BAYESIAN OPTIMIZATION IN SEQUENCE-TO-ARCHITECTURE LATENT SPACE FOR ZERO-SHOT NAS

Anonymous authors
Paper under double-blind review

ABSTRACT

Zero-shot Neural Architecture Search removes the prohibitive cost of traditional NAS, but its search process is typically based on the evolutionary algorithm (EA); lacking an explicit model of the objective, it often resorts to a near-random search through mutation. Bayesian Optimization offers a principled alternative by modeling the objective and aggregating information across iterations, but scales poorly to the high-dimensional, discrete, graph-structured spaces of modern NAS, restricting its use to only small networks. In this paper, we bring Bayesian Optimization to zero-shot NAS for large-scale architectures by learning a latent space via a Variational Autoencoder trained to reconstruct a novel prefix encoding of architectures and propose a proxy scalarization that combines several zero-shot proxies into a single Bayesian Optimization objective. After only 10,000 iterations of the proposed search algorithm (8 hours on a single GPU), our method found a network architecture which under the given model parameter count constraints achieves state-of-the-art results on three separate tasks – image classification, object detection and semantic segmentation. The search algorithm code, the latent space model as well as the discovered architecture is available at https://github.com/HiddenForReview.

1 INTRODUCTION

Zero-shot (training-free) Neural Architecture Search (NAS) methods have been recently gaining popularity, because they democratize NAS by removing the tremendous computational requirements associated with traditional NAS, and as such they allow to search for novel, more efficient and/or more accurate deep network architectures at a fraction of a cost and energy. The key component of every zero-shot NAS method is the search algorithm that determines which new candidate architectures should be evaluated, based on estimated performance of network architectures evaluated in previous steps, ideally steering the search towards more promising candidates as the algorithm progresses. The majority of zero-shot methods (Lee & Ham, 2024; Tybl & Neumann, 2025; Li et al., 2023) rely on the Evolutionary Algorithm (EA), because it can handle the large search space of modern deep network architectures. But EA has no explicit model of the objective and as such the results are often very close to a random search; often the only operation used to traverse the search space is the mutation operation, which essentially is just a random perturbation of the current state.

Bayesian Optimization (BO), on the other hand is a formal framework that explicitly builds a probabilistic model of the objective function and is therefore able to aggregate information obtained in previous steps to specifically guide the search towards more promising regions of the search space. Bayesian Optimization (BO) however **scales poorly to high dimensions and discrete, structured spaces**, which unfortunately is exactly the type of search space encountered in NAS methods, where the architectures are discrete graphs consisting of nodes, each of which has several hyper-parameters. This is why the application of Bayesian Optimization in zero-shot NAS Kandasamy et al. (2018); Ru et al. (2021); Shen et al. (2023); Camero et al. (2021) has so far been limited to only toy problems or small networks, simply because direct application to modern large-scale architectures with hundreds of nodes is computationally prohibitive.

In this paper, we present a novel method that allows to apply Bayesian Optimization (BO) to searching large-scale network architectures in zero-shot NAS, thus eliminating the computational burden of traditional NAS, as well as removing complexity constraints of naively applying BO to the high-dimensional space of large network architectures. We achieve this by introducing a novel latent

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

space that can represent modern large-scale architectures including EfficientNet Tan & Le (2019) and Vision Transformer Dosovitskiy et al. (2020), by training a Variational Autoencoder (VAE) that learns to reconstruct network architectures expressed by prefix encoding that serializes each architecture block into a compact, bijective string over a vocabulary of primitive and compound operations. The resulting latent space provides a smooth, low-dimensional representation of large-scale heterogeneous architectures suitable as the search domain for Bayesian Optimization.

Additionally, we propose a new proxy scalarization method that allows us to combine several state-of-the-art zero-shot proxies into a single objective. The scalarization whitens and decorrelates the proxy dimensions and, unlike rank-based aggregations, can be evaluated pointwise for any architecture, which is a key prerequisite for modern gradient-free optimization. As a result, our method – unlike previous methods based on evolutionary algorithms – is able to learn and exploit the properties of the NAS search space during the search process.

Finally, we use our newly introduced latent space to run Bayesian Optimization, using the scalarized proxy of classification accuracy as the objective function, to search for new architectures. The best architecture as per the proxy value (found without training any of the candidate networks) is only then trained using the standard UniNAS training recipe Tybl & Neumann (2026) to obtain the actual accuracy in three different vision tasks (image classification, object detection/instance segmentation, semantic segmentation).

In summary, we make the following contributions:

1. We introduce a novel latent space that can represent modern large-scale architectures including EfficientNet Tan & Le (2019) or Vision Transformer Dosovitskiy et al. (2020), but which at the same time can be used in modern optimization frameworks such as Bayesian Optimization (BO),
2. we propose using prefix encoding as the sequential (string-like) representation of network architectures, which yields compact and a bijective mapping to the standard DAG representation,
3. we propose proxy scalarization method that allows to combine several zero-shot proxies into a single continuous scalar score, so that modern zero-shot approaches can be incorporated into a standard Bayesian Optimization framework.

2 Related Work

Neural Architecture Search is characterized by two core components: a *performance signal* – ranging from zero-shot proxies to full training accuracy – and a *search strategy* that governs how the architecture space is navigated given the signal. We organize prior work along the search-strategy axis, showing how each strategy interacts and scales with the underlying architecture search space.

