Dressing the Imagination: A Dataset for AI-Powered Translation of Text into Fashion Outfits and A Novel NeRA Adapter for Enhanced Feature Adaptation

Supplementary Material

This supplementary material includes the following sections:

- **Additional Qualitative Results**
- **Human Evaluation Results**
- **Ablation Study**
- **FLORA Dataset**
- **Fine-Tuning Template for Outfit Sketch Generation**
- **Implementation and FLORA Test Set Details**
- **NeRA vs. LoRA: A Mathematical Justification**

7. Additional Qualitative Results

Figure 8 provides additional qualitative comparison of zero-shot, MLP-based adapters and the proposed NeRA using FLUX and Stable Diffusion 3 as baselines. These examples illustrate model performance on a variety of fashion prompts, covering diverse silhouettes, color schemes, and accessory details.

8. Human Evaluation Results

Human Evaluation Metrics For a thorough evaluation of the generated fashion sketches, we defined seven key metrics, each capturing specific aspects of image quality and alignment with the prompt. Table 3 presents human evaluation results. These metrics allowed us to perform a comprehensive assessment, ensuring that both technical quality and alignment with the prompt are rigorously evaluated:

- **Prompt Relevance:** Evaluates how accurately the generated image reflects the essential elements described in the prompt. This metric focuses on effectively visualizing the main idea or theme specified.
- **Visual Quality:** Assesses the technical quality of the image, including factors such as resolution, sharpness, clarity, and the overall visual appeal.
- **Creativity:** Measures the degree of innovation and imaginative interpretation in the image. This metric considers how the generated image demonstrates creativity while staying coherent with the prompt's details.
- **Pose and Model Accuracy:** Examines the alignment between the model's physical representation in the image, such as pose, gender, and stance and the specifications given in the prompt.
- **Color Palette Alignment:** Compares the colors used in the generated image to the colors described in the prompt, assessing accuracy in terms of color scheme and consistency.
- **Contextual Fit:** Evaluates how well the image fits the broader context implied by the prompt, including aspects like mood, setting, and style.
- **Overall Appeal:** Provides a holistic assessment of the image's attractiveness and visual impact, taking into account its general appeal to viewers.

LoRA Rank		
LoRA Rank	**FID ↓**	**CLIP SIM ↑**
$r = 8$	8.76	0.2711
$r = 16$	8.13	0.2832
$r = 32$	7.92	0.2901
$r = 64$ (Default)	**7.88**	**0.2987**
NeRA's Downsample Rate		
Downsample Rate	**FID ↓**	**CLIP SIM ↑**
$m/16$	7.18	0.3298
$m/8$	6.66	0.3319
$m/4$	6.49	0.3401
$m/2$ (Default)	**6.05**	**0.3412**
Finetuning Techniques		
	FID ↓	**CLIP SIM ↑**
DreamBooth	8.37	0.3217
Textual Inversion	8.88	0.3158

Table 7. Ablation study comparing the effect of varying ranks and downsample rate for LoRA and NeRA respectively, also different finetuning techniques using FLUX as the baseline.

9. Ablation Study

In this section, we investigate the use of NeRA adapters as an alternative to LoRA. Extending our previous ablation studies, we explore the effect of varying ranks for LoRA and downsample rates for NeRA. For the ablation, we choose FLUX as our baseline model for conducting the experiments. Table 7 provides the CLIP-SIM and FID scores for different ranks and downsampling rates for the respective adapters.

Effect of Rank on LoRA Performance: For LoRA, we observe a progressive improvement in both CLIP-SIM and FID scores as the rank increases. The model achieves the highest performance at rank $r = 64$, yielding a CLIP-SIM of 0.2987 and an FID of 7.88. This indicates that increasing the rank allows LoRA to better capture domain-specific details relevant to our sketch generation task.

Effect of Downsample Rate on NeRA's Performance: The downsample rate in the NeRA refers to the dimensionality reduction applied to the input features. Specifically, as explained in proposed methodology section, the NeRA incorporates two main functions: the ϕ_{down} and ϕ_{up}. ϕ_{down}

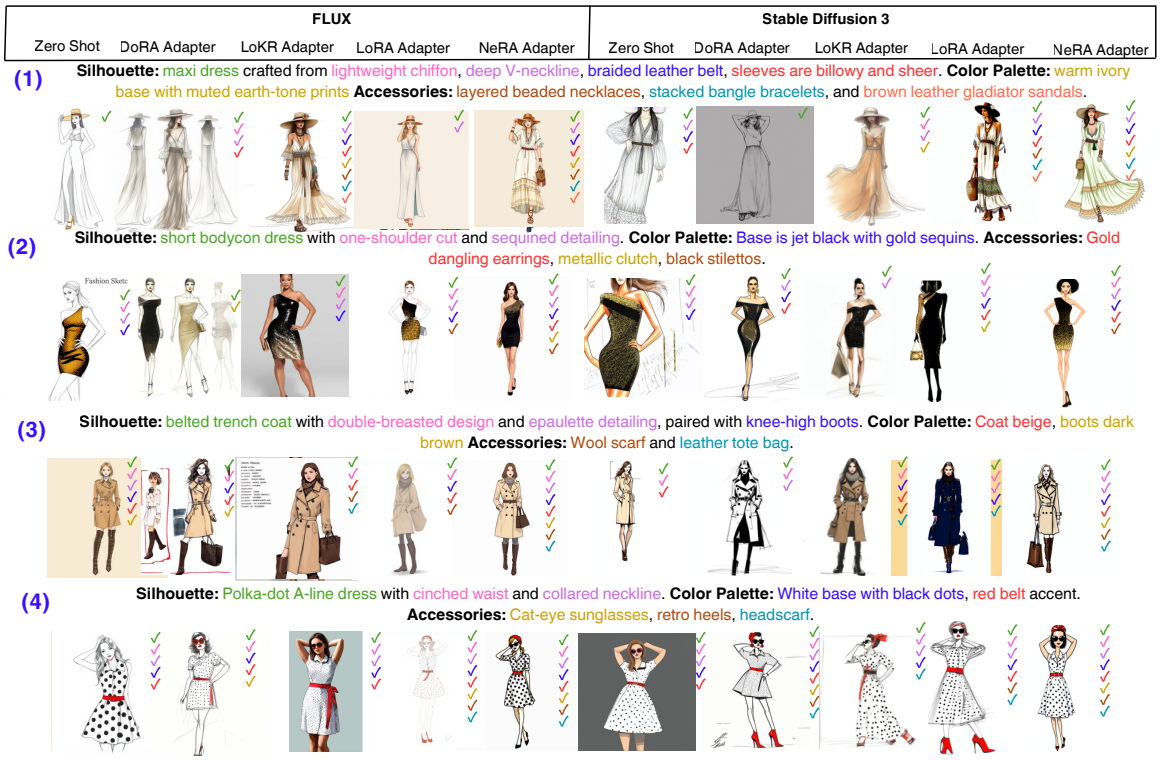

Figure 8. Additional qualitative results comparing NeRA, LoRA, and zero-shot performance. Colored arrows mark whether the prompt elements written in that color were correctly generated.

