NepScript Genesis: Efficient Neural Architecture Search Strategies for Devanagari Handwritten Digit Synthesis

Mausam Gurung, Prabin Neupane, Sajjan Acharya
Independent Researcher

Abstract

Manual design of Generative Adversarial Network (GAN) architectures is time-consuming, requires domain expertise, and typically explores less than 1% of possible architecture combinations. Repeated full training cycles further waste computational resources, making efficient architecture search critical for practical applications.

This work presents NepScript Genesis, a framework that investigates efficient Neural Architecture Search (NAS) strategies to automate GAN architecture discovery for Devanagari (Nepali) handwritten digit synthesis. We systematically compare five NAS algorithms—Random Search, Progressive Search, Adaptive Exploration, Multi-fidelity Search, and Adversarial Gradient-based Search—evaluating their search efficiency across architectural dimensions including latent space size, normalization schemes, activation functions, and dropout configurations.

Performance is assessed using Fréchet Inception Distance (FID), Inception Score (IS), Precision-Recall metrics, and a domain-specific Nepali script quality evaluation on 20,000 generated samples. Experimental results reveal that Adaptive Exploration achieves the best FID score (79.12) with high precision (0.98), while Adversarial NAS attains perfect precision (1.0) with FID of 109.29. Multi-fidelity Search demonstrates the highest diversity (IS: 2.08) while reducing computational overhead through early termination of unpromising architectures.

Key contributions include: (1) systematic comparison of NAS algorithm efficiency for GAN architecture search, (2) comprehensive multi-metric evaluation framework combining generation quality with computational efficiency, and (3) an open-source framework for reproducible NAS experimentation on underrepresented scripts. This work provides empirical guidance for selecting efficient NAS strategies in resource-constrained GAN development.

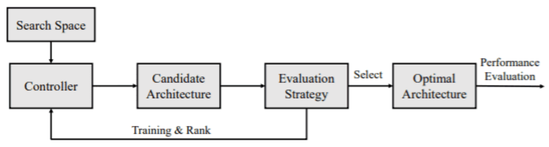
Introduction

Challenge: Manual GAN design is slow, resource-intensive, and highly suboptimal.
Solution: NepScript Genesis applies Neural Architecture Search (NAS) to automatically discover efficient GAN architectures.
Objective: Achieve an optimal trade-off between generation quality (FID) and computational efficiency.

Methodology: NAS Framework

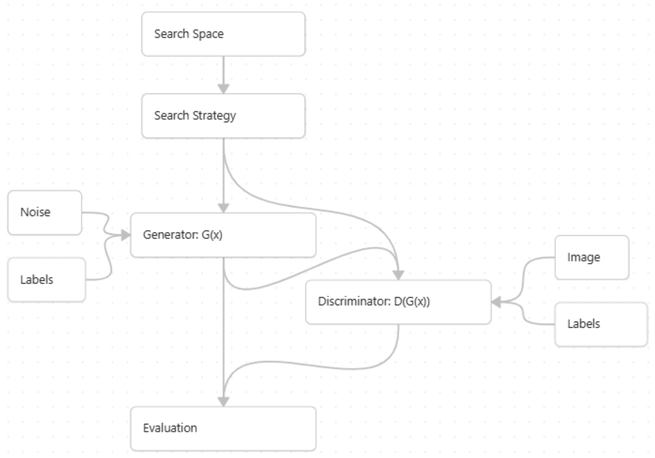
Neural Architecture Search replaces manual design by automatically identifying high-performing architectures.
Core Components:
- Search Space
- Search Strategy
- Performance Estimation

NAS Flow



NAS general framework [1]

NAS Framework



Generative Adversarial Network (GAN)

A GAN consists of two competing neural networks: a *Generator* that synthesizes realistic samples from random noise, and a *Discriminator* that distinguishes between real and generated data.
GAN Objective Function:

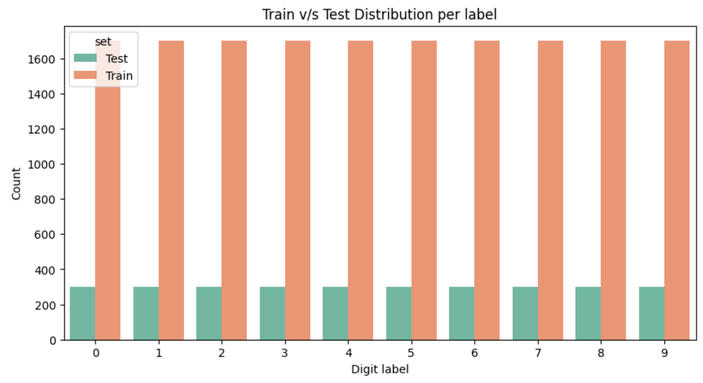
$$\min_G \max_D \; \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$$

