# The Alchemist's Handbook: A Comprehensive Guide to Advanced Block-Level Model Merging in ComfyUI

## Section 1: The Geometry of Merging: From Loss Landscapes to Task Vectors

The practice of model merging, which combines multiple specialized models into a single, more capable entity, appears almost alchemical. However, its effectiveness is not magic but is grounded in the fundamental geometric properties of neural network training and parameter spaces. Understanding these principles—the nature of the loss landscape, the concept of mode connectivity, and the abstraction of task vectors—is essential for moving beyond simple trial-and-error and toward a deliberate, scientific approach to creating custom diffusion models.

### 1.1. The Loss Landscape and the "Basin of Attraction"

The training of any neural network is an optimization problem: finding a set of parameters (weights) that minimizes a high-dimensional, non-convex loss function.1 The topography of this function is referred to as the "loss landscape." While theoretically fraught with local minima that could trap the optimization process, practical experience shows that simple gradient-based methods are surprisingly effective at finding "good" solutions—parameter configurations that yield low loss and generalize well.1

A key discovery that underpins model merging is the phenomenon of **Linear Mode Connectivity (LMC)**. LMC describes the observation that it is often possible to find a path between two independently trained solutions (minima) in the loss landscape along which the loss remains consistently low.1 When two models are fine-tuned from the same pre-trained base, they are highly likely to converge to minima that reside within the same broad "basin of attraction" in the loss landscape. This shared lineage means their parameter spaces are not arbitrarily different; instead, they are connected by these low-loss pathways, making linear interpolation between their weights—the fundamental operation of merging—a viable strategy.4

The causal relationship here is critical: the shared pre-trained initialization is the *cause* of the models residing in a shared loss basin, and this shared basin is the *effect* that enables successful merging. This principle is so fundamental that it can be exploited for adversarial purposes. Research into disrupting model merging has demonstrated that intentionally pushing a model's parameters out of this shared basin, for instance through specific parameter rearrangements, effectively breaks the LMC and causes subsequent merge attempts to fail catastrophically.5 This confirms that a shared architectural and pre-training origin is the most critical prerequisite for any successful merge. Models with different fundamental architectures (e.g., SD1.5 vs. SDXL) or entirely different training histories exist in disconnected, distant basins, and attempting to merge them via simple arithmetic is futile.

### 1.2. Task Vectors: The Atomic Unit of Model Arithmetic

While the concept of a shared loss basin explains *why* merging is possible, the theory of **Task Arithmetic** provides a powerful framework for *how* to merge models in a more controlled, semantic way.9 This framework introduces the "task vector" (

τ), which is defined as the element-wise difference between the weights of a fine-tuned model (θfinetuned​) and the weights of its pre-trained base model (θbase​).4

The equation for a task vector is:

τ=θfinetuned​−θbase​

This vector is theorized to isolate and represent the specific knowledge, skill, or style learned during the fine-tuning process. For example, if a base model is fine-tuned on a dataset of anime-style images, the resulting task vector, τanime​, conceptually represents the "anime-ness" that was added to the model.

This abstraction reframes model merging from a brute-force mixing of millions of parameters into a more elegant form of vector arithmetic. Instead of simply averaging two fine-tuned models, Task Arithmetic proposes adding the desired skills back to the original base model. The merged model's parameters (θmerged​) are constructed by adding a scaled sum of the task vectors to the base model's parameters:

θmerged​=θbase​+λi∑​τi​

where λ is a scaling coefficient that controls the intensity of the added skills.12

This approach allows for a more intuitive and powerful manipulation of model capabilities. One can add skills (e.g., base + τ\_photorealism + τ\_portraits), subtract them to perform a type of "unlearning," or even perform analogies. This conceptual framework, where skills are treated as manipulable vectors, forms the foundation for nearly all advanced merging algorithms, which seek to combine these task vectors in increasingly sophisticated ways to mitigate interference and maximize synergy.

## Section 2: A Taxonomy of Merging Algorithms

The field of model merging has evolved rapidly from simple averaging to complex algorithms designed to resolve the inherent conflicts that arise when combining distinct parameter sets. These algorithms can be categorized based on their core mechanism, each offering a different trade-off between simplicity, computational cost, and the ability to mitigate interference.

### 2.1. Interpolation Methods: The Foundation

These methods represent the most direct application of the linear mode connectivity principle, treating the weights of two models as points in a high-dimensional space and finding a point on the path between them.

* **Simple Weighted Averaging:** This is the most fundamental merging technique, calculating the new model's weights (θmerged​) as a linear combination of two parent models, Model A (θA​) and Model B (θB​), controlled by a ratio α:  
  θmerged​=(1−α)⋅θA​+α⋅θB​  
  While simple and computationally inexpensive, this method is highly susceptible to parameter interference. When two models have learned conflicting ways to use the same parameters, their simple average can result in a functionally degraded or incoherent state, often leading to performance loss.3
* **Spherical Linear Interpolation (SLERP):** SLERP offers a more geometrically sound approach to interpolation. Instead of a straight line, it interpolates between the two model vectors along the shortest arc on the surface of a high-dimensional sphere.14 This has two main advantages: it maintains a constant rate of change during interpolation, and it prevents the reduction in vector magnitude (i.e., the scale of the weights) that can occur with linear interpolation. By preserving the directional information of the weights, which is often more meaningful than their magnitude, SLERP can produce more stable and coherent merges. Its primary limitation is that it is designed to merge only two models at a time.14

### 2.2. Task Vector Arithmetic and Its Derivatives

This class of algorithms builds upon the concept of task vectors to more intelligently combine model capabilities while actively trying to mitigate the negative effects of parameter interference.

