OP-NAS Algorithm.

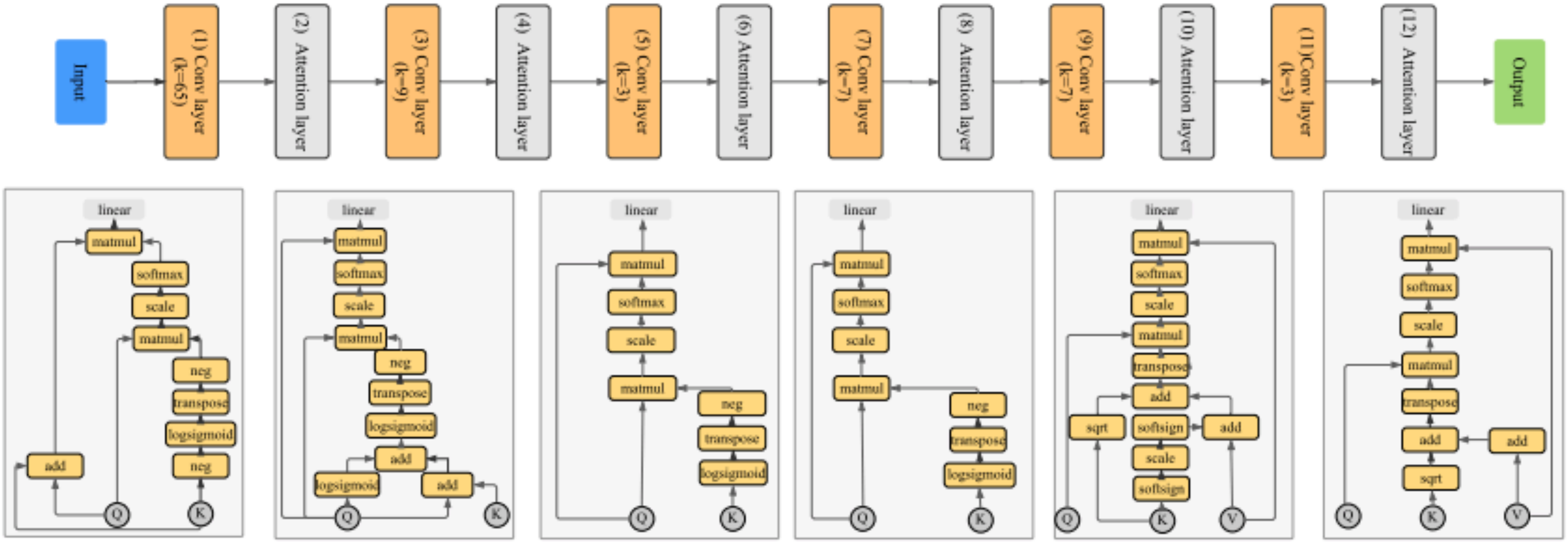
Initialize population $\mathcal{M}$ from search space $\mathcal{A}$;
Model evaluation in $\mathcal{M}$;
repeat
  $\mathcal{P} \leftarrow$ Top-K ($\mathcal{M}$);
  for each parent p in $\mathcal{P}$ **do**
    $p' \leftarrow Mutation_{InterLayer}(p)$;
    $c \leftarrow Mutation_{IntraLayer}(p', \text{UCB})$;
    Initialize c with BIWS strategy ;
    Evaluate c on the proxy task;
  end for
  Update $\mathcal{M}$ with the newly evaluated children.
  Update UCB scores by Equation (3);
until convergence

$$u_i = \mu_i + \alpha\sqrt{2\log N/N_i}$$



(a) Attention layer weight-sharing (b) Convolution layer weight-sharing

AutoBERT-Zero: Evolving BERT Backbone from Scratch [Gao et al., AAAI'22]



Attention in 2nd layer Attention in 4th layer Attention in 6th layer Attention in 8th layer Attention in 10th layer Attention in 12th layer