Evolutionary algorithms (EAs). EA leverages population-based optimization to explore architecture spaces without requiring differentiable relaxations. In the classical setting, new candidates are generated by applying mutation operators to the current population. When combined with zero-shot performance proxies – which bypass full training of candidate architectures – this paradigm has achieved competitive results (Lee & Ham, 2024; Tybl & Neumann, 2025; Li et al., 2023) on ImageNet-1K (Deng et al., 2009). A more recent line of work replaces mutation with learned diffusion models as the generative mechanism (An et al., 2023; Zhou et al., 2025). However, these diffusion-based approaches require training of a large number of networks to fit the generative model, incurring substantial computational cost; put simply, they are not zero-shot. More broadly, evolutionary methods – whether mutation- or diffusion-based – require many architecture evaluations to converge, as they rely on stochastic variation and selection rather than model-guided search. Moreover, because each new candidate is derived from local perturbations of existing ones, the search tends to remain in the neighborhood of the initial population, limiting its ability to discover novel architectures.

Bayesian optimization with hand-crafted encodings. Bayesian optimization (BO) offers a principled, sample-efficient alternative to evolutionary search by constructing a surrogate model of the objective function. A central design choice when applying BO to NAS is how architectures are represented. Several works define hand-crafted encodings paired with tailored kernel functions: Kandasamy et al. (2018) proposed a kernel based on optimal transport over graph representations, evaluated on a space of simple ConvNets; Ru et al. (2021) introduced an interpretable kernel on

similar cell-based spaces as a refinement of Kandasamy et al. (2018); White et al. (2021) employed a path-based encoding with an ensemble of neural predictors; and Jin et al. (2019) used an edit-distance kernel over architectures. Zero-shot variants have also been explored, with Shen et al. (2023) applying proxy-based BO on NASBench-201 and Camero et al. (2021) combining BO with a hyperparameter encoding kernel for recurrent architectures. *All of these methods, however, rely on the encoding remaining low-dimensional, which fundamentally limits their scalability* – none have been demonstrated on large scale datasets such as ImageNet-1K (Deng et al., 2009). The notable exception is Ouertatani (2024), who operates at ImageNet scale but restricts the search to a narrow set of configuration choices, sidestepping the encoding scalability problem rather than solving it.

Nearly all of these methods have only been validated on small, cell-based search spaces such as NASBench-201; the few ImageNet-1K results rely on transfer from a proxy task and yield modest accuracy (Li et al., 2020; Panda et al., 2021). More fundamentally, each encoding is tied to a narrow operation set and breaks down when the space grows to 20 or more node types – as required to jointly cover convolutional and transformer operations. Edit-distance kernels (Jin et al., 2019) face a combinatorial explosion over operation types; path-based encodings (White et al., 2021) assume a small, fixed operation set; vector encodings (Li et al., 2020; Ouertatani, 2024) demand manual redesign for every new vocabulary; and optimal-transport kernels (Kandasamy et al., 2018) incur costs that grow with both graph size and operation diversity. No existing hand-crafted encoding offers a general, scalable representation for the mixed search spaces that modern architecture design demands.

Bayesian optimization with learned encodings. An alternative to hand-crafting architecture representations is to learn them from data, typically via graph neural networks. Sun & Shen (2025) and Hemmi et al. (2024) paired BO with variational graph autoencoders (VGAEs), reporting results on CIFAR-100 and CIFAR-10 (Krizhevsky et al., 2009), respectively, while Hemmi et al. (2025) explored random perturbations in a VGAE-learned latent space on CIFAR-10 (transferred to ImageNet-1K with accuracy well below 80%). Graph convolutional networks (GCNs) have also served as encoders: Shi et al. (2020) combined BO with a GCN-based predictor, achieving 74.6% on ImageNet-1K via transfer from NASBench-201, and Ma et al. (2019) in the context of small images (like CIFAR, MNIST). Existing results thus remain confined to small benchmarks or narrow search spaces, and it is unclear whether learned representations transfer to the large, heterogeneous spaces we target.

Differentiable architecture search. DARTS-based methods (Ye et al., 2022) relax the discrete search space into a continuous one, enabling gradient-based optimization. Ye et al. (2022) achieved 76.2% on ImageNet-1K. Multi-objective extensions such as NSGANetV2 (Lu et al., 2020) reached 80.4% on ImageNet-1K but required training approximately 2 000 networks during search. Transfer-based approaches have also been explored: Panda et al. (2021) applied BO on CIFAR-100 and transferred the discovered architecture to ImageNet, reaching roughly 71% top-1 accuracy – underscoring the difficulty of cross-dataset transfer in NAS.

3 Sequence-to-architecture Latent Space

In this section, we begin by describing the architecture search space we adapted, followed by our method of encoding network architectures into a unified latent space (Figure 1 – left). The latent space is then used in Bayesian optimization framework to find network architectures with best accuracy (Figure 1 – right).

3.1 Architecture space

We adopt UniNAS (Tybl & Neumann, 2026) – a unified generic search space of network architectures – which contains many well-established hand-crafted topologies including ResNet He et al. (2016), EfficientNet Tan & Le (2019), ViT Dosovitskiy et al. (2020), and CoAtNet Dai et al. (2021). At the same time, the search space is also a superset of popular but small-scale NAS benchmarks such as NAS-Bench Dong & Yang (2020); Ying et al. (2019), Zen-NAS Lin et al. (2021) and AutoFormer Chen et al. (2021a;b), and therefore results achieved in the UniNAS can be viewed as a generalization of the smaller benchmarks.

