Vol. 06, No. 2 (2024) 439-452, doi: 10.24874/PES06.02.001

Proceedings on Engineering Sciences

www.pesjournal.net

SAMPLE-BASED DYNAMIC HIERARCHICAL TRANSFORMER WITH LAYER AND HEAD FLEXIBILITY VIA CONTEXTUAL BANDIT

Fanfei Meng [1]
Lele Zhang
Yu Chen
Yuxin Wang

Received 29.03.2023.
Received in revised form 14.06.2023.
Accepted 21.08.2023.
UDC – 004.032.26

Keywords:

AutoML, Network Compression, Transformer and Deep Learning

A B S T R A C T

Abstract: Transformer requires a fixed number of layers and heads which makes them inflexible to the complexity of individual samples and expensive in training and inference. To address this, we propose a sample-based Dynamic Hierarchical Transformer (DHT) model whose layers and heads can be dynamically configured with single data samples via solving contextual bandit problems. To determine the number of layers and heads, we use the Uniform Confidence Bound algorithm while we deploy combinatorial Thompson Sampling in order to select specific head combinations given their number. Different from previous work that focuses on compressing trained networks for inference only, DHT is not only advantageous for adaptively optimizing the underlying network architecture during training but also has a flexible network for efficient inference. To the best of our knowledge, this is the first comprehensive data-driven dynamic transformer without any additional auxiliary neural networks that implement the dynamic system. According to the experiment results, we achieve up to 74% computational savings for both training and inference with a minimal loss of accuracy.

1. INTRODUCTION

Transformers have demonstrated significant success in various linguistic tasks, including text classification (Shahen et al., 2020; Korthikanti et al., 2023), machine translation (Vaswani et al., 2017; Baid et al., 2023), and knowledge graphs (Bosselut et al., 2019). These achievements are attributed to the powerful multi-head attention, which projects sequential tokens into parallel attention subspaces at relatively low memory and computational costs. Following the transformer framework, BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020; Hassani et al., 20203) are proposed to train specific linguistic tasks using pre-trained weights.

Although transformers are powerful in various aspects, it is still difficult to embed such a large network into mobile devices constrained by limited power and memory, and standard transformer is still expensive in both training and inference. Furthermore, traditional network compression (Hoffman and Crament, 2019; Neil, 2020; Lee and Hwang, 2021) methods require a full layer training firstly and then reduce the model size

[1] Corresponding author: Fanfei Meng
Email: fanfeimeng2023@u.northwestern.edu

through layer-wise network compression or knowledge distillation (Hinton et al., 2015; Jiao et al., 2015; Tang et al., 2019; Pope et al., 2023; Xiao et al., 2023). On the one hand, this two-step compression procedure can be of high time complexity. On the other hand, different tasks may require different light-weight transformers making the uniform compression inflexible. Therefore, we would like to design a dynamic, data-driven transformer model whose size can be optimized during training, skipping the separate compression step while maintaining a decent predicting capability.

Inspired by neural architecture search (NAS) and network compression (Zoph and Le, 2017; Han et al., 2016), we propose a sample-based Dynamic Hierarchical Transformer (DHT) to bring dynamics to both layers and heads and formulate an adaptive network routine for every data sample. Instead of highly complex neuron-based NAS, we shed light on searching for self-attention space and limit the maximum search space by first obtaining the number of layers needed for a sample, and then moving forward to prune heads layer-by-layer to further reduce the network size. In order to determine the configurations of layers and heads for each sample in a real-time fashion, we integrate Uniform Confidence Bound multi-arm contextual bandits (UCB) (Li et al., 2010) and Thompson Sampling semi-bandits (TSP) (Wang et al., 2017) into our DHT model, where the UCBs are responsible for determining the number of layers as well as the number of heads to keep in each layer, and the TSB the combination of heads to keep in each layer.

A very unique aspect of our approach is the fact that the underlying layers and heads are sample specific, exploiting the strategy that customizes the number of layers and heads for each single sample. In addition, considering the effect of head interactions and the order samples appear during training, our work formulate rewards of batch level rather than one-step gains, which successfully mitigates model performance reductions.

The major contribution of our model is that it alleviates researchers in deep learning from fine-tuning transformer configurations by learning to fine-tune them automatically. Specifically, (i) our work designed a dynamic transformer model that may have different, customized number of layers, number of heads, or head combinations for each individual sample to reduce model size and training time, while maintaining good predicting performance; (ii) our work pioneered a new transformer training mechanism by adopting UCB to determine the optimal number of layers and heads and adopting TSB to select at each layer the optimal head combination for a given sample; (iii) our work study the effect of different subsets of transformer heads, while former approaches, to the best of our knowledge, merely study the effect of each individual head, lacking a more holistic observation on multi-head attention pruning that we achieve.

The structure of this paper is as follows. First, our work list existing work similar to ours and compare how our work is distinct from them in. We further introduce in two algorithms upon which our methodology is founded. Then, our work introduce our methodology in and describe its implementation in. After that, our work explain the experiments we perform, and show and analyze the result in. Lastly, our work summarize this work in.

2. RELATED WORK

Network size compression research is progressing at a exhilarating pace. Some offer inspiring insights of applying compression to multiple deep models (Hinton et al. 2015; Han et al., 2015; Luo et al., 2015; Ullrich et al., 2017; Kim et al., 2019; Molchanov et al., 2019), while others dynamically compress layers using knowledge distillation and network compression for efficient inference (Hou et al, 2020; Liu et al., 2020; Xiao et al., 2019; Li et al., 2021). With respect to transformers, network quantization is developed while imposing specific bandwidth precision compatibility for devices (Chung et al., 2020; Bhandare et al., 2019). The role of each head is revealed and the semantic and syntax functions for each corresponding sample are visualized (Jo and Myaeng, 2020), allowing the development of a head pruning method based on head importance score metrics (Michel et al., 2019). Furthermore, work has been done to shed light on linguistic representations of different heads and employ differential L0 normalization to dynamically prune the heads (Voita, et al., 2019).

In addition to the head pruning strategy, there are methods focusing on explaining attention dynamics. One work illustrates similarities among heads to learn an optimal attention span for controlling the computational time (Sukhbaatar et al., 2020;. Following this work, another explores the deployment of deep Q-learning to achieve a dynamic attention span (Kumar et al., 2020). Moreover, adaptive depth of transformers is investigated, where the layer depth is determined step-by-step (Elbayad et al., 2020).

Sample-based architecture trainings (McCallum, 1996; Wang et al., 2019; Zhang et al., 2020) are investigated for improving training effects by adaptively tackling the complexity of each sample. One work reveals the variance of a data sample in model training (Chang et al., 2017). Following this work, another tests the effect of each sample in back-propagation, and reveals that the learning effect of a single sample varies for the whole network even when the same learning rate is used (Lopes et al., 2017). Inspired by the experiment, dynamic, sample-based learning rate is proposed to train networks more effectively, where the loss is used to optimize the learning rate (Takase et al., 2018).

All these works lay foundations for our sample-based implementation. However, in addition to the training loss, we also utilize such model configuration information as the number of model layers and number of heads at each layer. Such information serves as reward factors that help decide the best model configuration for efficient training and inference by enabling adaptive underlying network updates. Furthermore, our work take a more comprehensive approach to multi-head attention pruning, as the research study the effect of head combinations instead of the effect of individual ones.

3. BACKGROUND

In this section, our paragraph describe the two algorithms that serve as the foundation for our methodology: Uniform Confidence Bound and Thompson Sampling Semi-Bandits.

