# **An Expert Report on the Block Merging of Divergent SDXL Checkpoints in ComfyUI**

## **I. Deconstructing the SDXL Architecture: A Foundation for Merging**

To comprehend the complexities and challenges of merging Stable Diffusion XL (SDXL) models, particularly those that have architecturally diverged from the baseline, a foundational understanding of the core SDXL 1.0 architecture is imperative. The model is not a monolithic entity but a sophisticated, interconnected system of components. The success or failure of a merge operation hinges on the compatibility of these individual parts.

### **The UNet Backbone: Analyzing the 2.6B+ Parameter Core**

The heart of SDXL's image generation capability is its UNet backbone, a deep convolutional neural network responsible for the iterative denoising process that transforms a random noise tensor into a structured latent image.1 The most significant architectural leap from previous Stable Diffusion versions is the sheer scale of this UNet. With approximately 2.6 billion parameters, it is nearly three times larger than the 860 million parameter UNet found in Stable Diffusion v1.5.2 This substantial increase in parameter count is a primary driver of SDXL's ability to generate higher-fidelity, more detailed images at a native resolution of 1024x1024 pixels.4

Beyond its size, the SDXL UNet incorporates a critical design evolution: the strategic integration of transformer blocks.2 Unlike its predecessors, SDXL does not apply these attention-based mechanisms uniformly. Instead, it selectively employs transformer blocks at the mid-levels of the UNet while excluding them at the highest feature level and removing the lowest level of the UNet entirely.2 This heterogeneous distribution of transformer blocks is an optimization that enhances the model's capacity to interpret the complex relationships between textual prompts and visual elements without incurring prohibitive computational costs.2 This internal structure, divided into input blocks, a middle block, and output blocks, forms the very basis of the block-weighted merging techniques that will be explored later in this report.6

### **The Dual Text Encoder System: OpenCLIP ViT-bigG and CLIP ViT-L**

One of the most profound architectural changes in SDXL is its use of two distinct text encoders in tandem to interpret user prompts.3 This dual-encoder system provides a far richer and more nuanced understanding of text than was possible with the single encoder of previous versions.

1. **OpenAI CLIP ViT-L:** This encoder provides a 768-dimensional embedding vector for each token in the prompt. Its inclusion maintains a degree of conceptual continuity with the prompting styles developed for Stable Diffusion v1.5.3
2. **OpenCLIP ViT-bigG:** This is a significantly larger and more powerful text encoder, trained on the expansive LAION-2B dataset. It outputs a 1280-dimensional embedding vector for each token.2

During processing, the embeddings from both encoders are concatenated. This creates a final conditioning vector with a dimension of 2048 (768+1280) for each prompt token.3 This high-dimensional representation is then fed into the UNet's cross-attention layers, allowing the model to guide the diffusion process with an unprecedented level of textual detail and semantic depth.

### **The Role of Pooled Embeddings and Cross-Attention Context**

In addition to the per-token concatenated embeddings, SDXL employs a second text-based conditioning mechanism: pooled text embeddings.2 The OpenCLIP ViT-bigG model generates a single, 1280-dimensional vector that represents a "pooled" or summary embedding of the entire prompt. This vector provides a global contextual signal to the UNet, conditioning it on the overall theme and subject matter of the prompt, which complements the specific, token-level guidance from the cross-attention mechanism.2 This dual-conditioning approach is a key factor in SDXL's improved prompt adherence and compositional coherence.

The components of the SDXL architecture are a deeply interdependent system, not a collection of modular, interchangeable parts. The cross-attention layers within the UNet are not generic; they are specifically designed and sized to accept the 2048-dimensional concatenated embeddings from the dual text encoders. Likewise, the model is conditioned on the specific 1280-dimensional pooled output from the OpenCLIP model. This creates a hard-coded dimensional linkage between the text encoders and the UNet. Any fine-tuning process that significantly alters the architecture or output dimensions of the text encoders will break this link, rendering the model incompatible with a standard UNet. This fundamental principle of architectural dependency is the primary source of the "shape mismatch" errors that are central to the challenge of merging divergent models.

### **The Optional Refiner Model and the Two-Stage Pipeline**

The full SDXL pipeline is designed as a two-stage process, often described as an "ensemble of experts".4 The process begins with the base model, which performs the primary text-to-image generation, creating a full latent image from noise.8 Subsequently, an optional refiner model can be used to enhance this initial output.2

The refiner is a separate diffusion model that is also conditioned on the prompt but is specifically trained to operate on low-noise latents. It excels at adding high-frequency details, refining textures, improving facial features, and correcting minor artifacts left by the base model.2 While the refiner is a crucial part of achieving maximum image quality with the official SDXL pipeline, the focus of this report is on the merging of full base model checkpoints. Therefore, the refiner itself is not typically involved in the merging process but understanding its function provides a complete picture of the intended workflow.

## **II. Architectural Analysis of Key SDXL Fine-Tunes: A Comparative Study**

The open-source nature of SDXL has led to the creation of numerous fine-tuned checkpoints, each specialized for a particular aesthetic or purpose. These fine-tunes often diverge from the base architecture through intensive training on niche datasets. Understanding these divergences is the first step toward planning a successful merge. The models requested for analysis—Illustrious, NoobAI, and RouWei—form a distinct lineage of anime-style models, where each subsequent model builds upon and further specializes the work of its predecessor.