projects the input features from the original dimensionality (m) to a reduced dimensionality(n), and the ϕ_{up} projects the reduced-dimensional features back to their original size m. Here, n is the downsampling rate, which is hyper-parameter and is decided by the ratio by which we want to reduce dimension (e.g. $m/16$, $m/8$, $m/4$, $m/2$). Where a smaller downsample rate ($m/16$) implies greater reduction and larger rates ($m/2$) retain more of the original feature space.

The ablation study, as shown in the table 7, evaluates the performance of NeRA at different downsample rates. As the downsampling rate increases from $m/16$ to $m/2$ results in improvements in both CLIP-SIM and FID scores. This progression highlights the trade-off between dimensionality reduction and feature retention, with higher downsample rates striking an optimal balance between computational efficiency and model performance. These findings emphasize the importance of selecting an appropriate downsample rate to maximize the NeRA's effectiveness.

Comparison with Existing Personalization Methods: To assess the strengths of our adapter-based fine-tuning approach, we compare it against two widely used personalization techniques - DreamBooth [38] and Textual Inversion [9] each representing different ends of the efficiency fidelity trade-off. Both methods were evaluated using the Flux variant of our model. DreamBooth achieved a FID of 8.37 and a CLIP-SIM of 0.3217, while Textual Inversion scored 8.88 and 0.3158, respectively. Although both methods outperform zero-shot baselines, they fall short of the performance achieved by Flux fine-tuned with our LoRA and NeRA (see Table 1 from main paper). DreamBooth, despite its high-fidelity generations, tends to overfit in few-shot settings and struggles with abstract or compositional prompts. Textual Inversion, on the other hand, lacks the flexibility needed for task-specific or multi-domain generation. In contrast, adapter-based approach, featuring multi-stage adaptation and prompt-aligned tuning more effectively captures compositional semantics and delivers superior generalization across tasks.

10. FLORA Dataset

10.1. Ethical and legal considerations in use of FLORA

The images included in FLORA were sourced from publicly available platforms. This dataset is strictly intended for non-commercial, research purposes, and its use must comply with copyright and intellectual property laws.

10.2. Dataset Creation and Filtering Process

To create the FLORA dataset, we employed a multi-stage filtering and description process to ensure that the images

met quality standards and contained only the desired fashion elements. Below, we describe the stages of filtering, noise removal, and detailed description generation, along with the prompts used at each step.

Initial Filtering of Noise Using VLM Model: In the initial filtering stage, we used a VLM model (LLaVa 32b) to identify and remove images that did not meet the criteria for inclusion in the dataset. Specifically, we aimed to retain images that contained a single outfit sketch without extraneous objects. To achieve this, we provided each image with the following prompts and used the model's responses to segregate the data:

Prompt 1: *"Is there a single outfit present in the image? Yes or No?"*
Prompt 2: *"Is the outfit present in the image a sketch? Yes or No?"*
Prompt 3: *"Does the image only contain the sketch without any other objects? Yes or No?"*
Prompt 4: *"Are there multiple outfits present in the image? Yes or No?"*

If the VLLM model answered "Yes" to the questions indicating the desired characteristics (single outfit, sketch format, no additional objects, and no multiple outfits), the image proceeded to the next filtering stage. If the answer to any of these prompts was "No," the image was discarded.

Watermark Detection and Removal: For images that passed the initial filtering, we implemented an additional check to detect and remove watermarks, which could interfere with model training. At this stage as well we utilized the LLaVa 32b with the following prompt:

Prompt: *"Does this image contain a watermark? A watermark may appear as text or a logo superimposed over the image. Yes or No"*

If the model identified a watermark ("Yes" response), we performed image inpainting to remove the watermark, as described in the main paper. This ensured that the images in the dataset were clean and free from visual obstructions.

Input Prompt Generation: Once the images were filtered, we used GPT-4o to generate precise and comprehensive descriptions of each outfit. This process involved a multi-turn approach, where each turn focused on describing a specific aspect of the image. The following prompts were used to extract detailed information:

Model Pose Description: "Describe the model's posture. Is the model standing, sitting, walking, or in any other specific pose? Detail the position of the model's hands, legs, and overall body stance. Provide enough detail so that someone reading the description could recreate the sketch with high accuracy. Note: Generate a response in a single paragraph with not more than 50 words. Please note that - while describing pose, do not describe the dress the model is wearing or do not use any adjectives. Just in simple words, explain the pose of the model, nothing else."

Outfit Description: "Describe the outfit present in the image. Describe in a way that a fashion designer can read that description and create an exact similar sketch of the outfit. Use fashion design terminology. Describe garment type, fabric, cut, and any special features. Note: Generate a response in a single paragraph with not more than 200 words."

Color Details: "Provide a detailed description of the color scheme used in the outfit. Mention the primary color, any secondary colors, and any unique color patterns. Note: Generate the response in a single paragraph with not more than 50 words. Just return color-related information."

Accessories Description: "Mention and describe any accessories or additional design elements included in the sketch, such as belts, shoes, jewelry, or headpieces. Provide enough detail so that someone reading the description could recreate the sketch with high accuracy. Note: Generate the response in a single paragraph with not more than 50 words. If no accessories are present, respond with 'no additional accessories present'."

This multi-turn approach allowed us to capture specific details of each image, resulting in a dataset that accurately reflects the visual and stylistic characteristics required for high-quality, fashion-focused AI training.

Additionally, we extracted information on the presence and gender of the model in each image using LLaVa 32b. By using this layered approach - initial filtering, watermark detection, and detailed multi-turn descriptions; we created a high-quality, fashion-specific dataset (FLORA) that supports the generation of accurate and stylistically rich images from textual descriptions. This structured dataset creation process ensures that the FLORA dataset is diverse, precise, and highly applicable to fashion-focused generative AI tasks.

