DIFFUSIONNAG: PREDICTOR-GUIDED NEURAL ARCHITECTURE GENERATION WITH DIFFUSION MODELS

Sohyun An[1]*, **Hayeon Lee**[1]*, **Jaehyeong Jo**[1], **Seanie Lee**[1], **Sung Ju Hwang**[1,2]
KAIST[1], DeepAuto.ai[2], Seoul, South Korea
`{sohyunan, hayeon926, harryjo97, lsnfamily02, sjhwang82}@kaist.ac.kr`

ABSTRACT

Existing NAS methods suffer from either an excessive amount of time for repetitive sampling and training of many task-irrelevant architectures. To tackle such limitations of existing NAS methods, we propose a paradigm shift from NAS to a novel conditional Neural Architecture Generation (NAG) framework based on diffusion models, dubbed DiffusionNAG. Specifically, we consider the neural architectures as directed graphs and propose a graph diffusion model for generating them. Moreover, with the guidance of parameterized predictors, DiffusionNAG can flexibly generate task-optimal architectures with the desired properties for diverse tasks, by sampling from a region that is more likely to satisfy the properties. This conditional NAG scheme is significantly more efficient than previous NAS schemes which sample the architectures and filter them using the property predictors. We validate the effectiveness of DiffusionNAG through extensive experiments in two predictor-based NAS scenarios: Transferable NAS and Bayesian Optimization (BO)-based NAS. DiffusionNAG achieves superior performance with speedups of up to 20× when compared to the baselines on Transferable NAS benchmarks. Furthermore, when integrated into a BO-based algorithm, DiffusionNAG outperforms existing BO-based NAS approaches, particularly in the large MobileNetV3 search space on the ImageNet 1K dataset. Code is available at https://github.com/CownowAn/DiffusionNAG.

1 INTRODUCTION

While Neural Architecture Search (NAS) approaches automate neural architecture design, eliminating the need for manual design process with trial-and-error (Zoph & Le, 2017; Liu et al., 2019; Cai et al., 2019; Luo et al., 2018; Real et al., 2019; White et al., 2020), they mostly suffer from the high search cost, which often includes the full training with the searched architectures. To address this issue, many previous works have proposed to utilize parameterized property predictors (Luo et al., 2018; White et al., 2021a;b; 2023; Ning et al., 2020; 2021; Dudziak et al., 2020; Lee et al., 2021a;b) that can predict the performance of an architecture without training. However, existing NAS approaches still result in large waste of time as they need to explore an extensive search space and the property predictors mostly play a passive role such as the evaluators that rank architecture candidates provided by a search strategy to simply filter them out during the search process.

To overcome such limitations, we propose a paradigm shift from NAS (Neural Architecture Search) to a novel conditional NAG (Neural Architecture Generation) framework that enables the *generation* of desired neural architectures. Specifically, we introduce a novel predictor-guided **Diffusion**-based **N**eural **A**rchitecture **G**enerative framework called DiffusionNAG, which explicitly incorporates the predictors into generating architectures that satisfy the objectives (e.g., high accuracy or robustness against attack). To achieve this goal, we employ the diffusion generative models (Ho et al., 2020; Song et al., 2021b), which generate data by gradually injecting noise into the data and learning to reverse this process. They have demonstrated remarkable generative performance across a wide range of domains. Especially, we are inspired by their parameterized model-guidance mechanism (Sohl-Dickstein et al., 2015; Vignac et al., 2022) that allows the diffusion generative models to excel in conditional generation over diverse domains such as generating images that match specific labels (Ramesh et al., 2021) or discovering new drugs meeting particular property criteria (Lee et al., 2023).

---

*These authors contributed equally to this work.

In this framework, we begin by training the base diffusion generative model to generate architectures that follow the distribution of a search space without requiring expensive label information, e.g., accuracy. Then, to achieve our primary goal of generating architectures that meet the specified target condition, we deploy the trained diffusion model to diverse downstream tasks, while controlling the generation process with property predictors. Specifically, we leverage the gradients of parameterized predictors to guide the generative model toward the space of the architectures with desired properties. The proposed conditional NAG framework offers the key advantages compared with the conventional NAS methods as follows: Firstly, our approach facilitates efficient search by generating architectures that follow the specific distribution of interest within the search space, minimizing the time wasted exploring architectures that are less likely to have the desired properties. Secondly, DiffusionNAG, which utilizes the predictor for both NAG and evaluation purposes, shows superior performance compared to the traditional approach, where the same predictor is solely limited to the evaluation role. Lastly, DiffusionNAG is easily applicable to various types of NAS tasks (e.g., latency or robustness-constrained NAS) as we can swap out the predictors in a plug-and-play manner without retraining the base generative model, making it practical for diverse NAS scenarios.

Additionally, to ensure the generation of valid architectures, we design a novel score network for neural architectures. In previous works on NAS, neural architectures have been typically represented as directed acyclic graphs (Zhang et al., 2019) to model their computational flow where the input data sequentially passes through the multiple layers of the network to produce an output. However, existing graph diffusion models (Niu et al., 2020a; Jo et al., 2022) have primarily focused on *undirected* graphs, which represent structure information of graphs while completely ignoring the directional relationships between nodes, and thus cannot capture the computational flow in architectures. To address this issue, we introduce a score network that encodes the positional information of nodes to capture their order connected by directed edges.

We demonstrate the effectiveness of DiffusionNAG with extensive experiments under two key predictor-based NAS scenarios: 1) Transferable NAS and 2) Bayesian Optimization (BO)-based NAS. For Transferable NAS using transferable dataset-aware predictors, DiffusionNAG achieves superior or comparable performance with the speedup of up to 20× on four datasets from Transferable NAS benchmarks, including the large MobileNetV3 (MBv3) search space and NAS-Bench-201. Notably, DiffusionNAG demonstrates superior generation quality compared to MetaD2A (Lee et al., 2021a), a closely related *unconditional* generation-based method. For BO-based NAS with task-specific predictors, DiffusionNAG outperforms existing BO-based NAS approaches that rely on heuristic acquisition optimization strategies, such as random architecture sampling or architecture mutation, across four acquisition functions. This is because DiffusionNAG overcomes the limitation of existing BO-based NAS, which samples low-quality architectures during the initial phase, by sampling from the space of the architectures that satisfy the given properties. DiffusionNAG obtains especially large performance gains on the large MBv3 search space on the ImageNet 1K dataset, demonstrating its effectiveness in restricting the solution space when the search space is large. Furthermore, we verify that our score network generates 100% valid architectures by successfully capturing their computational flow, whereas the diffusion model for undirected graphs (Jo et al., 2022) almost fails.

