GANji: A Framework for Introductory AI Image Generation

1st Chandon Hamel
Anderson College of Business and Computing
Regis University
Denver, Colorado
chamel@regis.edu

2nd Mike Busch
Anderson College of Business and Computing
Regis University
Denver, Colorado
mbusch@regis.edu

***Abstract*—The comparative study of generative models often requires significant computational resources, creating a barrier for researchers and practitioners. This paper introduces GANji, a lightweight framework for benchmarking foundational AI image generation techniques using a dataset of 10,314 Japanese Kanji characters. It systematically compares the performance of a Variational Autoencoder (VAE), a Generative Adversarial Network (GAN), and a Denoising Diffusion Probabilistic Model (DDPM). The results demonstrate that while the DDPM achieves the highest image fidelity, with a Fréchet Inception Distance (FID) score of 26.2, its sampling time is over 2,000 times slower than the other models. The GANji framework is an effective and accessible tool for revealing the fundamental trade-offs between model architecture, computational cost, and visual quality, making it ideal for both educational and research purposes.**

***Index Terms*—Generative Models, Generative Adversarial Networks, GAN, Denoising Diffusion Probabilistic Models, DDPM, Variational Autoencoder, VAE, Benchmarking, Kanji, Deep Learning**

I. Introduction

In the quest to understand and implement AI image generation techniques at a low computational cost, the choice of dataset significantly impacts both the feasibility and effectiveness of experimentation. Kanji characters, with their inherent logical structure and visual complexity, serve as an excellent medium for this purpose. Despite being represented as low-resolution, single-channel grayscale images, Kanji possess sufficient detail to effectively train and evaluate generative AI models. Their varied yet recognizable structures offer a straightforward means of qualitatively assessing model performance, allowing anyone with reasonable exposure to Japanese or Chinese writing to distinguish high-quality generated samples. This paper introduces **GANji**, a framework designed to serve as a "Hello, World" for researchers and practitioners entering the field of generative imaging, allowing for meaningful model comparison without prohibitive computational requirements.

Building on this premise, this study compares and analyzes the ability of three foundational generative models to produce Kanji-like images: the Variational AutoEncoder (VAE), the Generative Adversarial Network (GAN), and the Denoising Diffusion Probabilistic Model (DDPM). These models were trained on a dataset of 10,314 Kanji characters to evaluate the impact of hyperparameter adjustments and architectural modifications, thereby comparing the relative strengths and weaknesses of each technique. The emergence of coherent Kanji-like structures in sampled outputs served as a continuous indicator of model performance during training, and the quality of the final generated images could be readily assessed. Ultimately, this work provides both a methodology for comparing image generation models under strict resource constraints and a direct comparison of these models' capabilities on this unique and challenging dataset.

II. Methodology

A. Tools

1) Hardware: Experiments were conducted on a personal machine running Ubuntu 22.04 LTS and equipped with an NVIDIA RTX 4070 GPU (12 GB VRAM). This hardware was selected to demonstrate the framework's feasibility on consumer-grade equipment, which offers significant computational power while having known limitations compared to industrial-grade accelerators.

2) Software Environment: The project was implemented in Python 3.9. Model architectures were defined using PyTorch 2.0, with PyTorch Lightning employed for managing training loops and hardware acceleration. Additional libraries included NumPy for numerical operations and Matplotlib for visualization. To evaluate model performance, Fréchet Inception Distance (FID) scores were computed using the `clean-fid` Python package [7].

3) Version Control and Reproducibility: To ensure reproducibility, the project was managed with Git for version control. The complete source code is publicly available on GitHub at: https://github.com/Feebami/GANji.

4) Data Management: The dataset was stored locally. Data loading was handled by the PyTorch `DataLoader`, configured with prefetching and parallel processing to ensure efficient batch retrieval during training.

B. Dataset and Preprocessing

1) Dataset Characteristics: The study utilizes the Kanji Dataset, a collection of 10,314 black-and-white character images sourced from Kaggle [2]. Each image has a resolution of

48×48 pixels and is rendered in a standardized Mincho-style font. This choice of a synthetic, typographically consistent dataset over handwritten samples was deliberate; it controls for artistic variation and ensures clean topological features. This is a crucial decision for a study focused on comparing the architectural merits of generative models rather than their ability to replicate handwriting. Figure 1 displays a sample of characters from the dataset, illustrating their structural complexity.