10.3. Dataset Insights and Diversity Analysis

Figure 9 summarizes the gender representation and model presence in the FLORA dataset. Of the images, 97.5% (4,220) feature female outfits, 0.27% (12) male outfits, and 2.26% (98) show "no gender," where the outfit is displayed without a model. Similarly, 97% of the images include a model wearing the outfit, while 3% display only the outfit. These distributions in the FLORA dataset are influenced by the types of fashion images commonly found on the internet. For example, the high percentage of female outfits and model presence results from the dataset being sourced from online data, where such trends are prevalent.

Our dataset consists of a wide array of fashion outfit styles, which we categorize into nine distinct classes, each with its own set of sub-classes. These terminologies include **Garment Types** comprising 29 sub-classes such as *blazers, gowns, hoodies, kimonos, skirts and jackets*. The **Styles & Details** class has 17 sub-classes including attributes like *sleeveless, bateau, turtle-neck and halter*. **Materials & Patterns** with 21 sub-classes describes fabric types and patterns like *silk, cotton, denim, satin, floral and striped*. **Construction & Detailing** includes 40 sub-classes covering elements such as *embroidery, beading, embellishments, metallic, cinched, knitting and vintage or retro aesthetics*. The **Technical & Functional** class has five sub-classes which focus on functional features such as *breathability, lightweight materials and detachable components*. **Occasion & Aesthetic** includes 18 sub-classes categorizing outfits by their intended use and style such as *casual, formal, traditional, sophisticated, beachwear and whimsical*. **Construction Techniques** with 10 sub-classes includes garment construction methods like *stitching, buttoning, zippers and lining*. **Detailing Techniques** with 19 sub-classes describes specific design techniques like *applique, beading, draping, gathering, pleating and layering*. Finally, the **Finishing Techniques** class has five sub-classes that address finishing touches like *fading, top-stitching and softening*. The distribution of these terminologies across the dataset is presented in log-scaled figures (10 - 18), which provide an insightful representation of their relative prevalence.

11. Fine-Tuning Template for Outfit Sketch Generation

We developed a structured prompt template for fine-tuning the models. This template was designed to capture essential aspects of each outfit sketch, including model presence, gender, pose, outfit description, color details and accessories. By standardizing these attributes, we aimed to provide the model with clear, detailed inputs that could guide the generation process toward accurate, high-quality fashion sketches. The template used is as follows:

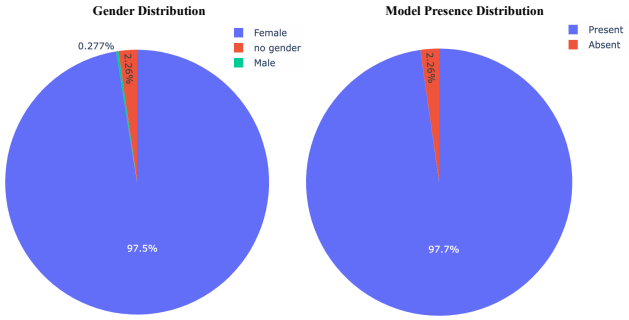

Figure 9. Distribution of gender representation and model presence in the FLORA dataset.

```
MODEL PRESENCE: {MODEL_PRESENCE}
MODEL GENDER: {MODEL_GENDER}
MODEL POSE: {MODEL_POSE}
OUTFIT DESCRIPTION: {OUTFIT_DESCRIPTION}
COLOR DETAILS: {COLOR_DETAILS}
ACCESSORIES: {ACCESSORIES}
```

12. Implementation and FLORA Test Set Details

Implementation Details: All models were fine-tuned with a batch size of 12 using gradient accumulation steps of 12, leveraging the AdamW optimizer for enhanced generalization through weight decay. The initial learning rate was set to 1×10^{-5} and dynamically adjusted with a cosine annealing schedule, gradually varying between 1×10^{-7} to 2×10^{-3}. This scheduling approach allowed for smooth convergence by reducing the learning rate as training progressed. The training was conducted on a robust setup of 4 x 8 Nvidia H100 GPUs with 80GB VRAM. For all the baselines, elastic weight consolidation [21] was performed to determine where to add LoRA and NeRA.

FLORA Test Set: To demonstrate practical relevance of FLORA, we created a test set of 500 innovative fashion outfit descriptions. Each description was generated using a structured prompt (refer supplementary) with GPT-4o [30], incorporating standard industry-specific terminology. We introduce novel and unique design elements in the prompts to assess the models' capability to interpret both familiar and innovative fashion concepts. Thus, our experiments showcase dataset's effectiveness in real-world applications.

13. NeRA vs. LoRA: A Mathematical Justification

This section provides mathematical proof demonstrating why Nonlinear low-rank Expressive Representation Adapters (NeRA) are strictly more expressive than Low-Rank Adaptation (LoRA) adapters. We begin by formally defining LoRA and NeRA, followed by a Taylor series-based proof demonstrating why NeRA provides a superior

function approximation. Finally, we connect NeRA's capabilities with the Kolmogorov–Arnold Representation Theorem to reinforce its universal approximation ability.

13.1. Recap: LoRA and NeRA Adapter

LoRA - A Linear, Low-Rank Adaptation: LoRA modifies a model by adding a low-rank update ΔW to an existing frozen weight matrix W_0. Mathematically, this is defined as:

$$W = W_0 + \Delta W, \quad \Delta W = W_{\text{up}}\, W_{\text{down}} \tag{6}$$

where:

- $W_{\text{down}} \in \mathbb{R}^{d\times r}$ and $W_{\text{up}} \in \mathbb{R}^{r\times d}$
- $r \ll d$ ensures the update remains low-rank

Given an input $x \in \mathbb{R}^d$, the transformed output is:

$$\text{LoRA-Output} = W_0\, x + W_{\text{up}}\,(W_{\text{down}}\, x) \tag{7}$$

This transformation remains purely linear and is restricted to rank r, meaning LoRA cannot capture nonlinear interactions between features.

NeRA Adapter - A More Expressive Nonlinear Adapter: NeRA adapter builds on LoRA but introduces basis function expansions that allow for nonlinear transformations. The key difference is that NeRA adapter consists of:

- A down-projection, which maps input features into a transformed space using nonlinear functions and basis expansions.
- An up-projection, which reconstructs an adapted version of the input.