3.1 Uniform Confidence Bound

Contextual bandit is used to find the actions for *d*-dimensional input samples X = {x_1,…, x_n} ∈ R^d given a finite arm set Z. The aggregated vector corresponding to arm γ with arm feature vector z_γ is constructed as $w_{i,\gamma}$ = [x_i ; z_γ]∈ R^{2d} (for simplicity assume that both have dimension *d*). The aggregated vector corresponding to arm γ with arm feature vector z_γ is constructed as $w_{k,\gamma}$ = [x_k ; z_γ]. The expected reward $r_{k,\gamma}$ with trainable $\theta_\gamma \in$ R^{2d} is defined as follows (Li et al., 2010):

$$E[r_{k,\gamma}|\mathbf{w}_{k,\gamma}] = \mathbf{w}_{k,\gamma}^T\theta_\gamma.$$

Let $D_{i,\gamma}$ ∈ $R^{k\times 2d}$ be the design matrix reflecting prior k sample i and $C_{i,\gamma}$ ∈ $R^{k\times 2d}$ be the reward matrix obtained from the previous observations. Following regression the optimal parameters $\theta_{i,r}$ read

$$\theta_{i,\gamma} = (\mathbf{D}_{i_\gamma}^T\mathbf{D}_{i,\gamma} + \lambda\mathbf{I})^{-1}\mathbf{D}_{i,\gamma}^T\mathbf{C}_{i,\gamma},$$

where λ is the ridge regularization factor.

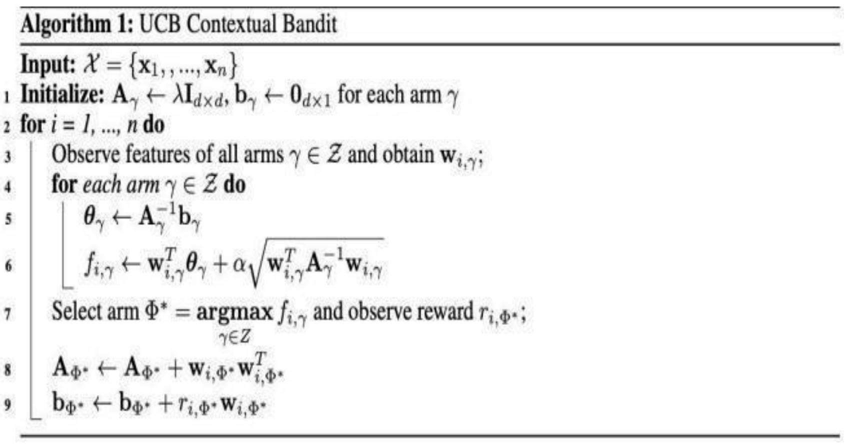
UCB contextual bandit is summarized in Algorithm 1.

Initialize: $A_\gamma \leftarrow \lambda I_{d\times d}$, $b_\gamma \leftarrow 0_{d\times 1}$ for each arm γ
Observe features of all arms $\gamma \in Z$ and obtain $w_{i,\gamma}$;

$$\theta_\gamma \leftarrow \mathbf{A}_\gamma^{-1}\mathbf{b}_\gamma$$
$$f_{i,\gamma} \leftarrow \mathbf{w}_{i,\gamma}^T\theta_\gamma + \alpha\sqrt{\mathbf{w}_{i,\gamma}^T\mathbf{A}_\gamma^{-1}\mathbf{w}_{i,\gamma}}$$

Select arm $\Phi^* = \underset{\gamma\in Z}{\mathbf{argmax}}\, f_{i,\gamma}$ and observe reward r_{i,Φ^*}

$$\mathbf{A}_{\Phi^*} \leftarrow \mathbf{A}_{\Phi^*} + \mathbf{w}_{i,\Phi^*}\mathbf{w}_{i,\Phi^*}^T$$
$$\mathbf{b}_{\Phi^*} \leftarrow \mathbf{b}_{\Phi^*} + r_{i,\Phi^*}\mathbf{w}_{i,\Phi^*}$$



```
Algorithm 1: UCB Contextual Bandit
Input: $\mathcal{X} = \{x_1, , ..., x_n\}$
Initialize: $A_\gamma \leftarrow \lambda I_{d\times d}$, $b_\gamma \leftarrow 0_{d\times 1}$ for each arm $\gamma$
for $i = 1, ..., n$ do
    Observe features of all arms $\gamma \in \mathcal{Z}$ and obtain $w_{i,\gamma}$;
    for each arm $\gamma \in \mathcal{Z}$ do
        $\theta_\gamma \leftarrow A_\gamma^{-1} b_\gamma$
        $f_{i,\gamma} \leftarrow w_{i,\gamma}^T\theta_\gamma + \alpha\sqrt{w_{i,\gamma}^T A_\gamma^{-1} w_{i,\gamma}}$
    Select arm $\Phi^* = \underset{\gamma\in Z}{\mathbf{argmax}}\, f_{i,\gamma}$ and observe reward $r_{i,\Phi^*}$;
    $A_{\Phi^*} \leftarrow A_{\Phi^*} + w_{i,\Phi^*} w_{i,\Phi^*}^T$
    $b_{\Phi^*} \leftarrow b_{\Phi^*} + r_{i,\Phi^*} w_{i,\Phi^*}$
```

Algorithm 1. UCB contextual bandit

3.2 Thompson Sampling Semi-Bandits

In the architecture, loss is directly related to rewards of all contextual bandits, resulting in the obstacles in quantifying combinatorial head contribution due to the more significant parts that dynamic layers play. Besides, it is much tougher to measure combinatorial dynamics by modelling continuous value compared with the concrete number selections of numerical dynamics. To address the issues, our work simplify the observations of superarm as binary values, whose feedback is polarily classified as one otherwise zero. To this end, our work are able to detect the most efficient head combinations by dynamically clustering all heads into two groups.

Let us assume that reward r_i ∈ 0, 1 is Bernoulli distributed with $Pr(r_i = 1|w_{i,\gamma})$ (Wang et al., 2017)

Let us initialize the mean vector as $g_{i,\gamma}$ = $I_{2d\times 1}$ and inverse covariance as $s_\gamma = 0_{2d\times 2d}$. In order to learn the likelihood probability of reward 1, the learnable weight vector $\mathrm{p}_{i,\gamma} \in \mathrm{R}^{2d}$ approximates $E_{i,\gamma} = Pr(r_i = 1|\mathrm{w}_{i,\gamma})$ as

$$\mathbf{p}_{i,\gamma} = N(\mathbf{g}_\gamma, \mathbf{s}_\gamma^{-1}),$$
$$E_{i,\gamma} = \sigma(\mathbf{p}_{i,\gamma}^T \mathbf{w}_{i,\gamma}),$$

where σ is the sigmoid function and N is the normal distribution. The learner chooses K actions out of the whole set Z, thus the optimal combinatorial subset is selected according to the K highest $E_{i,\gamma}$. Here denote the optimal subset as

$$\mathbf{\Phi}^*: \{\gamma_1, \gamma_2, \ldots, \gamma_K\} \subset \mathcal{Z}.$$

After the TSB is executed, our model receive a binary reward r_i as the feedback of Φ^* to refresh N. The estimation on refreshed g_γ is to find the maximizer (Bishop, 2006):

$$\underset{\mathbf{g}_\gamma}{\mathbf{argmax}}\left(\frac{\lambda}{2}\sum_{j=1}^{2d} \mathbf{s}_\gamma\, (\mathbf{p}_\gamma - \mathbf{g}_\gamma)^2 + \sum_{i=1}^{n} log(\sigma(r_i \mathbf{p}_\gamma^T \text{\bmw}_{i,\gamma}))\right),$$

where j is the index of feature dimension of the vector, λ is the regularization parameter. Then update s_γ

$$\mathbf{s}_\gamma = \mathbf{s}_\gamma + \sum_{i=1}^{n} \sigma(\mathbf{p}_\gamma^T \mathbf{w}_{i,\gamma})(1 - \sigma(\mathbf{p}_\gamma^T \mathbf{w}_{i,\gamma}))\mathbf{w}_{ij,\gamma}^2.$$

4. PROPOSED METHODOLOGY

In this section, we will discuss our sample-based search on transformer attention space. Our model begin by recalling the multi-head attention and head masking mechanism. Then, our paper introduce the definitions of states, actions for deciding heads and layers. Lastly, our paper demonstrate how to formulate rewards to update all contextual bandits based on the searching routine.

