electronics

MDPI

Article

Leveraging Neighbor Attention Initialization (NAI) for Efficient Training of Pretrained LLMs

Qiao Tan and Jingjing Zhang *

School of Information Science and Techonlogy, Fudan University, Shanghai 200433, China; 21210720074@m.fudan.edu.cn
* Correspondence: jingjingzhang@fudan.edu.cn

Abstract: In the realm of pretrained language models (PLMs), the exponential increase in computational resources and time required for training as model sizes expand presents a significant challenge. This paper proposes an innovative approach named neighbor attention initialization (NAI) to expedite the training process of larger PLMs by leveraging smaller PLMs through parameter initialization. Our methodology hinges on the hypothesis that smaller PLMs, having already learned fundamental language structures and patterns, can provide a robust foundational knowledge base for larger models, which is called function preserving. Specifically, we present a comprehensive framework detailing the process of transferring learned features on transformer-based language models mainly using the neighbor attention head and neighbor layer. We conducted experiments in GPT-2 and demonstrated that our method yields considerable savings in training costs compared to standard approaches, including learning from scratch and bert2BERT, indicating a notable improvement in training efficiency for large PLMs.

Keywords: large pretrained language models; efficient training; function preserving

check for updates

Citation: Tan, Q.; Zhang, J. Leveraging Neighbor Attention Initialization (NAI) for Efficient Training of Pretrained LLMs. *Electronics* **2024**, *13*, 1550. https://doi.org/10.3390/electronics13081550

Academic Editor: Chang Wook Ahn

Received: 20 March 2024
Revised: 9 April 2024
Accepted: 15 April 2024
Published: 19 April 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Recent advancements in Natural Language Processing (NLP) have been significantly driven by PLMs such as BERT [1], GPT series [2–4], XLNet [5], T5 [6], RoBERTa [7] and LLaMa series [8,9]. These models have demonstrated exceptional performance across a range of NLP tasks. However, the pre-training phase of these large-scale PLMs entails substantial computational resources. The field has observed a growing inclination towards constructing and training exceptionally large models. This trend aims to explore the upper bounds of PLMs' capabilities. Illustrative examples include GPT-2 (117M, 345M, 762M, 1542M parameter); GPT-3 (175B parameters); PanGu-α [10] with 200B parameters; Switch Transformers [11] comprising an unprecedented 1571B parameters; LLaMa series with 7B, 13B, 33B, 70B parameters [8,9]; and Qwen (1.8B, 7B, 14B, 72B parameters) [12]. These models have shown promising results in language comprehension and generation. Notwithstanding their performance, a notable limitation is that these colossal models are generally trained from scratch, not leveraging the already acquired knowledge from their smaller counterparts. However, our empirical studies suggest the presence of overlapping knowledge domains across PLMs of varying sizes, indicating potential training efficiencies to be gained from small PLMs. For example, in Figure 1, the attention patterns of the two GPT-2s with different sizes are quite similar in the same layer and same attention head. And, similar phenomena have been found in previous studies [13]. Furthermore, in Figure 2, we compare the attention patterns of identical heads across different layers in GPT-2 (117M) and GPT-2 (345M). Notably, the 12th layer of GPT-2 (117M), which is its final layer, closely resembles the 13th layer of GPT-2 (345M). This observation underpins the potential for depth expansion in smaller PLMs.

Drawing from these observations, our research presents neighbor attention initialization (NAI), a knowledge transfer approach from smaller to larger PLMs, aimed at boosting

PLMs training efficiency and cost-effectiveness. NAI mainly extends function preserving training [13,14] to Transformer-based PLMs by reuse the parameters of the existing smaller PLMs, and in the process, NAI strictly retains the ability of small models, thereby providing the larger model with an optimized starting point for subsequent training phases. Additionally, NAI employs the closest attention head for width expansion and the nearest layer for depth expansion, for which we observe that there is a large number of similar attention distributions across both neighbor attention heads and neighbor layers.

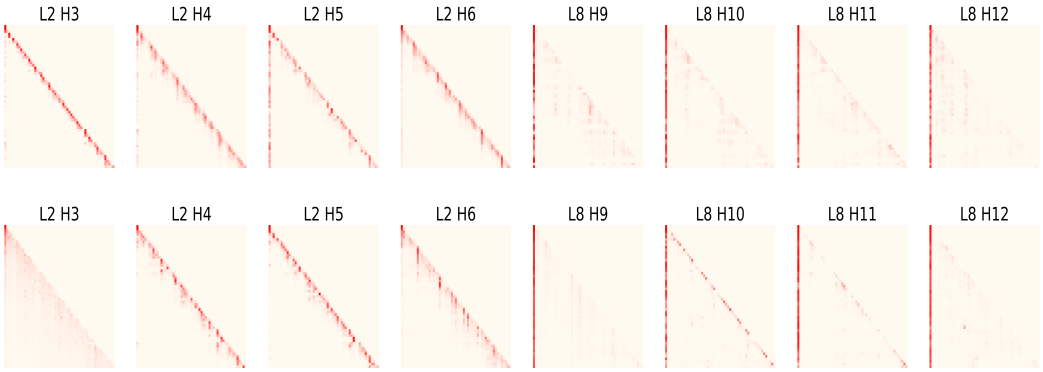

Figure 1. Visualization of attention distributions of pretrained GPT-2 (117M) and GPT-2 (345M) for a random sentence. In each heatmap, the color depth of the j-th element in the i-th row reflects the attention weight from position i to position j. "L2 H3" denotes the third head of the second layer. The upper ones are the attention patterns of the GPT-2 (117M) model of a architecture of {L = 12, D = 768}, and the lower ones are the attention patterns of the GPT-2 (345M) whose architecture is {L = 24, D = 1024}. We find that there are a large number of similar attention distributions in the same layer and the same head of the two models, highlighting the potential for parameter reuse from pretrained small PLMs to expedite the pre-training process of large PLMs.

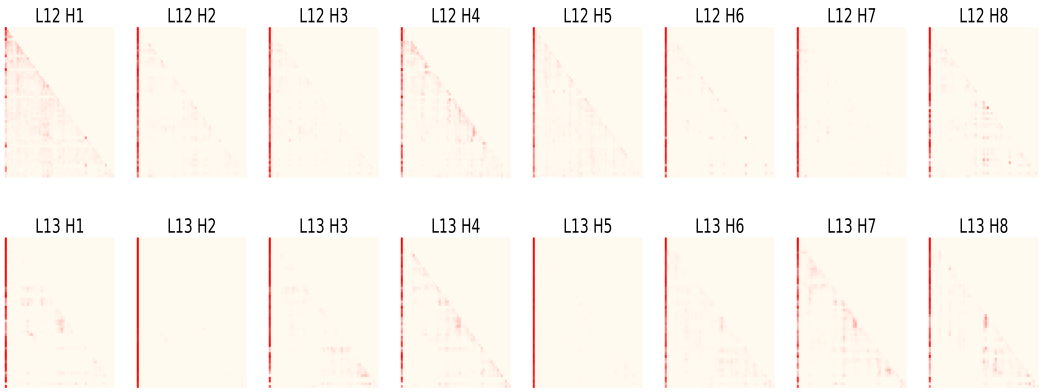

Figure 2. Attention distributions for pretrained GPT-2 models with configurations {L = 12, D = 768} and {L = 24, D = 1024} are visualized. The upper ones shows the final layer of the {L = 12, D = 768} model, while the lower depicts the 13th layer of the {L = 24, D = 1024} model. We find that there is also a large number of similar attention distribution, which underscores the feasibility of expanding the depth in smaller PLMs.