Search Space

```
latent_dim      : [64, 100, 128, 200]
norm_type       : [batch, instance]
activation      : [relu, leaky_relu]
dropout_rate    : [0.0, 0.1, 0.2, 0.3]
use_residual    : [False, True]
```

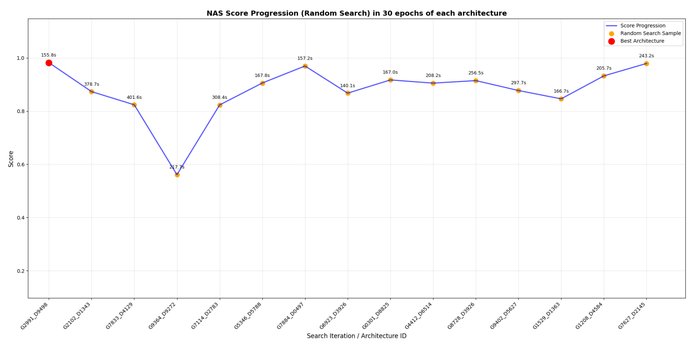
Dataset

- 20,000 handwritten Devanagari digit samples
- Balanced across 10 digit classes

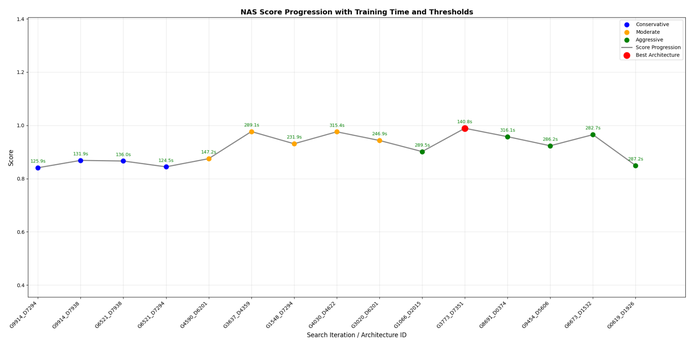


Search Strategies

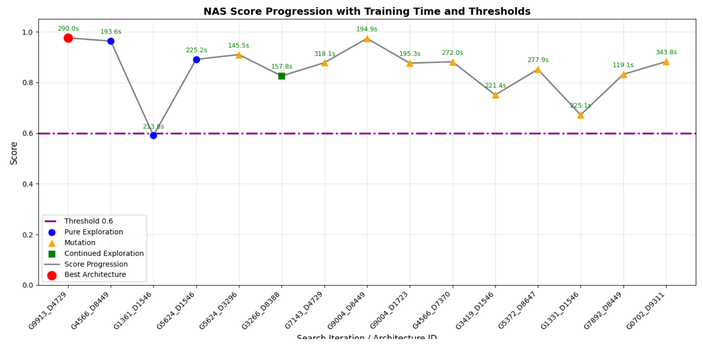
Random Search: Random sampling with no learning between iterations.



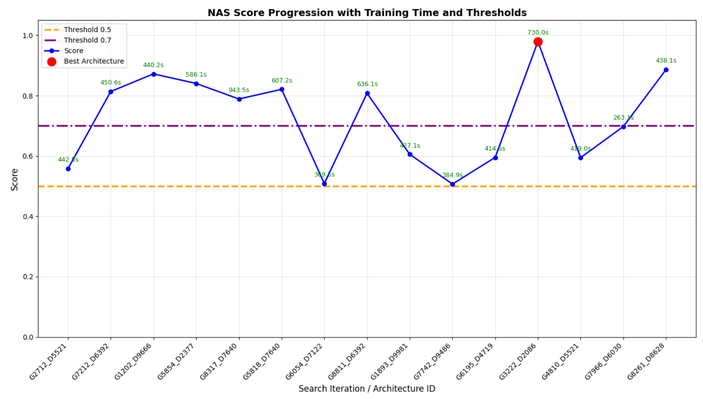
Progressive Search: Searches in phases, gradually increasing architectural complexity.