* **Task Arithmetic:** As defined previously, this method serves as the baseline for multi-model (>2) merging. It operates by isolating the "skills" as task vectors and adding them back to a common base model.9 Its primary weakness is that it sums all task vectors indiscriminately, which can lead to significant interference when skills are conflicting.
* **TIES-Merging (Trim, Elect Sign, Merge):** TIES-Merging is a powerful algorithm designed specifically to resolve the interference that plagues simpler methods.12 It operates in a three-step process on the task vectors 12:
  1. **Trim:** It first identifies and retains only the most significant parameter changes within each task vector. By keeping only the top-k% of weights by magnitude (e.g., top 20%) and zeroing out the rest, it creates a sparse representation of the task, filtering out redundant or minor adjustments that are likely to cause noise.
  2. **Elect Sign:** It then resolves sign conflicts. If one task vector pushes a parameter in a positive direction and another pushes it in a negative direction, TIES elects a single "winning" sign based on which direction has the greater total magnitude across all task vectors.
  3. **Disjoint Merge:** Finally, it computes the new merged task vector by averaging only the parameter values from the trimmed task vectors that align with the elected sign for that parameter. Conflicting values are discarded, not averaged. This ensures that only consistent, reinforcing changes are integrated into the final model.
* **DARE (Drop And REscale):** DARE provides an alternative strategy for mitigating interference by introducing sparsity. It randomly "drops" (zeros out) a specified percentage of the parameters in the delta between the fine-tuned model and the base model. The remaining non-zero parameters are then rescaled to preserve the overall magnitude of the original delta.20 By creating a sparse task vector, DARE reduces the number of parameters that can conflict during the merge. It is often used as a pre-processing step for TIES-Merging, a combination known as  
  DARE-TIES.

### 2.3. Advanced and Data-Driven Methods

These methods require additional computational steps or access to data to achieve a more principled merge, often outperforming simpler techniques at the cost of increased complexity.

* **Fisher Merging:** This technique draws from information theory, weighting parameters based on their importance as estimated by the **Fisher Information Matrix**. The Fisher matrix quantifies how much information a parameter holds about the data distribution. By weighting the average of the models' parameters by their Fisher information, this method gives more influence to parameters that are considered more critical for a model's performance, aiming to maximize the joint likelihood of the merged model's parameters.23
* **RegMean (Regression Mean):** RegMean approaches merging as a regression problem. For each linear layer in the models, it seeks to find a new weight matrix that minimizes the L2 distance between the output activations of the merged model and the output activations of the parent models. This can be solved in a closed form using least-squares regression, effectively finding a new layer that best approximates the function of all parent layers for a given set of input data.25

### Table 1: Comparative Analysis of Merging Algorithms

| Algorithm Name | Core Mechanism | Key Strengths | Key Weaknesses/Challenges | Ideal Use Case |
| --- | --- | --- | --- | --- |
| **Simple Weighted Avg.** | Linear interpolation of all parameters. | Simple, fast, no dependencies. | Highly prone to parameter interference, often degrades performance. | Quick test for model compatibility; merging very similar models. |
| **SLERP** | Spherical interpolation of parameters. | Preserves weight vector magnitude and geometry; smoother interpolation. | Limited to merging only two models at a time. | High-quality merge of two models, especially for preserving distinct styles. |
| **Task Arithmetic** | Adds scaled task vectors (θ\_ft - θ\_base) to a base model. | Intuitive semantic manipulation; base for more advanced methods. | Does not resolve conflicts; performance degrades as more vectors are added. | Merging multiple models where interference is expected to be low. |
| **TIES-Merging** | Trims sparse task vectors, resolves sign conflicts, averages aligned weights. | Excellent at mitigating interference; allows for merging many or dissimilar models. | Can be overly aggressive in pruning, losing subtle details. Requires density tuning. | Merging 3+ models with diverse skills; combining very different styles. |
| **DARE** | Randomly drops and rescales delta parameters before merging. | Introduces sparsity to reduce conflicts; computationally efficient. | Stochastic nature means results can vary with seed; requires tuning drop rate. | Pre-processing step for TIES or Task Arithmetic to improve merge stability. |
| **Fisher Merging** | Weighted average based on the Fisher Information Matrix. | Principled, data-driven weighting of parameters by importance. | Computationally expensive; requires calculating the Fisher matrix. | Advanced merging where maximizing joint probability is the goal. |
| **RegMean** | Solves a least-squares problem to match activations. | Optimizes for functional equivalence at the layer level. | Requires access to data to compute activations; complex implementation. | Scenarios where preserving the functional output of each layer is critical. |

## Section 3: The Anatomy of a Diffusion Model for Merging

To perform advanced, block-level merging, one must first understand the diffusion model not as a monolithic entity, but as a system of distinct, interacting components. Each component—the UNET, the CLIP text encoder, and the VAE—plays a specific role in the generative process. By dissecting the model's anatomy, we can map its architectural structure to a functional hierarchy, enabling us to target specific generative attributes with surgical precision.

