LPZero: Language Model Zero-cost Proxy Search from Zero

Anonymous ACL submission

Abstract

Neural Architecture Search (NAS) facilitates the automatic effective neural network designs while requiring the substantial computational resource particularly for language models. Zero-shot NAS exploits Zero-cost (ZC) proxies to estimate model performance, thereby markedly reducing computational demands. However, existing ZC proxies rely heavily on in-depth expert knowledge and repetitive trial-and-error costs. Moreover, most of existing ZC proxies fail to surpass the performance of the naive baseline (number of parameters). To address these challenges, we introduce a novel framework called **LPZero** (**L**anguage model zero-cost **P**roxy search from **Zero**). It is designed to automate the design of efficient ZC proxies for language models, and achieve the higher ranking consistency. Specifically, we initially consolidate existing ZC proxy designs into a unified framework as the search space, and then apply an evolutionary algorithm to heuristically identify new, promising proxy candidates for language models. To enhance the efficiency of the search process, we introduce a Predictive-Pruning Strategy (PPS). This strategy is designed to preemptively eliminate unpromising proxies, thereby mitigating the risk of proxy degradation. Extensive experiments on the FlexiBERT and GPT-2 search space demonstrate the effectiveness of our algorithm. Notably, the consistency in performance ranking achieved by our method significantly surpasses that observed with current proxies.

1 Introduction

Traditional neural network design, heavily dependent on expert knowledge and experience (Krizhevsky et al., 2017; He et al., 2016), is both time-intensive and prone to trial-and-error. Neural Architecture Search (NAS) emerged to automate and refine this process by identifying optimal architectures from a set of possibilities using various strategies. However, early NAS methods (Zoph and Le, 2017; Real et al., 2019) require extensive computation. For instance, NASNet (Zoph and Le, 2017) require 500 GPUs for four days. This substantially limits their accessibility and widespread use.

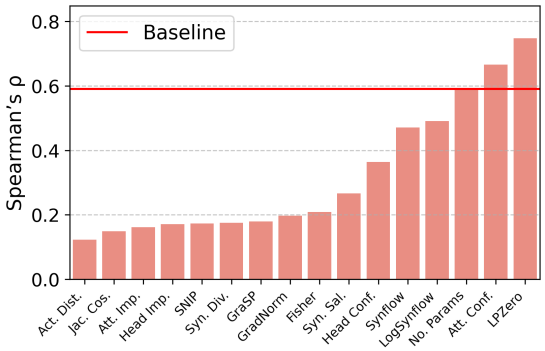

Figure 1: Comparison of zero-cost proxies on FlexiBERT (Serianni and Kalita, 2023) using Spearman correlation coefficients (higher values indicate better performance). The red line represents the baseline method, defined by the number of parameters (Abdelfattah et al., 2021).

To alleviate this issue, recent advancements in Zero-shot NAS (Lin et al., 2021; Li et al., 2023; Mellor et al., 2021; Abdelfattah et al., 2021; Ying et al., 2019; Krishnakumar et al., 2022; Zhou et al., 2022) aim to significantly reduce training costs by employing Zero-cost (ZC) proxies, which circumvent the traditional training process and decrease computational demands. Zero-shot NAS predicts the performance of neural network architectures without the need for actual training, using models that are randomly initialized. This approach enables a rapid and efficient estimation of architecture performance, eliminating the time and resources typically consumed in training processes. To evaluate the effectiveness of ZC proxies, Spearman's ρ or Kendall's τ are utilized to measure the congruence between the performance rankings predicted by ZC proxies and ground-truth derived from fully

Proxy Name	Formula
Activation Distance	$\mathcal{S} = \log\lvert K_H \rvert$
Synaptic Saliency	$\mathcal{S} = \frac{\partial \mathcal{L}}{\partial W} \odot W$
Jacobian Cosine	$S = \left[J_n J_n^t - I \right]^{\frac{1}{20}}$
Synaptic Diversity	$S = \left\lVert \frac{\partial \mathcal{L}}{\partial W} \right\rVert \odot \lVert W \rVert_{\text{nuc}}$
Attention Confidence	$\mathcal{S} = \max(\text{Att}(h, (x_n)))$
Softmax Confidence	$\mathcal{S} = \max(\text{Sft}(h, (x_n)))$
Attention Importance	$\mathcal{S} = \left\lvert \partial \text{Att}(I) \frac{\partial \mathcal{L}(I)}{\partial \text{Att}(I)} \right\rvert$
SNIP	$\mathcal{S} = \left\lvert \frac{\partial \mathcal{L}}{\partial W} \odot W \right\rvert$
GraSP	$\mathcal{S} = -\left(H \frac{\partial \mathcal{L}}{\partial W} \right) \odot W$
Fisher	$\mathcal{S} = \frac{\partial \mathcal{L}}{\partial A} \times A$
LogSynflow	$\mathcal{S} = W \cdot \left\lvert \log \left\lvert \frac{\partial \mathcal{L}}{\partial W} \right\rvert \right\rvert$
Synflow	$\mathcal{S} = \frac{\partial \mathcal{L}}{\partial \mathcal{W}} \odot W$
GradNorm	$\mathcal{S} = \lvert\lvert \frac{\partial \mathcal{L}}{\partial \mathcal{W}} \rvert\rvert_F$

Table 1: Overview of mainstream handcrafted Zero-cost proxies for Transformers, notating K_H as the Kernel Matrix, J as the Jacobian w.r.t. Mini-Batch Input I, Att as attention head, Sft as softmax output, A as activation, and H as the Hessian matrix.

trained models. A high ranking correlation indicates the reliability of ZC proxies in forecasting the potential success of architectures. However, existing Zero-cost (ZC) proxies (Serianni and Kalita, 2023; Javaheripi et al., 2022) are heavily dependent on in-depth expert knowledge and a repetitive trial-and-error, which can be both time-intensive and demanding in terms of effort. For instance, Attention Confidence (Serianni and Kalita, 2023) utilizes normalization techniques to refine attention mechanisms for enhanced performance. Meanwhile, pruning-based proxies such as SNIP (Lee et al., 2019), Fisher (Turner et al., 2020), GraSP (Wang et al., 2020), GradNorm (Abdelfattah et al., 2021) and Synflow (Tanaka et al., 2020) involve complex combination of mathematical operations that critically influence their ranking capabilities. Notably, LogSynflow (Cavagnero et al., 2023) implements logarithmic operations to address gradient explosion issues inherent in Synflow. Furthermore, we observe that most of proxies cannot surpass the baseline performance, measured by the number of parameters, as illustrated in Figure 1. This limitation raises a fundamental but critical question: ***How to devise new proxies efficiently and automatically for language models?***

To answer this question, we break it down to two steps: (1) **Devise a unified search space for existing ZC proxies.** (2) **Employ evolutionary algorithm for discover new proxies.**

For the **first step**, we revisit the existing ZC proxies, as detailed in Table 1, and design a comprehensive search space that encompasses current ZC proxies. Specifically, these proxies are categorized into six types based on the input type: Activation (*A*), Jacobs (*J*), Gradients (*G*), Head (*H*), Weight (*W*) and Softmax (*S*), illustrated in Figure 2. Within this unified framework, we select two types of inputs, denoted as θ, from these categories. Each input undergoes transformation through n unary operations $f(\cdot)$, and the results are combined using a binary operation $g(\cdot)$. This process generates a candidate proxy, $\varphi(f, g, \theta)$, for our search space. More details can be found in Appendix A.

For the **second step**, we propose a novel **LPZero** framework, denoting **L**anguage model **P**roxy Search from **Zero**. As illustrated in Figure 3, we initially select p candidate proxies to establish the population and assess their ranking consistency within the FlexiBERT search space. Through tournament selection, we identify two promising parent proxies ($\varphi^{n,m}$). Subsequently, we perform crossover and mutation operations to generate the offspring proxy φ^q. To evaluate its ranking consistency Spearman ρ^q, we employ this proxy to score each architecture Ω_i with $\varphi^q(\Omega_i)$ and compare the results with their respective ground truth gt_i (e.g., average accuracy). Given the sparsity of the search space, we advocate for a Predictive-Pruning Strategy (PPS) aimed at eliminating ineffective proxies, thereby enhancing the search efficiency. Our main **contributions** are:

- We design a comprehensive and high-quality search space that encompasses most of existing ZC proxies tailored for language models.
- We introduce the Language Model Proxy Search from Zero (LPZero) framework, incorporating a Predictive-Pruning Strategy (PPS) to prevent proxy degradation and thereby improve search efficiency.
- Experiments on FlexiBERT and GPT-2 substantiate the superiority of the proxies identified by our LPZero, indicating the effectiveness of our proposed approach.