Adaptive Search: Balances exploration and exploitation using previous results.



Multi-fidelity Search: Trains models for fewer epochs initially and discards poor performers early.



Adversarial Gradient-based Search: Uses gradient information to jointly optimize generator and discriminator architectures.

Performance Metrics

- **Fréchet Inception Distance (FID):**
  Measures distributional similarity between real and generated images. Lower FID indicates better quality.

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r\Sigma_g}\right)$$

- **Inception Score (IS):**
  Evaluates image quality and diversity. Higher IS implies realistic samples.

$$\text{IS} = \exp\left(\mathbb{E}_{x\sim p_g}[D_{KL}(p(y|x)\|p(y))]\right)$$

- **Precision:**
  Quantifies the proportion of generated images that closely resemble real samples, reflecting sample realism.

$$\text{Precision} = \frac{\text{Generated images that look realistic}}{\text{Total generated images}}$$

- **Recall:**
  Measures how well the generator captures the full diversity of the real data distribution.

$$\text{Recall} = \frac{\text{Real image modes covered by generator}}{\text{Total real image modes}}$$

Experimental Results

NAS Method	FID	IS	Prec.	Recall
Random Search	124.81	1.85	0.999	0.515
Progressive Search	127.70	1.72	0.961	0.515
Adaptive Search	**79.12**	1.72	0.98	**0.531**
Multi-fidelity	123.67	**2.08**	**1.00**	0.512
Adversarial Search	109.29	1.83	**1.00**	0.504

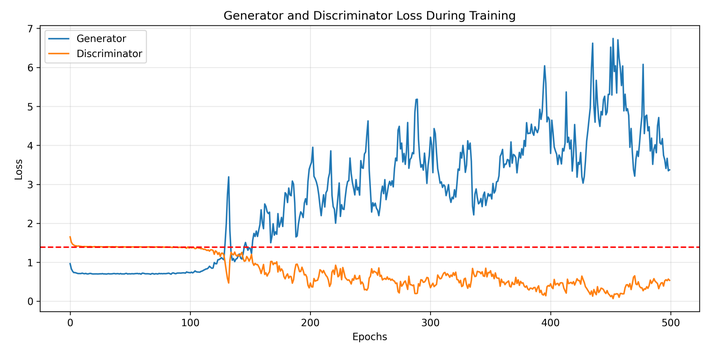
Key Findings

- **Adaptive Search** achieved the best image quality with lowest FID (**79.12**).
- **Adversarial NAS** attained perfect Precision (**1.00**).
- **Multi-fidelity Search** showed highest diversity (IS = **2.08**) with reduced computational cost.

Generated Samples (Adaptive Search)

Synthetic Digit Samples at Epoch 500

Training Loss Curve



Architecture Configuration
- Label Smoothing: 0.1, Learning Rate: 0.002, Batch size: 32

Generator:
- Latent: 128,
- Normalization: BatchNorm
- Activation Function: ReLU
- Dropout: 0.2,
- Residual: Enabled

Discriminator:
- Normalizaiton: BatchNorm
- Activation Function: LeakyReLU
- Dropout: 0.5
- Residual: Enabled

Conclusion

Adaptive Exploration is the most efficient NAS strategy for GAN architecture search, achieving the best balance between image quality and coverage. **Multi-fidelity Search** is ideal for applications prioritizing diversity with reduced computational cost. Intelligent NAS strategies significantly outperform random exploration.

References

[1] C. Gao et al., *AdversarialNAS: Adversarial Neural Architecture Search for GANs*, arXiv:1912.02037, 2019.
[2] S. Ioffe & C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, arXiv:1502.03167, 2015.
[3] J. Ba et al., *Layer Normalization*, arXiv:1607.06450, 2016.
[4] I. Goodfellow et al., *Generative Adversarial Networks*, arXiv:1406.2661, 2014.
[5] X. He, K. Zhao, & X. Chu, *AutoML: A Survey of the State-of-the-Art*, arXiv:1908.00709, 2019.
[6] R. Almeida, *Label Smoothing in Deep Learning*, Blog post, 2023. `almeidaraul.github.io`
[7] M. Mustafa, *GANs Specialization Part 5*, Medium, 2023.
[8] AI Summer, *Neural Architecture Search: A Survey*, 2023. `theaisummer.com`