

UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

UNIVERSITY OF MODENA AND REGGIO EMILIA

"Enzo Ferrari" Department of Engineering

Master's Degree in Artificial Intelligence Engineering (LM-32)

Composing and Removing Concepts in Diffusion Models through Task Arithmetic

Supervisor:
Dr. Angelo Porrello
Co-supervisor:
Prof. Simone Calderara
Giacomo Salici

Candidate:
Alessandro Paludetto
Student ID 207362

ACADEMIC YEAR 2025/2026

Abstract

Stable Diffusion models can be adapted to specific visual concepts from a handful of examples through fine-tuning. Task arithmetic represents each adaptation as a *task vector*, the difference between the fine-tuned and the pre-trained parameters. Adding a single vector to the pre-trained checkpoint recovers the corresponding fine-tuned model; combining or subtracting vectors aims to compose or remove learned capabilities without further training. Prior work has studied task arithmetic in image classification and language tasks. Its behaviour when composing similar visual concepts in a text-to-image diffusion model remains less clear: independently learned concepts may interfere when their task vectors are combined, producing visual hybrids or causing one concept to suppress another.

This thesis aims to deepen our understanding of task arithmetic in text-to-image generation by addressing three questions about the *disentanglement* of generative task vectors. First, can vectors learned separately on *similar* classes of the same dataset be composed while preserving the generation quality of the individually fine-tuned models and keeping the classes distinct? Second, can subtracting a vector from a composed model selectively remove the corresponding generation capability while preserving those contributed by the remaining vectors? Third, how do the fine-tuning procedure and textual conditioning influence these properties?

To address these questions, the thesis fine-tunes a Stable Diffusion model on a dataset of closely related visual classes. The research first examines how different fine-tuning approaches affect the mergeability of the resulting task vectors. It then investigates the influence of different types of training prompts on this property. Each class vector is learned independently and evaluated through composition and subtraction.

Across the four selected class pairs, we find that requested-class recognition under averaged composition is higher with attention-only adaptation for the recipes using class-specific prompts, with and without identifiers. The same pattern is observed with generic wording and an identifier on all four evaluated class pairs.

Contents

Abstract **3**

1 Introduction **1**

1.1 Context and Motivation 1

1.2 Research Problem and Objectives 2

1.3 Approach and Scope 3

2 Background and Related Work **5**

2.1 Diffusion Models 5

2.1.1 Diffusion Probabilistic Models 6

2.1.2 Latent Diffusion Models 9

2.1.3 Stable Diffusion 13

2.1.4 Text Conditioning 15

2.2 Fine-Tuning Diffusion Models 18

2.2.1 Fine-Tuning and Personalisation 18

2.2.2 DreamBooth 20

3 Task Arithmetic **25**

3.1 Task Vectors 25

3.1.1 Difference between Base Parameters and Fine-Tuned Ones 25

3.1.2 Task Arithmetic 27

3.1.3 Composition, Negation and Scaling 29

3.2 Task Disentanglement 30

3.2.1 Definition 30

3.2.2 Why Disentanglement Is Required by Task Arithmetic 32

3.2.3 The Interference Problem between Tasks 34

3.2.4 Established Results and Open Questions 37

3.3 Task Arithmetic for Image Generation 39

3.3.1 Composing Concepts in Diffusion Models . . . 39
3.3.2 Machine Unlearning and Concept Erasure . . . 42
3.3.3 Removing Concepts from Diffusion Models . . . 43
3.3.4 Positioning of This Work . . . 45

4 Method **47**
4.1 Overview . . . 47
4.2 Experimental Questions and Comparisons . . . 48
4.3 Experimental Setup . . . 49
4.3.1 Pre-trained Model . . . 49
4.3.2 Dataset and Evaluated Classes . . . 50
4.4 Conditioning and Training Recipes . . . 52
4.5 Fine-Tuning Regimes and Task Vector Extraction . . . 53
4.5.1 Training Objective . . . 53
4.5.2 Trainable Parameters . . . 54
4.5.3 Optimisation Settings . . . 56
4.6 Weight-Space Configurations . . . 56
4.7 Evaluation Metrics . . . 58
4.7.1 Classifier-Based Evaluation . . . 58
4.7.2 LPIPS Control on Non-Traffic-Sign Prompts . . . 60
4.7.3 Interpreting Evidence for Task Disentanglement . . . 62
4.8 Evaluation Protocol . . . 62

5 Results and Discussion **65**
5.1 Selected Class Pairs and Aggregation . . . 65
5.2 Effect of Training Recipe on Composition . . . 68
5.2.1 Classic Fine-Tuning Without an Identifier . . . 68
5.2.2 DreamBooth-Style Training with Distinct Identifiers . . . 68
5.2.3 Classic Training with Distinct Identifiers . . . 70
5.2.4 Generic Wording with an Identifier . . . 70
5.3 Composition and Class Confusion . . . 70

5.4 Negative Vectors and Selective Control 74
5.5 Control Prompts Outside the Traffic-Sign Task 79
5.6 Discussion 82
5.7 Findings 83

6 Conclusions **85**

References **87**

List of Figures

1.1 Examples of visually similar GTSRB classes. 2

2.1 Forward and reverse diffusion processes. 7
2.2 Progressive image formation in DDPM sampling. 8
2.3 Rate–distortion trade-off in pixel-space diffusion. 10
2.4 Reconstruction fidelity under latent compression. 11
2.5 Latent diffusion model architecture. 13
2.6 Cross-attention in text-to-image generation. 16
2.7 Concept learning with Textual Inversion. 19
2.8 Subject preservation across DreamBooth contexts. 21
2.9 Subject adaptation with DreamBooth. 23

3.1 Task vectors and task arithmetic operations. 26
3.2 Weight disentanglement across task domains. 32
3.3 Interference in task-vector merging. 35
3.4 The three stages of TIES-Merging. 35
3.5 Full, linearised and attention-only fine-tuning. 38
3.6 Concept adaptation with Custom Diffusion. 40
3.7 Optimising a subject-and-style merge with ZipLoRA. 41
3.8 Concept erasure with Erased Stable Diffusion. 44

4.1 Experimental pipeline for task-vector evaluation. 49
4.2 Overview of the 43 GTSRB classes. 51
4.3 Trainable parameters in full and attention-only fine-tuning. 55
4.4 U-Net parameter counts by stage and parameter type. 55

5.1 Specialisation and AVG composition on $(20, 36)$. 67
5.2 AVG recognition by requested class, pair and recipe. 69
5.3 Classifier outcomes under AVG composition. 72

5.4 AVG generations across the four training recipes. 73
5.5 Classifier outcomes under negated-class prompts. 76
5.6 Matched subtraction directions on $(20, 36)$. 77
5.7 Composition and retained-task outputs for the speed-limit pair. 78
5.8 LPIPS control by prompt and single-vector edit sign. 80
5.9 Base and single-vector generations on LPIPS control prompts. 81

1. Introduction

1.1 Context and Motivation

Text-to-image diffusion models can generate varied images from natural-language prompts [41]. Adapting such a model to a specific visual concept, however, usually requires additional training. When several concepts are needed, training a separate model for each one is straightforward, but storing and using a different checkpoint for every combination is less practical.

Task arithmetic offers a way to reuse these adaptations. A *task vector* is the difference between the parameters of a fine-tuned model and those of its pre-trained base model [20]. Adding a vector to the base parameters recovers the corresponding capability. Adding or subtracting several vectors may then allow concepts learned independently to be composed or removed without further training. Whether these operations preserve the intended concepts depends on how much the vectors are entangled with one another.

This question is especially relevant for visually similar concepts. Traffic signs, for example, often share their shape and colour while differing in a small symbol or number. An image generator must reproduce those details to depict the correct class (Figure 1.1). The German Traffic Sign Recognition Benchmark (GTSRB) contains 43 such classes [50] and provides the setting for this thesis.



Figure 1.1: Examples of visually similar GTSRB classes. Speed-limit signs (top) share shape, colour and layout and differ only in the numeral; danger signs (bottom) share the red triangular frame and differ only in the pictogram. Images from the GTSRB training split [50].

1.2 Research Problem and Objectives

This thesis investigates whether task vectors learned separately for GTSRB classes can be used to edit a text-to-image diffusion model through addition and subtraction. It addresses three questions:

1. Can vectors learned separately for visually similar classes of the same dataset be composed while preserving the generation quality of the individually fine-tuned models and keeping the classes distinct?

2. Can subtracting a vector from a composed model selectively remove its class while preserving the generation capabilities contributed by the remaining vectors?

3. How do the fine-tuning procedure and textual conditioning affect these properties of task-vector composition and subtraction?

Weight disentanglement motivates the comparison between full U-Net and attention-only fine-tuning. Ortiz-Jimenez et al. [33] show that task arithmetic benefits when parameter updates act on distinct regions of a model's input space. In image-classification transformers, Jin et al. [22] find that fine-tuning only attention modules improves this property and reduces interference between task vectors. This thesis tests whether restricting updates to the attention projections of Stable Diffusion's U-Net has a similar effect for vectors learned from visually similar traffic-sign classes.

1.3 Approach and Scope

The experiments use pre-trained Stable Diffusion 1.5 and classes selected from GTSRB. Each evaluated class is fine-tuned independently, starting from the same base checkpoint. The difference between each fine-tuned checkpoint and the base defines a class-specific task vector. The study compares vectors obtained by fine-tuning the full U-Net with vectors obtained by updating only its attention projections. It also varies the training loss and prompts. The final comparison averages results over four selected class pairs. A ResNet classifier trained on all 43 GTSRB classes assesses whether generations match the requested class. Chapter 5 reports the arithmetic results by training recipe and pair.

A complementary experiment uses five non-traffic-sign prompts and 100 shared seeds per prompt, and computes LPIPS between corresponding base and single-vector edited outputs. This measures perceptual changes caused by the task vectors and, together with class recognition, provides empirical evidence for assessing task disentanglement. Generated examples help interpret both measurements. The same arithmetic operations are used across training conditions, so that the comparison focuses on the vectors produced by those conditions. The experimental results and the conclusions supported by them are presented in the later chapters.

2. Background and Related Work

This chapter provides the technical background on diffusion models and fine-tuning required by the remainder of the thesis. Section 2.1 reviews diffusion probabilistic models and their latent-space formulation, which constitute the generative backbone adopted in this work. Section 2.2 describes how such models are adapted to a small set of images. Chapter 3 develops the task-vector framework and positions this study within the related literature.

2.1 Diffusion Models

Deep generative modelling addresses the problem of approximating an unknown data distribution q from a finite set of samples, so that new samples can be drawn from the learned approximation p_θ. Until recently the field was dominated by Generative Adversarial Networks (GANs) [11], which achieve high perceptual quality at the cost of an unstable min-max optimisation and a well-documented tendency towards mode collapse. Diffusion models have since displaced GANs as the state of the art in image synthesis [4], offering a tractable denoising training objective, strong distribution coverage and a sampling procedure that is naturally amenable to conditioning and control.

For the present study, learning a distribution is more demanding than producing one convincing example. A model specialised on a traffic-sign class should retain the identifying numeral or pictogram across different backgrounds, viewpoints and random seeds. Conversely, a visually realistic sign can belong to the wrong class. This distinction between appearance and semantic identity motivates the use of both recognition scores and matched image comparisons later in the thesis. The background developed here explains how the same generative architecture can support those different forms of variability and why a change to its parameters can affect more than the intended concept.

2.1.1 Diffusion Probabilistic Models

Diffusion probabilistic models [47, 17] define generation as the time reversal of a fixed stochastic corruption process. Let $\mathbf{x}_0 \sim q$ denote a data sample. The *forward process* is a Markov chain of T steps that progressively injects isotropic Gaussian noise according to a variance schedule $\{\beta_t\}_{t=1}^{T}$, with $\beta_t \in (0, 1)$:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\, \beta_t \mathbf{I}\right), \qquad \mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (2.1)$$

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}). \quad (2.2)$$

A defining property of this formulation is that the marginal at an arbitrary step t admits a closed form, which allows sampling $\mathbf{x}_t$ directly from $\mathbf{x}_0$ without simulating the intermediate states. Defining $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, one obtains

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\, (1-\bar{\alpha}_t)\mathbf{I}\right), \qquad \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad (2.3)$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The schedule is chosen so that $\bar{\alpha}_T \approx 0$, hence $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$: all information about the datum is destroyed and the terminal state is a tractable prior.

The forward process has no learned parameters: its purpose is to construct related denoising problems at different difficulty levels. At a small timestep, the corrupted image still supplies most of the information needed to recover its structure. At a large timestep, many plausible clean images are compatible with the same noisy observation. The network must then rely more strongly on the distribution learned from training data and, in a conditional model, on the information supplied by the prompt. Generation therefore requires a prior over plausible images rather than a filter that merely removes visible noise.

A useful way to read Eq. (2.3) is through the signal-to-noise ratio

$$\mathrm{SNR}(t) = \frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}. \quad (2.4)$$

For example, $\bar{\alpha}_t = 0.9$ gives a ratio of 9, whereas $\bar{\alpha}_t = 0.1$ gives a ratio of approximately 0.11. These values describe the relative coefficients of signal and noise, rather than the recognisability of a particular image. Nevertheless, they clarify why the schedule determines which reconstruction problems the model encounters. Alternative schedules redistribute the training emphasis across

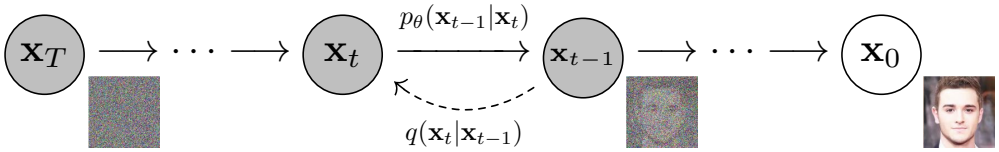


Figure 2.1: Graphical model of a diffusion probabilistic model. The forward process $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ progressively corrupts a data sample $\mathbf{x}_0$ into Gaussian noise $\mathbf{x}_T$, while the learned reverse process $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ inverts the corruption step by step. Figure reproduced from [17].

noise levels [30]. In a fine-tuning comparison, retaining the same schedule keeps that distribution of problems fixed while the trainable parameters or conditioning recipe are changed.

Generation reverses this trajectory (Figure 2.1). Since $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is intractable, it is approximated by a parametric Gaussian whose mean is predicted by a neural network with parameters θ:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\big(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\, \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\big)\,. \tag{2.5}$$

The Gaussian form is justified in the limit of small β_t, where the true reverse conditional is itself approximately Gaussian. Training maximises a variational lower bound on the data log-likelihood; Ho et al. [17] show that, under a fixed variance $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ and a specific reparameterisation of the mean, its denoising terms become weighted noise-prediction errors. Their simplified objective omits those timestep-dependent weights and trains the network $\boldsymbol{\epsilon}_\theta$ to predict the noise that was added:

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{\mathbf{x}_0,\, \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\, t\sim\mathcal{U}\{1,T\}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\Big(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},\, t\Big)\right\|_2^2\right]. \tag{2.6}$$

Equation (2.6) is a regression problem: it is stable, admits unbiased stochastic gradients and requires no adversarial critic. This simplicity is the practical reason behind the scalability of diffusion models.

One training example does not require unrolling the full reverse chain. An image, a timestep and a noise tensor are sampled, the noisy input is constructed directly, and one network evaluation predicts the sampled noise. The target is known because the corruption was generated by the training procedure itself. Repeated exposure to different timesteps teaches a single shared network to operate throughout the trajectory. The timestep embedding distinguishes these problems; without it, the same visual observation could require different corrections depending on its noise level.

This also explains a limitation of interpreting the training loss. A smaller average noise-



Figure 2.2: Progressive image formation in an unconditional DDPM on CIFAR-10. Each row follows one sample, with successive estimates of the clean image shown from left to right during reverse diffusion. Structure and recognisable content emerge gradually. Reproduced from Figure 6 of [17].

prediction error does not directly establish that a model draws the correct traffic-sign numeral or preserves diversity. The loss aggregates many coordinates and noise levels, whereas class identity may depend on a small part of the final image. Fine-tuning optimises a surrogate for generation, and the semantic effects of the resulting update must be evaluated separately.

Figure 2.2 gives a visual interpretation of the reverse trajectory. The clean-image estimates become increasingly structured as sampling proceeds: broad appearance is established before fine detail. The illustration shows intermediate estimates of the final image, rather than simply displaying the noisy inputs supplied to the denoiser.

The noise-prediction network is closely related to the score function of the perturbed distribution, since $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) \approx -\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)/\sqrt{1-\bar{\alpha}_t}$. This identity connects denoising diffusion to score-based generative modelling through stochastic differential equations [49], and establishes that the discrete formulation of Eq. (2.2) is the discretisation of a continuous-time diffusion process. The connection is relevant to this thesis because it clarifies *what* the parameters θ encode: a vector field over the data space. Editing θ, as done in Chapter 4, therefore amounts to editing this field.

Architecturally, $\boldsymbol{\epsilon}_\theta$ is implemented as a time-conditional U-Net [42]: an encoder–decoder convolutional backbone with skip connections, augmented with self-attention blocks at the lower-resolution stages and with a sinusoidal embedding of the timestep t injected into every residual block. The multi-scale structure is functionally meaningful: different resolutions provide different receptive fields and routes for exchanging global and local information. Coarse features can support layout and fine features can support detail, but semantic content is not confined to a single stage.

This distinction matters when analysing where an adaptation can change the denoising computation.

The main limitation of the original formulation is inference cost: sampling requires T sequential network evaluations, with T typically on the order of 10^3. Denoising Diffusion Implicit Models (DDIM) [48] address this by constructing a family of non-Markovian forward processes that share the marginals of Eq. (2.3), and therefore the same training objective. The zero-noise variant is deterministic and can be evaluated on a subsequence of timesteps, reducing the number of function evaluations by one to two orders of magnitude. Determinism does not by itself guarantee exact invertibility of a finite-step sampler. Its relevant property here is that fixed initial noise and sampling settings define a reproducible trajectory for a fixed model.

