NEURAL ARCHITECTURE SEARCH FOR TRANSFORMER NETWORKS: A COMPREHENSIVE SURVEY AND ANALYSIS

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE AND SUSTAINABLE TECHNOLOGIES
VOL 1 (01), JULY-DECEMBER 2026

Arunendra singh,
Department of Computer Science and
Engineering JSS University, Noida
Email: arunendra.singh@jssuninoida.edu.in
Dr. Nitima Malsa
Department of Computer Science and
Engineering JSS University, Noida
Email: nitima.malsa@jssuninoida.edu.in

ABSTRACT

Transformer models have emerged as the dominant architecture for a wide spectrum of machine learning tasks, from natural language understanding to computer vision. Their success, however, heavily depends on meticulous architectural design, which demands both expert knowledge and extensive computational resources. Neural Architecture Search (NAS) of-fers a systematic way to automate the design of transformer networks, often yielding models that surpass hand-crafted coun-terparts. This paper presents a comprehensive survey of NAS techniques tailored for transformers, including BERT and Vi-sion Transformers. We first provide a unified perspective on the transformer architecture and its variants, then categorise existing search spaces into attention-only and hybrid convolution-attention spaces. We review the three main search paradigms – reinforcement learning, evolutionary algorithms, and one-shot differentiable methods – and discuss how hardware awareness is incorporated. A detailed comparison of over 40 state-of-the-art methods is presented, covering performance, parame-ter efficiency, and inference speed. We also outline a generic NAS process flow that can be adapted to any transformer-based application. Finally, we identify critical open challenges and propose promising directions for future research, including efficient search space design, low-cost performance proxies, and joint network-accelerator co-design.

Keywords: *Neural Architecture Search, Transformers, Vi-sion Transformers, BERT, Hardware-aware NAS, Self-Attention, Differentiable Search, Evolutionary Search.*

1. INTRODUCTION

The advent of the Transformer architecture [1] marked a paradigm shift in deep learning. Unlike recurrent networks, which process sequences sequentially, transformers rely on a self-attention mechanism that captures long-range dependen-cies in parallel, leading to substantial gains in both training efficiency and model capacity. Initially designed for machine translation, transformers quickly became the backbone of modern natural language processing (NLP), with models such as BERT [2] and GPT [3] achieving human-level performance on a variety of benchmarks.

The success of transformers has not been confined to language. The Vision Transformer (ViT) [4] demonstrated that an image can be treated as a sequence of patches, and that a pure transformer can outperform convolutional neural networks (CNNs) on large-scale image classification. Since then, a flurry of ViT variants [5], [6] have further improved accuracy and efficiency, making transformers the new state-of-the-art in computer vision as well.

Despite their remarkable performance, designing a trans-former architecture for a specific task remains a formidable challenge. The number of design choices – including the num-ber of layers, the number of attention heads, the embedding dimension, the feed-forward expansion ratio, and the patch size for vision – is vast, and the optimal configuration varies across datasets and hardware platforms. Traditional manual design relies on intuition, heuristic rules, and extensive trial-and-error, which is time-consuming and often suboptimal.

Neural Architecture Search (NAS) [7] has emerged as a powerful solution to this problem. NAS automates the dis-covery of high-performing neural architectures by searching over a predefined space of possible networks. Early NAS methods employed reinforcement learning or evolutionary algorithms, but their prohibitive computational cost limited their practicality. Recent advances, such as weight-sharing and differentiable search [8], have drastically reduced search costs, making NAS applicable to large-scale transformer models.

The intersection of NAS and transformers has given rise to a vibrant research area, with dozens of methods proposed in the last two years alone.

However, the literature is scattered, and there is no unified treatment that compares these methods systematically. This paper aims to fill this gap by providing a comprehensive, up-to-date survey of NAS for transformers.