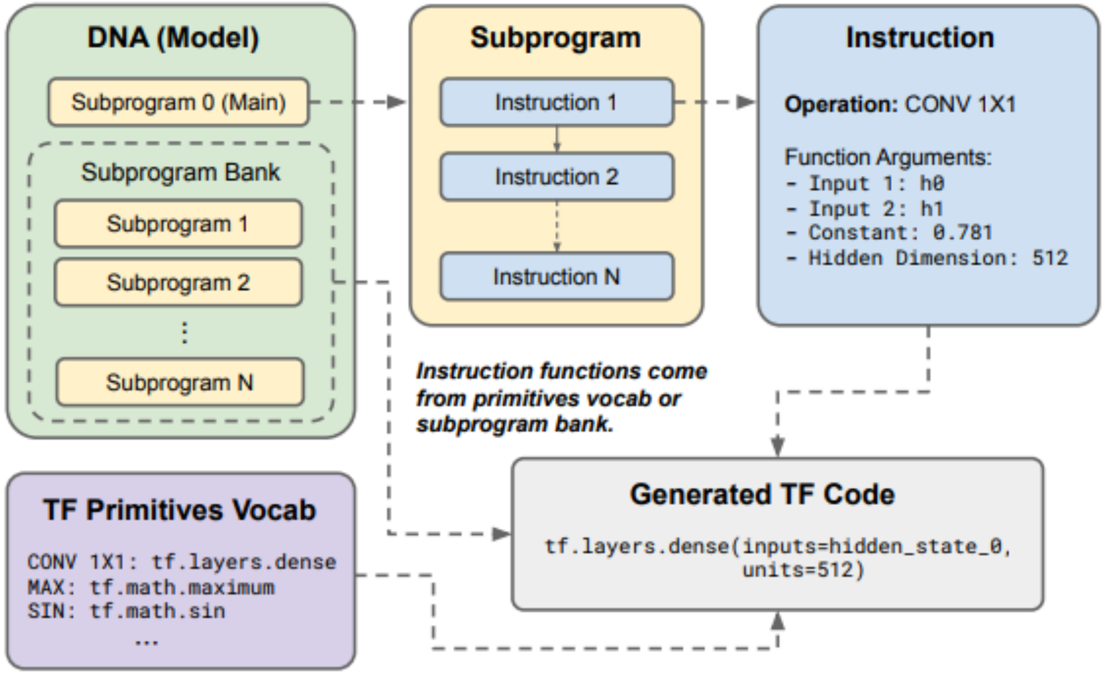
$$\hat{Attn}(X)_{L_2} = \sigma(Q \log(1 + \exp(K^\top))/\sqrt{d_h})(K + Q)W_O^\top.$$

$$\hat{Attn}(X)_{L_{12}} = \sigma(Q(K/\sqrt{d_h} + V)^\top/\sqrt{d_h})VW_O^\top.$$