Changing the weights while retaining that initial noise supplies a controlled comparison: differences between two outputs can be attributed to the edited generator under the shared sampling procedure, subject to numerical execution differences. The images need not retain exactly the same layout because the edited predictions change subsequent inputs to the denoiser. A task vector can thus have a modest effect at an individual step and a substantial cumulative effect on the final image. The distinction between one denoising prediction and the complete sampling trajectory will also be relevant to task disentanglement. Because the same weights are reused at every timestep, a fine-tuning update is not an edit to one isolated transition. It changes the predictor across the noise levels encountered during sampling. Its practical effect therefore depends on both the learned prediction changes and the sequence of inputs on which those changes are applied.

2.1.2 Latent Diffusion Models

Operating the diffusion process directly in pixel space is computationally prohibitive at high resolution: the cost of each of the T denoising steps scales with the number of pixels, and a substantial fraction of the model capacity is spent modelling imperceptible high-frequency detail. Latent Diffusion Models (LDMs) [41] resolve this inefficiency by separating *perceptual compression* from *semantic generation* (Figure 2.3).

In a first stage, an autoencoder is trained to map images into a lower-dimensional latent space. An encoder $\mathcal{E}$ maps $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$ to $\mathbf{z} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{h\times w\times c}$ with $h = H/f$ and $w = W/f$, and a decoder $\mathcal{D}$ reconstructs $\tilde{\mathbf{x}} = \mathcal{D}(\mathbf{z}) \approx \mathbf{x}$. The downsampling factor f is typically 8. Training combines a pixel-wise reconstruction term with a perceptual loss [55] and a patch-based adversarial term [5],

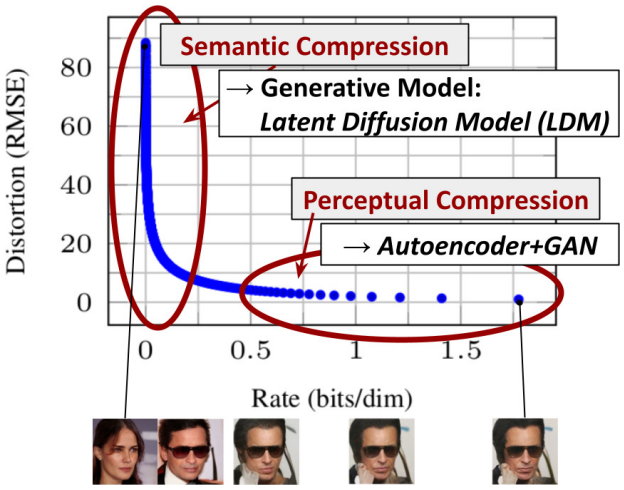


Figure 2.3: Rate–distortion trade-off of a trained pixel-space diffusion model. Most bits of an image encode imperceptible detail (perceptual compression), while the semantic content is concentrated at low rates (semantic compression). Latent diffusion delegates the former to an autoencoder and the latter to the diffusion model. Figure reproduced from [41].

which together preserve local realism that an ℓ_2 objective alone would blur. To prevent the latent space from attaining arbitrarily high variance, the autoencoder is regularised either by a slight Kullback–Leibler penalty towards a standard normal prior, as in a variational autoencoder [23], or by vector quantisation [32].

Compression changes the object being modelled. A latent coordinate is not a smaller version of an RGB pixel: its meaning depends on the learned encoder and decoder, and information is distributed across channels and spatial locations. The denoiser learns which latent configurations decode into plausible images, while the decoder supplies the mapping back to visible colours and textures. The separation is useful because the expensive iterative computation operates on the compact representation and decoding is required only after sampling. It does not imply that the latent variables form independent semantic factors or that one channel corresponds to a particular object category.

The compression factor introduces a trade-off. A larger factor reduces the spatial size of the diffusion problem, but the reconstruction must preserve the features relevant to the intended application. In traffic signs, two classes can share an outline and differ only in a small numeral or pictogram. Such a detail may occupy little image area while determining the entire class label.



Figure 2.4: Reconstruction comparison from the latent-diffusion literature. Input images are followed by reconstructions using the authors' autoencoder ($f = 4$), a DALL-E autoencoder ($f = 8$), and VQGAN ($f = 16$). The enlarged regions expose differences in retained detail. These are published autoencoder examples, rather than outputs of the thesis experiments. Reproduced from Figure 1 of [41].

Consequently, an autoencoder that preserves general perceptual realism can still alter a feature important for fine-grained recognition. This is a conceptual limitation of the representation, not evidence that it caused a specific failure in the experiments reported here.

The reconstruction examples in Figure 2.4 make this trade-off visible. The original image and enlarged details can be compared with the decoded representations. The columns use different autoencoding methods as well as different compression factors, so the visual differences should not be attributed to the factor f alone. Their relevance here is that reconstruction fidelity limits which details are available to the subsequent latent-space generator.

In a second stage, the diffusion process of Section 2.1.1 is defined over the latent variable $\mathbf{z}$ rather than over $\mathbf{x}$. The training objective becomes

$$\mathcal{L}_{\text{LDM}}(\theta) = \mathbb{E}_{\mathcal{E}(\mathbf{x}),\boldsymbol{\epsilon},t}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \mathbf{c})\|_2^2\right], \tag{2.7}$$

where $\mathbf{c}$ denotes an optional conditioning signal, discussed in Section 2.1.4. With $f = 8$ the spatial resolution of the denoising problem is reduced by a factor of 64. This substantially reduces

computational cost, although the actual saving also depends on channel counts, network architecture and the autoencoder. Reconstruction quality is an empirical property of the learned decoder: information lost during encoding cannot be recovered exactly merely by decoding.

Training and generation use this representation in different ways. During training, real images are encoded and their latents are corrupted according to the selected noise level. During text-to-image generation, there is no reference image to encode: the sampler starts from latent noise, repeatedly predicts a correction, and decodes its final latent. The encoder therefore provides training targets, whereas the decoder remains part of every generated output. Fine-tuning the denoiser adjusts the latent distribution it produces without changing the reconstruction rule applied to that distribution.

Holding the autoencoder fixed gives the comparisons in this thesis a shared output interface. If it were independently adapted for each class, differences between generated signs could arise from both denoising and decoding, and arithmetic between U-Net updates would no longer describe the complete model change. With a shared decoder, the parameter intervention is easier to define: the same latent tensor would decode identically in the base and edited configurations. Different images arise because the edited denoiser generates different latent tensors.

Two consequences of this design matter for the present work. First, the autoencoder is trained once and kept frozen: every adaptation and every weight-space edit considered in this thesis acts exclusively on the parameters θ of the latent denoiser, so the space in which task vectors are defined is well delimited. Second, the compact latent representation makes repeated fine-tuning and generation experiments computationally practical. Neither compression nor a shared decoder guarantees that denoiser updates will be disentangled; that property must still be assessed from their behaviour.

It is helpful to distinguish latent-space operations from weight-space operations at this point. Modifying a sampled latent changes one image or one trajectory. Modifying the denoiser parameters changes the function used across prompts and random seeds. The task vectors studied in this thesis belong to the second category. Their selectivity concerns how broadly the new function differs from the original, rather than whether two particular latent codes can be combined smoothly. A shared latent convention is also required: scaling the encoded tensors or changing the encoder would change the inputs for which the denoiser was trained. Matching tensor dimensions alone would not preserve that relationship. Retaining the same surrounding components keeps the reference

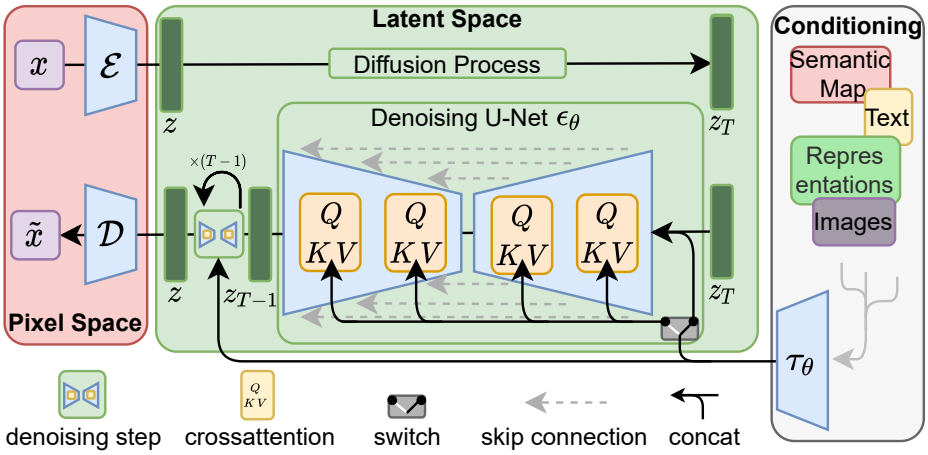


Figure 2.5: Architecture of a latent diffusion model. The encoder $\mathcal{E}$ and decoder $\mathcal{D}$ map between pixel space and latent space; the denoising U-Net $\boldsymbol{\epsilon}_\theta$ operates on the latent $\mathbf{z}_t$ and receives the conditioning, encoded by τ_θ, through cross-attention layers. Figure reproduced from [41].

representation fixed across all the task-vector experiments.

2.1.3 Stable Diffusion

Stable Diffusion is the concrete, open-weight instantiation of the latent diffusion framework used throughout this thesis. It comprises three components (Figure 2.5): (i) a convolutional VAE with downsampling factor $f = 8$ and $c = 4$ latent channels, which maps a $512 \times 512 \times 3$ image to a $64 \times 64 \times 4$ latent tensor; (ii) a frozen text encoder derived from CLIP [37], which converts a prompt into a sequence of token embeddings; and (iii) a conditional U-Net of roughly 860M parameters, which performs the denoising in latent space. Its pre-training uses large collections of image–text pairs derived from LAION [45]; the exact subsets and training stages depend on the checkpoint.

The U-Net alternates residual convolutional blocks with transformer blocks [51]. Each transformer block contains a self-attention layer, which models spatial dependencies within the latent map, and a cross-attention layer, which injects the textual condition. The attention projections are therefore the focus of the attention-only adaptation studied in this thesis, which compares vectors confined to these layers with vectors spanning the whole U-Net.

These components play different roles in the generation pipeline. The text encoder interprets the prompt before denoising starts. The U-Net repeatedly combines that interpretation with the

current noisy latent and timestep. The decoder finally translates the sampled latent into an image. A successful adaptation need not modify all three components: it can instead teach the U-Net to associate an existing text representation with a more specific visual distribution. This is the setting adopted here. The text representations and decoder remain shared, while the denoising function changes independently for each traffic-sign class.

The distinction also prevents an ambiguity in the word "model". A complete Stable Diffusion checkpoint contains more than the parameters used in a task vector. In this thesis, arithmetic concerns the U-Net, and an edited generator consists of its modified parameters together with the unchanged surrounding components. The resulting image depends on all of them, but the experimental intervention is confined to one. This makes it possible to compare adaptation regimes without attributing text-encoder or autoencoder changes to the task vector.

Within the U-Net, convolutional residual blocks and attention blocks provide complementary operations. Convolutions combine nearby spatial features with shared filters. Self-attention lets a position use information from other positions, while cross-attention lets it use the textual condition. Their outputs are integrated into a common computation through residual connections. Consequently, an update to an attention projection can influence downstream convolutional activations even though the convolutional weights are frozen. Restricting the trainable weights therefore restricts where an intervention is introduced, rather than isolating an independent subsystem whose effects stay inside the attention block.

For traffic signs, this interaction can be understood through a simple example. The shared backbone already has machinery for drawing edges, coloured surfaces and photographic backgrounds. Adaptation for a particular sign may reuse that machinery while changing which shape or numeral is selected by its prompt. Alternatively, a full update can change both selection and rendering. Neither case proves how the trained model actually stores a concept, but the example explains why comparing full and attention-only adaptation is informative: the regimes allow different routes for representing the same training target. They may therefore differ in both specialisation and collateral changes.

Later releases scale the same recipe: SDXL [36] enlarges the U-Net, employs two concatenated text encoders and adds explicit conditioning on image size and crop coordinates, while more recent variants replace the U-Net with a transformer backbone operating on rectified-flow trajec-

tories [6]. The methodological contribution of this thesis is architecture-agnostic, as it operates on parameter differences rather than on a specific topology; Stable Diffusion is adopted because its open weights and its widely reproduced fine-tuning pipelines make the experimental protocol verifiable. Architecture-agnostic arithmetic should not be confused with architecture-independent results: parameters from different backbones cannot be added coordinate by coordinate, and successful edits in Stable Diffusion 1.5 do not establish the same behaviour in later systems. Even within one architecture, the pre-training checkpoint supplies the reference function around which adaptation occurs. A different base may recognise the same prompt or support the same visual class differently. The chosen backbone and frozen components therefore form part of the scope of the conclusions. Sampling settings complete that specification. The number of denoising steps, scheduler and guidance scale influence the trajectory evaluated with a given U-Net. A checkpoint comparison therefore identifies both the weights and the procedure used to generate their outputs; recording only the model name would leave several sources of variation unresolved.

2.1.4 Text Conditioning

Conditioning is what turns an unconditional sampler into a controllable one. In Stable Diffusion a prompt y is tokenised and encoded by the frozen text encoder τ_ϕ into a sequence of contextual embeddings $\mathbf{c} = \tau_\phi(y) \in \mathbb{R}^{L \times d_\tau}$, with L the maximum token length. These embeddings are consumed by the cross-attention layers of the U-Net, where the queries are computed from the intermediate visual representation $\varphi(\mathbf{z}_t)$ and the keys and values from the text (Figure 2.6):

$$\mathbf{Q} = \varphi(\mathbf{z}_t)\mathbf{W}_Q, \qquad \mathbf{K} = \mathbf{c}\mathbf{W}_K, \qquad \mathbf{V} = \mathbf{c}\mathbf{W}_V, \tag{2.8}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2.9}$$

The attention map $\text{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{d_k})$ assigns to each spatial location a distribution over tokens, and therefore constitutes an explicit, inspectable interface between linguistic and visual content — a property exploited by a large body of training-free editing methods [14]. The projections $\mathbf{W}_K$ and $\mathbf{W}_V$ are precisely the weights that bind a token to its visual realisation, which explains why modifying them is sufficient to add, remove or replace a concept in some editing and customisation settings [25, 10].

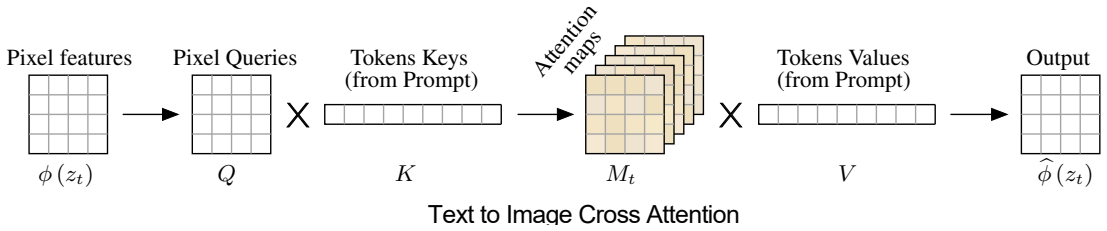


Figure 2.6: Text-to-image cross-attention. Queries Q are projected from the spatial features $\varphi(\mathbf{z}_t)$, keys K and values V from the prompt tokens; the resulting attention maps M_t assign each spatial location a distribution over tokens. Figure adapted from [14].

With spatial features represented by rows, $\varphi(\mathbf{z}_t)$ has N rows, one for each latent position at the relevant resolution. The text sequence has L rows, one for each token. The projected matrices therefore have shapes $N \times d_k$, $L \times d_k$ and $L \times d_v$ for queries, keys and values respectively. The product $\mathbf{QK}^\top$ contains one compatibility score for every spatial-position–token pair. Applying the softmax across tokens produces a weighted combination of textual values at each position. Multi-head attention performs several such operations in parallel and combines their outputs [51].

This description distinguishes two ways in which fine-tuning can change the use of text. Updating keys or queries can change the compatibility scores and therefore where the condition is attended to. Updating values or the output projection can change the information passed to the visual stream after those scores are computed. These roles are intertwined across layers and timesteps. An attention map can help locate a relationship between a word and an image region, but it is not a complete record of that word's causal influence on the final output. The surrounding network continues processing the attended features after the map has been formed.

Tokenisation adds a further distinction between a written identifier and the conditioning received by the model. A short identifier string can be split into several existing tokens; it need not correspond to a newly created vocabulary entry. With the text encoder frozen, fine-tuning can still associate the resulting sequence with a target class by changing the U-Net. Different identifiers may help distinguish the sequences, but their usefulness depends on the learned association and cannot be inferred solely from the fact that their spellings differ. This is why the method chapter reports both the identifier and the rest of the prompt for each recipe.

Conditioning alone does not guarantee that samples adhere to the prompt. Classifier guidance [4] improves adherence by steering the sampling trajectory with the gradient of an auxiliary classifier,

at the cost of training such a classifier on noisy inputs. Classifier-free guidance [18] removes this dependency: a single network is trained jointly on the conditional and the unconditional tasks by randomly replacing $\mathbf{c}$ with a null embedding $\varnothing$ during training, and at inference the two predictions are extrapolated,

$$\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, t, \mathbf{c}) = \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \varnothing) + w\left[\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \mathbf{c}) - \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \varnothing)\right], \tag{2.10}$$

where $w > 1$ is the guidance scale and trades sample diversity for prompt fidelity. Equation (2.10) deserves emphasis beyond its practical role: it is a linear combination of model outputs used to amplify a semantic direction. Task arithmetic, introduced in Section 3.1, applies the same principle one level lower, in parameter space rather than in output space.