### **SDXL Illustrious: The High-Resolution Illustrator**

* **Core Identity:** Illustrious is a prominent anime-focused model series developed by OnomaAI. It is not a direct fine-tune of the base SDXL 1.0 but is built upon a specific checkpoint, kohaku-xl-beta5.10 Its training is heavily based on the Danbooru dataset, a large repository of anime-style images with detailed tags, making Illustrious highly proficient with tag-based prompting.12
* **Key Architectural Divergence 1: Native High Resolution.** A defining feature of Illustrious XL (v1.0 and later) is its native support for generating images at resolutions up to 1536x1536 pixels without requiring a separate high-resolution fix pass.13 This capability is a significant departure from the standard SDXL's native 1024x1024 resolution and suggests that internal layers, possibly related to positional encoding or attention mechanisms, have been modified to handle larger latent dimensions effectively.
* **Key Architectural Divergence 2: Hybrid Prompting System.** While its foundation is in Danbooru tags, later versions of Illustrious were explicitly trained to improve their understanding of natural language prompts.13 This creates a flexible hybrid system where users can combine descriptive sentences with precise tags, indicating a sophisticated fine-tuning of the model's dual text encoders.
* **Compatibility:** Due to the thoroughness of its training and its significant divergence from the base model, Illustrious is noted to be "not fully compatible with 'regular' SDXL LoRAs, ControlNets, etc.".10 It has effectively spawned its own ecosystem of compatible tools and extensions.

### **SDXL NoobAI: The Tag-Driven Anime Specialist**

* **Core Identity:** NoobAI-XL is a direct descendant of Illustrious, specifically fine-tuned from an early version, Illustrious-xl-early-release-v0.16 It undergoes an even more extensive training regimen, utilizing a massive dataset of approximately 13 million images from Danbooru and e621 (a furry art site).16 This makes NoobAI exceptionally powerful and predictable when using Danbooru tags, particularly for anime and furry concepts.16
* **Key Architectural Divergence 1: Prediction Type.** NoobAI models introduce a critical architectural split. They are released in two incompatible versions: eps-prediction and v-prediction.20 Standard diffusion models, including base SDXL and Illustrious, are  
  eps-prediction models, meaning the UNet is trained to predict the noise (epsilon) that was added to the image. In contrast, v-prediction models are trained to predict a different variable (v), which can lead to better performance in certain scenarios but represents a fundamental change in the model's output objective. An eps model and a v model cannot be merged directly using standard techniques, as their UNets are predicting different targets.
* **Key Architectural Divergence 2: Extreme Specialization.** The intense focus on tag-based datasets makes NoobAI highly reliant on specific prompting structures, such as artist tags and quality tags.16 This specialization represents a significant architectural drift, as the text encoders and UNet become optimized for a structured, symbolic language, potentially at the expense of the more generalized natural language understanding of its parent, Illustrious.
* **Compatibility:** As a derivative of an already-divergent model, NoobAI is "largely incompatible with extensions and LoRA from 'regular' SDXL models" and requires its own set of compatible ControlNets and other tools.20

### **SDXL RouWei: The Aesthetic Successor**

* **Core Identity:** The RouWei model is a more recent checkpoint that exists within the same anime-focused lineage. It is often used as a base for further merges, such as the JANKU v4.0 model, which explicitly states it is trained on a RouWei base and uses NoobAI/Illustrious quality triggers.23 This places it firmly in the same family.
* **Architectural Lineage:** RouWei is described as an eps-prediction model, making it compatible with Illustrious and the eps version of NoobAI from a prediction-type standpoint.23 Its configuration file for the second text encoder reveals a standard  
  CLIPTextModelWithProjection with a hidden size of 1280, which is consistent with the OpenCLIP component of the base SDXL architecture.24
* **Key Features:** Like its relatives, RouWei supports native generation at high resolutions (up to 1536px) and has a VAE baked into the checkpoint file for convenience.23 It is trained on over 35,000 artist styles, aiming to provide "easy and versatile prompting with great aesthetic, anatomy, and stability".23

The development path of these models reveals a clear lineage: the foundational SDXL architecture was adapted by the kohaku-xl-beta checkpoint, which was then fine-tuned to create Illustrious. An early version of Illustrious was then used as the base for the even more specialized NoobAI. RouWei appears as a parallel branch or a subsequent merge within this same family. The feasibility of any merge operation is directly related to the models' proximity on this family tree. Merging two models from the Illustrious branch (e.g., Illustrious v1.0 and RouWei) is an exercise in style blending and is far more likely to produce a coherent result than attempting to merge a model from this family with an unrelated SDXL fine-tune designed for photorealism. Researching this "genealogy" is the critical first step in planning a successful merge, as it provides a strong indicator of underlying architectural compatibility.

| Model Name | Base Model/Lineage | Key Training Data | Native Resolution | Prediction Type | Text Encoder Characteristics | Core Strengths | Known Incompatibilities |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SDXL 1.0 Base** | N/A | Broad, general-purpose dataset | 1024x1024 4 | Epsilon (eps) | Dual CLIP (ViT-L + OpenCLIP ViT-bigG) 2 | Versatility, good prompt understanding, photorealism | Incompatible with SD1.5/2.x LoRAs and ControlNets 25 |
| **SDXL Illustrious** | SDXL (via kohaku-xl-beta5) 10 | Danbooru dataset (anime/illustration tags) 13 | Up to 1536x1536 13 | Epsilon (eps) | Fine-tuned for hybrid tag/natural language prompting 13 | High-quality anime style, high resolution, good anatomy | Not fully compatible with standard SDXL LoRAs/ControlNets 10 |
| **SDXL NoobAI (eps)** | Illustrious-xl-early-release-v0 16 | Danbooru & e621 datasets (~13M images) 16 | Up to 1344x768 (recommended) 16 | Epsilon (eps) | Heavily optimized for Danbooru tag structure 16 | Excellent tag adherence, character/artist knowledge, furry content | Incompatible with standard SDXL ecosystem; requires own ControlNets 20 |
| **SDXL NoobAI (v-pred)** | Illustrious-xl-early-release-v0 22 | Danbooru & e621 datasets 18 | Up to 1344x768 (recommended) 22 | V-prediction (v) | Same as eps version but with different prediction target | Same as eps version; may offer different quality trade-offs | Incompatible with eps-prediction models, including base SDXL and Illustrious 21 |
| **SDXL RouWei** | Illustrious/NoobAI lineage 23 | 35k+ artist styles, gacha game images 23 | Up to 1536x1536 23 | Epsilon (eps) | Uses Illustrious/NoobAI prompting style 23 | Vibrant colors, aesthetic stability, good anatomy | Shares incompatibilities with the Illustrious/NoobAI family |

## **III. The Challenge of Incompatibility: Understanding Architectural Divergence and Its Consequences**

The desire to merge different SDXL models stems from the creative impulse to combine their unique strengths. However, this process is frequently halted by technical barriers, most notably the "Tensor Shape Mismatch" error. This error is not a superficial bug; it is a clear symptom of fundamental architectural incompatibilities between the models being merged.