Our core objective is to flexibly fabricate multiple bandits into the transformer framework to search for optimal network update routine given each input. Our work design two search modes to demonstrate DHT training, constructing UCB as numerical dynamics and TSB as combinatorial dynamics for selecting the number of heads (layers) and head combinations, respectively.

4.1 Multi-Head Attention and Pruning

Given the input feature vector $\mathrm{x}_l \in \mathrm{R}^d$ at l-th layer of an input sample $\mathrm{x} \in \mathrm{R}^d$, the pruned multi-attention layer function $f_l(\mathrm{x}_l)$ can be formulated as follows (Vaswani et al., 2017; Michel et al., 2019)

$$f_l(\mathbf{x}_l) = \sum_{h=1}^{N_h} \xi_h \mathrm{Att}_{\mathrm{W_{k,h}}, \mathrm{W_{q_l,h}}, \mathrm{W_{v_l,h}}, \mathrm{W_{o_l,h}}}(\mathbf{x}_\mathrm{l})$$

where Att is the single-head attention parameterized by $W_{k,h}, W_{q_l,h}, W_{v_l,h}, W_{o_l,h} \in \mathbb{R}^{d\times d}$, N_h is the number of full heads, and ξh is the masking variable with values in {0,1} and each value corresponds to heads in relative positions. Simplifying the notation of the single attention for a head h in Equation ([multi]) as Att_h our model define the combinatorial heads as $\mathbf{\Pi}_l = \{\xi_1 \mathrm{Att}_1, \ldots, \xi_{\mathrm{N_h}} \mathrm{Att}_{\mathrm{N_h}}\}$. If all ξhs are equal to 1, the formula corresponds to the multi-attention layer in a vanilla Transformer model. If some ξh have values of 0, the corresponding heads are pruned in the layer. The preserved number of heads is hence $H_l = \sum_{h=1}^{N_h} \xi_h$

4.2 Architecture Space Search

Let the maximum number of layers a Transformer can have be NL, sample x only needs to go through $L \leq NL$ layers as search limit during training or inference to obtain good performance.

To balance the trade-off between searching comprehensiveness and computational efficiency, we present two searching modes to enhance searching flexibility: tree search and integrated search, with the former focusing on comprehensiveness and the latter efficiency. In the case of tree search, there are two UCBs: UCBL and UCBH to make decisions on L and Hl, respectively. UCBL is executed once to determine L, namely the number of layers a sample x needs to go through. UCBH and TSB are utilized to jointly determine the number of heads to preserve and which heads to preserve for each layer, respectively. To speed up the training, our work employ UCBH and TSB not for every single layer l, but rather for a group of c adjacent layers, to determine the pruning information for all c layers at once. In the case of integrated search, instead of determining the number of layers L and the number of heads at each layer Hl separately, our work use one complex UCBcp to determine L and H at once, i.e. the number of heads to keep at each layer is the same. However, our work still apply TSB on every c adjacent layers to determine the combination of heads l to keep.

The optimal arms are $\Phi L*$, $\Phi H,*$ or $\Phi cp*$, and the optimal superheads are $\Phi ts,l*$ at l-th depth. To obtain the arm features, our model pass all samples through all layers of the full model once, without back propagation. For each option of layer numbers in our DHT model (i.e. L=3, L=6, etc.), our model take all sample outputs of the model (feature representation matrices) and compute an average as the arm feature for UCBL

corresponding to the layer depth. e.g., the arm feature of L=3 is the average of all samples' output matrices at the end of 3rd layer, the arm feature of L=6 is the average of all samples' output matrices at the end of 6th layer. Similarly, to obtain UCBH, each sample is randomly and evenly assigned a head number given each layer. Our model average all sample feature representation matrices as the arm feature corresponding to the same head number. If the searching mode is complex, the arm feature of UCBcp is the average of all samples' feature vector at both corresponding same layer depth and head number, such as the arm feature of L=3,H=8 is the average of all feature representation matrices at 3rd layer with 8 preserved heads. In terms of the arm feature of TSB, our model collect all feature representation matrices at all layers with all kinds of head numbers, and split each vector into NH sub-matrices along the embedding dimension space. Each averaged sub-matrix corresponds to a specific head, formulating the arm feature of the corresponding TSB arm feature. Thus there are NH TSB arms in total.

In the following, our work illustrate both searching modes using an example sample x. Because our model need some contextual information of x during each training epoch, the first b layers of the model are kept unpruned. All remaining layers are dynamic and c is a hyperparameter serving as the number of adjacent layers UCBH and TSB jointly operate on to determine the number of heads and the combination of heads to keep. wL, wH, are the aggregated vectors at l-th layer for UCBL and UCBH.

4.2.1 Tree Search

Step 1 $l = b$

x initially passes through the first b layers to reach state $S1$. Our work combine xl with the arm feature vector to formulate a set of aggregated vector wL. The weights of b layers are inherited from t−1. UCBL selects $\Phi L*$ to find L, the number of layers x needs to pass through.

Step 2 $l = b$

x repasses through the first b layers to reach $S2$,0, during which layer weights are updated through back-propagation. Our model continue to execute UCBH to obtain all Hl for the next group of of c layer(s). Similarly, our model construct the aggregated vector wH,l = [xl ; ZH,l; el], where el is a column one-hot vector to mark the layer depth. According to ΦH,$l*$, TSB is employed to select Φts,$l*$ to obtain Πl. Note that for all upcoming c layer(s), the heads to be pruned are in the same relative positions.

Step 3 $l = b+oc$

Our model increment o by one, where o is the number of times UCBH and TSB have been executed. If l<b+L, go back to step 2 and repeat UCBH and TSB; otherwise, go to step 4.

Step 4 $l = b+L$

Our model observe the reward $r\Phi L*$,$r\Phi H$,$l*$ and $r\Phi ts$,$l*$ and begins the back-propagation.

4.2.2 Integrated Search

Step 1 and 2 l=b

Similar to step 1 and 2 in the tree search, our model first utilize UCBcp to obtain both L and H for x at $S1$. Then, x proceeds through the first b layer(s) again and stop at $S2$,0.

Step 3 $l = b+oc$

Our model increment o by one and, if l<b+L, go back to step 2 and run the TSB and obtain the head combinations Πl with H heads; otherwise, our model move forward to step 4.

Step 4 $l = b+L$

Our model observe the reward $r\Phi cp*$ and $r\Phi ts$,$l*$ and begins the back-propagation.

4.3 Reward Formulation

According to the rule of Markov Decision Process, the current state depends on the previous one. Therefore, our model formulate rewards through comparing loss and the number of heads and layers between the current batch samples and the past batch samples together to maximize the cumulative training rewards. Given a sequential input batch samples X = {$x1$,…, xi} at time stamp t, Lt,i and Ht,i,l, the number of layers and heads selected for xt,i, serve as the search limit, and δL,t,i, defined below, is the product of loss and Lt,i.

$$\delta_{L,t,i} = loss(\mathbf{x}_{t,i}) \cdot L_{t,i}.$$

The reward $r\Phi L$,,$*$ at timestamp t for i-th sample is

$$r_{\Phi^*_{L,t,i}} = \alpha \cdot \frac{(\sum_{j=1}^{bs} \delta_{1,t-1,j})/bs}{\delta_{t,i}},$$

where α is the positive hyper-parameter. A smaller δL,t,i in Equation ([eq:r2]) denominator represents fewer loss (better training results) and fewer layers (fewer costs) for the sample at the ongoing round in relative to averaged gains during the last round, so the reward will be higher to encourage this trend.