Under the convention of Eq. (2.10), $w = 0$ uses the unconditional prediction and $w = 1$ uses the conditional prediction. Values greater than one extrapolate beyond it. The guidance coefficient controls inference, whereas the coefficient of a task vector controls the parameters used to make both predictions. Increasing one is therefore not equivalent to increasing the other. An edited model may change its conditional and unconditional branches by different amounts, and guidance can amplify their difference. Comparisons between task vectors consequently retain the same guidance setting so that this additional source of variation is controlled.

Prompt wording likewise remains part of the experimental input. A detailed class description can supply information already represented during pre-training, while a generic description with an identifier places more of the burden on the newly learned association. A difference between these prompts can change the base generator before any arithmetic is applied. Evaluating an edit against the base under the same prompt separates that conditioning effect from the effect of changing the weights. It also clarifies why a negative task-vector coefficient and a negative prompt are distinct interventions, even when both are intended to reduce the appearance of a concept. Closely related class descriptions also share much of their token sequence. The phrase “traffic sign” can contribute common conditioning, while a numeral or direction supplies the distinguishing information. Learning to use that distinction is central to class-selective adaptation, even when the overall text embeddings remain strongly related.

2.2 Fine-Tuning Diffusion Models

A text-to-image model pre-trained on web-scale data covers a broad but generic visual vocabulary. It may not reliably render a specific instance — a particular object, a fine-grained category, or a class that appears only sparsely in the pre-training corpus — from its ordinary textual description. Adapting the model to such concepts from a handful of images is the problem of personalisation.

2.2.1 Fine-Tuning and Personalisation

The most direct strategy is to continue optimising Eq. (2.7) on the target images. With a few examples and hundreds of millions of parameters, this naive approach can fail in three characteristic ways. It can *overfit*, collapsing the output distribution onto the training images and destroying pose and context variability. It can induce *catastrophic forgetting* [28, 8], degrading the prior acquired during pre-training. It can also produce *language drift* [43]: the meaning of the class token used in the captions shifts towards the specific instance, so that the model can no longer reliably generate other members of the same class.

These failures concern different aspects of adaptation. Memorisation is a failure to generalise from the training images to new instances or contexts. Forgetting is a loss of previously available behaviour. Language drift is a change in the relationship between a prompt and the distribution it requests. They can occur together, but one does not establish the others. For example, a model might reproduce its training sign accurately while ignoring requests for a new background, or preserve varied backgrounds while confusing the numeral that distinguishes two classes. A useful evaluation must therefore identify which property is being measured rather than treating low training loss as evidence that all three problems have been avoided.

The available images also determine what the update can learn. If every example places a sign against a similar background, the loss does not explicitly tell the network which visual features belong to the class and which are incidental. An adaptation can encode both. More examples can improve coverage of nuisance variation, but their number alone does not guarantee that the relevant factors are separated. This is especially pertinent when task vectors are trained independently: shared backgrounds, colours or outlines may contribute common directions to updates whose intended class labels are different. The later arithmetic then combines those directions along with

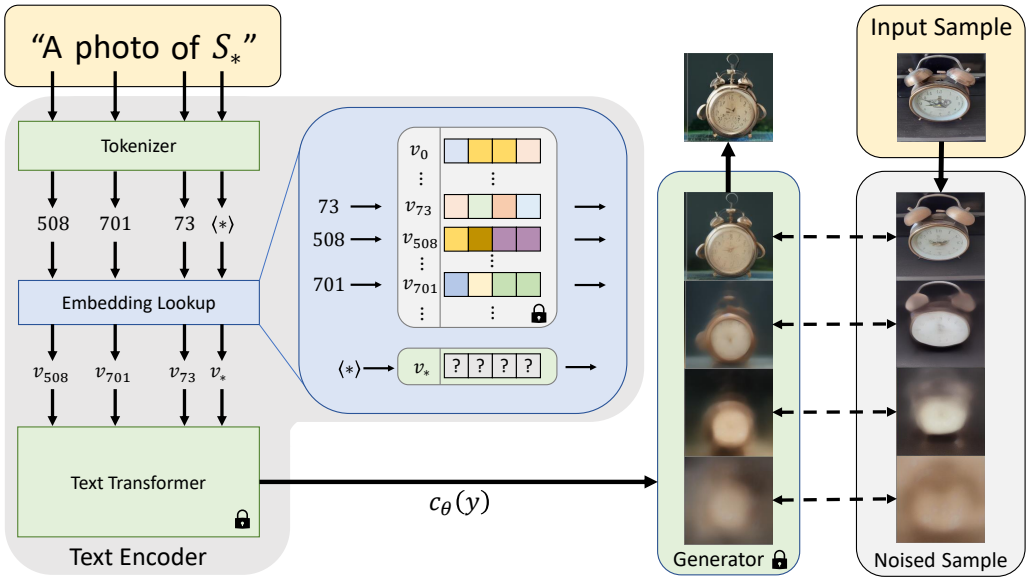


Figure 2.7: Textual Inversion. The placeholder token S_* is mapped to a new embedding v_*, which is the only quantity optimised; the text encoder and the generator remain frozen, and the embedding is learned by minimising the denoising loss on the few input images. Figure reproduced from [9].

the class-specific ones.

Two families of remedies exist. The first constrains *what* is optimised. Textual Inversion [9] freezes the entire model and learns only a new embedding vector $\mathbf{v}_*$ for a placeholder token S_*, optimising Eq. (2.7) with respect to $\mathbf{v}_*$ alone (Figure 2.7). The result is extremely compact — a few kilobytes — and cannot damage the prior by construction, but its expressive power is bounded by the range of the frozen conditioning interface, which can limit fidelity on concepts far from the pre-training distribution. The second family constrains *how* the weights are optimised through explicit regularisation (Section 2.2.2).

Restricting the trainable parameter set provides another way to control capacity. Full U-Net adaptation allows every denoising layer to change, while attention-only adaptation retains the surrounding weights and updates selected projections. This restriction can reduce the ways in which the model departs from its base function, but it can also limit the changes needed to represent a difficult class. Fewer trainable parameters do not guarantee either higher fidelity or greater disentanglement. The practical question is whether the restricted update remains effective for its own task while perturbing other tasks less. The experiments compare these outcomes directly.

The distinction between these families is not merely practical. Empirical evidence indicates that fine-tuning tends to remain within a limited region of weight space around the pre-trained solution [29], and that models fine-tuned from a shared initialisation frequently lie in a single low-loss basin, connected by paths of approximately constant loss [7, 52]. This regularity is the structural precondition that makes the weight-space operations of Section 3.1 meaningful rather than arbitrary.

That empirical motivation must be distinguished from a guarantee. A low-loss path on one objective does not show that every point along it preserves other capabilities, and findings for discriminative models need not carry over unchanged to a conditional generator. The learning rate, number of updates, regularisation and training captions all influence the endpoint reached by fine-tuning. Two vectors for the same class can consequently differ even when their final recognition scores are similar. From the perspective of this thesis, adaptation is therefore both a way to acquire a concept and the procedure that determines whether its parameter displacement can later be combined with another one. Checkpoint selection consequently determines the object later manipulated by arithmetic. An early checkpoint may underlearn the class, while a later one may encode unnecessary changes. The evaluated protocol uses recorded final checkpoints; their behaviour should be understood at that training duration rather than as the best possible endpoint for every recipe.

2.2.2 DreamBooth

DreamBooth [43] personalises the full denoiser while explicitly counteracting overfitting and language drift. The subject is bound to a rare token identifier $[V]$ — chosen among sequences with weak prior associations in the tokeniser — and the training captions take the form "a $[V]$ `[class noun]`". Anchoring the identifier to a class noun lets the model reuse the visual prior it already possesses for that class instead of learning the subject from scratch.

The identifier and class noun serve complementary purposes. The noun supplies a broad category already accessible to the pre-trained generator; the identifier distinguishes the subject from other members of that category. For a particular dog, preserving subject identity does not require the model to learn the entire category of dogs again. In the traffic-sign setting, the same idea is adapted to classes rather than individual physical objects. An identifier can refer to the visual identity of a



Figure 2.8: Subject-driven generation with DreamBooth. Reference photographs appear on the left, followed by generated scenes that retain the subject across different settings and activities. The examples illustrate personalisation of an individual subject, whereas this thesis adapts traffic-sign classes. Reproduced from Figure 1 of [43].

sign category across many photographs, while the description provides information about its shape or meaning. This changes the granularity of the target and should be kept separate from the original few-image subject-personalisation setting.

Figure 2.8 illustrates the intended generalisation: reference photographs establish the subject's identity, while different prompts request environments and activities absent from those photographs. Identity preservation and contextual variation must be assessed together; reproducing the reference pose alone would not meet this objective.

To preserve the class prior, the method introduces a *prior-preservation* term. Before fine-tuning, a set of images $\mathbf{x}^{\mathrm{pr}}$ of the generic class is sampled from the frozen model; during fine-tuning the network is required to keep reproducing them. The objective is

$$\mathcal{L}_{\mathrm{DB}}(\theta) = \mathbb{E}\big[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \mathbf{c})\|_2^2\big] + \lambda\, \mathbb{E}\Big[\big\|\boldsymbol{\epsilon}' - \boldsymbol{\epsilon}_\theta(\mathbf{z}_{t'}^{\mathrm{pr}}, t', \mathbf{c}^{\mathrm{pr}})\big\|_2^2\Big], \tag{2.11}$$

where $\mathbf{c}$ encodes the instance prompt, $\mathbf{c}^{\mathrm{pr}}$ the class prompt, and λ balances the two terms. The second term acts as a self-generated regulariser that encourages preservation of the class distribution while the identifier is specialised. Preservation is encouraged by this objective rather than guaranteed. The resulting training procedure is summarised in Figure 2.9.

Both terms in Eq. (2.11) remain denoising losses. The prior term does not force a generated image to be an exact copy of a stored reference; it trains predictions on noisy versions of the prior examples under their associated captions. Its influence depends on which examples are used, how

often they are sampled, and their weight relative to the instance term. The coefficient λ in this equation weights a training objective. It is distinct from the coefficient that will later scale a saved task vector at inference, even though both control the strength of an intervention at different stages.

Prior preservation also introduces a tension between retaining generic behaviour and acquiring a specialised association. A strong prior term can oppose changes useful for the target, while a weak term may provide little protection against drift. There is no reason to assume that one balance is best for all classes, parameter subsets or data regimes. For this thesis, the relevant question is broader than whether the single adapted model produces good signs: the endpoint selected by the combined loss becomes a task vector, and its behaviour after averaging or subtraction can differ from that of a vector trained with the instance term alone.

The exact origin of the prior images is part of that interpretation. The original DreamBooth procedure uses images generated by the base model. The DreamBooth-style recipes evaluated here instead draw a disjoint subset of real images of the same traffic-sign class, as specified in Section 4.4. They retain the two-term training idea but do not reproduce every detail of the original procedure. Calling them DreamBooth-style distinguishes this methodological relationship from a claim that the experimental configuration is identical to the published method.

DreamBooth attains high subject fidelity from three to five images, but it updates all the weights of the denoiser and therefore produces one full model copy per concept, of the order of several gigabytes. This is impractical when many concepts must be stored, and — more importantly for this thesis — it provides no mechanism for combining concepts that were learned separately.

Independent adaptation is nevertheless valuable because it avoids requiring all concepts to be available in one training run. Each specialised checkpoint can be compared with its common base to extract an update. The unresolved question is then whether the updates retain their intended roles when used together. Prior preservation constrains a model during its own training; it does not explicitly constrain its interaction with a second, independently learned update. This distinction leads to the task-arithmetic framework in the next chapter, where the objects of interest are the saved displacements and their effects under a common arithmetic rule. In particular, preserving a class prompt during one adaptation does not explicitly teach the network how to distinguish that class from a second adapted class after merging. The prior term constrains a single training problem, whereas composition introduces a relationship between the outcomes of two such problems.

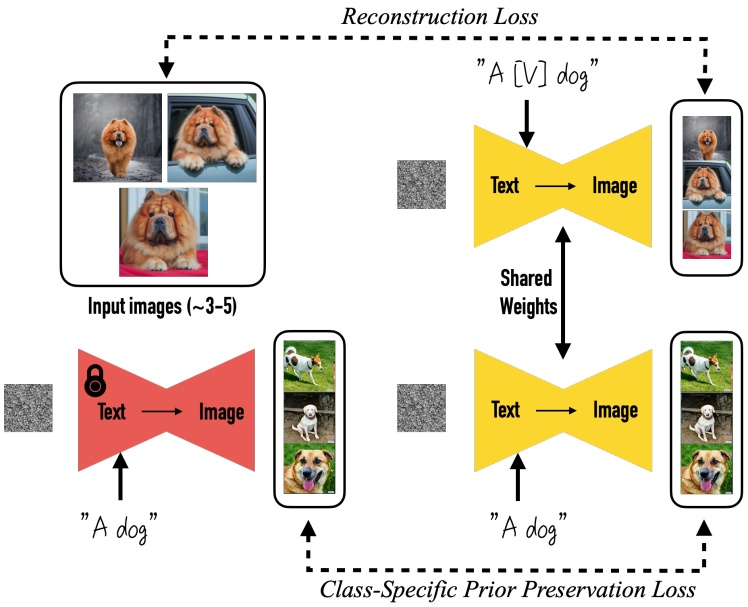


Figure 2.9: DreamBooth fine-tuning. The model is trained to reconstruct the subject images from prompts containing the identifier "a [V] dog" (reconstruction loss), while images of the generic class sampled from the frozen model are used to preserve the class prior (class-specific prior preservation loss). Figure reproduced from [43].

3. Task Arithmetic

This chapter presents the task-arithmetic framework and positions the study within the literature on composing and removing visual concepts. Section 3.1 introduces task vectors and the arithmetic operations used to combine independently acquired capabilities. Section 3.2 formalises task disentanglement and analyses the interference phenomena that limit weight-space composition. Finally, Section 3.3 reviews composition and concept removal in diffusion models and explains the positioning of this thesis.

3.1 Task Vectors

The methods of Section 2.2 produce specialised models. Task arithmetic reinterprets these models not as endpoints, but as *directions* in weight space that can be extracted, rescaled and recombined.

3.1.1 Difference between Base Parameters and Fine-Tuned Ones

Let $\theta_0 \in \mathbb{R}^n$ be the parameters of a pre-trained model and $\theta_t \in \mathbb{R}^n$ those obtained by fine-tuning it on a task t. The corresponding *task vector* is defined [20] as the element-wise difference

$$\boldsymbol{\tau}_t = \theta_t - \theta_0 \in \mathbb{R}^n. \tag{3.1}$$

The definition is trivial to compute and requires neither access to the training data nor any additional optimisation, yet it is not vacuous: it isolates *what fine-tuning changed*, discarding the shared component θ_0 (Figure 3.1a). The vector inherits the topology of the network, so each of its coordinates is attached to a specific layer and, in the case of a diffusion U-Net, to a specific resolution and attention role. This makes $\boldsymbol{\tau}_t$ an object that can be inspected structurally — by norm, sparsity, spectral content or layer distribution — and not merely applied.

The shared reference is essential. Two checkpoints can implement similar functions while representing them with different hidden-unit orderings or different parameterisations. Coordinate-wise subtraction would then compare quantities that do not have the same internal role. Starting each adaptation from the exact same checkpoint retains a common coordinate system: the entry

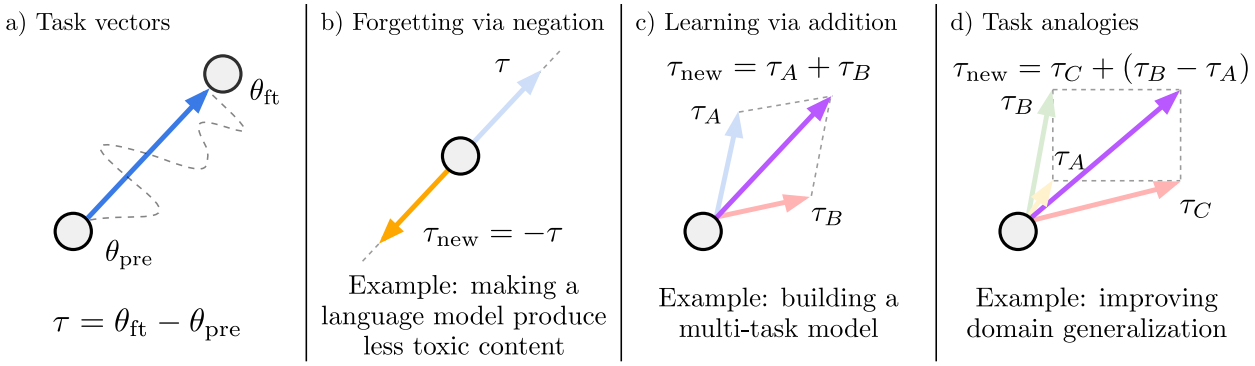


Figure 3.1: Task vectors and the arithmetic operations defined on them. (a) A task vector is the difference between fine-tuned and pre-trained weights. (b) Negating it suppresses the corresponding behaviour. (c) Adding several task vectors yields a multi-task model. (d) Task analogies combine vectors of related tasks. Figure reproduced from [20].

for a particular filter or attention projection has the same architectural location in every vector. A matching architecture alone is insufficient, because independently trained base models need not have aligned coordinates. The definition in Eq. (3.1) therefore includes a provenance requirement as well as an arithmetic operation.

Explicit transport extends reuse beyond a shared base: GradFix masks a source task vector using gradient signs from a few labelled target examples, without further fine-tuning [39]. Theseus instead transfers updates between models of different widths by aligning intermediate representations through orthogonal Procrustes analysis, without further training [40].