### 3.1. The UNET: Architect of Composition and Detail

The UNET is the heart of the diffusion model, responsible for the iterative denoising process that transforms random noise into a structured image in the latent space. Its U-shaped architecture, comprising a contracting path, a bottleneck, and an expanding path, is not arbitrary; it directly facilitates a hierarchical generation process from coarse composition to fine detail.31 The standard Stable Diffusion UNET is composed of 25 blocks:

* **Input Blocks (IN00-IN11):** This is the contracting path of the U-Net, consisting of 12 blocks. It progressively downsamples the latent image, extracting features at decreasing resolutions. The earliest blocks (e.g., IN00-IN03) operate on the highest-resolution latent representations and are therefore primarily responsible for establishing the global composition, the overall structure, character poses, and anatomy.
* **Middle Block (M00):** This single block is the bottleneck of the UNET, where the latent representation is at its most compressed. At this stage, the information is most abstract and semantic. The middle block is therefore critical for defining the core *concept* or high-level *style* of the subject matter.
* **Output Blocks (OUT00-OUT11):** This is the expanding path, also consisting of 12 blocks. It progressively upsamples the latent representation, re-introducing detail and refining the image. This process is guided by information passed from the corresponding input blocks via skip connections. The later output blocks (e.g., OUT08-OUT11) operate as the image approaches its final resolution and are thus responsible for rendering high-frequency information such as textures, material properties, intricate patterns, and lighting effects.

This architectural-to-functional mapping is the central principle that enables strategic block-level merging. To control the overall layout of an image, one must manipulate the weights of the early input blocks. To influence the fundamental style or identity of a subject, one targets the middle block and its surrounding layers. To graft the texture of one model onto the composition of another, one intervenes in the final output blocks.

### Table 2: UNET Block Functional Mapping

| Block Name(s) | Resolution (Relative) | Primary Function in Generation |
| --- | --- | --- |
| **IN00 - IN03** | High -> Medium | Global composition, anatomy, major shapes, camera angle, subject placement. |
| **IN04 - IN08** | Medium -> Low | Refinement of subject structure, introduction of medium-level style elements. |
| **IN09 - IN11** | Low -> Lowest | Transition to core semantic concepts, strong style influence. |
| **M00** | Lowest | Core concept/subject identity, highest-level style definition, bottleneck of information. |
| **OUT00 - OUT02** | Lowest -> Low | Initial reconstruction, propagation of core style and concept from the middle block. |
| **OUT03 - OUT07** | Low -> Medium | Re-introduction of structural details, blending of style with form. |
| **OUT08 - OUT11** | Medium -> High | Rendering of fine details, texture, material properties, lighting, noise/grain. |

### 3.2. The CLIP Text Encoder: The Semantic Guide

The CLIP text encoder acts as the interpreter, translating the user's text prompt into a set of conditioning vectors that guide the UNET's denoising process.4 This guidance is injected into the UNET via cross-attention layers, which are present in the UNET blocks.31

Different fine-tuned models develop distinct semantic biases in their CLIP encoders. A model fine-tuned on photorealistic portraits will have a CLIP that is highly sensitive to photographic terms ("85mm lens", "f/1.4", "cinematic lighting"), while a model fine-tuned on anime art will be more attuned to style descriptors ("moe," "kawaii," specific artist names).

Merging CLIP encoders is therefore a method for creating a new semantic interpreter with a blended vocabulary.36 A merged CLIP can understand and respond to prompts that mix concepts from both parent models, enabling more nuanced and creative control than either parent model could achieve alone. For example, merging a photorealistic CLIP with an anime CLIP could allow for the generation of a "photorealistic anime character," a concept that might be poorly interpreted by either parent model individually.

### 3.3. The VAE: The Lens of Perception

The Variational Autoencoder (VAE) is the component that translates images between the high-dimensional pixel space and the compressed latent space where the diffusion process occurs.38 While the UNET and CLIP build the image in the latent space, the VAE is responsible for the final translation back into a viewable image. As such, it has a profound impact on the final aesthetic qualities, particularly color, contrast, and fine textures.39

A common failure mode in model merging is the output appearing desaturated, washed out, or entirely gray. This is very often a VAE-related issue, caused by an incompatibility between the merged UNET's output and the VAE's decoding capabilities.41 Therefore, VAE merging (or replacement) is a critical tool for aesthetic control and problem correction. By blending VAEs, one can achieve a custom color profile, combining the vibrancy of one model with the softness of another. More pragmatically, if a merge produces color artifacts, one can often fix the issue entirely by discarding the merged VAE and simply using the VAE from one of the parent models or a known-good community VAE.

## Section 4: The Core Recipes: Block-Level UNET Merging

This section presents a series of specific, actionable recipes for block-level UNET merging. These recipes are designed to achieve distinct artistic outcomes by strategically weighting the influence of two parent models (Model A and Model B) across the different functional blocks of the UNET. The weights are provided with a precision of 0.01 and assume a simple weighted sum where the weights for Model A and Model B sum to 1.0. These recipes provide a strong starting point for experimentation and can be adapted for various merging algorithms.

### 4.1. Recipe 1: Style Transfer & Subject Preservation

**Objective:** To apply the distinct art style of Model B onto the composition and subject matter defined by Model A. This is one of the most common goals in model merging, for example, rendering a character model in the style of a landscape painting model.