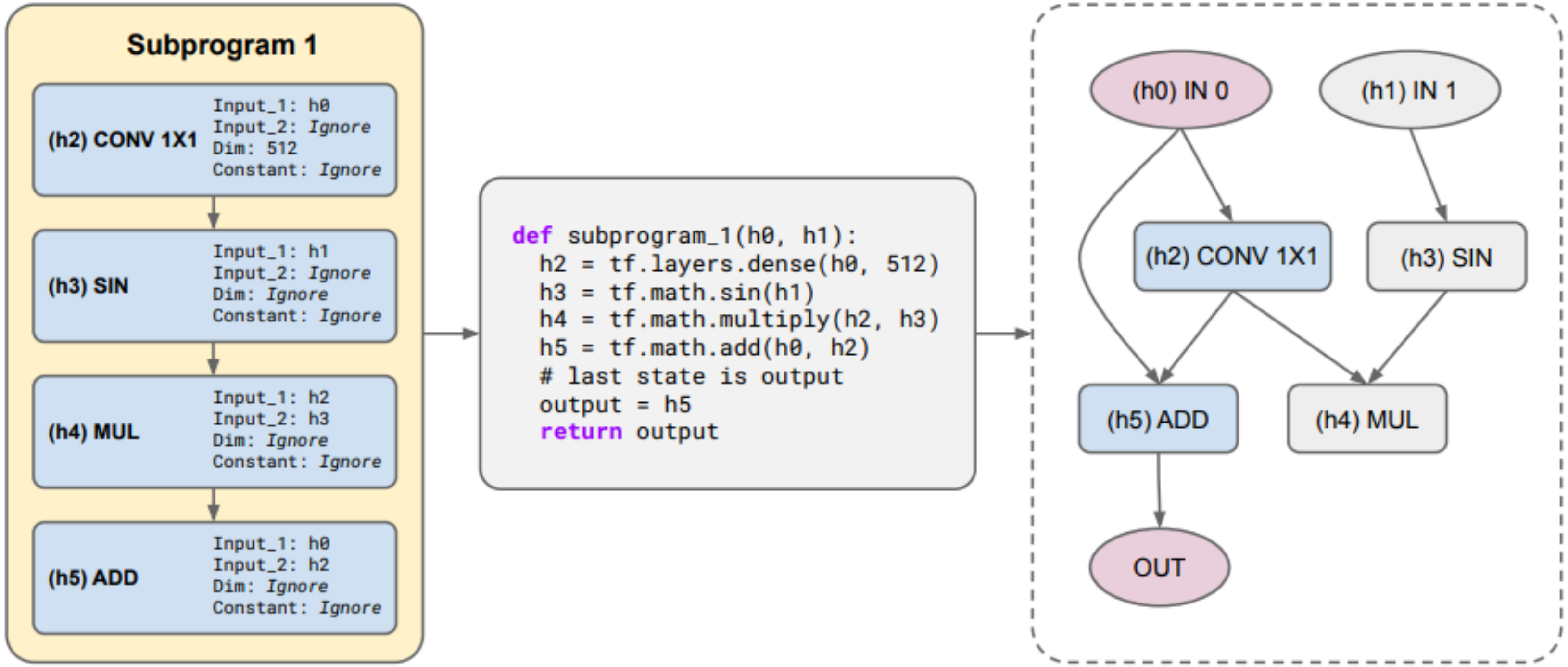
AutoBERT-Zero: Evolving BERT Backbone from Scratch [Gao et al., AAAI'22]

	#Params	Infer FLOPs	CoLA	MRPC	MNLI-(m/mm)	STS-B	RTE	QQP	QNLI	SST-2	AVG
Development Set											
BERT-base(ours)	110M	2.9e10	58.1	89.7	84.8/85.2	88.8	69.0	88.2	91.5	92.9	83.1
AutoBERT-att	104M	2.3e10	65.4	92.2	84.6/85.0	90.4	81.6	88.5	91.8	93.8	85.9
AutoBERT-conv	104M	2.2e10	63.8	92.6	84.4/84.6	90.1	80.5	88.3	91.7	93.5	85.5
AutoBERT-w/o-desc	104M	2.3e10	65.1	92.8	84.5/85.0	90.5	78.7	88.2	91.6	93.7	85.6
AutoBERT-Zero	104M	2.3e10	64.5	93.3	85.5/85.3	90.8	81.9	88.9	92.0	94.2	86.3
AutoBERT-Zero*	104M	2.3e10	**67.3**	**93.8**	**86.4/86.3**	**90.8**	**85.2**	**91.7**	**92.5**	**95.2**	**87.7**
Test Set											
GPT(Radford et al. 2018)	117M	3.0e10	45.4	82.3	82.1/81.4	82.0	56.0	70.3	88.1	91.3	75.4
BERT-base(Devlin et al. 2019)	110M	2.9e10	52.1	88.9	84.6/83.4	85.8	66.4	71.2	90.5	93.5	79.6
DynaBERT-base(Hou et al. 2020)	110M	2.9e10	54.9	87.9	84.5/84.1	84.4	69.9	72.1	91.3	93.0	80.2
ConvBERT-base (Jiang et al. 2020)	106M	2.7e10	53.7	89.3	84.6/83.6	86.1	72.1	71.3	90.1	93.5	80.5
Roberta-base (Liu et al. 2019b)	110M	2.9e10	50.5	90.0	86.0/85.4	88.1	73.0	70.9	92.5	94.6	81.1
BERT-Large(Devlin et al. 2019)	340M	8.7e10	60.5	89.3	86.7/89.5	86.5	70.1	72.1	92.7	94.9	82.1
AutoBERT-Zero	104M	2.3e10	55.9	89.5	85.4/84.9	88.3	77.8	71.8	91.2	94.6	82.2
AutoBERT-Zero*	104M	2.3e10	**59.5**	**90.5**	**86.1/86.0**	**88.9**	**80.2**	**72.8**	**92.1**	**95.1**	**83.5**
AutoBERT-Zero-Large	318M	6.8e10	**63.8**	**90.7**	**87.7/87.1**	**90.1**	**80.4**	72.1	**93.6**	**95.4**	**84.5**