To demonstrate the superiority of our method, we conducted experiments on GPT-2. We employed the GPT-2 model for the following reasons: (1) The GPT-2 model architecture is publicly available in four different size variants, facilitating our research on knowledge transfer across models of varying scales within the same family. (2) While previous work, such as bert2BERT [13], has primarily focused on encoder-based models like BERT [1] as a research foundation, there has been relatively limited exploration of knowledge transfer techniques for decoder-only models. As a seminal decoder-only model, GPT-2 provides

an ideal testbed for investigating efficient pre-training methods tailored to this important class of language models. (3) The model architecture and parameters of the GPT-3 model, a larger successor to GPT-2, are not publicly available. Furthermore, the sheer scale of GPT-3's parameter exceeds the computational resources at our disposal for pre-training models of such magnitude.

Our results show that our method can save a significant amount of computation in pre-training compared to the traditional way of learning from scratch and outperform the existing initialization methods bert2BERT [13]. One typical example is that, using GPT-2 (117M) as a base for GPT-2 (345M) initialization, NAI reduces computation costs by 31% compared to conventional pre-training from scratch.

In general, PLMs within the same series often have varying model sizes, as larger models tend to demonstrate enhanced overall capabilities while incurring greater computational resource requirements. However, for relatively simple tasks such as speech generation, smaller models may suffice. Existing common methods for training PLMs of varying scales often involve training each model from scratch, without leveraging the knowledge encapsulated in previously trained smaller PLMs. Therefore, the primary research question we aim to address is as follows: For PLMs within the same series but with different model sizes, how can we leverage the existing smaller PLMs to train larger PLMs in a computationally efficient manner?

2. Related Work

Efficient pre-training in NLP has been a topic of exploration in previous research. Some approaches aim to enhance efficiency by reusing parameters from existing PLMs and employing progressive learning techniques. These works, such as [15–17], are motivated by the observation that different layers of PLMs possess similar knowledge, such as attention patterns. The approach involves initially pre-training a smaller model with fewer Transformer layers and subsequently expanding the model iteratively by adding the already-trained layers on top. Another line of research proposes "knowledge inheritance" or "back distillation", wherein the knowledge of small models is transferred into larger models [18]. Another aspect of efficient pre-training focuses on data efficiency [19], where rare words are given special attention during the pre-training process to enhance the model's understanding when encountering them subsequently.

Efficiency in pre-training can also be improved through the use of specific pre-training tasks. One notable example is ELECTRA [20], which introduces a task called replaced token detection. This task involves predicting whether each token in a corrupted input has been replaced or not. Our proposed method is orthogonal to this line of work. Additionally, there are several other techniques that can be employed for efficient pre-training, including mixed precision training [21], large batch optimization [16], model architecture innovation [22], and layer dropping training techniques [23].

Another relevant topic is reusable neural networks, which is closely related to transfer learning [24] and has primarily been explored in the field of computer vision. Net2Net [14] is a seminal work in this area, which introduces the concept of function-preserving transformations to enable the reuse of neural networks. However, Net2Net randomly selects neurons to be split, and subsequent work [25] addresses this issue by employing a functional steepest-descent approach to determine the optimal subset of neurons for splitting. The pruning technique [26] has also been employed for reusable neural networks [27]. In addition to [27], another notable study introduces the concept of hierarchical pre-training. This approach effectively reduces both the time required for pre-training and enhances overall performance by leveraging an already pre-trained vision model as an initialization step in the pre-training process. Recently, ref. [13] extended the previous function-preserving [14] Transformer-based language model and further improved it by proposing advanced knowledge for a large model's initialization, which uses the parameters of the next transformer layer to expand each layer. In this paper, we focus on the reusable pre-training of language models and present a new method called NAI, which accelerates the pre-training of GPT-2.

3. Method

3.1. Preliminary

Before discussing our methodology, we briefly outline the GPT-2 architecture, which comprises an embedding layer followed by several Transformer decoders [3,28]. The architecture is shown in Figure 3.

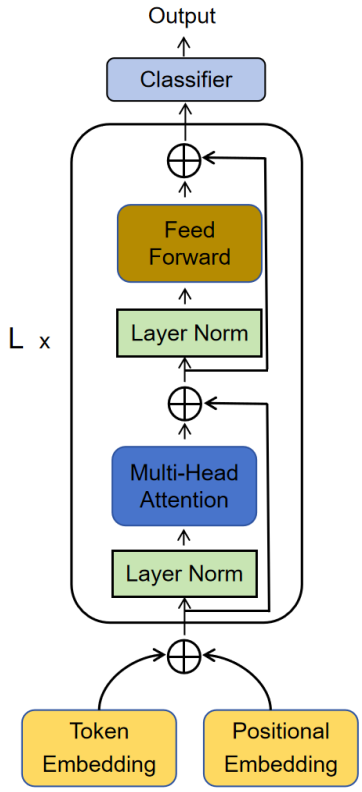

Figure 3. The model architecture of GPT-2.

3.1.1. Embedding Layer

The embedding layer consists of token embedding and positional embedding, which maps the tokens in a sentence into vectors with two embedding matrices $\boldsymbol{W}^{TE}$ and $\boldsymbol{W}^{PE}$. Then, we obtain the initial hidden states $\boldsymbol{H}_0$.

3.1.2. Transformer Layer

The hidden states are iteratively processed by multiple Transformer layers:

$$\boldsymbol{H}_l = \text{Transformer}_l(\boldsymbol{H}_{l-1}), l \in [1, L] \tag{1}$$

where L denotes the number of Transformer layers, each including a multi-head casual attention (MHCA) and a feed-forward network (FFN), and both of them have residual connections [29].

Layer Normalization. Both MHCA and FFN modules utilize a layer normalization [30] to stabilize hidden state dynamics. Formally, it is written as:

$$\text{LayerNorm}(\boldsymbol{H}) = \left(\frac{\boldsymbol{H} - \mu}{\sqrt{\sigma^2 + \epsilon}}\right) \odot \boldsymbol{W}^{LN} + \boldsymbol{b}^{LN}, \tag{2}$$

where $\odot$ means the element-wise multiplication, the statistics of μ and σ are the calculated mean and variance of hidden states $\boldsymbol{H}$ respectively, ϵ is a small constant added for numerical stability, and $\boldsymbol{W}^{LN}$ and $\boldsymbol{b}^{LN}$ are the scale and shift parameters.