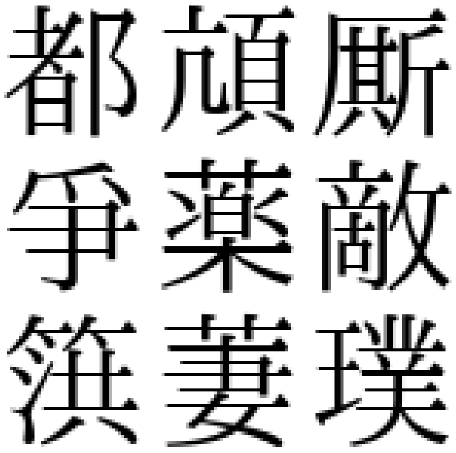

Fig. 1. A 3×3 grid of sample Kanji characters from the dataset [2].

2) Preprocessing and Data Loading: A standardized preprocessing pipeline was applied before training. First, images were upscaled to 64×64 pixels using bilinear interpolation to balance detail preservation with computational load. Next, they were converted to single-channel ($1 \times 64 \times 64$) PyTorch tensors. Finally, pixel values were normalized based on model requirements: to a $[0, 1]$ range for the VAE and a $[-1, 1]$ range (using $\mu = 0.5, \sigma = 0.5$) for the GAN and DDPM.

The models were trained using a batch size of 128 for the VAE and GAN, and 32 for the DDPM due to its higher memory requirements. The data loader was configured with four worker processes, pinned memory, and a prefetch factor of two to ensure efficient GPU utilization. Notably, no data augmentation techniques such as rotation or flipping were applied, as the goal was to preserve the precise structural integrity of the characters for pure generation.

C. Model Architectures and Training

1) Experimental Design: Three generative approaches were implemented and compared: a Variational AutoEncoder (VAE), a Generative Adversarial Network (GAN), and a Denoising Diffusion Probabilistic Model (DDPM). For each architecture, critical hyperparameters were systematically tested, including latent dimension size (VAE), loss functions (GAN), and noise schedules (DDPM). Model selection was guided by the need for computational efficiency on the available hardware. All models were trained for 100 epochs, after which they generated 10,368 images to ensure a sufficient sample size for accurate Fréchet Inception Distance (FID) score calculation.

2) Variational Autoencoder (VAE): The first model evaluated was a VAE featuring a symmetric residual architecture for its encoder and decoder. Each component was composed of seven residual blocks that progressively adjusted the number of convolutional channels from 64 up to 512 and back down to 64. A standard residual block contained two 3x3 convolutional layers, batch normalization, and ReLU activations, along with a shortcut connection to improve gradient flow. The encoder performed downsampling using convolutional strides of 2, while the decoder used transposed convolutions with corresponding strides for upsampling.

To investigate the impact of representational capacity, the VAE was tested with latent space dimensions of 64, 128, and 256. The model was trained by minimizing a loss function comprising a reconstruction term and a regularization term, as shown in Equation (1):

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}[\log p(x|\hat{x})] + D_{KL}(q(z|x)||p(z)) \tag{1}$$

where the first term is the binary cross-entropy (BCE) reconstruction loss and the second is the Kullback-Leibler (KL) divergence, which regularizes the latent space to follow a standard normal distribution [4].

Training was performed using the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) and employed bf16 mixed-precision. A one-cycle learning rate schedule was used, starting at 4×10^{-5}, increasing to a maximum of 1×10^{-3} over 30 epochs, and finally annealing down to 1×10^{-7}.

3) Generative Adversarial Network (GAN): The GAN's architecture consisted of a generator and a discriminator, both built with residual blocks and initialized using Xavier uniform initialization. The **generator** was composed of five residual blocks that progressively decreased the channel count from 512 to 64. Each block contained two 3x3 convolutional layers, batch normalization, and ReLU activations, with transposed convolutions used for upsampling. A final Tanh activation layer scaled the output pixels to the required $[-1, 1]$ range.

The **discriminator** comprised four residual blocks that increased the channel count from 32 to 512. It used standard convolutions with a stride of 2 for downsampling and LeakyReLU activation functions (negative slope of 0.2). A final linear layer produced a single logit as its output.

To identify the most effective objective for training, three distinct **loss function** formulations were evaluated. The first was the standard GAN objective using Binary Cross-Entropy (BCE) loss. The second was the Wasserstein GAN with Gradient Penalty (WGAN-GP) loss, which minimizes the Wasserstein distance and uses a gradient penalty term to improve training stability [1]. The third was a hinge loss formulation, another common variant for stabilizing GAN training [5].