Contributions of this survey:

We present a structured overview of transformer archi-tectures, including the core self-attention mechanism, the encoder-decoder framework, and the family of BERT and Vision Transformer models. We categorise transformer NAS search spaces into attention-only, convolution-only, and hybrid spaces, and discuss the trade-offs between micro-cell and macro-layer designs. We review and compare the three main search algorithms reinforcement learning, evolutionary learning, and one-shot differentiable methods – with a focus on their applicability to transformers. We provide an in-depth analysis of hardware-aware NAS, where latency, energy, and memory constraints are incorporated into the search objective. We compile a comprehensive table of over 40 trans-former NAS methods, summarising their search space, algorithm, performance, and hardware targets. We propose a generic methodology and process flow that can serve as a blueprint for designing transformer NAS systems. We identify critical open challenges, including efficient performance estimation, joint optimisation of sparsity and quantisation, and network-accelerator co-search, and outline future research directions.

The remainder of this paper is organised as follows. Section II reviews related work on NAS, transformers, and hardware-aware design. Section III presents the foundational concepts of self-attention and transformer architectures. Section IV describes the generic NAS pipeline and the specific search spaces for transformers. Section V discusses the different search algorithms. Section VI covers hardware-aware NAS. Section VII presents a detailed results and discussion section, comparing the performance of various methods. Section VIII concludes the paper and outlines future research directions.

2. LITERATURE REVIEW

The literature relevant to transformer NAS spans three interconnected fields: neural architecture search, transformer architectures, and hardware-aware deep learning. In this sec-tion, we provide a brief overview of each area and position our work within the existing body of surveys.

Neural Architecture Search

NAS has evolved significantly since its inception. Early work by Zoph and Le [7] used reinforcement learning to generate CNN architectures, achieving competitive results on CIFAR-10 but requiring thousands of GPU days. To reduce this cost, evolutionary methods [**?**] were introduced, which maintain a population of architectures and evolve them through mutation and crossover. While these methods are more paral-lelisable, they still require training many networks.

The breakthrough came with weight-sharing approaches [**?**], where a single supernetwork encodes all candidate architec-tures, and sub-networks share weights. This idea was refined in DARTS [8], which relaxes the discrete search space and uses gradient descent to optimise architectural parameters. DARTS reduced the search cost to a few GPU days, making NAS practical. Subsequent works, such as ProxylessNAS [**?**] and OFA [**?**], further improved efficiency and added hardware awareness.

Numerous surveys have covered NAS [9], [10], but most focus on CNNs. A few surveys touch on transformer NAS, but none provide a dedicated, in-depth treatment. Our paper fills this void.

Transformer Architectures

The transformer architecture [1] has been surveyed ex-tensively [11], [12]. These surveys cover the evolution of transformer models, from the original to efficient variants like Longformer, Reformer, and Performer. Vision transformer surveys [**?**], [13] discuss the adaptation of transformers to image data, including patch embeddings, positional encodings, and hybrid models with convolutions.

However, these surveys do not focus on the architectural search problem. They describe existing designs but do not discuss how to automate the search for better designs. Our survey complements these by focusing exclusively on NAS methods that generate transformer architectures.

Hardware-Aware Deep Learning

Deploying deep neural networks on edge devices requires optimising for latency, energy, and memory. Techniques such as pruning, quantisation, and knowledge distillation have been extensively studied [**?**]. Hardware-aware NAS [**?**] extends this by searching for architectures that are inherently efficient on a given hardware platform. CNN-based HW-NAS methods like MNasNet [**?**] and FBNet [**?**] have achieved impressive results. For transformers, hardware awareness is even more critical due to their large parameter count and memory bandwidth requirements. HAT [**?**] was one of the first to address this, showing that a transformer optimised for a CPU may be suboptimal for a GPU,

and vice versa. ShiftAddNAS [**?**] fur-ther explored multiplication-free operations to reduce energy consumption.

Gap in Existing Surveys

While there are excellent surveys on NAS [9] and on transformers [11], none specifically target the intersection. The existing NAS surveys tend to mention transformer NAS only briefly, while transformer surveys do not cover NAS at all. This paper is the first to provide a comprehensive, unified review of NAS methods for transformers, including both language and vision applications, and with a strong emphasis on hardware awareness.