MHCA. The Transformer decoder's key component, MHCA, processes the layer's hidden states $\boldsymbol{H}_l$ through multiple heads, concatenating their outputs to form the final output as follows:

$$\begin{aligned}
\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i &= \boldsymbol{H}_l \boldsymbol{W}_{l,i}^{Q}, \boldsymbol{H}_l \boldsymbol{W}_{l,i}^{K}, \boldsymbol{H}_l \boldsymbol{W}_{l,i}^{V} \\
\boldsymbol{h}_{l,i} &= \text{softmax}(\frac{\boldsymbol{Q}_i \boldsymbol{K}_i^T}{\sqrt{d_k}} \odot \boldsymbol{W}^{MASK}) \boldsymbol{V}_i \\
\boldsymbol{H}_l &= \text{Concat}(\boldsymbol{h}_{l,1}, \boldsymbol{h}_{l,2}, \ldots, \boldsymbol{h}_{l,a}) \\
\boldsymbol{H}_l^{MHCA} &= \boldsymbol{H}_l \boldsymbol{W}_l^{O} + \boldsymbol{H}_{l-1}.
\end{aligned} \tag{3}$$

After the Layer Normalization, the output $\boldsymbol{H}_l$ is linearly projected to queries ($\boldsymbol{Q}_i$), keys ($\boldsymbol{K}_i$) and values ($\boldsymbol{V}_i$) using different weights $\boldsymbol{W}_{l,i}^{Q}, \boldsymbol{W}_{l,i}^{K}, \boldsymbol{W}_{l,i}^{V}$, respectively. $\boldsymbol{h}_l^i$ indicates the context-aware vector, which is obtained by the scaled dot-product of queries and keys in the i-th casual attention head. a represents the number of self-attention heads. d_k is the head dimension acting as the scaling factor. $\boldsymbol{W}^{MASK}$ is the attention mask to ensure that the model, during self-attention calculations, only considers its previous words and not the subsequent words. And, $\boldsymbol{H}_l^{\text{MHCA}}$ is the input of the next sub-module FFN.

FFN. FFN consists of two linear layers and one NewGeLU activation function [3], which is formatted as:

$$\begin{aligned}
\boldsymbol{H}_l &= \text{LayerNorm}(\boldsymbol{H}_l^{MHCA}) \\
\boldsymbol{H}_l^{FNN} &= \text{NewGeLU}(\boldsymbol{H}_l \boldsymbol{W}_l^{1} + \boldsymbol{b}_l^{1}) \boldsymbol{W}_l^{2} \\
\boldsymbol{H}_l &= \boldsymbol{H}_l^{FNN} + \boldsymbol{H}_l^{MHCA}
\end{aligned} \tag{4}$$

3.2. Overview

We aim to accelerate the pre-training of target model $T(L^t, D^t)$ by transferring the knowledge of an existing pre-trained model $S(L^s, D^s)$, where L^s, L^t means the number of Transformer layers, and D^s, D^t means the model width (i.e., hidden size), satisfying $L^s \leq L^t$ and $D^s \leq D^t$. Similar to [13], NAI initializes the target model T using the parameters from the existing model S by the width-wise expansion ($D^s \rightarrow D^t$) and depth-wise expansion ($L^s \rightarrow L^t$). This expansion enables direct knowledge transfer from the source model's parameters to the target model. Essentially, the width-wise expansion can be decomposed into two kinds of operations: in-dimension and out-dimension expansion. As illustrated in Figure 4, the matrix expansion enlarges a parameter matrix $\boldsymbol{W} \in \mathbb{R}^{d_{in}^{w} * d_{out}^{w}}$ of source model S to $\boldsymbol{U} \in \mathbb{R}^{d_{in}^{u} * d_{out}^{u}}$ of target model T. In the following sections, we introduce our neighbor attention initialization (NAI), which employs the closest attention head for width expansion and the nearest layer for depth expansion, simultaneously retaining the capabilities of the original model.

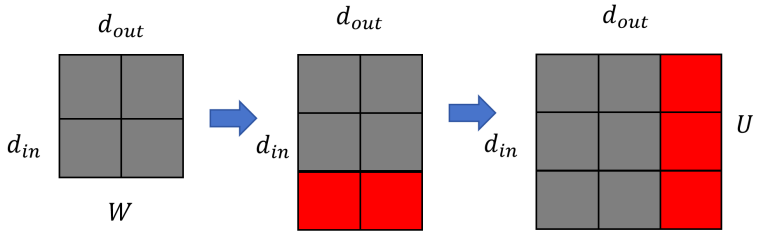

Figure 4. Overview of the matrix expansion. It first enlarges the size of in-dimension and then enlarges the size of out-dimension.

3.3. NAI

We observe that there is a large number of similar attention distributions across both neighbor attention heads and neighbor layers in GPT-2 [2–4], which is also mentioned of

Bert [1] in previous works [31,32]. For example, in Figure 5, we can find that there are lots of similar distributions across the neighbor attention heads in the same layer.

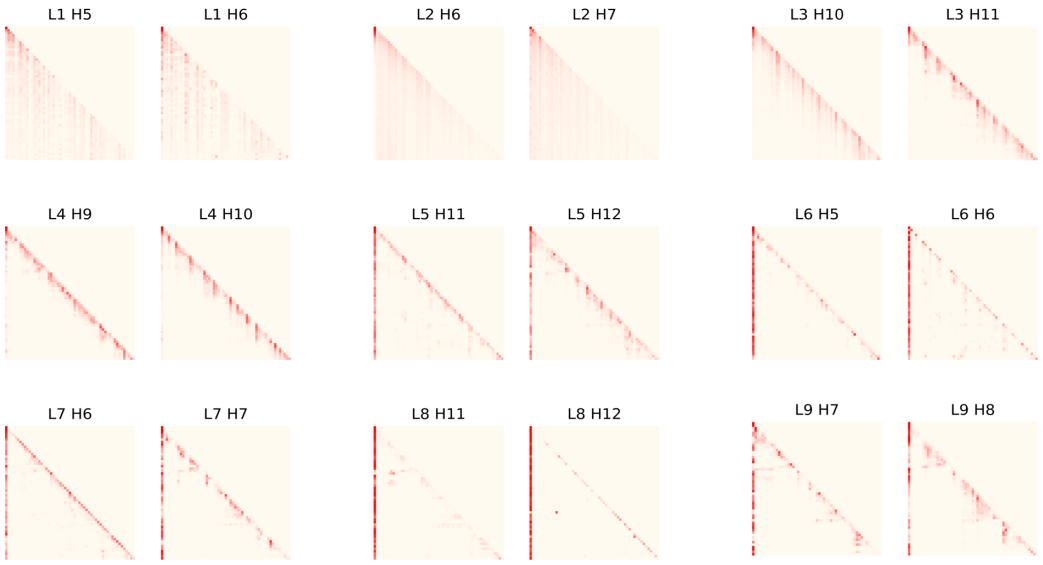

Figure 5. Attention visualization of neighbor attention heads in the same layer for pretrained GPT-2 (L = 24, D = 1024). We find that there is a large number of similar attention distributions in the neighbor heads of same layer.