For a generator, the word “task” also needs a concrete interpretation. Here it refers to adapting the denoiser to images of one class under a specified conditioning recipe. The saved displacement records the outcome of that adaptation; it is not a symbolic representation containing only the class name. Background statistics, prompt associations and optimisation effects can all contribute to it. Two vectors trained for the same class under different recipes may therefore differ, while vectors trained for different classes may share components. This makes the training conditions part of the meaning of the vector and explains why the study compares recipes rather than treating all class-labelled displacements as interchangeable.

The reason this arithmetic is meaningful at all is that fine-tuned models originating from a common initialisation typically remain linearly mode connected [7, 29]: the loss along the segment

joining them does not encounter significant barriers. Under such conditions weight interpolation can retain useful behaviour. This is an empirical motivation for arithmetic, rather than proof that a low-loss region is closed under arbitrary sums or negative displacements.

Frozen parameters provide a useful structural special case. If a coordinate is never updated, its task-vector entry is exactly zero. Attention-only adaptation thus defines a displacement supported on a selected subset of U-Net coordinates, while full adaptation permits a broader support. Both can be represented in the same complete parameter space by retaining zeros outside the updated subset. Structural sparsity says where changes are allowed; it does not establish that the nonzero entries influence only one class. Their effects still propagate through the full network during generation.

The displacement also aggregates the whole optimisation run. It equals the sum of successive parameter changes, but is not generally proportional to the gradient of one example or one final loss evaluation. Optimiser state, sampled batches and the changing current weights influence those steps. Extracting the endpoint difference preserves their net result without retaining their history.

3.1.2 Task Arithmetic

Task arithmetic [20] builds an edited model by applying a linear combination of task vectors to the pre-trained weights:

$$\theta_{\text{edit}} = \theta_0 + \sum_{t=1}^{T} \lambda_t \boldsymbol{\tau}_t, \qquad \lambda_t \in \mathbb{R}. \tag{3.2}$$

The coefficients λ_t are usually selected on a held-out set and are frequently shared across tasks. The operation is closed under composition, reversible, and costs a single pass over the parameters; no gradient computation, no training data and no access to the original optimisation procedure are required. It thus provides an editing interface for models that are otherwise available only as frozen checkpoints.

The data-free property applies to constructing an edit from available vectors. Learning those vectors required training data, and selecting useful coefficients or evaluating the result can require further observations. Likewise, algebraic reversibility concerns weights: subtracting the exact displacement that was just added restores the original checkpoint, up to numerical precision. It does not establish that applying a negative vector to the base reverses the semantic learning process or reconstructs a model trained without the target data. Those are separate behavioural questions.

After the weights are combined, the edited model has the same architecture as the base. Each denoising prediction is produced by one set of parameters; the number of vector components used to construct it does not introduce additional network evaluations. This differs from an ensemble, in which several models are evaluated and their predictions combined. An ensemble mixes outputs of different functions, whereas weight arithmetic first constructs a new function. Because the network is nonlinear in its weights, the two procedures need not agree even when they use the same coefficients. This difference is particularly relevant to diffusion, where an output-space mixture may be recomputed at every sampling step.

The empirical finding that motivates the framework is that Eq. (3.2) approximately transfers *functional* behaviour: adding $\boldsymbol{\tau}_t$ improves accuracy on task t with a limited degradation elsewhere, and the sign of λ_t controls whether the behaviour is acquired or suppressed. This has been verified across image classifiers, language models and, more recently, text-to-image models [33]. Related diffusion-editing methods also construct parameter changes for concept removal [10]. The framework is also directly related to model merging: uniform averaging of fine-tuned checkpoints [52] is the special case $\lambda_t = 1/T$, while Fisher-weighted merging [27] replaces the scalar coefficients with a curvature-dependent weighting.

For Low-Rank Adaptation (LoRA), Core Space merging re-expresses the updates in a compact shared basis without information loss, reducing the cost of merging [34].

For two vectors, the relation to averaging can be derived explicitly:

$$\frac{\theta_1 + \theta_2}{2} = \theta_0 + \frac{\boldsymbol{\tau}_1 + \boldsymbol{\tau}_2}{2}. \tag{3.3}$$

An average therefore reduces each displacement's coefficient relative to its single-task endpoint. A unit-coefficient sum instead gives $\theta_1 + \theta_2 - \theta_0$, a different checkpoint. If the average performs worse than a single-task model, the difference can reflect reduced edit strength as well as interaction with the second task. The experimental protocol uses the average and states this limitation explicitly. Keeping the coefficients visible avoids describing distinct weight-space configurations with the same informal word "composition".

Arithmetic is commutative for a fixed collection of saved vectors: swapping their order in a sum gives the same parameters. Sequential fine-tuning on the corresponding tasks need not be commutative, because the second optimisation starts from an already modified model. Independent

adaptation followed by arithmetic therefore defines a different procedure from learning the two classes one after the other.

3.1.3 Composition, Negation and Scaling

Three canonical operations follow from Eq. (3.2); the first two are illustrated in Figure 3.1b–c.

Composition ($\lambda_t > 0$). Adding several task vectors produces a model that exhibits multiple capabilities simultaneously, $\theta_0 + \sum_t \lambda_t \boldsymbol{\tau}_t$. In the generative setting this is the first operation of interest for this thesis: given two adapters, each encoding a different class of the same dataset, composition should yield a model that renders either class on request, each as faithfully as the corresponding single-adapter model, and without mixing the two. Whether it does so depends on how strongly the two vectors interfere.

This meaning of composition should be distinguished from requesting both objects inside one image. In the latter case, a prompt also asks the model to allocate spatial regions and satisfy a joint description. In the present class-availability setting, one model is asked for each class separately. Preserving both responses is already a meaningful requirement: a checkpoint that always generates the stronger class has acquired neither a selective interface nor useful coexistence, even if its individual images look plausible.

Negation ($\lambda_t < 0$). Subtracting a task vector, $\theta_0 - \gamma \boldsymbol{\tau}_t$ with $\gamma > 0$, can suppress the corresponding behaviour. This has been used to reduce toxic generation from language models [20]. Dedicated erasure methods also learn weight changes for styles, objects or unsafe content in diffusion models [10, 24]. Negation is the second operation of interest for this thesis, and it is also a diagnostic instrument: subtracting a vector from a composed model should remove only the class that vector encodes, and the extent to which it degrades the classes contributed by the remaining vectors is a direct measure of how localised each vector is.

The starting point determines what subtraction means. Removing $\boldsymbol{\tau}_2$ from $\theta_0 + \boldsymbol{\tau}_1 + \boldsymbol{\tau}_2$ simply restores $\theta_0 + \boldsymbol{\tau}_1$. By contrast, $\theta_0 + \boldsymbol{\tau}_1 - \boldsymbol{\tau}_2$ moves beyond the base in the opposite direction of the second adaptation. The experiments study this latter intervention, together with subtraction from the base. Its outcome cannot be predicted from the cancellation identity alone.

Scaling. The magnitude of λ_t modulates the intensity of the parameter displacement, from no edit at $\lambda_t = 0$ to extrapolation for large values, where image quality may deteriorate. The usable range can narrow as more vectors are composed, because the cumulative displacement may grow and leave the region in which interpolation retains useful behaviour. Scaling is therefore part of the experimental configuration: a composition that works only within a narrow window of λ may reflect interference, nonlinear extrapolation or unequal adaptation strengths.

Even for one vector, scaling need not translate linearly into recognition rate or visual fidelity. A coefficient of one recovers its fine-tuned endpoint, zero recovers the base, and intermediate values interpolate the weights. The generated outputs can nevertheless change abruptly when a small parameter change redirects the sampling trajectory. With multiple vectors, the relevant displacement depends on their relative directions as well as their coefficients. A narrow useful range is therefore a diagnostic observation that requires matched single-vector baselines before it can be assigned to a specific cause.

Coefficients should also be matched when testing cancellation. Subtracting a unit vector from an average leaves a negative half-vector for that task rather than removing its contribution exactly. If the intended operation is simply to undo an earlier addition, the subtraction must use the coefficient that was actually added. Otherwise the resulting checkpoint represents a new intervention.

A further property, first observed in language models, is analogy: $\boldsymbol{\tau}_D \approx \boldsymbol{\tau}_C + (\boldsymbol{\tau}_B - \boldsymbol{\tau}_A)$ for related task tuples (A, B, C, D) [20]. Its relevance here is conceptual, as it suggests that task vectors capture relations between tasks rather than isolated task identities.

3.2 Task Disentanglement

Equation (3.2) manipulates parameters, but the quantity of interest is the resulting *function*. Task disentanglement is the property that makes the passage from the former to the latter reliable.

3.2.1 Definition

Let $f(\mathbf{x}; \theta)$ denote the function implemented by the network. A set of task vectors $\{\boldsymbol{\tau}_t\}_{t=1}^{T}$ is said to be selective across a family of input domains $\{\mathcal{D}_t\}_{t=1}^{T}$ when the following task-arithmetic condition

holds, motivated by the weight-disentanglement framework of [33]:

$$f\left(\mathbf{x};\, \theta_0 + \sum_{t=1}^{T} \lambda_t \boldsymbol{\tau}_t\right) = f(\mathbf{x};\, \theta_0 + \lambda_s \boldsymbol{\tau}_s) \quad \text{for all } \mathbf{x} \in \mathcal{D}_s, \tag{3.4}$$

that is, if within the domain of task s the composed model behaves exactly as the model edited by $\boldsymbol{\tau}_s$ alone. Each task vector then acts on a distinct region of the input space and the vectors do not perturb one another's behaviour, as illustrated in Figure 3.2.

The coefficients are understood to range over a specified region of practical interest. Requiring the equality for arbitrarily large positive or negative displacements would be much stronger than requiring it near the base model. Likewise, exact equality is an idealisation; experiments generally investigate approximate preservation using sampled inputs and observable outputs. A complete selectivity requirement would also preserve the base function outside the adapted task domains. Agreement on the requested tasks alone does not exclude collateral changes elsewhere.

For a diffusion denoiser, an input includes the current noisy latent, its timestep and the text condition. Two class prompts can distinguish conditions even though their noisy latents occupy the same space. The relationship to a classifier with separate task datasets is therefore conceptual rather than a direct identification of image domains. One can ask whether another vector changes the prediction under a given class prompt, but the full generator also feeds that prediction into subsequent sampling steps. Selectivity at sampled denoising inputs and selectivity of final image distributions are related questions with different evaluation requirements.

This distinction suggests several levels of evidence. Matching final images under a fixed prompt and seed is a strong sample-level comparison. Matching recognition rates is weaker: two collections can receive the same labels while their backgrounds, appearance or diversity differ. Perceptual similarity on unrelated prompts supplies a different control, but it does not show that the edited model represents its target class successfully. A useful interpretation of approximate disentanglement therefore combines effectiveness on the task with preservation beyond it, rather than selecting whichever single metric changes least.

The comparison in Eq. (3.4) likewise retains the same coefficient for the requested task on both sides. Comparing a half-strength composition with a unit-strength single vector is useful experimentally, but it is not that exact equality test. This coefficient matching requirement helps separate the theoretical ideal from the practical configurations evaluated in the results chapter.

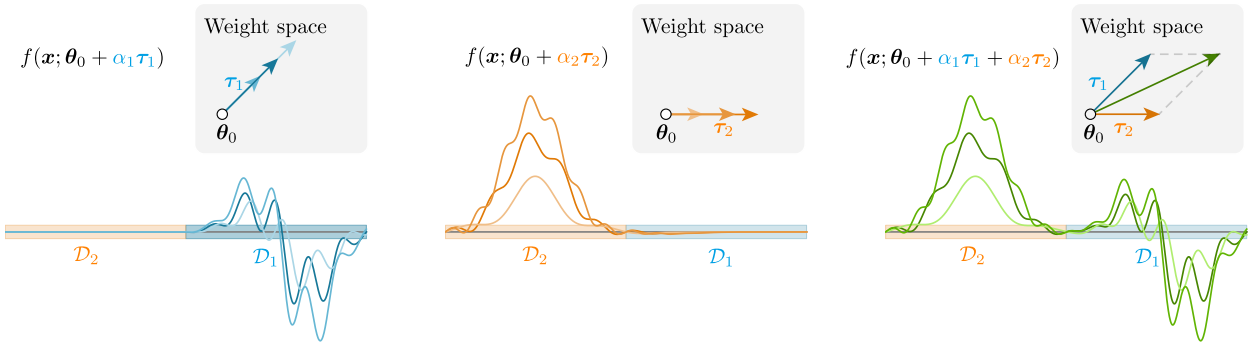


Figure 3.2: Weight disentanglement. Each task vector $\boldsymbol{\tau}_t$ modifies the output of the network only on its own input domain $\mathcal{D}_t$ (left and centre); their sum therefore reproduces the behaviour of each single-task model on the corresponding domain (right). Figure reproduced from [33].

Two clarifications are in order. First, this notion is distinct from *representation* disentanglement in the sense of latent-variable models [16], where the objective is that individual latent coordinates correspond to independent factors of variation: here the property is required of the *parameters*, not of an intermediate code. Second, Eq. (3.4) is a property of the pre-trained model and of the fine-tuning procedure jointly, not of the arithmetic operation. Ortiz-Jimenez et al. [33] show that it emerges during pre-training and that it is neither implied by nor equivalent to linearity of the network in its weights; they further show that fine-tuning in the tangent space, i.e. using the linearised model governed by the neural tangent kernel [21], enhances disentanglement and consequently improves task arithmetic, at a higher computational cost.

3.2.2 Why Disentanglement Is Required by Task Arithmetic

The success of Eq. (3.2) presupposes an approximate additivity that the network does not provide a priori. A deep model is a highly nonlinear function of its parameters, so in general

$$f(\mathbf{x};\,\theta_0+\boldsymbol{\tau}_1+\boldsymbol{\tau}_2)\neq f(\mathbf{x};\,\theta_0+\boldsymbol{\tau}_1)+f(\mathbf{x};\,\theta_0+\boldsymbol{\tau}_2)-f(\mathbf{x};\theta_0). \tag{3.5}$$

A first-order expansion around θ_0 makes the condition explicit:

$$f(\mathbf{x};\theta_0+\boldsymbol{\tau})=f(\mathbf{x};\theta_0)+\mathbf{J}_{\theta_0}(\mathbf{x})\boldsymbol{\tau}+O(\|\boldsymbol{\tau}\|^2), \tag{3.6}$$

where $\mathbf{J}_{\theta_0}(\mathbf{x})$ is the parameter Jacobian at the base model. The linear term is additive in $\boldsymbol{\tau}$. Within this approximation, selectivity additionally requires that vectors for other tasks have negligible

response on the input domain being considered. A small displacement helps control the higher-order remainder, while domain-specific Jacobian responses describe the functional localisation needed for disentanglement.

Linearity and localisation answer different questions. A linearised model adds the responses to two vectors exactly, but a vector may still have a nonzero response on both task domains. It would then alter the first task when introduced for the second, despite perfect additivity. Conversely, nonlinear changes can remain selective if they are confined to the relevant domains. The Taylor expansion is useful for explaining one route to favourable arithmetic, rather than establishing linearity as either a universal guarantee or a necessary description of every successful edit.

A simple scalar example shows where an interaction can arise. Consider a function $f(\theta_a, \theta_b) = \theta_a\theta_b$. Changing only θ_a by u changes the output by $u\theta_b$, and changing only θ_b by v changes it by $v\theta_a$. Applying both changes adds the further term uv. The parameters modified by the two updates are different, yet their effects interact through the function. In a network, successive layers and attention operations similarly combine quantities affected by different coordinates. Disjoint parameter support is therefore insufficient by itself to establish independent behaviour.

Diffusion adds a second source of interaction: repeated evaluation. Once an edited denoising prediction changes the next latent, later predictions are evaluated at a different input. Even if the first step admits a good local linear approximation, that observation does not bound the whole trajectory's deviation. This provides a reason to inspect final generated images alongside the parameter geometry. It also explains why coefficients that appear small in weight space can produce meaningful perceptual differences.

Disentanglement is what allows task arithmetic to scale. If it holds, adding a new capability requires one additional pass over its parameter vector, and capabilities can be stored, distributed and combined independently. If it fails, every combination must be validated empirically, coefficients must be retuned for each subset of tasks, and the method loses the compositional guarantee that motivates it. In the generative setting studied here the consequence is directly observable: an entangled pair of adapters produces samples in which the two classes blend into a hybrid, in which one class suppresses the other, or in which removing one adapter also erases the class encoded by the other.

The computational advantage concerns the absence of a new joint training problem. Arithmetic

still costs $O(n)$ for a vector with n coordinates, and validation can remain substantial. In particular, satisfactory pairwise compositions do not automatically establish that every larger subset will behave well: the combined displacement and possible higher-order interactions change with the subset. The pairwise study in this thesis investigates the elementary case in which those effects can be interpreted against clear single-vector baselines.

Even the size of a two-vector displacement depends on their interaction:

$$\|\boldsymbol{\tau}_1 + \boldsymbol{\tau}_2\|^2 = \|\boldsymbol{\tau}_1\|^2 + \|\boldsymbol{\tau}_2\|^2 + 2\langle \boldsymbol{\tau}_1, \boldsymbol{\tau}_2 \rangle. \tag{3.7}$$

The cross term shows why adding a second vector does not always increase the norm by the same amount. Alignment increases the combined norm, opposition reduces it, and orthogonality removes this cross term. None of those numerical cases determines selectivity without considering the function being edited.

3.2.3 The Interference Problem between Tasks

When Eq. (3.4) does not hold, the composed model can deviate from its intended task responses. The literature describes a number of identifiable mechanisms.

Parameter redundancy. Task vectors are strongly redundant: removing the smallest-magnitude components, often the vast majority of the coordinates, can leave single-task performance largely unaffected [53]. The discarded components carry little signal but contribute to the norm of the sum, diluting the informative directions of the other tasks.