Our contributions can be summarized as follows:

- We propose a paradigm shift from conventional NAS approaches to a novel conditional Neural Architecture Generation (NAG) scheme, by proposing a framework called DiffusionNAG. With the guidance of the property predictors, DiffusionNAG can generate task-optimal architectures for diverse tasks.
- DiffusionNAG offers several advantages compared with conventional NAS methods, including efficient and effective search, superior utilization of predictors for both NAG and evaluation purposes, and easy adaptability across diverse tasks.
- Furthermore, to ensure the generation of valid architectures by accurately capturing the computational flow, we introduce a novel score network for neural architectures that encodes positional information in directed acyclic graphs representing architectures.
- We have demonstrated the effectiveness of DiffusionNAG in Transferable NAS and BO-NAS scenarios, achieving significant acceleration and improved search performance in extensive experimental settings. DiffusionNAG significantly outperforms existing NAS methods in such experiments.

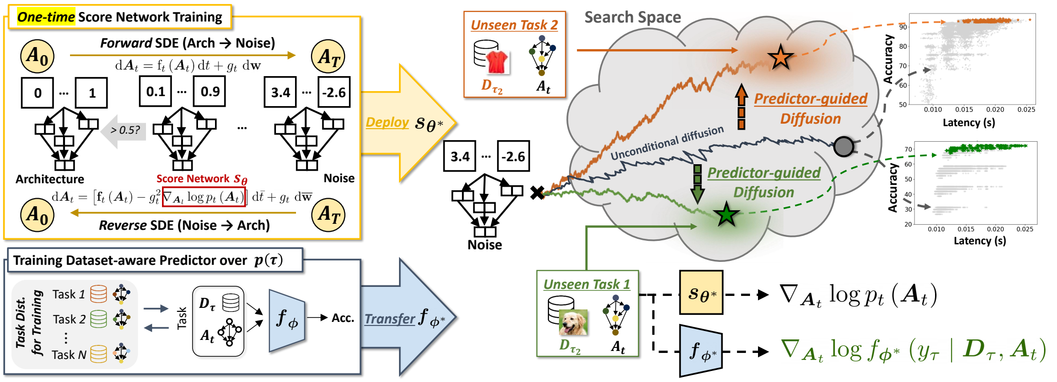

One-time Score Network Training
Forward SDE (Arch → Noise)
A_0
A_T
$\mathrm{d}\boldsymbol{A}_t = \mathrm{f}_t(\boldsymbol{A}_t)\,\mathrm{d}t + g_t\,\mathrm{d}\mathbf{w}$
0
1
0.1
0.9
3.4
-2.6
> 0.5?
Architecture
Score Network s_θ
Noise
$\mathrm{d}\boldsymbol{A}_t = [\mathrm{f}_t(\boldsymbol{A}_t) - g_t^2 \boxed{\nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t)}]\,\mathrm{d}\bar{t} + g_t\,\mathrm{d}\bar{\mathbf{w}}$
Reverse SDE (Noise → Arch)
Deploy s_{θ^*}
Training Dataset-aware Predictor over $p(\tau)$
Task Dist. for Training
Task 1
Task 2
Task N
Task
D_τ
A_t
f_ϕ
Acc.
Transfer f_{ϕ^*}
Unseen Task 2
D_{τ_2}
A_t
Search Space
Predictor-guided Diffusion
Unconditional diffusion
Predictor-guided Diffusion
3.4
-2.6
Noise
Accuracy
Latency (s)
90
80
70
60
50
0.010
0.015
0.020
0.025
Accuracy
Latency (s)
70
60
50
40
30
20
0.010
0.015
0.020
0.025
Unseen Task 1
D_{τ_2}
A_t
s_{θ^*}
f_{ϕ^*}
$\nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t)$
$\nabla_{\boldsymbol{A}_t} \log f_{\boldsymbol{\phi}^*}(y_\tau \mid \boldsymbol{D}_\tau, \boldsymbol{A}_t)$

Figure 1: **Illustration of DiffusionNAG in Transferable NAS Scenarios.** DiffusionNAG generates desired neural architectures for a given unseen task by guiding the generation process with the transferable dataset-aware performance predictor $f_{\boldsymbol{\phi}^*}(y_\tau|\boldsymbol{D}_\tau, \boldsymbol{A}_t)$.

2 METHOD

In Section 2.1, we first formulate the diffusion process for the generation of the architectures that follow the distribution of the search space. In Section 2.2, we propose a conditional diffusion framework for NAG that leverages a predictor for guiding the generation process. Finally, we extend the architecture generation framework for Transferable NAS in Section 2.3.

Representation of Neural Architectures A neural architecture $\boldsymbol{A}$ in the search space $\mathcal{A}$ is typically considered as a directed acyclic graph (DAG) (Dong & Yang, 2020b). Specifically, the architecture $\boldsymbol{A}$ with N nodes is defined by its operator type matrix $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{N\times F}$ and upper triangular adjacency matrix $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{N\times N}$, as $\boldsymbol{A} = (\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}}) \in \mathbb{R}^{N\times F} \times \mathbb{R}^{N\times N}$, where F is the number of predefined operator sets. In the MobileNetV3 search space (Cai et al., 2020), N represents the maximum possible number of layers, and the operation sets denote a set of combinations of the kernel size and width.