**Rationale:** The merging strategy is to give strong dominance to Model A in the early input blocks, which control composition and anatomy. The influence of Model B is gradually introduced and becomes dominant in the middle and later blocks, which control the core style and fine-grained textural details.

### Table 3: Recipe for Style Transfer (Style B onto Subject A)

| UNET Block(s) | Function | Weight A | Weight B | Justification |
| --- | --- | --- | --- | --- |
| **IN00 - IN02** | Composition | 0.90 | 0.10 | Heavily favors Model A to lock in the overall scene structure and anatomy. |
| **IN03 - IN05** | Transition | 0.75 | 0.25 | Begins to introduce Model B's influence as the features become more abstract. |
| **IN06 - IN08** | Subject Core | 0.60 | 0.40 | Model A still leads to preserve the subject's identity, but style B starts to assert itself. |
| **IN09 - IN11** | Pre-Middle Style | 0.40 | 0.60 | The balance tips; Model B's style now dominates as we approach the semantic core. |
| **M00** | Style Core | 0.25 | 0.75 | Strong dominance by Model B at the bottleneck to ensure its style is the primary concept. |
| **OUT00 - OUT02** | Post-Middle Style | 0.30 | 0.70 | Reinforces Model B's style during the initial upsampling phase. |
| **OUT03 - OUT05** | Style Refinement | 0.40 | 0.60 | Continues to favor Model B while re-integrating some of Model A's structural information. |
| **OUT06 - OUT11** | Detail/Texture | 0.50 | 0.50 | A 50/50 split often creates a novel and interesting blend of both models' textural qualities. |

### 4.2. Recipe 2: Balanced Concept Blending

**Objective:** To create a harmonious and balanced fusion of two distinct concepts or aesthetics, such as merging a "fantasy armor" model (Model A) with a "sci-fi technology" model (Model B).

**Rationale:** This recipe employs a more balanced, "M-shaped" weighting curve. Unlike style transfer, where one model dominates, here both models are given significant influence throughout the network. The key is to achieve an equal partnership in the middle blocks, where the core concepts are defined, allowing for a true conceptual blend.

### Table 4: Recipe for Balanced Concept Blending

| UNET Block(s) | Function | Weight A | Weight B | Justification |
| --- | --- | --- | --- | --- |
| **IN00 - IN05** | Composition | 0.60 | 0.40 | A slight bias towards Model A to establish a primary compositional base. |
| **IN06 - M00 - OUT05** | Core Concepts | 0.50 | 0.50 | The crucial step: an equal 50/50 split across all middle blocks for a true fusion of concepts. |
| **OUT06 - OUT11** | Detail/Texture | 0.40 | 0.60 | A slight bias towards Model B to incorporate its specific textural details (e.g., glowing lights). |

### 4.3. Recipe 3: Detail & Texture Grafting

**Objective:** To use a specialized model (Model B) to add high-frequency details, such as realistic skin texture or intricate fabric patterns, to a more general-purpose model (Model A) that has good composition but lacks fine detail.

**Rationale:** This is a surgical operation. Model A should be in complete control for almost the entire generation process to establish the composition and subject. The influence of the specialized detail model (Model B) is only introduced in the final few output blocks, where the image is being upscaled and high-frequency details are rendered.

### Table 5: Recipe for Detail Grafting (Details B onto Base A)

| UNET Block(s) | Function | Weight A | Weight B | Justification |
| --- | --- | --- | --- | --- |
| **IN00 - OUT05** | Composition & Subject | 1.00 | 0.00 | Model A has full control to generate the entire image up to the final refinement stages. |
| **OUT06 - OUT08** | Initial Detail | 0.85 | 0.15 | A small amount of Model B is introduced to begin overlaying its detailed characteristics. |
| **OUT09 - OUT11** | Final Texture | 0.70 | 0.30 | Model B's influence is increased to ensure its textures and fine details are clearly present in the final output. |

## Section 5: Semantic and Aesthetic Control: Merging CLIP and VAE

While the UNET is the primary architect of the image's structure and style, the CLIP and VAE components provide critical semantic guidance and final aesthetic rendering. Merging these components is a necessary step for creating a truly cohesive and customized model.

### 5.1. Merging CLIP Encoders for Semantic Blending

The CLIP text encoder's role is to interpret the user's prompt. Merging CLIPs allows for the creation of a model that understands a broader, more nuanced range of concepts than either of its parents.

* **Strategy:** For most use cases, a simple weighted average is effective for merging CLIP encoders. In ComfyUI, this can be accomplished with the built-in CLIPMergeSimple node, which takes two CLIP models and a ratio as input.36 The merge formula used is  
  (1.0 - ratio) \* clip1 + ratio \* clip2.
* **Recipe:** To create a model that understands both photorealistic and anime-style prompts, one might merge the CLIP from a photoreal model (Model A) and an anime model (Model B).
  + **CLIP Model A:** clip1 input of CLIPMergeSimple.
  + **CLIP Model B:** clip2 input of CLIPMergeSimple.
  + **Ratio:** 0.40.
  + **Resulting Weights:** Model A: 0.60, Model B: 0.40.
  + **Justification:** This gives a slight dominance to the photorealistic model's vocabulary while still robustly integrating the anime-specific terminology from Model B. The resulting merged CLIP can more effectively interpret prompts like "a photorealistic portrait of a character in the style of [anime artist]," which might otherwise be misinterpreted.