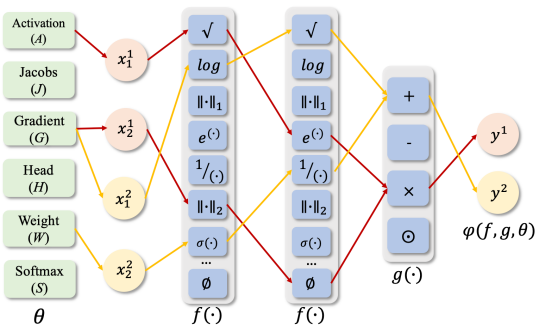

Figure 2: Search space of our LPZero framework.

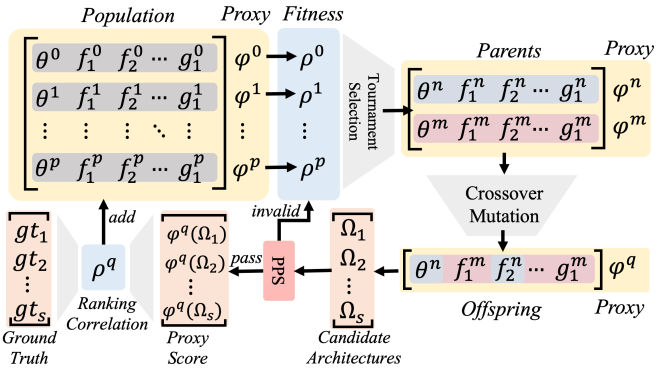

Figure 3: Overview of our LPZero framework.

2 Related Work

Zero-shot NAS In recent years, Zero-shot NAS has gained prominence as a cost-effective strategy for evaluating the accuracy of candidate neural network architectures during the initialization phase, without the need for extensive training. This approach offers a more computation-efficient alternative to traditional One-shot NAS methods. The cornerstone of Zero-shot NAS is its accuracy ranking proxy, which critically determines its effectiveness. While the majority of existing proxies have been developed for computer vision (CV) tasks, there has been relatively limited exploration in the context of natural language processing (NLP) tasks.

NWOT (Mellor et al., 2021) leverages the local Jacobian values across various images to construct an indicator for model ranking based on the correlation of input Jacobians. Similarly, ZenNAS (Lin et al., 2021) assesses candidate architectures by employing the gradient norm of input images as a ranking criterion. Furthermore, Zero-cost NAS (Abdelfattah et al., 2021) draws inspiration from the Optimal Brain Damage principle (LeCun et al., 1989), introducing pruning-based metrics as zero-cost proxies. This includes a variety of indicators such as GradNorm (Abdelfattah et al., 2021), Plain (Abdelfattah et al., 2021), SNIP (Lee et al., 2019), GraSP (Wang et al., 2020), Fisher (Turner et al., 2020), and Synflow (Tanaka et al., 2020). These proxies evaluate the significance of network parameters and aggregate layer-wise values to estimate the overall performance.

Zero-cost Proxies for Transformer Recent efforts (Serianni and Kalita, 2023) have revitalized the application of zero-cost proxies for transformer-based networks, marking a significant milestone in the domain. LiteTransformerSearch (Javaheripi et al., 2022) observes that the zero-cost proxies, which exhibit promising performance in CV tasks, do not outperform the baseline methods in terms of the number of parameters of decoder when applied to NLP. Serianni and Kalita (2023) re-arouse the significance of Zero-cost (ZC) proxies in the context of RNN and BERT-based Transformer models, utilizing the FlexiBERT benchmark. It introduces an array of proxies, such as Synaptic Diversity, Synaptic Saliency, Activation Distance, Jacobian Cosine, Attention Confidence, and Head Importance, highlighting their potential in streamlining the architecture search process without extensive training.

Automatic Search for ZC Proxies Several studies explore how to search for ZC proxies automatically, notably EZNAS (Akhauri et al., 2022) and EMQ (Dong et al., 2023). EZNAS introduces a search space dedicated to convolution-based networks, achieving commendable performance across various benchmarks (Ying et al., 2019; Dong and Yang, 2020). However, its effectiveness is notably diminished when applied to Transformer-based networks. On the other hand, EMQ (Dong et al., 2023) develops a specialized search space tailored for mixed-precision quantization proxies but it is not optimized for Transformer-based networks. In contrast, our LPZero framework is specifically designed for language models, particularly Transformer architectures, and shows superior and more promising performance.

3 Methodology

In this section, we devise a search space and detail the evolution framework - LPZero with analysis to this framework.

OP	Symbols	Description
f_{01}	$\log(\cdot)$	$y = \log(x_1)$
f_{02}	$\lvert\log(\cdot)\rvert$	$y = \lvert\log(x_1)\rvert$
f_{03}	$\lvert\cdot\rvert$	$y = \lvert x_1\rvert$
f_{04}	$(\cdot)^2$	$y = (x_1)^2$
f_{05}	$e^{(\cdot)}$	$y = e^{x_1}$
f_{06}	$\sqrt{\cdot}$	$y = \sqrt{x_1}$
f_{07}	ReLU$(\cdot)$	$y = \max(0, x_1)$
f_{08}	$\frac{1}{(\cdot)}$	$y = \frac{1}{x_1}$
f_{09}	$\lvert\lvert\cdot\rvert\rvert_F$	$y = \lvert\lvert x_1\rvert\rvert_F$
f_{10}	norm_sum$(\cdot)$	$y = \frac{\sum_i^N x_1^i}{\text{numel}(x_1)}$
f_{11}	$\lvert\lvert\cdot\rvert\rvert_1$	$y = \lvert\lvert x_1\rvert\rvert_1$
f_{12}	softmax$(\cdot)$	$y = \frac{e^{x_1^i}}{\sum_{j=1}^n e^{x_1^j}}$
f_{13}	sigmoid$(\cdot)$	$y = \frac{1}{1+e^{-x_1^i}}$
f_{14}	log_softmax$(\cdot)$	$y = \log(f_{12}(x_1))$
f_{15}	$\sqrt{(\cdot)}$	$y = \sqrt{x_1}$
f_{16}	$-(\cdot)$	$y = -x_1$
f_{17}	min-max scaling$(\cdot)$	$y = \frac{(x-\min(x_1))}{\max(x_1)-\min(x_1)}$
f_{18}	average$(\cdot)$	$y = \frac{\sum_i^N x_1^i}{N}$
f_{19}	$\sigma(\cdot)$	$y = \sqrt{\frac{1}{N}\sum_{i=1}^N (x_1^i - \mu)^2}$
f_{20}	$\emptyset$	$y = x_1$
f_{21}	$\varnothing$	$y = \varnothing$
g_{01}	$(\cdot) + (\cdot)$	$y = x_1 + x_2$
g_{02}	$(\cdot) - (\cdot)$	$y = x_1 - x_2$
g_{03}	$(\cdot) \times (\cdot)$	$y = x_1 \cdot x_2$
g_{04}	$(\cdot) \odot (\cdot)$	$y = x_1 \odot x_2$

Table 2: Primitive operation set $\mathcal{K}$. Summary of unary (denoted by f) and binary Operations (denoted by g).

3.1 LPZero Search Space Design

The search spaces of most AutoML approaches (Real et al., 2020; Liu et al., 2019) are specifically designed for particular purposes and not suitable for proxy search. Previous auto loss search methods (Li et al., 2021b,a; Gu et al., 2022) takes the output of network y and ground truth $\hat{y}$ as input (scalar), which is relatively easy to handle. However, the search spaces of these methods are primitives, which is most similar to ours. However, for ZC proxies search problem, we involve more operations that taking scalar, vector and matrix as input, which might deduce the shape mismatching problem.