2.1 NEURAL ARCHITECTURE DIFFUSION PROCESS

As a first step, we formulate an unconditional neural architecture diffusion process. Following Song et al. (2021b), we define a forward diffusion process that describes the perturbation of neural architecture distribution (search space) to a known prior distribution (e.g., Gaussian normal distribution) modeled by a stochastic differential equation (SDE), and then learn to reverse the perturbation process to sample the architectures from the search space starting from noise.

Forward process We define the forward diffusion process that maps the neural architecture distribution $p(\boldsymbol{A}_0)$ to the known prior distribution $p(\boldsymbol{A}_T)$ as the following Itô SDE:

$$\mathrm{d}\boldsymbol{A}_t = \mathbf{f}_t(\boldsymbol{A}_t)\mathrm{d}t + g_t\mathrm{d}\mathbf{w}, \tag{1}$$

where t-subscript represents a function of time ($F_t(\cdot) := F(\cdot, t)$), $\mathbf{f}_t(\cdot)\colon \mathcal{A} \to \mathcal{A}$ is the linear drift coefficient, $g_t\colon \mathcal{A} \to \mathbb{R}$ is the scalar diffusion coefficient, and $\mathbf{w}$ is the standard Wiener process. Following Jo et al. (2022), we adopt a similar approach where architectures are regarded as entities embedded in a continuous space. Subsequently, during the forward diffusion process, the architecture is perturbed with Gaussian noise at each step.

Reverse process The reverse-time diffusion process corresponding to the forward process is modeled by the following SDE (Anderson, 1982; Song et al., 2021b):

$$\mathrm{d}\boldsymbol{A}_t = \left[\mathbf{f}_t(\boldsymbol{A}_t) - g_t^2 \nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t)\right]\mathrm{d}\bar{t} + g_t\mathrm{d}\bar{\mathbf{w}}, \tag{2}$$

where p_t denotes the marginal distribution under the forward diffusion process, $\mathrm{d}\bar{t}$ represents an infinitesimal negative time step and $\bar{\mathbf{w}}$ is the reverse-time standard Wiener process.

In order to use the reverse process as a generative model, the score network $\boldsymbol{s_\theta}$ is trained to approximate the score function $\nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t)$ with the following score matching (Hyvärinen, 2005; Song et al., 2021a) objective, where $\lambda(t)$ is a given positive weighting function:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{\boldsymbol{A}_0} \mathbb{E}_{\boldsymbol{A}_t|\boldsymbol{A}_0} \left\| \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{A}_t, t) - \nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t) \right\|_2^2 \right\}. \tag{3}$$

Once the score network has been trained, we can generate neural architectures that follow the original distribution $p(\boldsymbol{A}_0)$ using the reverse process of Equation (2). To be specific, we start from noise sampled from the known prior distribution and simulate the reverse process, where the score function is approximated by the score network $\boldsymbol{s}_{\boldsymbol{\theta}^*}(\boldsymbol{A}_t, t)$. Following various continuous graph diffusion models (Niu et al., 2020a; Jo et al., 2022), we discretize the entries of the architecture matrices using the operator $\mathbb{1}_{>0.5}$ to obtain discrete 0-1 matrices after generating samples by simulating the reverse diffusion process. Empirically, we observed that the entries of the generated samples after simulating the diffusion process do not significantly deviate from integer values of 0 and 1.

Score Network for Neural Architectures To generate valid neural architectures, the score network should capture 1) the dependency between nodes, reflecting the computational flow (Dong & Yang, 2020a; Zhang et al., 2019), and 2) the accurate position of each layer within the overall architecture to comply with the rules of a specific search space. Inspired by Yan et al. (2021a) on architecture encoding, we use L transformer blocks (T) with an attention mask $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ that indicates the dependency between nodes, i.e., an upper triangular matrix of DAG representation (Dong & Yang, 2020a; Zhang et al., 2019), to parameterize the score network. (See Appendix B for more detailed descriptions) Furthermore, we introduce positional embedding $\mathbf{Emb}_{pos}(\mathbf{v}_i)$ to more accurately capture the topological ordering of layers in architectures, which leads to the generation of valid architectures adhering to specific rules within the given search space as follows:

$$\mathbf{Emb}_i = \mathbf{Emb}_{ops}(\mathbf{v}_i) + \mathbf{Emb}_{pos}(\mathbf{v}_i) + \mathbf{Emb}_{time}(t), \text{ where } \mathbf{v}_i : i\text{-th row of } \boldsymbol{\mathcal{V}} \text{ for } i \in [N], \tag{4}$$

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{A}_t, t) = \text{MLP}(\boldsymbol{H}_L), \text{ where } \boldsymbol{H}_i^0 = \mathbf{Emb}_i, \boldsymbol{H}^l = \text{T}\left(\boldsymbol{H}^{l-1}, \boldsymbol{M}\right) \text{ and } \boldsymbol{H}^l = [\boldsymbol{H}_1^l \cdots \boldsymbol{H}_N^l], \tag{5}$$

where $\mathbf{Emb}_{ops}(\mathbf{v}_i)$ and $\mathbf{Emb}_{time}(t)$ are embeddings of each node $\mathbf{v}_i$ and time t, respectively.

While simulating Equation (2) backward in time can generate random architectures within the entire search space, random generation is insufficient for the main goal of DiffusionNAG. Therefore, we introduce a *conditional* NAG framework to achieve this goal in the following section.