Sign conflicts. A parameter may be pushed in opposite directions by two tasks. Summation then yields a value close to zero, cancelling both contributions instead of preserving either. TIES-Merging [53] addresses both mechanisms explicitly through a three-step procedure — trimming low-magnitude entries, electing a dominant sign per coordinate, and averaging only the entries in agreement with it — and reports consistent gains over plain summation. Figure 3.3 contrasts the two mechanisms with the non-conflicting case.

The corresponding operator is shown in Figure 3.4. Reading it from left to right separates the removal of small updates from sign selection and the final aggregation. The merged displacement

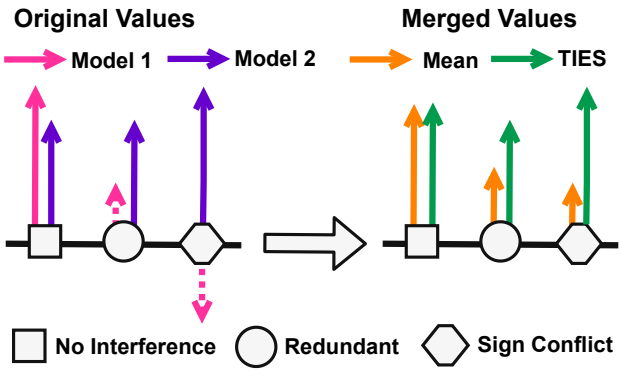


Figure 3.3: Types of interference between two task vectors and the merged values produced by plain averaging and by TIES-Merging. Redundant parameters dilute the informative update of the other model, and sign conflicts cancel both contributions under averaging. Figure reproduced from [53].

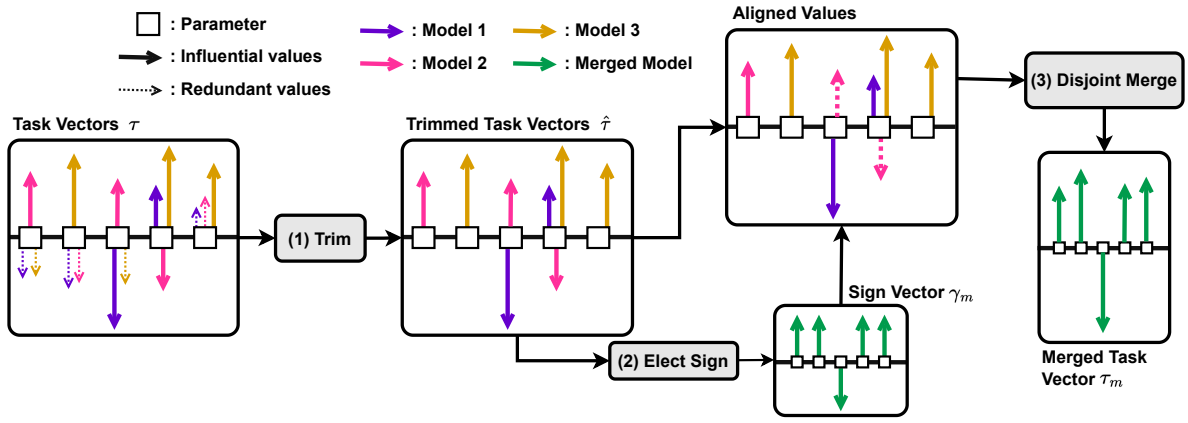


Figure 3.4: TIES-Merging pipeline. Small-magnitude task-vector entries are trimmed, a sign is elected for each coordinate, and retained entries agreeing with that sign are aggregated. The colours distinguish input models and the resulting merged update. Reproduced from Figure 1 of [53].

is consequently not the ordinary mean of all entries: only retained contributions with the selected sign enter each coordinate.

Subspace overlap. Parameter geometry can help describe where updates overlap, but a global cosine similarity does not determine whether their functional effects interfere. ZipLoRA [46] learns column-wise mixing coefficients for subject and style adapters with a penalty on their alignment, while Mix-of-Show [12] optimises a merged model through gradient fusion. DARE [54] instead randomly drops update entries and rescales those retained; its results in language models show that

substantial sparsification can preserve useful adaptation in the settings studied. These methods address different aspects of merging and do not supply a general orthogonality guarantee.

Magnitude imbalance. Task vectors obtained under different data sizes, learning rates or numbers of steps have widely different norms. In an unweighted sum a larger update can dominate, which is why per-task coefficients or explicit normalisation are typically necessary.

Sign conflicts are easiest to understand coordinate by coordinate. Updates of $+0.02$ and -0.02 cancel in a sum, while updates of $+0.02$ and $+0.01$ reinforce each other. Neither pattern establishes whether the result is desirable: opposite changes might encode genuinely different class-specific requirements, and reinforcing changes might accumulate an unwanted background association. A merging rule that elects one sign resolves the numerical conflict but also makes a choice about which contributions to retain. Its success must be assessed on both tasks rather than inferred from the absence of cancellation alone.

Global similarity likewise hides layer-dependent structure. Two vectors may be broadly aligned because they share many small changes, yet disagree at the few projections important for class selection. Conversely, a near-zero total inner product can result from cancellation between positive and negative layer-wise contributions. Comparing norms or cosines by layer can provide more detail, but these remain descriptions of weights. Their interpretation requires the input-dependent Jacobian responses discussed above; Euclidean orthogonality in parameter space does not imply that generated concepts are independent.

Magnitude imbalance should be treated with the same care. A large norm can result from many modest changes or from a small set of large ones, and parameters in different layers need not have equal functional sensitivity. Normalising every vector to the same norm would control one geometric quantity while potentially weakening a useful update or amplifying an ineffective one. Likewise, restricting the trainable parameter set changes the space in which the norm is measured. These observations motivate comparing behaviour under recorded coefficients and training settings, rather than using norm alone as a measure of concept strength.

These mechanisms can coexist in one pair. Trimming may reduce redundant coordinates while leaving a strong sign conflict at another layer; rescaling may balance total norms while preserving overlapping functional effects. A visible failure should therefore not be assigned to one mechanism

solely because a corresponding weight statistic is present. Several explanations can be compatible with the same generated image.

In diffusion models these mechanisms manifest as concrete visual artefacts: *concept blending*, where the attributes of two subjects merge into a single hybrid; *concept dominance*, where one adapter suppresses the other; *identity drift*, where the subject loses the traits established by its own adapter; and a general loss of fidelity to the pre-trained prior, with saturated colours and incoherent structure, as $\|\sum_t \lambda_t \boldsymbol{\tau}_t\|$ grows. Because a generative model has no scalar accuracy, these failures cannot be summarised by a single number, which complicates both diagnosis and evaluation.

3.2.4 Established Results and Open Questions

The related literature can be summarised along three axes. *First*, weight-space arithmetic can be useful: task vectors have been composed and negated in image classification and language-model settings [20, 52, 27]. *Second*, interference is understood at the level of coordinates. Trimming, sign election and sparsification recover a large fraction of the degradation caused by naive summation [53, 54]. *Third*, the property that governs composition has been formalised and can be improved: weight disentanglement is a measurable quantity, it emerges from pre-training, and tangent-space fine-tuning increases it [33].

MoDER provides a related example of class-specific reuse, combining learned textual experts to construct prototypes for unseen classes in incremental classification [35].

A complementary direction restricts which parameters are adapted. Figure 3.5 contrasts full nonlinear fine-tuning, full-model linearisation and attention-projection fine-tuning in the transformer setting studied by Jin et al. [22]. It helps distinguish a restriction on trainable weights from a change to the forward model. The attention-only diffusion experiments retain the nonlinear forward computation and test a related parameter restriction in a U-Net.

Several questions remain open, and they delimit the contribution of this thesis. The cited formalisation of weight disentanglement uses CLIP-based image classifiers with distinct task datasets. In a text-to-image diffusion model the tasks share their visual input space: two classes of the same dataset are requested through closely related prompts, are rendered in the same output space, and share most of their visual statistics. Equation (3.4) therefore does not apply verbatim, and whether its generative counterpart holds for concepts this close is an open empirical question.

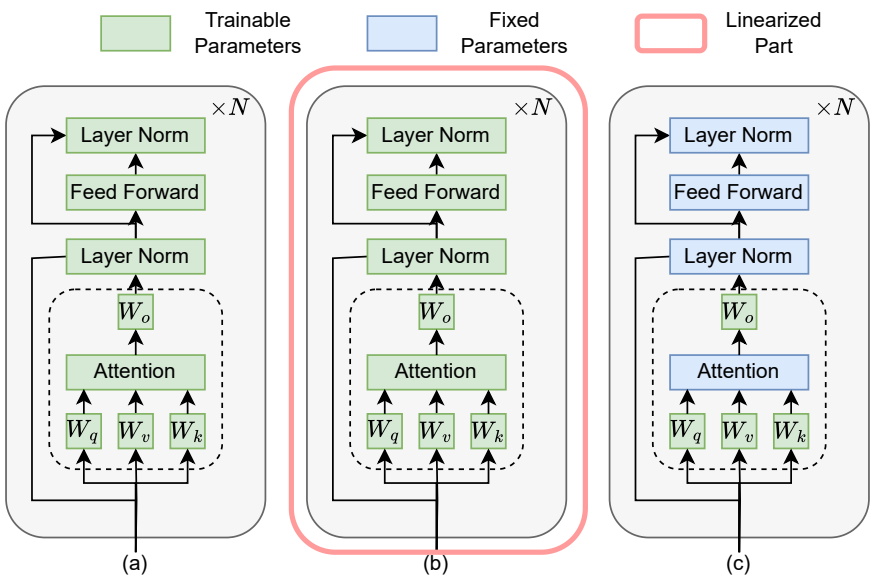


Figure 3.5: Three adaptation paradigms for a transformer: (a) full nonlinear fine-tuning; (b) full-model linearisation; (c) fine-tuning the attention projections. Green denotes trainable parameters, blue denotes fixed parameters, and the red outline identifies the linearised computation. The schematic describes the cited transformer study. Reproduced from Figure 4 of [22].

Furthermore, the reviewed composition methods intervene at different levels. ZipLoRA [46] and Mix-of-Show [12] optimise a merge, Custom Diffusion [25] constrains selected conditioning projections, and composable diffusion [26] combines denoising outputs at inference. Their objectives differ from the focused characterisation pursued here: whether a single vector transfers its class, whether averaged vectors retain similar classes under their respective prompts, and whether a negative vector suppresses its target without harming a positively weighted class. The cited erasure methods chiefly target concepts already accessible in the base [10, 24], whereas the present vectors are learned through class-specific adaptation. Finally, the U-Net's convolutional and attention components offer different locations for introducing an update, even though their effects interact. Characterising these elementary operations under full and attention-only adaptation is the specific empirical question addressed by this thesis.

The transfer from existing results therefore involves several levels of uncertainty. Evidence that a merging rule improves classification motivates a generative experiment, but it does not identify which visual attributes a diffusion update will preserve. Evidence that an adapter can be merged with a style adapter does not establish that two fine-grained classes remain distinguishable under

their own prompts. Evidence of target suppression does not establish preservation of a related class. These are different claims, and the protocol needs a reference configuration for each one.

Closely related traffic-sign classes make those distinctions observable. Their common outline or colour can be retained while the decisive numeral or pictogram changes. Such an output may remain perceptually plausible and still represent interference from the perspective of class identity. At the same time, a classifier can confuse malformed images or assign a label to an invalid sign. The task structure therefore calls for both a consistent recognition proxy and visual inspection. It provides a focused test bed for selectivity without claiming that all aspects of image generation are captured by the sign label.

Finally, the cited literature is used to motivate questions rather than to assert that every possible comparison is absent from prior work. The contribution pursued here is a specific empirical characterisation under a shared base model, recorded training recipes and fixed arithmetic configurations. Keeping that scope explicit allows the study to connect to broader merging and disentanglement research while reserving general claims about other architectures, tasks and coefficients for further experiments.

A further distinction is between variation over generation seeds and variation over training runs. Repeated sampling characterises a fixed edited generator. Repeating adaptation would instead test how consistently a recipe produces useful vectors. The present study provides the first kind of evidence; its sample-level uncertainty should not be interpreted as covering the unmeasured variability of independently trained checkpoints.

3.3 Task Arithmetic for Image Generation

The final section reviews how the composition and removal of concepts have been approached in text-to-image diffusion models, and positions the study carried out in this thesis with respect to that literature.

3.3.1 Composing Concepts in Diffusion Models

The need to render several personalised concepts in a single model has been addressed at three different levels. At the level of the sampling procedure, composable diffusion [26] interprets the

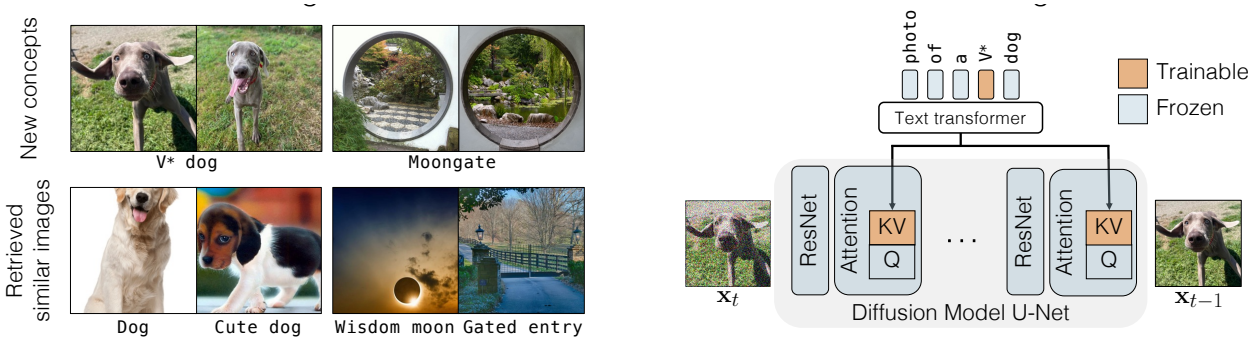


Figure 3.6: Custom Diffusion. Images of the new concepts are paired with retrieved real images of similar captions, used as regularisation (left); only the key and value projections of the cross-attention layers of the U-Net, together with the embedding of the modifier token V^*, are trained (right). Figure reproduced from [25].

denoiser as an energy-based model and combines the noise estimates obtained under different conditions, so that concepts are composed in output space at every denoising step without touching the parameters. At the level of training, Custom Diffusion [25] fine-tunes only the key and value projections of the cross-attention layers (Figure 3.6) on several concepts jointly, or merges separately trained adapters by solving a constrained least-squares problem over those projections; Mix-of-Show [12] fuses several adapters into one through gradient fusion, and ZipLoRA [46] learns per-column mixing coefficients that minimise the overlap between a subject and a style adapter. At the level of parameters, plain task arithmetic sums the adapters without any further optimisation.

The methods of the second family are the direct evidence that summation alone is often not sufficient: each of them was introduced because naively adding adapters produced the blending and dominance artefacts described in Section 3.2.3. Their evaluations, however, concern pairs of concepts that are semantically distant, typically a subject and a style or two unrelated objects, and report the merged result rather than a systematic account of where and how plain summation fails. The regime in which the composed concepts are similar, and in which the sum is most likely to conflate them, is largely unexplored.

ZipLoRA provides a concrete illustration of an optimised merge (Figure 3.7). Its mixing coefficients are learned while comparing the merged model with the individual subject and style models. The diagram makes the additional coordination step explicit: the original updates supply the ingredients, but their final combination is selected by an optimisation objective rather than by

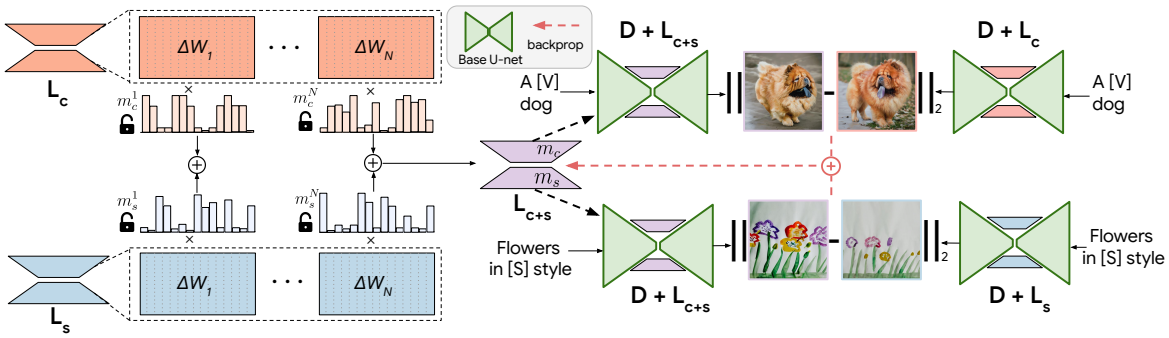


Figure 3.7: ZipLoRA merging procedure. Subject and style updates receive learned column-wise mixing coefficients, with reference generations from the separate models used to retain their respective properties. This illustrates an optimised subject-and-style composition. Reproduced from Figure 2 of [46].

fixed arithmetic alone.

The level of intervention determines what must be available when concepts are combined. An inference-time method can retain separate conditional predictions and assemble them during sampling. A method that learns a merge introduces an additional optimisation problem, with its own objective and reference examples or predictions. Plain arithmetic uses only the recorded parameter updates once they have been trained. These differences matter when judging convenience, cost and modularity, but they do not by themselves determine which approach gives the best semantic result. They define different compromises between independent adaptation and explicit coordination.