In UniNAS, each network is represented as a sequence of stages, each containing multiple blocks. Each block is a directed acyclic graph (DAG), which consists of 27 node types that represent elementary operations covering convolution types, matrix multiplications, normalizations, nonlinearities, additions, chunking, and other general tensor operations with varying input/output shapes. The

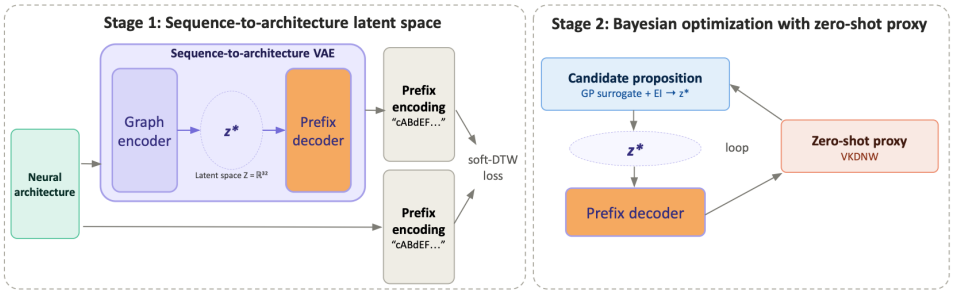

Figure 1: We first train a variational autoencoder (VAE) to create a smooth, low-dimensional representation of large-scale heterogeneous architectures (left). Then, the latent space is used for Bayesian Optimization with zero-shot accuracy proxies aggregate as the objective function (right).

only structural constraint is that the block's input and output must have identical dimensions and that adjacent nodes in the DAG have matching tensor shapes.

3.2 Prefix Encoding

The DAG representation of UniNAS nodes is flexible but difficult to encode compactly, and it does not surface the structured patterns – such as inverted bottleneck, residual skip connection, multi-head attention – that characterize modern architectures. We therefore introduce *compound nodes*, which wrap one or more inner sequential operations into a single node (represented as `Sequential` operation in PyTorch). The child nodes of *compound nodes* are again drawn from the same vocabulary, yielding a recursive grammar. We refer to the number of independent sequential operations of a *compound node* as its *arity*.

For example, a **global average pooling node** constitutes a compound node with *arity* of one, because it only wraps a single sequence of operations; a **branching node** that distributes its input into two branches and then adds the results of these branches is a compound node with *arity* of two, because it has two independent sequential operations. Each compound node type also uniquely determines the spatial dimensionality of children node inputs as well as of the compound node output. Using the same example as above, the children nodes of a **global average pooling node** always have the input dimension of $(C, 1, 1)$ due to the spatial dimensionality reduction coming from global average; on the other hand, the children nodes of a **branch node** have the same input dimension as the parent compound node. For full description of all compound nodes, see Table 5 in Appendix.

With the introduction of compound nodes, we obtain a more compact version of the UniNAS space with 20 different node types (13 *simple nodes* and 7 *compound nodes*), but at the same time we do not lose the expressive power to encode modern convolution and transformer architectures and their hybrids. In our method, we therefore represent each block as a pair (prefix, shape), where prefix is a string encoding of the block's parse-graph topology described below and shape denotes the block's input (and hence output) dimensions (C, H, W)[1].

The encoding exploits the fact that every compound node has a fixed arity, i.e. a predetermined number of inner compound nodes, which makes the serialization self-delimiting without brackets or separators. Specifically, the encoding is defined inductively:

1. **Leaf nodes.** If a DAG consists of a single primitive operation from Table 5, it is assigned a unique single character.
2. **Sequential composition.** If a DAG is composed of two sub-DAGs in sequence, each having prefix encoding s_1 and s_2, it is assigned a prefix encoding $s_1 s_2$.
3. **Compound nodes.** If a DAG is composed of a compound node having the single-character prefix encoding c with sub-modules with prefix encodings $s_1, \ldots, s_n$, then it is assigned a prefix encoding $cs_1 \cdot \cdots s_n$.

As a result, every UniNAS block is associated with a unique encoding (prefix, shape). De-serializing the encoding into an actual computation graph that can be executed on a GPU requires one additional check: that the declared input/output shape can be consistently propagated to all internal nodes. For

[1]Defining only the input shape suffices, since internal node shapes are uniquely determined by propagating through the graph.

instance, a block of shape $(4, 16, 16)$ cannot comprise two nested convolutions each reducing the channel count by a factor of 4, since the intermediate dimensions would become infeasible. This shape-consistency condition is the *only* constraint beyond syntactic well-formedness that must be verified for a prefix string to specify a valid block. This yields a compact, bijective serialization of the full computational DAG topology (see Appendix for worked example).