Mathematically, this is defined as:

$$H = \phi_{\text{down}}(X) \;=\; W_b^{\text{down}}\, g(X) \;+\; W_s^{\text{down}} \sum_{i=1}^{n} c_i\, \gamma_i(X), \tag{8}$$

$$X'_{\text{adapter}} = \phi_{\text{up}}(H) \;=\; W_b^{\text{up}}\, g(H) \;+\; W_s^{\text{up}} \sum_{j=1}^{m} d_j\, \gamma_j(H), \tag{9}$$

where:

- $g(\cdot)$ is a nonlinear activation function (e.g., ReLU, GeLU),
- $\gamma_i(\cdot)$ are basis functions (e.g., Taylor polynomials, B-splines, Fourier expansions),
- W_s^{down} and W_s^{up} control the contributions of the basis expansions.

Since NeRA adapter explicitly models nonlinear transformations, it can capture complex relationships between features, unlike LoRA.

13.2. Justification via Taylor Series Expansion

A key property of NeRA adapter is its ability to approximate arbitrary smooth functions, which LoRA cannot do. We prove this using Taylor series expansions.

Step 1: Approximating Functions with Taylor Series For any sufficiently smooth function $f : \mathbb{R}^d \to \mathbb{R}^d$ a Taylor series expansion around $x = 0$ gives:

$$f_k(x) = \underbrace{f_k(0)}_{\text{constant term}} + \sum_{\alpha} \frac{1}{\alpha!} \left(\frac{\partial^{|\alpha|} f_k}{\partial x^\alpha}\Big|_{x=0} \right) x^\alpha + (\text{higher-order remainder})$$

where,

- $\alpha = (\alpha_1, \ldots, \alpha_d)$ is a multi-index
- $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$ are monomial terms.

Truncating this expansion at order L provides a polynomial approximation:

$$f(x) \approx P_L(x) = \sum_{i=1}^{N_L} c_i\, \gamma_i(x), \tag{10}$$

where, $\gamma_i(x)$ are monomials up to degree L. Since NeRA adapter incorporates these basis functions explicitly, it can approximate any smooth function arbitrarily well—something LoRA cannot achieve with its strictly linear transformations.

Step 2: How NeRA adapter Uses Taylor Expansions for Adaptation

In a NeRA adapter, let the down-projection be:

$$H \;=\; W_b^{\text{down}}\, g(x) \;+\; W_s^{\text{down}} \sum_{i=1}^{N_L} c_i\, \gamma_i(x), \tag{11}$$

where,

- $\{\gamma_i\}$ are polynomial (Taylor) monomials up to degree L.

Then let the up-projection be:

$$X'_{\text{adapter}} \;=\; W_b^{\text{up}}\, g(H) \;+\; W_s^{\text{up}} \sum_{j=1}^{m} d_j\, \gamma_j(H). \tag{12}$$

Now X'_{adapter} is (in principle) a polynomial composition in x:

$$X'_{\text{adapter}} \;=\; W_b^{\text{up}}\, g\Big(W_b^{\text{down}} g(x) + \ldots\Big) + W_s^{\text{up}} \sum_{j=1}^{m} d_j\, \gamma_j\Big(H\Big). \tag{13}$$

Expanding $\gamma_j\big(H\big)$ yields a polynomial in the entries of H, each of which is itself a polynomial (plus nonlinearity g) in x. Provided g is a smooth or well-behaved activation, X'_{adapter} *can approximate a wide class of continuous functions* $\mathbb{R}^d \to \mathbb{R}^d$, surpassing strictly linear transformations.

Step 3: Universality Argument (Local Approximation)

We now formalize the fact that NeRA adapters can approximate any smooth function locally, while LoRA is limited to low-rank transformations.

Theorem (Polynomial Approximation): Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be analytic in a neighborhood of 0. Then for any $\epsilon > 0$, there exists a polynomial $P_L(x)$ of finite degree L such that,

$$\| f(x) - P_L(x) \| \ < \ \epsilon \tag{14}$$

for all x in some sufficiently small region.

This means that by choosing sufficiently many polynomial terms, NeRA adapter's basis expansions can approximate any smooth function arbitrarily well within a local region. LoRA, however, is constrained to:

$$f_{\text{LoRA}}(x) = W_{\text{up}}\,(W_{\text{down}}\,x) + \text{constant terms}, \tag{15}$$

which is a $rank - r$ linear map in x and cannot capture higher-order (nonlinear) dependencies. This formally proves that NeRA adapter is strictly more expressive than LoRA in local function approximation.

Step 4: Kolmogorov–Arnold Representation Theorem Perspective

While the Taylor series argument above shows *local* universal approximation, the classical Kolmogorov–Arnold Representation Theorem states that *any multivariate continuous function* on a *compact domain* can be written as a sum and composition of *univariate* continuous functions. Formally, in one version:

Kolmogorov–Arnold Representation: For any continuous function

$$F : [0,1]^d \ \to \ \mathbb{R}, \tag{16}$$

there exist *univariate* continuous functions ϕ_q, ψ_{qk} such that

$$F(x_1, \ldots, x_d) = \sum_{q=0}^{2d} \phi_q\Big(\sum_{k=1}^{d} \psi_{qk}(x_k)\Big). \tag{17}$$

This theorem implies that if a network architecture can realize *generic compositions of univariate basis expansions*, it can approximate any continuous multivariate function on a cube $[0,1]^d$. NeRA-style adapters, which rely on basis expansions and compositions (ϕ_{down}, ϕ_{up}), inherently align with this representation principle. Thus, by Kolmogorov-Arnold, NeRA adapter can approximate *any continuous* function, whereas LoRA's linear approach cannot achieve such universality in higher dimensions.

13.3. Why NeRA adapter Is Strictly More Expressive than LoRA

LoRA as a Special (Linear) Case of NeRA adapter: If we *disable* the basis expansions and nonlinearities in NeRA adapter:

$$W_s^{\text{down}} = 0, \quad W_s^{\text{up}} = 0, \quad g(x) = x, \tag{18}$$

then

$$\phi_{\text{down}}(x) = W_b^{\text{down}}\,x, \quad \phi_{\text{up}}(H) = W_b^{\text{up}}\,H, \tag{19}$$

giving

$$X'_{\text{adapter}} = W_b^{\text{up}}\,\big(W_b^{\text{down}} x\big), \tag{20}$$

which is precisely a rank-n linear map akin to LoRA. Thus, *LoRA is embedded in NeRA adapter* as a degenerate (linear-only) configuration.

Nonlinear and Higher-Order Terms in NeRA adapter: In general, NeRA adapter includes expansions:

$$\sum_i c_i \gamma_i(x), \quad \gamma_i(x) = \text{(e.g. polynomial monomial)}. \tag{21}$$

Hence, X'_{adapter} can contain *terms of the form* $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$ or other advanced expansions (Fourier, spline, etc.). *No purely linear LoRA block can replicate such higher-order interactions* without artificially stacking multiple linear LoRA blocks across many layers.