The word "adapter" is also used at more than one level in this literature. Some methods store low-rank factors, others store selected projection updates, and a full task vector can describe an entire denoiser displacement. The arithmetic considered here is defined on the actual changes to the U-Net weights, regardless of how an update is stored. Compact storage does not guarantee independence: two small adapters can still redirect the same prompt association. Conversely, a larger update can preserve another task if its functional effect is sufficiently selective. The trainable subset and the merging procedure are therefore separate methodological choices.

For similar traffic-sign classes, coexistence is tested by asking the same merged model to answer both prompts. A favourable result would preserve the distinctive content of each class across the evaluated seeds. A model that generates a convincing sign with the wrong pictogram demonstrates the importance of this distinction: visual plausibility does not establish task availability. The present

evaluation is deliberately confined to that question and does not test spatial composition of several personalised objects within a single image.

The appropriate reference therefore depends on the intended composition. Joint subject-and-style generation asks whether both attributes appear together; class availability asks whether either class can be selected when requested. A metric designed for one may overlook failure in the other. The task-vector evaluation specifies its prompt coverage before interpreting recognition as evidence of coexistence.

3.3.2 Machine Unlearning and Concept Erasure

Machine unlearning concerns removing the influence of selected training data from a learned model without necessarily retraining it from scratch [3]. In the data-removal setting, the natural reference is a model trained without the data to be forgotten. Text-to-image models pose a related but different problem: a request may target a visual concept, such as an object or style, rather than a known set of training examples. In this setting, *concept erasure* aims to reduce the model's ability to generate the target concept while retaining its ability to generate other content [10, 24].

These two requirements must be assessed separately. Suppression under one prompt does not show that the concept is absent under other descriptions, and broad changes to the model may also damage related concepts. For a generative model, examples of the resulting images help reveal what was removed or altered; quantitative comparisons can complement them, but neither by itself proves that the influence of particular training examples has been eliminated.

The reference model determines the strength of an unlearning claim. Returning to the original base after an adaptation removes that adaptation's parameter displacement, but the base may already contain knowledge of the target concept. Likewise, moving in the opposite direction of a learned vector does not supply the checkpoint that would have resulted from retraining without its examples. Fine-tuning is an optimisation trajectory with data-dependent updates; its endpoint difference does not record a general inverse of that trajectory. These observations explain why weight subtraction is a valid intervention to study without treating it as a certificate of forgetting.

Concept-level evaluation introduces another issue: the target must be specified independently of the literal prompt. A traffic-sign class can be described by its name, numeral, shape or purpose. Suppression under one saved description may reflect a weakened conditioning association rather

than the inability to render the corresponding visual pattern. Broader prompt coverage could test whether the effect persists across descriptions, but the present protocol keeps the recorded prompts fixed to compare arithmetic configurations. Its results consequently describe suppression under those conditions.

Preservation must also distinguish a related class from an unrelated control. Removing a red-bordered sign by damaging all red-bordered signs would lower target recognition while violating selectivity. A control image of a house could remain similar and still fail to expose that collateral effect. Retained class tests and unrelated-prompt controls therefore complement one another: they probe nearby and more distant consequences of an edit. Neither expands the coverage of the other, and both remain conditional on the prompts and seeds actually evaluated.

Baseline frequency is particularly relevant when the target is already rare. If the base almost never generates a class under the saved prompt, a low rate after subtraction supplies little evidence of an additional suppressive effect. The comparison must ask what changed from that reference, not merely whether the final target count is small.

The present thesis studies a narrower operation. Its class-specific task vectors are obtained by fine-tuning a common base model; a negative vector is then applied in weight space to test whether its class becomes less generable while other classes remain available. This is a test of selective concept removal and interference between vectors, not a guarantee of data unlearning in the sense of reproducing a model retrained without particular examples.

3.3.3 Removing Concepts from Diffusion Models

The complementary problem of removing a concept has been studied mainly for safety and copyright, with the aim of erasing from the pre-trained model a style, an object or an unsafe category that it has already learned. Erased Stable Diffusion [10] fine-tunes the denoiser so that the noise estimate under the target concept is steered away from it, using the model's own classifier-free guidance as the training signal (Figure 3.8); Ablating Concepts [24] matches the distribution generated under the target prompt to that of an anchor prompt. Both methods obtain a fine-tuned model whose difference from the original is, by construction, a task vector, and task arithmetic itself has been shown to remove behaviours by subtracting the vector learned on them [20].

These works do not address how a negative task vector interacts with positive vectors learned

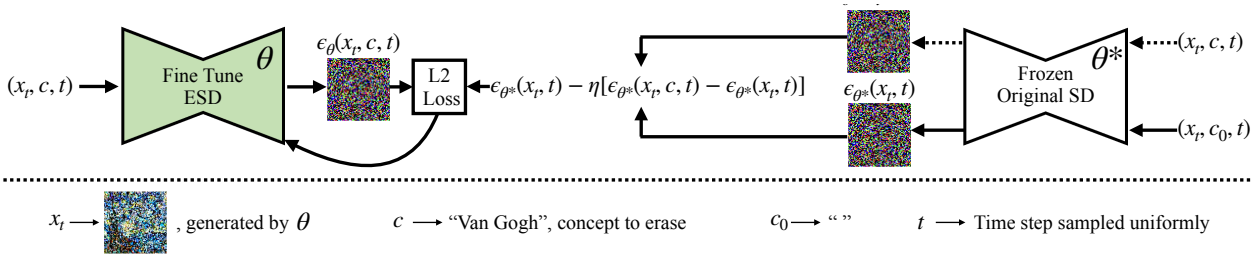


Figure 3.8: Erased Stable Diffusion. The fine-tuned model θ is trained to match a target noise estimate obtained from the frozen model θ^* by applying negative classifier-free guidance in the direction of the concept c to be erased. Figure reproduced from [10].

for similar concepts. Subtracting a vector from the base model, or from a model containing another class's vector, may suppress the target class but could also impair the remaining class. This thesis evaluates that selectivity as a probe of interference between independently learned vectors, rather than as a safety mechanism or a claim of data unlearning.

An erasure-trained displacement and a class-learning displacement have different objectives. If an erasure method produces parameters θ_{erase}, its difference $\theta_{\text{erase}} - \theta_0$ is an update whose positive application recovers the erasure-trained model. A vector obtained by learning a class instead has a positive endpoint intended to improve that class. Applying its negative is an extrapolation whose effects need to be measured. Representing both as parameter differences does not make their training signals or semantic directions equivalent.

The outcome of removal also needs a description of what replaces the target. A generator continues producing an image even if it no longer draws the requested class. It might produce a related sign, a different dataset class, an unrelated object or an invalid structure. A reduction in requested-class recognition cannot distinguish these outcomes by itself. In particular, removing a confusion with the positively weighted class can leave other incorrect labels unchanged. The later analysis therefore keeps predictions for the target, its paired class and other labels separate, and examines the corresponding image grids.

Subtracting from the base and subtracting in the presence of a positive vector answer complementary questions. The first describes how the negative direction changes the pre-trained generator. The second asks whether that direction interferes with an independently added capability. Comparing only the latter with an unedited base would mix both changes. Including single-vector references

makes the retained class and the negated class interpretable separately, while still leaving the result conditional on the chosen coefficients. This is the role of negation as a diagnostic operation in the present study.

Negative extrapolation also visits weights outside the segment joining the base and the positive endpoint. Good specialisation at the endpoint says little about the quality of that opposite region. A useful removal effect must consequently be distinguished from general deterioration of the generator, which could reduce recognition of both the target and retained classes at once.

3.3.4 Positioning of This Work

This thesis studies plain task arithmetic on a text-to-image latent diffusion model, without training on pairs of concepts or adding an inference-time composition mechanism. Task vectors are learned independently for the evaluated classes by adapting either the full U-Net or its attention projections. The recorded recipes vary class-specific wording, distinct identifiers and a DreamBooth-style prior term. Addition is evaluated by comparing each edited model with the pre-trained baseline; averaged composition tests whether two classes remain distinct when their vectors are combined; and negation tests whether a negative vector suppresses its target class without damaging the positively weighted class. The training conditions use the same arithmetic rule, so differences in behaviour can be examined in relation to how the task vectors were obtained.

The attention-only comparison is also connected to work on the locus of adaptation. Jin et al. [22] study attention-module fine-tuning as a way to improve weight disentanglement in task arithmetic. That work motivates asking whether restricting updates can improve their independent use, but its findings do not establish the outcome for this diffusion generator and its closely related sign classes. The present study tests the question using the ordinary nonlinear U-Net and its complete sampling procedure, rather than assuming that a structural restriction automatically gives functional separation.

Three comparisons organise the interpretation. Specialisation asks whether a vector is useful at all for its own class. Averaged composition asks whether both classes remain available under a shared edited checkpoint. Negative-vector interventions ask how target suppression and retained-class behaviour change together. The non-traffic-sign control then examines perceptual agreement with the base under single-vector edits. These comparisons address complementary parts of the

problem; a result for one does not substitute for a result in another. In particular, an ineffective vector can preserve the base while contributing no useful capability to a composition.

The empirical scope follows from the recorded configurations. The study evaluates selected class pairs and training recipes at fixed checkpoints and sampling settings. Full and attention-only regimes also use different learning rates, so a difference between them concerns the practical configurations as a whole. The results can support a pattern of greater behavioural selectivity within those conditions without isolating its cause to the trainable subset. This framing connects the theoretical discussion to the method chapter: parameter restrictions, prompt recipes and arithmetic coefficients must be stated precisely before evidence for task disentanglement can be assessed.

Using shared generation seeds makes these comparisons more informative by holding initial noise fixed. It allows both aggregate recognition and paired perceptual changes to be examined. The resulting evidence remains attached to the sampled prompts and checkpoints, but the pairing reduces ambiguity about whether an observed difference arose simply from drawing different starting noise for the two models. Chapter 4 formalises this protocol and describes the experiments.

4. Method

This chapter presents the experimental framework for constructing and evaluating class-specific U-Net task vectors for Stable Diffusion. The experiments test specialisation, composition and subtraction, and compare how the trainable parameters, training loss and text conditioning affect these operations. The evaluation has two components: recognition of generated traffic signs by an independent classifier, and a paired LPIPS control measuring changes to the base model on prompts outside the traffic-sign domain. Section 4.3 introduces the model and data; Sections 4.4 and 4.5 specify conditioning and fine-tuning; Section 4.6 defines the weight-space interventions; and Sections 4.7 and 4.8 describe the metrics and experimental protocol.

4.1 Overview

Each adaptation starts independently from the same pre-trained U-Net parameters θ_0 and uses images from one GTSRB class. Its task vector is

$$\boldsymbol{\tau}_c = \theta_c - \theta_0. \tag{4.1}$$

At inference, vectors from the same training recipe and parameter regime, each trained for 12,000 steps, are combined with θ_0 by linear arithmetic. The combined models require no additional training on pairs of classes.

The classifier experiment evaluates specialisation, composition and negative-vector interventions on selected pairs of traffic-sign classes. The separate LPIPS experiment compares the base model with models edited by one positive or negative vector on five control prompts. These measurements address different questions: whether the requested traffic-sign class is recognised, and how much an edit changes generations outside the adaptation domain. Generated examples accompany the numerical scores.

4.2 Experimental Questions and Comparisons

The starting question is whether adaptations for two classes of the same dataset remain usable after weight-space arithmetic. A useful vector must first make its own class recognisable when added to the base model. A composed model should then answer each class prompt without consistently substituting the other class. Finally, subtracting one vector should reduce recognition of its class while leaving a positively weighted class recognisable. Section 4.6 defines the exact weights and baselines for these three tests. In particular, the tested positive composition is an *average* of two vectors, with coefficients $(\frac{1}{2}, \frac{1}{2})$; an unscaled sum $(1, 1)$ is not evaluated.

Four training recipes examine the contribution of class descriptions, identifiers and the training loss. Each recipe is evaluated with both full U-Net and attention-only updates:

1. **Classic fine-tuning without an identifier (Classic–0).** Class-specific descriptions provide the baseline condition. Full U-Net and attention-only updates ask whether limiting the trainable parameters improves arithmetic between independently learned classes.

2. **DreamBooth-style adaptation with distinct identifiers (DB–C).** A separate identifier for each class and a same-class prior term test a DreamBooth-style approach to separating the adaptations. Comparing the two parameter regimes tests the role of the update location under this recipe.

3. **Classic fine-tuning with distinct identifiers (Classic–C).** The instance-only loss with class-specific descriptions and identifiers tests the contribution of identifiers within Classic fine-tuning. Comparing Classic–0 and Classic–C changes only the presence of the identifier within the Classic recipe.

4. **DreamBooth-style adaptation with generic wording (DB–G).** The class name is removed from the instance prompt while its identifier remains. This asks whether the identifier still supports class-specific composition when the prompt says only “traffic sign”.

All four recipes are evaluated on the same four class pairs. The $(20, 36)$ pair provides a detailed example of the behaviour underlying the mean classifier scores. A single-vector LPIPS

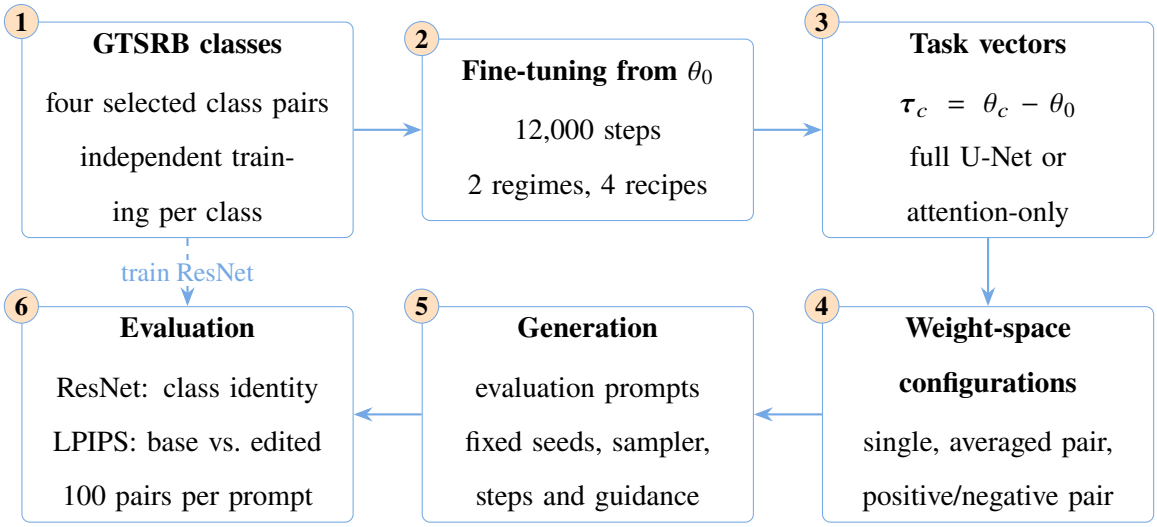


Figure 4.1: Experimental pipeline. Independent class-specific adaptations of the pre-trained U-Net define task vectors, which are combined in weight space. An independent 43-class GTSRB classifier evaluates the generated signs. A separate paired LPIPS control compares base and single-vector edited generations on five non-traffic-sign prompts.

control measures changes on prompts outside the traffic-sign task. Table 4.2 gives the exact recipe differences and Section 4.8 specifies the evaluation protocol.

4.3 Experimental Setup

4.3.1 Pre-trained Model

The generator is Stable Diffusion 1.5 [41], loaded from `sd-legacy/stable-diffusion-v1-5`. Its variational autoencoder and CLIP text encoder [37] remain frozen. All adaptations and arithmetic operations therefore concern the conditional U-Net. The latent decoder and text-conditioning network are shared across the base and edited configurations.

The implementation optimises a zero-initialised parameter displacement δ and evaluates the U-Net at $\theta_0 + \delta$. Thus the saved δ directly represents τ_c in Eq. (4.1). The experiments described here use the ordinary nonlinear forward pass at the edited weights.

Table 4.1: Classes selected for the four-pair mean. Counts refer to the available per-class diffusion training pools, before separating instance and prior images where required.

Class ID	Class description	Identifier	Images
1	Speed limit 30 km/h	`nzk02`	1,500
2	Speed limit 50 km/h	`trv03`	1,500
12	Priority road	`qnk13`	1,410
13	Yield	`vjt14`	1,440
15	No vehicles	`gzk16`	420
18	General caution	`kxd19`	810
20	Dangerous curve to the right	`zmt21`	240
36	Go straight or right	`qxb37`	270

4.3.2 Dataset and Evaluated Classes

GTSRB contains 43 traffic-sign classes [50]. Similar colours and outlines can occur across classes whose identity depends on a small numeral or pictogram (Figure 4.2). The diffusion training code loads the `train` split of `tanganke/gtsrb`. The version used in this study contains 26,640 training images and 12,630 test images.

The classifier covers all 43 classes. The mean for pairwise task-vector evaluation covers eight classes, organised into four pairs: $(1, 2)$, $(12, 18)$, $(20, 36)$ and $(13, 15)$. Their training-set counts and identifier strings are reported in Table 4.1. These pairs include signs differing in numerals, shapes and pictograms.

For diffusion fine-tuning, images are converted to RGB, resized with bicubic interpolation so that the shorter side is 512 pixels, centre-cropped to 512×512, and normalised to $[-1, 1]$. Each run loads only its target class. A fixed number of optimisation steps is used, rather than a fixed number of epochs. Consequently, smaller class pools are revisited more often; differences in sample counts do not imply different numbers of parameter updates.

The evaluation classifier uses the GTSRB training archive exposed by Torchvision, whose file names retain sequence identifiers. The local training pool contains 26,640 images and is split by



Figure 4.2: The 43 GTSRB classes. The recognition model covers the complete label set, whereas the pairwise task-vector experiments use the eight classes in Table 4.1.

Table 4.2: Recipes used in the four-pair classifier comparison. The additional prior term uses real images of the target class.

