ETAS: Zero-Shot Transformer Architecture Search via Network Trainability and Expressivity

Anonymous ACL submission

Abstract

Transformer Architecture Search (TAS) methods aim at automates searching the optimal Transformer architecture configurations for a given task. However, they are impeded by the prohibitive cost of evaluating Transformer architectures. Recently, several Zero-Shot TAS methods have been proposed to mitigate this problem by utilizing zero-cost proxies for evaluating Transformer architectures without training. Unfortunately, they are limited to specific tasks and lack theoretical guarantees. To solve this problem, we develop a new zero-cost proxy called NTSR that combines two theoretically-inspired indicators to measure the trainability and expressivity of Transformer networks separately. We then integrate it into an effective regularized evolution framework called ETAS demonstrate its efficacy on various tasks. The results show that our proposed NTSR proxy can consistently achieve a higher correlation with the true performance of Transformer networks on both computer vision and natural language processing tasks. Further, it can significantly accelerate the search process for finding the best-performing Transformer network architecture configurations.

1 Introduction

Transformer networks Li et al. (2022); Zhou et al. (2022); Chitty-Venkata et al. (2022) have attracted tremendous interest over the last few years due to their effectiveness in learning long-range dependencies in data and superior performance across various tasks. They have gradually replaced traditional neural networks, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), in a variety of domains including Natural Language Processing (NLP) (Javaheripi et al., 2022), Computer Vision(CV) (Chen et al., 2022), Speech Signal Processing (Chitty-Venkata et al., 2022), and Healthcare(Chitty-Venkata et al., 2022). Recently, the Transformer architecture has become the de facto backbone for most large language models (LLM). While in real applications, it is often necessary to adjust the Transformer architecture configurations according the specific tasks (Javaheripi et al., 2022), such as the depth of the network, the number of attention heads, embedding dimension, and the inner dimension of the feed-forward layer. Manual tuning these parameters requires repeated refinement with expert experience, which is time-consuming and computationally expensive.

To solve this problem, various Transformer Architecture Search (TAS) methods have been proposed, which automates searching the optimal Transformer architecture configurations for a given task and data. The current popular TAS methods include reinforcement learning evolutionary search, one-shot and predictor-based search. However, during the search process, they still demands a high computational cost to evaluate several hundreds or thousands of architectures. Training a Transformer network can take hours or even days, thus hindering the practical application of TAS. Recently, zero-shot Neural Architecture Search (NAS) methods have attracted much attention as they design zero-cost proxies to estimate the performance of a network at the initialization stage. They can quickly evaluate the performance of a network in a few seconds by computing statistics from a single forward/backward propagation pass of the network with a minibatch of data at initialization. Nevertheless, Zhou et al. (2022) have showed the majority of existing zero-cost proxies are specifically designed for the CNN search spaces ($e.g.$, NAS-Bench 101, NAS-Bench 201, DARTS (Zela et al., 2022)) and perform worse on the Transformer search space. They leverage the characteristics of Transformer networks and design a DSS zero-cost proxy that estimates the synaptic diversity of multi-head self-attention (MSA) and the synaptic saliency of multi-layer perceptron (MLP) in the Transformer network

to rank its performance in a Vision Transformer (ViT) search space. Later, Chen et al. (2022) introduce a zero-cost proxy that measures the complexity of manifold propagation through ViT to estimate how complex function can be approximated by a Vision Transformer network. Javaheripi et al. (2022) choose the number of decoder parameters in auto-regressive Transformers as a zero-cost proxy for perplexity of the language model without need for any model training.

Unfortunately, the existing zero-cost for Transformer proxies are limited to specific computer vision or natural language processing tasks. Nonetheless, most of them are designed based on empirical observations and lack theoretical assurances. **Can we design a theoretically-inspired zero-cost proxy applicable to multiple vision and language tasks ?** To this end, we propose a novel zero-cost proxy called NTSR that combines two theoretically-inspired indicators to measure the trainability and expressivity of Transformer networks. In particular, based on the theoretical underpinnings of deep neural network training, we design the NTKT metic that utilizes the trace of the mean Neural Tangent Kernel (NTK) to quantify the trainability of Transformer networks. Meanwhile, we design another SEPT metic that utilizes the upper bound of separation rank induced by the Transformer network to measure the capacity of transformer networks to represent input dependencies. To demonstrate the effectiveness of our proposed NTSR zero-cost proxy for Transformer networks, We compare it to other popular zero-cost proxies in multiple search spaces. The results show that NTSR can consistently achieve a higher correlation with the true performance of Transformer networks on both computer vision and natural language processing tasks. To investigate the ability of our proposed NTSR zero-cost proxy to accelerate the search of the best-performing Transformer network, we further integrate it into an effective regularized evolution framework called ETAS. The results show that NTSR can significantly speed up the process of finding the best-performing Transformer network.

2 Method

Notations Assume a standard depth-L Transformer network has one input embedding layer and L transformer blocks. The input of Transformer network is a sequence of T tokens $\{\boldsymbol{x}_t \in [V]\}_{t=1}^{T}$, where V is the number of vocabulary tokens. The embedding layer transforms the input sequence $\{\boldsymbol{x}_t \in [V]\}_{t=1}^{T}$ to T sequenced d_x^1-dimensional vectors $\boldsymbol{z}_t^1, t \in [T]$, which is defined as $\boldsymbol{z}_t^1 = \boldsymbol{W}_V \boldsymbol{x}_t + \boldsymbol{p}_t$, where $\boldsymbol{W}_V \in \mathbb{R}^{d_x^1 \times V}$ is the learned word-embedding matrix, $\boldsymbol{p}_t$ is the positional embedding vector. After that, $\boldsymbol{z}_t^1$ is recursively transformed into T sequenced d_x^l dimensional vectors $\boldsymbol{z}_t^l, t \in [T], l \in [L] := \{1, \ldots, L\}$ through L transformer blocks. Each transformer block consists of two sublayers, $i.e.$, a multi-head self-attention sublayer and a position-wise feed-forward sublayer. Each block operation is defined as:

$$\text{Attn}_t^{l,h} = \text{SM}\left\{\frac{1}{\sqrt{d_a}}\left\langle \boldsymbol{W}^{q,l,h}\boldsymbol{z}_t^l, \boldsymbol{W}^{k,l,h}\boldsymbol{z}_{t'}^l \right\rangle\right\}, \tag{1}$$

$$\boldsymbol{f}_{\text{MHSA}}^{l,t} = \sum_{t'=1}^{T}\sum_{h=1}^{H} \text{Attn}_t^{l,h}\boldsymbol{W}^{o,l,h}\boldsymbol{W}^{v,l,h}\boldsymbol{z}_{t'}^l, \tag{2}$$

$$\boldsymbol{f}_{\text{FFN}}^{l,t} = \boldsymbol{W}_{\text{FFN2}}^l \sigma(\boldsymbol{W}_{\text{FFN1}}^l \text{LN}(\boldsymbol{f}_{\text{MHSA}}^{l,t} + \boldsymbol{z}_t^l)), \tag{3}$$

$$\boldsymbol{z}_t^{l+1} = \text{LN}(\boldsymbol{f}_{\text{FFN}}^{l,t}), \tag{4}$$

where SM and LN represent the softmax and layernorm operations, respectively. $\text{Attn}_t^{l,h}$ is the attention score matrix between the vector $\boldsymbol{z}_t^l$ at position $t \in [T]$ and other vectors $\boldsymbol{z}_{t'}^l$ at position $t' \in [T]$ in the l-th transformer block. $\boldsymbol{f}_{\text{MHSA}}^{l,t}$ and $\boldsymbol{f}_{\text{FFN}}^{l,t}$ is the output of the multi-head self-attention sublayer and the point-wise feed-forward sublayer in the l-th transformer block, separately. $\boldsymbol{W}^{q,l,h}, \boldsymbol{W}^{k,l,h}, \boldsymbol{W}^{v,l,h} \in \mathbb{R}^{d_z^l \times d_a^l}$ represent query, key, value weight matrix, respectively. $\boldsymbol{W}^{o,l,h} \in \mathbb{R}^{d_a^l \times d_z^l}$ represents the aggregated weights across H heads. d_a^l is the dimension of the transformer block l, $i.e.$, the width of the entire block. H is the number of heads and the dimension of each head in the transformer block l is $d_a^l = d_z^l/H$. σ represents the ReLU activation function. $\boldsymbol{W}_{\text{FFN1}}^l \in \mathbb{R}^{d_z^l \times d_{\text{in}}^l}$, $\boldsymbol{W}_{\text{FFN2}}^l \in \mathbb{R}^{d_{\text{in}}^l \times d_z^l}$ represent the inner feed-forward weight matrices. d_{in}^l is the inner dimension of the feed-forward sublayer, which is usually set to four times of d_z^l. Unlike traditional transformer network stacking blocks with fixed sizes, in this study, we allow each transformer block l has different d_z^l and d_{in}^l dimensions for enhancing its flexibility across different tasks and datasets.