### 5.2. Merging VAEs for Aesthetic Control

The VAE determines the final look of the image, especially its color palette, contrast, and the rendering of fine textures. VAE merging is a powerful tool for both correcting artifacts and achieving a specific aesthetic.

* **Strategy 1: Correction:** A common artifact of UNET/CLIP merging is a desaturated or color-shifted output.41 This is often due to a mismatch between the merged components. The simplest and most effective solution is often not to merge the VAEs at all.
  + **Recipe:** After merging the UNET and CLIP, connect the VAE from only one of the parent models (whichever has a more desirable color profile) or use a well-regarded community VAE (e.g., kl-f8-anime2 for stylized images, or the VAE from SD 1.5 for more realistic tones). This completely bypasses the VAE merge and often resolves color issues instantly.
* **Strategy 2: Blending:** To create a unique aesthetic, VAEs can be merged using a simple weighted average, similar to CLIPs. This can be done with a CheckpointLoaderSimple node followed by a ModelMergeSimple node acting on the VAE component, or by merging two full checkpoints and only using the resulting VAE.
  + **Recipe:** To achieve a soft, slightly desaturated but still vibrant look, one could merge a standard realistic VAE (Model A) with a VAE known for pastel or muted colors (Model B).
  + **VAE Model A Weight:** 0.70
  + **VAE Model B Weight:** 0.30
  + **Justification:** This blend will temper the high saturation of the realistic VAE with the softness of the pastel VAE, creating a unique and desirable color palette that is distinct from either parent.

## Section 6: Practical Implementation in ComfyUI

Translating the theoretical recipes into practice within ComfyUI requires familiarity with its node-based workflow and a few key custom node packages that provide the necessary tools for advanced merging.

### 6.1. Essential Custom Nodes

While ComfyUI has some built-in merging capabilities, advanced block-level control is best achieved through community-developed custom nodes. These can be installed via the ComfyUI Manager.

* **ComfyUI-DareMerge:** This is a comprehensive suite of merging tools based on advanced algorithms like DARE and TIES. It provides nodes such as ModelMerger (Advanced/DARE) and ModelMerger (Block/DARE), which offer granular control over the merging process, including the use of gradients and masks for targeted parameter updates.22
* **Mecha Merge Node Pack:** This is an alternative merging suite with a strong focus on memory efficiency. It uses a "recipe" system where complex merge operations are defined and then executed, which is particularly useful for intricate, multi-stage merges. It supports block-level hyperparameter specification via nodes like Blocks Mecha Hyper.43
* **Built-in Nodes:** For simpler merges, ComfyUI's native nodes are sufficient. These include ModelMergeSimple for basic weighted averaging, CLIPMergeSimple for text encoders, and ModelMergeBlocks for basic UNET block-level control.36

### 6.2. Workflow for a Block-Level UNET Merge

The following steps outline a typical workflow for implementing one of the UNET recipes from Section 4 using the built-in ModelMergeBlocks node.

1. **Load Checkpoints:** Add two LoadCheckpoint nodes to the canvas. In the first, select Model A. In the second, select Model B.
2. **Add Block Merge Node:** Add the ModelMergeBlocks node (found under advanced -> model\_merging).
3. **Connect Models:** Connect the MODEL output from the Model A checkpoint node to the model1 input of the ModelMergeBlocks node. Connect the MODEL output from the Model B checkpoint node to the model2 input.
4. **Enter Recipe Weights:** The ModelMergeBlocks node has float inputs for input, middle, and output blocks. These correspond to the 12 input blocks, 1 middle block, and 12 output blocks of the UNET. The weights should be entered as a comma-separated list of values between 0.0 and 1.0, where the value represents the influence of model2 (Model B).
   * For example, to implement the first row of the Style Transfer recipe (Table 3), where Model A has a weight of 0.90 and Model B has a weight of 0.10 for blocks IN00-IN02, you would enter 0.1, 0.1, 0.1,... into the input field. The node calculates Model A's weight as 1.0 - weight\_B.
   * **Input Field:** Enter the 12 weights for blocks IN00 through IN11.
   * **Middle Field:** Enter the single weight for block M00.
   * **Output Field:** Enter the 12 weights for blocks OUT00 through OUT11.
5. **Connect CLIP and VAE:** Connect the desired CLIP and VAE outputs (either from one of the parent models or from a separate merge operation as described in Section 5) to a KSampler node.
6. **Connect Merged Model:** Connect the MODEL output from the ModelMergeBlocks node to the model input of the KSampler node.
7. **Generate and Save:** Configure the KSampler with prompts and settings to test the merge. Once satisfied with the result, you can connect the MODEL output from the ModelMergeBlocks node to a CheckpointSave node to save the newly created model as a single file.

## Section 7: Advanced Strategies and Mitigating Pitfalls

Achieving mastery in model merging requires more than just following recipes; it involves developing strategies for selecting compatible models, understanding the risks of the process, and knowing how to diagnose and correct common failures.

### 7.1. Selecting Compatible Parents: The "Model Kinship" Heuristic

Not all models are suitable for merging, even if they share a base architecture. The concept of **"Model Kinship"** provides a useful heuristic for predicting the potential success of a merge.50 Model kinship is defined as the degree of similarity or relatedness between models, typically calculated based on the cosine similarity or correlation of their task vectors.