3.3.1. Width-Wise Expansion

Width-wise expansion includes two kinds of operations: in-dimension and out-dimension expansion. Based on the results of Figure 5, NAI first conducts the in-dimension expansion by copying the weights of the last dimension and then performs the weight normalization to ensure that the output is the same as the original output. Second, NAI conducts the out-dimension expansion, which just does the copy operation. Note that because of the multi-head attention mechanism, all the operations are conducted in the head-wise. Lastly, to break the symmetry more rapidly [14], a small amount of noise is added to the expanded dimension. Figures 6 and 7 illustrate how NAI conducts width-wise expansion. We hereby denote the weight expansion as

$$\boldsymbol{U} = \mathrm{EXPN}_{\mathrm{out}}(\mathrm{EXPN}_{\mathrm{in}}(\boldsymbol{W})), \tag{5}$$

which includes two stages of expansion, in-dimension expansion $\mathrm{EXPN}_{\mathrm{in}}$ and out-dimension expansion $\mathrm{EXPN}_{\mathrm{out}}$.

Expansion for all modules. We apply width-wise expansion of NAI for all modules of GPT-2 via matrix expansion $\mathrm{EXPN}_{\mathrm{in}}$ and $\mathrm{EXPN}_{\mathrm{out}}$. Specifically, for the embedding matrix $\boldsymbol{U}^{TE}$ and $\boldsymbol{U}^{PE}$, we only conduct the out-dimension expansion:

$$\begin{aligned} \boldsymbol{U}^{TE} &= \mathrm{EXPN}_{\mathrm{out}}(\boldsymbol{W}^{TE}), \\ \boldsymbol{U}^{PE} &= \mathrm{EXPN}_{\mathrm{out}}(\boldsymbol{W}^{PE}). \end{aligned} \tag{6}$$

Layer normalization is a frequently employed component that is typically applied on top of MHCA and FFN modules. This normalization technique ensures that the input data undergoes normalization. Its expansion is formulated as follows:

$$\boldsymbol{U}^{LN} = \mathrm{EXPN}_{\mathrm{out}}(\boldsymbol{W}^{LN}) \tag{7}$$

The MHCA module can be decomposed into multiple parallel self-attention heads. It is worth noting that in Equation (3), when we calculate attention score, it will be scaled according to dimension $\sqrt{d_k}$, so after completing the expansion of Q, K, we need to multiply

the overall attention by the coefficient $\frac{D^t}{D^s}$. At the same time, in order to scale the impact of coefficient $\frac{D^t}{D^s}$ on the original matrix, we allocate it to both Q and K. A more detailed explanation of this operation will be discussed later. The expansion can be formulated as:

$$\begin{aligned} \boldsymbol{U}^{Q|K} &= \sqrt{\sqrt{\frac{D^t}{D^s}}}\mathrm{EXPN}_{\mathrm{out}}(\mathrm{EXPN}_{\mathrm{in}}(\boldsymbol{W}^{Q|K})), \\ \boldsymbol{U}^{V} &= \mathrm{EXPN}_{\mathrm{out}}(\mathrm{EXPN}_{\mathrm{in}}(\boldsymbol{W}^{V})), \\ \boldsymbol{U}^{O} &= \mathrm{EXPN}_{\mathrm{out}}(\mathrm{EXPN}_{\mathrm{in}}(\boldsymbol{W}^{O})). \end{aligned} \tag{8}$$

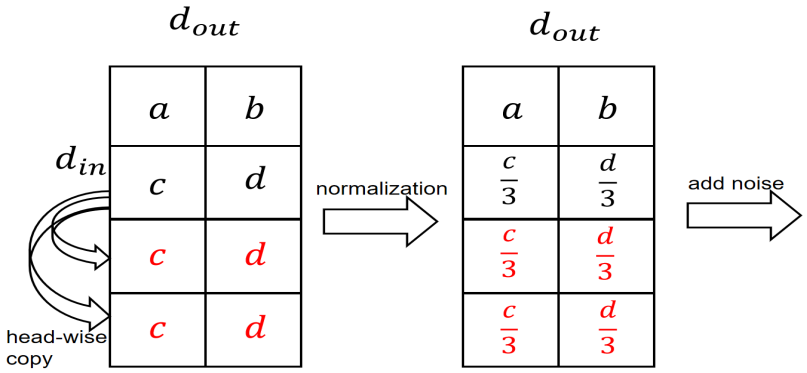

Figure 6. In-dimension expansion of NAI. The normalization ensures the output is the same as the original output.

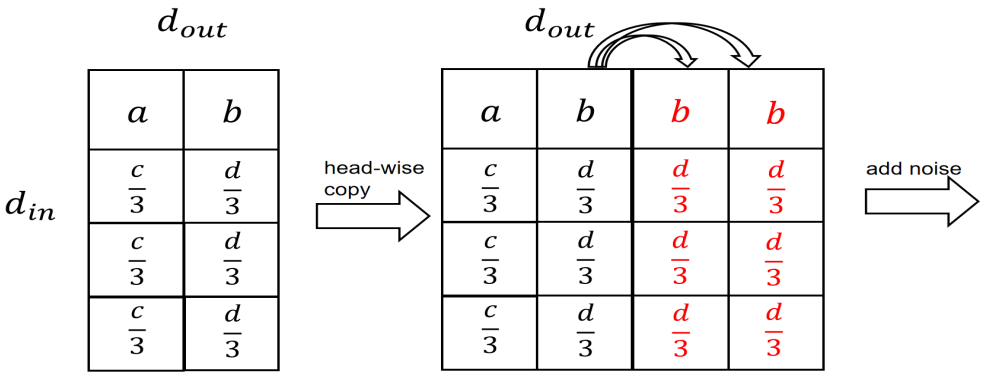

Figure 7. Out-dimension expansion of NAI.