Similarly, with $\beta 1$ and βcp as the hyper-parameters for head tuning, δH,t,i,l and δcp,t,i,l as the products of loss, number of layers and heads at the l-th layer for UCBH and UCBcp, defined below

$$r_{\Phi^*_{H,t,i,l}} = \beta_1 \cdot \frac{(\sum_{i=1}^{bs} \delta_{2,t-1,i,l})/bs}{\delta_{2,t,i,l}},$$

$$r_{\Phi^*_{cp,t,i}} = \beta_{cp} \cdot \frac{(\sum_{i=1}^{bs} \delta_{cp,t-1,i})/bs}{\delta_{cp,t,i}}.$$

Inspired by a work that illustrates some positional heads steadily contribute significantly and others work incrementally or negatively (Voita et al., 2019), we simplify our observations of combinatorial heads $\mathbf{\Pi}_{t,i,l}$ as binary values, where 1 is more effective and 0 is less effective when compared with superheads in the previous batch $\mathbf{\Pi}_{t-1,i,l}$.

$$r_{\Phi^*_{ts,i,l}} = \{\begin{matrix} 0 & r_{\Phi^*_{H,t,i}}(r_{\Phi^*_{cp,t,i,l}})>1 & , \\ 1 & r_{\Phi^*_{H,t,i}}(r_{\Phi^*_{cp,t,i,l}}) \le 1 & . \end{matrix}$$

x passes the first b layer weights at last round and stops at $S1$;
Run UCB_L $andobtain\ L$←$\Phi L*\in ZL$;

x repasses the first b layer weights and stops at $S2$,0;
Run UCB_H $toobtain\ Hl$←$\Phi H,l*\in ZH$;

Run TSB to obtain the head combination $\mathbf{\Pi}_l \leftarrow \Phi^*_{ts,l}$ heads;

l←l+c

Sample(s) proceed(s) c layer(s) with \emph$Hl$ heads and stops at $S2$,;

x finishes (b+L) layer(s) with adaptive heads and obtain the loss of x;

Receive rewards: $r_{\Phi^*_L}, r_{\Phi^*_{H,l}}\ r_{\Phi^*_{cp}}$ and $r_{\Phi^*_{ts,l}}$ as Equation ([eq:r2]) - ([eq:r6]); Update all contextual bandits;

5. IMPLEMENTATION

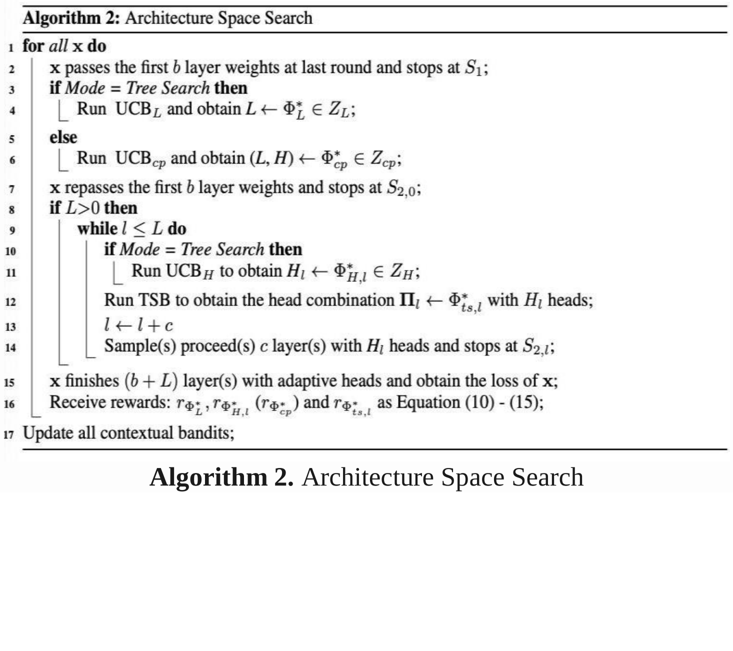
During training, our model first construct arm feature vectors for all contextual bandits. Then, our model train all samples dynamically by utilizing sample-dependent layer and head combination information provided by UCBL, UCBH and TSB, as described in Section 4. Because each sample corresponds to different number of layers, as is determined by UCBL, our model cannot directly group samples into batches since our model will not have known the appropriate model with the right L layers for those samples. Therefore, our model implement a queuing strategy that categorizes each sample into a particular queue based on the optimal L=$L*$, determined for it enabling the parallel execution of UCBs and TSB.

Let Q={$QL1$,...,QLn} be a set of queues corresponding to a set of layer numbers {$L1$,...,Ln}, where each queue has a size of q. After the execution of UCBL or UCBcp, xt, with Ln is allocated to the corresponding queue QLn. When the number of samples inside QLn reaches q, all are dequed to form a batch. Otherwise, samples will temporarily stay in the queue and wait for more corresponding samples to be added. There is no training or inference until one queue is full at $S1$.

It is probable that the samples in one queue come from different time rounds, so our model just use the historical weights of the first b-th layer to obtain $S1$ without updating any neutral networks until the training batch is created. $S2$,0 is reached upon all samples having passed the first b-th layers. In this way, our model stabilize the accuracy of gradient descent and ensure the training effects. Our queuing strategy is summarized in Algorithm 3.

Allocate xi,t to join QhLn at $S1$; All samples in QLn are released and proceed as line 7 - 16 in Algorithm 2;

$$len(\mathbf{Q}_{L_n}) \leftarrow 0$$



```
Algorithm 2: Architecture Space Search
1  for all x do
2      x passes the first $b$ layer weights at last round and stops at $S_1$;
3      if Mode = Tree Search then
4          Run UCB$_L$ and obtain $L \leftarrow \Phi^*_L \in Z_L$;
5      else
6          Run UCB$_{cp}$ and obtain $(L, H) \leftarrow \Phi^*_{cp} \in Z_{cp}$;
7      x repasses the first $b$ layer weights and stops at $S_{2,0}$;
8      if $L>0$ then
9          while $l \le L$ do
10             if Mode = Tree Search then
11                 Run UCB$_H$ to obtain $H_l \leftarrow \Phi^*_{H,l} \in Z_H$;
12             Run TSB to obtain the head combination $\mathbf{\Pi}_l \leftarrow \Phi^*_{ts,l}$ with $H_l$ heads;
13             $l \leftarrow l + c$
14             Sample(s) proceed(s) $c$ layer(s) with $H_l$ heads and stops at $S_{2,l}$;
15     x finishes $(b + L)$ layer(s) with adaptive heads and obtain the loss of x;
16     Receive rewards: $r_{\Phi^*_L}, r_{\Phi^*_{H,l}}$ $(r_{\Phi^*_{cp}})$ and $r_{\Phi^*_{ts,l}}$ as Equation (10) - (15);
17 Update all contextual bandits;
```

Algorithm 2. Architecture Space Search

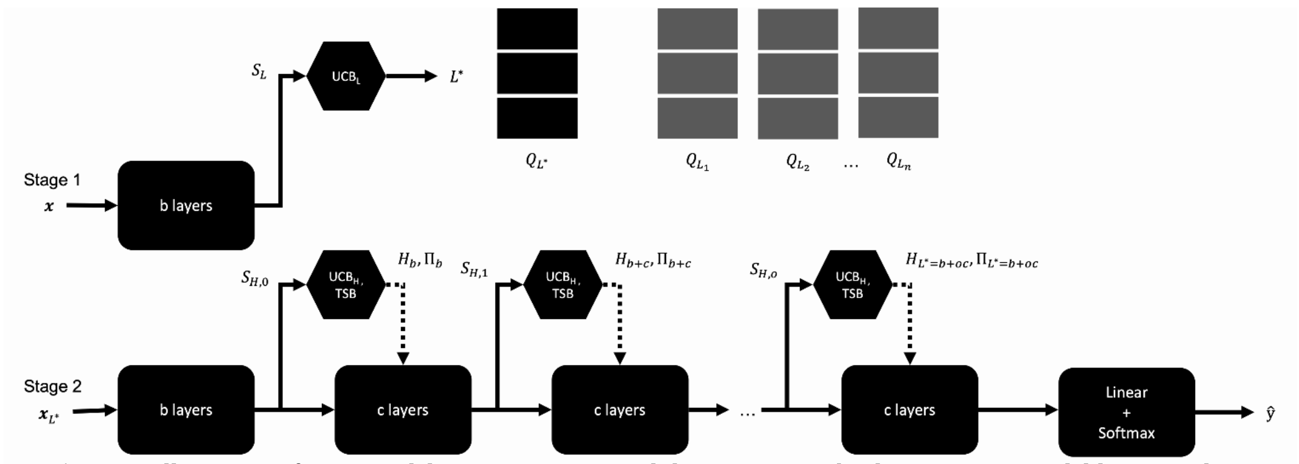
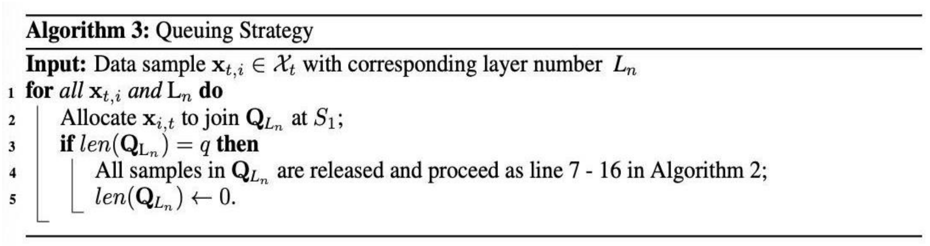