Problem setting In the context of TAS, the goal of zero-cost proxy is to accurately estimate the ranking of Transformer network's performance at

initialization. Here, we choose the widely used Spearman's ρ (Krishnakumar et al., 2022) and Kendall's τ (Ning et al., 2021) rank correlation metrics to evaluate its predictive ability, which are defined as follows:

$$\tau = \frac{2}{N(N-1)} \sum_{i<j} \operatorname{sgn}\left(u_i - u_j\right) \operatorname{sgn}\left(y_i - y_j\right), \tag{5}$$

$$\rho = \frac{\operatorname{cov}(\mathrm{R}(\boldsymbol{u}), \mathrm{R}(\boldsymbol{y}))}{\sigma_{\mathrm{R}(\boldsymbol{u})} \sigma_{\mathrm{R}(\boldsymbol{y})}}, \tag{6}$$

where N is the number of networks, $\boldsymbol{u} = (u_1, \cdots, u_N)$ and $\boldsymbol{y} = (y_1, \cdots, y_N)$ are scores of the zero-cost proxy and the true accuracy of networks separately, $\mathrm{R}(\boldsymbol{u})$ and $\mathrm{R}(\boldsymbol{y})$ represent the rankings of networks converted from $\boldsymbol{u}$ and $\boldsymbol{y}$, respectively. $\sigma_{\mathrm{R}(\boldsymbol{u})}$ and $\sigma_{\mathrm{R}(\boldsymbol{y})}$ are the standard deviations of $\mathrm{R}(\boldsymbol{u})$ and $\mathrm{R}(\boldsymbol{y})$, separately.

In fact, a good neural network architecture should have good trainability ($i.e.$ how fast a network can convergence via a gradient descent algorithm) and high expressivity ($i.e.$ how complex functions a network can represent) (Chitty-Venkata et al., 2022; Chen et al., 2021c). In Sec. 3.1 and 3.2, we design two theoretically-inspired indicators to reflect the trainability and expressivity of Transformer networks, separately. We then propose NTSR zero-cost proxy that combines the two important indicators to measure the trainability and expressivity of a given Transformer network, and integrate it in an evolutionary search framework called ETAS to find the best-performing Transformer network architecture in section 3.3.

2.1 Trainability of Transformer Network

With the rapid development of deep learning theory on neural networks, the Neural Tangent Kernel (NTK) has emerged as an effective tool for characterizing the training dynamics of infinite wide (Jacot et al., 2021) or finite wide (Novak et al., 2022) deep networks. It solved a classic question of "*how does training of neural network work so well despite being highly nonconvex?*"(Yang, 2020). Lee et al. (2019) have demonstrated under the large width limit and constant NTK assumption, the predictions of a neural network evolves like a linear model throughout gradient descent training.

NTK Formally, assume a deep neural network f parameterized by $\boldsymbol{w}$ has D output dimension. Let $(\mathcal{X}, \mathcal{Y})$ be the training samples, and $\mathcal{L}$ the loss function. The outputs of network are $\boldsymbol{f}(\mathcal{X}, \boldsymbol{w}) \in \mathbb{R}^{ND}$, where N is the number of training samples. During the gradient descent training, the evolution of parameters $\boldsymbol{w}_s$ and output $\boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s)$ at time step s can be expressed as follows:

$$\dot{\boldsymbol{w}}_s = -\eta \nabla_{\boldsymbol{w}} \boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s)^{\intercal} \nabla_{\boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s)} \mathcal{L}, \tag{7}$$

$$\begin{aligned} \dot{\boldsymbol{f}}(\mathcal{X}, \boldsymbol{w}_s) &= \nabla_{\boldsymbol{w}} \boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s) \dot{\boldsymbol{w}}_s \\ &= -\eta \nabla_{\boldsymbol{w}} \boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s) \boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s)^{\intercal} \nabla_{\boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s)} \mathcal{L} \\ &= -\eta \boldsymbol{\Theta}_s(\mathcal{X}, \mathcal{X}) \nabla_{\boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s)} \mathcal{L}, \end{aligned} \tag{8}$$

where $\boldsymbol{\Theta}_s(\mathcal{X}, \mathcal{X}) \in \mathbb{R}^{ND \times ND}$ is the Neural Tangent Kernel (NTK) at time step s, defined as:

$$\boldsymbol{\Theta}_s(\mathcal{X}, \mathcal{X}) = \nabla_{\boldsymbol{w}} \boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s) \nabla_{\boldsymbol{w}} \boldsymbol{f}(\mathcal{X}, \boldsymbol{w}_s)^{\intercal}. \tag{9}$$

The NTK exactly captures the training dynamics of the network. Especially for the infinite wide network, under the mean-squared loss and a constant NTK assumption $i.e.$, $\boldsymbol{\Theta}_s(\mathcal{X}, \mathcal{X}) \equiv \boldsymbol{\Theta}_0(\mathcal{X}, \mathcal{X})$, Equation (8) has a closed-form solution:

$$\boldsymbol{f}\left(\mathcal{X}, \boldsymbol{w}_s\right) = \left(\mathbf{I} - e^{-\eta \boldsymbol{\Theta}_0 s}\right) \boldsymbol{y} + e^{-\eta \boldsymbol{\Theta}_0 s} \boldsymbol{f}\left(\mathcal{X}, \boldsymbol{w}_0\right), \tag{10}$$

where $\boldsymbol{f}\left(\mathcal{X}, \boldsymbol{w}_s\right)$ represents the outputs of the network at time step s, $\mathbf{I}$ is the identity matrix, $\boldsymbol{f}\left(\mathcal{X}, \boldsymbol{w}_0\right)$ is the outputs of the network at initiation, η is the learning rate. This implies that the output of the network is determined by the training samples $(\mathcal{X}, \mathcal{Y})$, the initial weights $\boldsymbol{w}_0$ and initial NTK $\boldsymbol{f}\left(\mathcal{X}, \boldsymbol{w}_0\right)$. Through the NTK at initiation, we can estimate the training convergence of a network. Arora et al. (2019) have demonstrated that the training convergence spped is faster in the direction corresponding to the larger NTK eigenvalues of the network.

NTK of Transformer Assume there exists a batch of M sequences $\mathcal{X} = \{\boldsymbol{x}_{\alpha,1}, \cdots, \boldsymbol{x}_{\alpha,T}\}_{\alpha=1}^{M}$, the output of Transformer network at the L-th block is $\{\boldsymbol{z}_{\alpha,1}^{L}, \cdots, \boldsymbol{z}_{\alpha,T}^{L}\}_{\alpha=1}^{M}$. The Transformer network parameters $\boldsymbol{w}$ consists of $\{\boldsymbol{W}_V, \boldsymbol{W}^{q,l,h}, \boldsymbol{W}^{k,l,h}, \boldsymbol{W}^{v,l,h}, \boldsymbol{W}^{o,l,h}, \boldsymbol{W}_{\text{FFN1}}^{l}, \boldsymbol{W}_{\text{FFN2}}^{l}\}_{l=1}^{L}$. Then, we define the NTK of a Transformer network $\mathbf{K}(\mathcal{X}, \mathcal{X}) \in \mathbb{R}^{Md_z^L \times T \times Md_z^L \times T}$, each element of the 4-dimensional NTK tensor is :

$$\mathbf{K}_{\alpha_1, t_1, \alpha_2, t_2} = \nabla_{\boldsymbol{w}} \boldsymbol{z}_{\alpha_1, t_1}^{L} \left(\nabla_{\boldsymbol{w}} \boldsymbol{z}_{\alpha_2, t_2}^{L}\right)^{\intercal}, \tag{11}$$

where $\alpha_1, \alpha_2 \in \mathcal{X}$ are pair of inputs sampled from the training batch $\mathcal{X}$. $t_1, t_2 \in [T]$ is the token index.