3. METHODOLOGY AND PROCESS FLOW

This section describes the generic pipeline for applying NAS to transformer models. The pipeline consists of five main stages: (1) problem definition, (2) search space design, (3) search strategy, (4) performance estimation, and (5) final evaluation. We also discuss how hardware constraints can be integrated into this pipeline.

Problem Formulation

Let S denote the search space, which contains all pos-sible transformer architectures built from a set of primitive operations. Let D_{train} and D_{val} be the training and validation datasets. The goal of NAS is to find an architecture $a^* \in S$ that minimises a loss function $L(a, w^*a)$, where w^*a are the optimal weights of architecture a, and possibly subject to hardware constraints $C(a) \leq T$ (e.g., latency, memory, energy). Mathe-matically, the problem can be expressed as:

$$a^* = \arg\min_{a \in S} L_{val}(a, w^*(a)) \quad \text{s.t.} \quad C(a) \leq T,$$
$$w^*(a) = \arg\min_{w} L_{train}(a, w).$$

This bilevel optimisation is the cornerstone of most NAS methods. The inner loop trains the weights of a given archi-tecture, while the outer loop searches for the best architecture.

Search Space Design

The search space defines the set of architectures that can be explored. For transformers, the search space typically includes:

Macro-structure: number of encoder/decoder layers, whether to use skip connections, and connection patterns between layers.**MHSA parameters**: number of heads, head dimension, and whether to use relative positional encoding **FFN parameters**: expansion ratio (hidden dimension / input dimension), activation function (ReLU, GELU, Swish). **Embedding parameters**: embedding dimension, patch size (for ViT), position encoding type. **Convolution parameters** (if hybrid): kernel size, stride, channel width, expansion ratio (for MBConv).Search spaces can be classified as: **Cell-based (micro)***:* A small directed acyclic graph (DAG) with a few nodes is searched; this cell is then repeated L times to build the full network. This reduces the search complexity, but the same cell is used across all layers, which may limit flexibility.**Layer-wise (macro):** Each layer is searched indepen-dently, allowing heterogeneous configurations. This is more expressive but also increases the search space size. Furthermore, spaces can be: **Attention-only***:* Contains only transformer operations (MHSA and FFN). Suitable for NLP tasks where convolutions are not needed. **Hybrid convolution-attention***:* Includes both convolutional and attention operations, allowing the model to capture both local and global features. This is particularly effective for vision tasks.

Search Strategies

Three main families of search strategies have been applied to transformers:

Reinforcement Learning (RL) Based: An RL controller (typically an RNN) generates a sequence of actions that correspond to architectural choices. The resulting architecture is trained, and its validation accuracy serves as a reward. The controller is updated via policy gradient to favour actions that yield higher rewards. While RL is flexible, it is computa-tionally expensive because each sampled architecture must be trained from scratch. Efficient variants like ENAS [**?**] share weights across architectures to reduce cost.

Evolutionary Algorithms (EA): A population of archi-tectures is initialised randomly. At each generation, the fittest individuals (based on validation accuracy) are selected, and new individuals are created through mutation (changing one or more architectural choices) and crossover (combining parts of two parents). EA is highly parallelisable and can be easily adapted to multi-objective settings (e.g., accuracy and latency). However, like RL, it may require many evaluations.

One-Shot and Differentiable Methods: To overcome the cost of training many architectures, one-shot methods train a single supernetwork that contains all possible architectures as sub-networks. Weight sharing allows the performance of a sub-network to be estimated without retraining.

Differentiable methods (e.g., DARTS [8]) take this a step further by relaxing the discrete choices into

continuous param-eters. The supernetwork is trained end-to-end using gradient descent, and after training, the discrete architecture is obtained by selecting the operation with the highest weight. This is much faster than RL or EA, but the training of the supernet-work can be unstable, especially for transformers with large hidden sizes.