Figure 1. Illustration of DHT model training. Here, a solid arrow means the data is pass to model layers and a a dotted arrow means the information determined by UCBs and TSBs at a particularstate S is used to formulate the upcoming c layers in the dynamic model. Our model first determinethe number of layers needed for a particular sample x, L^* in our example. Then, our model put $x$into a queue corresponding to the required number of layers L^*. Once a queue for L^* is filled up, our model compile all samples in the queue to form a batch $\mathbf{x}_{L^*}$. Afterwards, our model pass $\mathbf{x}_{L^*}$ through the first b layers in order to obtain such information as the number of heads and head combinationsfor the upcoming c layers. Our model repeat the process o time such that $L = b + oc$. Lastly, our model update our model with rewards computed based on y hat.

Illustration of DHT model training. Here, a solid arrow means the data is pass to model layers and a a dotted arrow means the information determined by UCBs and TSBs at a particular state S is used to formulate the upcoming c layers in the dynamic model. Our model first determine the number of layers needed for a particular sample x, L^* in our example. Then, our model put x into a queue corresponding to the required number of layers L^*. Once a queue for L^* is filled up, our model compile all samples in the queue to form a batch bf{x}_{L^*}. Afterwards, our model pass bf{x}_{L^*} through the first b layers in order to obtain such information as the number of heads and head combinations for the upcoming c layers. Our model repeat the process o time such that L=b+oc. Lastly, our model update our model with rewards computed based on y hat. Illustration of DHT model training. Here, a solid arrow means the data is pass to model layers and a a dotted arrow means the information determined by UCBs and TSBs at a particular state S is used to formulate the upcoming c layers in the dynamic model. Our model first determine the number of layers needed for a particular sample x, $L*$ in our example. Then, our model put x into a queue corresponding to the required number of layers $L*$. Once a queue for $L*$ is filled up, our model compile all samples in the queue to form a batch $xL*$. Afterwards, our model pass $xL*$ through the first b layers in order to obtain such information as the number of heads and head combinations for the upcoming c layers. Our model repeat the process o time such that L=b+oc. Lastly, our model update our model with rewards computed based on y hat.



```
Algorithm 3: Queuing Strategy
Input: Data sample $\mathbf{x}_{t,i} \in \mathcal{X}_t$ with corresponding layer number $L_n$
1 for all $\mathbf{x}_{t,i}$ and $\mathrm{L}_n$ do
2     Allocate $\mathbf{x}_{i,t}$ to join $\mathbf{Q}_{L_n}$ at $S_1$;
3     if $len(\mathbf{Q}_{\mathrm{L}_n}) = q$ then
4         All samples in $\mathbf{Q}_{L_n}$ are released and proceed as line 7 - 16 in Algorithm 2;
5         $len(\mathbf{Q}_{L_n}) \leftarrow 0$.
```

Algorithm 3. Queuing Strategy

6. EXPERIMENTS

In this section, our work first describe our experiment configurations. Then, the section describes the baseline models to be compared with and the dataset on which our work run the experiments, and analyze the experiment results.

6.1 Reward Formulation

Our work experiment with four configurations under the standard 12-layer and 12-head framework. The embedding size is 768 and sequence length varies among datasets. The hyperparametersc$b$ and c are 3. Along with the preset configurations for L, this guarantees that $L-b$ is dividable by o, the number of times UCBH and TSB get executed.

Tree-search DHT (T-DHT): Following the tree search mode, the options for the number of T-DHT layer (L) are 3,6,9,12, and and the options for number of heads (H) at each layer are 8,10,12, each corresponding to 70, 28, 1 type(s) of head combination(s).

Specialized DHT (S-DHT): Following the integrated search mode, this configuration only applies specialized dynamics to DHT to explore the least contextual bandits execution. The available options of UCBcp are 3L12H, 6L12H, 9L12H, 12L8H, 12L10H, 12L12H. Specifically, 3L12H means the model has three layers and each layer has twelve heads.

Full DHT (F-DHT): Following the integrated search mode, this configuration concentrates on full dynamics for DHT. The available options of UCBcp are 3L12H, 6L8H, 6L10H, 6L12H, 9L8H, 9L10H, 9L12H, 12L8H, 12L10H, 12L12H.

Partial-frozen DHT (P-DHT): First, model training follows the configuration of Full DHT (F-DHT) for two epochs. Afterwards, our work train all but the last two layers by freezing their weights. The available options of UCBcp are 1L8H+2L12H, 1L10H+2L12H, 1L12H+2L12H, 4L8H+2L12H, 4L10H+2L12H, 4L12H+2L12H, 7L6H+2L12H, 7L9H+2L12H, 7L12H+2L12H, 10L8H+2L12H, 10L10H+2L12H, 10L12H+2L12H. Specifically, 1L8H+2L12H means there are three layers in total, where the first layer has eight heads and the last two layers have twelve heads each.