3.3 Latent Space Encoding

To allow continuous optimization over the space of UniNAS blocks, we train a Variational Autoencoder (VAE) that defines a mapping from latent vectors to block prefix strings. The encoder maps a computational graph G to a diagonal-Gaussian posterior $q_\phi(\mathbf{z} \mid G) = \mathcal{N}\big(\boldsymbol{\mu}_\phi(G),\, \operatorname{diag} \boldsymbol{\sigma}^2_\phi(G)\big)$ in $\mathbb{R}^{d_z}$, where $d_z = 32$; the decoder autoregressively reconstructs the block's prefix token sequence $\mathbf{x} = (x_1, \ldots, x_T)$ from a sampled latent vector. Both modules are trained end-to-end by maximizing the evidence lower bound

$$\mathcal{L}(\phi, \theta;\, G) \;=\; \mathbb{E}_{q_\phi(\mathbf{z}|G)}\big[\log p_\theta(\mathbf{x} \mid \mathbf{z})\big] \;-\; \beta\, D_{\mathrm{KL}}\big(q_\phi(\mathbf{z} \mid G) \,\|\, \mathcal{N}(\mathbf{0}, \mathbf{I})\big), \tag{1}$$

where β controls the rate–distortion trade-off Higgins et al. (2017).

Alignment-aware reconstruction loss. Rather than standard position-wise cross-entropy, we use *soft-DTW* loss Cuturi & Blondel (2017) as the reconstruction term $\log p_\theta(\mathbf{x} \mid \mathbf{z})$. Let $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_{T'})$ denote the decoded token sequence and let $e\colon \mathcal{V} \to \mathbb{R}^d$ be the shared token embedding. Soft-DTW first constructs a pairwise cost matrix $\mathbf{C} \in \mathbb{R}^{T' \times T}$ with entries

$$C_{ij} \;=\; \big\| f_\theta(\hat{x}_i,\, \mathbf{z}) - e(x_j) \big\|^2, \tag{2}$$

where f_θ maps each decoded position to the same embedding space. The loss then computes the minimum-cost monotone alignment between $\hat{\mathbf{x}}$ and $\mathbf{x}$ via the γ-smoothed soft-minimum recursion (a log-sum-exp relaxation of the standard DTW dynamic programme) Cuturi & Blondel (2017).

This alignment-aware formulation confers a concrete advantage over strict position-wise cross-entropy as it is *shift-tolerant*: Suppose that $\hat{\mathbf{x}}$ and $\mathbf{x}$ differ only by a single prefix token inserted at the beginning, that is $\hat{\mathbf{x}} = (\hat{x}_1, x_1, \ldots, x_{T-1})$. Then, the cross-entropy becomes large as it detects misalignment at each position of the sequence. However, soft-DTW remains small, as this alignment cost is controlled by the edit distance (up to the smoothing parameter γ) and architecturally minor perturbations remain close in the reconstruction loss Cuturi & Blondel (2017); Le Guen & Thome (2019). This is specifically useful in our situation as if we insert a single node into a DAG, its prefix encoding shifts entirely while the corresponding network is actually very similar (one operation more does not usually completely change the architecture).

Encoder. The encoder treats the block's DAG as an undirected graph – one node per operation, with bidirectional parent-child edges – and processes it with $L_{\mathrm{enc}} = 3$ graph-convolutional layers of width $d = 256$, each followed by a residual connection, GELU activation, and layer normalization. Global mean-pooling over all node representations yields a single d-dimensional vector, which a linear head projects to $\boldsymbol{\mu} \in \mathbb{R}^{d_z}$ and $\log \boldsymbol{\sigma}^2 \in \mathbb{R}^{d_z}$ for reparameterization sampling.

Decoder. The latent vector $\mathbf{z} \in \mathbb{R}^{d_z}$ is expanded into $M = 8$ *memory tokens* of dimension $d = 256$ by a learned linear projection. An autoregressive Transformer decoder ($L_{\mathrm{dec}} = 4$ layers, $d = 256$, $H = 4$ heads, FFN width 512) then generates the prefix token sequence one token at a time, attending to the memory tokens and all previously generated tokens via causal masking. At inference time, decoding is constrained by a finite-state machine (FSM) over the prefix grammar, which guarantees that every emitted string is a syntactically valid prefix without post-hoc repair. We note, however, that the FSM enforces only syntactic well-formedness of the prefix (i.e. conformance to its inductive definition), not semantic feasibility of the resulting DAG: the decoder may still produce prefixes whose corresponding graphs violate shape-matching constraints. Such infeasible samples are discarded during search; in practice, fewer than 5% of decoded prefixes are rejected.

Relation to graph variational autoencoders. Because our VAE encodes and decodes a *sequential* prefix string rather than an adjacency matrix, it differs from standard Graph VAEs Kipf & Welling (2016); Simonovsky & Komodakis (2018) in the decoder. This design choice is deliberate: the sequential representation admits a natural grammar constraint (the FSM above) that is difficult to impose on adjacency-matrix decoders, and it avoids the quadratic scaling of adjacency-based reconstruction with respect to the number of nodes. The encoder, by contrast, aligns with prior graph-VAE encoders.

4 SEARCH

4.1 ZERO-SHOT PROXY

The principal bottleneck in neural architecture search is the cost of evaluating candidate architectures: naïvely, each candidate must be trained to convergence before its quality can be assessed. Zero-shot (aka training-free) NAS methods sidestep this cost by replacing the full training-and-evaluation loop with a lightweight *zero-cost proxy* – a scalar score computed without training the network, yet demonstrably correlated with the final trained accuracy Abdelfattah et al. (2021); Lin et al. (2021); Lee & Ham (2024); Tybl & Neumann (2025).

Our method fully embraces this paradigm: the entire search phase optimizes only a zero-cost proxy, with no gradient-based training whatsoever. Only after the search concludes we train a single architecture – the top-scoring candidate according to the proxy – to obtain the final model. This design makes the search itself orders of magnitude cheaper than approaches that require training each candidate, and concentrates the computational budget on a single, well-chosen training run.