### **The Root of the Problem: Why Direct Merging Fails**

At its core, model merging is a mathematical operation performed on the model's parameters, which are stored in multi-dimensional arrays called tensors.6 The simplest form of merging, weighted sum, calculates a new tensor for each layer by averaging the corresponding tensors from the parent models. This process inherently assumes two conditions are met: first, that the tensors for a given layer in both models have the exact same dimensions (or "shape"), and second, that the layer serves the same functional purpose in both architectures.28

Intensive fine-tuning, especially on highly specialized or divergent datasets, can violate these assumptions. The process of training doesn't just alter the numerical values within the tensors; in some cases, it can lead to changes in the model's very structure, breaking the one-to-one correspondence between layers and making a direct mathematical combination impossible.10

### **Technical Deep Dive: The "Tensor Shape Mismatch" Error Explained**

The "Tensor Shape Mismatch" error is the most common and explicit indicator of architectural incompatibility. Error messages generated by frameworks like PyTorch are highly informative and can be used to diagnose the precise point of failure.

Consider a typical error message encountered during a merge attempt:

size mismatch for model.diffusion\_model.input\_blocks.1.1.transformer\_blocks.0.attn2.to\_v.weight: copying a param with shape torch.Size() from checkpoint, the shape in current model is torch.Size().31

This message can be deconstructed to reveal the root cause:

* **Layer Path:** model.diffusion\_model.input\_blocks.1.1.transformer\_blocks.0.attn2.to\_v.weight pinpoints the exact location of the problematic tensor. It is in the first input block of the UNet, within a transformer block's second attention mechanism (attn2), specifically the weight tensor for the value (v) projection. This is a cross-attention layer, which is responsible for integrating the text prompt conditioning.
* **The Mismatch:** The core of the issue lies in the tensor shapes: torch.Size() versus torch.Size().
  + The first dimension, 320, represents the number of channels in the UNet at this particular block. The fact that this matches indicates that the general convolutional structure of the UNets is compatible at this stage.
  + The second dimension, 1024 versus 768, is the dimension of the text conditioning vector that the layer is designed to receive. This mismatch is the "smoking gun." It proves that the two models were trained to interface with text encoders that produce different-sized outputs. One model expects a 1024-dimensional vector (characteristic of SD2.x models or certain fine-tunes), while the other expects a 768-dimensional vector (characteristic of SD1.5 or the ViT-L portion of SDXL).25 They are, quite literally, speaking different languages at an architectural level.

Another critical example of shape mismatch occurs with specialized models like inpainting models. Users report errors where the first convolutional layer of the UNet expects a different number of input channels: vs..33 This is because a standard model's UNet takes 4 latent channels as input, whereas an inpainting model's UNet is architecturally modified to accept an additional 5 channels: 4 for the encoded masked image and 1 for the mask itself, for a total of 9 input channels.33 Attempting to merge these two types of models will fail at the very first layer due to this hard-coded structural difference.

### **Beyond the UNet: How Divergence in Text Encoders and VAEs Complicates Merging**

A successful merge requires more than just a compatible UNet. The resulting model must be paired with a coherent text encoder (CLIP) and Variational Autoencoder (VAE) to function correctly.

If Model A and Model B have text encoders that have been fine-tuned on vastly different data and prompting styles (e.g., the heavily tag-trained encoder of Pony Diffusion vs. a standard SDXL encoder), simply picking one or averaging them can lead to disastrous results. The merged UNet, being a hybrid, may not fully understand the "language" of either parent CLIP, leading to severe concept confusion and poor prompt adherence.35 Similarly, VAEs can be fine-tuned for specific aesthetics (e.g., anime styles), and using a mismatched VAE can introduce color shifts or artifacts. Fortunately, interfaces like ComfyUI provide the essential capability to handle the UNet, CLIP, and VAE as separate components, allowing an advanced user to merge the UNet while selecting the most appropriate, un-merged CLIP and VAE for the task.6

The very "incompatibility" of specialized models like Illustrious and NoobAI is not an accidental flaw but a direct and necessary consequence of their advanced capabilities. The features that make them desirable—such as native high-resolution output or unparalleled mastery of tag-based prompting—are achieved by deliberately modifying their architecture and training them in ways that cause them to diverge from the general-purpose base model.13 This specialization reconfigures the model's internal neural pathways, making it an expert in one domain at the cost of being a generalist. This reconfiguration is precisely what leads to the technical incompatibilities. Therefore, the user's objective should not be to crudely force two incompatible architectures together, but to employ advanced techniques to surgically transfer the desired

*knowledge* or *style* from one model to another while respecting and preserving the fundamental structure of the recipient model.

## **IV. Practical Guide to Block Merging in ComfyUI: From Simple Blends to Granular Control**

Transitioning from theory to practice, ComfyUI offers a powerful and flexible node-based environment for experimenting with model merging. This section provides a practical guide to using ComfyUI's built-in tools for block merging, a technique that offers more granular control than a simple weighted average.