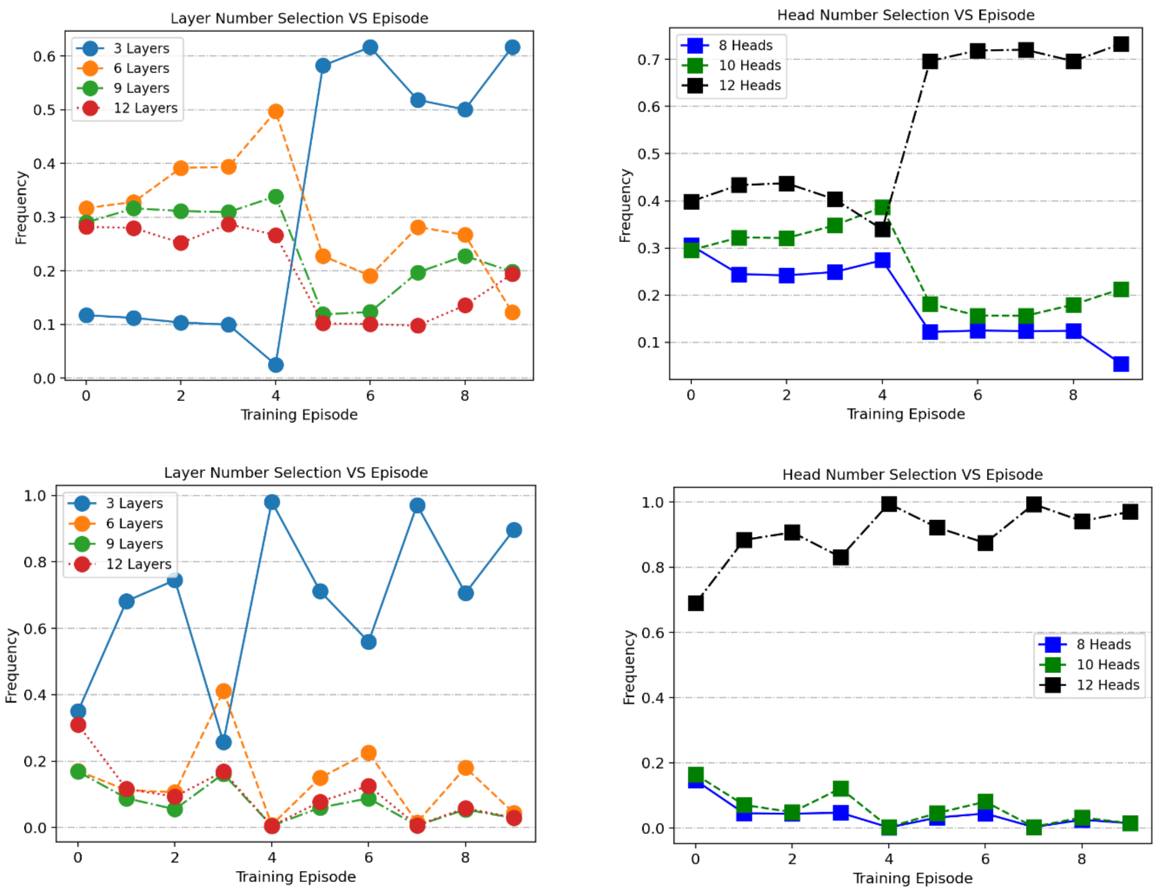


Figure 2. The trends of layer number and head number throughout training. It could be observed that the selection of 3-layer configuration with 12 heads on each layer becomes dominant as training progresses forward. This indicates that our DHT models indeed adapt themselves to reduce com- putational cost for training and inference (From left upside to right upside: P-DHT Dynamics for **Ag. News**: Layer number, head number selection; From left downside to right donwside: S-DHT Dynamics for **Ag. News**: Layer number, head number selection).

6.2 Baselines, Datasets and Analysis

Our work use four text classification datasets (Zhang et al., 2016) to evaluate the DHT configurations: **Dbpedia Dataset (Dbpedia)**, **Ag News Dataset (Ag. News)** , **Yelp Review Polarity Dataset (Yelp.P)** and **Yelp Review Full Dataset (Yelp.F)**.**Ag. News** contains 120,000 training datasets and 7,600 testing datasets with four classes. Our work set 128 as sequence length. **Dbpedia** contains 560,000 training datasets and 70,000 testing datasets with fourteen classes. Our work set 128 as sequence length. **Yelp. P** contains 560,000 training datasets and 38,000 testing datasets with two classes. Our work set 300 as sequence length. **Yelp. F** contains 650,000 training datasets and 50,000 testing datasets with five classes. Our work set 300 as sequence length.

Floating-point operation (FLOP) is a measure of the computational complexity of models. Our work quantify computational efficiency using a FLOP ratio regarding the full vanilla BERT with 12 layers and 12 heads as the baseline. That is, if our configuration is twice as efficient as the vanilla BERT model, then our FLOP ratio is 0.50x, etc.. In addition, our work record F1-score to characterize the accuracy of classification tasks.

Our work compare DHT from scratch against three baselines, including sample-based lightening, layer only distillation and the vanilla BERT model. For the FLOP ratio comparison, DHT is compared with baseline 1 (BL1) for training and with all baselines for inference. Baseline 2 and 3 are trained with full layers and heads. All FLOP ratios are estimated based on the layer/head number selection in comparison with a model with full layers and full heads.

BL1: Full Vanilla BERT (**12-VaBERT**) with 12 layers and 12 heads. Our work define our benchmark FLOP ratio (1.00x) in training and inference based on its FLOP count.

BL2: DistilBERT (Sanh et al., 2020) released by Huggingface with 1, 3, 6 distilled layers, where each configuration of distilled layers corresponds to FLOP ratio of 0.08x, 0.25x, 0.50x regarding **12-VaBERT**. This is the most classical methods for reducing BERT inference complexity. Our work denote them as as **1, 3, 6-DisBERT**, respectively.

BL3: Adaptive sample-based **FastBERT** using knowledge distillation. The decision on depth selection relies on the preset inference speedup: 0.1x, 0.5x, and 0.8x (approximately FLOP ratios of 0.08x, 0.25x, 0.50x regarding **12-VaBERT**, depending on different datasets). Our work denote them as **1, 3, 6-FastBERT**, respectively.

Table 1. Training Comparisons: Inference F1 scores (F1.) and ratios of the FLOP count of our four DHT configurations to that of BL1.

Models	**Datasets**	**Ag. News**		**Dbpedia**		**Yelp. P**		**Yelp. F**	
		F1.	FLOPs	F1.	FLOPs	F1.	FLOPs	F1.	FLOPs
BL 1	12-VaBERT	0.944	1.00x	0.993	1.00x	0.960	1.00x	0.659	1.00x
DHT	T-DHT	0.945	0.42x	0.992	0.43x	0.958	0.47x	0.635	0.45x
	P-DHT	0.935	0.59x	0.993	0.41x	0.955	0.44x	0.634	0.58x
	S-DHT	0.941	0.38x	0.994	0.39x	0.956	0.26x	0.632	0.29x
	F-DHT	0.943	0.59x	0.993	0.58x	0.957	0.29x	0.639	0.43x

Table 2. Inference Comparisons: F1 scores (F1.) and ratios of the FLOP count of our four DHT configurations to that of the BL1, BL2 and BL3

Models	**Datasets**	**Ag. News**		**Dbpedia**		**Yelp. P**		**Yelp. F**	
		F1.	FLOPs	F1.	FLOPs	F1.	FLOPs	F1.	FLOPs
BL 1	12-VaBERT	0.944	1.00x	0.993	1.00x	0.960	1.00x	0.659	1.00x
BL 2	6-DisBERT	0.946	0.50x	0.991	0.50x	0.953	0.50x	0.642	0.50x
	3-DisBERT	0.939	0.25x	0.990	0.25x	0.932	0.25x	0.635	0.25x
	1-DisBERT	0.928	0.08x	0.989	0.08x	0.916	0.08x	0.585	0.08x
BL 3	6-FastBERT	0.943	0.28x	0.992	0.10x	0.959	0.31x	0.659	0.95x
	3-FastBERT	0.931	0.10x	0.990	0.09x	0.960	0.16x	0.659	0.95x
	1-FastBERT	0.925	0.08x	0.990	0.09x	0.960	0.08x	0.647	0.46x
DHT	T-DHT	0.945	0.81x	0.992	0.76x	0.958	0.83x	0.635	0.87x
	P-DHT	0.935	0.39x	0.993	0.41x	0.955	0.28x	0.634	0.35x
	S-DHT	0.941	0.37x	0.994	0.44x	0.956	0.26x	0.632	0.26x
	F-DHT	0.943	0.35x	0.993	0.29x	0.957	0.31x	0.639	0.29x

From Table 1, most DHT training costs are lower than 0.5 in relative to the vanilla BERT model. In particular, the cost of S-DHT is even lower than 0.3 for **Yelp. P** and **Yelp. F**. That means, in theory, our DHT models can be trained more efficiently. From Table 2, DHT models also have competitive accuracy with significant complexity mitigation. Specifically, in comparison with baseline 2, our DHT models outperform the uniform-level knowledge distillation in both model complexity and inference accuracy. In comparison with baseline 3, our work have competitive inference performance in terms of more economic training costs. That is, even though our inference costs are higher than those of FastBERT, our accuracy does not decrease as dramatically. Integrating both inference and training costs, DHT are apparently more advantageous than DisBERT.

Furthermore, the mechanism of DHT is different from that of DisBERT and FastBERT, where the level of computational complexity is reduced in the beginning. DHT takes care of training effects and costs simultaneously and dynamically, so the accuracy loss is minimal even if tremendous speedup, such as 0.26x speedup for both training and inference of S-DHT for **Yelp. P** but accuracy loss is only 0.6% in relative to vanilla BERT, 0.29x speedup for training and 0.26x for inference of S-DHT for **Yelp. F** but only 4.1% accuracy loss, according to Table 2.