Training-free proxy. We adopt VKDNW Tybl & Neumann (2025) as the underlying zero-cost proxy. VKDNW achieves state-of-the-art rank correlation classification accuracy on multiple benchmarks and search spaces while requiring only a single forward pass through the network on random input data.

VKDNW, like several other modern training-free proxies, is inherently *multi-dimensional*: it produces a vector of scores characterizing different aspects of an architecture (e.g., per-layer or per-block statistics) rather than a single scalar. To serve as the objective function of a search algorithm, these dimensions must be aggregated into a single scalar that can rank and compare architectures.

A natural baseline is the nonlinear ranking scheme of AZ-NAS Lee & Ham (2024), which converts each proxy dimension into a rank and multiplicatively penalizes architectures that are not consistently well-ranked across all dimensions. However, this ranking-based aggregation is designed for search strategies that only require a relative ordering over a fixed, finite population of candidates (e.g., Evolutionary Algorithms operating on the current generation). Our setting demands a *continuous scalar score* that can be evaluated for any architecture in the search space – including previously unseen ones – because our optimization strategy must assess the absolute quality of arbitrary candidates rather than merely rank a pre-enumerated set. Ranking-based aggregation cannot satisfy this requirement: the rank of a new architecture is undefined without re-ranking the entire population.

Proxy scalarization. We therefore propose a Mahalanobis-distance-based scalarization of the multi-dimensional proxy. Let $\mathbf{p}(\alpha) \in \mathbb{R}^k$ denote the k-dimensional vector of k individual training-free proxy values of an architecture α, and let $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ denote the empirical mean and whitening matrix of proxy vectors computed over a calibration set of randomly sampled architectures from the search space. The scalarized proxy score $s(\alpha) \in \mathbb{R}$ is defined as

$$s(\alpha) = \big(\mathbf{p}(\alpha) - \boldsymbol{\mu}\big)^\top \boldsymbol{\Sigma}^{-1} 1, \tag{3}$$

where 1 denotes a vector of all ones, i.e. it is the sum of the standardized (whitened) marginal proxies, closely related to Mahalanobis distance McLachlan (1999) of the proxy vector from the population mean.

This formulation has two desirable properties. *(i) Scale normalization and decorrelation.* Multiplication by $\boldsymbol{\Sigma}^{-1}$ simultaneously standardizes the scale of each proxy dimension—accounting for potentially disparate magnitudes and variances—and decorrelates the dimensions, preventing redundant information shared between correlated proxy components from being double-counted. Equivalently, equation 3 first applies a whitening transformation Kessy et al. (2018) to the proxy vectors and then sums entries, so that each transformed dimension contributes equally and independently to the final score. *(ii) Pointwise evaluability.* Unlike ranking-based methods, our score is a continuous function of the proxy vector and can be evaluated for any architecture without reference to a fixed population. This makes it compatible with gradient-free optimization, Bayesian optimization, or any other search strategy that evaluates candidates on the fly.

4.2 BAYESIAN OPTIMIZATION

We search for high-scoring architectures by running Bayesian optimization (BO) directly in the continuous latent space $\mathcal{Z} = \mathbb{R}^{d_z}$, which was obtained by training the VAE (see Section 3.3). A candidate architecture is produced by decoding a latent vector $z \in \mathcal{Z}$ into a prefix string that is, in turn, decoded into a UniNAS block specification. The block is then used to build a full UniNAS

model (Tybl & Neumann, 2026) following the CoAtNet-style macro-structure (Dai et al., 2021): a convolutional stem is followed by two stages of increasing channel width and decreasing spatial resolution in which the discovered block template is replicated, followed by two transformer stages. This yields a complete network architecture that is scored with the zero-cost proxies.

We adopt the standard BO framework with a Gaussian process (GP) surrogate (Rasmussen & Williams, 2006) and expected improvement (EI) (Jones et al., 1998; Mockus et al., 1978) as the acquisition function; a detailed description of the framework is provided in Section A.4 in Appendix. BO is a well-established tool for hyperparameter tuning (Snoek et al., 2012) and has more recently been applied to neural architecture search (Kandasamy et al., 2018; White et al., 2021; Ru et al., 2021), but only on small networks due to its computational complexity on larger networks in its naive formulation.

4.2.1 Design choices for latent-space architecture search

We now describe the design choices that enable standard BO to operate effectively over the architecture latent space while respecting a target parameter budget.

Model-size control. Because parameter count correlates strongly with proxy performance (Tybl & Neumann, 2025; Lee & Ham, 2024), controlling for model size is essential: without it, the search would tend to select larger networks rather than architecturally superior ones at a given scale. We enforce the parameter budget at two complementary levels.

Soft penalty. A penalty on deviation from a target parameter count $n_\star$ is incorporated directly into the objective that enters the GP training set, shaping the surrogate landscape so that it rewards architectures near the desired size. Concretely, let $s(\alpha)$ denote the scalarized proxy score from equation 3 and $n(z)$ the parameter count of the decoded architecture. The penalized objective is then

$$\tilde{s}(z) \;=\; s\big(\alpha(z)\big) \;-\; \pi\big(n(z)\big), \qquad \pi(n) = \exp\Big(\frac{|\log(n/n_\star)|}{\rho}\Big) - 1, \tag{4}$$

where $\rho > 0$ is a penalty-scale hyperparameter. The exponential form grows super-linearly with the log-ratio deviation from $n_\star$, strongly discouraging large departures while remaining nearly flat in a neighborhood around the target.