### **Setting Up the Workflow: Essential Nodes**

A typical model merging workflow in ComfyUI involves a set of core nodes that allow for loading, merging, testing, and saving models.38

* **Load Checkpoint:** Two of these nodes are needed to load the parent models, Model A and Model B.
* **Merging Node:** This is the central node where the merge operation is defined. ComfyUI offers several options, with ModelMergeBlocks being the most common for this task.
* **KSampler (or KSampler Advanced):** This node takes the merged model output and generates a test image, allowing for live preview of the merge's result. It should be connected to the standard CLIP Text Encode and Empty Latent Image nodes.
* **Save Checkpoint:** Once a desirable merge is achieved, the output model from the merging node is connected to this node to save the new hybrid checkpoint to disk. This node is often muted during testing and unmuted for the final save.38

### **The ModelMergeBlocks Node: The Standard Tool for Block Weighting**

The ModelMergeBlocks node is the workhorse for most block-weighted merging experiments. It provides a straightforward interface to control the influence of each model on the three main sections of the UNet.6

The node has the following key input parameters:

* model1: The first model to be merged (the base).
* model2: The second model to be merged (the influence).
* input: A floating-point value from 0.0 to 1.0. This slider controls the contribution from the UNet's input blocks. A value of 0.0 uses 100% of model1's input blocks, while 1.0 uses 100% of model2's.
* middle: A float controlling the weight for the single middle block of the UNet.
* out: A float controlling the weight for the UNet's output blocks.

While the exact function of each block can vary between models, user experimentation has established a set of useful heuristics for what these blocks generally control 6:

* **Input Blocks (input):** These layers, processed early in the denoising step, tend to have a strong influence on the overall composition, structure, color palette, and background elements of the image.
* **Middle Block (middle):** This block affects both composition and some of the finer details, acting as a bridge between the broad strokes and the final rendering.
* **Output Blocks (out):** These layers, processed later in the denoising step, have a more significant impact on the final details, textures, lighting, and the specific rendering of the subject's style and features.

These are not absolute rules, and the key to successful merging is systematic experimentation to discover how these blocks interact for the specific models being merged.6

### **The ModelMergeSDXL Node: A Layer-by-Layer Approach for Precision Merging**

For users requiring even more fine-grained control, ComfyUI provides the ModelMergeSDXL node. This advanced node exposes nearly every block of the SDXL UNet as an individual slider, allowing for extremely precise adjustments.41 It includes separate float sliders for the

time\_embed and label\_emb layers, as well as for each of the nine input blocks (input\_blocks.0 to input\_blocks.8), three middle blocks (middle\_block.0 to middle\_block.2), and nine output blocks (output\_blocks.0 to output\_blocks.8). This level of control is powerful but can be overwhelming. It is best reserved for scenarios where a user has already used ModelMergeBlocks to identify a specific block or small set of blocks responsible for a desired effect and wishes to target them with surgical precision.

### **Step-by-Step Walkthrough: A Hypothetical Merge of RouWei and Illustrious**

This practical example demonstrates a systematic process for merging two closely related models, SDXL Illustrious and SDXL RouWei, using the ModelMergeBlocks node.

1. **Workflow Setup:** In ComfyUI, create two Load Checkpoint nodes. Load Illustrious.safetensors as Model A (the base) and RouWei.safetensors as Model B (the style donor). Connect their MODEL and CLIP outputs to a ModelMergeBlocks node. For this initial merge of closely related models, the VAE can be taken directly from either parent model. Connect the output of the merge node to a KSampler.
2. **Systematic Testing:** The goal is to understand what each block contributes. A robust method is to create a test grid by generating an image with a fixed prompt and seed for each of the eight primary permutations of the block weights.6 This involves setting the  
   input, middle, and out sliders to either 0.0 (using Illustrious's block) or 1.0 (using RouWei's block) and generating an image for each combination (e.g., IN=0, MID=0, OUT=1; IN=0, MID=1, OUT=0; etc.).
3. **Analysis:** Arrange the eight resulting images in a grid for comparison. Analyze the outputs to identify which blocks from RouWei are responsible for its desired characteristics (e.g., "vibrant colors, smooth gradients" 23) and which blocks from Illustrious are essential for maintaining its core strengths (e.g., strong composition and prompt adherence). For instance, one might find that using RouWei's output blocks (  
   out=1.0) imparts the desired color and lighting, while using Illustrious's input blocks (input=0.0) preserves better anatomy and background coherence.
4. **Refinement and Blending:** Based on the analysis, move from binary (0.0/1.0) weights to fractional weights. If RouWei's output blocks provide a desirable but overly strong effect, reduce the out slider to a value like 0.7 or 0.5 to create a more subtle blend.
5. **CLIP and VAE Handling:** Since these models are from the same lineage, their CLIPs are likely to be highly compatible. A good starting strategy is to use the CLIP from the model you prefer for prompt understanding (e.g., the later-version Illustrious). Alternatively, a CLIPMerge node could be used to blend them at a specific ratio. The VAE can be taken from whichever model produces more pleasing colors.
6. **Saving the Checkpoint:** Once a combination of block weights produces a consistently desirable result across several test prompts, unmute the CheckpointSave node, give the new model a descriptive name, and execute the workflow one last time to save the final .safetensors file.

| Node Name | Primary Function | Control Granularity | Key Parameters | Typical Use Case | Complexity |
| --- | --- | --- | --- | --- | --- |
| **ModelMergeSimple** | Weighted sum of two full models. | Low (Single Ratio) | ratio | Quick, simple blends; initial experimentation. 6 | Low |
| **ModelMergeBlocks** | Weighted sum of UNet block sections. | Medium (3 Ratios) | input, middle, out | Standard block-weighted merging; balancing composition and style. 6 | Medium |
| **ModelMergeSDXL** | Weighted sum of individual UNet blocks. | High (20+ Ratios) | input\_blocks.0-8, middle\_blocks.0-2, etc. | Surgical merging to target specific layers identified in prior tests. 41 | High |
| **Model Merger (Advanced/DARE)** | Advanced merging using DARE-TIES algorithm. | Very High (Algorithmic) | density, seed, mask, gradient | Transferring "skills" between incompatible models; preserving base model integrity. 42 | Very High |
| **Model Merger (Block/DARE)** | Applies DARE algorithm to UNet block sections. | High (Algorithmic + 3 Ratios) | input, middle, out, density, seed | Applying DARE's intelligent merging to specific sections of the UNet. 42 | High |