Theorem 2.1 (Yang (2020)). *Let f be a neural network of standard architecture with scalar output and randomly initialized weights $\omega \sim \mathcal{N}\left(0, \sigma_\omega^2\right)$. If it satisfies Condition 1 in Yang (2020) and its nonlinearities have polynomially bounded weak derivatives, then its NTK Θ converges almost surely, over any finite set of inputs, to a deterministic kernel Θ_0, i.e., $\Theta \xrightarrow{a.s.} \Theta_0$ as its widths go to infinity.*

This theorem indicates the NTK of standard Transformer network at initialization $\mathbf{K}(\mathcal{X}, \mathcal{X})$ has a well-defined infinite-width limit. We can use it to predict the training convergence of the network. As shown in Equation (10), we can estimate the training convergence of network through NTK at initialization. If $\mathbf{\Theta}_0$ can be represented through its eigenvectors and corresponding eigenvalues λ_i, then it can be inferred that the eigenvectors of $\mathbf{\Theta}_0$ coincide with those of $e^{-\eta \mathbf{\Theta}_0 t}$, with a transformation of eigenvalues to $e^{-\eta \lambda_i t}$. This observation reveals that the network's convergence rate is intimately connected to the eigenstructures of its NTK $\mathbf{\Theta}_0$. Consequently, a network with a NTK characterized by a greater total sum of eigenvalues, $i.e.$, $\sum_i \lambda_i$, is likely to achieve faster convergence and lower loss.

However, directly computing $\sum_i \lambda_i$ of the 4-dimensional NTK tensor $\mathbf{K}(\mathcal{X}, \mathcal{X})$ of a Transformer network is extremely challenging. This difficulty arises due to the dynamic nature of the network connections in Transformers, where the output at each token index focuses on different parts of the input sequence, as shown in Equation (1), resulting in distinct outputs for each token index. To solve this problem, we take a mean of the NTK tensor $\mathbf{K}(\mathcal{X}, \mathcal{X})$ over the token indexes, which is defined as $\bar{\mathbf{K}}(\mathcal{X}, \mathcal{X}) \in \mathbb{R}^{Md_z^L \times Md_z^L}$, where each element is :

$$\bar{\mathbf{K}}_{\alpha_1,\alpha_2} = \frac{1}{T^2} \sum_{t_1=1}^{T} \sum_{t_2=1}^{T} \nabla_{\boldsymbol{w}} \boldsymbol{z}_{\alpha_1,t_1}^L \left(\nabla_{\boldsymbol{w}} \boldsymbol{z}_{\alpha_2,t_2}^L\right)^\intercal . \tag{12}$$

Through the above operation, the four-dimensional NTK tensor $\mathbf{K}(\mathcal{X}, \mathcal{X})$ is reduced to a standard two-dimensional matrix $\bar{\mathbf{K}}_{\alpha_1,\alpha_2}$. Besides, the matrix is symmetric and positive semi-definite, the sum of eigenvalues is equivalent to the trace of $\bar{\mathbf{K}}(\mathcal{X}, \mathcal{X})$. Calculating the trace of a matrix offers computational efficiency, effectively bypassing the complexities involved in eigenvalue determination. Consequently, we use the trace of $\bar{\mathbf{K}}_{\alpha_1,\alpha_2}$ as an indicator called NTKT for the training convergence of the network, which is defined as:

$$\begin{aligned} \mathrm{NTKT}(\boldsymbol{f}) &= \left\|\bar{\mathbf{K}}(\mathcal{X}, \mathcal{X})\right\|_{\mathrm{tr}} \\ &= \sum_{i=1}^{M} \left\|\bar{\mathbf{K}}_{\alpha_i,\alpha_i}\right\|_{\mathrm{tr}} \\ &= \frac{1}{T^2} \sum_{\alpha_i=1}^{M} \sum_{t_i=1}^{T} \sum_{j=1}^{d_z^L} \left\|\nabla_{\boldsymbol{w}} z_{\alpha_i,t_i,j}^L\right\|_2^2 . \end{aligned} \tag{13}$$

Theorem 2.2. *Assume a standard Transformer net $\boldsymbol{f}$ with randomly initialized weights $\omega \sim \mathcal{N}\left(0, \sigma_\omega^2\right)$. Under vanilla stochastic gradient descent (SGD) optimizer and the first-order Taylor expansion, the mean of output over tokens at time step $s+1$ satisfies: $\bar{\boldsymbol{f}}_{s+1}(\mathcal{X}) - \bar{\boldsymbol{f}}_s(\mathcal{X}) = -\eta_s \bar{\mathbf{K}}(\mathcal{X}, \mathcal{X}) \mathcal{L}'\left(\bar{\boldsymbol{f}}_s(\mathcal{X})\right)$, where η_s is the learning rate at steo s. Then, $\mathrm{NTKT}(\boldsymbol{f})$ is positively correlated with the convergence rate of the network $\boldsymbol{f}$.*

The proof of this theorem can be found in the supplementary material. This theorem shows that NTKT provides a reasonable estimate for the training convergence of the network. In general, a larger NTKT score indicates that the corresponding network will converge faster and learn more efficiently.

2.2 Expressivity of Transformer Network

The success of Transformer networks largely reliance on the core attention mechanism to learn complex dependencies among input sequences for different tasks. Intuitively, the stronger a Transformer network can model dependencies between inputs, the greater its capability to represent input information. **Is there exists a quantitative metric to measure the expressivity of Transformer network at initialization?** To solve this problem, we introduce separation rank as a measure that quantifies the ability of a Transformer network to model dependencies between inputs.

Separation Rank Separation rank is first proposed by Beylkin and Mohlenkamp (2002) for high-dimensional numerical analysis, and then applied to various areas including machine learning (Ghassemi et al., 2019), chemistry (Chinnamsetty et al., 2007), and physics(Validi, 2014). Recently, it has been applied to measure the input dependencies modeled by deep convolutional and recurrent networks (Cohen and Shashua, 2017; Cohen et al.,

2016). Let (A, B) be a partition of the input locations, *i.e.*, A and B are disjoint subsets of input sequence $[T] := \{1, \ldots, T\}$ and $A \cup B = [T]$. The separation rank of a function $y(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ *w.r.t.* partition (A, B) is the minimal number of summands that together sum up to equal y, where each summand is multiplicatively separable *w.r.t.* the partition (A, B), *i.e.*, each summand is equal to a product of two functions — one that takes in only inputs from the subset $\{\boldsymbol{x}_i : i \in A\}$ and another that intakes only inputs from the other subset $\{\boldsymbol{x}_j : j \in B\}$. Formally, the separation rank of $y : \left(\mathbb{R}^{d_x}\right)^T \to \mathbb{R}$ *w.r.t.* the partition (A, B) is defined as:

$$\begin{aligned}\mathrm{sep}_{(A,B)}(y) := \min \Big\{ & R \in \mathbb{N} \cup \{0\} : \\ & \exists g_1^A, \ldots, g_R^A, g_1^B, \ldots, g_R^B : \left(\mathbb{R}^{d_x}\right)^{N/2} \to \mathbb{R}, \\ & y(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \\ & \sum_{r=1}^{R} g_r^A\left(\{\boldsymbol{x}_i : i \in A\}\right) g_r^B\left(\{\boldsymbol{x}_j : j \in B\}\right) \Big\}.\end{aligned} \tag{14}$$

The separation rank quantifies the amount of input inter-dependency induced by the function $y(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ *w.r.t.* partition (A, B). If the separation rank of a function $y(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ *w.r.t.* partition (A, B) is 1, the function $y(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ is multiplicatively separable *w.r.t.* partition (A, B), meaning that $\{\boldsymbol{x}_i : i \in A\}$ and $\{\boldsymbol{x}_j : j \in B\}$ are statistically independent. The larger $\mathrm{sep}_{(A,B)}(y)$ is, the more it models inter-dependency between $\{\boldsymbol{x}_i : i \in A\}$ and $\{\boldsymbol{x}_j : j \in B\}$. In other words, it means that the function $y(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ can learn higher correlation between $\{\boldsymbol{x}_i : i \in A\}$ and $\{\boldsymbol{x}_j : j \in B\}$.

Separation rank for Transformer As shown in Equation (14), the separation rank directly reflect the success behind the core attention mechanism in Transformer network. The self-attention layer dynamically learns to correlate any inter-dependent subsets of the inputs (Levine et al., 2020). When a transformer network learn more dependencies between inputs, the separation rank induced by this network will have a higher value. Levine et al. (2020) have demonstrated a tight bound on the separation rank of Transformer network with respect to the dependence on depth and width. Further, they leverage this bound to determine the optimal depth-to-width for a given Transformer network size.