2.2 CONDITIONAL NEURAL ARCHITECTURE GENERATION

Inspired by the parameterized model-guidance scheme (Sohl-Dickstein et al., 2015; Vignac et al., 2022; Dhariwal & Nichol, 2021), we incorporate a parameterized predictor in our framework to actively guide the generation toward architectures that satisfy specific objectives. Let y be the desired property (e.g., high accuracy or robustness against attacks) we want the neural architectures to satisfy. Then, we include the information of y into the score function. To be specific, we generate neural architectures from the conditional distribution $p_t(\boldsymbol{A}_t|y)$ by solving the following conditional reverse-time SDE (Song et al., 2021b):

$$\mathrm{d}\boldsymbol{A}_t = \left[\mathbf{f}_t(\boldsymbol{A}_t) - g_t^2 \nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t|y)\right] \mathrm{d}\bar{t} + g_t \mathrm{d}\bar{\mathbf{w}}. \tag{6}$$

Here, we can decompose the conditional score function $\nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t|y)$ in Equation (6) as the sum of two gradients that is derived from the Bayes' theorem $p(\boldsymbol{A}_t|y) \propto p(\boldsymbol{A}_t)\ p(y|\boldsymbol{A}_t)$:

$$\nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t|y) = \nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t) + \nabla_{\boldsymbol{A}_t} \log p_t(y|\boldsymbol{A}_t). \tag{7}$$

By approximating the score function $\nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{A}_t)$ with the score network $\boldsymbol{s}_{\theta^*}$, the conditional generative process of Equation (6) can be simulated if the term $\nabla_{\boldsymbol{A}_t} \log p_t(y|\boldsymbol{A}_t)$ could be estimated. Since $\log p_t(y|\boldsymbol{A}_t)$ represents the log-likelihood that the neural architecture $\boldsymbol{A}_t$ satisfies the target property y, we propose to model $\log p_t(y|\boldsymbol{A}_t)$ using a pre-trained predictor $f_{\boldsymbol{\phi}}(y|\boldsymbol{A}_t)$ parameterized by $\boldsymbol{\phi}$, which predicts the desired property y given a perturbed neural architecture $\boldsymbol{A}_t$:

$$\nabla_{\boldsymbol{A}_t} \log p_t(y|\boldsymbol{A}_t) \approx \nabla_{\boldsymbol{A}_t} \log f_{\boldsymbol{\phi}}(y|\boldsymbol{A}_t). \tag{8}$$

As a result, we construct the guidance scheme with the predictor as follows, where k_t is a constant that determines the scale of the guidance of the predictor:

$$\mathrm{d}\boldsymbol{A}_t = \left\{\mathbf{f}_t(\boldsymbol{A}_t) - g_t^2 \left[\boldsymbol{s}_{\boldsymbol{\theta}^*}(\boldsymbol{A}_t, t) + k_t \nabla_{\boldsymbol{A}_t} \log f_{\boldsymbol{\phi}}(y|\boldsymbol{A}_t)\right]\right\} \mathrm{d}\bar{t} + g_t \mathrm{d}\bar{\mathbf{w}}. \tag{9}$$

Intuitively, the predictor guides the generative process by modifying the unconditional score function which is estimated by $s_{\boldsymbol{\theta}^*}$ at each sampling step. The key advantage of this framework is that we only need to train the score network **once** and can generate architectures with various target properties by simply changing the predictor. Our approach can reduce significant computational overhead for the conditional NAG compared to the classifier-free guidance scheme (Hoogeboom et al., 2022) that requires retraining the diffusion model every time the conditioning properties change.

Table 1: **Comparison with Transferable NAS on MBv3 Search Space.** The accuracies are reported with 95% confidence intervals over 3 runs. The *p-value* represents the result of a t-test conducted on the accuracies of 30 architecture samples obtained by our method and each baseline method.

	Stats.	Transferable NAS MetaD2A (Lee et al., 2021a)	Transferable NAS TNAS (Shala et al., 2023)	DiffusionNAG
CIFAR-10	Max	97.45±0.07	97.48±0.14	**97.52±0.07**
	Mean	97.28±0.01	97.22±0.00	**97.39±0.01**
	Min	97.09±0.13	95.62±0.09	**97.23±0.06**
	p-value	0.00000191	0.0024	-
CIFAR-100	Max	86.00±0.19	85.95±0.29	**86.07±0.16**
	Mean	85.56±0.02	85.30±0.04	**85.74±0.04**
	Min	84.74±0.13	81.30±0.18	**85.42±0.08**
	p-value	0.0037	0.0037	-
Aircraft	Max	82.18±0.70	**82.31±0.31**	82.28±0.29
	Mean	81.19±0.11	80.86±0.15	**81.47±0.05**
	Min	79.71±0.54	74.99±0.65	**80.88±0.54**
	p-value	0.0169	0.0052	-
Oxford-IIIT Pets	Max	95.28±0.50	95.04±0.44	**95.34±0.29**
	Mean	94.55±0.03	94.47±0.10	**94.75±0.10**
	Min	93.68±0.16	92.39±0.04	**94.28±0.17**
	p-value	0.0025	0.0031	-

2.3 TRANSFERABLE CONDITIONAL NEURAL ARCHITECTURE GENERATION

Transferable NAS (Lee et al., 2021a; Shala et al., 2023) offers practical NAS capabilities for diverse real-world tasks, by simulating human learning. They acquire a knowledge from past NAS tasks to improve search performance on new tasks. In this section, to achieve highly efficient Transferable NAS, we extend the conditional NAG framework discussed earlier into a diffusion-based transferable NAG method by combining our framework with the transferable dataset-aware predictors from Transferable NAS methods (Lee et al., 2021a; Shala et al., 2023). A dataset-aware predictor $f_{\boldsymbol{\phi}}(\boldsymbol{D}, \boldsymbol{A}_t)$ is conditioned on a dataset $\boldsymbol{D}$. In other words, even for the same architecture, if datasets are different, the predictor can predict accuracy differently. The predictor is meta-learned with Equation (21) over the task distribution $p(\mathcal{T})$ utilizing a meta-dataset $\mathcal{S} := \{(\boldsymbol{A}^{(i)}, y_i, \boldsymbol{D}_i)\}_{i=1}^{K}$ with K tasks consisting of (dataset, architecture, accuracy) triplets for each task. We use the meta-dataset collected by Lee et al. (2021a). The key advantage is that by exploiting the knowledge learned from the task distribution, we can conduct fast and accurate predictions for unseen datasets without additional predictor training. We integrate the meta-learned dataset-aware predictor $f_{\boldsymbol{\phi}}(\boldsymbol{D}, \boldsymbol{A}_t)$ into the conditional neural architecture generative process (Equation (9)) for an unseen dataset $\tilde{\boldsymbol{D}}$ as follows:

$$\mathrm{d}\boldsymbol{A}_t = \Big\{ \mathbf{f}_t(\boldsymbol{A}_t) - g_t^2 \Big[\boldsymbol{s}_{\boldsymbol{\theta}^*}(\boldsymbol{A}_t, t) + k_t \nabla_{\boldsymbol{A}_t} \log f_{\boldsymbol{\phi}}(y|\tilde{\boldsymbol{D}}, \boldsymbol{A}_t) \Big] \Big\} \mathrm{d}\bar{t} + g_t \mathrm{d}\bar{\mathbf{w}}. \tag{10}$$

3 EXPERIMENT

We validate the effectiveness of DiffusionNAG on two predictor-based NAS scenarios: Transferable NAS (Section 3.1) and BO-based NAS (Section 3.2). In Section 3.3, we demonstrate the effectiveness of the proposed score network.