* **High Kinship:** Models with high similarity are said to have high kinship. Merging these models is generally safe and predictable, but it can lead to performance stagnation or "inbreeding," where the resulting model offers little novelty or improvement over the parents.
* **Low Kinship:** Models with low similarity have low kinship. Merging these models is riskier and more likely to result in artifacts or incoherence. However, a successful low-kinship merge can yield significant gains and novel capabilities that would be impossible to achieve from high-kinship parents.

**Practical Application:** Before committing to a complex and time-consuming block-level merge, perform a quick compatibility test. Use a ModelMergeSimple node to create a 50/50 average of the two prospective parent models.

* If the output is a coherent, albeit potentially flawed, image, the models likely have **high kinship** and are good candidates for a more nuanced block merge.
* If the output is chaotic noise, the models have **low kinship** and may reside in different loss basins. A successful merge will be challenging and may require more advanced techniques or a different pairing.

### 7.2. The Specter of Catastrophic Forgetting

A significant risk in model merging is **catastrophic forgetting**, where the merged model loses essential knowledge from its base or parent models.55 This occurs when the parameter adjustments from one model overwrite critical information learned by another. For example, in an attempt to add a specific artistic style, the merge might inadvertently damage the model's fundamental understanding of anatomy.

This risk is amplified by aggressive merging techniques like TIES or DARE, which intentionally discard a large portion of the parameters. While this is effective at reducing interference, it can also eliminate subtle but crucial information.

**Mitigation Strategy:** The primary way to mitigate catastrophic forgetting is to be less aggressive with merge weights. If a key capability from Model A is lost after a merge, the solution is to increase the influence of Model A in the UNET blocks functionally responsible for that capability. For instance, if a character's anatomy becomes distorted, increase the weight of the model with better anatomy in the early input blocks (IN00-IN05). This represents a direct trade-off: maximizing the combination of new skills versus preserving the integrity of existing ones.

### Table 6: Troubleshooting Common Merging Artifacts

| Artifact Description | Likely Cause (Block/Component) | Suggested Corrective Action |
| --- | --- | --- |
| Image is gray, desaturated, overly contrasted, or has color noise (e.g., magenta/green splotches). | **VAE Conflict.** The merged UNET is producing latent data that the VAE cannot properly decode. | **Do not merge the VAEs.** Use the VAE from Model A, Model B, or a known-good third-party VAE (e.g., kl-f8-anime2, SD 1.5 VAE). |
| Composition is chaotic; subjects are nonsensically blended or duplicated; scene is incoherent. | **Conflict in early UNET input blocks (IN00-IN05).** The models have fundamentally different ideas about composition. | **Establish dominance.** Give one model a very high weight (e.g., 0.90 or higher) in these blocks to force a single, coherent compositional structure. |
| Anatomy is distorted (e.g., extra limbs, warped faces), but the overall composition is acceptable. | **Conflict in the middle blocks (IN06-M00-OUT05).** The core semantic representations of the subjects are interfering. | **Attempt a balanced merge.** A 50/50 split in these blocks can sometimes create a stable hybrid. If that fails, favor one model more heavily (e.g., 0.70/0.30) to establish a clear "dominant" anatomy. |
| The desired style is weak or washed out; the final image looks too much like the base model. | **Insufficient weight in the style-defining blocks.** The style model's influence is too low. | **Increase the style model's weight** in the middle and late output blocks (M00, OUT00-OUT11). Also, consider increasing the overall merge ratio or the lambda in Task Arithmetic. |
| The prompt is misinterpreted; the model generates the wrong subject or ignores key descriptors. | **CLIP Conflict.** The merged CLIP does not understand the prompt as intended. | **Adjust the CLIP merge ratio.** If concepts from Model B are being ignored, increase Model B's ratio in the CLIPMergeSimple node. If the issue persists, use only the CLIP from the model that is most semantically aligned with your prompt. |

## Section 8: Evaluating and Refining Merged Models

Creating a merged checkpoint is not the final step; it is the beginning of a cycle of evaluation and refinement. A successful merge is not just one that avoids artifacts but one that achieves a specific, desired compositional goal.

### 8.1. Beyond "Looks Good": Metrics for Evaluation

Standard evaluation metrics that measure a model's performance on its original training tasks are often insufficient for assessing a merge. The true measure of a successful merge is its **compositional generalization**—its ability to handle novel combinations of concepts, styles, and subjects inherited from its parents.29

**Practical Tests for Compositional Generalization:**

* **Prompt Faithfulness:** The most direct test. Craft complex prompts that explicitly combine keywords, styles, and subjects unique to each parent model. For a merge of a "cyberpunk" model and a "fantasy" model, test prompts like "a knight in glowing neon armor" or "an elf with cybernetic implants." A good merge will render these concepts cohesively.
* **Style Consistency:** To test the robustness of a new style, generate a batch of images with a fixed style descriptor but varied subjects (e.g., "in the style of [merged model], a car," "..., a house," "..., a cat"). The style should remain consistent across all subjects.
* **Robustness and Stress Testing:** Evaluate the merged model's stability. Test it with complex negative prompts to see if it can correctly exclude concepts. Gradually increase the CFG scale; a well-merged model should remain stable at higher CFG values without producing overly saturated or distorted images.