Performance Estimation

Once an architecture is sampled or proposed, its perfor-mance must be estimated to guide the search. Full training on the target dataset gives the most accurate estimate but is too slow. To accelerate, several proxy strategies are used:

Low-fidelity training: Train for fewer epochs, use a smaller subset of the data, or reduce the image resolution.
Weight sharing: Use the weights from the supernetwork without retraining.
Zero-cost proxies: Estimate performance using metrics computed from the untrained network, such as gradient norm, synaptic saliency, or the degree of rank collapse [31], [35].
Predictor models: Train a separate surrogate model (e.g., a Gaussian process or neural network) to predict the accuracy of unseen architectures based on their features. The choice of performance estimation method significantly impacts the overall search cost and the quality of the final architecture.

Hardware Awareness

To make the searched models deployable on real devices, hardware metrics such as inference latency, energy consumption, and peak memory usage must be considered. Hardware-aware NAS (HW-NAS) integrates these metrics into the search objective, either as constraints or as additional objectives in a multi-objective optimisation.
Common approaches include:

- Modifying the reward (RL) or fitness (EA) to include a penalty for high latency.
- Adding a hardware cost term to the loss function (differ-entiable methods).
- Training a latency predictor that takes architectural pa-rameters as input and outputs the estimated latency on the target hardware.
- Measuring latency directly on the hardware during the search (e.g., using a profiler or compiler).

Once-for-all training [**?**] is particularly attractive for HW-NAS because a single supernetwork can be trained once, and then different sub-networks can be specialised for different hardware platforms without additional training.

Process Flow Summary

The overall process can be summarised in the following steps:

- **Define the task and hardware target**. Specify the dataset, performance metric, and any hardware con-straints.
- **Design the search space**. Choose the primitive opera-tions and the connectivity patterns. Decide whether to use a cell-based or layer-wise space, and whether to include convolutions.
- **Select the search strategy**. Choose between RL, EA, one-shot, or hybrid approaches based on available com-pute and desired flexibility.
- **Choose a performance estimation method**. Decide whether to train architectures fully, use weight sharing, or employ zero-cost proxies.
- **Integrate hardware metrics**. Add latency/energy mod-els or measurements into the objective.
- **Run the search**. Iteratively sample architectures, esti-mate their performance, and update the search policy.
- **Select the best architecture**. From the Pareto-front of accuracy vs. hardware cost, pick the architecture that best meets the requirements.
- **Retrain and evaluate**. Train the selected architecture from scratch (or fine-tune) on the full dataset and report final performance.

This generic flow applies to all transformer NAS methods, with variations in the specific choices made at each step.

4. RESULTS AND DISCUSSION

In this section, we present a comparative analysis of the transformer NAS methods reviewed in this paper. We group the methods by the target architecture (vanilla transformer, BERT, or Vision Transformer) and discuss their performance, efficiency, and hardware awareness. We also highlight key trends and trade-offs.

Performance Metrics

For language tasks, the most common metrics are BLEU score (for translation), perplexity (for language modelling), and F1 / accuracy (for classification and question answering). For vision tasks, top-1 and top-5 accuracy on ImageNet are the de facto standards. Hardware efficiency is typically measured in terms of parameter count, FLOPs, latency (ms), and sometimes energy consumption (Joules).

Vanilla Transformer NAS

The early works on transformer NAS focused on the original encoder-decoder architecture for machine

translation.

Evolved Transformer [14] used regularised evolution to search over a cell-based space that included convolutions, attention, and gated linear units. It achieved BLEU scores comparable to the original transformer with 22% fewer parameters. However, the search cost was enormous – 250,000 GPU hours – making it impractical for most researchers.

Primer [15] introduced a novel multi-depthwise-convolution attention head and squared ReLU activation. Using a similar evolutionary search but with a smaller budget, Primer matched the transformer's performance with 4.2× less compute during training, demonstrating that careful architecture modifications can yield substan-tial efficiency gains.