For the FFN module, we perform the expansion on the parameter matrices $\boldsymbol{W}^{1|2}$ (Equation (4)) as follows:

$$\boldsymbol{U}^{1|2} = \mathrm{EXPN}_{\mathrm{out}}(\mathrm{EXPN}_{\mathrm{in}}(\boldsymbol{W}^{1|2})). \tag{9}$$

Additionally, we introduce an extension to the classifier layer, typically comprising a linear layer, as our observations have indicated that this modification can expedite the convergence of the loss function:

$$\boldsymbol{U}^{CLS} = \mathrm{EXPN}_{\mathrm{in}}(\boldsymbol{W}^{CLS}). \tag{10}$$

We provide a simplified example to illustrate the knowledge transfer process of NAI. In this example, we consider a GPT-2 structured source model S {12, 128}, which is extended to a target model T {12, 192}. The dimension of the attention head is 64, and the hidden layer state is represented head-wise. That means, for example, we represent the 128-dimensional vector $\boldsymbol{X}$ as $[\boldsymbol{X}_0, \boldsymbol{X}_1]$. We denote the hidden state after passing through each module of the source model S as the left-hand side of the arrow, and the hidden state after passing

through T as the right-hand side. Assume that the input data batch is 1; after passing through the embedding layer, we have the initial hidden state:

$$\boldsymbol{H}_0 = [\boldsymbol{H}_{0,0}, \boldsymbol{H}_{0,1}] \rightarrow \boldsymbol{H}_0^{'} = [\boldsymbol{H}_{0,0}, \boldsymbol{H}_{0,1}, \boldsymbol{H}_{0,1}]. \tag{11}$$

After the layer normalization layer, if we ignore the influence of the mean and variance of the hidden layer state , we have the following:

$$\boldsymbol{H}_1 = [\boldsymbol{H}_{1,0}, \boldsymbol{H}_{1,1}] \rightarrow \boldsymbol{H}_1^{'} = [\boldsymbol{H}_{1,0}, \boldsymbol{H}_{1,1}, \boldsymbol{H}_{1,1}]. \tag{12}$$

Then $\boldsymbol{H}_1$ and $\boldsymbol{H}_1^{'}$ are projected through $\boldsymbol{W}^{Q|K|V}$ and $\boldsymbol{U}^{Q|K|V}$:

$$\begin{aligned} \boldsymbol{Q} = [\boldsymbol{Q}_0, \boldsymbol{Q}_1] \rightarrow \boldsymbol{Q}^{'} &= \sqrt{\sqrt{\frac{D^t}{D^s}}}[\boldsymbol{Q}_0, \boldsymbol{Q}_1, \boldsymbol{Q}_1] \\ \boldsymbol{K} = [\boldsymbol{K}_0, \boldsymbol{K}_1] \rightarrow \boldsymbol{K}^{'} &= \sqrt{\sqrt{\frac{D^t}{D^s}}}[\boldsymbol{K}_0, \boldsymbol{K}_1, \boldsymbol{K}_1] \\ \boldsymbol{V} = [\boldsymbol{V}_0, \boldsymbol{V}_1] \rightarrow \boldsymbol{V}^{'} &= [\boldsymbol{V}_0, \boldsymbol{V}_1, \boldsymbol{V}_1]. \end{aligned} \tag{13}$$

Then the attention score of the i-th head in target model T can be written as:

$$\begin{aligned} \frac{\boldsymbol{Q}_i^{'}\boldsymbol{K}_i^{'T}}{\sqrt{D^t}} &= \frac{\sqrt{\sqrt{\frac{D^t}{D^s}}}\boldsymbol{Q}_i\sqrt{\sqrt{\frac{D^t}{D^s}}}\boldsymbol{K}_i^T}{\sqrt{D^t}} \\ &= \frac{\boldsymbol{Q}_i\boldsymbol{K}_i^T}{\sqrt{D^s}}, \end{aligned} \tag{14}$$

which is the attention score of the i-th head in the source model S. The analysis of the subsequent layers follows a similar pattern. Also, we expand a pretrained GPT-2 of {L = 12, D = 768} to a target GPT-2 model of {L = 12, D = 1024}. Figure 8 shows that the newly formed attention heads by NAI basically retains the capabilities of the original neighbor heads.

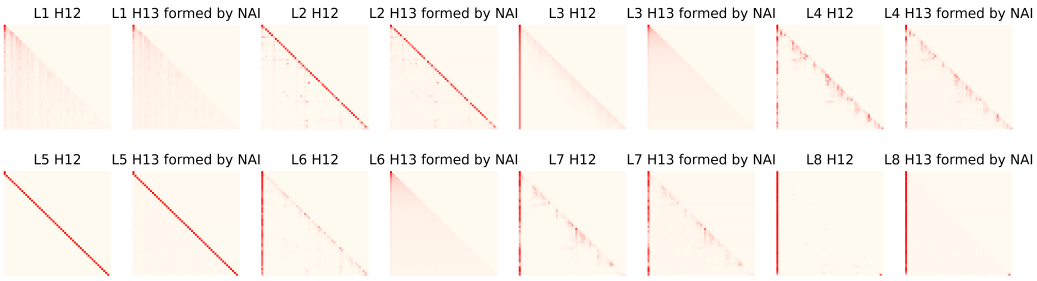

Figure 8. Attention visualization of the original heads and the newly formed heads of a GPT-2 model {L = 12, D = 1024} expanded by NAI with the source model pretrained GPT-2 {L = 12, D = 1024}. We can find that the newly formed attention heads basically retains the capabilities of the original neighbor heads. The attention visualization of the original and expanded heads in a GPT-2 model {L = 12, D = 1024}, augmented by NAI with the pretrained GPT-2 model {L = 12, D = 1024}, reveals that the newly formed attention heads largely preserve the functionalities of the original neighbor heads.

3.3.2. Depth-Wise Expansion

Previous findings [31,32] show that adjacent Transformer layers have similar functionality, and we also find this phenomenon in GPT-2 as illustrated in Figure 2. Based on this phenomenon, when we perform depth-wise expansion, we continuously copy the last layer of the original model to reach the number of layers of the target model. From an intuitive perspective, adopting weight copying from the last layer appears to align more consistently with the gradual and progressive learning process observed in each layer of the transformer

model. To preserve the capabilities of the original model, we set the output layers of the MHCA and FFN module to zero, leveraging the residual connection architecture. Moreover, it was observed that failing to set the output layers of the MHCA and FFN modules to zero can lead to volatile training behavior, ultimately causing divergence in the learning process. Figure 9 provides a concise and comprehensible depiction of the depth-wise expansion of NAI. By initializing the deeper model with transferred knowledge from the L^s-layer trained model, we anticipate accelerated learning compared to training the deeper model from scratch. The whole procedure is summarized in Algorithm 1.