Search Space We validate our framework on two Transferable NAS benchmark search spaces (Lee et al., 2021a): MobileNetV3 (MBv3) (Cai et al., 2020) and NAS-Bench-201 (NB201) (Dong & Yang, 2020b). Especially, MBv3 is a large search space, with approximately $\mathbf{10^{19}}$ architectures. (Please see Appendix C.1 for detailed explanations.)

Training Score Network The score network is trained *only once* for all experiments conducted within each search space. Note that training the score network *only requires architectures (graph) without the need for accuracy which is expensive information*. The training process required 21.33 GPU hours (MBv3) and 3.43 GPU hours (NB201) on Tesla V100-SXM2, respectively.

3.1 COMPARISON WITH TRANSFERABLE NAS METHODS

Experimental Setup Transferable NAS methods (Shala et al., 2023; Lee et al., 2021a) are designed to leverage prior knowledge learned from previous NAS tasks, making NAS more practical on an

Table 2: **Comparison with Transferable NAS on NB201 Serach Space.** We present the accuracy achieved on four unseen datasets. Additionally, we provide the number of neural architectures (**Trained Archs**) that are actually trained to achieve accuracy. The accuracies are reported with 95% confidence intervals over 3 runs.

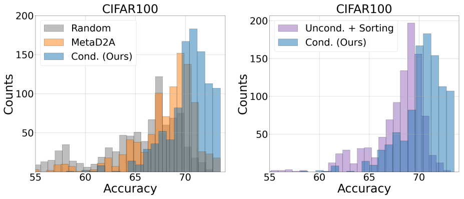
Type	Method	CIFAR-10 Accuracy (%)	CIFAR-10 Trained Archs	CIFAR-100 Accuracy (%)	CIFAR-100 Trained Archs	Aircraft Accuracy (%)	Aircraft Trained Archs	Oxford-IIIT Pets Accuracy (%)	Oxford-IIIT Pets Trained Archs
	ResNet (He et al., 2016)	93.97$_{\pm 0.00}$	N/A	70.86$_{\pm 0.00}$	N/A	47.01$_{\pm 1.16}$	N/A	25.58$_{\pm 3.43}$	N/A
	RS (Bergstra & Bengio, 2012)	93.70$_{\pm 0.36}$	> 500	71.04$_{\pm 1.07}$	> 500	-	-	-	-
	REA (Real et al., 2019)	93.92$_{\pm 0.30}$	> 500	71.84$_{\pm 0.99}$	> 500	-	-	-	-
	REINFORCE (Williams, 1992)	93.85$_{\pm 0.37}$	> 500	71.71$_{\pm 1.09}$	> 500	-	-	-	-
One-shot NAS*	RSPS (Li & Talwalkar, 2019)	84.07$_{\pm 3.61}$	N/A	52.31$_{\pm 5.77}$	N/A	42.19$_{\pm 3.88}$	N/A	22.91$_{\pm 1.65}$	N/A
	SETN (Dong & Yang, 2019a)	87.64$_{\pm 0.00}$	N/A	59.09$_{\pm 0.24}$	N/A	44.84$_{\pm 3.96}$	N/A	25.17$_{\pm 1.68}$	N/A
	GDAS (Dong & Yang, 2019b)	93.61$_{\pm 0.09}$	N/A	70.70$_{\pm 0.30}$	N/A	53.52$_{\pm 0.48}$	N/A	24.02$_{\pm 2.75}$	N/A
	PC-DARTS (Xu et al., 2020)	93.66$_{\pm 0.17}$	N/A	66.64$_{\pm 2.34}$	N/A	26.33$_{\pm 3.40}$	N/A	25.31$_{\pm 1.38}$	N/A
	DrNAS (Chen et al., 2021)	94.36$_{\pm 0.00}$	N/A	**73.51$_{\pm 0.00}$**	N/A	46.08$_{\pm 7.00}$	N/A	26.73$_{\pm 2.61}$	N/A
BO-based NAS	BOHB (Falkner et al., 2018)	93.61$_{\pm 0.52}$	> 500	70.85$_{\pm 1.28}$	> 500	-	-	-	-
	GP-UCB	**94.37$_{\pm 0.00}$**	58	73.14$_{\pm 0.00}$	100	41.72$_{\pm 0.00}$	40	40.60$_{\pm 1.10}$	11
	BANANAS (White et al., 2021a)	**94.37$_{\pm 0.00}$**	46	**73.51$_{\pm 0.00}$**	88	41.72$_{\pm 0.00}$	40	40.15$_{\pm 1.59}$	17
	NASBOWL (Ru et al., 2021)	94.34$_{\pm 0.00}$	100	**73.51$_{\pm 0.00}$**	87	53.73$_{\pm 0.83}$	40	41.29$_{\pm 1.10}$	17
	HEBO (Cowen-Rivers et al., 2022)	94.34$_{\pm 0.00}$	100	72.62$_{\pm 0.20}$	100	49.32$_{\pm 6.10}$	40	40.55$_{\pm 1.15}$	18
Transferable NAS	TNAS (Shala et al., 2023)	**94.37$_{\pm 0.00}$**	29	**73.51$_{\pm 0.00}$**	59	**59.15$_{\pm 0.58}$**	26	40.00$_{\pm 0.00}$	6
	MetaD2A (Lee et al., 2021a)	**94.37$_{\pm 0.00}$**	100	73.34$_{\pm 0.04}$	100	57.71$_{\pm 0.20}$	40	39.04$_{\pm 0.20}$	40
	DiffusionNAG (Ours)	**94.37$_{\pm 0.00}$**	**1**	**73.51$_{\pm 0.00}$**	**2**	58.83$_{\pm 3.75}$	**3**	**41.80$_{\pm 3.82}$**	**2**

* We report the search time of one-shot NAS methods in Appendix C.3.

