**Model train types**

**1. Full Training (Base Model Training)**

**What it is:**

* Training **from scratch** using billions of (image, text) pairs.
* Builds the entire model, including tokenizer, CLIP encoder, VAE, and U-Net.

**How it works:**

* Dataset: LAION-2B+, or custom image-caption sets
* Phases:
  + Train autoencoder (VAE)
  + Train text encoder (e.g., CLIP)
  + Train diffusion model (U-Net) on latent images
* Requires 100,000s of GPU hours

**Pros:**

* Complete control over the model
* Can tailor to specific languages, cultures, or art styles
* Better performance with a well-curated dataset

**Cons:**

* Extremely expensive (can cost millions)
* Requires huge computing power and storage
* High risk of poor quality if data isn’t curated well

**2. Fine-Tuning (Full Model Fine-Tune)**

**What it is:**

* Further training of a **base model** on a **new, smaller dataset** (thousands to millions of images).
* Adjusts all model weights to adapt to a new domain or style.

**How it works:**

* Uses same diffusion training process
* Often done with lower learning rates to preserve existing knowledge
* Example: Deliberate, RealisticVision models

**Pros:**

* Creates a full-featured model for a new style
* Better generalization than personal models
* Can be shared on platforms like CivitAI

**Cons:**

* Still requires strong GPU (e.g., 24GB VRAM+)
* Can overfit or "forget" base model capabilities
* Large file size (4–7GB per model)

**3. LoRA (Low-Rank Adaptation)**

**What it is:**

* A small, modular patch trained to **modify specific layers** of a base model.
* Injected into U-Net (and optionally text encoder) at runtime.

**How it works:**

* Trains only a few weights using rank decomposition (lightweight matrix ops)
* Uses as little as 10–1000 images
* Saves as a separate file (2–200MB)

**Pros:**

* Very small size
* Efficient to train (runs on 8–16GB GPUs)
* You can mix, stack, and swap LoRAs freely

**Cons:**

* Needs a compatible base model
* Some LoRAs only work well in certain prompt ranges
* May require prompt engineering to activate properly

**4. Textual Inversion**

**What it is:**

* Learns a **custom embedding** (a single word/token) that represents a style, person, or concept.
* **How it works:**
* Add a fake token like <sksgirl>
* Train it to associate with a set of images
* Result is a small .pt or .bin file (~50KB)

**Pros:**

* Tiny size, fast to train (30 mins–2 hours)
* Great for simple concepts or styles
* Easy to share and use

**Cons:**

* Doesn’t modify the model itself
* Hard to represent complex or dynamic styles
* Needs very consistent training images

**5. DreamBooth**

**What it is:**

* Personalization method to teach the model a **specific subject** (e.g., your face, a pet, a product) using just 3–10 images.

**How it works:**

* Trains on images with a rare token like "photo of sks dog"
* Locks identity by regularizing against class images ("dog") to avoid overfitting

**Pros:**

* Very accurate personalization
* Great for generating consistent subjects
* Can include context (pose, lighting)

**Cons:**

* Requires more GPU than LoRA (12–24GB)
* Can overwrite other parts of the model (overfitting)
* Final models are large (2–7GB unless you export as LoRA)

**6. ControlNet**

**What it is:**

* An **auxiliary network** trained to condition generation on **external input maps** like pose, depth, canny edges, scribbles, etc.

**How it works:**

* Trains with paired image + control maps
* Adds extra layers parallel to U-Net blocks
* During inference, control map guides image structure while text guides style

**Pros:**

* Gives users **precise control** over generation
* Works for pose transfer, consistent scenes, etc.
* Stacks with LoRAs or styles

**Cons:**

* Requires custom training data (pose, depth maps, etc.)
* Heavier to run at inference time
* Doesn’t add style or concepts — just structure

**7. Hypernetwork**

Note: This is now considered outdated — replaced by LoRA in most workflows.

**What it is:**

* A side-network that modifies hidden layer activations during inference.