The **training protocol** used separate Adam optimizers for the generator ($\beta_1 = 0.0, \beta_2 = 0.99$) and discriminator ($\beta_1 = 0.9, \beta_2 = 0.999$), with a constant learning rate of 1×10^{-4} for both. To balance the training dynamics, an adaptive discriminator update ratio was implemented: the discriminator was updated five times for every generator update. However, this inner loop would break prematurely if the discriminator's accuracy on a batch exceeded 90%, preventing it from overpowering the generator and speeding up training. All GAN models were trained using full 'float32' precision due to the sensitivity of the training process.

4) Denoising Diffusion Probabilistic Model (DDPM): The third approach was a DDPM implemented with a U-Net architecture [8]. The network's structure included downsampling, bottleneck, and upsampling paths, each containing three custom residual blocks. Downsampling was performed using 2x2 max pooling, while upsampling utilized 2x bilinear interpolation with corner alignment. The main paths were preceded and followed by single convolutional layers.

Each residual block was composed of three 3x3 convolutional layers, 4-group group normalization, and ReLU activations. To incorporate the diffusion time step, sinusoidal position embeddings were processed by a linear layer and added to the output of the convolutional layers, along with the original residual shortcut connection. Filter dimensions increased from 64 to 256 in the downsampling path, rose to 512 in the bottleneck, and decreased from 512 to 64 in the upsampling path. Skip connections concatenated feature maps from the downsampling path to corresponding blocks in the upsampling path.

Two noise schedules were evaluated over 1024 time steps: a linear schedule increasing from 1×10^{-4} to 2×10^{-2} [3], and a cosine schedule [6]. The model was trained to predict the noise added at a given time step t by minimizing the simplified loss objective [3]:

$$L_{\text{simple}} = \mathbb{E}_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right] \tag{2}$$

where ϵ is the true noise and ϵ_θ is the noise predicted by the model. Training used the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) with bf16 mixed-precision and a cosine annealing learning rate schedule that started at 5×10^{-5} and decayed to zero.

D. Evaluation Metrics

The performance of each model was assessed across several key areas. Generative quality was quantified using the **Fréchet Inception Distance** (FID) score. Computational efficiency was evaluated by measuring **training time**, **sampling time** (to generate 10,368 images), and peak **VRAM usage**. Finally, model complexity was compared by reporting the total number of trainable **model parameters**.

III. Results

The quantitative results of all experiments are summarized in Tables I, II, and III. The data reveals that the DDPM models achieved markedly superior performance in terms of Fréchet Inception Distance (FID). The top-performing DDPM configuration (linear beta schedule) yielded an FID score of **26.2**, significantly outperforming the other architectures. For comparison, the best GAN (BCE loss) and VAE (latent dimension 64) achieved considerably higher (i.e., worse) FID scores of **74.6** and **79.9**, respectively.

A clear trade-off emerges when analyzing computational costs. The DDPM's high-fidelity generation required an extremely slow sampling phase, taking over 100 minutes to generate the evaluation set. This is more than 2,000 times slower than the GAN, which required only 3 seconds. In terms of training efficiency, the VAEs were the fastest (14 minutes). Conversely, the GANs, despite having the fewest parameters (8.7M), demanded the longest training times, with the WGAN-GP variant being the most computationally expensive (184.1 minutes).

However, a qualitative assessment of the generated images (Figures 2 through 9) highlights a notable discrepancy between the quantitative FID scores and the perceptual quality of the outputs. Images generated by the VAE (Figs. 2-4) are consistently characterized by a soft, blurry appearance that obscures fine structural details. In contrast, the GAN-generated images (Figs. 5-7), while algorithmically scoring similar to the VAE's, exhibit sharper edges and more defined stroke-like features, rendering them more visually plausible as Kanji characters than their VAE counterparts in this author's qualitative analysis.

TABLE I
VAE Results

Latent space size	**64**	**128**	**256**
FID	79.9	91.3	86.9
Training Time (min)	14.4	14.0	14.4
Sampling Time (sec)	4	4	4
VRAM Usage (MB)	2458	2464	2560
Model Parameters (M)	22.2	23.8	26.9
Generated Images	Fig. 2	Fig. 3	Fig. 4

TABLE II
GAN Results

Discriminator Loss	**BCE**	**Wasserstein**	**Hinge**
FID	74.6	78.9	78.4
Training Time (min)	48.8	184.1	69.2
Sampling Time (sec)	3	3	3
VRAM Usage (MB)	4190	6046	6046
Model Parameters (M)	8.7	8.7	8.7
Generated Images	Fig. 5	Fig. 6	Fig. 7