13.4. Proof Summary

- **Taylor-series-based argument**: By leveraging polynomial expansions up to order L, NeRA can approximate local behavior of smooth functions to arbitrary precision. LoRA, being linear and low-rank, lacks this capacity.
- **Kolmogorov–Arnold-based argument**: NeRA's composition of univariate basis expansions aligns with the Kolmogorov-Arnold Theorem, thus enabling universal approximation for continuous functions on compact domains in $\mathbb{R}^d$.
- **LoRA is a special case**: Setting all basis expansions to zero and removing nonlinearities recovers a linear rank-r form, demonstrating that NeRA *strictly generalizes* LoRA.

Therefore, **NeRA can realize richer, higher-order function approximations**, surpassing the purely linear rank-r approach of LoRA. Consequently, **NeRA adapter works better than LoRA** in scenarios demanding expressive, nonlinear transformations while retaining the efficient, adapter-based paradigm.

In practice, a NeRA may incorporate polynomial or spline expansions of a modest order, often capturing crucial nonlinearities with fewer parameters than a naive full-rank update. Empirical results generally confirm that NeRA-like architectures can adapt large models more flexibly than LoRA alone, especially under distribution shifts or tasks requiring higher-order feature interactions.

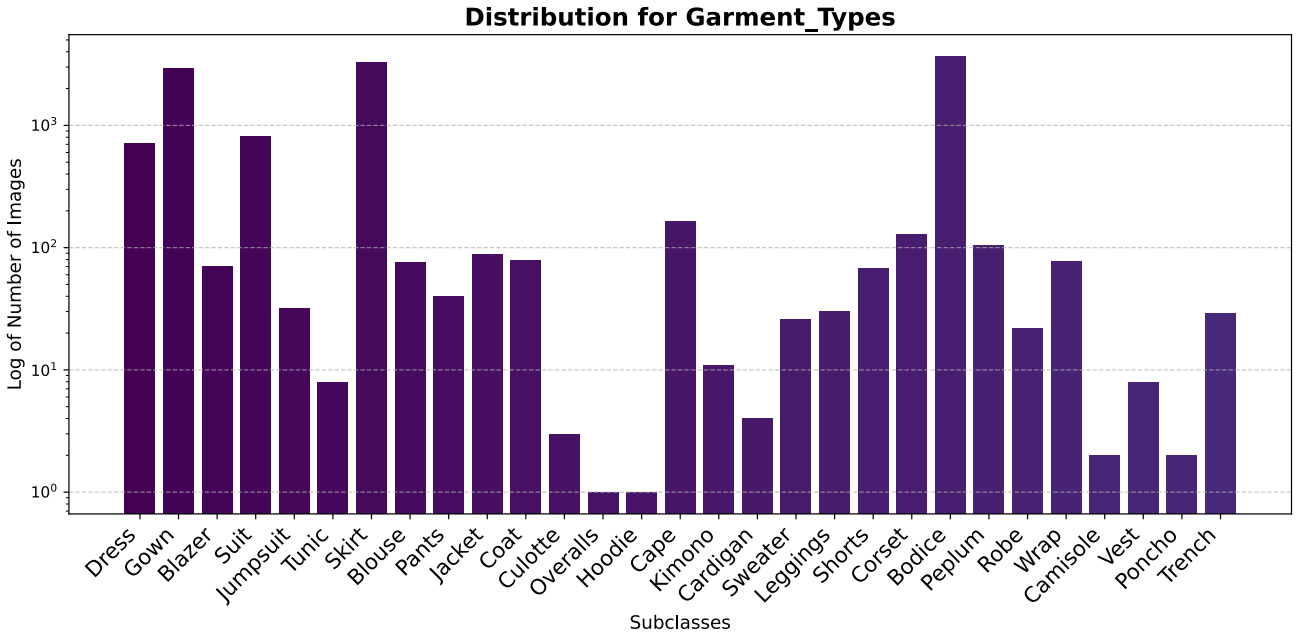

Figure 10. Image Distribution for 29 Garment Types available in FLORA

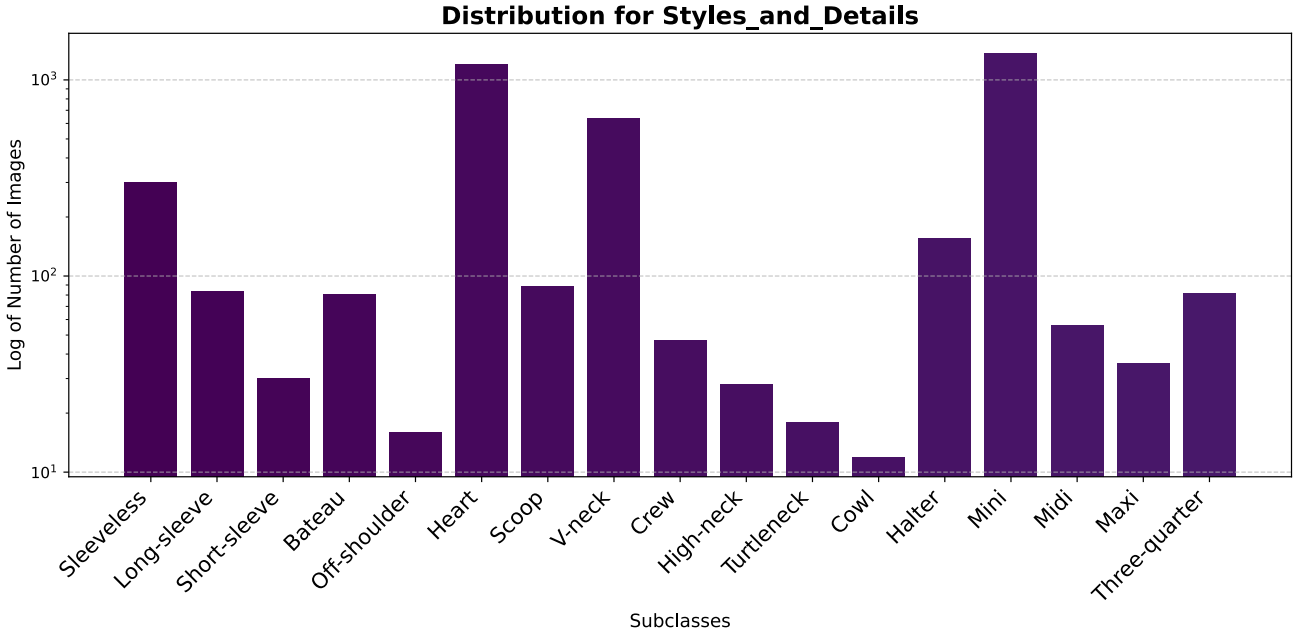

Figure 11. Image Distribution for 17 Styles

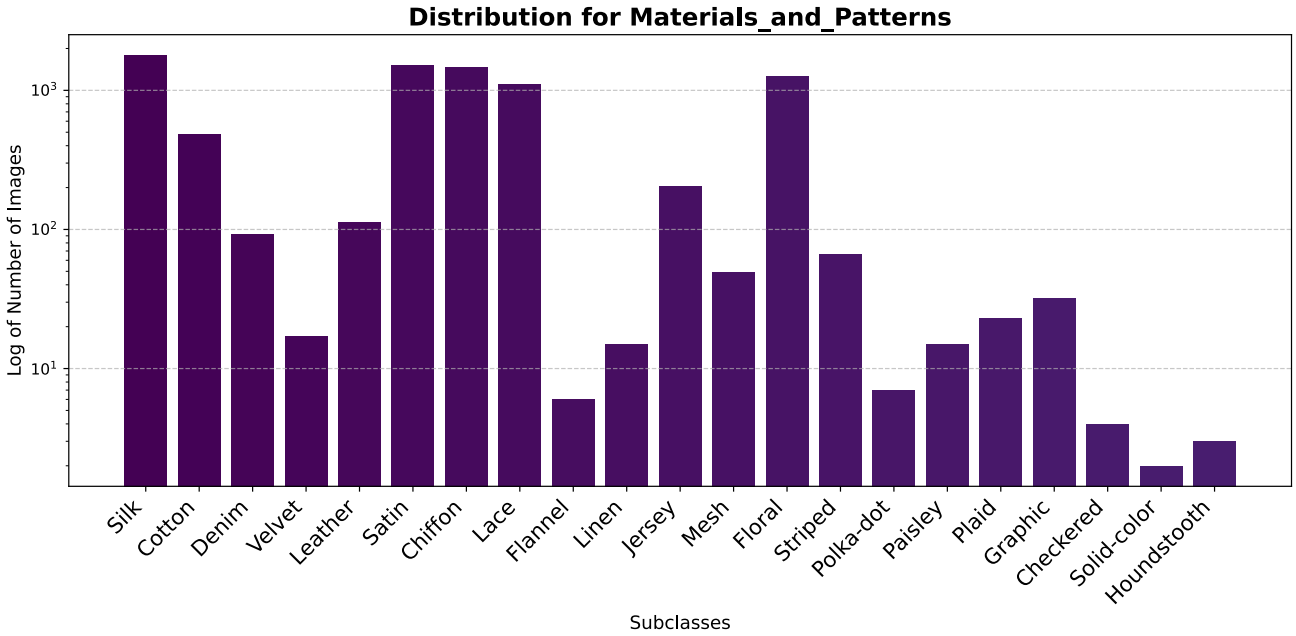

Figure 12. Image Distribution for 21 Materials & Patterns

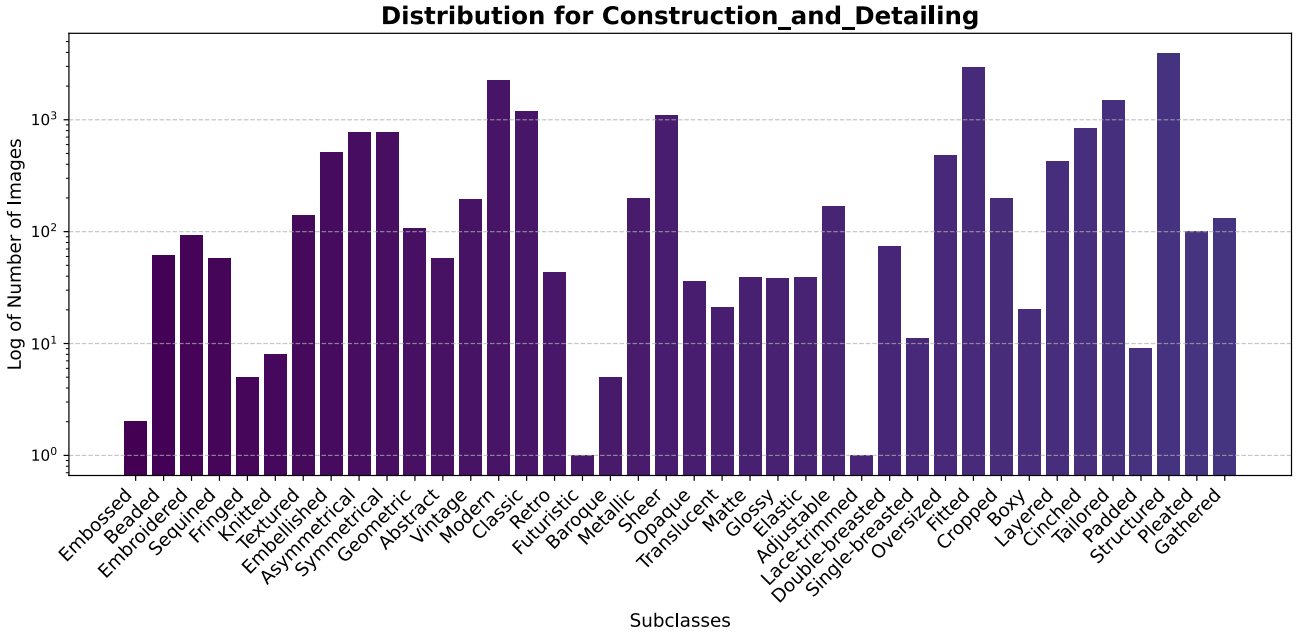