Theorem 2.3 (Levine et al. (2020))**.** *Let $y^{i,t,L,d_x,H}$ be the i-th scalar output at the t token index of a standard depth-L Transformer net $\boldsymbol{f}$ with dimension d_x and the number of heads$H$ per layer. Then, its separation rank w.r.t. balanced partitions, which obey $A \cup B = [T], |A| = |B| = T/2$, is invariant to the identity of the partition, i.e., $A \cup B = [T], \tilde{A} \cup \tilde{B} = [T]$, s.t. $|A|, |B|, |\tilde{A}|, |\tilde{B}| = T/2$:* $\mathrm{sep}_{(A,B)}(y^{i,t,L,d_x,H}) = \mathrm{sep}_{(\tilde{A},\tilde{B})}(y^{i,t,L,d_x,H})$.

This theorem reveals that the separation rank induced by the standard Transformer network keeps consistent under different balanced partition of inputs. Next, we will omit the specification of any balanced partition of inputs, denoting as $\mathrm{sep}(y^{i,t,L,d_x,H})$. Wies et al. (2021) further demonstrate the existence of an embedding rank bottleneck that limits the expressivity of the Transformer network. They showed that $\log \mathrm{sep}(y^{i,t,L,d_x,H}) = \tilde{O}\left(L \cdot \min\{r, d_x\}\right)$, where r is the embedding rank defined as $r = \mathrm{rank}(\boldsymbol{W}_V)$.

However, the current study mainly focus on the separation rank of the Transformer network with the same dimension d_x per layer. In this study, we extend the separation rank to the Transformer network with each block l has different d_z^l and d_{in}^l dimensions. This makes it more difficult to analyze. To solve this problem, we do some relaxations followed by Wies et al. (2021) and Levine et al. (2020). We put all the position-wise feed-forward layer at the end since the feed-forward operation does not mix different locations and learn the dependencies between inputs. We then remove all the normalization layers, and omit the ReLU and softmax nonlinearities. The reasons to do it can be referred to Wies et al. (2021) and Levine et al. (2020). Though these relaxations are shown to weaken the overall network performance, they are much less pertinent to the ability of the core self-attention mechanism to model dependencies between different places at the input. Consequently, the multi-head attention operation of each block l in Equation (2) can be simplified as:

$$\boldsymbol{f}_t^{l+1} = \sum_{t'=1}^{T} \sum_{h=1}^{H} \left\langle \boldsymbol{W}^{q,l,h} \boldsymbol{f}_t^l, \boldsymbol{W}^{k,l,h} \boldsymbol{f}_{t'}^l \right\rangle \boldsymbol{W}^{o,l,h} \boldsymbol{W}^{v,l,h} \boldsymbol{f}_{t'}^l. \tag{15}$$

By forward propagating the above operations layer-by-layer, We can obtain the upper bound on the

separation rank of the Transformer network with different dimensions d_z^l per layer.

Theorem 2.4. *Let f_t^L be the output at the t token index of a standard depth-L Transformer net $\boldsymbol{f}$ with dimension d_z^l and the number of heads H per layer. Then, its separation rank $w.r.t.$ balanced partitions, which satisfies* $\log(\text{sep}(f_t^L)) \leq \log(\sum_{l=1}^{L} d_z^l) + \log(\sum_{l=1}^{L} \frac{1-d_z^{lT+1}}{1-d_z^l})$.

The proof of this theorem is shown in appendix. This theorem shows that the separation rank of the Transformer net $\boldsymbol{f}$ can be upper bounded by a constant, which is related to the depth of the network L, the number of tokens T and the sum of dimensions $\sum_{l=1}^{L} d_z^l$. Thus, we use this bound to measure the amount of dependencies modeled by the Transformer net. We name this measure as SEPT, which is defined as:

$$\text{SEPT}(\boldsymbol{f}) = \log(\sum_{l=1}^{L} d_z^l) + \log(\sum_{l=1}^{L} \frac{1-d_z^{lT+1}}{1-d_z^l}) \tag{16}$$

SEPT measures the Transformer network's ability to model dependencies between different places of the input. In general, a larger SEPT score indicates that the corresponding Transformer network has a stronger expressivity about different dependencies between inputs.

2.3 Zero-Shot Transformer Architecture Search via Network Trainability and Expressivity

NTSR How to combine the two NTKT and SEPT theoretically-inspired indicators to find the best-performing Transformer network at initialization? To answer this question, we propose a new zero-cost proxy called NTSR to make a trade-off between the network trainability and expressivity. Note that the magnitudes of NTKT and SEPT score may differ much and the ranges of them are agnostic before computing on the whole search space. Hence, we can't directly normalize NTKT and SEPT scores before searching the network. To avoid this problem, instead of using the numerical values of NTKT and SEPT, we combine the relative ranking of NTKT and SEPT by comparing the sampled set of architectures, which is defined as:

$$\text{NTSR}(\boldsymbol{f}) = \frac{1}{2}(\text{R}(\text{NTKT}(\boldsymbol{f})) + \text{R}(\text{SEPT}(\boldsymbol{f}))), \tag{17}$$

where $\text{R}(\text{NTKT}(\boldsymbol{f}))$ and $\text{R}(\text{SEPT}(\boldsymbol{f}))$ represent the relative ranking of the network $\boldsymbol{f}$ converted from the $\text{NTKT}(\boldsymbol{f})$ and $\text{SEPT}(\boldsymbol{f})$ among the sampled set of architectures. In general, a higher rank of NTSR indicates that the network has a better trainability and expressivity.

ETAS As described above, our proposed NTSR zero-cost proxy provides a reasonable estimation of the network's performance. **The next major question is how to construct an efficient NAS framework on top of it?** To solve this problem, we integrate it into a popular regularized evolution framework called ETAS. We first randomly sample N_0 network architectures from the search space and then pick the top n_0 networks according the ranking of NTSR as the initial parent population to warmup the whore algorithm. Through warmup, we can find a better local initial population, thus potentially expediting the discovery of the optimal network. Note that the sample size N_0 is much larger than the number of networks we can afford to train. After that, we generate N_m candidate architectures by mutating the parent architectures at each iteration m and then select top n_m architectures from the current N_m candidate architectures according the ranking of NTSR. We then evaluate the selected architectures and update the parent population. The algorithm details of our proposed ETAS framework is given in the Appendix.

3 Experiments and Discussion

In this section, we choose to search the most popular Vision Transformer (ViT) and GPT-2 architectures in computer vision and natural language processing domains separately. We compare our proposed NTSR zero-cost proxy with several zero-cost proxies for Transformer include DSS(Zhou et al., 2022), Length Distortion (LD)(Chen et al., 2022) and conventional zero-cost proxies for CNN like snip, grad_norm, NASWOT, zico and MeCo. Since Ning et al. (2021) have demonstrated the number of flops and parameters serve as the competitive proxies for the network performance than most zero-cost proxies under the CNN search space, we also add them for comparison. To make an ablation study on our proposed NTSR proxy, we add NTKT and SEPT alone as zero-cost proxies for comparison.