TABLE III
DDPM Results

Configuration	**Linear Beta**	**Cosine Beta**
FID	26.2	29.1
Training Time (min)	31.8	31.2
Sampling Time (min)	101.7	101.7
VRAM Usage (MB)	1806	1848
Model Parameters (M)	23.8	23.8
Generated Images	Fig. 8	Fig. 9

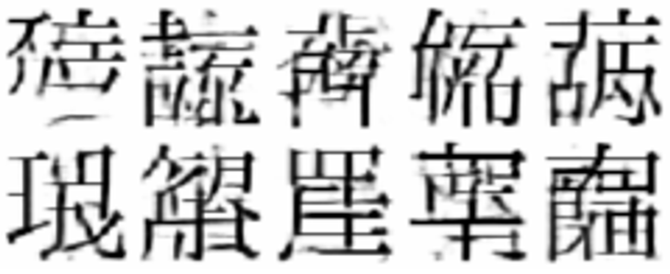
Fig. 2. Generated images of VAE with 64 dimensional latent space

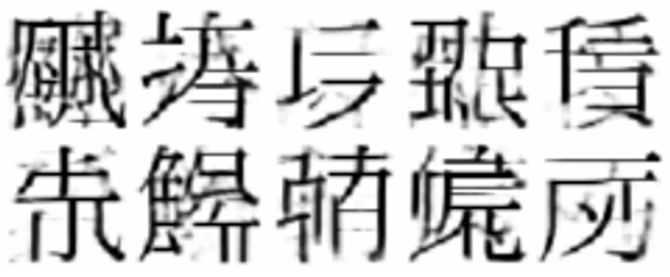
Fig. 3. Generated images of VAE with 128 dimensional latent space

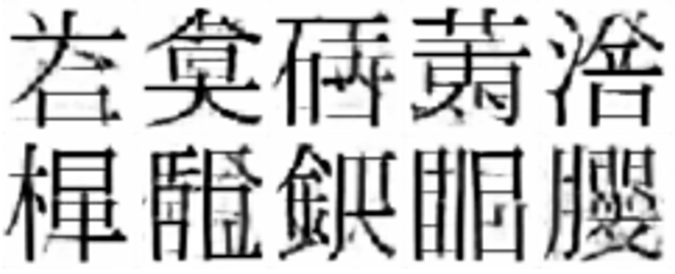
Fig. 6. Generated images of GAN using Wasserstein distance loss

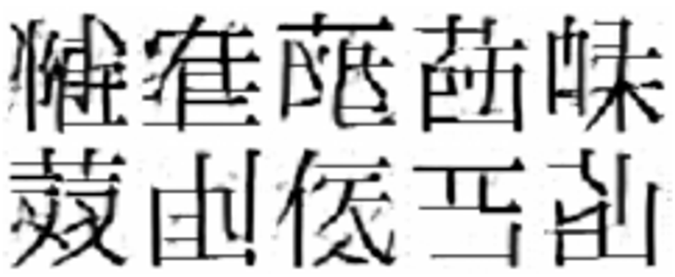
Fig. 7. Generated images of GAN using Hinge loss

IV. Discussion

The results of this study offer a clear comparative analysis of VAEs, GANs, and DDPMs within the specific context of Kanji generation. The superior FID scores of the DDPMs align with established findings that diffusion models excel at high-fidelity, pixel-perfect image synthesis. However, this quality comes at a steep computational price, with sampling times that are orders of magnitude slower than other methods, rendering them impractical for many real-time applications.

Perhaps the most salient finding is the observed paradox between quantitative metrics and qualitative assessment. GANs, despite their challenging training dynamics and susceptibility to mode collapse, produced characters with sharp, defined strokes. In contrast, VAEs, which were the simplest to implement and train, generated blurry, indistinct forms. This suggests that the FID score, while a standard benchmark, may not fully capture the features that are most critical for perceptual quality in structured, line-based data like Kanji.