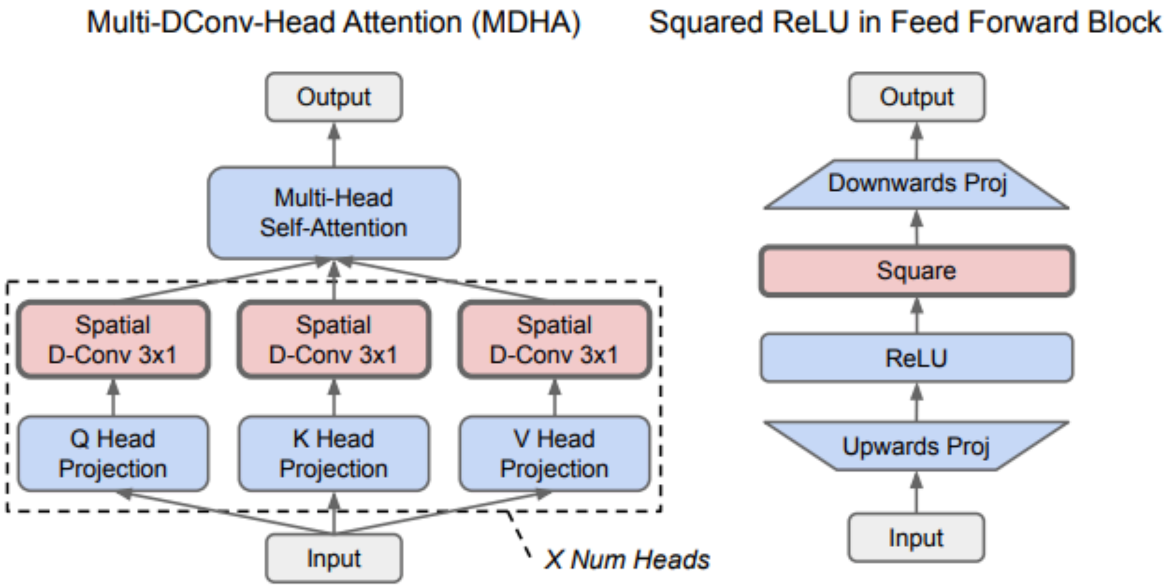
Primer: Searching for Efficient Transformers for Language Modeling [So et al., NeurIPS'21]



Primer: Searching for Efficient Transformers for Language Modeling [So et al., NeurIPS'21]

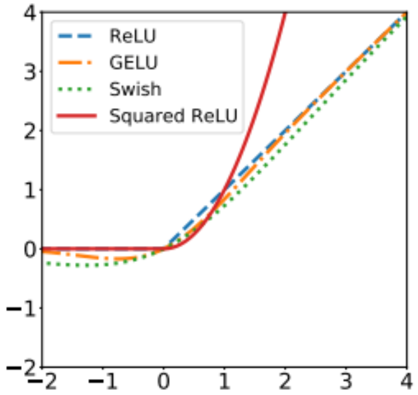


Primer: Searching for Efficient Transformers for Language Modeling [So et al., NeurIPS'21]