DARTSformer [16] tackled the memory bottleneck of applying DARTS to transformers by incorporating re-versible networks. This allowed the use of larger hidden sizes and more candidate operations. The searched model outperformed the baseline transformer on WMT'14 En-De and En-Fr, with a significantly lower search cost than Evolved Transformer.

AutoTrans [17] used ENAS to search over both the macro-structure (e.g., encoder-decoder connections) and micro-parameters (e.g., number of heads, FFN dimen-sions). It achieved better BLEU scores than a tuned transformer baseline on multiple translation tasks.

The key takeaway from these works is that evolutionary and reinforcement learning methods, while powerful, are extremely expensive. Differentiable methods like DARTSformer offer a more cost-effective alternative, but they require careful stabilisation to avoid performance collapse.

BERT NAS

BERT's large size makes it a natural target for NAS-based compression.

AdaBERT [18] used differentiable NAS with a task-specific distillation loss. By searching for a compact architecture tailored to a downstream task, it achieved 11–17× compression and 12–29× speedup over the base BERT on GLUE benchmarks.

NAS-BERT [19] took a task-agnostic approach, search-ing a chain-like space of MHSA, FFN, and separable convolution layers. Using block-wise search and perfor-mance approximation, it produced models that outper-formed AdaBERT on both GLUE and SQuAD with fewer parameters.

LightHuBERT [20] applied once-for-all search to the HuBERT speech model, enabling dynamic trade-off be-tween accuracy and model size. It reduced parameters by 29% while maintaining competitive performance on ASR tasks.

AQ-BERT [21] formulated mixed-precision quantisation as a differentiable NAS problem. It assigned different bit-widths to different layers and sub-groups, achieving better accuracy than uniform quantisation methods like Q-BERT.

BERT NAS has clearly demonstrated that automation can produce models that are not only smaller and faster, but also maintain or even improve accuracy compared to manually designed distillation or pruning techniques.

Vision Transformer NAS

The number of ViT NAS papers has exploded, with many achieving state-of-the-art results on ImageNet. Table I summarises the performance of selected methods.

Several observations emerge:

Attention-only vs. hybrid spaces: Hybrid methods (e.g., GLiT, NASViT, BurgerFormer) tend to achieve higher accuracy per FLOP than pure attention-only methods, especially for smaller models. This confirms that convo-lutions provide useful inductive biases for vision tasks.

Once-for-all training: Methods that adopt the OFA paradigm (AutoFormer, NASformer, S3-NAS) are able to support multiple subnetworks from a single trained supernetwork, enabling efficient deployment across dif-ferent resource budgets.

Zero-cost proxies: TF-TAS demonstrates that training-free proxies can drastically reduce search cost while still finding competitive architectures. This is a promising direction for democratising NAS.

Scaling laws: As-ViT [32] showed that automatically scaling up a seed architecture can achieve high accuracy without the need for expensive search over the entire space.

Hardware-Aware NAS

Incorporating hardware constraints has proven to be essen-tial for practical deployment.

HAT [**?**] demonstrated that a transformer optimised for a GPU may be significantly slower on a CPU, and vice versa. By training a single supertransformer and performing hardware-specific evolutionary search, HAT achieved up to 3× speedup on Raspberry Pi compared to the vanilla transformer.

ShiftAddNAS [**?**] replaced energy-hungry multiplica-tions with additions and bit-shifts, resulting in substantial energy savings on the Eyeriss accelerator. The search algorithm jointly optimises for accuracy and hardware efficiency.

LiteTransformerSearch [33] used a zero-cost proxy (decoder parameter count) combined with direct latency measurement on target hardware. This allowed the

search to be performed without training any network, making it extremely fast.

SuperShaper [34] focused on layer-wise shape search for BERT, showing that adapting the hidden dimensions per layer can yield better accuracy-latency trade-offs than uniform compression.

The main challenge in HW-NAS for transformers is the large memory footprint and high computational cost of the supernetwork. Techniques like reversible networks, gradient checkpointing, and progressive shrinking are being explored to mitigate this.