Hard budget mask. In addition, a hard mask zeros the acquisition function for any candidate whose predicted parameter count falls outside a feasible interval $[n_{\min}, n_{\max}]$, preventing the search from proposing architectures that would exceed memory or compute constraints (see Equation (12) in Appendix).

Latent decoding with retries. Each latent vector z proposed by BO is decoded via the VAE decoder described in Section 3.3. Because the decoder does not guarantee that every latent point maps to a valid architecture satisfying all shape-matching constraints, we employ a retry mechanism: if the resulting prefix cannot be instantiated as a UniNAS model, we perturb the latent vector up to $M = 100$ times with progressively increasing noise

$$z^{(t)} = z + \delta\sqrt{t}\varepsilon^{(t)}, \qquad \varepsilon^{(t)} \sim \mathcal{N}\big(0,\, I_{d_z}\big), \tag{5}$$

where $\delta > 0$ and $t \in \{1, \ldots, M\}$ is the retry index. The linearly growing variance ensures that early retries remain close to the original proposal (exploiting local smoothness of the latent space) while later retries explore more broadly if the local neighborhood contains no valid architecture.

5 Experiments

5.1 Sequence-to-Architecture Latent Space

Training VAE. We train the sequence-to-architecture VAE on ∼96K randomly generated UniNAS block prefixes, holding out 2 048 blocks for validation. Because feasible blocks are generated purely by symbolic graph enumeration, this stage requires no GPU computation. The VAE uses a maximum token length of 96. The KL weight is linearly annealed from $\beta = 0$ to $\beta = 10^{-2}$ over the first 5 epochs and held fixed thereafter; training runs for 50 epochs with batch size 128. We optimize with Adam Kingma & Ba (2014) using a learning rate of 10^{-3}, a weight decay of 10^{-4}, and a gradient clipping at 1.0. The model has 4.2M parameters and trains in approximately 12 GPU hours on a single NVidia A100.

Reconstruction error. To evaluate how faithfully the autoencoder preserves architectural structure through the encode–decode round trip, we measure reconstruction quality via the *normalised Levenshtein distance* (Levenshtein, 1966) between the decoded and ground-truth token sequences. Given

Table 1: Top-1 image classification accuracy on ImageNet-1K (Deng et al., 2009). All models have a similar number of parameters and were trained using the UniNAS training protocol.

Model	Params	FLOPs	**Top-1** (%)
ResNet He et al. (2016)	24.8M	9.0G	75.19
EfficientNet Tan & Le (2019)	24.0M	6.0G	80.52
ViT Dosovitskiy et al. (2020); Dai et al. (2021)	26.4M	18.4G	80.64
CoAtNet Dai et al. (2021)	26.9M	8.44G	80.58
ZenNAS Lin et al. (2021)	26.3M	18.6G	80.48
AZ-NAS Lee & Ham (2024)	26.7M	19.0G	80.39
ZiCo Li et al. (2023)	27.0M	9.7G	80.85
UniNAS-A Tybl & Neumann (2026)	26.8M	9.0G	81.15
ours	27.0M	9.1G	**81.51**

Table 2: Comparison of zero-shot (training-free) NAS optimization strategies by best reported top-1 accuracy on ImageNet-1K (Deng et al., 2009). † denotes methods where optimal architecture was found on a smaller dataset (CIFAR) and then upscaled before training on ImageNet-1K.

Method	**Search strategy**	**Top-1 (%)**	**GPU days**
β-DARTS† Ye et al. (2022)	Differentiable (DARTS space)	76.1	0.4
GP-NAS† Li et al. (2020)	Bayesian Opt. w/ hand-crafted encoding	73.4	0.9
BONAS† Shi et al. (2020)	Bayesian Opt. w/ learned encoding	75.4	10.0
UniNAS-A Tybl & Neumann (2026)	Evolutionary Algorithm	81.15	0.5
ours	Bayesian Opt. w/ seq-to-arch latent enc.	**81.51**	0.33

two token sequences $\mathbf{x}$ and $\hat{\mathbf{x}}$ of lengths T and T', the normalized edit distance is

$$d_{\text{lev}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{\text{ed}(\mathbf{x}, \hat{\mathbf{x}})}{\max(T, T')}, \tag{6}$$

where $\text{ed}(\cdot, \cdot)$ is the Wagner–Fischer edit distance (Wagner & Fischer, 1974), i.e. the minimum number of single-token insertions, deletions, and substitutions needed to transform one sequence into the other. Dividing by the longer sequence length yields a scale-invariant metric bounded in $[0, 1]$, where 0 indicates a perfect token-for-token match.

Our sequence-to-architecture VAE achieves substantially higher reconstruction accuracy than a standard VGAE and comparable graph-based generative baselines, confirming that the token-level formulation better captures the compositional structure of UniNAS blocks (see Appendix fig. 2). This stronger reconstruction is a prerequisite for our approach: because the search operates on prefix-conditioned latent codes, the encoder must faithfully map diverse block topologies into a smooth latent space—a requirement that graph-level embeddings do not satisfy to the same degree.