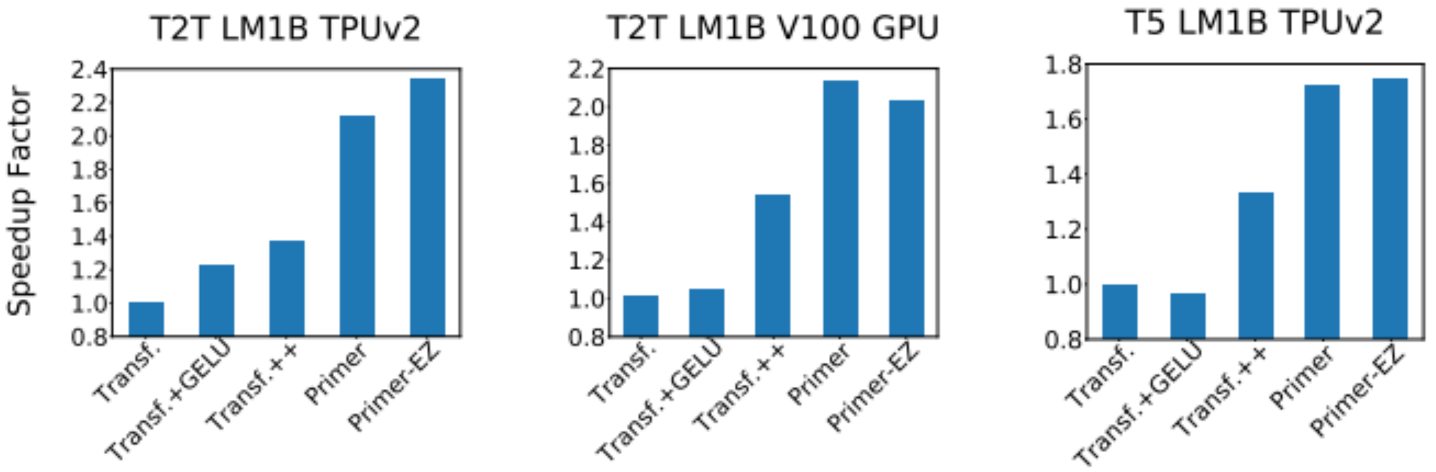


MDHA Projection Pseudo-code

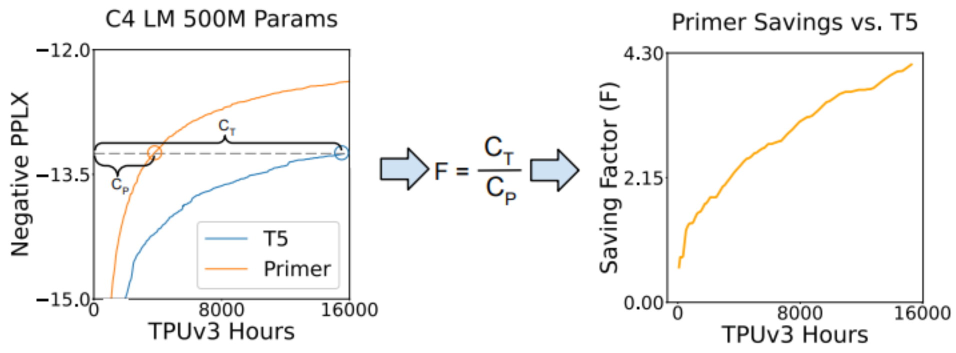
```
# Use to create each K, Q,
     and V head of size 'hs'.
def mdha_projection(x, hs):
  # Create head.
  x = proj(x,
           head_size=hs,
           axis="channel")

  # Apply D-Conv to head.
  x = d_conv(x,
             width=3,
             head_size=hs,
             axis="spatial",
             mask="causal")
  return x
```



Primer: Searching for Efficient Transformers for Language Modeling [So et al., NeurIPS'21]



Model	Steps	TPUv3 Hours	PPLX
Original T5	1M	15.7K	13.25
T5++	251K	4.6K	13.25
Primer	207K	**3.8K**	13.25
T5++	1M	16.5K	12.69
Primer	480K	**8.3K**	12.69
Primer	1M	17.3K	12.35



LiteTransformerSearch: Training-free On-device Search for Efficient Autoregressive Language Models [Javaheripi et al., arXiv'22]