### 8.2. Iterative Refinement

Model merging should be viewed as an iterative process, not a one-shot operation. The results from the evaluation phase should directly inform a second, refined iteration of the merge. This data-driven approach is the key to moving from a "good enough" merge to a precisely tailored custom model.

**Example Refinement Loop:**

1. **Initial Merge:** You perform a Style Transfer merge (Table 3) to apply a "watercolor" style (Model B) to a "character" model (Model A).
2. **Evaluation:** You generate images and find that while the style is present, some of the character's anatomical details from Model A are bleeding through, making the watercolor effect look muddy.
3. **Analysis:** The issue is likely that Model A's influence is too strong in the middle and output blocks.
4. **Refinement:** In the second iteration of the merge, you adjust the weights. For the M00 block, you change the weight from A: 0.25, B: 0.75 to A: 0.20, B: 0.80. For the OUT06-OUT11 blocks, you change the weight from A: 0.50, B: 0.50 to A: 0.40, B: 0.60.
5. **Re-evaluation:** You generate new images with the refined model and find that the watercolor style is now more dominant and coherent, successfully overwriting the unwanted details from Model A.

By following this cycle of merging, evaluating, and refining, practitioners can move beyond the limitations of existing models and create truly unique and powerful generative tools tailored to their specific creative vision.