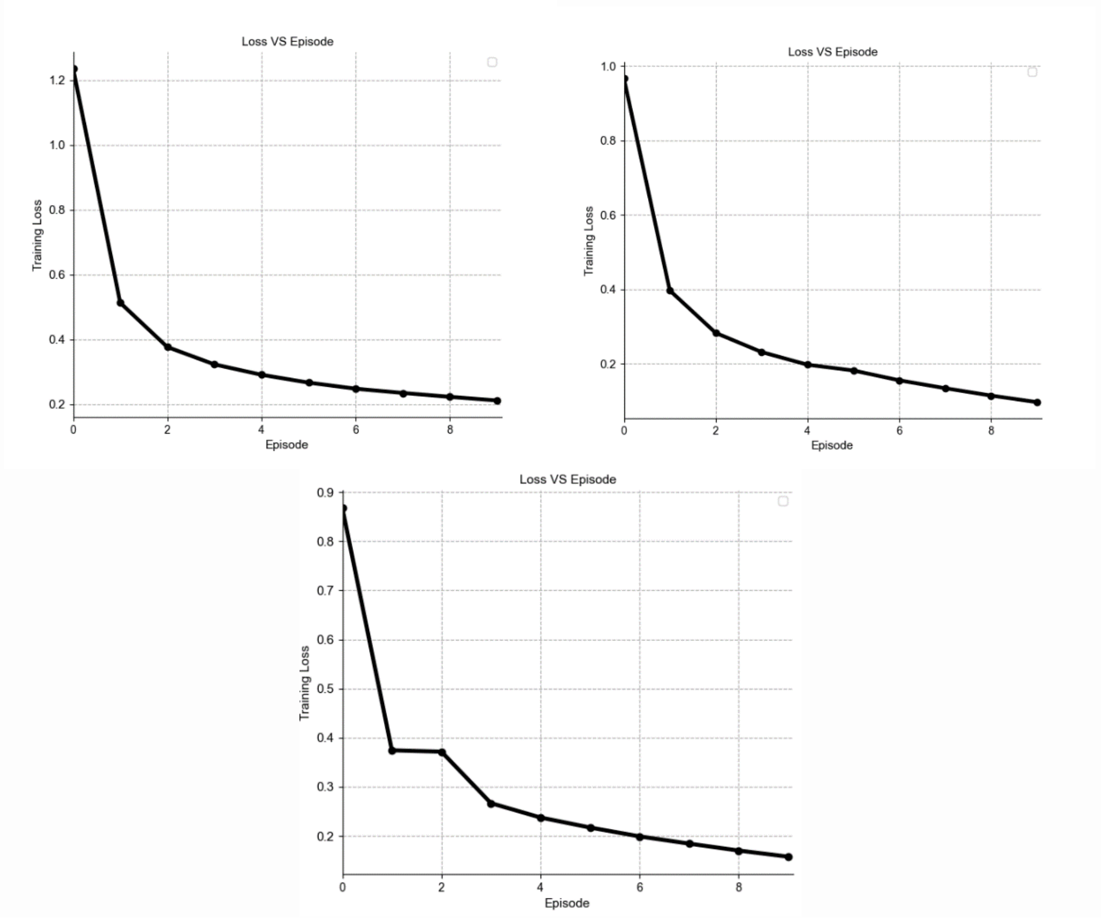


Figure 3. Loss Trends: P-DHT for Ag. News, S-DHT for Ag. News and F-DHT for Yelp. P

6.3 Dive into the Dynamics: Layer, Head and Training Stability

From Figure 2, the 3-layer option is selected less frequently in the beginning and more frequently after 5 episodes. In contrast, other layer options demonstrate downtrends as training progresses forward. This indicates that, throughout time, the less complex option regarding the number of layers is chosen by the trained UCBL model, so training costs are gradually decreasing. With respect to head selection, the 12-head option is dominant towards the end of the training.

From Figure 3, although the layer routine is adaptive across the training, the loss maintains the downtrend and then converges smoothly. It proves that the dynamic training is effective for mitigating complexity without hurting training stability.

7. CONCLUSIONS

Our work propose to search optimal layers and heads hierarchically via contextual bandits. Contextual bandits are deployed at low costs and successfully reduce the computational complexity for both model training and inference while maintaining the model performance. The objective of DHT is to maintain the model performance with self-adaptive training and inference mechanism, which takes care of both the goal of automatic machine learning (AutoML) and model compression. Our work investigate many aspects in the paper such as dynamic weights training, weight pruning and sample-based training.

Different from traditional network compression methods, our approach is customized by layerwise and headwise dynamics with more sample-oriented flexibility, which means less inference accuracy loss. On the other hand, the down-to-top order optimization is much more efficient than top-to-down streaming in NAS. In the future, the team hope to extend this methodology into more deep learning frameworks.

7.1 Sampling Methods

In natural language processing, the input has three dimensions, i.e. batch size bs, sequence length sl and embedding size emb, our sampling is based on one

instance (bs=1). Because each sample corresponds to sl×emb features, keeping all features can be very memory expensive. Therefore, for all contextual bandits, we use mean pooling as the sampling method to generate context or arm vectors in order to reduce the memory footprint.
In terms of either sl or emb, we perform mean pooling iteratively and interchangeably. The elements in one dimension is sequentially and evenly split into: 1, 2, 4, 6, 8, ..., $2n$ parts. Depending on the choice of n and initial value m for i, we have $\sum i=mn2i$ sampled features for this dimension. For example, if we are sampling for the sequence length (sl) dimension and n=2 and m=0, the new feature is composed of seven sampled vectors: mean(sl1,1), mean(sl2,1), mean(sl2,2), mean(sl4,1), mean(sl4,2), mean(sl4,3), mean(sl4,4). There are two values in each subscript. The first one indicates for the selected dimension (i.e. sl) how many parts the original input is split, and the second one indicates for which segment of the split input we compute the mean value for.

In our experiment, we first sample from sl and then sample from emb. Specifically, for UCBL and UCBcp we first sample (1+2)=3 sl features and 4+8+16=26 emb features; for UCBH we first sample (1+2)=3 sl features and 2+4+8=14 emb features; for TSB we first sample (1)=1 sl features and 4+8=12 emb features. As a result, the dimension of the sampled UCBL and UCBcp context vectors is (1,3×36), the dimension of the sampled UCBH context vector is (1,3×14) and the dimension of the sampled TSB context vector is (1,1×12).

7.2 Normalization Methods

In this end, the context vectors are sampled from instance representations in time step, so they appear in time series and rolling (running) normalization is better for fitting linear distributions of users. The normalization algorithm is introduced by.

7.3 Contextual Bandits Settings

We set the size of training sample vector of UCBL (UCBcp), UCBH and TSB as 168, 87, 13 respectively. The learning rate are adaptive to the types of datasets, we normally run the whole training process for 3 to 6 epochs. Each experiment is conducted on NVIDIA GTX 1080, GTX 2080, Tesla V100 16G GPU separately.

7.4 Supplementary Results and Analysis

We complement analysis on different configurations besides Section 6.3. S-IDHT in Figure 7 aligns with the trends with Figure 3 in content. From Figure 4 concerning F-IDHT, 3 layers is also selected mostly but the frequency is stable, 6 layers is secondary and stable as well. In terms of Figure 6 concerning T-DHT, 12 layers are the least one firstly and gradually dominant (≈ 1.0) after 4 episodes. With regards to head selection, 12 heads presents downtrend and 8 heads and 12 heads are in uptrend for F-IDHT. 8 heads and 12 heads are preferable by T-DHT after 3 episodes.

For TSB superhead selection, as long as the incomplete multi-heads attentions are chosen, two figures align with previous analysis that some heads are less contributive to model performances and maintain their roles across the training. From loss trends, the tendencies for two configurations are the same that training stability is not negatively affected by dynamic layer routines.

Acknowledgments: 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.

References:

Baid, G., Cook, D. E., Shafin, K., Yun, T., Llinares-López, F., Berthet, Q., Belyaeva, A., Töpfer, A., Wenger, A. M., Rowell, W. J., et al. (2023). DeepConsensus improves the accuracy of sequences with a gap-aware sequence transformer. *Nature Biotechnology, 41*(2), 232–238.

