MERINO: ENTROPY-DRIVEN DESIGN FOR MOBILE-FRIENDLY GENERATIVE LANGUAGE MODELS - APPENDIX

Anonymous authors
Paper under double-blind review

APPENDIX

In the appendix, we provide preliminary knowledge of autoregressive transformers (Appendix A), detailed one-shot learning results (Appendix B), design details of MeRino (Appendix C), including search space configurations of our entropy-driven design, structural details of MeRino, and detailed Evolutionary Algorithm (EA) and Mutation algorithm, and limitations (Appendix D).

A PRELIMINARIES

Autoregressive Transformers Decoder-only, or autoregressive transformers, operate by predicting the next element in a sequence based on the preceding elements. A standard autoregressive transformer comprises an embedding layer to project sequences of tokens to hidden dimensions and stacks of transformer layers to capture long-term dependencies between input tokens using the self-attention mechanism. A transformer layer includes two main components: a multi-head attention (MHA) module and a position-wise feed-forward network (FFN). The MHA module facilitates capturing contextual information by attending to different positions within the input sequence, while the FFN performs element-wise transformations to introduce non-linearity and improve representational capacity.

Multi-Head Attention (MHA) Multi-head attention (MHA) is a crucial component within the transformer architecture that enables the model to selectively attend to different segments of the input sequence. This mechanism involves projecting the input sequence into multiple attention heads, each of which calculates an independent attention distribution. In MHA computation, there are specifically four main matrices involved: attention matrices $W^Q, W^K, W^V \in \mathbb{R}^{d_{in} \times d_{in}/h}$ and output project matrix $W^O \in \mathbb{R}^{d_{in} \times d_{out}}$. Given the output of previous layers $X \in \mathbb{R}^{n \times d_{in}}$ as input, the attention function is formulated as:

$$Q, K, V = XW^Q, XW^K, XW^V \tag{1}$$

$$\text{Attn}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_{in}/h}})(V) \tag{2}$$

where Q, K, and V represent queries, keys, and values, respectively.

MHA is defined by concatenating h attention heads and producing outputs as follows:

$$\text{MHA}(X) = \text{Concat}(\text{Attn}_i, ..., \text{Attn}_h)W^O \tag{3}$$

In addition, the transformer layer adopts residual connection and layer normalization on top of MHA to compute the final outputs.

$$X^{\text{MHA}} = \text{LayerNorm}(X + \text{MHA}(X)) \tag{4}$$

Position-wise Feed-forward Network (FFN) In addition to the MHA, each transformer layer includes a feed-forward network (FFN). The FFN applies two point-wise fully connected layers followed by a non-linear activation function, such as ReLU. Operations within FFN can be formulated as follows:

$$X^{\text{FFN}} = \text{ReLU}(X^{\text{MHA}}W^{\text{FFN}_1} + b_1)W^{\text{FFN}_2} + b_2 \tag{5}$$

Similarly, the FFN also incorporates residual connections and layer normalization to compute the final outputs:

$$X^{\text{FFN}} = \text{LayerNorm}(X^{\text{MHA}} + X^{\text{FFN}}) \tag{6}$$

B One-Shot Learning Results

We report additional one-shot comparison results in Table 1. We can see that our designed models still achieve competitive performance against state-of-the-art LLMs with reduced parameters and computation.

Table 1: Detailed one-shot downstream task results for MeRino and publicly available pretrained LLMs.

	MeRino			OPT		Pythia		Cerebras-GPT	GPT-2
Params (↓)	52 M	61 M	64 M	125 M	350 M	70 M	162 M	111 M	124 M
FLOPs (↓)	60 G	110 G	160 G	210 G	720 G	100 G	270 G	260 G	290 G
HellaSwag	0.262	0.260	0.270	0.264	0.279	0.266	0.296	0.265	**0.308**
WinoGrande	0.517	0.486	0.495	0.504	0.519	**0.522**	0.506	0.494	0.500
ARC-Easy	0.339	0.351	0.353	0.396	**0.413**	0.344	0.387	0.348	0.399
ARC-Challenge	0.214	0.208	0.237	0.229	0.238	0.208	0.225	0.218	0.235
OpenBookQA	0.234	0.240	0.262	0.232	0.258	0.238	0.266	0.262	**0.266**
BoolQ	0.536	0.539	0.570	0.547	0.583	0.521	0.560	**0.605**	0.526
WIC	0.467	0.489	0.472	0.483	**0.506**	0.464	0.467	0.475	0.464
CB	0.411	0.482	**0.482**	0.464	0.429	0.464	0.482	0.464	0.482
WSC	**0.423**	0.413	0.365	0.365	0.365	0.365	0.365	0.365	0.365
RTE	**0.574**	0.542	0.549	0.484	0.523	0.538	0.520	0.552	0.549
PubmedQA	0.404	0.466	0.513	0.444	0.462	0.478	0.521	0.463	0.425
LogiQA	0.264	0.256	0.269	0.246	0.252	**0.284**	0.258	0.255	0.250
Average	0.387	0.394	0.403	0.388	0.402	0.391	**0.404**	0.397	0.397