5.2 Architecture Search

The search is seeded from the union of three complementary architecture pools - randomly grown graphs, random-walk graphs, and mutation-perturbed graphs—designed to provide broad initial coverage of the design space. We sample $K_{\text{seed}} = 256$ architectures uniformly from this union and evaluate the scalarized zero-shot proxy (see Section 4.1). The resulting scores are used to fit the scalarizer statistics $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ (Eq. 3) once; these are frozen for the remainder of the search. Each seed architecture is encoded to its deterministic posterior mean $\boldsymbol{\mu}_\phi(\mathbf{s})$ and inserted into the BO observation set together with its penalized composite score. We run the process for 10 000 iterations, resulting in a total 8 GPU hours.

Table 3: Accuracy of object detection & instance segmentation on MS-COCO Lin et al. (2014) and semantic segmentation on ADE20K Zhou et al. (2017). AP^b, AP^m and mIoU denote box and mask average precision and mean intersection over union, respectively.

		Object Detection and Seg.				**Semantic Segmentation**		
Model	Params	AP^b	AP^m	FPS	FLOPs	mIoU	FPS	FLOPs
ResNet He et al. (2016)	24.8M	37.7	34.7	102.8	184G	39.3	481.2	47G
EfficientNet Tan & Le (2019)	24.0M	39.0	35.8	43.1	124G	37.0	152.6	32G
CoAtNet Dai et al. (2021)	26.9M	41.3	38.4	14.4	296G	42.4	61.7	51G
UniNAS-A Tybl & Neumann (2026)	26.8M	42.4	39.0	14.2	297G	45.6	88.2	51G
ours	27.0M	**42.9**	**39.8**	13.5	305G	**46.1**	**62.4**	51G

Table 4: **Ablation of the main components.** We remove each component and replace it with the closest alternative. †BO did not improve the proxy, and the final architecture training diverged.

Variant	Search	Latent space	Proxy	Top-1 (%)	Δ
Ours (full)	BO	prefix-based	scalarized	**81.5**	–
w/o latent space	EA	none	scalarized	80.0	−1.5
w/o proxy scalarization	BO	prefix-based	VKDNW (single)	77.2	−4.3
w/o prefix encoding	BO	graph-based (VGAE)	scalarized	diverged†	–

Image classification. After the fixed number of iterations, we collect all evaluated architectures whose parameter count falls within the target budget (centered on the $n_\star$ from equation 4, as the search concentrates its evaluations in this region). From this budget-compliant subset, we select the architecture at the 99th percentile of the penalized objective $\tilde{s}$ rather than the outright maximum, as we find that the very top of the proxy-score distribution is susceptible to noise and proxy miscalibration. The architecture was then trained from scratch on ImageNet-1K (Deng et al., 2009) using the UniNAS training protocol Tybl & Neumann (2026), which we follow without modification. The training took 32 hours on 8 A100 GPUs.

The model achieves state-of-the-art top-1 accuracy on the ImageNet validation set among models trained with the standard UniNAS protocol (Table 1). Compared to accuracy reported by other zero- and one-shot approaches on the same task (Table 2), our method also clearly discovers highest-scoring architecture for the smallest GPU cost, demonstrating that the combination of a well-structured latent space with BO-guided exploration outperforms baselines.

Downstream tasks. To assess the generality of the discovered architecture beyond image classification, we fine-tune the ImageNet-pretrained backbone on two downstream tasks – object detection and instance segmentation on MS-COCO (Lin et al., 2014) and semantic segmentation on ADE20K (Zhou et al., 2017). The training took 24 hours, respectively 15 hours on 4 A100 GPUs. All fine-tuning hyper-parameters, data augmentation, and training schedules follow the standard UniNAS protocol Tybl & Neumann (2026), so that differences in reported metrics can be attributed solely to the choice of backbone architecture. As demonstrated in Table 3, the advantage carries over to downstream tasks: our backbone exceeds competing architectures under the same budget constraints, confirming that the design found by our search generalizes beyond the classification.

Ablations In Table 4, we replace each of our three components with its closest alternative while keeping the rest of the pipeline and the training protocol fixed. Without the latent space, BO is not applicable, so we fall to evolutionary search as in Tybl & Neumann (2026), with the same search budget and our scalarized proxy; accuracy drops to 80.0% (−1.5 pp). When removing the scalarizer we cannot use a composite proxy as rank-aggregations cannot drive BO (unlike EA), whose surrogate needs an absolute score for each architecture, so we optimize the strongest single proxy, VKDNW Tybl & Neumann (2025), instead; accuracy falls to 77.2%. Replacing the sequence-to-architecture VAE with a graph-based VGAE is the most damaging change: its reconstruction error stays close to the random level (see Supplementary Material), BO does not improve the proxy at all, and the selected architecture diverges during training. All three components are therefore necessary.

6 Conclusion

We proposed a novel zero-shot NAS method which uses Bayesian Optimization in a newly proposed latent space. The method handles modern large-scale architectures and it allows for using a combination of training-free proxies, thanks to the newly proposed proxy scalarization objective.

After only 10 000 iterations of the search (8 hours on a single GPU), our method found a network architecture which under the given model parameter count constraints achieves state-of-the-art results on three separate tasks – image classification, object detection/instance segmentation and semantic segmentation; this result was achieved using the official standard UniNAS training recipe, i.e. without any additional bells and whistles or hyper-parameter tuning.