LPZero aims to identify the most suitable Zero-cost (ZC) proxy to accurately assess network performance. The primary objective is to optimize the Spearman's rank correlation coefficient (ρ), which measures the ranking consistency of each ZC proxy. Thus, our training-free approach is formulated as follows:

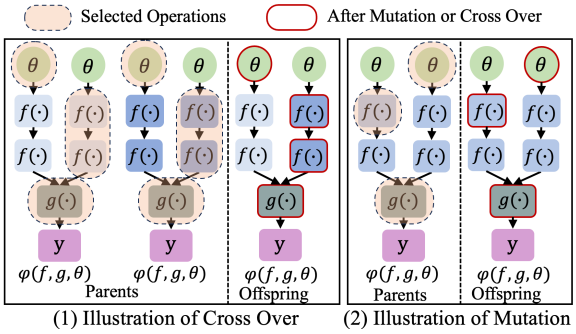

Figure 4: Illustration of Crossover and Mutation.

$$\varphi^* = \underset{\varphi \in \mathcal{S}}{\text{argmax}}(\rho(\varphi)),\ \varphi = \varphi(f, g, \theta). \quad (1)$$

where φ represents the candidate ZC proxies within the search space $\mathcal{S}$. Each proxy φ is defined as a function of unary and binary operations (f and g) applied to input parameters θ.

Zero-cost Proxy Representation. The ZC proxy φ is represented symbolically as an algorithmic expression (AE). As illustrated in Figure 2, the algorithmic expression can be represented by the combination of unary operations $f(\cdot)$ and binary operations $g(\cdot)$. Therefore, AE can be represented as $\varphi(f, g, \theta)$, where inputs x_1 and x_2 is chosen from six candidates θ, including Activation (*A*), Jacobs (*J*), Gradients (*G*), Head (*H*), Weight (*W*) and Softmax (*S*).

Primitive Operations. Table 2 summarizes the primitive operation set $\mathcal{K}$ used in our search space. This set comprises 20 unary operations and four binary operations, facilitating information exchange across dimensions. These operations are non-parametric, meaning they do not have adjustable parameters, making them highly efficient and effective in various computational tasks. Unary operations act on a single input, while binary operations operate on pairs of inputs. Notably, f_{20} and f_{21} are unique unary operations; f_{20} signifies a pass-through where the input is returned without any modification, and f_{21} represents a pruning operation that results in the removal of the branch, effectively returning nothing. By incorporating this diverse set of operations, our search space can explore a wide range of function transformations, enabling the discovery of novel architectures and enhancing the flexibility of our approach.

Analysis for the Search Space. In Figure 2, we illustrate the search space by showcasing two proxies depicted in red and yellow lines, demonstrating

Algorithm 1 LPZero Algorithm

1: **Input:** Initial population size p, number of generations G, crossover rate C_r, mutation rate M_r
2: **Output:** ZC proxy with highest Spearman
3: Initialize population with p random ZC proxies
4: **for** $g = 1$ to G **do**
5: Evaluate fitness of each proxy in the population
6: Pick top $\mathcal{R}$ ratio as pool $\mathcal{Q}$
7: Select parents $\varphi^{n,m}$ randomly from $\mathcal{Q}$
8: CrossOver $\varphi^q = \text{CrossOver}(\varphi^n, \varphi^m)$ with probability C_r.
9: Mutation $\varphi^q = \text{Mutate}(\varphi^q)$ with probability M_r
10: **if** PPS(φ^q) is valid **then**
11: Add offspring to population
12: **else**
13: Jump to Line 8 and regenerate offspring φ^q
14: **end if**
15: Evaluate fitness of new offspring φ^q
16: Keep the top-p proxies for the next generation
17: **end for**
18: **return** the proxy with the highest Spearman

the variability and richness of architectural configurations. With a total of 21 unary operations and 4 binary operations available, the search space is expansive, yielding a combinatorial space of $C_6^2 \times 21^2 \times 4 = 26,460$ potential ZC proxies. This vast space enables exploration of a wide spectrum of architectural designs, allowing for the discovery of innovative solutions tailored to the specific requirements of NLP tasks.

3.2 Search Algorithm

Inspired by the AutoML (He et al., 2021; Li et al., 2019), evolutionary algorithm serve as the core mechanism for our search algorithm design. Evolutionary algorithms, a subset of genetic algorithms, mimic the process of natural selection by generating, evaluating, and selecting individuals in a population to solve optimization problems. Figure 3 illustrates the search pipeline of our LPZero framework. At initialization, we uniformly sample p ZC proxies from the search space to form the initial population. Then, we measure the ranking correlation on the search space to measure the predictability of each proxy. Then, for each iteration, we conduct tournament selection to pick $\mathcal{R}$ ratios from population ($\mathcal{R} = 10\%$ by default) as promising candidates, and then randomly sample two of them as parents $\varphi^{n,m}$. Then, the parents are utilized to perform crossover and mutation with probability of C_r and M_r respectively to get the offspring. To verify the effectiveness of offspring, we sample S candidate architectures from the search space and compute the ranking correlation of ground truth and proxy score. As the search space is very sparse with a large number of unpromising or even invalid ZC proxies, we propose Early-Stopping Strategy to filter out the candidates.

Crossover and Mutation. Each Algorithmic Expression (AE) consists of two branches and one aggregate node. These branches represent the individual components or operations within the proxy architecture, while the aggregate node combines the outputs of these branches to form the final proxy score. As shown in Figure 4, we present the illustration of CrossOver and Mutation. During the crossover operation, two parent AEs are selected, and genetic information is exchanged between them to generate offspring. This process involves swapping segments of the parent AEs to create new combinations of operations and architectures. Conversely, the mutation operation introduces random alterations to the genetic makeup of a single AE, potentially introducing novel architectures into the population.

Predictive-Pruning Strategy. The Predictive-Pruning Strategy in the LPZero framework serves a crucial role in managing the computational challenges posed by the expansive and sparsely populated search space. It works to promptly identify and discard unpromising or invalid Zero-cost (ZC) proxies, thereby conserving computational resources and expediting the search for optimal solutions. By utilizing predefined criteria as presented in Appendix B this strategy evaluates the viability of candidate proxies. Those failing to meet the specified criteria are removed from the population, reducing the search space and focusing computational efforts on promising candidates. Overall, this strategic filtering process enhances the efficiency and effectiveness of the LPZero framework, facilitating swifter progress towards the discovery of high-quality proxy architectures.

Searched ZC Proxy. Based on LPZero framework, we present the searched Zero-cost (ZC) proxy tailored for the FlexiBERT search space, character-

ized by a unique combination of structural and operational elements. The architecture of this proxy is delineated as follows: the input structure comprises heads and activation functions, and the tree structure utilizes operations such as element-wise reversion, element-wise power, Frobenius norm, and log softmax. The binary operation defined within this context is the element-wise summation. The mathematical formulation of the searched ZC proxy is given by:

$$\varphi(\theta_H, \theta_A) = \sum_{i=0}^{N}((\frac{1}{\theta_H})^2 + \log\left(\eta\left(||\theta_A||_F\right)\right)) \quad (2)$$

where θ_H denotes the parameters associated with the heads in the Multi-head Attention, θ_A represents the activation values of each block within the network, and η symbolizes the softmax operation.

The formulated Zero-cost (ZC) proxy equation effectively evaluates neural architectures by considering both their structural efficiency and functional performance. The first term prioritizes models with fewer, yet efficient, parameters in the attention mechanism $\left(\frac{1}{\theta_H}\right)^2$, highlighting the goal of Zero-cost NAS towards computational efficiency. The second term $\log\left(\eta\left(\|\theta_A\|_F\right)\right)$ focuses on the diversity and distribution of activations, aiming for architectures that ensure balanced and effective information processing. Together, these aspects form a comprehensive approach for the holistic evaluation of architectures in the FlexiBERT search space, which is critical for identifying optimal models for NLP tasks.

4 Experiments

In this section, we first detail the experimental setup and implementation details of LPZero. Then, we present the ranking correlation evaluation on FlexiBERT and GPT-2 Search Space. Subsequently, we assess LPZero's performance by examining the ranking correlation in the FlexiBERT and GPT-2 search spaces. Lastly, we conduct an ablation study to evaluate the impact of our evolutionary algorithm, the Predictive-Pruning Strategy (PPS), and other variables such as the number of unary operations and the initial population size.