Table 3: **Statistics of the generated architectures.** Each method generates 1,000 architectures.

Target Dataset	Stats.	Oracle Top-1,000	Random	MetaD2A	Uncond. + Sorting	Cond. (Ours)
CIFAR10	Max	94.37	**94.37**	**94.37**	**94.37**	**94.37**
	Mean	93.50	87.12	91.52	90.77	**93.13**
	Min	93.18	10.00	10.00	10.00	**86.44**
CIFAR100	Max	73.51	72.74	**73.51**	73.16	**73.51**
	Mean	70.62	61.59	67.14	66.37	**70.34**
	Min	69.91	1.00	1.00	1.00	**58.09**

Figure 2: **The distribution of generated architectures.**



unseen task. To achieve this, all Transferable NAS methods, including our DiffusionNAG, utilize a *transferable dataset-aware accuracy predictor*, as described in Section 2.3. The dataset-aware predictor is meta-trained on the meta-dataset provided by Lee et al. (2021a), which consists of 153,408/4,230 meta-training tasks for MBv3/NB201, respectively. For more details, please refer to Lee et al. (2021a). **MetaD2A** (Lee et al., 2021a), which is the most closely related to our work, includes an unconditional architecture generative model that *explicitly excludes the dataset-aware predictor* during the generation process. Instead, MetaD2A needs to search for optimal architectures across multiple tasks, train these architectures to obtain their accuracy data, and use this costly accuracy collection to train its generative model. Besides, it uses the dataset-aware predictor only during the subsequent evaluation stage to rank the generated architectures. During the test phase, it first *objective-unconditionally generates* architectures and then evaluates the top architectures using its predictor. **TNAS** (Shala et al., 2023) enhances the meta-learned dataset-aware predictor's adaptability to unseen datasets by utilizing BO with the deep-kernel GP strategy without involving any generation process (Please see Appendix C.2 for details of the baselines.). **DiffusionNAG** *conditionally generates* architectures with the diffusion model guided by the dataset-aware predictor. Our generation process, with a sampling batch size of 256, takes up to 2.02 GPU minutes on Tesla V100-SXM2 to sample one batch. Finally, we select the top architectures sorted by the predictor among the generated candidates. We conduct experiments on Transferable NAS benchmarks (Lee et al., 2021a) such as four unseen datasets - CIFAR-10, CIFAR-100, Aircraft (Maji et al., 2013), and Oxford IIT Pets (Parkhi et al., 2012) from large search space MBv3 (Table 1) and, NB201 (Table 2).

Results on MBv3 Search Space In Table 1, MetaD2A, TNAS, and DiffusionNAG obtain the top 30 neural architectures for each datasets, following the descriptions in the **Experimental Setup** section. Subsequently, we train these architectures on the datasets following the training pipeline described in Appendix C.5. Once the architectures are trained, we analyze the accuracy statistics for each method's group of architectures. Additionally, we calculate *p-value* to assess the statistical significance of performance differences between the architecture groups obtained via DiffusionNAG and each method. A *p-value* of 0.05 or lower denotes that a statistically meaningful difference exists in the performances of the generated architectures between the two groups.

The results demonstrate that, except for the Aircraft dataset, DiffusionNAG consistently provides architectures with superior maximum accuracy (**max**) compared to other methods across three

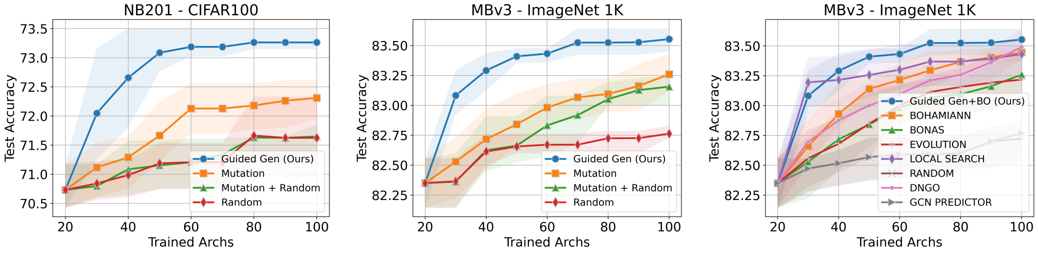
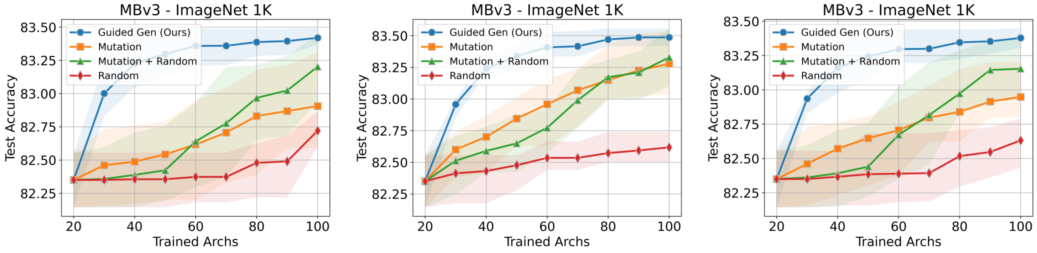


Figure 3: **Comparison Results on Existing AO Strategies.** *Guided Gen (Ours)* strategy provides a pool of candidate architectures, guiding them toward a high-performance distribution using the current population with DiffusionNAG. We report the results of multiple experiments with 10 different random seeds.