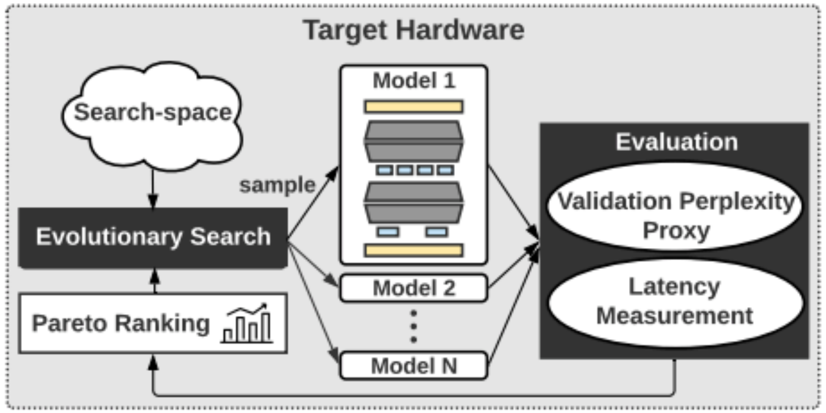
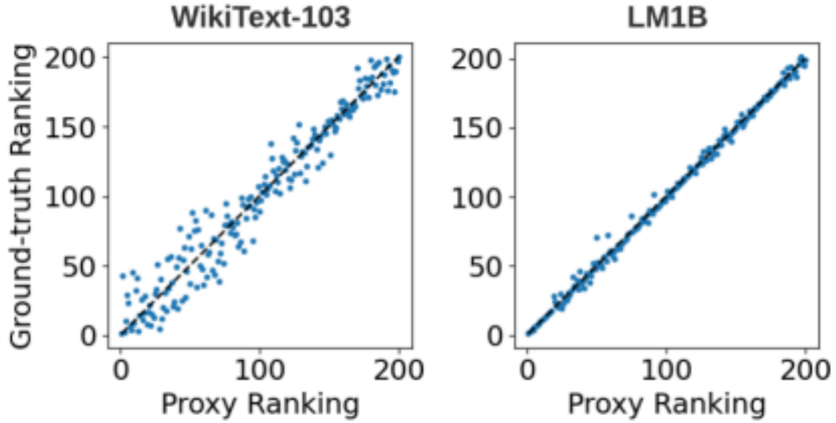


Figure 1: High-level overview of LTS. We propose a zero-cost proxy for evaluating the validation perplexity of candidate architecture candidates. Our search is powered by evolutionary algorithms which use the proposed proxy along with real latency measurements on the target hardware to evaluate sampled architectures.



Algorithm 1: LTS's training-free NAS

Input: Search space $\mathcal{D}$, n_{iter}

Output: Perplexity-latency pareto-frontier $\mathbb{F}$

$\mathcal{L}, \mathcal{P}, \mathbb{F} \leftarrow \emptyset, \emptyset, \emptyset$

while $N \leq n_{iter}$ **do**

  $\mathbb{F}' \leftarrow$ Subsample($\mathbb{F}$)

  $\mathbb{S}_N \leftarrow EA(\mathbb{F}', \mathcal{D})$

  `// measure latency`

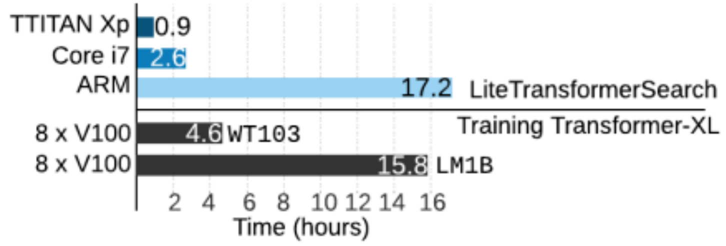
  $\mathcal{L} \leftarrow \mathcal{L} \bigcup$ Latency($\mathbb{S}_N$)