4.1 Implementation Details

Datasets. FlexiBERT (Serianni and Kalita, 2023) is built on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) including several tasks. We adopt the average performance of these tasks as ground truth to measure the ranking consistency. We utilize OpenWebText dataset (Gokaslan et al., 2019) during searching ZC proxies on FlexiBERT search space. For GPT-2 search space, we conduct experiments on WikiText-103 dataset (Merity et al., 2016). During evolution searching, we only require a mini-batch of input (batch size of 128 and 16 for BERT and GPT-2) to calculate the input statistics.

Criteria. The effectiveness of ZC proxies is measured by Kendall's τ and Spearman's ρ, with values from -1 (negative correlation) to 1 (positive correlation), where 0 indicates no correlation. These metrics allow us to assess the alignment between the proxies' predictions and actual model performance, providing a quantitative basis for comparison.

Search Space. We employ two existing benchmarks as the search space. FlexiBERT Benchmark (Serianni and Kalita, 2023) is a challenging benchmark that encompasses over 10^7 architectures (Refer to Appendix D.1 for more details.) We adopt the GPT-2 Benchmark (Javaheripi et al., 2022) on WikiText-103, which provides 10^{54} architectures (Refer to Appendix D.2 for more details).

Evolution Settings. The configuration of our evolutionary algorithm is as follows: The total number of generations, denoted as G, is established at 1,000, with the initial population size, p, set to 80 individuals. The probabilities for crossover and mutation operations are both set at $C_r = 0.5$ and $M_r = 0.5$, respectively. The selection pressure, represented by the ratio $\mathcal{R}$, is fixed at 10%. A consistent seed of 42 is utilized to ensure reproducibility across experiments. These experiments are conducted using A6000 GPUs to leverage their computational efficiency. To expedite the evolutionary search process, we assess the ranking consistency by sampling 50 architectures. Upon finalizing the search proxy, we proceed to evaluate its performance by applying it to two distinct datasets: FlexiBERT, comprising 500 architectures, and GPT-2, encompassing 200 architectures. The whole evolution process require 10 GPU hours.

Training and Evaluation. We leverate the open-source code by Serianni and Kalita (2023) and Abdelfattah et al. (2021) to implement the FlexiBERT and various proxies as shown in Table 1. We further use the source code in Javaheripi et al. (2022) to implement the GPT-2 search space and we collect the benchmark data from their open-sourced repository. To assess ranking consistency, we randomly sample

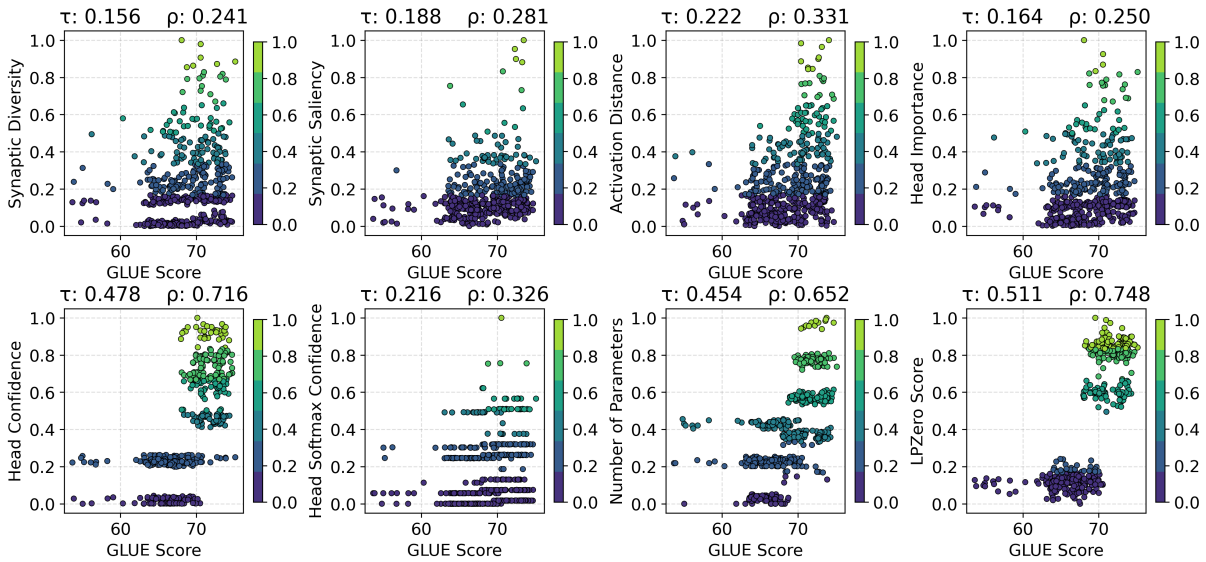

Figure 5: Spearman's ρ and Kendall's τ Correlation of training-free proxies with GLUE Score across 500 architectures randomly sampled from FlexiBERT Search Space.

Proxy Name	τ	ρ
Synaptic Diversity (Zhou et al., 2022)	0.021	0.175
Head Importance (Serianni and Kalita, 2023)	0.050	0.171
Activation Distance (Mellor et al., 2021)	0.081	0.123
Jacobian Cosine (Celotti et al., 2020)	0.116	0.149
SNIP (Lee et al., 2019)	0.119	0.173
GraSP (Wang et al., 2020)	0.122	0.179
GradNorm (Abdelfattah et al., 2021)	0.133	0.197
Fisher (Turner et al., 2020)	0.139	0.209
Synaptic Saliency (Tanaka et al., 2020)	0.157	0.266
Synflow (Tanaka et al., 2020)	0.322	0.471
LogSynflow (Cavagnero et al., 2023)	0.334	0.491
No.Params. (Abdelfattah et al., 2021)	0.454	0.590
Attention Confidence (Serianni and Kalita, 2023)	0.475	0.666
EZNAS (Akhauri et al., 2022)	0.483	0.698
LPZero (Ours)	**0.511**	**0.748**

Table 3: Ranking correlation of Zero-cost proxies on the FlexiBERT benchmark over 500 architectures with Kendall's τ and Spearman's ρ.

Proxy Name	τ	ρ
Jacobian Cosine (Celotti et al., 2020)	0.227	0.362
EZNAS (Akhauri et al., 2022)	0.489	0.704
No.Params (Abdelfattah et al., 2021)	0.582	0.737
Synflow (Tanaka et al., 2020)	0.632	0.730
Activation Distance (Mellor et al., 2021)	0.644	0.818
Attention Confidence (Serianni and Kalita, 2023)	0.676	0.850
Fisher (Turner et al., 2020)	0.691	0.872
GraSP (Wang et al., 2020)	0.765	0.922
GradNorm (Abdelfattah et al., 2021)	0.834	0.958
LogSynflow (Cavagnero et al., 2023)	0.836	0.962
Synaptic Diversity (Zhou et al., 2022)	0.841	0.957
Decoder.Params (Javaheripi et al., 2022)	0.847	0.967
Synaptic Saliency (Tanaka et al., 2020)	0.855	0.970
SNIP (Lee et al., 2019)	0.858	0.970
Head Importance (Serianni and Kalita, 2023)	0.861	0.971
LPZero (Ours)	**0.886**	**0.980**

Table 4: Ranking correlation of Zero-cost proxies on the GPT-2 search space over 200 architectures with Kendall's τ and Spearman's ρ.

500 architectures from the FlexiBERT benchmark, with findings presented in Table 3. Similarly, for the GPT-2 benchmark, we randomly sample 200 architectures to evaluate their ranking consistency, as detailed in Table 4.

4.2 Ranking Evaluation

Performance on FlexiBERT As illustrated in Table 3, we benchmark the Kendall's τ and Spearman's ρ of 14 zero-cost proxies over 500 architectures from the FlexiBERT search space. The baseline (number of parameters) serves as a competitive counterparts and most of proxies fail to surpass the baseline, which is also shown in Figure 1. Our LPZero model demonstrates superior ranking consistency, as evidenced by the values of $\tau = 0.51$ and $\rho = 0.75$ for the respective coefficients. Furthermore, we elucidate the correlation between GLUE scores and Zero-cost (ZC) proxies through Figure 5, which contrasts LPZero with the existing ZC proxies (Serianni and Kalita, 2023) in their study on training-free evaluation methods. This comparison clearly illustrates that our methodology exhibits the highest ranking consistency among the evaluated frameworks.