## **V. Advanced Merging with ComfyUI-DareMerge: A Deep Dive into DARE-TIES, Masking, and Gradient Control**

For merging tasks that involve architecturally divergent or highly specialized models, standard block-weighted averaging is often insufficient and can result in "fried" or incoherent models. The ComfyUI-DareMerge custom node pack offers a suite of advanced tools based on state-of-the-art merging algorithms, enabling a more surgical approach to model combination.42

### **Introduction to the DARE-TIES Method**

The ComfyUI-DareMerge extension is primarily based on methods described in the research paper "Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch".42 This research introduces techniques that move beyond simple weight averaging to intelligently integrate knowledge from one model into another.

* **DARE (Drop and Rescale):** This technique addresses the issue of "conflicting edits" during a merge. It involves calculating the delta (the difference in weights) between a fine-tuned model and its base. It then randomly "drops" a fraction of these deltas (setting them to zero) and "rescales" the remaining ones to preserve the overall magnitude of the update. This prevents the merge from trying to apply contradictory changes from the parent models.
* **TIES-Merging (Trim, Elect, and Merge):** This is a more sophisticated method that first "trims" the delta tensor, discarding small-magnitude changes that are likely noise. It then "elects" the most significant changes by resolving sign disagreements between the parent models. Finally, it "merges" only this filtered set of non-conflicting, high-impact updates.43

The fundamental concept behind these methods is to stop treating models as monolithic blocks of weights to be averaged. Instead, they treat a fine-tune as a set of *changes* or *skills* applied to a base model. The goal of a DARE-TIES merge is to identify and transfer only the most important and non-conflicting skills, thereby "teaching" the base model a new ability without destroying its existing knowledge.43

### **Core DareMerge Nodes: The Surgical Tools**

The DareMerge extension provides several key nodes for performing these advanced merges:

* **Model Merger (Advanced/DARE):** This is the most powerful node, implementing the full DARE-TIES algorithm. It allows for highly targeted control using gradients and masks, making it the tool of choice for complex merges between incompatible models.42
* **Model Merger (Block/DARE):** This node combines the DARE methodology with the block-based approach. It allows the user to apply the DARE algorithm with specific ratios to the input, middle, and output blocks of the UNet, offering a bridge between standard block merging and full DARE-TIES.42

### **The Power of Masking: Protecting What's Important**

Masking is a central concept in DareMerge that enables the surgical precision of the merge. A mask is a tensor that specifies which parameters in a model should be included in (whitelisted) or excluded from (blacklisted) the merge operation. This allows a user to "protect" critical parts of a base model from being overwritten.42

The most crucial node for this purpose is the **Magnitude Masker**. This node creates a mask by comparing a fine-tuned model to its original base model. It calculates the delta for each parameter and creates a mask that identifies the parameters with the largest magnitude changes. These high-magnitude parameters represent the most significant "learning" that occurred during the fine-tuning process.42

In a practical scenario, a user can create a magnitude mask of their primary model (e.g., Illustrious) to isolate its most essential, highly trained parameters. This mask is then used during the merge to protect these parameters while allowing changes from a secondary model (e.g., NoobAI) to be applied only to the less critical, unmasked areas. This is the essence of performing model surgery: preserving the vital organs of the host while transplanting new tissue.

### **Advanced Workflow: Implementing a DARE-TIES Merge to Integrate NoobAI into an Illustrious Base**

This workflow outlines a procedure for a challenging merge that would likely fail with standard methods: transferring the superior Danbooru tag knowledge of NoobAI into the more compositionally stable Illustrious model.

1. **Define the Goal and Roles:** The objective is to enhance Illustrious with NoobAI's encyclopedic knowledge of anime characters and tags, without corrupting Illustrious's high-resolution stability and aesthetic.
   * **Model A (Host):** Illustrious.safetensors. This is the model whose core structure and style we want to preserve.
   * **Model B (Donor):** NoobAI-eps.safetensors. This is the model containing the "skills" we want to transfer.
   * **Base Model (Reference):** A common ancestor model, such as kohaku-xl-beta5.safetensors or an early Illustrious version. This is needed to calculate the deltas for masking.
2. **Create a Protective Mask:**
   * In ComfyUI, add a Magnitude Masker node.
   * Connect Model A (Illustrious) and the Base Model to its inputs.
   * Configure the masker to create a mask that selects for the parameters with the *highest* magnitude. This mask now represents the "essence" of Illustrious—the parts of the model that were most changed during its fine-tuning and are most critical to its identity.
3. **Perform the Surgical Merge:**
   * Add a Model Merger (Advanced/DARE) node.
   * Connect Model A (Illustrious) to the model1 input.
   * Connect Model B (NoobAI) to the model2 input.
   * Crucially, connect the protective mask created in the previous step to the mask input of the merger node. This instructs the algorithm to apply changes from NoobAI *only* to the parameters of Illustrious that are *not* protected by the mask.
4. **Tune and Iterate:** The DARE-TIES merge process involves random sampling, so the results will vary with the seed. The node includes parameters like density which control the sparsity of the merge.
   * Experiment with different integer seeds in the DARE merger node. A different seed can lead to a dramatically different—and potentially better—outcome.42
   * Adjust the density parameter. A lower density results in a more conservative merge, transferring fewer parameters from Model B.
5. **Evaluation:** Test the newly created model rigorously.
   * Use prompts containing specific, obscure Danbooru tags or character names that NoobAI is known to handle well.
   * Check if the output now recognizes these concepts while still retaining the overall visual style, composition, and high-resolution quality of the Illustrious base model. If successful, the merge has effectively transplanted the knowledge without rejecting the organ.