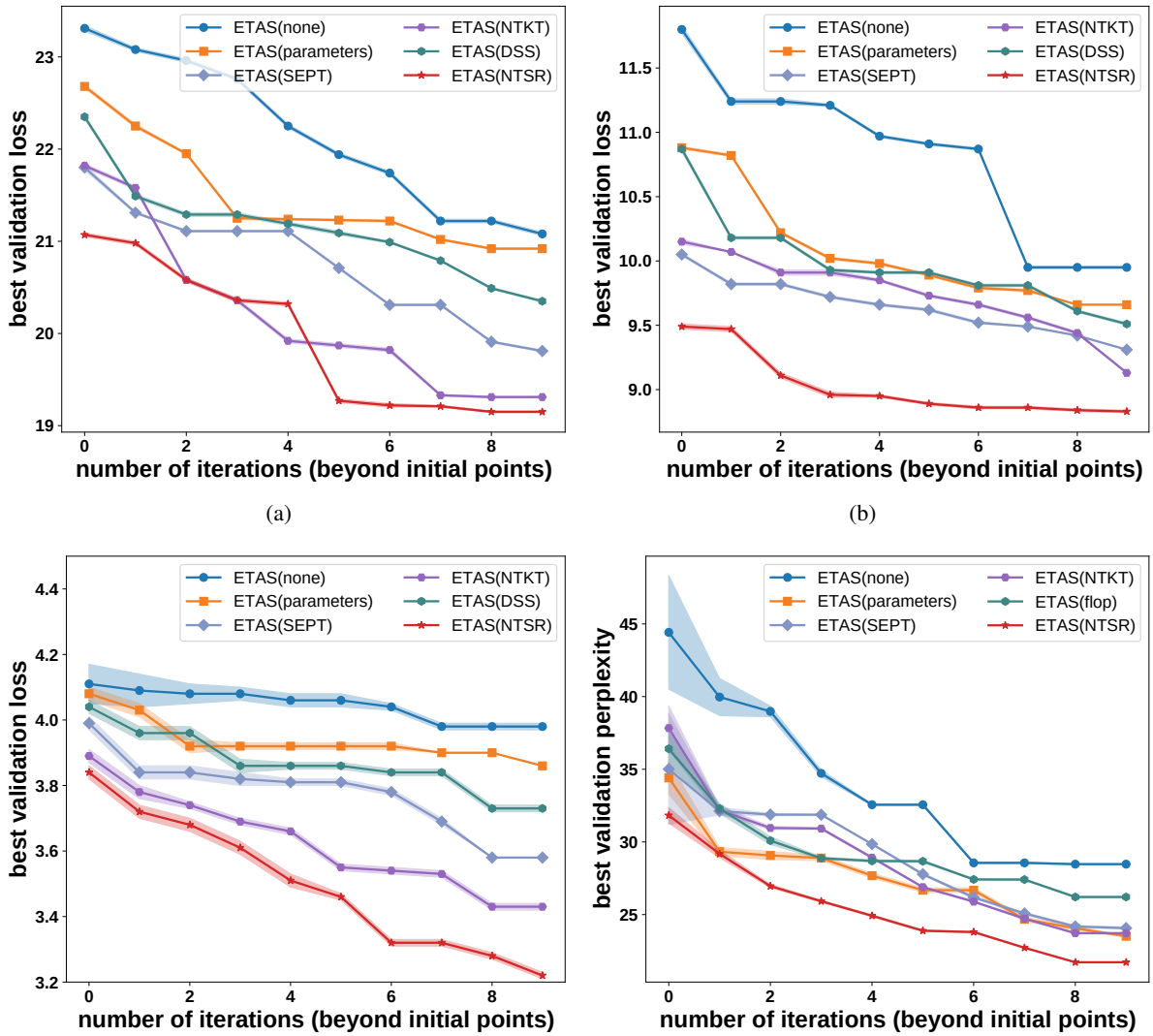

Figure 1: The best-found valid loss over the number of iterations (beyond initial points) of various methods on (a) the ImageNet1k dataset of the Autoformer tiny search space, (b) the ImageNet1k dataset of the Autoformer small search space, (c) the ImageNet1k dataset of the Autoformer base search space, (d) the Wikitex103 dataset of the GPT-2 search space.

Zero-cost	tiny		small		base	
proxies	KT	SPR	KT	SPR	KT	SPR
params	0.57±0.00	0.76±0.00	0.64±0.00	0.81±0.00	0.60±0.00	0.81±0.00
flops	0.49±0.00	0.68±0.00	0.58±0.00	0.78±0.00	0.52±0.00	0.74±0.00
snip	0.56±0.03	0.74±0.03	0.59±0.01	0.76±0.03	0.58±0.06	0.79±0.08
grad_norm	0.33±0.05	0.51±0.04	0.54±0.03	0.72±0.03	0.58±0.04	0.77±0.06
NASWOT	0.54±0.02	0.71±0.02	0.57±0.03	0.75±0.04	0.50±0.02	0.71±0.03
zico	0.48±0.03	0.65±0.03	0.43±0.02	0.62±0.03	0.43±0.04	0.62±0.05
MeCo	0.46±0.03	0.68±0.02	0.47±0.01	0.66±0.03	0.47±0.01	0.66±0.02
DSS	0.66±0.01	0.80±0.02	0.65±0.02	0.82±0.02	0.64±0.03	0.84±0.05
LD	0.52±0.02	0.69±0.02	0.60±0.03	0.81±0.04	0.59±0.03	0.80±0.03
SEPT	0.71±0.00	0.89±0.00	0.72±0.00	0.89±0.00	0.69±0.00	0.87±0.00
NTKT	0.70±0.02	0.87±0.02	0.70±0.03	0.86±0.02	0.72±0.02	0.90±0.02
NTSR	**0.74±0.01**	**0.90±0.02**	**0.74±0.02**	**0.91±0.02**	**0.75±0.01**	**0.92±0.01**

Table 1: The ranking correlation of different zero-cost proxies on the AutoFormer search space over 5 independent runs with different random seeds, where KT and SPR represent the Kendall's τ and Spearman's ρ rank correlation metrics separately.

3.1 Searching for ViT

To make a fair comparison, we employ the same of search space of AutoFormer(Chen et al., 2021a), which searchs the key components of the ViT architecture include embedding dimension, Q-K-V dimension, number of heads, MLP ratio, and network depth. It contains more than 1.7×10^{16} candidate architectures covering three common ranges of model size $i.e.$, tiny (4-9 M), small (14-34 M), and base (42-75 M). We randomly sample 1000 networks for each network setup since it is computationally infeasible to evaluate all networks of the large AutoFormer search space. After that, we compute the rank correlation between these zero-cost proxy scores and true accuracy of these sampled networks on ImageNet-1K dataset. Table Table 1 shows the performance of various zero-cost proxies across three setups $i.e$, tiny, small and base. To demonstrate the effect of NTSR proxy in on accelerating TAS, we integrate it into the ETAS framework. Here, we choose to incorporate top-5 zero-cost proxies on the Table Table 1 into our proposed ETAS framework for evaluating their performance on impact on speeding up the regularized evolution algorithm in discovering the best ViT architecture. These methods' implementation details and experimental settings are summarized in the Appendix Appendix A.4.1. Figures 1(a) to 1(c) shows the best-found valid loss over the number of iterations of various methods. The test loss of various methods on the test set of ImageNet-1k dataset are showed in Table 2. The results reveal that our proposed NTSR zero-cost proxy can achieve the highest rank correlation across three setpus. It can find a Transformer network architecture with lower validation loss using fewer iterations.

3.2 Searching for GPT-2

The GPT-2 search space used in this section is largely based on the design of LTS (Javaheripi et al., 2022) search space on the two WikiText-103 (Merity et al., 2017) and One Billion Word (LM1B) (Chelba et al., 2014) datasets. It consists the number of layers (n_layer $\in \{2, \ldots, 16\}|1$), number of attention heads (n_head $\in \{2, 4, 8\}$), decoder output dimension for each layer (d_model $\in \{128, \ldots, 1024\}|64$), inner dimension of the feed forward network for each layer (d_inner $\in \{256, \ldots, 4096\}|64$), and embedding dimension (d_embed $\in 128, 256, 512$). Unlike LTS search space, we fix the adaptive input embedding factor to $k = 4$ to approximate the standard Transformer network with non-adaptive input embedding. We add a constraint that the d_inner must large than 2d_model to avoid the training collapse of the network. For each dataset, we randomly sample 500 networks and compute zero-cost proxies for each network. We train each GPT-2 model from scratch to obtain its true validation perplexity. After that, we compute the rank correlation between these zero-cost proxy scores and true validation perplexity of these sampled networks on WikiText-103 and One Billion Word datasets. Table 3 shows the performance of various zero-cost proxies. To demonstrate the effect of NTSR proxy in on speeding up TAS, we integrate it into the ETAS framework. Here, we choose to incorporate top-5 zero-cost proxies on the Table Table 3 into our proposed ETAS framework for evaluating their performance on impact on speeding up the regularized evolution algorithm in discovering the best GPT-2 architecture in the WikiText-103 dataset. These methods' implementation details and experimental settings are summarized in the Appendix Section 3.2. The validation perplexity of various methods on the GPT-2 search space are showed in Table 4. The results reveal that our proposed NTSR zero-cost proxy have a higher rank correlation with the true validation perplexity . It can significantly speed up finding a better performance Transformer Network architectures with fewer iterations. In contrast, the zero-cost proxies for Transformer like DSS and LD perform worse on this GPT-2 search space. This may be because they are specially designed for the ViT transformer architecture and thus generalize worse on the GPT Transformer network.