Performance on GPT-2 As illustrated in Table 4, we benchmark the Kendall's τ and Spearman's ρ of 15 zero-cost proxies over 200 randomly sam-

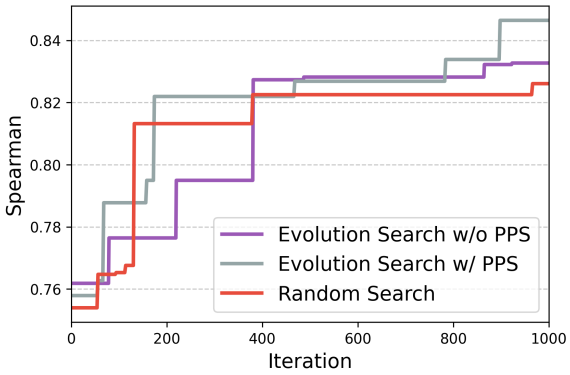

Figure 6: Performance comparison of evolution search with and without the Predictive-Pruning Strategy (PPS) and random search across iterations.

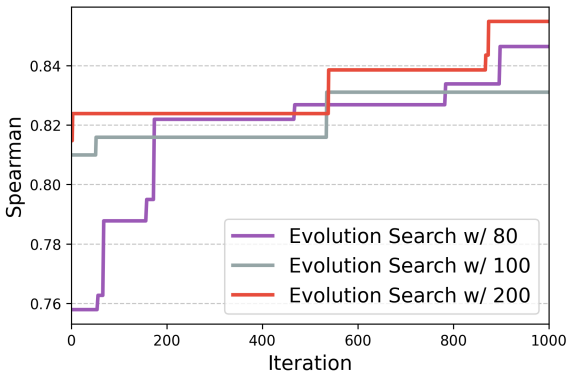

Figure 7: Performance comparison of different size of population.

pled architectures from GPT-2 search space. The additional proxy (Javaheripi et al., 2022) is "Decoder.Params", which represent the parameter of decoder in GPT-2 models. Our LPZero achieve the SOTA performance among all ZC proxies, achieving $\tau = 0.87$ and $\rho = 0.98$. Compared with FlexiBERT search space, the ranking consistency is much higher than GPT-2 search space.

4.3 Ablation Study

We conduct extensive ablation study over the four factors: (1) Evolutionary Algorithm. (2) Predictive-Pruning Strategy. (3) Initial Population Size. (4) Number of Unary Operation.

(1) Effectiveness of Evolutionary Algorithms. As depicted in Figure 6, we limit the number of iterations to 1,000, maintaining an initial population size of 80 throughout the process. The findings reveal that the Evolutionary Algorithm substantially surpasses the performance of Random Search. This indicates that the evolutionary algorithm can heuristically enhance the speed of the search process, thereby significantly improving search efficiency.

#Unary	2	3	4	5
Spearman's ρ	**86.48%**	77.47%	75.15%	78.12%
Winning Rate	**25.61%**	7.96%	8.69%	6.25%

Table 5: Influence of the Number of Unary Operations on Spearman's ρ and Winning Rate.

(2) Effectiveness of Predictive-Pruning Strategy (PPS). As illstrated in Figure 6, we present the performance of the Predictive-Pruning Strategy (PPS). Our findings indicate that for iterations fewer than 400, PPS not only achieves higher Spearman's ρ but also significantly outperforms evolutionary search methodologies not incorporating PPS, highlighting its critical role in enhancing search efficiency.

(3) Initial Population Size. As illustrated in Figure 7, we compare Spearman's ρ across varying initial population sizes of 80, 100, and 200. The data indicate a positive correlation between population size and the initial Spearman's Coefficient value: larger initial populations yield higher Spearman's ρ at the outset.

(4) Number of Unary Table 5 presents an ablation study examining the effect of unary operation counts on Spearman's rank correlation coefficient and winning rate. The study shows that a lower number of unary operations (2) yields the highest Spearman correlation (86.48%) and winning rate (25.61%), indicating that large unary operations may lead to over-complex proxies.

5 Conclusion

In this paper, we present the LPZero framework, an innovative approach for discovering proxies for language models without involving extensive training or expert intervention. Our approach encompasses the design of a comprehensive search space, capturing a wide array of existing ZC proxies. Utilizing an Evolutionary Algorithm, we efficiently unearth promising ZC proxies within this space. To expedite the search, we implement a Predictive-Pruning Strategy, eliminating less promising proxies early in the process. To verify the effectiveness of our LPZero, we conduct experiments on FlexiBERT and GPT-2 search space to measure the ranking consistency of the searched proxy. Experimental results demonstrate that our LPZero have better ranking ability compared with previous ZC proxies, and surpass the baseline by a large margin. Our findings pave the way for future explorations in Zero-cost proxies for language models.

6 Limitations

This study undertakes a comprehensive review of existing Zero-cost (ZC) proxies specifically tailored for Transformer architectures, integrating them into a unified framework for evaluation. By benchmarking these ZC proxies within the FlexiBERT and GPT-2 search spaces, we rigorously assess their ranking capabilities through Kendall's τ and Spearman's ρ. This approach allows us to present a systematic comparison of their effectiveness in identifying promising language model architectures without the need for extensive computational resources. Our evaluation focuses on the architectural aspects of language models, aiming to streamline the search process for efficient and effective neural network designs.

However, it's important to note that our research primarily concentrates on the structural design and optimization of language models, sidelining enhancements in specific functional areas such as inference capabilities, logical analysis, advanced language generation, nuanced natural language understanding, and the retrieval and integration of knowledge. These critical components of language model performance and applicability in real-world applications are not directly addressed by our current framework. Recognizing these gaps, we identify substantial opportunities for future research to delve into these aspects. Expanding the scope of Zero-cost proxy evaluation to include these functionalities could significantly elevate the utility and comprehensiveness of language models, offering a more holistic approach to their development and assessment in the field of artificial intelligence.

7 Ethics Statement

Our LPZero framework addresses the technical development of language model architectures, sidestepping direct ethical or social considerations. Our work is likely to increase the adoption of NAS in the NLP domain, providing an economic way to perform estimation in language models.

Despite this focus, we recognize that the application of our findings—aimed at reducing computational demands and streamlining language model development—could intersect with broader ethical issues in natural language processing, such as data privacy, algorithmic bias, and the potential for misuse. We advocate for future research to integrate ethical considerations, scrutinize training data sources for biases, and ensure the responsible deployment of language models, acknowledging their profound societal impact. We acknowledge the significant capabilities and prospects offered by artificial intelligence, particularly ChatGPT, in refining written materials. As we utilize this technology to enhance paragraphs, we pledge to adhere strictly to the utmost ethical guidelines, thereby guaranteeing the preservation of integrity, the respect of intellectual property rights, and the support of inclusivity. It is important to clarify that our use of ChatGPT is limited to the refinement of existing content rather than the generation of new content for the paper.

References

Mohamed S. Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D. Lane. 2021. Zero-Cost Proxies for Lightweight NAS. In *ICLR*.

Yash Akhauri, Juan Pablo Munoz, Nilesh Jain, and Ravishankar Iyer. 2022. EZNAS: Evolving zero-cost proxies for neural architecture scoring. In *NeurIPS*.

Niccolò Cavagnero, Luca Robbiano, Barbara Caputo, and Giuseppe Averta. 2023. Freerea: Training-free evolution-based architecture search. In *WACV*, pages 1493–1502.

Luca Celotti, Ismael Balafrej, and Emmanuel Calvet. 2020. Improving zero-shot neural architecture search with parameters scoring. *OpenReview*. https://openreview.net/forum?id=4QpDyzCoH01.

Peijie Dong, Lujun Li, Zimian Wei, Xin Niu, Zhiliang Tian, and Hengyue Pan. 2023. Emq: Evolving training-free proxies for automated mixed precision quantization. In *ICCV*, pages 17076–17086.

Xuanyi Dong and Yi Yang. 2020. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Hongyang Gu, Jianmin Li, Guang zhi Fu, Chifong Wong, Xinghao Chen, and Jun Zhu. 2022. Autolossgms: Searching generalized margin-based softmax loss function for person re-identification. *CVPR*, pages 4734–4743.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. Automl: A survey of the state-of-the-art. *Knowl. Based Syst.*, 212:106622.

Mojan Javaheripi, Gustavo de Rosa, Subhabrata Mukherjee, S. Shah, Tomasz L. Religa, Caio Cesar Teodoro Mendes, Sébastien Bubeck, Farinaz Koushanfar, and Debadeepta Dey. 2022. Litetransformersearch: Training-free neural architecture search for efficient language models. In *NeurIPS*.

Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. 2022. Nasbench-suite-zero: Accelerating research on zero cost proxies. *NeurIPS*, 35:28037–28051.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Communications of The ACM*, 60(6):84–90.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *NeurIPS*, volume 2.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2019. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*.

Chuming Li, Xin Yuan, Chen Lin, Minghao Guo, Wei Wu, Junjie Yan, and Wanli Ouyang. 2019. Am-lfs: Automl for loss function search. In *ICCV*.

Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. 2023. Zico: Zero-shot NAS via inverse coefficient of variation on gradients. In *ICLR*.

Hao Li, Tianwen Fu, Jifeng Dai, Hongsheng Li, Gao Huang, and Xizhou Zhu. 2021a. Autoloss-zero: Searching loss functions from scratch for generic tasks. *CVPR*, pages 999–1008.

Hao Li, Chenxin Tao, Xizhou Zhu, Xiaogang Wang, Gao Huang, and Jifeng Dai. 2021b. Auto seg-loss: Searching metric surrogates for semantic segmentation. In *ICLR*.

Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. 2021. Zen-nas: A zero-shot nas for high-performance image recognition. *ICCV*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: differentiable architecture search. In *7th ICLR, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, volume abs/1806.09055.

Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. 2021. Neural architecture search without training. In *ICML*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *AAAI*.

Esteban Real, Chen Liang, David So, and Quoc Le. 2020. Automl-zero: Evolving machine learning algorithms from scratch. In *ICML*.

Aaron Serianni and Jugal Kalita. 2023. Training-free neural architecture search for RNNs and transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2522–2540, Toronto, Canada. ACL.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. *NeurIPS*, 33:6377–6389.

Jack Turner, Elliot J. Crowley, Michael O'Boyle, Amos Storkey, and Gavin Gray. 2020. Blockswap: Fisher-guided block substitution for network compression on a budget. In *ICLR*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020. Picking winning tickets before training by preserving gradient flow. In *ICLR*.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. 2019. NAS-Bench-101: Towards reproducible neural architecture search. In *ICML*.

Qinqin Zhou, Kekai Sheng, Xiawu Zheng, Ke Li, Xing Sun, Yonghong Tian, Jie Chen, and Rongrong Ji. 2022. Training-free transformer architecture search. In *CVPR*, pages 10894–10903.

Barret Zoph and Quoc V Le. 2017. Neural architecture search with reinforcement learning. In *ICLR*.

A Additional Related Work

Activation Distance Activation Distance, specifically in the context of NWOT (Mellor et al., 2021), leverages binary activation patterns to measure the correlation between input data across ReLU (Rectified Linear Unit) layers within a neural network. This proxy is crucial for understanding how different inputs activate the network's architecture, providing insights into the diversity and richness of the learned representations. The formula provided,

$$\mathcal{S} = \log |K_H| \tag{3}$$

where K_H represents the kernel matrix, quantifies the similarity (or distance) between activation patterns. The determinant of the kernel matrix ($|K_H|$) captures the volume of the space spanned by the activations, and taking its logarithm transforms this volume measure into a more manageable scale.

Synaptic Saliency Synaptic Saliency, or Synflow (Tanaka et al., 2020), is a criterion used to identify the importance of parameters (weights) in a neural network, aiming to approximate the impact on the loss function when a specific parameter is removed. This concept is framed within the equation,

$$\mathcal{S} = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \tag{4}$$

where $\frac{\partial \mathcal{L}}{\partial \theta}$ denotes the gradient of the loss function with respect to the parameters (θ), and $\odot$ represents the Hadamard product, signifying element-wise multiplication between the gradient and the parameters themselves. This approach to quantifying parameter importance is designed to prevent layer collapse during the pruning process of network training, ensuring that the pruning does not disproportionately affect any single layer which could result in significant performance degradation.

Jacobian Score Cosine The Jacobian Score Cosine (JSC) (Celotti et al., 2020) is a Zero-cost Proxy designed to evaluate the sensitivity and stability of neural network architectures with respect to their input data. By analyzing the Jacobian matrix, which represents the first derivatives of the network's outputs with respect to its inputs, the JSC offers insights into how small variations in the input can affect the output, thereby assessing the network's robustness and generalization capability. The JSC is computed using the following formula:

$$S = 1 - \frac{1}{N^2 - N} \sum_{i=1}^{N} \left[J_n J_n^t - I \right]^{\frac{1}{20}}, \tag{5}$$

where S denotes the Jacobian Score, N is the number of inputs to the network, J_n represents the Jacobian matrix for the nth input, J_n^t is the transpose of J_n, and I is the identity matrix. This equation calculates the average cosine similarity between the Jacobian vectors of all pairs of inputs, adjusted by the identity matrix to normalize self-similarity, and finally raised to the power of $\frac{1}{20}$ to scale the measure.

Synaptic Diversity The concept of *Synaptic Diversity* within the context of Training-Free Transformer Architecture Search (TF-TAS) (Zhou et al., 2022) represents a novel approach towards evaluating and selecting Vision Transformer (ViT) architectures. By circumventing the need for extensive training, this methodology significantly enhances computational efficiency in Transformer Architecture Search (TAS). The TF-TAS scheme, delineated in the studies by Zhou et al., employs a modular strategy that assesses ViT architectures through two theoretical lenses: synaptic diversity and synaptic saliency, collectively referred to as the DSS-indicator.

Synaptic Diversity, particularly in relation to multi-head self-attention (MSA) modules of ViTs, is instrumental in gauging the performance of these architectures. This proxy evaluates the heterogeneity of synaptic connections by utilizing the Nuclear-norm as an approximate measure for the rank of weight matrices within MSA modules. A higher Nuclear-norm indicates a greater diversity, which suggests a potential for enhanced performance due to the ability to encapsulate a broader spectrum of features and relationships within the data. The computation of Synaptic Diversity is formalized as follows:

$$S = \sum_{m} \left\| \frac{\partial \mathcal{L}}{\partial W_m} \right\| \odot \|W_m\|_{\text{nuc}} \tag{6}$$

Here, S symbolizes the synaptic diversity score, $\frac{\partial \mathcal{L}}{\partial W_m}$ denotes the gradient of the loss function with respect to the weights of the m-th MSA module, and $\|W_m\|_{\text{nuc}}$ is the Nuclear-norm of the weight matrix, serving as a proxy for the rank and thus the diversity of the synaptic connections.

Hidden Covariance The Hidden Covariance proxy provides a sophisticated means to analyze the behavior and interaction of hidden states within a specific layer of a Recurrent Neural Network (RNN) when processing a minibatch of N input sequences $X = \{x_n\}_{n=1}^{N}$. This proxy is particularly insight-

ful for examining the internal dynamics and dependencies of the hidden states across different time steps or sequences. Given the hidden state matrix $H(X)$ for a minibatch, we first compute the covariance matrix C as follows:

$$C = (H - M_H)(H - M_H)^T, \tag{7}$$

where M_H is the mean matrix derived from the hidden states, with its elements defined by:

$$(M_H)_{ij} = \frac{1}{N}\sum_{n=1}^{N} H_{in}, \tag{8}$$

indicating the average activation across the minibatch for each hidden unit. This step captures the variance and covariance of the hidden states, highlighting the variability and correlation of activations in response to the input batch. Subsequently, to normalize and interpret the covariance values, we calculate the Pearson product-moment correlation coefficients matrix R as:

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}}, \tag{9}$$

which standardizes the covariance matrix into a correlation matrix R, providing a normalized measure of linear dependencies between pairs of hidden units.

Building upon the framework established by Mellor et al. (2021), the final proxy $S(H)$ is derived using the Kullback–Leibler divergence from the eigenvalues of the kernel of R, computed as:

$$S(H) = -\sum_{n=1}^{N}\left(\log(\lambda_n + k) + \frac{1}{\lambda_n + k}\right), \tag{10}$$

where $\lambda_1, \ldots, \lambda_N$ are the eigenvalues of R, and $k = 10^{-5}$ is a small constant added to stabilize the logarithm and reciprocal operations.

Note that Hidden Covariance is designed for RNN architectures, which means it is not working for Transformer-based networks. That is why we don not report the performance of Hidden Covariance on FlexiBERT and GPT-2 search space.