C Design Details of MeRino

C.1 Search Space

Table 2 presents details of the search space defined for our entropy-driven design method. In addition, we set the embedding projection dimension as 768 and the maximum position embedding dimension as 2048. Our search space encapsulates over 216k different autoregressive transformer architectures.

Table 2: Search space hyperparameters for MeRino.

Embedding Dimension - E_i	[64, 128, 256, 384, 512, 640, 768, 896, 1024]
FFN Ratio - R_i	[1, 1.5, 2, 2.5, 3, 3.5, 4]
Number of Layers Per Block - L_i	[1, 2, 3, 4]

C.2 Evolutionary Algorithm

We give a detailed description of the Evolutionary Algorithm (EA) and Mutation algorithm in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Evolutionary Algorithm

Require: Search space $\mathcal{D}$, number of iterations T, computation budget constraint $\mathcal{C}$, population size M, parent size K
Ensure: Optimal architecture $\mathcal{A}^*$
  Initialize population $\mathcal{P}$
  while $i \leq T$ **do**
    while $len(\mathcal{P}) < M$ **do**
      Random select $\mathcal{A}_i \in \mathcal{P}$ as parent.
      Mutate $\hat{\mathcal{A}}_i = \textbf{MUTATE}(\mathcal{A}_i, \mathcal{D})$
      if ComputeCost$(\hat{\mathcal{A}}_i) \leq \mathcal{C}$ **then**
        Calculate entropy $\mathcal{Z} = H(\hat{\mathcal{A}}_i)$
        Add $\hat{\mathcal{A}}_i$ to $\mathcal{P}$
      else
        Do nothing
      end if
    end while
    Remove $(M - K)$ networks with smallest entropy scores
  end while
  Return $\mathcal{A}^*$, the architecture with highest entropy in $\mathcal{P}$

Algorithm 2 MUTATE

Require: Search space $\mathcal{D}$, architecture $\mathcal{A}_i$.
Ensure: Mutated architecture $\hat{\mathcal{A}}_i$
  Randomly select a block in $\mathcal{A}_i$
  Randomly alternate block depth, embedding dimension, and FFN ratio within a certain range
  Return the mutated architecture $\hat{\mathcal{A}}_i$

C.3 Detail Structure of MeRino

The searched network structures of MeRino are listed in Tables 3. We use four blocks for our entropy-driven design. E_i denotes the embedding dimension for each transformer block, R_i denotes the FFN ratio, and L_i denotes the number of layers (depth) of each transformer block.

Table 3: Structure Configuration of MeRino.

Model	E_i	R_i	L_i	Params	FLOPs
MeRino	[512, 512, 640, 896]	[1, 1, 1, 1]	[2, 3, 2, 1]	52 M	60 G
	[640, 768, 896, 1024]	[1, 1.5, 1, 1]	[2, 2, 2, 2]	61 M	110 G
	[640, 896, 1024, 1024]	[1.5, 1.5, 1, 1]	[3, 3, 2, 3]	64 M	160 G

D Limitations

As no research is perfect, MeRino has several limitations as well. First, the design of MeRino explores entropy only from parameter subspace due to its straightforwardness. Further exploration of entropy in the feature space could provide a better theoretical understanding of transformer architecture and potentially lead to improved model designs. Second, our design only focuses on the "macro-structure" of the LLMs (channels/depths/heads). Other key components, such as residual connections, layer normalization, and nonlinear activations, are also essential to achieve good performance. However, the theoretical foundation for these components is not well-studied, especially from an information theory perspective. How to integrate these components in our entropy-based framework remains an open question and we would leav it for our future research.