Recipe	Wording	Identifier	Prior term
Classic–0	Class-specific	None	No
DB–C	Class-specific	Class-specific	Yes
Classic–C	Class-specific	Class-specific	No
DB–G	Generic	Class-specific	Yes

sequence for classifier training and validation, as described in Section 4.7.1. The official test images are used only for the classifier's real-image evaluation. Real GTSRB images are not references in the LPIPS control.

4.4 Conditioning and Training Recipes

The training recipes distinguish the instance prompt used for adaptation from the class prompt used for the additional prior term, when present. A class-specific prompt is taken from the prompt CSV; a generic prompt is `a photo of a traffic sign.` A class-specific identifier is inserted after the article following "photo of". For example, the instance prompt for class 20 becomes

```
a photo of a zmt21 dangerous curve to the right traffic sign.
```

With generic wording, it instead becomes `a photo of a zmt21 traffic sign.`

These identifiers are strings processed by the existing CLIP tokeniser. No vocabulary entries or trainable text embeddings are introduced; an identifier need not correspond to a single tokeniser token. Its association with a class is learned through the U-Net adaptation.

Table 4.2 specifies the four training recipes. The names distinguish the complete training recipes, because their differences extend beyond textual wording. Min-SNR weighting is disabled in all evaluated runs.

The comparison evaluates the four recipes at 12,000 steps in both parameter regimes. All four recipes cover the same four selected pairs: $(1, 2)$, $(12, 18)$, $(20, 36)$ and $(13, 15)$. These

comparisons target the effects of adding distinct identifiers, using a prior term and replacing class-specific descriptions with generic wording.

The names DB–C and DB–G denote the DreamBooth-style recipes used here. Their additional loss is evaluated on a disjoint subset of real images from the same target class, using the prompt without the identifier. These prior images are not generated by the pre-trained model. If the available class pool contains n_c images, the implementation assigns

$$n_c^{\text{prior}} = \min(200, \lfloor n_c/2 \rfloor), \qquad n_c^{\text{inst}} = n_c - n_c^{\text{prior}}. \tag{4.2}$$

The split follows the loaded image order. For classes 36 and 20 this gives, respectively, 135 and 120 images in each subset. For the other six classes, 200 images are assigned to the prior subset. Both subsets participate in optimisation; the prior subset is not a validation set.

Classic–0 versus Classic–C across the four selected pairs compares the absence and presence of class-specific identifiers at otherwise matching recipe settings. Classic–C versus DB–C compares training without and with the real-image prior term, including the associated division of the training pool. DB–G versus DB–C compares generic and class-specific wording at otherwise matching recipe settings.

4.5 Fine-Tuning Regimes and Task Vector Extraction

4.5.1 Training Objective

Let $E(\mathbf{x})$ denote a sampled, scaled VAE latent, p a prompt, t a uniformly sampled training timestep and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. Training uses the pre-trained noise schedule and the epsilon-prediction objective

$$\mathcal{L}(\theta; \mathcal{D}, p) = \mathbb{E}_{\mathbf{x},t,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \epsilon_\theta(\mathbf{z}_t, t, p)\|_2^2\right], \tag{4.3}$$

where $\mathbf{x} \in \mathcal{D}$, $\mathbf{z}_t$ is the noised latent, and the implementation averages the squared error over latent coordinates.

Classic–0 and Classic–C minimise the instance loss alone. DB–C and DB–G minimise

$$\mathcal{L}_{\text{train}} = \mathcal{L}(\theta; \mathcal{D}_c^{\text{inst}}, p_c^{\text{inst}}) + \lambda_{\text{prior}}\mathcal{L}(\theta; \mathcal{D}_c^{\text{prior}}, p_c^{\text{class}}), \qquad \lambda_{\text{prior}} = 1. \tag{4.4}$$

The class prompt omits the identifier. With generic wording it still accompanies real images of the particular class being adapted; it does not represent a broad sample of the pre-trained model's output distribution.

4.5.2 Trainable Parameters

Full U-Net. A dense displacement is learned for every U-Net parameter, including convolutional, attention, feed-forward, normalisation and embedding parameters. The effective fine-tuned parameters are $\theta_c^{\text{full}} = \theta_0 + \delta_c^{\text{full}}$, and

$$\boldsymbol{\tau}_c^{\text{full}} = \theta_c^{\text{full}} - \theta_0 = \delta_c^{\text{full}}. \tag{4.5}$$

Attention-only. Jin et al. [22] report that task vectors obtained by fine-tuning only the attention modules of CLIP vision transformers yield higher classification accuracy after composition than vectors obtained by full-model fine-tuning. This finding motivates the attention-only regime evaluated here.

Dense displacements are learned only for parameters whose names contain `to_q`, `to_k`, `to_v` or `to_out.0`. This includes the query, key, value and output projection weights of self-attention and cross-attention, together with projection biases where present. Other U-Net parameters remain at their pre-trained values. These are full projection updates, with no low-rank factorisation. The saved vector omits frozen coordinates; mathematically, it can be embedded in the full U-Net parameter space by filling them with zeros.

The full vector contains 859,520,964 scalar parameters. The attention-only vector contains 93,536,640, approximately 10.9% of the full count: 49,574,080 in self-attention and 43,962,560 in cross-attention. The ambient parameter space is the same in both regimes; the number of trainable coordinates differs.

At 512×512 image resolution, the final encoder stage and first decoder stage operate on 8×8 latents and have no transformer blocks. Their parameters are unchanged by attention-only adaptation. The middle block also operates at 8×8 and contributes trainable attention projections.

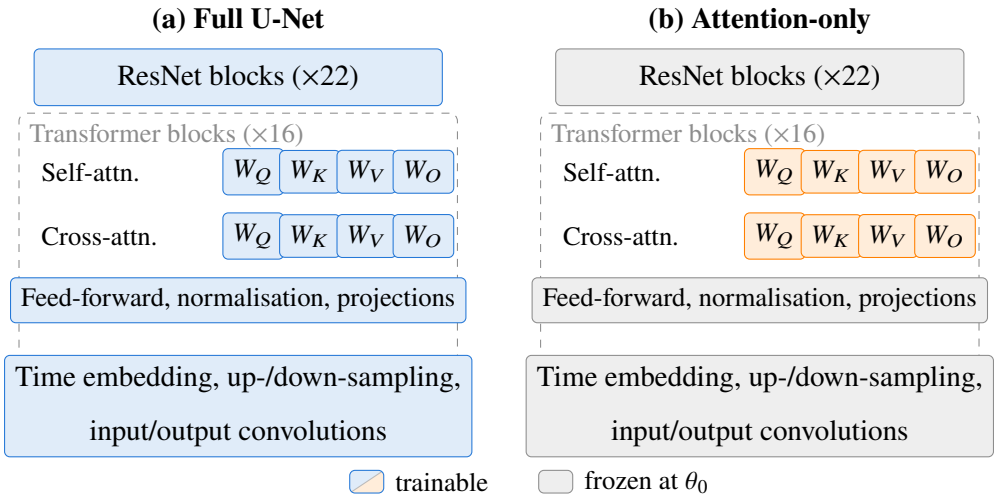


Figure 4.3: Trainable parameters in the two regimes. Full fine-tuning updates all U-Net coordinates. Attention-only fine-tuning updates the selected self-attention and cross-attention projections in the 16 transformer blocks, including the middle block.

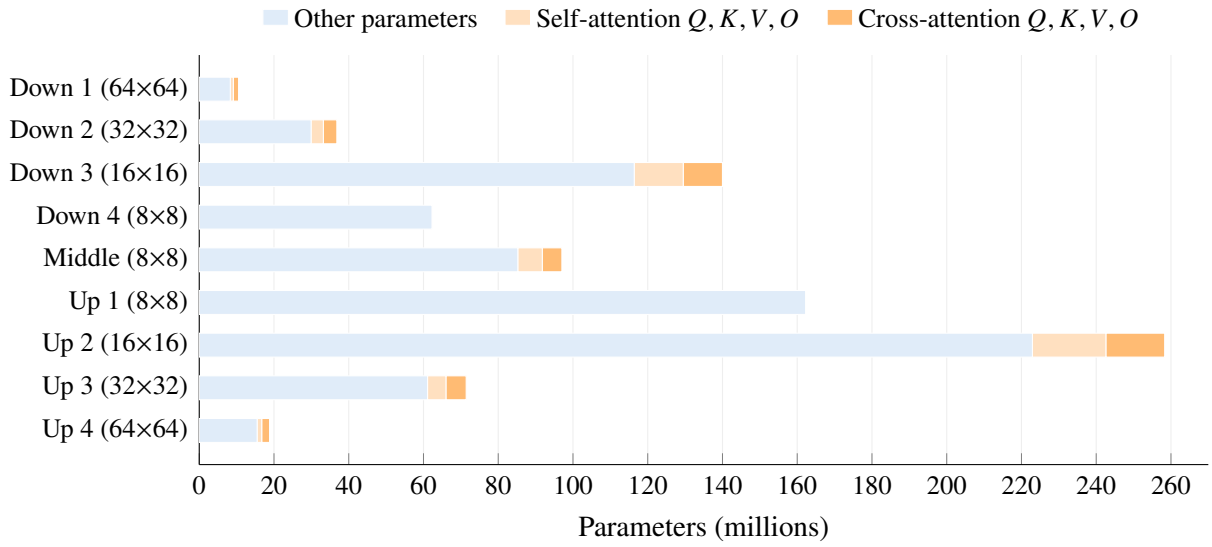


Figure 4.4: Distribution of U-Net parameters by stage, separating the selected attention projections from other parameters. At 8×8 latent resolution, the middle block contains attention projections, while the encoder and decoder stages contain only other parameters.

Table 4.3: Diffusion fine-tuning settings. The two regimes use different initial learning rates.

Setting	Full U-Net	Attention-only
Resolution	512×512	512×512
Optimiser	AdamW	AdamW
Initial learning rate	5×10^{-6}	10^{-5}
Learning-rate schedule	Cosine, no warm-up	Cosine, no warm-up
Instance micro-batch size	1	1
Gradient accumulation	4	4
Optimiser updates	12,000	12,000
Gradient clipping norm	1	1
Weight decay	0	0
Training precision	Float32	Float32
Training seed	1337	1337

4.5.3 Optimisation Settings

Table 4.3 reports the fine-tuning settings. Each optimiser step accumulates four micro-batches of one instance image. When the prior term is enabled, each micro-batch also contains one independently drawn prior image. The evaluated runs use 12,000 optimiser updates and the final saved task vector.

Training uses no exponential moving averages, conditioning dropout, orthogonality regularisation or displacement-norm projection. Apart from the recipe-specific loss terms above, the task vectors are used without post-training pruning or sign-based merging. The full-versus-attention comparison concerns two fine-tuning configurations that differ in their trainable coordinates and initial learning rate. Results therefore describe the combined effect of these settings.

4.6 Weight-Space Configurations

For a pair (c_1, c_2), let $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ denote vectors trained with the same recipe and regime. Define

$$\theta(a, b) = \theta_0 + a\boldsymbol{\tau}_1 + b\boldsymbol{\tau}_2. \tag{4.6}$$

Table 4.4: Prompt coverage in the classifier experiment. Each listed prompt is evaluated with 100 seeds, giving 12 sets of 100 images per pair, recipe and regime.

Configuration	Coefficients (a, b)	Evaluated prompts
Base	$(0, 0)$	p_1, p_2
Positive single vector	$(1, 0)$; $(0, 1)$	p_1; p_2, respectively
Negative single vector	$(-1, 0)$; $(0, -1)$	p_1; p_2, respectively
Averaged composition	$(\frac{1}{2}, \frac{1}{2})$	p_1, p_2
Positive/negative composition	$(1, -1)$	p_1, p_2
Reverse composition	$(-1, 1)$	p_1, p_2

The classifier experiment evaluates the following configurations:

$$\theta^{(0)} = \theta(0, 0) = \theta_0, \tag{4.7}$$

$$\theta^{(+1)} = \theta(1, 0) = \theta_0 + \boldsymbol{\tau}_1, \tag{4.8}$$

$$\theta^{(+2)} = \theta(0, 1) = \theta_0 + \boldsymbol{\tau}_2, \tag{4.9}$$

$$\theta^{(\text{avg})} = \theta(\tfrac{1}{2}, \tfrac{1}{2}) = \theta_0 + \tfrac{1}{2}\boldsymbol{\tau}_1 + \tfrac{1}{2}\boldsymbol{\tau}_2, \tag{4.10}$$

$$\theta^{(1-2)} = \theta(1, -1) = \theta_0 + \boldsymbol{\tau}_1 - \boldsymbol{\tau}_2, \tag{4.11}$$

$$\theta^{(2-1)} = \theta(-1, 1) = \theta_0 - \boldsymbol{\tau}_1 + \boldsymbol{\tau}_2, \tag{4.12}$$

$$\theta^{(-1)} = \theta(-1, 0) = \theta_0 - \boldsymbol{\tau}_1, \tag{4.13}$$

$$\theta^{(-2)} = \theta(0, -1) = \theta_0 - \boldsymbol{\tau}_2. \tag{4.14}$$

Specialisation. The positive single-vector model is compared with the base on its own class prompt. This measures whether adaptation increases the frequency of generating a sign recognised as the requested class.

Composition. The composed model uses coefficients $(0.5, 0.5)$: its weights are the arithmetic mean of the two adapted parameter vectors. Each class prompt is evaluated separately to test whether the same model can generate both classes on request. Relative to a single-vector model

at coefficient one, AVG both introduces the second task and halves the requested task's coefficient; both changes contribute to the comparison.

Positive and negative vectors. For $\theta^{(1-2)}$, recognition of c_1 is compared with the positive single-vector model to assess retention. Recognition of c_2 is compared with the base and with the negative single-vector model to characterise the effect of subtracting its vector. The reverse composition exchanges the two roles. These configurations test the interaction between a positively weighted task and a negatively weighted task.

Negation from the base. The configurations $\theta^{(-1)}$ and $\theta^{(-2)}$ test subtraction without another positive vector. Each is evaluated on the prompt for the negated class. A low recognition rate after subtraction is evidence of suppression only in relation to the corresponding baseline; it is not sufficient when the base already rarely generates that class.

4.7 Evaluation Metrics

4.7.1 Classifier-Based Evaluation

An independent ResNet-18 [13] recognises all 43 GTSRB classes. It is initialised with the Torchvision ImageNet-1K V1 weights, its final layer is replaced with a 43-output linear layer, and the entire network is fine-tuned on real traffic-sign images. It shares no weights with the generator, is not trained on generated images, and supplies neither a loss to diffusion fine-tuning nor guidance during sampling. Its parameters remain fixed throughout the generator evaluation.

Training and real-image validation. The local training archive is split by sequence, using the track identifier in each image file name and keeping every track entirely in one subset. Tracks are shuffled separately within each class with seed 2026 and approximately 20% are assigned to validation. The saved split contains 21,360 training images and 5,280 validation images. All 43 classes occur in both subsets. The separate 12,630-image test set is excluded from training and checkpoint selection.

Table 4.5: Configuration and real-image performance of the frozen evaluation classifier.

Setting	Value
Architecture	ResNet-18, 43 outputs
Initialisation	ImageNet-1K V1
Input resolution	224×224
Training / validation images	21,360 / 5,280
Optimiser / loss	AdamW / cross-entropy
Learning rate / weight decay	10^{-4} / 10^{-4}
Batch size / epochs	64 / 20
Training and split seed	2026
Checkpoint selection	Highest validation macro-recall
Selected epoch	12
Test images	12,630
Test accuracy	98.46%
Test macro-recall	97.55%

Images are converted to RGB and resized directly to 224×224 with bilinear interpolation and antialiasing. Pixel values in $[0, 1]$ are normalised with ImageNet mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$. Training augmentation uses rotations up to $10°$, translations up to 5% of the image dimensions, and brightness and contrast jitter of 0.2. There are no flips or random crops. Evaluation on real and generated images uses the same resize and normalisation, without augmentation.

The checkpoint with the highest validation macro-recall is selected, using lower validation loss to break ties. The selected checkpoint is from epoch 12. Its real-image test performance establishes a source-domain reference for the recognition model used to evaluate generated signs.

Requested-class recognition. Let $h_k(\mathbf{x})$ be the classifier's softmax output for class k, and $\hat{y}(\mathbf{x}) = \arg\max_k h_k(\mathbf{x})$. For the set $\mathcal{G}_c^{(j)}$ generated with configuration j and prompt p_c, define

$$\mathrm{Acc}^{(j)}(c) = \frac{1}{|\mathcal{G}_c^{(j)}|} \sum_{\mathbf{x} \in \mathcal{G}_c^{(j)}} \mathbf{1}[\hat{y}(\mathbf{x}) = c]. \tag{4.15}$$

There are 100 images in each evaluated configuration–prompt set. This score is the requested-class recognition rate, not a measure of visual quality, diversity or compliance with traffic-sign standards. A higher score is desirable for specialisation and retained classes; a lower score on a negated class may indicate suppression when interpreted with its baseline and the generated examples.

Paired prediction changes. Identical seeds allow reference and edited images to be compared sample by sample. Alongside recognition rates, the scoring code records the fraction whose predicted label changes, the fraction of initially correct predictions that become incorrect, and the number of initially incorrect predictions that become correct. If no reference image is classified correctly, the conditional loss-of-correctness rate is undefined. Confusions between the two evaluated classes are retained separately from predictions in any of the other 41 classes. This avoids interpreting all errors as transfer to the second task.

Recognition-rate summaries use percentile bootstrap intervals at 95%, with 10,000 resamples and bootstrap seed 2026. Differences between the full and attention-only regimes are computed with seeds paired across regimes. These pointwise intervals describe variability over generation seeds for fixed trained models; they do not include variability across independent training runs.