Discussion: Trade-offs and Trends

Across all the methods reviewed, several recurring themes and trade-offs can be identified:

- **Search cost vs. performance**: RL and evolutionary methods generally find better architectures but at a high cost. Differentiable methods are much faster but may suffer from instability and rank collapse. Zero-cost proxies offer the best speed but are less mature.
- **Generality vs. specialisation**: Task-agnostic search (e.g., NAS-BERT) produces models that can be fine-tuned to multiple tasks, while task-specific search (e.g., AdaBERT) yields more efficient models for a single application.
- **Convolutions vs. attention**: For vision, hybrid models consistently outperform pure attention models, espe-cially when the training data is limited. For language, convolutions are less common but have been used in models like ConvBERT to capture local patterns.
- **Hardware heterogeneity**: No single architecture is op-timal for all hardware platforms. This underscores the importance of HW-NAS and once-for-all approaches that can adapt to different devices without retraining.

5. CONCLUSIONS AND FUTURE SCOPE

Conclusions

This survey has provided a comprehensive overview of Neural Architecture Search methods for transformer models. We have traced the evolution from the original transformer to the latest BERT and Vision Transformer variants, and have systematically categorised the search spaces, algorithms, and hardware-awareness techniques that have been proposed.

Key findings include:

- Transformer NAS has matured rapidly, with many meth-ods now able to discover architectures that outperform hand-designed models.
- The choice of search space is as important as the search algorithm. Hybrid convolution-attention spaces are particularly effective for vision tasks.
- Differentiable one-shot methods offer the best trade-off between search cost and performance, but they require careful tuning to avoid convergence issues.
- Hardware awareness is critical for real-world deployment, and recent methods have shown that it is possible to design transformers that are both accurate and efficient on a variety of devices.

Despite the progress, many challenges remain, and the field is far from saturation.

Future Scope

Based on our analysis, we identify the following promising directions for future research:

- **Automated Search Space Generation**: Most current works manually define the search space. Automating this process, as in S3-NAS, could lead to more novel and unexpected architectures.
- **Low-Cost Performance Estimation**: Zero-cost proxies have shown great potential but are not yet reliable for large, hybrid spaces. Developing more accurate and generalisable proxies would make NAS accessible to a wider audience.
- **Integration of Sparsity and Quantisation**: Pruning and mixed-precision quantisation are usually applied after NAS. Jointly searching for the architecture, sparsity pattern, and bit-width could yield even more efficient models.
- **Network-Accelerator Co-Design**: Searching for the architecture and the hardware accelerator simultaneously (e.g., via reinforcement learning or Bayesian optimisa-tion) could unlock better accuracy-efficiency trade-offs than optimising them separately.
- **NAS Benchmarks for Transformers**: The lack of transformer-specific benchmarks makes it difficult to compare methods fairly. Building a comprehensive benchmark with latency measurements on multiple de-vices would accelerate progress.
- **Applications Beyond Classification and Translation**: Transformer NAS has been applied mainly to classifi-cation, translation, and ASR. Expanding to tasks like object detection,

segmentation, video understanding, and multi-modal learning is a natural next step.

- **Theoretical Understanding**: Why do certain architec-tures perform better? A deeper theoretical understanding of the impact of attention heads, depth, and width would help guide the design of both search spaces and algorithms.
- **Federated and Privacy-Preserving NAS**: Searching for transformer architectures on decentralised data without sharing sensitive information is an emerging challenge, especially in healthcare and finance.

We believe that transformer NAS will continue to be a vibrant and impactful area of research, and we hope this survey provides a useful foundation for researchers and practitioners alike.

TABLE 1
PERFORMANCE COMPARISON OF VISION TRANSFORMER NAS METHODS ON IMAGENET.