4 Limitation and Conclusion

In this study, we propose a novel zero-cost proxy for Transformer network called NTSR to evaluate the performance of Transformer at initialization. Compared to other popular zero-cost proxies for Transformer networks, our proposed proxy is designed based the deep neural network learning theory in terms of the trainability and expressivity of Transformer networks. However, our proposed method is limited to decoder-only Transformer architecture. In future, we hope our proposed proxy can extend to other complex types of Transformer Networks like Bert (Devlin et al., 2019) and T5 (Raffel et al., 2020) .

References

Mohamed S. Abdelfattah, Abhinav Mehrotra, Lukasz Dudziak, and Nicholas Donald Lane. 2021. Zero-cost proxies for lightweight NAS. In *ICLR*.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. 2019. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *ICML*, volume 97, pages 322–332.

Gregory Beylkin and Martin J Mohlenkamp. 2002. Numerical operator calculus in higher dimensions. In *PNAS*, volume 99, pages 10246–10251.

Ciprian Chelba, Tomás Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*, pages 2635–2639. ISCA.

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. 2021a. Autoformer: Searching transformers for visual recognition. In *ICCV*, pages 12250–12260. IEEE.

Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. 2021b. Only train once: A one-shot neural network training and pruning framework. In *NeurIPS*, pages 19637–19651.

Wuyang Chen, Xinyu Gong, and Zhangyang Wang. 2021c. Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective. In *ICLR*.

Wuyang Chen, Wei Huang, Xianzhi Du, Xiaodan Song, Zhangyang Wang, and Denny Zhou. 2022. Auto-scaling vision transformers without training. In *ICLR*. OpenReview.net.

Sambasiva Rao Chinnamsetty, Mike Espig, Boris N Khoromskij, Wolfgang Hackbusch, and Heinz-Jürgen Flad. 2007. Tensor product approximation with optimal rank in quantum chemistry. *The Journal of Chemical Physics*, 127(8).

Krishna Teja Chitty-Venkata, Murali Emani, Venkatram Vishwanath, and Arun K. Somani. 2022. Neural architecture search for transformers: A survey. *IEEE Access*, 10:108374–108412.

Nadav Cohen, Or Sharir, and Amnon Shashua. 2016. On the expressive power of deep learning: A tensor analysis. In *COLT*, volume 49, pages 698–728. JMLR.org.

Nadav Cohen and Amnon Shashua. 2017. Inductive bias of deep convolutional networks through pooling geometry. In *ICLR*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Mohsen Ghassemi, Zahra Shakeri, Waheed U. Bajwa, and Anand D. Sarwate. 2019. Sample complexity bounds for low-separation-rank dictionary learning. In *ISIT*, pages 2294–2298. IEEE.

Arthur Jacot, Franck Gabriel, and Clément Hongler. 2021. Neural tangent kernel: convergence and generalization in neural networks. In *STOC*, page 6. ACM.

Mojan Javaheripi, Gustavo de Rosa, Subhabrata Mukherjee, Shital Shah, Tomasz 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*.

Tangyu Jiang, Haodi Wang, and Rongfang Bie. 2023. Meco: Zero-shot NAS with one data and single forward pass via minimum eigenvalue of correlation. 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. In *NeurIPS*.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. 2019. Wide neural networks of any depth evolve as linear models under gradient descent. In *NeurIPS*, pages 8570–8581.

Yoav Levine, Noam Wies, Or Sharir, Hofit Bata, and Amnon Shashua. 2020. Limits to depth efficiencies of self-attention. In *NeurIPS*.

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

Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. 2022. Efficientformer: Vision transformers at mobilenet speed. In *NeurIPS*.

Joe Mellor, Jack Turner, Amos J. Storkey, and Elliot J. Crowley. 2021. Neural architecture search without training. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 7588–7598. PMLR.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *ICLR*. OpenReview.net.

Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. 2021. Evaluating efficient performance estimators of neural architectures. In *NeurIPS*, pages 12265–12277.

Roman Novak, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. 2022. Fast finite width neural tangent kernel. In *ICML*, volume 162, pages 17018–17044.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21:140:1–140:67.

Yao Shu, Shaofeng Cai, Zhongxiang Dai, Beng Chin Ooi, and Bryan Kian Hsiang Low. 2022. NASI: label- and data-agnostic neural architecture search at initialization. In *ICLR*.

David R. So, Quoc V. Le, and Chen Liang. 2019. The evolved transformer. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5877–5886. PMLR.

David R. So, Wojciech Manke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V. Le. 2021. Searching for efficient transformers for language modeling. In *NeurIPS*, pages 6010–6022.

Xiu Su, Shan You, Jiyang Xie, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, Xiao-Gang Wang, and Chang Xu. 2022. Vitas: Vision transformer architecture search. In *ECCV (21)*, volume 13681 of *Lecture Notes in Computer Science*, pages 139–157. Springer.

Renbo Tu, Nicholas Roberts, Mikhail Khodak, Junhong Shen, Frederic Sala, and Ameet Talwalkar. 2022. Nas-bench-360: Benchmarking neural architecture search on diverse tasks. In *NeurIPS*.

AbdoulAhad Validi. 2014. Low-rank separated representation surrogates of high-dimensional stochastic functions: Application in bayesian inference. *Journal of Computational Physics*, 260:37–53.

Noam Wies, Yoav Levine, Daniel Jannai, and Amnon Shashua. 2021. Which transformer architecture fits my data? A vocabulary bottleneck in self-attention. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 11170–11181.

Greg Yang. 2020. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training BERT in 76 minutes. In *ICLR*. OpenReview.net.

Arber Zela, Julien Niklas Siems, Lucas Zimmer, Jovita Lukasik, Margret Keuper, and Frank Hutter. 2022. Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks. In *ICLR*. OpenReview.net.

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 10884–10893. IEEE.

Wei Zhu. 2021. Autorc: Improving BERT based relation classification models via architecture search. In *ACL*, pages 33–43. Association for Computational Linguistics.

Wei Zhu, Xiaoling Wang, Yuan Ni, and Guotong Xie. 2021. Autotrans: Automating transformer design via reinforced architecture search. In *NLPCC*, volume 13028 of *Lecture Notes in Computer Science*, pages 169–182. Springer.

A Appendix

A.1 Related Work

Transformer Architecture Search The goal of Transformer Architecture Search (TAS) aims at searching the best-performing Transformer configurations in an automated way for a given task. Recently, various TAS methods have been proposed. One most popular method is evolution-based Transformer architecture search (So et al., 2019, 2021; Su et al., 2022). They use evolutionary search algorithm to find optimal Transformer architecture at a given target budget. Another classic method is reinforcement learning (Zhu et al., 2021; Zhu, 2021), which uses a controller to sample high-quality Transformer architectures and updates the controller using the network's performance as a reward. However, current evolutionary search and reinforcement learning approaches need collecting a substantial number of network samples to train, which is costly and time-consuming. To reduce the search cost of TAS, One-shot methods (Chen et al., 2021a,b; **?**) have been proposed. They only trained a huge supernetwork and obtain the performance of the sampled subnetwork through weight sharing. Nevertheless, training a huge supernetwork is non-trival and memory consumption of Transformer Supernetwork increases with hidden size and runs out even with small values (Chitty-Venkata et al., 2022). Predictor-based methods train a surrogate model using a certain number of Transformer architecture-accuracy pairs and then use it to estimate the performance of unseen networks. Overall, the current popular TAS methods still need fully training a large number of architectures to find the optimal network, which demand a high computational cost. This highlights the great potential of developing zero-shot methods to replace the expensive training process in NAS with zero-cost proxies.

Zero-Cost Proxies In recently years, various type of zero-cost proxies have been proposed to rank the performance of networks at the initialization stage. One type consists of parametric saliency-based zero-cost proxies. They score the whole network by summing the changes in the saliency metric when a specific parameter of the network is removed. Abdelfattah et al. (2021) adopt several pruning-at-initialization metrics include snip, grasp, synflow, and fisher as parameter-level saliency-based zero-cost proxies for the network performance with a minibatch of data at initialization. They show that these zero-cost proxies achieve a high correlation with the true accuracy of networks in both both NAS-Bench-101 and NAS-Bench-201 benchmarks. Another type is Network expressivity-based zero-cost proxies. They estimate the network performance by the expressivity of the network(Tu et al., 2022). Mellor et al. (2021) develop a heuristic NASWOT metic that utilizes the correlation of network activations between different datas at initialization. Meanwhile, Chen et al. (2021c) select the number of linear activated regions represented by the network to measure the expressivity of a network. Additionally, there exits a type of zero-cost proxies inspired by the deep learning theory. Chen et al. (2021c) choose the condnum of Neural Tangent Kernel (NTK) to measure the trainability of networks and show it is negatively correlated with network performance. Later, Shu et al. (2022) develop a label-agnostic and data-agnostic NASI metric that leverage the trace of NTK as an indicative for network performance. More recently, Li et al. (2023) found that the mean value and standard deviation of gradients across different samples affects the training convergence of networks. Based on the generalization theory of deep neural networks, they propose a theoretically-inspired zero-cost proxy called ZiCo that considers both the mean value and standard deviation of gradients in each layer of the network. Meanwhile, Jiang et al. (2023) use the minimum eigenvalue of the pearson correlation matrix upon each layer of the feature maps to indicate the convergence of the network.