In future work, we plan to evaluate how well our trained autoencoder generalizes to different search spaces and whether the learned latent representations remain effective when paired with alternative search strategies, such as CMA-ES Hansen (2016), beyond the BO framework employed here. Second, our proxy scalarizer remains fixed throughout the search, yet the distribution of zero-shot values evolves as the search progresses. Adaptively recalibrating the weights to account for this drift could further improve the search efficiency.

AI USE STATEMENT

In this work, we used generative AI tools for text polishing, readability improvement, and spell-checking. We did not use generative AI tools for idea generation. We have reviewed all AI-assisted work and take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The search algorithm code, the latent space model as well as the discovered architecture is available at a GitHub repository which is now hidden for the double-blinded review. The link will be provided in camera-ready if accepted. Specific script including hyper-parameter choices for the training of the final model can be found in the Supplementary Material.

REFERENCES

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight NAS. *arXiv preprint arXiv:2101.08134*, 2021.

Sohyun An, Hayeon Lee, Jaehyeong Jo, Seanie Lee, and Sung Ju Hwang. Diffusionnag: Predictor-guided neural architecture generation with diffusion models. *arXiv preprint arXiv:2305.16943*, 2023.

Andrés Camero, Hao Wang, Enrique Alba, and Thomas Bäck. Bayesian neural architecture search using a training-free performance metric. *Applied Soft Computing*, 106:107356, 2021.

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12270–12280, 2021a.

Minghao Chen, Kan Wu, Bolin Ni, Houwen Peng, Bei Liu, Jianlong Fu, Hongyang Chao, and Haibin Ling. Searching the search space of vision transformer. *Advances in Neural Information Processing Systems*, 34:8714–8726, 2021b.

Marco Cuturi and Mathieu Blondel. Soft-DTW: A differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 894–903, 2017.

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in neural information processing systems*, 34:3965–3977, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kazuki Hemmi, Yuki Tanigaki, and Masaki Onishi. Navigator-d3: Neural architecture search using variational graph auto-encoder toward optimal architecture design for diverse datasets. In *International Conference on Artificial Neural Networks*, pp. 292–307. Springer, 2024.

Kazuki Hemmi, Yuki Tanigaki, Kaisei Hara, and Masaki Onishi. Graph-embedded neural architecture search: A variational approach for optimized model design. *Computers, Materials, & Continua*, 84(2):2245, 2025.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

Haifeng Jin, Qingquan Song, and Xia Hu. Auto-Keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 1946–1956, 2019.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P. Xing. Neural architecture search with Bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Vincent Le Guen and Nicolas Thome. Shape and time distortion loss for training deep time series forecasting models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Junghyup Lee and Bumsub Ham. Az-nas: Assembling zero-cost proxies for network architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5893–5903, 2024.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. ZiCo: Zero-shot NAS via inverse coefficient of variation on gradients. In *The Eleventh International Conference on Learning Representations*, 2023.

Zhihang Li, Teng Xi, Jiankang Deng, Gang Zhang, Shengzhao Wen, and Ran He. GP-NAS: Gaussian process based neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11933–11942, 2020.

Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance image recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 347–356, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European conference on computer vision*, pp. 35–51. Springer, 2020.

Lizheng Ma, Jiaxu Cui, and Bo Yang. Deep neural architecture search with deep graph Bayesian optimization. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 500–507. IEEE, 2019.

Goeffrey J McLachlan. Mahalanobis distance. *Resonance*, 4(6):20–26, 1999.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

Houssem Ouertatani. *Efficient Deep Neural Architecture Search via Bayesian Optimization: An application to Computer Vision*. PhD thesis, Université de Lille, 2024.

Rameswar Panda, Michele Merler, Mayoore S Jaiswal, Hui Wu, Kandan Ramakrishnan, Ulrich Finkler, Chun-Fu Richard Chen, Minsik Cho, Rogerio Feris, David Kung, et al. Nastransfer: Analyzing architecture transferability in large scale neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9294–9302, 2021.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=j9Rv7qdXjd.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. *Taking the Human Out of the Loop: A Review of Bayesian Optimization*, volume 104. IEEE, 2016.

Yu Shen, Yang Li, Jian Zheng, Wentao Zhang, Peng Yao, Jixiang Li, Sen Yang, Ji Liu, and Bin Cui. Proxybo: Accelerating neural architecture search via bayesian optimization with zero-cost proxies. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 9792–9801, 2023.

Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James T. Kwok, and Tong Zhang. Bridging the gap between sample-based and one-shot neural architecture search with BONAS. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pp. 412–422. Springer, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

Zijun Sun and Yanning Shen. Learn to explore: Meta nas via bayesian optimization guided graph generation. *arXiv preprint arXiv:2508.09467*, 2025.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Ondrej Tybl and Lukas Neumann. Training-free neural architecture search through variance of knowledge of deep network weights. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14881–14890, 2025.

Ondrej Tybl and Lukas Neumann. Universal neural architecture space: Covering convnets, transformers and everything in between. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3679–3688, 2026.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

Colin White, Willie Neiswanger, and Yash Savani. BANANAS: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10293–10301, 2021.

Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. b-darts: Beta-decay regularization for differentiable architecture search. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10874–10883, 2022.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International conference on machine learning*, pp. 7105–7114. PMLR, 2019.

Bingye Zhou, Caiyang Yu, and Chenwei Tang. Evolution meets diffusion: Efficient neural architecture generation. *arXiv preprint arXiv:2504.17827*, 2025.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.