(a) EI (b) ITS (c) UCB

Figure 4: **Experimental Results on Various Acquisition Functions.** *Ours* consistently outperforms the heuristic approaches on various acquisition functions. We run experiments with 10 different random seeds.

datasets. Additionally, the **mean** accuracy and minimum accuracy (**min**) of architectures within the DiffusionNAG group are higher across all datasets. In particular, the *p-values* obtained from comparing the groups of architectures suggested by DiffusionNAG and those from other baselines are consistently below the 0.05 threshold across all datasets. This indicates that the architectures generated by DiffusionNAG have shown statistically significant performance improvements compared to those provided by the baseline methods when using transferable dataset-aware predictors. Furthermore, the results clearly support the superiority of the proposed predictor-guided conditional architecture generation method compared with either excluding predictors during generation (MetaD2A) or relying solely on predictors without generating architectures (TNAS).

Results on NB201 Search Space We highlight two key aspects from the results of Table 2. Firstly, the architectures generated by DiffusionNAG attain oracle accuracies of $\mathbf{94.37\%}$ and $\mathbf{73.51\%}$ on CIFAR-10 and CIFAR-100 datasets, respectively, and outperform architectures obtained by the baseline methods on Aircraft and Oxford-IIIT Pets datasets. While MetaD2A and TNAS achieve accuracies of $\mathbf{59.15\%/57.71\%}$ and $\mathbf{40.00\%/39.04\%}$ on Aircraft and Oxford-IIIT Pets datasets, respectively, DiffusionNAG achieves comparable or better accuracies of $\mathbf{58.83\%}$ and $\mathbf{41.80\%}$, demonstrating its superiority. Secondly, DiffusionNAG significantly improves the search efficiency by minimizing the number of architectures that require full training (**Trained Archs**) to obtain a final accuracy (For CIFAR-10 and CIFAR-100, an accuracy is retrieved from NB201 benchmarks) compared to all baselines. Specifically, when considering the Aircraft and Oxford-IIIT Pets datasets, DiffusionNAG only needs to train $\mathbf{3/2}$ architectures for each dataset to complete the search process while MetaD2A and TNAS require $\mathbf{40/26}$ and $\mathbf{6/40}$ architectures, respectively. This results in a **remarkable speedup** of at least $\mathbf{8.4}\times$ and up to $\mathbf{20}\times$ on average.

Further Anaylsis We further analyze the accuracy statistics of the distribution of architectures generated by each method within the NB201 search space. Specifically, we conduct an in-depth study by generating 1,000 architectures using each method and analyzing their distribution, as presented in Table 3 and Figure 2. We compare DiffusionNAG with two other methods: random architecture sampling (**Random**) and **MetaD2A**. Additionally, to assess the advantage of using a predictor in both the NAG and evaluation phases compared to an approach where the predictor is solely used in the evaluation phase, we unconditionally generate 10,000 architectures and then employ the predictor to select the top 1,000 architectures (**Uncond. + Sorting**). DiffusionNAG (**Cond.**) leverages the dataset-aware predictor $f_{\boldsymbol{\phi}}(\boldsymbol{D}, \boldsymbol{A}_t)$ to guide the generation process following Equation (10).

The results from Table 3 and Figure 2 highlight three key advantages of our model over the baselines. Firstly, our model generates a higher proportion of high-performing architectures for each target dataset, closely following the Oracle Top-1000 distribution within the search space. Secondly, our model avoids generating extremely low-accuracy architectures, unlike the baseline methods, which generate architectures with only 10% accuracy. This suggests that our model is capable of focusing on a target architecture distribution by excluding underperforming architectures. Lastly, as shown in Figure 2, DiffusionNAG (**Cond.**) outperforms sorting after the unconditional NAG process (**Uncond. + Sorting**). These results highlight the value of involving the predictor not only in the evaluation phase but also in the NAG process, emphasizing the necessity of our *conditional NAG* framework.

3.2 IMPROVING EXISTING BAYESIAN OPTIMIZATION-BASED NAS

In this section, we have demonstrated that DiffusionNAG significantly outperforms existing heuristic architecture sampling techniques used in Bayesian Optimization (BO)-based NAS approaches, leading to improved search performance in BO-based NAS.

BO-based NAS The typical BO algorithm for NAS (White et al., 2023) is as follows: 1) Start with an initial population containing neural architecture-accuracy pairs by uniformly sampling n_0 architectures and obtaining their accuracy. 2) Train a predictor using architecture-accuracy pairs in the population, and 3) Sample c candidate architectures by the Acquisition Optimization strategy (**AO strategy**) (White et al., 2021a) and choose the one maximizing an acquisition function based on the predictions of the predictor. 4) Evaluate the accuracy of the selected architecture after training it and add the pair of the chosen architecture and its obtained accuracy to the population. 5) Repeat steps 2) to 4) during N iterations, and finally, select the architecture with the highest accuracy from the population as the search result. (For more details, refer to Algorithm 1 in the Appendix.)

Our primary focus is on replacing the existing **AO strategy** in step 3) with DiffusionNAG to improve the search efficiency of BO-based NAS approaches. **Baseline AO Strategy**: The simplest AO strategy is randomly sampling architecture candidates (**Random**). Another representative AO strategy is **Mutation**, where we randomly modify one operation in the architecture with the highest accuracy in the population. **Mutation + Random** combines two aforementioned approaches. **Guided Gen (Ours)**: Instead of relying on these heuristic strategies, we utilize DiffusionNAG to generate the candidate architectures. Specifically, we train a predictor $f_{\phi}(y|\boldsymbol{A}_t)$, as described in Equation (8), using architecture-accuracy pairs in the population. The trained predictor guides the generation process of our diffusion model to generate architectures. We then provide these generated architecture candidates to the acquisition function in step 3) (See Algorithm 2 in the Appendix.)