However, most of the existing zero-cost proxies are specially designed for CNN networks, whose internal architecture is distinctly different from the Transformer network. Zhou et al. (2022) have showed that the current zero cost proxies for CNN can not generalize well to the Transformer search space. To adapt well for the Transformer network, we develop a novel zero-cost proxy called NTSR that combines two theoretically-inspired indicators for measuring the trainability and expressivity of the Transformer network.

A.2 Proof

Proof of Theorem 2.2. Under vanilla stochastic gradient descent (SGD) optimizer, the weight parameter is updated as follows:

$$\boldsymbol{w}_{s+1} = \boldsymbol{w}_s - \eta_s \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}\bigg|_{\boldsymbol{w}=\boldsymbol{w}_s} \tag{18}$$

Then, under the first-order Taylor expansion, the mean of output over tokens at time step $s+1$ satisfies:

$$\begin{aligned}\bar{\boldsymbol{f}}_{s+1}(\mathcal{X}) &= \bar{\boldsymbol{f}}_s(\mathcal{X}) - \eta_s\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}\frac{\partial \bar{\boldsymbol{f}}_s(\mathcal{X})}{\partial \boldsymbol{w}}\right) \\ &= \bar{\boldsymbol{f}}_s(\mathcal{X}) - \eta_s \mathcal{L}'(\bar{\boldsymbol{f}}_s)\left(\frac{\partial \bar{\boldsymbol{f}}_s(\mathcal{X})}{\partial \boldsymbol{w}}\right)\left(\frac{\partial \bar{\boldsymbol{f}}_s(\mathcal{X})}{\partial \boldsymbol{w}}\right)^{\mathsf{T}} \\ &= \bar{\boldsymbol{f}}_s(\mathcal{X}) - \eta_s \mathcal{L}'(\bar{\boldsymbol{f}}_s)\bar{\mathbf{K}}(\mathcal{X},\mathcal{X}). \end{aligned} \tag{19}$$

This reduces to a kernel gradient descent method. Thus, the convergence rate of the network is determined by the eigenstructure of the mean NTK $\bar{\mathbf{K}}(\mathcal{X},\mathcal{X})$. If it can be diagonalized by eigenfunctions with corresponding eigenvalues λ_i, the sum of eigenvalues $\sum \lambda_i$ provides a upper bound for the convergence of the network. Since the mean NTK $\bar{\mathbf{K}}(\mathcal{X},\mathcal{X})$ matrix is symmetric and symmetric. The trace of $\bar{\mathbf{K}}(\mathcal{X},\mathcal{X})$ $i.e.$, NTKT is equal to the sum of eigenvalues $\sum \lambda_i$. Therefore, a larger NTKT score of the Transformer network indicates it converge faster. □

Proof of Theorem 2.4. According to the Equation (15), the output of each block l can be expressed as follows:

$$\boldsymbol{f}_t^{l+1} = \sum_{t'=1}^{T}\sum_{h=1}^{H}\left\langle \boldsymbol{W}^{q,l,h}\boldsymbol{f}_t^l, \boldsymbol{W}^{k,l,h}\boldsymbol{f}_{t'}^l \right\rangle \boldsymbol{W}^{o,l,h}\boldsymbol{W}^{v,l,h}\boldsymbol{f}_{t'}^l. \tag{20}$$

Let $\boldsymbol{M}^{l,h} = \boldsymbol{W}^{q,l,h}\boldsymbol{f}_t^l \boldsymbol{f}_t^{l^{\mathrm{T}}} \boldsymbol{W}^{k,l,h^{\mathrm{T}}}$, the Equation (15) can be expressed as:

$$\boldsymbol{f}_t^{l+1} = \sum_{t'=1}^{T}\sum_{h=1}^{H} \boldsymbol{W}^{o,l,h}\boldsymbol{M}^{l,h}\boldsymbol{W}^{v,l,h}\boldsymbol{f}_{t'}^l. \tag{21}$$

Assume there exits a balanced partition of the input locations (A, B), $i.e.$, each token index $P_t \in$

Method	tiny			small			base		
	top1(%)	params(M)	flops(B)	top1(%)	params(M)	flops(B)	top1(%)	params(M)	flops(B)
ViT/16	74.5	5.7	1.2	78.8	22.1	4.7	77.9	86	55.4
DeiT	72.2	5.7	1.2	79.9	22.1	4.7	81.8	86	17.5
CONViT	73.1	6.0	1.0	81.3	27.0	5.4	82.4	86	17.0
Autoformer	74.7	5.9	1.3	81.7	22.9	4.9	82.4	54	11.0
PRENAS	77.1	5.9	1.4	81.8	22.9	5.1	82.6	54	11.0
ETAS(none)	73.6	5.8	1.2	80.2	23.7	5.2	80.2	51	10.4
ETAS(parameters)	74.6	5.9	1.2	81.6	29.2	6.1	81.9	86	18.0
ETAS(SEPT)	77.2	5.9	1.2	82.3	27.6	5.8	82.4	54	11.2
ETAS(NTKT)	76.9	5.8	1.3	82.1	24.5	5.2	82.9	84	17.6
ETAS(DSS)	75.3	5.9	1.4	81.7	22.6	4.9	82.1	52	12.0
ETAS(NTSR)	**78.1**	5.9	1.4	**82.6**	28	5.9	**83.4**	82	15.5

Table 2: The test loss of various methods on the test set of ImageNet-1k dataset, where ETAS(none) represents the ETAS without zero-cost proxy.

Zero-cost proxies	WikiText103		LM1B	
	KT	SPR	KT	SPR
params	0.79±0.00	0.94±0.00	0.77±0.00	0.92±0.00
flops	0.65±0.00	0.80±0.00	0.51±0.00	0.72±0.00
snip	0.40±0.02	0.57±0.04	0.48±0.02	0.66±0.03
grad_norm	0.15±0.02	0.23±0.03	0.06±0.01	0.14±0.01
NASWOT	0.24±0.02	0.37±0.02	0.32±0.01	0.48±0.02
zico	0.18±0.01	0.36±0.02	0.15±0.01	0.26±0.02
MeCo	0.49±0.02	0.65±0.02	0.42±0.01	0.61±0.01
DSS	0.26±0.02	0.35±0.02	0.24±0.01	0.32±0.01
LD	0.48±0.03	0.64±0.03	0.37±0.01	0.50±0.02
SEPT	0.78±0.00	0.94±0.00	0.79±0.00	0.95±0.00
NTKT	0.80±0.02	0.96±0.02	0.78±0.01	0.92±0.02
NTSR	0.81±0.01	0.98±0.01	0.80±0.01	0.96±0.01

Table 3: The ranking correlation of different zero-cost proxies on the WikiText103 and LM1B dataset of the gpt-2 search space over 5 independent runs with different random seeds, where KT and SPR represent the Kendall's τ and Spearman's ρ rank correlation metrics separately.

Methods	valid perplexity (PPL)	params (M)	flop (B)
GPT-2 (117M)	29.41	117	-
GPT-2 (345M)	22.76	345	-
ETAS(none)	28.47±0.04	211	14.1
ETAS(parameters)	23.51±0.05	223	14.9
ETAS(flop)	26.21±0.05	213	14.5
ETAS (SEPT)	24.07±0.03	201	13.9
ETAS (NTKT)	23.72±0.04	182	13.6
ETAS(NTSR)	21.72±0.03	189	13.8

Table 4: The validation perplexity of different methods on the WikiText103 on the gpt-2 search space over 5 independent runs with different random seeds.