Method	Top-1 (%)	Params (M)	FLOPs (G)	Search Cost
AutoFormer (Base) [22]	82.4	54	11	1.5d (8×V100)
ViTAS (Twins-Large) [23]	84.0	124.8	16.1	–
S3-NAS (Base) [24]	84.0	96.6	13.6	–
NASformer (Base) [25]	83.9	96.6	13.6	–
GLiT (Base) [26]	82.3	96.1	17	–
BossNAS (T1) [27]	82.2	–	–	–
NASViT (6) [28]	82.9	–	1.88	–
BurgerFormer (Base) [29]	82.7	26	3.9	–
UniNet (3) [30]	83.5	24	4.3	–
TF-TAS (Base) [31]	82.2	54	12.0	0.5 GPU hours

REFERENCES

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszko-reit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV*, 2021, pp. 10012–10022.

H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Je´gou, "Training data-efficient image transformers & distillation through attention," in *ICML*, 2021, pp. 10347–10357.

B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv:1611.01578*, 2016.

H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *ICLR*, 2019.

T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–34, 2021.

T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *arXiv:2106.04554*, 2021.

S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Computing Surveys*, vol. 54, no. 10s, pp. 1–41, 2022.

K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE TPAMI*, vol. 45, no. 1, pp. 87–110, 2023.

D. So, Q. Le, and C. Liang, "The evolved transformer," in *ICML*, 2019, pp. 5877–5886.

D. So, W. Man˜ke, H. Liu, Z. Dai, N. Shazeer, and Q. V. Le, "Searching for efficient transformers for language modeling," in *NeurIPS*, 2021, pp. 6010–6022.

Y. Zhao, L. Dong, Y. Shen, Z. Zhang, F. Wei, and W. Chen, "Memory-efficient differentiable transformer architecture search," *arXiv:2105.14669*, 2021.

W. Zhu, X. Wang, Y. Ni, and G. Xie, "AutoTrans: Automating trans-former design via reinforced architecture search," in *NLPCC*, 2021, pp. 169–182.

D. Chen, Y. Li, M. Qiu, Z. Wang, B. Li, B. Ding, H. Deng, J. Huang, W. Lin, and J. Zhou, "AdaBERT: Task-adaptive BERT compression with differentiable neural architecture search," *arXiv:2001.04246*, 2020.

J. Xu, X. Tan, R. Luo, K. Song, J. Li, T. Qin, and T.-Y. Liu, "NAS- BERT: Task-agnostic and adaptive-size BERT compression with neural architecture search," in *KDD*, 2021, pp. 1933–1943.

R. Wang, Q. Bai, J. Ao, L. Zhou, Z. Xiong, Z. Wei, Y. Zhang, T. Ko, and H. Li, "LightHuBERT: Lightweight and configurable speech represen-tation learning with once-for-all hidden-unit BERT," *arXiv:2203.15610*, 2022.

C. Zhao, T. Hua, Y. Shen, Q. Lou, and H. Jin, "Automatic mixed-precision quantization search of BERT," *arXiv:2112.14938*, 2021.

M. Chen, H. Peng, J. Fu, and H. Ling, "AutoFormer: Searching trans-formers for visual recognition," in *ICCV*, 2021, pp. 12270–12280.

X. Su, S. You, J. Xie, M. Zheng, F. Wang, C. Qian, C. Zhang, X. Wang, and C. Xu, "ViTAS: Vision transformer architecture search," *arXiv:2106.13700*, 2021.

M. Chen, K. Wu, B. Ni, H. Peng, B. Liu, J. Fu, H. Chao, and H. Ling, "Searching the search space of vision transformer," in *NeurIPS*, 2021.

B. Ni, G. Meng, S. Xiang, and C. Pan, "NASformer: Neural architecture search for vision transformer," in *ACPR*, 2021.

B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, and W. Ouyang, "GLiT: Neural architecture search for global and local image transformer," in *ICCV*, 2021, pp. 12–21.

C. Li, T. Tang, G. Wang, J. Peng, B. Wang, X. Liang, and X. Chang, "BossNAS: Exploring hybrid CNN-transformers with block-wisely self-supervised neural architecture search," in *ICCV*, 2021, pp. 12281–12291.

C. Gong, D. Wang, M. Li, X. Chen, Z. Yan, Y. Tian, and V. Chandra, "NASViT: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training," in *ICLR*, 2022.