Confidence The Confidence proxy (Serianni and Kalita, 2023) quantifies the average maximum attention (or activation) that a neural network layer, specifically an attention mechanism, directs towards the most significant features or tokens for a set of inputs X. This is mathematically articulated as:

$$\mathcal{S} = \frac{1}{N}\sum_{n=1}^{N} \max(\text{Att}(h, x_n)) \tag{11}$$

In this expression, $\mathcal{S}$ symbolizes the average maximal attention score across all instances within the minibatch, where $\text{Att}(h, x_n)$ signifies the attention scores calculated for the n-th input by the function h.

Softmax Confidence Softmax Confidence (Serianni and Kalita, 2023) broadens the notion of Confidence to scenarios where softmax scores, derived from the softmax function σ, are utilized to gauge the network's prediction certainty. The formulation is given by:

$$\mathcal{S} = \frac{1}{N}\sum_{n=1}^{N} \max(\sigma(h, x_n)) \tag{12}$$

Here, $\sigma(h, x_n)$ computes the softmax probabilities for the outputs related to the n-th input, and the $\max$ operation selects the highest probability, denoting the model's most confident prediction for each input. The mean of these maxima across the minibatch offers a measure of the overall prediction confidence, valuable for assessing the certainty of classification decisions by the model.

Importance The **Importance** proxy (Serianni and Kalita, 2023) assesses the sensitivity of the cost function $\mathcal{C}(X)$ with respect to the attention mechanism $\text{Att}_h(X)$ for a given input set X. This sensitivity analysis is crucial for understanding the impact of changes in attention weights on the overall performance or cost of the neural network. The Importance proxy is mathematically represented as:

$$\mathcal{S} = \left|\frac{\partial \mathcal{C}(X)}{\partial \text{Att}_h(X)}\right| \tag{13}$$

This equation calculates the absolute value of the derivative of the cost function relative to the attention weights, quantifying the "importance" of the attention mechanism in the network's decision-making process. A higher value suggests that minor adjustments to the attention weights could lead to significant changes in the cost, underscoring the critical areas of the input that the network focuses on.

SNIP (Single-shot Network Pruning) (Lee et al., 2019) introduces a pruning criterion that can be applied early in the training process, even before the actual training commences. It is predicated on the sensitivity of the loss function $\mathcal{L}$ with respect to

each parameter θ, modulated by the parameter values themselves. The SNIP criterion is formulated as:

$$S(\theta) = \left| \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \right| \tag{14}$$

Here, the operation $\odot$ denotes the element-wise product. This expression evaluates the absolute value of the gradient of the loss function with respect to the parameters, weighted by the parameters themselves. This criterion aids in identifying parameters that have minimal impact on the loss function, allowing for their pruning to streamline the model architecture without significantly compromising performance.

GraSP (Gradient Signal Preservation) (Wang et al., 2020) introduces a pruning methodology aimed at preserving the gradient flow throughout the network's architecture. This strategy identifies and eliminates parameters that have the least effect on the gradient flow, thus minimizing their impact on the network's ability to learn. The GraSP criterion is quantitatively defined by the equation:

$$S(\theta) = -\left(H \frac{\partial \mathcal{L}}{\partial \theta} \right) \odot \theta \tag{15}$$

In this formulation, $S(\theta)$ denotes the pruning score assigned to each parameter θ, reflecting its significance in maintaining effective gradient flow within the network. The term H represents the Hessian matrix, which consists of the second-order derivatives of the loss function $\mathcal{L}$ with respect to the parameters, while $\frac{\partial \mathcal{L}}{\partial \theta}$ is the gradient of the loss with respect to the parameters. The operation $\odot$ signifies element-wise multiplication, and the negative sign indicates that parameters which contribute negatively to the gradient flow—and therefore potentially hinder learning—are prioritized for removal.

The principal insight of GraSP is its emphasis on the Hessian-gradient product, which offers a measure of the influence of parameter changes on the curvature of the loss landscape and, subsequently, on the dynamics of model training. By focusing on preserving parameters critical for the integrity of gradient flow, GraSP enables network pruning in a manner that is less likely to degrade performance.

In this paper, we have chosen not to incorporate the Hessian Matrix as part of our analysis due to its computationally intensive nature. However, it is worth noting that excluding considerations of computational load, the inclusion of the Hessian Matrix could potentially enhance performance significantly.

LogSynflow (Cavagnero et al., 2023) introduces a nuanced variation to the conventional pruning criteria by applying a logarithmic transformation to the gradients' magnitude. This adjustment is intended to enhance the pruning strategy by ensuring a more nuanced evaluation of parameter importance, especially for those with small but significant gradients. The LogSynflow criterion is mathematically expressed as:

$$S(\theta) = \theta \cdot \left| \log \left| \frac{\partial \mathcal{L}}{\partial \theta} \right| \right| \tag{16}$$

In this equation, $S(\theta)$ represents the score assigned to each parameter θ based on its importance, where $\frac{\partial \mathcal{L}}{\partial \theta}$ denotes the gradient of the loss function $\mathcal{L}$ with respect to the parameters. The use of the absolute value of the logarithm of the gradient magnitude aims to highlight the significance of parameters that might otherwise be overlooked due to their relatively small gradient values. By multiplying these logarithmic values by the parameters themselves, LogSynflow prioritizes the retention of parameters that are integral to the network's ability to learn, thereby facilitating a more informed pruning process that minimizes the loss of critical information.

B Predefined Criteria in PPS

In mathematics, understanding the relationships between various operations significantly impacts the LPZero search space. Table 6 summarizes the relationships among a set of operations, categorizing them based on their mathematical interactions. These relationships include inverse functions, derivatives, equivalence, special cases, and potential conflicts when certain operations are combined. This overview helps in recognizing how operations can complement or conflict with each other, thereby providing support for PPS.

C Correlation of Rank

As a complement to the visualization of ranking correlation, we follow LiteTransformerSearch (Javaheripi et al., 2022) and provide visualizations of GLUE Score Ranking and ZC Proxies Ranking. It can be observed that potential proxies are capable of dividing the candidate models into

OP 1	OP 2	Relationship	Description
log	exp	Inverse	e^x and log(x) are inverse functions.
abs	abs(log)	Derivative	Absolute value operation applied to log.
square	sqrt	Inverse	Squaring and square root are inverse operations.
ReLU	identity	Special case	ReLU acts as identity for x>0.
inverse	identity	Inverse for non-zero	Multiplicative inverse operation.
norm_sum	average	Equivalent	Norm sum divided by count is average.
softmax	log_softmax	Derivative	Log softmax is the logarithm of softmax.
sigmoid	logistic function	Equivalent	Sigmoid is also known as the logistic function.
min-max scaling	normalization	Type	Min-max scaling is a type of normalization.
standard deviation	variance	Square root	Standard deviation is the square root of variance.
L1-norm	abs	Generalization	L1-norm is a sum of absolute values.
F-norm	Euclidean norm	Equivalent	Frobenius norm for matrices, Euclidean norm for vectors.
-()	log	Conflict	Negation followed by log leads to undefined result for positive inputs.
-()	sqrt	Conflict	Negation followed by sqrt leads to undefined result for positive inputs.
identity (if zero)	inverse	Conflict	Identity ensuring zero input followed by inverse leads to division by zero.
-() (for positive)	sqrt	Conflict	Negation of positive numbers followed by sqrt is undefined.

Table 6: Summary of Predefined Criteria

two clusters at least through ranking. This further demonstrates the robustness of our LPZero results.

D Details of Search Space

D.1 Details of FlexiBERT

The table 7 provides a detailed overview of the FlexiBERT search space, highlighting the diverse range of hyperparameters available for tuning. FlexiBERT, designed to explore architectural variations within the BERT model framework, allows for configurations that span the specifications of BERT-Tiny and BERT-Mini. Key architectural elements are outlined along with their corresponding hyperparameters values, including hidden dimension sizes, the number of encoder layers, types of attention operators, and more. Notably, the hidden dimension and the number of encoder layers are consistent across the architecture, whereas other parameters vary across encoder layers, introducing a high degree of flexibility and customization. The table also specifies the conditions under which different attention operation parameters are applied, depending on the type of attention operator selected. With a total of 10,621,440 possible architectures, this search space represents a comprehensive framework for exploring and identifying efficient model configurations within the BERT architecture spectrum.