$\{A, B\}$, A and B are disjoint subsets of input sequence $[T] := \{1, \dots, T\}$ and $A \cup B = [T]$. The matrix multiplication in the $\boldsymbol{M}^{l,h}$ can be divided the sum over inputs indexed by A and B:

$$\begin{aligned}\boldsymbol{M}^{l,h}_{r_1,r_2} &= \sum_{t=1}^{T} [\boldsymbol{W}^{q,l,h} \boldsymbol{f}_t^l]_{r_1,t} [\boldsymbol{f}_t^{l^{\mathrm{T}}} \boldsymbol{W}^{k,l,h^{\mathrm{T}}}]_{t,r_2} \\ &= \sum_{t\in A} [\boldsymbol{W}^{q,l,h} \boldsymbol{f}_t^l]_{r_1,t} [\boldsymbol{f}_t^{l^{\mathrm{T}}} \boldsymbol{W}^{k,l,h^{\mathrm{T}}}]_{t,r_2} \\ &\quad + \sum_{t\in B} [\boldsymbol{W}^{q,l,h} \boldsymbol{f}_t^l]_{r_1,t} [\boldsymbol{f}_t^{l^{\mathrm{T}}} \boldsymbol{W}^{k,l,h^{\mathrm{T}}}]_{t,r_2}.\end{aligned} \tag{22}$$

Then the Equation (21) can be reformed as:

$$\begin{aligned}\boldsymbol{f}_t^{l+1} &= \sum_{h\in[H]} \sum_{r_1,\dots,r_T=1}^{d_a^l} \sum_{P_1,\dots,P_T} \boldsymbol{W}^{o,l,h} \boldsymbol{W}^{v,l,h} \boldsymbol{f}_t^l \\ &\quad \left(\prod_{t=1}^{T} [\boldsymbol{W}^{q,l,h} \boldsymbol{f}_t^l]_{r_t,t} [\boldsymbol{f}_t^{l^{\mathrm{T}}} \boldsymbol{W}^{k,l,h^{\mathrm{T}}}]_{t,r_t} \right) \\ &= \sum_{h\in[H]} \sum_{r_1,\dots,r_T=1}^{d_a^l} \sum_{P_1,\dots,P_T} \boldsymbol{W}^{o,l,h} \boldsymbol{W}^{v,l,h} \boldsymbol{f}_t^l \\ &\quad \left(\sum_{t\in A} [\boldsymbol{W}^{q,l,h} \boldsymbol{f}_t^l]_{r_t,t} [\boldsymbol{f}_t^{l^{\mathrm{T}}} \boldsymbol{W}^{k,l,h^{\mathrm{T}}}]_{t,r_t} \right. \\ &\quad \left. \sum_{t\in B} [\boldsymbol{W}^{q,l,h} \boldsymbol{f}_t^l]_{r_t,t} [\boldsymbol{f}_t^{l^{\mathrm{T}}} \boldsymbol{W}^{k,l,h^{\mathrm{T}}}]_{t,r_t} \right) \\ &\leq \sum_{t=1}^{T} \sum_{r_1,\dots,r_t=1}^{d_a^l} H = \sum_{t=1}^{T} (H d_a^l)^t \\ &= d_z^l \frac{1 - d_z^{l^{T+1}}}{1 - d_z^l} \quad .\end{aligned} \tag{23}$$

The separation rank at the l block satisfies:

$$\log(\frac{\text{sep}(f_t^{l+1})}{\text{sep}(f_t^l)}) \leq \log(d_z^l) + \log(\frac{1-d_z^{l^{T+1}}}{1-d_z^l}) \tag{24}$$

Then, the separation rank at the last block L satisfies:

$$\log(\text{sep}(f_t^L)) \leq \log(\sum_{l=1}^{L} d_z^l) + \log(\sum_{l=1}^{L} \frac{1-d_z^{l^{T+1}}}{1-d_z^l}) \tag{25}$$

□

A.3 Algorithm Details of ETAS

To integrate our proposed NTSR zero-cost proxy into the ETAS framework, we first randomly sample N_0 networks from the Transformer search space and compute the relative rankings of N_0 networks by NTSR. After that, we choose the top n_0 network structures with NTSR as the parent population and evaluate these networks to obtain their true performance. We add the network-peformance pairs into the observed set. For each iteration m, we generate a pool of N_m candidate architectures by mutating the current parent population and select the top n_m architectures from N_m in terms of NTSR. We then evaluate the n_m architectures and add their network-peformance pairs into the current observed set. The parent population is updated by selecting the top n_0 networks from the current observed set. At last, we select the top-performance architecture from the observed set. This process continues until the maximum number of iterations M is reached or the current best value that has not improved for five successive iterations. The detailed algorithm of our proposed ETAS framework is summarized in the Algorithm 1. Note that our proposed ETAS framework can also incorporate other zero-cost proxies.

Algorithm 1: ETAS

Input: Total number of iterations M, search space $\mathcal{A}$, parent population = $\oslash$, observed population = $\oslash$

Output: The best-performing architecture $\boldsymbol{f}^*$

Randomly sample N_0 networks from the search space $\mathcal{A}$;
Compute the relative ranking of N_0 initial networks by NTSR;
Select the top n_0 networks as the parent population;
Evaluate n_0 networks and add the corresponding validation accuracy into the observed population;
for $m = 1$ ***to*** M **do**
    Generate N_m candidate architectures by mutating the network from the parent population;
    Compute the relative ranking of N_m initial networks by NTSR;
    Select the top n_m networks and evaluate them to obtain the corresponding validation accuracy;
    Add the n_m networks and their corresponding validation accuracy into the current observed set;
    Update the parent population by selecting the top n_0 networks from the current observed set;
end
return the best-performing architecture from the observed set.

A.4 Experimental settings

A.4.1 Experimental settings of Section 3.1

To evaluate the performance of zero-cost proxies in the ImageNet-1K dataset, we compute Spearman's ρ (Krishnakumar et al., 2022) and Kendall's τ (Ning et al., 2021) rank correlation between the zero-cost proxy scores and the validation accuracy of these sampled networks. We run each experiment over 5 independent runs with different random seeds and compute the mean and standard

deviation of rank correlations. As showed by Chen et al. (2021a), the initial weight inherit from the pre-trained supernet can achieve the performance comparable to that of the retrained one, we obtain the true accuracies of sampled network when they inherit their weights from the pre-trained supernet. During the search of ETAS, we first randomly sample 500 networks from the Transformer search space and compute the relative rankings of 500 networks in terms of the zero-cost proxy. After that, we choose the top-3 network structures as the parent population and evaluate these networks to obtain their true performance. For each iteration m, we generate a pool of 100 candidate architectures by mutating the current parent population and select the top-3 architectures from 100 candidate architectures in terms of the zero-cost proxy. The mutation probability is set to 0.4. The parent population is updated by selecting the top-3 networks from the current observed set. We set the number of iterations M to 10. We then find the optimal ViT network from the observed set. At last, we follow the training configuration in AutoFormer Chen et al. (2021a) to train the optimal ViT network and obtain its test accuracy on the test set of ImageNet-1k dataset. All experiments are conducted in a machine with an Intel Xeon Gold 5218R CPU and two NVIDIA GeForce RTX 3090 GPUs.

A.4.2 Experimental settings of Section 3.2

We run each experiment over 5 independent runs with different random seeds and compute the mean and standard deviation of rank correlations. We train each GPT-2 model from scratch following the settings of Radford et al. (2019). Validation perplexity is measured over a sequence length of 192 and 32 tokens for WikiText-103 and LM1B datasets, respectively. We use the BPE tokenizer and set the vocab size to 50264. For the WikiText-103 dataset, we train the GPT-2 network $4e4$ steps by LAMB (You et al., 2020) optimizer with batch size 128, learning rate $1e-2$ with cosine scheduler, attention dropout is set to 0.1. For the LM1B dataset, we Train the GPT-2 network $1e5$ steps by LAMB optimizer with batch size 128, learning rate $3e-4$ with cosine scheduler, attention dropout is set to 0.1. During the search of ETAS, we first randomly sample 300 networks from the search space to warmup the entire ETAS algorithm. We compute the relative rankings of these networks in terms of the zero-cost proxy. After that, we choose the top-3 network structures as the parent population and evaluate these networks to obtain their true performance. For each iteration m, we generate a pool of 100 candidate architectures by mutating the current parent population and select the top-3 architectures from the 100 candidate architectures in terms of the zero-cost proxy. The mutation probability is set to 0.3. The parent population is updated by selecting the top-3 networks from the current observed set. We set the number of iterations M to 10. All experiments are conducted in a machine with an Intel Xeon Gold 5218R CPU and two NVIDIA GeForce RTX 3090 GPUs.