## **VI. Analysis of Merged Model Outcomes: Pros, Cons, and Artifacts**

Model merging is an experimental process of creative alchemy. The results can range from a transcendent hybrid that combines the best qualities of its parents to an unusable, incoherent failure. A thorough analysis of the potential outcomes and a systematic method for evaluation are essential for any practitioner.

### **Potential Benefits: The "Best of Both Worlds" Scenario**

When successful, a model merge can yield significant benefits that are unattainable with a single checkpoint or LoRA:

* **Novel Style Blending:** The most common goal is to create a unique aesthetic that sits in the creative space between two parent models. Merging a painterly model with a photographic one, for example, can produce a distinct "photo-artistic" style.6
* **Knowledge and Capability Transfer:** A merge can be used to imbue a stylistically superior model with the conceptual knowledge of another. For instance, merging a model with excellent anatomy into a highly stylized anime model can potentially improve the latter's ability to render complex poses.45
* **Creation of "All-in-One" Models:** Merging can consolidate capabilities, creating a single checkpoint that can handle a wider range of styles or subjects that would otherwise require switching between multiple models, such as combining a model proficient with human characters with one trained on furry characters.27

### **Common Pitfalls and Artifacts: When Merges Go Wrong**

The path of model merging is fraught with peril, and failures often manifest as distinct visual artifacts:

* **Style Bleed and Incoherence:** The most frequent issue is an undesirable blending of styles, where unwanted elements from one model contaminate the other, resulting in an aesthetically displeasing or compositionally messy output.
* **Concept Confusion:** The merged model's understanding of the world can become muddled. The neural pathways for different concepts get cross-wired, leading to bizarre hybrid objects or a failure to adhere to the prompt.
* **"Fried" Models:** This is the term for a catastrophic merge failure. The resulting model produces images that are noisy, heavily desaturated or grey, or filled with chaotic, distorted patterns. This is often a symptom of merging two architecturally incompatible models or using overly aggressive merge ratios.47 A common tell-tale sign of a "fried" merge is the appearance of strange purple or blue splotches on skin tones, resembling bruising.47
* **Inherited Artifacts:** The merged model will inherit the biases and flaws of its parents. If one of the parent models was trained on a dataset containing watermarks, the merged model may also generate faint watermarks or text artifacts.23

### **Evaluating a Merge: A Systematic Approach**

Given the experimental nature of merging, a rigorous and systematic evaluation process is critical to validate the quality of a new checkpoint.6

1. **Isolate Variables:** When testing merge settings, it is crucial to keep all other generation parameters—prompt, negative prompt, seed, sampler, steps, CFG scale—absolutely constant. The only variable should be the merge parameter being tested. This ensures an "apples-to-apples" comparison.6
2. **Comprehensive Prompting:** Test the merged model with a diverse set of prompts that cover its intended capabilities. This should include simple subjects, complex scenes with multiple characters, landscapes, and abstract concepts.
3. **Stability Testing:** Use a single, reliable prompt and generate images with a large number of different seeds. This tests the model's stability and consistency. A good merge should produce high-quality, coherent images across a wide range of seeds.
4. **Visual Comparison:** Use a tool like PureRef to create a large canvas where outputs from different merge settings can be arranged and compared side-by-side. This visual organization is invaluable for identifying subtle differences and trends.6

A successful merge does not merely combine the visual outputs of its parents; it creates an entirely new model with its own unique internal "language" and prompt sensitivities. The specific keywords, phrases, and weightings that were optimal for the parent models may no longer be effective for the child. For example, models like NoobAI are highly tuned for a specific Danbooru tag structure, while Illustrious responds to a hybrid of tags and natural language.13 When these models are merged, the underlying neural pathways that are activated by a prompt are now a composite of both parents. The merged model's "understanding" of a concept is fundamentally new. Consequently, users who simply reuse their old prompts with a merged model may get suboptimal results and mistakenly conclude that the merge "lost the benefits of both models".35 A crucial part of the post-merge process is a new phase of prompt engineering and exploration to discover the specific phrasing that "resonates" most strongly with the new, hybrid architecture.

## **VII. Expert Recommendations and Future Outlook**

Model merging is a powerful but complex technique that sits at the frontier of generative AI customization. Success requires a combination of technical understanding, systematic experimentation, and artistic judgment. This final section synthesizes the report's findings into a set of strategic recommendations and offers a perspective on the future of this evolving field.

### **Strategic Recommendations for Successful Merging**

* **Research Model Lineage:** Before attempting any merge, investigate the development history of the target models. The probability of a successful merge is significantly higher for models that share a close common ancestor or belong to the same "family" (e.g., two derivatives of Illustrious).
* **Define a Clear Objective:** Do not merge aimlessly. Have a specific, articulated goal, such as "I want to transfer the anatomical knowledge from Model A into the aesthetic style of Model B." This goal will guide all subsequent decisions.
* **Adopt a Tiered Approach to Complexity:** Begin with the simplest tools that can achieve the goal. Start with a ModelMergeBlocks node and perform systematic permutation testing.6 Only escalate to the more complex  
  ModelMergeSDXL or the highly advanced ComfyUI-DareMerge nodes when simpler methods prove insufficient for dealing with incompatibilities.
* **Practice Methodical Experimentation:** Isolate one variable at a time during testing. Whether it is the merge ratio, the prompt, or the seed, changing only one parameter allows for a clear understanding of its specific effect.6
* **Manage All Model Components:** Remember that a checkpoint contains a UNet, a CLIP, and a VAE. The merge strategy must account for all three. While the UNet is typically the focus of the merge operation, the choice of which CLIP to use (or whether to merge them) is equally critical for prompt adherence.
* **Embrace and Document Failure:** Expect that many, if not most, experimental merges will fail. Treat each failure as a data point. Document the models used, the merge settings, and the resulting artifacts. This documentation is invaluable for learning and refining future attempts.6

### **When to Merge vs. When to Use LoRAs**

Model merging and Low-Rank Adaptations (LoRAs) are both methods for customizing model output, but they serve fundamentally different purposes.