Comparison Results with Existing AO Strategies The left and middle sides in Figure 3 illustrates our comparison results with existing AO strategies. These results clearly highlight the effectiveness of DiffusionNAG (**Guided Gen (Ours)**), as it significantly outperforms existing AO strategies such as **Random**, **Mutation**, and **Mutation + Random** on the CIFAR100 dataset from NB201 and the large-scale ImageNet 1K (Deng et al., 2009) dataset within the extensive MBv3 search space. In particular, BO-based NAS methods employing **Random** or **Mutation** strategies often suffer from the issue of wasting time on sampling low-quality architectures during the initial phase (White et al., 2020; Zela et al., 2022). In contrast, DiffusionNAG effectively addresses this issue by offering relatively high-performing architectures right from the start, resulting in a significant reduction in search times. As a result, as shown in the right side of Figure 3, our approach outperforms existing BO-based NAS methods, by effectively addressing the search cost challenge of them.

Experimental Results on Various Acquisition Functions In addition to the Probability of Improvement (**PI**) used in Figure 3, we investigate the benefits of DiffusionNAG across various acquisition functions, such as Expected Improvement (**EI**), Independent Thompson sampling (**ITS**), and Upper Confidence Bound (**UCB**) as shown in Figure 4. (Please see Appendix D.2 for more details on acquisition functions.). The experimental results verify that DiffusionNAG (**Ours**) consistently outperforms heuristic approaches, including **Mutation**, **Random**, and **Mutation + Random** approaches, on four acquisition functions: **PI**, **EI**, **ITS**, and **UCB**, in the large MBv3 search space.

3.3 THE EFFECTIVENESS OF SCORE NETWORK FOR NEURAL ARCHITECTURES

In this section, we validate the ability of the proposed score network to generate architectures that follow the distribution of NB201 and MBv3 search spaces. For NB201, we construct the training set

Table 4: **Generation Quality.** We generate 1,000 samples with each method for 3 runs of different seeds.

	NAS-Bench-201			MobileNetV3		
Method	Validity (%) ↑	Uniq. (%) ↑	Novelty (%) ↑	Validity (%) ↑	Uniq. (%) ↑	Novelty (%) ↑
GDSS Jo et al. (2022)	4.56±1.44	-	-	0.00±0.00	-	-
Ours (w/o Pos. Emb.)	**100.00**±0.00	**98.96**±0.49	49.08±2.05	42.17±1.80	**100.00**±0.00	**100.00**±0.00
Ours (w/ Pos. Emb.)	**100.00**±0.00	98.70±0.66	**49.20**±1.96	**100.00**±0.00	**100.00**±0.00	**100.00**±0.00

by randomly selecting 50% of the architectures from the search space, while for MBv3, we randomly sample 500,000 architectures. We evaluate generated architectures using three metrics (Zhang et al., 2019): **Validity**, **Uniqueness**, and **Novelty**. **Validity** measures the proportion of valid architectures generated by the model, **Uniqueness** quantifies the proportion of unique architectures among the valid ones, and **Novelty** indicates the proportion of valid architectures that are not present in the training set. As shown in Table 4, our score network generates valid architectures with **100% Validity**, whereas **GDSS** (Jo et al., 2022), a state-of-the-art graph diffusion model designed for *undirected* graphs, fails to generate valid architectures, with the validity of only **4.56%** and **0.00%** for NB201 and MBv3, respectively. Furthermore, our positional embedding yields significant improvements, indicating that it successfully captures the topological ordering of nodes within the architectures. Notably, in the MBv3, **Validity** improves from **42.17%** to **100.00%**, highlighting the necessity of positional embedding for generating architectures with a large number of nodes (a.k.a. "long-range"). Additionally, our framework generates **49.20%/100.00%** novel architectures that are not found in the training set, as well as unique architectures **98.70%/100.00%** for NB201 and MBv3, respectively.

4 RELATED WORK

Neural Architecture Search NAS is an automated architecture search process (Ning et al., 2021; Zoph & Le, 2017) and roughly can be categorized into reinforcement learning-based (Zoph & Le, 2017; Zoph et al., 2018; Pham et al., 2018), evolutionary algorithm-based (Real et al., 2019; Lu et al., 2020), and gradient-based methods (Luo et al., 2018; Liu et al., 2019; Dong & Yang, 2019b; Xu et al., 2020; Chen et al., 2021). Recently, Shala et al. (2023); Lee et al. (2021a) have proposed Transferable NAS to rapidly adapt to unseen tasks by leveraging prior knowledge. However, they still suffer from the high search cost. DiffusionNAG addresses these limitations by generating architectures satisfying the objective with a guidance scheme of a meta-learned dataset-aware predictor.

Diffusion Models Diffusion models, as demonstrated in prior work (Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021b), are designed to reverse the data perturbation process, enabling them to generate samples from noisy data. They have achieved success in a variety of domains, including images (Nichol et al., 2022; Rombach et al., 2022), audio (Jeong et al., 2021; Kong et al., 2021), and graphs (Niu et al., 2020b; Jo et al., 2022). However, existing diffusion models are not well-suited for Neural Architecture Generation (NAG) because their primary focus is on unconditionally generating undirected graphs. To overcome this limitation, this study introduces a conditional diffusion-based generative framework tailored for generating architectures represented as directed acyclic graphs that meet specified conditions, such as accuracy requirements.

5 CONCLUSION

This study introduced a novel conditional Neural Architecture Generation (NAG) framework called DiffusionNAG, which is the paradigm shift from existing NAS methods by leveraging diffusion models. With the guidance of a property predictor for a given task, DiffusionNAG can efficiently generate task-optimal architectures. Additionally, the introduction of a score network ensures the generation of valid neural architectures. Extensive experiments under two key predictor-based NAS scenarios demonstrated that DiffusionNAG outperforms existing NAS methods, especially effective in the large search space. We believe that our success underscores the potential for further advancements in NAS methodologies, promising accelerated progress in the development of optimal neural architectures. For reproducibility, we include "NAS Best Practices Checklist" in Appendix A.

Acknowledge This work was conducted by the Center for Applied Research in Artificial Intelligence (CARAI), funded by DAPA and ADD (grant number UD190031RD), and further supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under the Korean government (MSIT) (grant number 2022-0-00713).