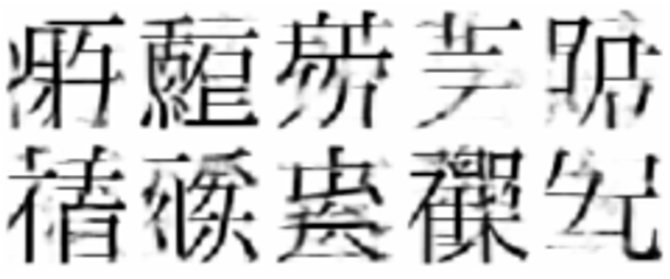
Fig. 4. Generated images of VAE with 256 dimensional latent space

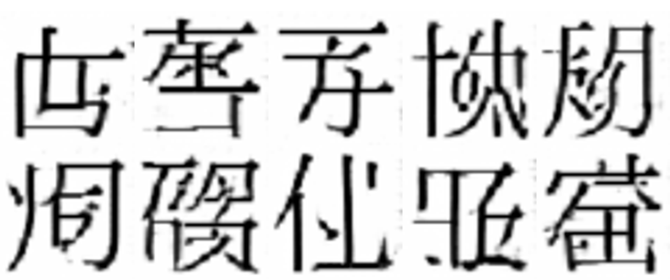
Fig. 5. Generated images of GAN using binary cross entropy loss

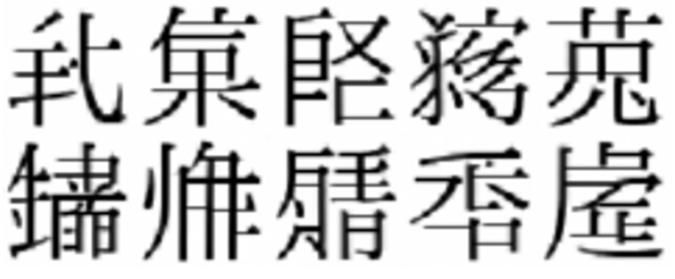
Fig. 8. Generated images of DDPM with linear beta

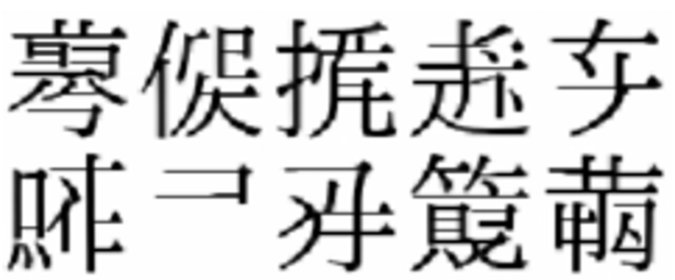
Fig. 9. Generated images of DDPM with cosine beta

For such datasets, human evaluators appear to prioritize structural integrity and sharpness over the pixel-level distribution matching what FID measures.

These findings validate the proposed **GANji** framework as an effective “Hello, World” for comparative studies in generative AI. The Kanji dataset [2] proved to be an excellent medium, as its inherent structure made the characteristic failure modes of each model—the VAE’s blurriness, the GAN’s occasional malformations, and the DDPM’s residual noise—immediately apparent. The framework successfully highlights the fundamental trade-offs between model complexity, training stability, resource consumption, and the nature of the generated output.

Nonetheless, this study has limitations. The qualitative evaluation of ”Kanji-ness” is inherently subjective and benefits from familiarity with the characters. Furthermore, the use of a single, clean font does not test the models’ robustness to stylistic variations found in handwritten data. Future work could address these points by developing a quantitative metric for structural coherence, conducting user studies for perceptual scoring, and applying the framework to more diverse or handwritten character sets.

V. Conclusion

This paper introduced and evaluated GANji, a lightweight framework for benchmarking generative models on the task of Kanji character generation. Through systematic experiments, it was demonstrated that while DDPMs achieve superior image fidelity as measured by FID, they are hindered by extreme sampling latency. Critically, this work highlighted a significant discrepancy between standard metrics and human perception, as GANs produced more visually coherent characters than VAEs despite having similar FID scores. Ultimately, the GANji framework proves to be a valuable and accessible tool for both educational purposes and preliminary research, effectively revealing the distinct performance profiles and trade-offs of foundational generative architectures.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML]. URL: https://arxiv.org/abs/1701.07875.

[2] frammie. *Unlabeled Kanji/East Asian Characters*. 2021. URL: https://www.kaggle.com/datasets/frammie/kanjieast-asian-characters.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[4] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[5] Jae Hyun Lim and Jong Chul Ye. *Geometric GAN*. 2017. arXiv: 1705.02894 [stat.ML]. URL: https://arxiv.org/abs/1705.02894.

[6] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG]. URL: https://arxiv.org/abs/2102.09672.

[7] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. “On Aliased Resizing and Surprising Subtleties in GAN Evaluation”. In: *CVPR*. 2022.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597.