**How it works:**

* Trained with a small dataset (100–1000 images)
* Affects activations without modifying model weights

**Pros:**

* Smaller than fine-tunes
* Adds unique style or behavior
* Works with A1111 easily

**Cons:**

* Not compatible with ComfyUI or SDXL
* Often unstable or subtle in effect
* Mostly replaced by LoRA due to better control

**Summary Table**

| **Type** | **Size** | **Train Time** | **Purpose** | **Pros** | **Cons** |
| --- | --- | --- | --- | --- | --- |
| Full Model | Huge (7GB+) | Weeks | Base model for all tasks | Total control, high quality | Expensive, complex, GPU-heavy |
| Fine-Tune | Large | Days | Domain-specific style/model | Powerful, custom styles | Overfitting risk, big files |
| LoRA | Small (5–200MB) | Hours | Add or modify concepts | Lightweight, stackable | Prompt-sensitive |
| Textual Inversion | Tiny (~50KB) | 1–2 hours | Teaches a token | Fast and easy | Only adds tokens, low complexity |
| DreamBooth | Medium | 2–6 hours | Personalization | High accuracy for subjects | Risk of overfitting, large export size |
| ControlNet | Medium | Days | Pose, depth, layout control | Structural control | Needs paired data, compute-heavy |
| Hypernetwork | Medium | 1–2 hours | Stylization (legacy) | Small and flexible | Outdated, weaker control |

**List of Stable Diffusion Training Techniques**

**1. DreamBooth**

* **Goal:** Teach the model a specific person, object, or concept.
* **Input:** 4–30 images of a subject + class images.
* **Output:** Personalized model that can generate images using a trigger word.
* **Use Case:** Custom people, pets, objects, brands.
* **Tools:** Google Colab (TheLastBen, ShivamShrirao), Kohya-ss

**2. LoRA (Low-Rank Adaptation)**

* **Goal:** Fine-tune a model without changing the entire model weights.
* **Input:** 10–100+ images.
* **Output:** A lightweight.safetensors file that plugs into base model.
* **Use Case:** Styles, faces, cinematic tones, clothing, objects.
* **Tools:** Kohya-ss GUI, ComfyUI

**3. Textual Inversion**

* **Goal:** Teach the model a *word* (embedding) that represents a new concept.
* **Input:** 3–20 images + a unique placeholder token (e.g., <cinematic\_guy>).
* **Output:** A .pt file (token embedding).
* **Use Case:** Simple styles, textures, poses, aesthetics.
* **Tools:** Automatic1111, Kohya GUI, Diffusers

**4. Fine-Tuning (Full Model)**

* **Goal:** Retrain both the UNet and text encoder for maximum model adaptation.
* **Input:** 1000s of images + GPU + time.
* **Output:** New full model.
* **Use Case:** Creating a new model for a specific domain or dataset.
* **Tools:** Huggingface Diffusers, Kohya-ss, DreamBooth + extra scripts

**5. Full Model Training (From Scratch)**

* **Goal:** Train a Stable Diffusion model from random weights.
* **Input:** Huge dataset (50k–500k images) + captions.
* **Output:** A new base model.
* **Use Case:** Research, commercial models (e.g., SDXL, RealisticVision).
* **Tools:** Custom PyTorch training scripts, Diffusers

**6. Hypernetwork Training (older method)**

* **Goal:** A modular training method to influence output without changing base model.
* **Status:** Largely replaced by LoRA.
* **Use Case:** Artistic or abstract style influence.

**7. ControlNet Training**

* **Goal:** Train models to respond to conditions like pose, depth, edge, etc.
* **Input:** Paired images and condition maps (e.g., pose images + normal photo).
* **Output:** A new ControlNet model file.
* **Use Case:** Cinematic scenes with exact poses, sketches, depth control.
* **Tools:** ControlNet training scripts (OpenPose, depth, canny, etc.)

**8. Adapter Training / T2I Adapters**

* **Goal:** Lightweight condition-based model tuning (similar to ControlNet, but smaller).
* **Use Case:** Depth, sketch, edge → image generation.
* **Status:** Experimental but effective for stylization and control.

**9. Style LoRA / Layer-wise Fine-Tuning**

* **Goal:** Train on *style* only using few images (e.g., "Christopher Nolan" style).
* **Input:** 20–100 images.
* **Use Case:** Cinematic look, lens effects, lighting tone.