Bhandare, A., Sripathi, V., Karkada, D., Menon, V., Choi, S., Datta, K., & Saletore, V. (2019). Efficient 8-bit quantization of transformer neural machine language translation model. *arXiv*. https://arxiv.org/abs/1906.00532

Bishop, C. M. (2006). *Pattern recognition and machine learning* (Information science and statistics). Springer-Verlag.

Bosselut, A., Rashkin, H., Sap, M., Malaviya, C., Celikyilmaz, A., & Choi, Y. (2019). COMET: Commonsense transformers for automatic knowledge graph construction. *arXiv*. https://arxiv.org/abs/1906.05317

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv*. https://arxiv.org/abs/2005.14165

Chang, H. S., Learned-Miller, E., & McCallum, A. (2017). Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems, 30*, 1002–1012.

Chung, I., Kim, B., Choi, Y., Kwon, S. J., Jeon, Y., Park, B., Kim, S., & Lee, D. (2020). Extremely low bit transformer quantization for on-device neural machine translation. *arXiv*. https://arxiv.org/abs/2009.07453

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*. https://arxiv.org/abs/1810.04805

Elbayad, M., Gu, J., Grave, E., & Auli, M. (2020). Depth-adaptive transformer. *arXiv*. https://arxiv.org/abs/1910.10073

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv*. https://arxiv.org/abs/1510.00149

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv*. https://arxiv.org/abs/1506.02626

Hassani, A., Walton, S., Li, J., Li, S., & Shi, H. (2023). Neighborhood attention transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 6185–6194).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv*. https://arxiv.org/abs/1503.02531

Hoffman, J. (2019). Cramnet: Layer-wise deep neural network compression with knowledge transfer from a teacher network. *arXiv*. https://arxiv.org/abs/1904.05982

Hou, L., Huang, Z., Shang, L., Jiang, X., Chen, X., & Liu, Q. (2020). DynaBERT: Dynamic BERT with adaptive width and depth. *arXiv*. https://arxiv.org/abs/2004.04037

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., & Liu, Q. (2020). TinyBERT: Distilling BERT for natural language understanding. *arXiv*. https://arxiv.org/abs/1909.10351

Jo, J. Y., & Myaeng, S. H. (2020). Roles and utilization of attention heads in transformer-based neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 3404–3417). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.311

Kim, H., Khan, M. U. K., & Kyung, C. M. (2019). Efficient neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Korthikanti, V. A., Casper, J., Lym, S., McAfee, L., Andersch, M., Shoeybi, M., & Catanzaro, B. (2023). Reducing activation recomputation in large transformer models. In *Proceedings of Machine Learning and Systems* (Vol. 5).

Kumar, S., Parker, J., & Naderian, P. (2020). Adaptive transformers in RL. *arXiv*. https://arxiv.org/abs/2004.03761

Lee, E., & Hwang, Y. (2021). Layer-wise network compression using Gaussian mixture model. *Electronics, 10*(1). https://doi.org/10.3390/electronics10010072

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World Wide Web - WWW '10*. https://doi.org/10.1145/1772690.1772758

Li, Z., Zhang, Z., Zhao, H., Wang, R., Chen, K., Utiyama, M., & Sumita, E. (2021). Text compression-aided transformer encoding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1. https://doi.org/10.1109/TPAMI.2021.3058341

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Deng, H., & Ju, Q. (2020). FastBERT: A self-distilling BERT with adaptive inference time. *arXiv*. https://arxiv.org/abs/2004.02178

Lopes, A. T., De Aguiar, E., De Souza, A. F., & Oliveira-Santos, T. (2017). Facial expression recognition with convolutional neural networks: Coping with few data and the training sample order. *Pattern Recognition, 61*, 610–628.

Luo, J. H., Wu, J., & Lin, W. (2017). ThiNet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

McCallum, R. (1996). Hidden state and reinforcement learning with instance-based state identification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26*, 464–473. https://doi.org/10.1109/3477.499796

Michel, P., Levy, O., & Neubig, G. (2019). Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems* (Vol. 32, pp. 14014–14024). Curran Associates, Inc.

Michel, P., Levy, O., & Neubig, G. (2019). Are sixteen heads really better than one? *arXiv*. https://arxiv.org/abs/1905.10650

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., & Kautz, J. (2019). Importance estimation for neural network pruning. *arXiv*. https://arxiv.org/abs/1906.10771

Neill, J. O. (2020). An overview of neural network compression. *arXiv*. https://arxiv.org/abs/2006.03669

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., & Dean, J. (2023). Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems* (Vol. 5).

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv*. https://arxiv.org/abs/1910.01108

Shaheen, Z., Wohlgenannt, G., & Filtz, E. (2020). Large scale legal text classification using transformer models. *arXiv*. https://arxiv.org/abs/2010.128714

Sukhbaatar, S., Grave, E., Bojanowski, P., & Joulin, A. (2019). Adaptive attention span in transformers. *arXiv*. https://arxiv.org/abs/1905.07799

Takase, T., Oyama, S., & Kurihara, M. (2018). Effective neural network training with adaptive learning rate based on training loss. *Neural Networks, 101*, 68–78.

Tang, R., Lu, Y., Liu, L., Mou, L., Vechtomova, O., & Lin, J. (2019). Distilling task-specific knowledge from BERT into simple neural networks. *arXiv*. https://arxiv.org/abs/1903.12136

Ullrich, K., Meeds, E., & Welling, M. (2017). Soft weight-sharing for neural network compression. *arXiv*. https://arxiv.org/abs/1702.04008

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *arXiv*. https://arxiv.org/abs/1706.03762

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv*. https://arxiv.org/abs/1905.09418

Wang, B., Qiu, M., Wang, X., Li, Y., Gong, Y., Zeng, X., Huang, J., Zheng, B., Cai, D., & Zhou, J. (2019). A minimax game for instance-based selective transfer learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (pp. 34–43). Association for Computing Machinery. https://doi.org/10.1145/3292500.3330841

Wang, Y., Ouyang, H., Wang, C., Chen, J., Asamov, T., & Chang, Y. (2017). Efficient ordered combinatorial semi-bandits for whole-page recommendation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (pp. 2746–2753). AAAI Press.

Xiao, H., Li, L., Liu, Q., Zhu, X., & Zhang, Q. (2023). Transformers in medical image segmentation: A review. *Biomedical Signal Processing and Control, 84*, 104791.

Xiao, L., Wang, H., & Ling, N. (2019). Image compression with deeper learned transformer. In *Proceedings of the 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (pp. 53–57). https://doi.org/10.1109/APSIPAASC47483.2019.9023342

Zhang, L., Guo, L., Gao, H., Dong, D., Fu, G., & Hong, X. (2020). Instance-based ensemble deep transfer learning network: A new intelligent degradation recognition method and its application on ball screw. *Mechanical Systems and Signal Processing, 140*, 106681. https://doi.org/10.1016/j.ymssp.2020.106681

Zhang, X., Zhao, J., & LeCun, Y. (2016). Character-level convolutional networks for text classification. *arXiv*. https://arxiv.org/abs/1509.01626

Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. *arXiv*. https://arxiv.org/abs/1611.01578

Fanfei Meng
Department of Electrical and Computer Engineering,
Northwestern University,
Evanston, 60208, IL,
United States
fanfeimeng2023@u.northwestern.edu
ORCID 0009-0009-9272-0665

Lele Zhang
Inistitute of Computing Technology,
Chinese Academy of Science,
Beijing,
100190, China
zhanglele@ict.ac.cn
ORCID 0000-0001-9661-2543

Yu Chen
Inistitute of Computing Technology,
Chinese Academy of Science,
Beijing, 100190,
China
chenyu19s@ict.ac.cn
ORCID 0000-0002-9363-105X

Yuxin Wang
Department of Electrical and Computer Engineering,
Northwestern University,
Evanston, 60208, IL,
United States
yuxinwang2023@u.northwestern.edu