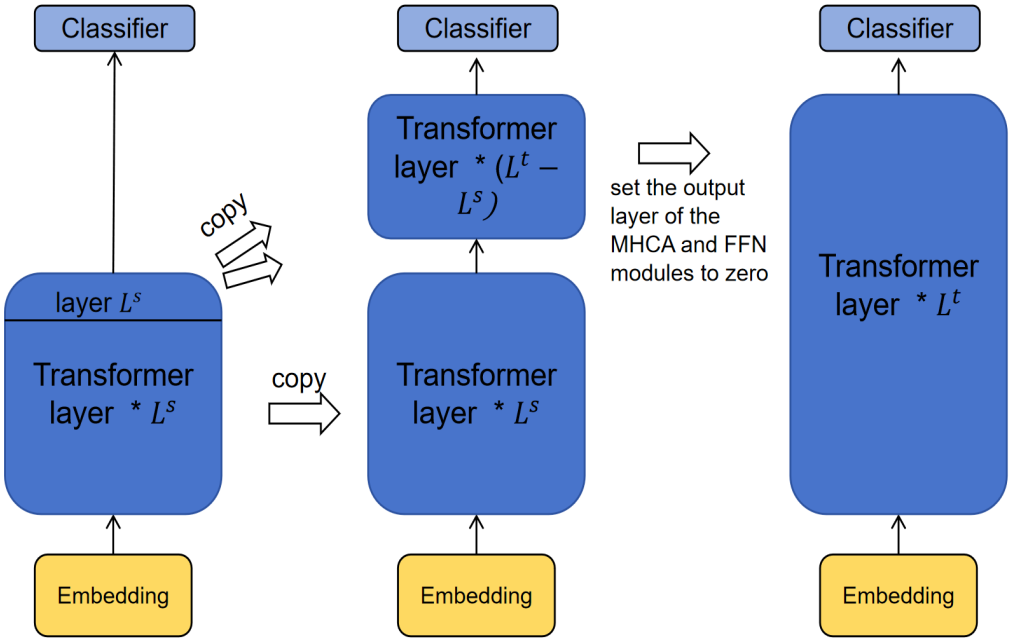

Figure 9. Depth-wise expansion of NAI. In order to ensure the preservation of the original model's capabilities and maintain stability during the training process, we set the output layers of the MHCA and FFN module to zero.

Algorithm 1 NAI algorithm

Input: the source model $S(L^s, D^s)$ and the target model $T(L^t, D^t)$
$T_{L^s}(L^s, D^t) \leftarrow S(L^s, D^s)$ by width-wise expansion of NAI
for $t = L^s + 1 \rightarrow L^t$ **do**
    $T_t(L^{t+1}, D^t) \leftarrow$ stack the last layer of T_{L^s} on top of T_{t-1}
    set the output layer of the MHCA and FFN modules to zero of layer t
end for
output: the target model $T(L^t, D^t)$

4. Experiment

4.1. Experimental Setup

Pre-training Details. All of our experiments are mainly based on our own reimplementation of the GPT-2 model. We use the concatenation of OpenWebText [33] and English Wikipedia as the pre-training data. For dataset OpenWebText, we performed a de-duplication operation and removed a non-English corpus. For both datasets, sentences with a length below 128 were excluded from both datasets. The settings of the pre-training are peak learning rate of 1.5×10^{-4} , warmup step of 10^4, training iteration of 10^6, and a small batch size of 128 due to limited computing resources. The AdamW optimizer was used with $[\beta_1, \beta_2] = [0.9, 0.999]$, weight decay = 0.01. Unless otherwise noted, all methods, including NAI and baselines, use the same pre-training settings for fair comparisons. In the settings of NAI and bert2BERT, the target models has a GPT-2 architecture of {24, 1024}

and {36, 1280}, and an architecture of source models {12, 768} {24, 1024} are evaluated. Both source models are pretrained using the above settings, and they are training from scratch without any optimization. The computations in this research were performed using the CFFF platform of Fudan University with 2 NVIDIA A100 GPUs.

Baselines. We initially introduce a method training from scratch without any optimization, named GPT_{base}. As mentioned above, all the source models used in NAI and bert2BERT are trained by this. Then, we introduce an approach called DirectCopy, which involves the direct replication of the source model into the target model, while employing random initialization for the parameters that have not been populated. bert2BERT is also included as the baseline. During the width-wise expansion in bert2BERT, advanced knowledge from the subsequent layer is utilized for initializing the large model; the detailed process can be found in [13]. In the depth-wise expansion phase of bert2BERT, the target model is constructed using the StackBERT technique [15], and the detailed process of StackBERT is provided in Algorithm 2. This approach involves the continuous replication of the entire source model after the width-wise expansion. Note that here, we only compare various initialization methods, so the two-stage pre-training in bert2BERT is not considered.

Algorithm 2 StackBERT

1: **Input:** the source model $S(L^s, D)$ and the target model $T(L^t, D)$
2: $k \leftarrow \lfloor L^t / L^s \rfloor, T_1 \leftarrow S(L^s, D)$
3: **for** $t = 2 \rightarrow k$ **do**
4: $\quad T_t(L^s * t, D) \leftarrow$ stack T_1 on top of T_{t-1}
5: **end for**
6: $T(L^t, D) \leftarrow$ stack the top $L^t - L^s * k$ layer of T_1 on top of T_k
7: **output:** the target model $T(L^t, D)$

4.2. Results and Analysis

We present the pre-training loss curves of GPT-2 models with config {24, 1024} in Figure 10. The results reveal the following insights: (1) DirectCopy—This method, which involves directly copying the trained parameters from the source model to the target model, exhibits minimal computational cost savings. This suggests that the simplistic approach of DirectCopy is not effective in achieving cost reductions. (2) bert2BERT—By utilizing advanced knowledge for initializing the large model and employing StackBERT for depth-wise expansion, bert2BERT achieves approximately 21% computational cost savings. However, the departure from the function-preserving initialization principle, caused by the utilization of advanced knowledge and depth-wise expansion, limits the acceleration of loss reduction during the early stages of training. Consequently, the preservation of the source model's capabilities is not substantially retained. (3) NAI (Our proposed method)—NAI outperforms the baselines in terms of performance. It achieves a 10% improvement in computational cost savings compared to bert2BERT. Notably, NAI adheres to the function-preserving initialization principle during both width-wise and depth-wise operations, resulting in a rapid decrease in loss during the initial stages of training. By preserving the capabilities of the source model, NAI ensures an effective initialization of the target model.

Additionally, we depict the pre-training loss curves of GPT-2 models with config {36, 1280} in Figure 11. In comparison with Figure 10, we observe the following trends: All the methods, including DirectCopy, bert2BERT, and NAI, achieve greater computational cost savings compared to the baseline model GPT_{base}. For example, DirectCopy demonstrates approximately 12% more cost savings compared to GPT_{base}, whereas it had negligible savings in Figure 10. Method bert2BERT achieves a cost reduction of 24%, surpassing the 21% achieved in Figure 10. Similarly, NAI achieves a cost savings of 33%, slightly higher than the 31% in Figure 10. And, it also achieves a 9% improvement compared to bert2BERT. We attribute this phenomenon to the closer scale between the source model and the target model. These detailed analyses provide a comprehensive understanding of

the experimental results, highlighting the effectiveness of NAI in achieving computational cost savings while preserving the capabilities of the source model.