* **Use LoRAs for Targeted Augmentation:** A LoRA is the ideal tool for adding a specific, well-defined concept, character, or style to a base model without altering its core behavior. LoRAs are non-destructive patches that can be applied at varying strengths. They are perfect for when the user is happy with their base model but wants to teach it a new, discrete "skill".47
* **Use Merging for Foundational Hybridization:** Merging is a destructive and transformative process. It should be used when the goal is to create a fundamentally new base model with a hybrid core behavior and aesthetic that cannot be achieved with a single LoRA or a stack of LoRAs. Merging creates a new foundation upon which LoRAs can then be applied.

### **The Future of Model Merging**

The practice of model merging is continuously evolving, driven by the innovation of the open-source community. The development of techniques like DARE-TIES represents a significant step away from crude averaging and toward more intelligent, surgical methods of knowledge transfer.

Looking ahead, the field is likely to see the emergence of even more sophisticated merging tools. One could envision AI-assisted merging systems that can automatically analyze the architectural similarities and differences between two models and recommend an optimal merge strategy, or even identify the specific layers responsible for certain artistic traits.

However, there remains a creative tension between the endless iteration of community merges—which can sometimes lead to "inbred" models that are mixes of mixes 47—and the fundamental need for new models trained from scratch on novel, high-quality datasets. While merging is a powerful tool for exploration and refinement, true leaps in capability will always depend on the foundational work of training new, powerful base models. The ultimate goal remains a future where the "skills" and "styles" of generative models can be treated as truly modular, compatible components, allowing creators to assemble custom models with the ease and predictability of a science rather than the trial-and-error of a craft.