L. Yang, Y. Hu, S. Lu, Z. Sun, J. Mei, Y. Han, and X. Li, "Searching for BurgerFormer with micro-meso-macro space design," in *ICML*, 2022, pp. 25055–25069.

J. Liu, H. Li, G. Song, X. Huang, and Y. Liu, "UniNet: Unified architec-ture search with convolution, transformer, and MLP," *arXiv:2110.04035*, 2021.

Q. Zhou, K. Sheng, X. Zheng, K. Li, X. Sun, Y. Tian, J. Chen, and R. Ji, "Training-free transformer architecture search," *arXiv:2203.12217*, 2022.

W. Chen, W. Huang, X. Du, X. Song, Z. Wang, and D. Zhou, "Auto-scaling vision transformers without training," *arXiv:2202.11921*, 2022.
Y. Bondarenko, M. Nagel, and T. Blankevoort, "LiteTransformerSearch: Training-free neural architecture search for efficient language models," in *NeurIPS*, 2021.

C. Li, C. Xu, and Z. Zhang, "SuperShaper: Task-agnostic BERT com-pression via layer-wise shape search," *arXiv:2102.04043*, 2021.

M. S. Abdelfattah, A. Mehrotra, L. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight NAS," in *ICLR*, 2021.

M. Ding, X. Lian, L. Yang, P. Wang, X. Jin, Z. Lu, and P. Luo, "HR-NAS: Searching efficient high-resolution neural architectures with lightweight transformers," in *CVPR*, 2021, pp. 2982–2992.

H. He, J. Liu, Z. Pan, J. Cai, J. Zhang, D. Tao, and B. Zhuang, "Pruning self-attentions into convolutional layers in single path," *arXiv:2111.11802*, 2021.

H. Zhang, K. Hao, W. Pedrycz, L. Gao, X. Tang, and B. Wei, "Vision transformer with convolutions architecture search," *arXiv:2203.10435*, 2022.

X. Shi, P. Zhou, W. Chen, and L. Xie, "Efficient gradient-based neural architecture search for end-to-end ASR," in *ICMI Companion*, 2021.

Y. Liu, T. Li, P. Zhang, and Y. Yan, "Improved conformer-based end-to-end speech recognition using neural architecture search," *arXiv:2104.05390*, 2021.

Y. Yin, S. Huang, and X. Zhang, "BM-NAS: Bilevel multimodal neural architecture search," in *AAAI*, 2022.

J. Xu, X. Tan, K. Song, R. Luo, Y. Leng, T. Qin, T.-Y. Liu, and J. Li, "Analyzing and mitigating interference in neural architecture search," *arXiv:2012.10550*, 2020.

J. Gao, H. Xu, H. Shi, X. Ren, L. Philip, X. Liang, X. Jiang, and Z. Li, "AutoBERT-Zero: Evolving BERT backbone from scratch," in *AAAI*, 2022.

Y. Wang, Y. Yang, Y. Chen, J. Bai, C. Zhang, G. Su, X. Kou, Y. Tong, M. Yang, and L. Zhou, "TextNAS: A neural architecture search space tailored for text representation," in *AAAI*, 2020.

J. Xu, L. Zhao, J. Lin, R. Gao, X. Sun, and H. Yang, "KNAS: Green neural architecture search," in *ICML*, 2021.

R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, "Semi-supervised neural architecture search," in *NeurIPS*, 2020.

Y. Ren, L. Li, X. Yang, and J. Zhou, "AutoTransformer: Automatic transformer architecture design for time series classification," in *PAKDD*, 2022.

J. Kim, J. Wang, S. Kim, and Y. Lee, "Evolved speech-transformer: Applying neural architecture search to end-to-end automatic speech recognition," in *Interspeech*, 2020.

W. Zhu, X. Qiu, Y. Ni, and G. Xie, "AutoRC: Improving BERT based re-lation classification models via architecture search," *arXiv:2009.10680*, 2020.