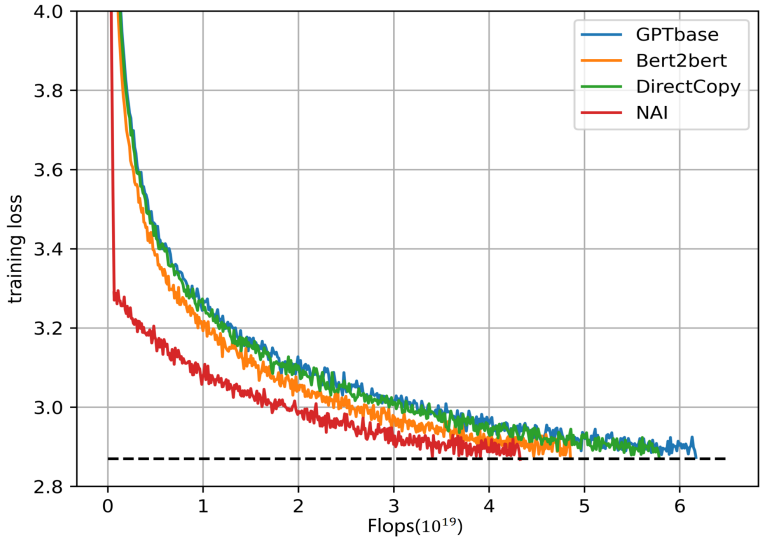

Figure 10. Pre-training loss curves of GPT-2 {24, 1024} of NAI and baselines.

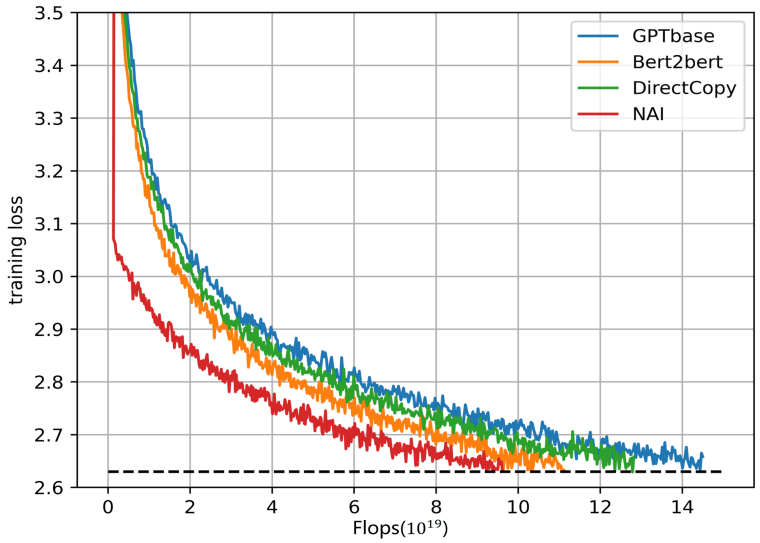

Figure 11. Pre-training loss curves of GPT-2 {36, 1280} of NAI and baselines.

We also evaluated the effectiveness of all methods on the LAMBADA, PTB, WikiText2, and WikiText103 test datasets; the following is a brief introduction to them.

LAMBADA: LAMBADA is a dataset designed for testing the ability of language models to perform pronoun resolution in context. It consists of narrative passages followed by a single line continuation. The task is to predict the last word of the continuation, which requires understanding the context provided by the passage.

PTB: PTB is one of the most widely used datasets in natural language processing. It consists of annotated text from various sources, including newswire, transcribed speech, and literature. The dataset is extensively used for tasks such as language modeling, part-of-speech tagging, and parsing.

WikiText2: WikiText2 is a dataset derived from Wikipedia articles. It is designed for language modeling tasks, including next-word prediction. The dataset contains high-quality text with diverse topics and writing styles.

WikiText103: WikiText103 is an extension of WikiText2 and contains a larger corpus of Wikipedia articles. Like WikiText2, it is designed for language modeling tasks and provides a diverse range of textual data.

All these datasets are commonly used in natural language processing research for training and evaluating various language models and algorithms.

The test results are shown in Tables 1 and 2. We employed a zero-shot approach for testing on these datasets, which means no training or fine-tuning was performed on the datasets. Perplexity (PPL) was used as the evaluation metric, and the average of three test results was taken as the final result for each test set. From the tables, we can conclude that both bert2BERT and NAI perform comparably to GPT_{base} and DirectCopy methods. Furthermore, both bert2BERT and NAI demonstrate significant cost savings in terms of computational resources, with NAI exhibiting even better cost savings in training.

Table 1. Zero-shot results on different datasets of GPT-2 {24, 1024}. We use the perplexity as the metric.

Model	Flops ($\times 10^{19}$)	LAMBADA	PTB	WikiText2	WikiText103
GPT_{base}	6.2	33.7	68.9	28.7	38.6
DirectCopy	5.8	33.4	68.3	28.4	38.5
bert2BERT	4.9	32.6	**67.5**	**28.1**	38.2
NAI	**4.3**	**32.2**	67.6	28.3	**37.7**

Table 2. Zero-shot results on different datasets of GPT-2 {36, 1280}. We use the perplexity as the metric.

Model	Flops ($\times 10^{19}$)	LAMBADA	PTB	WikiText2	WikiText103
GPT_{base}	14.4	18.2	45.6	23.2	26.6
DirectCopy	12.7	**17.6**	45.8	23.4	25.7
bert2BERT	11	17.8	45.5	**23.1**	**25.4**
NAI	**9.6**	17.8	**45.2**	23.7	26.5

5. Potential Limitations and Challenges of NAI

The NAI method, as presented in our study, offers a novel approach to efficiently train larger PLMs by leveraging the knowledge encapsulated in smaller PLMs. Despite its potential to reduce computational costs and expedite training, there are several limitations and challenges associated with applying NAI to different model architectures and domains.

1. Architectural Limitations: NAI is inherently tailored for Transformer-based architectures, which are distinguished mainly by their self-attention mechanisms. These architectural elements are pivotal for the function-preserving initialization process that NAI employs. The self-attention mechanism, in particular, enables the model to weigh the importance of different parts of the input data, whereas layer normalization facilitates the stabilization of the learning process by reducing the internal covariate shift. Given the constraints of computational resources, the empirical validation of NAI has been primarily confined to the GPT-2 model, which is a decoder-only PLMs. This limitation necessitates a cautious extrapolation of NAI's efficacy to other PLM architectures, such as those based on encoder-only or encoder–decoder configurations. Encoder-only models, such as BERT, utilize a bidirectional training strategy that contrasts with the unidirectional nature of GPT-2, while encoder–decoder models, akin to sequence-to-sequence frameworks, are designed to handle input–output mapping tasks that diverge from the autoregressive generation focus of GPT-2. The transference of NAI to these alternative architectures entails a rigorous examination

of potential modifications required to align with their distinct operational mechanisms. For instance, the adaptation of NAI to encoder-only models may necessitate strategies that effectively capture and preserve the bidirectional context during the initialization process. Similarly, encoder–decoder models may demand a reconceptualization of the NAI approach to accommodate the dual-input and dual-output structure characteristic of these architectures.