#### Cytowane prace

1. Visualizing the Loss Landscape of Neural Nets - NIPS papers, otwierano: lipca 14, 2025, <https://proceedings.neurips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf>
2. Layer-Wise Linear Mode Connectivity - Michael Kamp, otwierano: lipca 14, 2025, <https://michaelkamp.org/?p=747>
3. Rethinking Weight-Averaged Model-merging - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/html/2411.09263v4>
4. Dokumenty\_Pomocnicze.pdf
5. Disrupting Model Merging: A Parameter-Level Defense Without Sacrificing Accuracy - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/abs/2503.07661>
6. Disrupting Model Merging: A Parameter-Level Defense Without Sacrificing Accuracy - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/html/2503.07661v2>
7. Disrupting Model Merging: A Parameter-Level Defense Without Sacrificing Accuracy, otwierano: lipca 14, 2025, <https://www.researchgate.net/publication/389748596_Disrupting_Model_Merging_A_Parameter-Level_Defense_Without_Sacrificing_Accuracy>
8. Disrupting Model Merging: A Parameter-Level Defense Without Sacrificing Accuracy | AI Research Paper Details - AIModels.fyi, otwierano: lipca 14, 2025, <https://www.aimodels.fyi/papers/arxiv/disrupting-model-merging-parameter-level-defense-without>
9. [D] My thoughts on model merging : r/MachineLearning - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/MachineLearning/comments/1ay56e6/d_my_thoughts_on_model_merging/>
10. [2411.18607] Task Arithmetic Through The Lens Of One-Shot Federated Learning - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/abs/2411.18607>
11. [2505.24262] On Fairness of Task Arithmetic: The Role of Task Vectors - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/abs/2505.24262>
12. NeurIPS Poster TIES-Merging: Resolving Interference When Merging Models, otwierano: lipca 14, 2025, <https://neurips.cc/virtual/2023/poster/69979>
13. Any easy to understand updated merging guide? : r/StableDiffusion - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/16xptvf/any_easy_to_understand_updated_merging_guide/>
14. Merge Large Language Models with mergekit - Hugging Face, otwierano: lipca 14, 2025, <https://huggingface.co/blog/mlabonne/merge-models>
15. What is Model Merging? Techniques & Challenges - Deepchecks, otwierano: lipca 14, 2025, <https://www.deepchecks.com/glossary/model-merging/>
16. SLERP For Model Merging – A Primer - Coinfeeds AI, otwierano: lipca 14, 2025, <https://www.coinfeeds.ai/ai-blog/slerp-model-merging-primer>
17. Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models, otwierano: lipca 14, 2025, <https://neurips.cc/virtual/2023/poster/73073>
18. TIES-Merging: Resolving Interference When Merging Models ..., otwierano: lipca 14, 2025, <https://openreview.net/forum?id=xtaX3WyCj1>
19. Evolutionary Optimization of Model Merging Recipes - arXiv, otwierano: lipca 14, 2025, <http://arxiv.org/pdf/2403.13187>
20. Papers Explained Review 13: Model Merging | by Ritvik Rastogi - Medium, otwierano: lipca 14, 2025, <https://ritvik19.medium.com/papers-explained-review-13-model-merging-d0db49797b90>
21. From Task-Specific Models to Unified Systems: A Review of Model Merging Approaches, otwierano: lipca 14, 2025, <https://arxiv.org/html/2503.08998v1>
22. ComfyUI-DareMerge detailed guide - RunComfy, otwierano: lipca 14, 2025, <https://www.runcomfy.com/comfyui-nodes/ComfyUI-DareMerge>
23. Paper page - Merging Models with Fisher-Weighted Averaging - Hugging Face, otwierano: lipca 14, 2025, <https://huggingface.co/papers/2111.09832>
24. Fisher's method - Wikipedia, otwierano: lipca 14, 2025, <https://en.wikipedia.org/wiki/Fisher%27s_method>
25. Merging by Matching Models in Task Parameter Subspaces - OpenReview, otwierano: lipca 14, 2025, <https://openreview.net/forum?id=qNGo6ghWFB>
26. Fisher Statistics for Analysis of Diffusion Tensor Directional Information - PMC, otwierano: lipca 14, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC3314136/>
27. Merging Models with Fisher-Weighted Averaging - CatalyzeX, otwierano: lipca 14, 2025, <https://www.catalyzex.com/paper/merging-models-with-fisher-weighted-averaging>
28. Model Merging in Pre-training of Large Language Models - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/html/2505.12082v1>
29. Realistic Evaluation of Model Merging for Compositional Generalization | OpenReview, otwierano: lipca 14, 2025, <https://openreview.net/forum?id=Bq3fEAGXUL>
30. merging weights of language models - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/pdf/2212.09849>
31. U-Net for Stable Diffusion - labml.ai, otwierano: lipca 14, 2025, <https://nn.labml.ai/diffusion/stable_diffusion/model/unet.html>
32. UNet2DConditionModel - Hugging Face, otwierano: lipca 14, 2025, <https://huggingface.co/docs/diffusers/main/api/models/unet2d-cond>
33. Modality specific U-Net variants for biomedical image segmentation: a survey - PMC, otwierano: lipca 14, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC8886195/>
34. CLIP: Contrastive Language-Image Pre-Training - Viso Suite, otwierano: lipca 14, 2025, <https://viso.ai/deep-learning/clip-machine-learning/>
35. TokLIP: Marry Visual Tokens to CLIP for Multimodal Comprehension and Generation - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/html/2505.05422v1>
36. CLIPMerge Simple - ComfyUI Wiki, otwierano: lipca 14, 2025, <https://comfyui-wiki.com/en/comfyui-nodes/advanced/model-merging/clip-merge-simple>
37. ClipMergeSimple - ComfyUI Built-in Node Documentation, otwierano: lipca 14, 2025, <https://docs.comfy.org/built-in-nodes/ClipMergeSimple>
38. VAE. The Latent Bottleneck: Why Image Generation Processes Lose Fine Details - Medium, otwierano: lipca 14, 2025, <https://medium.com/@efrat_37973/vae-the-latent-bottleneck-why-image-generation-processes-lose-fine-details-a056dcd6015e>
39. Is it possible to merge VAEs? : r/StableDiffusion - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/11ssxhl/is_it_possible_to_merge_vaes/>
40. Video Colorization Based on Variational Autoencoder - MDPI, otwierano: lipca 14, 2025, <https://www.mdpi.com/2079-9292/13/12/2412>
41. VAE merging · comfyanonymous ComfyUI · Discussion #1110 - GitHub, otwierano: lipca 14, 2025, <https://github.com/comfyanonymous/ComfyUI/discussions/1110>
42. Model Merger (Advanced/DARE) - RunComfy, otwierano: lipca 14, 2025, <https://www.runcomfy.com/comfyui-nodes/ComfyUI-DareMerge/DM_AdvancedDareModelMerger>
43. Mecha Merge Node Pack detailed guide | ComfyUI, otwierano: lipca 14, 2025, <https://www.runcomfy.com/comfyui-nodes/comfy-mecha>
44. ComfyUI Node: Recipe List - RunComfy, otwierano: lipca 14, 2025, <https://www.runcomfy.com/comfyui-nodes/comfy-mecha/Mecha-Recipe-List>
45. Blocks - ComfyUI Cloud - Comfy.ICU, otwierano: lipca 14, 2025, <https://comfy.icu/node/Blocks-Mecha-Hyper>
46. Model Merge Add - ComfyUI Wiki, otwierano: lipca 14, 2025, <https://comfyui-wiki.com/en/comfyui-nodes/advanced/model-merging/model-merge-add>
47. Model Merge Simple - ComfyUI Wiki, otwierano: lipca 14, 2025, <https://comfyui-wiki.com/en/comfyui-nodes/advanced/model-merging/model-merge-simple>
48. ComfyUI Model Merging Examples | ComfyUI Wiki, otwierano: lipca 14, 2025, <https://comfyui-wiki.com/en/workflows/model-merging>
49. Model Merge Blocks - ComfyUI Wiki, otwierano: lipca 14, 2025, <https://comfyui-wiki.com/en/comfyui-nodes/advanced/model-merging/model-merge-blocks>
50. Exploring Model Kinship for Merging Large Language Models - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/html/2410.12613v2>
51. EXPLORING MODEL KINSHIP FOR MERGING LARGE LANGUAGE MODELS - OpenReview, otwierano: lipca 14, 2025, <https://openreview.net/pdf?id=MR6RZQKMby>
52. (PDF) Exploring Model Kinship for Merging Large Language Models, otwierano: lipca 14, 2025, <https://www.researchgate.net/publication/384974549_Exploring_Model_Kinship_for_Merging_Large_Language_Models>
53. Exploring Model Kinship for Merging Large Language Models - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/html/2410.12613v1>
54. [2410.12613] Exploring Model Kinship for Merging Large Language Models - arXiv, otwierano: lipca 14, 2025, <https://arxiv.org/abs/2410.12613>
55. What is Catastrophic Forgetting? | IBM, otwierano: lipca 14, 2025, <https://www.ibm.com/think/topics/catastrophic-forgetting>
56. [2409.18314] Realistic Evaluation of Model Merging for Compositional Generalization, otwierano: lipca 14, 2025, <https://arxiv.org/abs/2409.18314>