D.2 Details of GPT-2

Table 8 delineates the expansive search space leveraged for the GPT-2 architecture optimization, outlining a comprehensive set of hyperparameters targeted in the exploration process. It includes the number of layers (nlayer), representing the depth of the transformer model; the dimensionality of model embeddings (dmodel), indicative of the scale and capacity of the model; the inner dimension of the feed-forward networks (dinner), a critical parameter for the model's ability to process and integrate information within each transformer layer; the number of attention heads (nhead), which impacts the model's ability to focus on different parts of the input sequence; and the dimensions of adaptive input embeddings (dembed) along with their associated scaling factors (k), parameters that offer a novel approach to managing input representation complexity and efficiency. A noteworthy aspect of this search space is the adaptive setting of the dinner parameter, which is dynamically adjusted to be at least twice the dmodel size, a heuristic introduced to mitigate the risk of training collapse by ensuring a sufficient capacity in the feed-forward networks.

E Ablation Study of Unary

Figure 13 illustrates an ablation study that investigates the performance of systems with varying unary operations. It presents four graphs, each plotting performance metrics in 100 iterations for systems with two to five unary operations. The study finds that the system with two unary operations achieves and maintains the highest 'Best SP' score, indicating stable, optimal performance. Systems with more than two unary operations show more fluctuations in 'Best SP' and a lower Spearman rank correlation, suggesting that additional operations may lead to over-complexity and reduced performance. Thus, the optimal number of unary operations for this system is two, balancing complexity and performance.

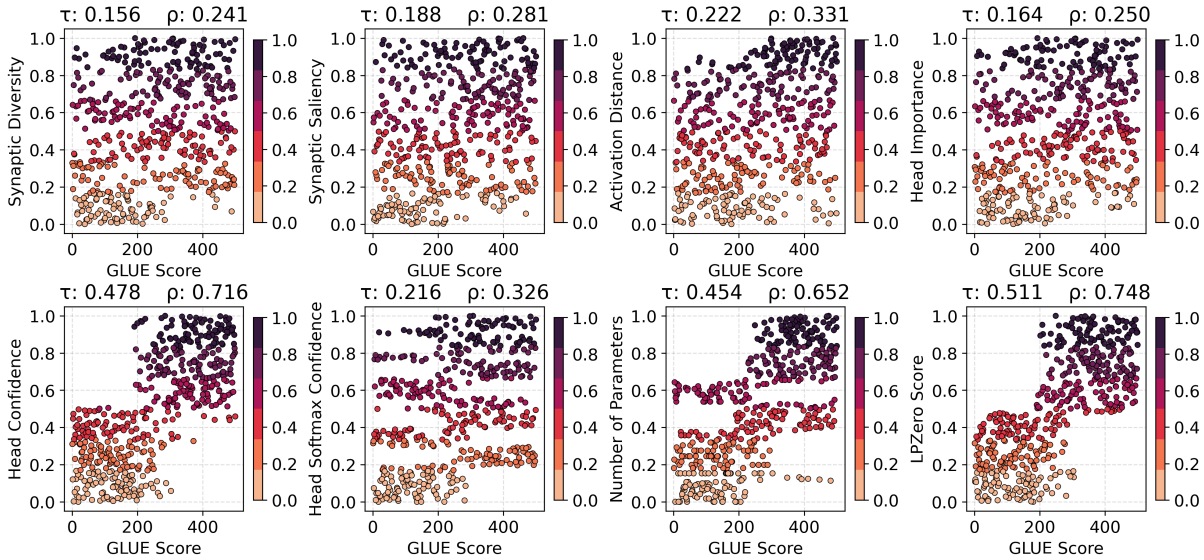

Figure 8: Correlation of training-free proxies ranking with GLUE Ranking on 500 architectures randomly sampled from FlexiBERT Search Space.

Architecture Element	**Hyperparameters Values**
Hidden dimension	{128, 256}
Number of Encoder Layers	{2, 4}
Type of attention operator	{self-attention, linear transform, span-based dynamic convolution}
Number of operation heads	{2, 4}
Feed-forward dimension	{512, 1024}
Number of feed-forward stacks	{1, 3}
Attention operation parameters	if self-attention {scaled dot-product, multiplicative} if linear transform {discrete Fourier, discrete cosine} if dynamic convolution convolution kernel size: {5, 9}

Table 7: The FlexiBERT search space, with hyperparameter values spanning those found in BERT-Tiny and BERT-Mini. Hidden dimension and number of encoder layers is fixed across the whole architecture; all other parameters are heterogeneous across encoder layers. The search space encompasses 10,621,440 architectures.

Architecture Element	**Hyperparameters Values**
Number of Layers (nlayer)	$\{2, 3, ..., 16\}$
Model Dimension (dmodel)	$\{128, 192, ..., 1024\}$
Inner Dimension (dinner)	$\{256, 320, ..., 4096\}$
Number of Attention Heads (nhead)	$\{2, 4, 8\}$
Adaptive Input Embedding Dimension (dembed)	$\{128, 256, 512\}$
Adaptive Input Embedding Factor (k)	$\{1, 2, 4\}$

Table 8: The GPT-2 search space, covering a broad spectrum of architectural configurations. Once a model dimension (dmodel) is chosen, the minimum inner dimension (dinner) is set to twice the value of dmodel to avoid training collapse. This adaptive approach ensures a wide range of effective and efficient architectures, summing up to more than 10^{54} unique configurations.

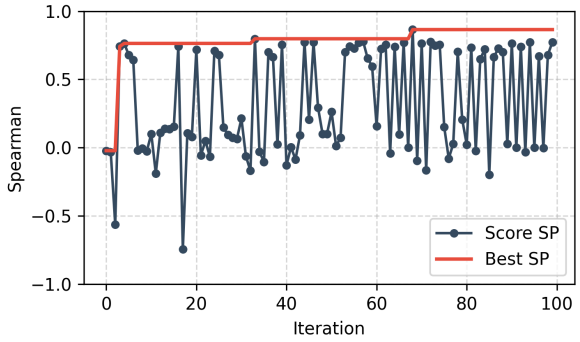

Figure 9: Two Unary Operations.

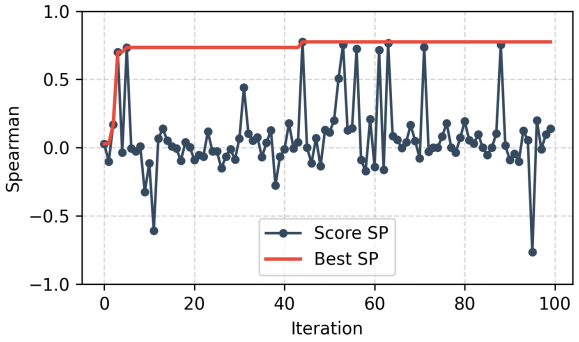

Figure 10: Three Unary Operations.

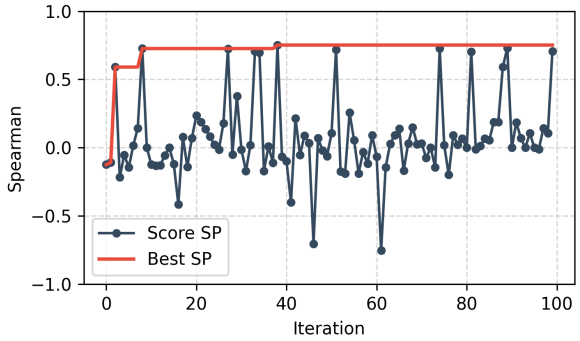

Figure 11: Four Unary Operations.

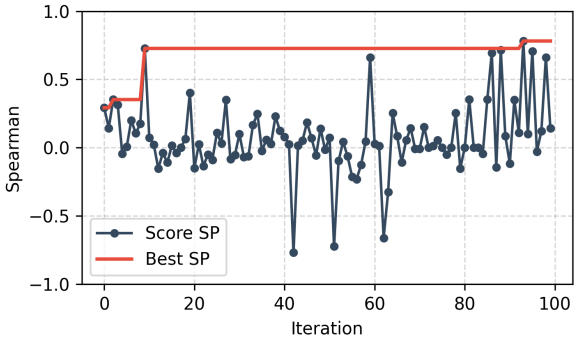

Figure 12: Five Unary Operations.

Figure 13: Ablation Study of the Number of Unary Operations.