Figure 13. Image Distribution for 40 Construction styles

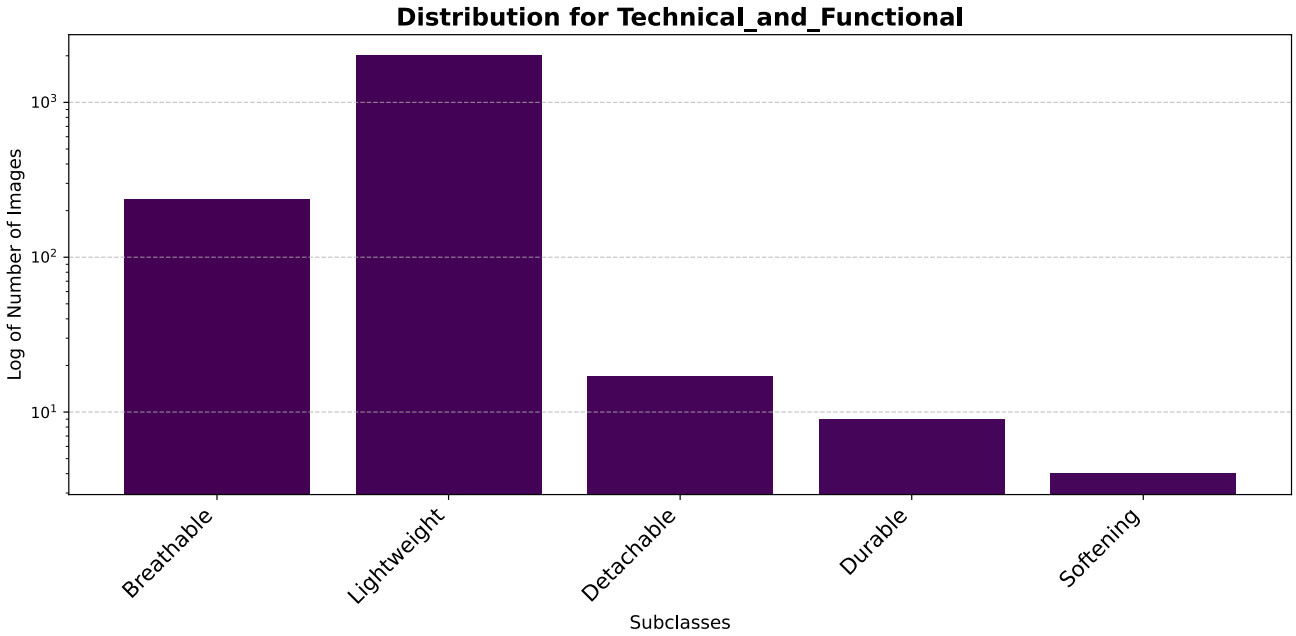

Figure 14. Image Distribution for Technical styles

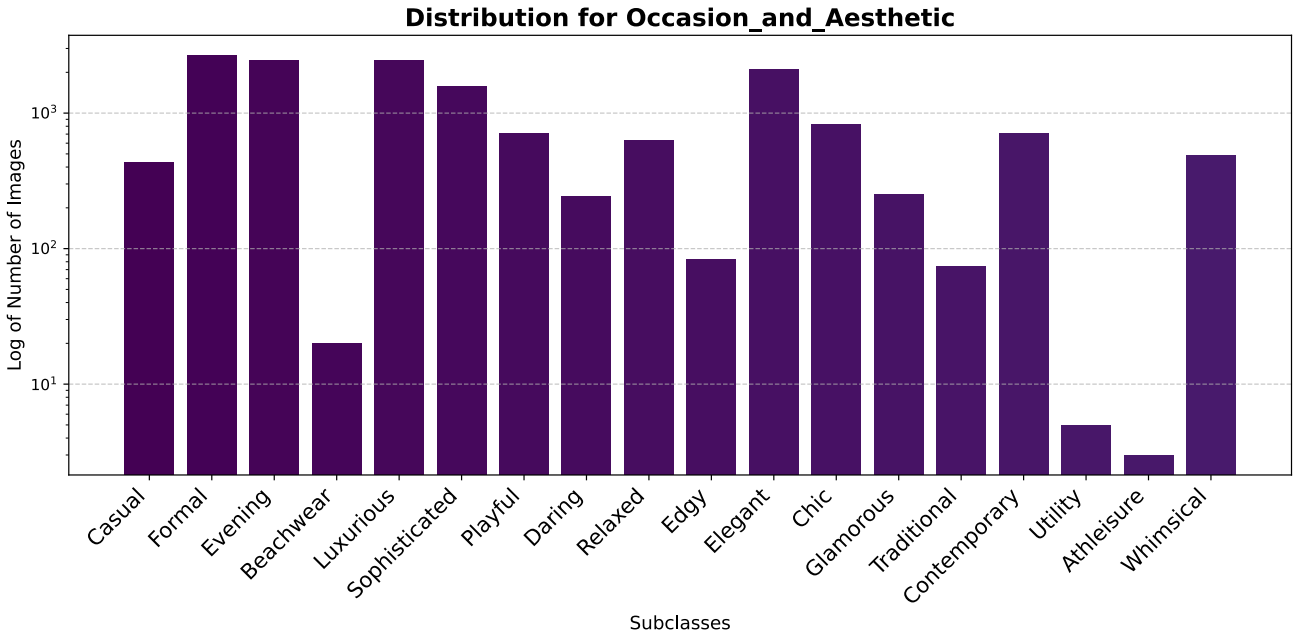

Figure 15. Image Distribution for 18 Occasion & Aesthetic styles

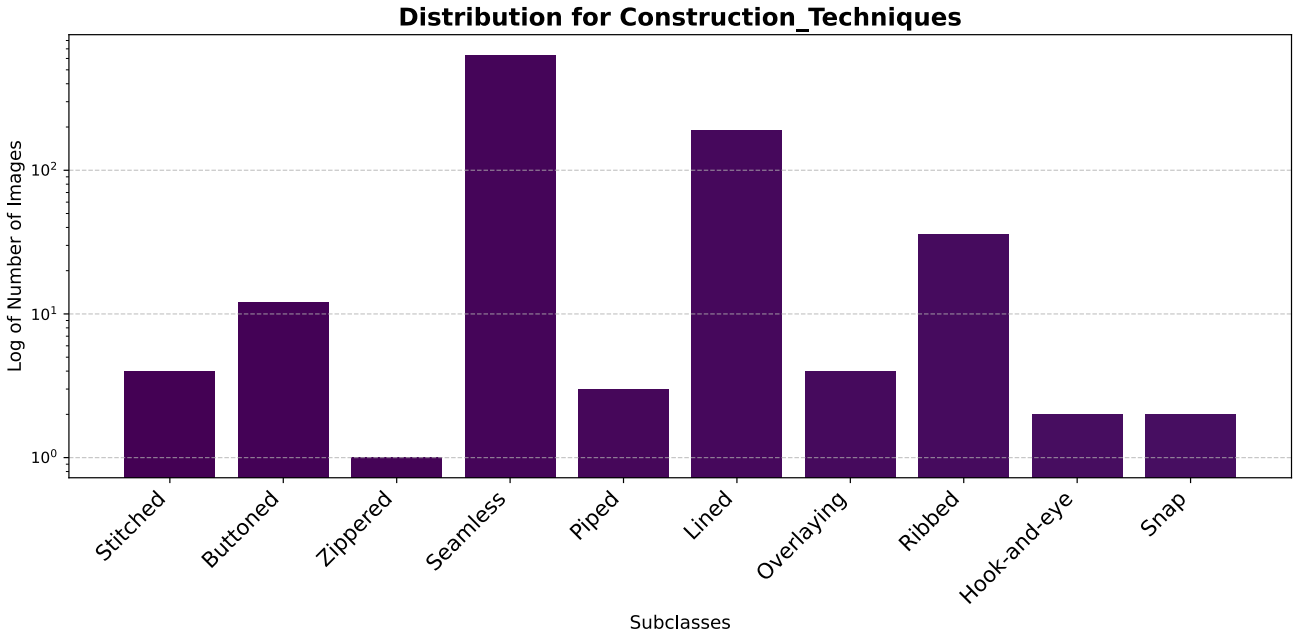

Figure 16. Image Distribution for 10 Construction techniques

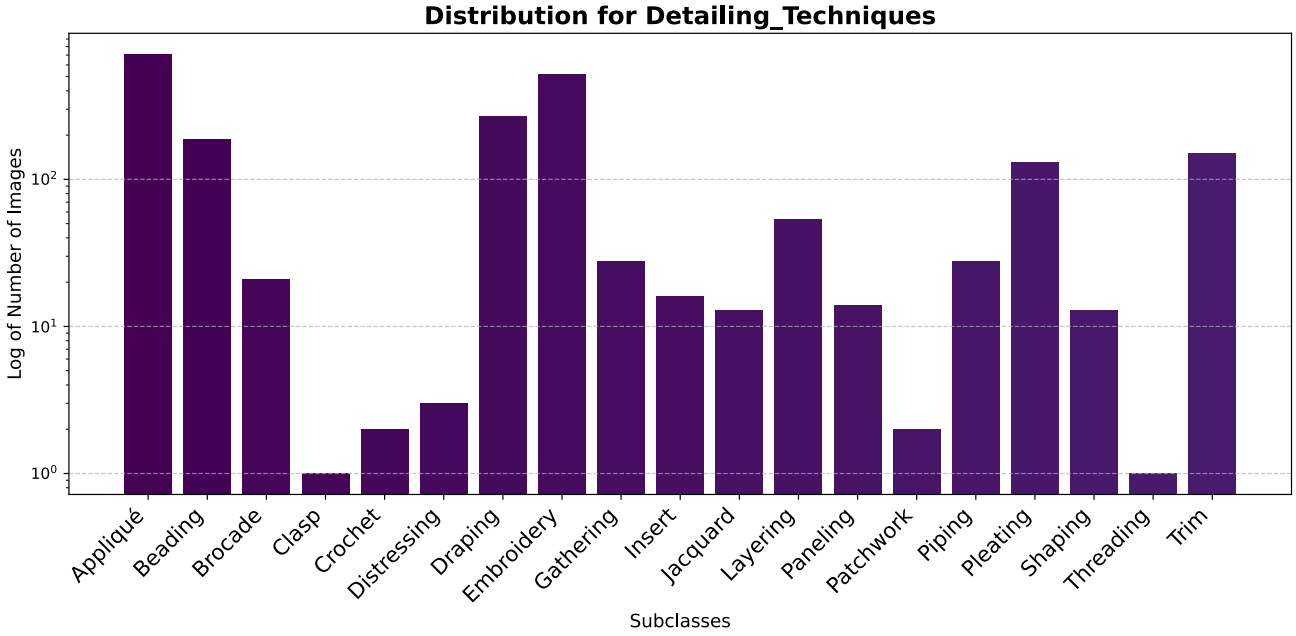

Figure 17. Image Distribution for 19 Detailing techniques

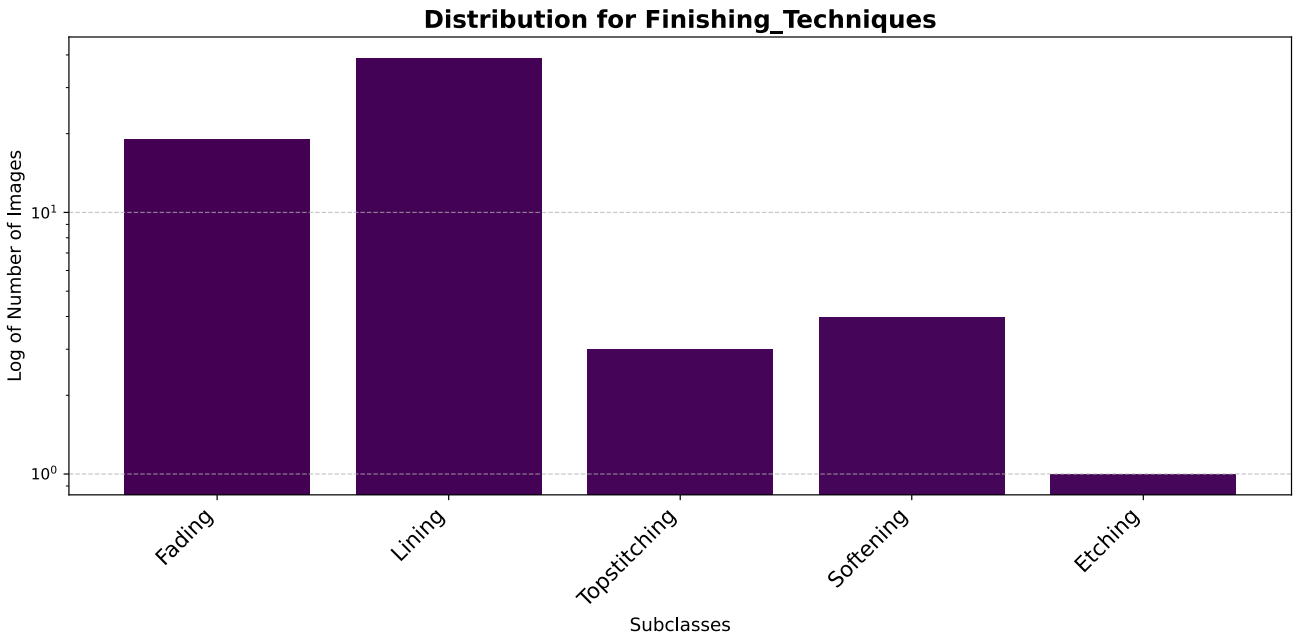

Figure 18. Image Distribution for Finishing techniques