2. Data Availability and Quality: NAI's efficiency is contingent on the availability and quality of the smaller PLMs used for initialization. In domains where high-quality, well-pretrained models are scarce, the effectiveness of NAI may be compromised. Additionally, the method assumes a certain level of overlap in the knowledge domains between the smaller and larger models, which might not always be the case, especially in niche or rapidly evolving domains.
3. Generalization and Adaptability: The success of NAI hinges on the assumption that smaller PLMs have learned foundational language structures that are generalizable to larger models. However, the extent to which this generalization holds true across various tasks and languages remains an open question. Furthermore, adapting NAI to non-English languages or multilingual settings may require additional considerations to account for linguistic diversity.

6. Conclusions and Future Work

This paper introduces a novel and efficient pre-training method called NAI. The key idea behind NAI is to employ the closest attention head for width expansion and the nearest layer for depth expansion to get a larger model. By leveraging the knowledge embedded in the small model, NAI aims to improve the efficiency and effectiveness of pre-training large models.

To evaluate the effectiveness of NAI, we applied this method to the popular GPT model. The experimental results demonstrate the benefits of NAI in terms of computational cost reduction and performance improvement on various test datasets. Notably, our method achieves significant savings in computational resources while maintaining competitive performance compared to alternative techniques.

However, it is important to note that due to the limitations of computational resources, we were unable to conduct extensive experiments to fully validate the algorithm. Despite this limitation, the promising results obtained in our initial experiments serve as a strong indication of the potential of NAI in pre-training large-scale language models.

In the future, we plan to extend the application of NAI to train even larger language models. Moreover, we aim to explore the feasibility and effectiveness of applying NAI to other domains and tasks beyond language modeling, thereby expanding its scope to very large PLMs in various fields. This would enable us to further investigate the benefits and limitations of NAI and contribute to the advancement of efficient and effective pre-training techniques.

Author Contributions: Conceptualization, Q.T.; methodology, Q.T. and J.Z.; software, Q.T.; validation, Q.T.; writing—original draft preparation, Q.T.; writing—review and editing, Q.T. and J.Z.; visualization, Q.T.; supervision, J.Z.; project administration, J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the National Natural Science Foundation of China Grant No. 62101134.

Data Availability Statement: The OpenWebtext dataset can be downloaded in http://Skylion007.github.io/OpenWebTextCorpus (accessed on 1 December 2023), and the Wikipedia dataset can be downloaded in https://dumps.wikimedia.org (accessed on 1 December 2023).

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
2. Radford, A.; Narasimhan, K. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://www.mikecaptain.com/resources/pdf/GPT-1.pdf (accessed on 15 November 2023)
3. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* **2019**, *1*, 9.
4. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.
5. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.G.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Proceedings of the Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
6. Raffel, C.; Shazeer, N.M.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2019**, *21*, 1–67.
7. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
8. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv* **2023**, arXiv:2302.13971.
9. Touvron, H.; Martin, L.; Stone, K.R.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv* **2023**, arXiv:2307.09288.
10. Zeng, W.; Ren, X.; Su, T.; Wang, H.; Liao, Y.; Wang, Z.; Jiang, X.; Yang, Z.; Wang, K.; Zhang, X.; et al. PanGu-α: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation. *arXiv* **2021**, arXiv:2104.12369.
11. Fedus, W.; Zoph, B.; Shazeer, N.M. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *J. Mach. Learn. Res.* **2021**, *23*, 1–39.
12. Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. Qwen Technical Report. *arXiv* **2023**, arXiv:2309.16609.
13. Chen, C.; Yin, Y.; Shang, L.; Jiang, X.; Qin, Y.; Wang, F.; Wang, Z.; Chen, X.; Liu, Z.; Liu, Q. bert2BERT: Towards Reusable Pretrained Language Models. *arXiv* **2021**, arXiv:2110.07143.
14. Chen, T.; Goodfellow, I.J.; Shlens, J. Net2Net: Accelerating Learning via Knowledge Transfer. *arXiv* **2015**, arXiv:1511.05641.
15. Gong, L.; He, D.; Li, Z.; Qin, T.; Wang, L.; Liu, T.Y. Efficient Training of BERT by Progressively Stacking. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019.
16. You, Y.; Li, J.; Reddi, S.J.; Hseu, J.; Kumar, S.; Bhojanapalli, S.; Song, X.; Demmel, J.; Keutzer, K.; Hsieh, C.J. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. *arXiv* **2019**, arXiv:1904.00962.
17. Gu, X.; Liu, L.; Yu, H.; Li, J.; Chen, C.; Han, J. On the transformer growth for progressive bert training. *arXiv* **2020**, arXiv:2010.12562.
18. Qin, Y.; Lin, Y.; Yi, J.; Zhang, J.; Han, X.; Zhang, Z.; Su, Y.; Liu, Z.; Li, P.; Sun, M.; et al. Knowledge Inheritance for Pre-trained Language Models. *arXiv* **2021**, arXiv:2105.13880.
19. Wu, L.; Liu, B.; Stone, P.; Liu, Q. Firefly Neural Architecture Descent: A General Approach for Growing Neural Networks. *arXiv* **2021**, arXiv:2102.08574.
20. Clark, K.; Luong, M.T.; Le, Q.V.; Manning, C.D. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 30 April 2020.
21. Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; Catanzaro, B. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv* **2019**, arXiv:1909.08053.
22. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 30 April 2020.
23. Zhang, M.; He, Y. Accelerating Training of Transformer-Based Language Models with Progressive Layer Dropping. *arXiv* **2020**, arXiv:2010.13369.
24. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
25. Liu, Q.; Wu, L.; Wang, D. Splitting Steepest Descent for Growing Neural Architectures. In Proceedings of the Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
26. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both Weights and Connections for Efficient Neural Network. In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015.
27. Feng, A.; Panda, P. Energy-efficient and Robust Cumulative Training with Net2Net Transformation. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–7.
28. Vaswani, A.; Shazeer, N.M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
30. Ba, J.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450.

31. Jawahar, G.; Sagot, B.; Seddah, D. What Does BERT Learn about the Structure of Language? In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019.
32. Clark, K.; Khandelwal, U.; Levy, O.; Manning, C.D. What Does BERT Look at? An Analysis of BERT's Attention. *arXiv* **2019**, arXiv:1906.04341.
33. Gokaslan, A.; Vanya Cohen, E.P.; Tellex, S. OpenWebText Corpus. Available online: https://skylion007.github.io/OpenWebTextCorpus/ (accessed on 1 December 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.