#### Cytowane prace

1. Deep Dive into Stability AI's Generative Models - Stable Diffusion XL Part I - GitHub, otwierano: lipca 14, 2025, <https://github.com/neobundy/Deep-Dive-Into-AI-With-MLX-PyTorch/blob/main/deep-dives/008-stable-diffusion-sdxl/README.md>
2. Deep Dive into Stability AI's Generative Models - Stable ... - GitHub, otwierano: lipca 14, 2025, <https://github.com/neobundy/Deep-Dive-Into-AI-With-MLX-PyTorch/blob/main/deep-dives/008-stable-diffusion-sdxl/README2.md>
3. The SDXL Model Pipeline, otwierano: lipca 14, 2025, <https://www.xta0.me/2025/01/20/GenAI-Stable-Diffusion-SDXL.html>
4. Stable Diffusion XL: Everything You Need to Know - Magai, otwierano: lipca 14, 2025, <https://magai.co/stable-diffusion-xl-1-0/>
5. Best settings for Stable Diffusion SDXL 1.0 - Generative AI Blog by Segmind, otwierano: lipca 14, 2025, <https://blog.segmind.com/sdxl-1-0-settings-guide/>
6. A First Timer's Guide to Cooking Up a Fine Tuned SDXL Model - Rob Laughter, otwierano: lipca 14, 2025, <https://roblaughter.medium.com/a-first-timers-guide-to-cooking-up-a-fine-tuned-sdxl-model-446652d018f4>
7. Checkpoint merge blocks : r/comfyui - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/comfyui/comments/1b0xdwf/checkpoint_merge_blocks/>
8. Ultimate Guide to Stable Diffusion's SDXL Model (2024) | Shakker AI, otwierano: lipca 14, 2025, <https://wiki.shakker.ai/en/stable-diffusion-sdxl-guide>
9. Adventures with Stable Diffusion XL | by Hitoruna | Jul, 2025 - Medium, otwierano: lipca 14, 2025, <https://medium.com/@hitorunajp/adventures-with-stable-diffusion-xl-7617fdb80f43>
10. What is Illustrious/IllustriousXL? : r/StableDiffusion - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/1jjubsr/what_is_illustriousillustriousxl/>
11. Illustrious Model Series Guide created with SeaArt AI, otwierano: lipca 14, 2025, <https://www.seaart.ai/articleDetail/ctuib45e878c73ekrc90>
12. Illustrious XL-Modular WF v1.0 - MimicPC, otwierano: lipca 14, 2025, <https://www.mimicpc.com/workflows/illustrious-xl-modular-wf-v10>
13. Comprehensive Guide of Illustrious XL - Tensor.Art, otwierano: lipca 14, 2025, <https://tensor.art/articles/831123524065191393>
14. illustrious-xl-v1.0 - PromptLayer, otwierano: lipca 14, 2025, <https://www.promptlayer.com/models/illustrious-xl-v10>
15. Illustrious XL - v1.0 - v1.0 | Stable Diffusion Model - CHECKPOINT | Tensor.Art, otwierano: lipca 14, 2025, <https://tensor.art/models/828830887698374324>
16. NOOBAI XL | SeaArt Guide, otwierano: lipca 14, 2025, <https://docs.seaart.ai/guide-1/6-permanent-events/high-quality-models-recommendation/noobai-xl>
17. Noobai XL 1.0 · Models - Dataloop, otwierano: lipca 14, 2025, <https://dataloop.ai/library/model/laxhar_noobai-xl-10/>
18. NoobAI-XL (NAI-XL) - Epsilon-pred 1.0-Version - Civitai, otwierano: lipca 14, 2025, <https://civitai.com/models/833294?modelVersionId=1022833>
19. WAI-NSFW-illustrious-SDXL VS NoobAI-XL (NAI-XL) which one is better? - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/comfyui/comments/1j45dqm/wainsfwillustrioussdxl_vs_noobaixl_naixl_which/>
20. Base Models | Krita AI Handbook - Interstice.cloud, otwierano: lipca 14, 2025, <https://docs.interstice.cloud/base-models/>
21. NoobAI-XL ControlNet - eps-canny - Civitai, otwierano: lipca 14, 2025, <https://civitai.com/models/929685?modelVersionId=1040650>
22. Noobai XL Vpred 0.75 Cyberfix · Models - Dataloop, otwierano: lipca 14, 2025, <https://dataloop.ai/library/model/panchovix_noobai-xl-vpred-075-cyberfix/>
23. JANKU v4 NSFW Trained + NoobAI EPS + RouWei Illustrious XL ..., otwierano: lipca 14, 2025, <https://civitai.com/models/1277670/janku-v4-nsfw-trained-noobai-eps-rouwei-illustrious-xl>
24. text\_encoder\_2/config.json · John6666/rouwei-v080-epsilon-sdxl at main - Hugging Face, otwierano: lipca 14, 2025, <https://huggingface.co/John6666/rouwei-v080-epsilon-sdxl/blob/main/text_encoder_2/config.json>
25. WARNING: shape mismatch when trying to apply embedding, embedding will be ignored 768 1280 : r/StableDiffusion - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/15gpqw3/warning_shape_mismatch_when_trying_to_apply/>
26. Exception using ControlNet model control\_v11f1e\_sd15\_tile not compatible with SDXL models. · Issue #139 - GitHub, otwierano: lipca 14, 2025, <https://github.com/lllyasviel/ControlNet-v1-1-nightly/issues/139>
27. How to Merge Models in Stable Diffusion Checkpoint Merger - MimicPC, otwierano: lipca 14, 2025, <https://www.mimicpc.com/learn/how-to-merge-models-in-stable-diffusion-checkpoint-merger>
28. Merging four models at an even 25% with Comfy builtins returns no ..., otwierano: lipca 14, 2025, <https://www.reddit.com/r/comfyui/comments/1bxrcxt/merging_four_models_at_an_even_25_with_comfy/>
29. Merging Models for SDXL - Stable Diffusion Art, otwierano: lipca 14, 2025, <https://stable-diffusion-art.com/forums/topic/merging-models-for-sdxl/>
30. Merging SD in XL model : r/StableDiffusion - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/15roz9a/merging_sd_in_xl_model/>
31. [Bug]: Size mismatch for model.diffusion\_model.output\_block · Issue #5116 · AUTOMATIC1111/stable-diffusion-webui - GitHub, otwierano: lipca 14, 2025, <https://github.com/AUTOMATIC1111/stable-diffusion-webui/issues/5116>
32. Timeline for SDXL Controlnet? · lllyasviel ControlNet · Discussion #468 - GitHub, otwierano: lipca 14, 2025, <https://github.com/lllyasviel/ControlNet/discussions/468>
33. Why is SDXL inpainting so trash ? : r/StableDiffusion - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/1bh53d6/why_is_sdxl_inpainting_so_trash/>
34. [Bug]: Error merging checkpoints: The size of tensor a (4) must match the size of tensor b (9) at non-singleton dimension 1 · Issue #7472 · AUTOMATIC1111/stable-diffusion-webui - GitHub, otwierano: lipca 14, 2025, <https://github.com/AUTOMATIC1111/stable-diffusion-webui/issues/7472>
35. What's the best way to merge Pony and non-Pony models together ..., otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/1dzp7rg/whats_the_best_way_to_merge_pony_and_nonpony/>
36. Is It Possible to Merge Pony Model and Illustrious XL for Ultimate ..., otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/1ho8r08/is_it_possible_to_merge_pony_model_and/>
37. L7: Live Model Merge in ComfyUI - Comfy Academy - YouTube, otwierano: lipca 14, 2025, <https://www.youtube.com/watch?v=wcoopBBhUrw>
38. Model Merging Examples | ComfyUI\_examples - GitHub Pages, otwierano: lipca 14, 2025, <https://comfyanonymous.github.io/ComfyUI_examples/model_merging/>
39. Model Merge Blocks | SeaArt ComfyUI WIKI, otwierano: lipca 14, 2025, <https://docs.seaart.ai/seaart-comfyui-wiki/core-nodes/advanced/model-merging/model-merge-blocks>
40. Model Merge Blocks - ComfyUI Wiki, otwierano: lipca 14, 2025, <https://comfyui-wiki.com/en/comfyui-nodes/advanced/model-merging/model-merge-blocks>
41. ModelMergeSDXL - RunComfy, otwierano: lipca 14, 2025, <https://www.runcomfy.com/comfyui-nodes/ComfyUI/ModelMergeSDXL>
42. ComfyUI-DareMerge detailed guide - RunComfy, otwierano: lipca 14, 2025, <https://www.runcomfy.com/comfyui-nodes/ComfyUI-DareMerge>
43. 54rt1n/ComfyUI-DareMerge: ComfyUI powertools for SD1.5 ... - GitHub, otwierano: lipca 14, 2025, <https://github.com/54rt1n/ComfyUI-DareMerge>
44. ComfyUI Node: Model Merger (Block/DARE), otwierano: lipca 14, 2025, <https://www.runcomfy.com/comfyui-nodes/ComfyUI-DareMerge/DM_DareModelMergerBlock>
45. I've managed to merge two models with very different text encoder ..., otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/1g6500o/ive_managed_to_merge_two_models_with_very/>
46. Need help with model merging. : r/StableDiffusion - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/1bszk58/need_help_with_model_merging/>
47. What is the point of the endless model merges? : r/StableDiffusion - Reddit, otwierano: lipca 14, 2025, <https://www.reddit.com/r/StableDiffusion/comments/11kau9d/what_is_the_point_of_the_endless_model_merges/>