  `// estimate perplexity`

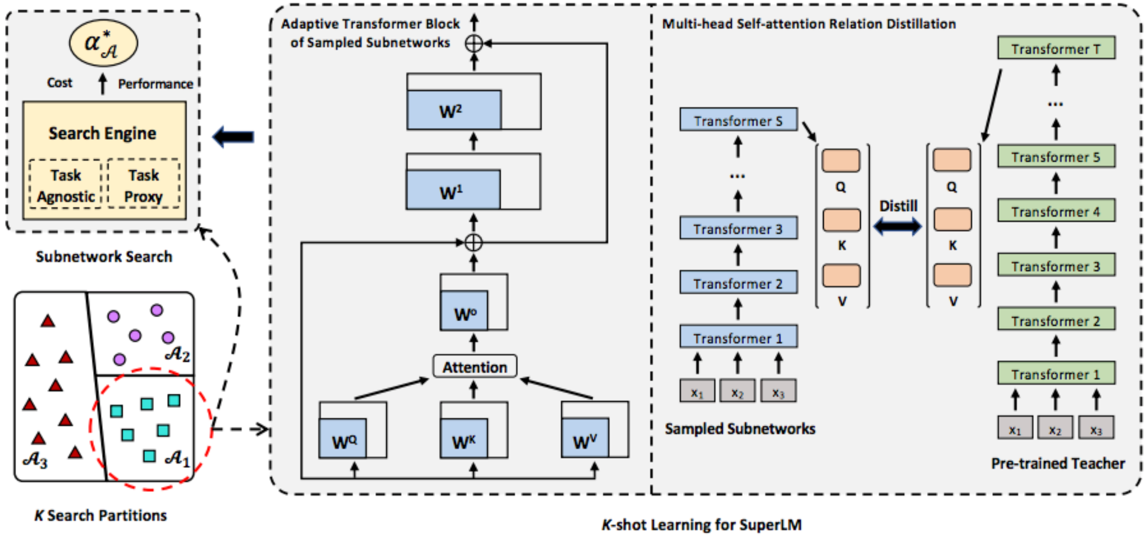
  $\mathcal{P} \leftarrow \mathcal{P} \bigcup$ Proxy($\mathbb{S}_N$)

  `// update the pareto front`

  $\mathbb{F} \leftarrow$ LowerConvexHull($\mathcal{P}, \mathcal{L}$)



AutoDistil: Few-shot Task-agnostic Neural Architecture Search for Distilling Large Language Models [Xu et al., arXiv'22]



Algorithm 1 Few-shot Task-agnostic Knowledge Distillation with `AutoDistil` .

Input: Partitioned K sub-spaces $\mathcal{A}_k$; initialized K SuperLMs S_k on $\mathcal{A}_k$; pre-trained teacher model T; unlabeled data D; training epochs E; sampling steps M
Output: Trained SuperLMs $\{S_k\}$
for $k = 1$ **to** K **do**
  for $i = 1$ **to** E **do**
    Get a batch of data from D
    for *batch* in D **do**
      Clear gradients in SuperLM S_k
      for $m = 1$ **to** M **do**
        Randomly sample a subnetwork s from S_k
        Calculate self-attention distil. loss between subnetwork s and teacher T with Eqn. (6)
        Accumulate gradients
      end for
      Update S_k with the accumulated gradients
    end for
  end for
end for

	SuperLM$_{\text{Tiny}}$	SuperLM$_{\text{Small}}$	SuperLM$_{\text{Base}}$	BERT
#Subnets	256	256	256	N/A
#Layers	(4, 7, 1)	(9, 12, 1)	(9, 12, 1)	12
#Hid_dim	(128, 224, 32)	(256, 352, 32)	(544, 640, 32)	768
MLP Ratio	(2.0, 3.5, 0.5)	(2.5, 4.0, 0.5)	(2.5, 4.0, 0.5)	4.0
#Heads	(7, 10, 1)	(7, 10, 1)	(9, 12, 1)	12
#FLOPs	40-367M	0.5-2.1G	2.1-7.9G	11.2G
#Params	4-10M	12-28M	39-79M	109M

Model	`AutoDistil`$_{\text{Agnostic}}$		
	ΔFLOPs	ΔPara	ΔAvg.
BERT$_{\text{BASE}}$ Devlin et al. [2019] (teacher)	81.1%	75.5%	-2.6
BERT$_{\text{SMALL}}$ Turc et al. [2019]	62.4%	59.7%	-0.3
Truncated BERT Williams et al. [2018]	62.4%	59.7%	+2.5
DistilBERTSanh et al. [2019]	62.4%	59.7%	+1.1
TinyBERT Jiao et al. [2020]	62.4%	59.7%	-0.3
MINILM Williams et al. [2018]	62.4%	59.7%	-1.4