Interpretation of classifier scores. A closed-set classifier assigns one of 43 labels even to a malformed or unrecognisable image. Its softmax scores are not treated as calibrated probabilities of a valid sign. Classifier scores serve as a proxy for class identity, supported by real-image test accuracy and inspected alongside generated examples. Synthetic-image reliability remains a separate evaluation question.

4.7.2 LPIPS Control on Non-Traffic-Sign Prompts

The second experiment measures how single-vector edits affect prompts outside the traffic-sign adaptation domain. Its five control prompts are

`A photo of a dog`

`A photo of a car`

`A photo of an airplane`

`A photo of a house`

`A photo of an elephant`

No traffic-sign identifier is added to these prompts. Here "outside the domain" refers to the fine-tuning tasks, not to categories necessarily absent from Stable Diffusion's pre-training data.

This control uses Classic–C vectors trained for 12,000 steps, in both full and attention-only regimes. Each vector is evaluated separately with coefficients $+1$ and -1 to measure its individual effect. The control covers all eight selected classes in Table 4.1, and the reported means include all eight vectors. The per-vector comparisons examine the effect of both positive and negative edits.

Let $F(\theta; p, s)$ denote generation with U-Net parameters θ, prompt p, seed s and fixed sampling settings. For each control prompt, regime r, vector $\boldsymbol{\tau}_c^r$ and sign $\sigma \in \{-1, +1\}$, generate

$$\mathbf{x}_{p,i}^{(0)} = F(\theta_0; p, s_i), \qquad \mathbf{x}_{p,i}^{(r,c,\sigma)} = F(\theta_0 + \sigma\boldsymbol{\tau}_c^r; p, s_i), \quad s_i = 1336 + i, \quad i = 1, \ldots, 100. \tag{4.16}$$

Each edited image is paired with the base image having the same prompt and seed. The sampler is DDIM with its default deterministic step ($\eta = 0$); the same seed supplies the same initial latent noise. Pairing therefore controls the sampling input when comparing the two parameter configurations, subject to numerical differences in execution.

LPIPS [55] is computed with its AlexNet backbone. Saved RGB images are mapped to $[-1, 1]$, resized to 256×256 by bilinear interpolation with `align_corners=False`, and evaluated in float32. The control uses the whole image, with no circular mask or sign crop. Its references are generated images from the base model, not real GTSRB images.

For each prompt, the reported quantity is

$$D_{\mathrm{LPIPS}}^{(r,c,\sigma)}(p) = \frac{1}{100} \sum_{i=1}^{100} d_{\mathrm{LPIPS}}\left(\mathbf{x}_{p,i}^{(0)}, \mathbf{x}_{p,i}^{(r,c,\sigma)}\right). \tag{4.17}$$

Lower values indicate closer perceptual agreement with the base output. Each prompt mean averages 100 matched pairs. In addition to the five prompt-specific means, the control reports their equally weighted mean, equivalent here to averaging the 500 individual paired distances for a fixed vector, regime and sign. The overall mean gives equal weight to all eight vectors and five prompts, averaging

4,000 distances for each regime and sign and 16,000 distances overall. For the overall attention-only minus full differences, pointwise 95% paired-bootstrap intervals are computed by first averaging classes and prompts at each shared seed, then resampling the 100 seed means 10,000 times with bootstrap seed 2026. These intervals are conditional on the selected tasks and checkpoints and do not measure training variability.

4.7.3 Interpreting Evidence for Task Disentanglement

Task disentanglement concerns selective changes in behaviour. The classifier experiment tests whether specialisation succeeds and whether requested classes remain recognisable under the evaluated combinations. The LPIPS control tests preservation of base generations on five prompts outside the adaptation domain. These observations should be interpreted together: a small LPIPS change can result from an ineffective vector, while a useful class intervention can still cause collateral perceptual changes.

Together, the two evaluations provide behavioural evidence for task disentanglement over the selected classes, prompts and seeds. The classifier measures the requested classes under composition and subtraction, while LPIPS measures how closely each individual edit preserves base outputs elsewhere. This empirical assessment is distinct from the exact functional equality in Eq. (3.4). Selective subtraction is assessed jointly through retained-class recognition, negated-class recognition and inspection of the generated images.

4.8 Evaluation Protocol

The classifier comparison covers four recipes and the same four selected pairs at 12,000 steps. Each condition uses full and attention-only regimes. For every regime and condition, the 12 configuration–prompt sets in Table 4.4 contain 100 images each, using seeds 1337–1436.

For a given class and recipe, all arithmetic configurations, including the base, use the exact instance prompt saved with the corresponding checkpoint. Thus the base also receives the identifier when one is present. Prompt wording is fixed within a configuration comparison, but can differ between recipes. A change between recipe-specific base scores is therefore a change in conditioning, not in pre-trained weights.

Table 4.6: Sampling settings shared by the classifier evaluation and LPIPS control.

Setting	Value
Generator	Stable Diffusion 1.5
Sampler	DDIM, $\eta = 0$
Denoising steps	50
Guidance scale	7.5
Resolution	512×512
Images per prompt/configuration	100
Seeds	1337–1436
Generation batch size	4
Generation precision	Float32

The same frozen classifier and preprocessing are applied to all sign images. Base and edited generations are matched by seed, with prompts, coefficients and checkpoint identities specifying each comparison.

The LPIPS control uses its own shared cache of 500 base images: 100 for each of the five prompts. This cache is reused only under matching sampling settings. For each traffic-sign class included in the control, two regimes and two vector signs yield 2,000 matched distances in total.

Results are reported by class pair, recipe, regime and arithmetic configuration. LPIPS controls also identify the edited class, vector sign and control prompt. Matched grids show the traffic-sign content behind classifier predictions and the whole-image changes measured by LPIPS.

5. Results and Discussion

This chapter compares the four training recipes under task-vector composition and subtraction. Attention-only adaptation improves averaged composition in all 16 pair–recipe comparisons, and Classic–C achieves the highest four-pair mean. The following sections examine these results, the behaviour under subtraction, and perceptual changes on prompts outside the traffic-sign task.

5.1 Selected Class Pairs and Aggregation

The selected ordered class pairs are

$$\mathcal{P} = \{(1, 2), (12, 18), (20, 36), (13, 15)\}.$$

Within each pair, A and B denote the first and second class. Both prompts and both subtraction directions receive equal weight. Each recipe contributes one evaluation per selected pair. For recipe r, update regime m and arithmetic configuration j, let $\mathrm{Acc}_{r,m,j}(c \mid A, B)$ be the percentage of generated images classified as the requested class c. Every configuration–prompt set has 100 shared generation seeds. The pair score weights the two prompts equally:

$$M_{r,m,j}(A, B) = \tfrac{1}{2}\left[\mathrm{Acc}_{r,m,j}(A \mid A, B) + \mathrm{Acc}_{r,m,j}(B \mid A, B)\right]. \tag{5.1}$$

All four recipes cover the same four selected pairs. The reported mean gives each pair equal weight:

$$\overline{M}_{r,m,j} = \frac{1}{|\mathcal{P}|} \sum_{(A,B)\in\mathcal{P}} M_{r,m,j}(A, B). \tag{5.2}$$

The same order of averaging applies separately to retained, negated and opposite-class prediction rates under subtraction. Differences between regimes are attention-only minus full at matching recipe and pair. The same classifier and sampling settings are used throughout. Prompts and seeds match between full and attention-only within a recipe; prompt wording differs between recipes. Each category mean for a recipe and regime comprises 800 predictions: four pairs, two prompts or directions per pair, and 100 seeds per cell. Differences are calculated before rounding. Base and positive single-vector generations provide the specialisation reference.

Table 5.1: Mean classifier rates (%) by recipe and update regime. Full columns are shaded blue and attention-only columns orange. Means weight the same four selected pairs equally for all recipes. Subtraction formulas show A positively weighted and B negated; both directions are averaged. Opposite-class predictions use the negated-class prompt.

Metric	Classic–0		DB–C		Classic–C		DB–G	
	Full	**Attn-only**	**Full**	**Attn-only**	**Full**	**Attn-only**	**Full**	**Attn-only**
AVG composition (↑) $(\theta_0 + \frac{1}{2}\boldsymbol{\tau}_A + \frac{1}{2}\boldsymbol{\tau}_B)$	66.88	86.88	64.13	78.75	**69.25**	**88.00**	54.38	64.25
Retained class (↑) $(\theta_0 + \boldsymbol{\tau}_A - \boldsymbol{\tau}_B)$	68.38	72.25	68.50	74.38	**67.88**	**79.88**	59.50	52.25
Negated class (↓) $(\theta_0 + \boldsymbol{\tau}_A - \boldsymbol{\tau}_B)$	**4.88**	**3.63**	1.75	2.50	3.25	3.63	0.50	2.00
Opposite class (↓) $(\theta_0 + \boldsymbol{\tau}_A - \boldsymbol{\tau}_B)$	47.13	21.50	45.00	25.50	**46.75**	**18.13**	53.88	34.13

The positive composition uses

$$\theta_{\text{AVG}} = \theta_0 + \tfrac{1}{2}\boldsymbol{\tau}_A + \tfrac{1}{2}\boldsymbol{\tau}_B. \tag{5.3}$$

Both class prompts are tested separately, so the mean measures whether one composed model generates each class under its own prompt.

For the Classic–C and DB–C conditions pooled equally over four pairs and two recipes, requested-class recognition averages 7.00% for the base. Positive single-vector models reach 99.69% with full adaptation and 99.06% with attention-only adaptation. These scores establish successful class-specific adaptation in both regimes and provide the reference for evaluating composition. Figure 5.1 places the base, positive single-vector and AVG conditions side by side for visual inspection of specialisation and composition on $(20, 36)$.

Fine-tuning / prompt	**Base** (θ_0)		**Single +1** ($\theta_0 + \tau_c$)		**AVG** ($\theta_0 + \frac{1}{2}\tau_{20} + \frac{1}{2}\tau_{36}$)	
Seed	1337	1338	1337	1338	1337	1338
Full Class 20						
Attn-only Class 20						
Full Class 36						
Attn-only Class 36						

c: requested class. Class 20: right curve; class 36: straight or right.

Figure 5.1: Matched Classic–C generations on $(20, 36)$ at 12,000 steps. Rows show full/class 20, attention-only/class 20, full/class 36 and attention-only/class 36. Columns show the base, the positive single-vector model for the requested class and AVG composition with coefficients $(\frac{1}{2}, \frac{1}{2})$. Each cell contains seeds 1337 and 1338, left to right, using the same instance prompt.

Table 5.2: Mean AVG requested-class recognition (%) across the same four selected pairs. Δ is attention-only minus full in percentage points. All vectors use 12,000 training steps.

Metric	Classic–0		DB–C		Classic–C		DB–G	
	Full	**Attn-only**	**Full**	**Attn-only**	**Full**	**Attn-only**	**Full**	**Attn-only**
AVG composition (↑) $(\theta_0 + \frac{1}{2}\tau_A + \frac{1}{2}\tau_B)$	66.88	86.88	64.13	78.75	**69.25**	**88.00**	54.38	64.25
Difference Δ (pp) (↑)	+20.00		+14.63		+18.75		+9.88	

5.2 Effect of Training Recipe on Composition

Table 5.2 compares the training recipes under averaged task-vector composition. The comparisons examine the contribution of identifiers, the prior term and prompt wording. Pair-level examples show the class-specific behaviour behind each mean. Figure 5.2 separates the two requested classes for every pair and recipe, showing which class contributes to each improvement or imbalance in the pair mean.

5.2.1 Classic Fine-Tuning Without an Identifier

Classic–0 uses class-specific descriptions and no identifier. Its four-pair AVG mean is 66.88% with full adaptation and 86.88% with attention-only adaptation, a difference of 20.00 percentage points (pp). On $(20, 36)$ the scores are 57.5% and 81.5%, respectively. Attention-only improves AVG recognition on each of the four pairs; this advantage is present with the instance-only loss and descriptive prompts alone.

5.2.2 DreamBooth-Style Training with Distinct Identifiers

DB–C adds distinct class identifiers and a real-image prior term. Its four-pair mean is 64.13% for full and 78.75% for attention-only. On $(20, 36)$ the scores are 53.5% and 83.0%. Attention-only therefore retains its composition advantage under the DreamBooth-style recipe. Relative to

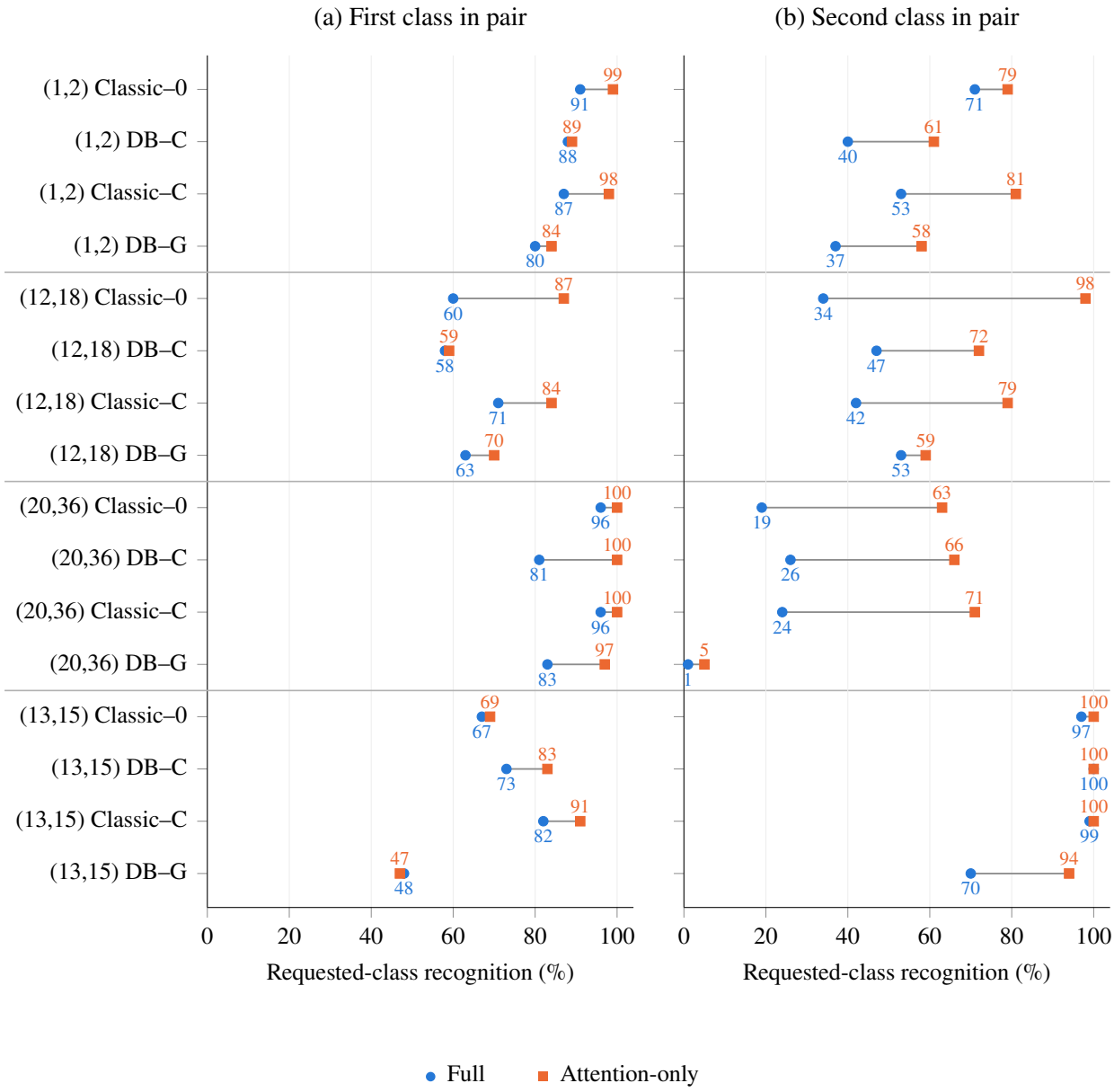


Figure 5.2: Requested-class recognition under AVG composition at 12,000 training steps. Panels separate the first and second class in each displayed pair. Blue circles show full adaptation and orange squares attention-only; grey lines connect matching evaluations. Each point represents 100 shared generation seeds under the recipe's instance prompt. AVG coefficients are $(\frac{1}{2}, \frac{1}{2})$.

Classic–0 on this pair, the full score decreases by 4.0 pp and the attention-only score increases by 1.5 pp. Since DB–C changes both the identifier and the loss, the Classic–C comparison separates their contributions.

5.2.3 Classic Training with Distinct Identifiers

Classic–C has the same class descriptions and distinct identifiers without the prior term. Its four-pair mean is 69.25% for full and 88.00% for attention-only, above DB–C in both regimes. On $(20, 36)$ it scores 60.0% and 85.5%, 2.5 and 4.0 pp above Classic–0. The Classic–C versus DB–C comparison uses the same four selected pairs and shows an advantage for Classic–C in both update regimes. Thus the highest mean composition score is achieved with the instance-only loss, class descriptions and distinct identifiers.

5.2.4 Generic Wording with an Identifier

DB–G keeps distinct identifiers but uses a generic traffic-sign instance prompt. Its four-pair AVG means are 54.38% for full and 64.25% for attention-only, a 9.88 pp difference. Attention-only improves AVG recognition on each pair, with pair scores ranging from 51.0% to 71.0%. On $(20, 36)$, the scores are 42.0% and 51.0%, respectively, below the other three recipes. Class 36 is recognised in only 1% and 5% of its requested AVG generations, while class 20 reaches 83% and 97%. This imbalance highlights the contribution of class-specific descriptions to composition with distinct identifiers.

5.3 Composition and Class Confusion

Classic–C with attention-only updates has the highest score among all four recipes on $(20, 36)$: 85.5%. DB–G reaches 42.0% with full adaptation and 51.0% with attention-only on the same pair. Classic–C's four-pair mean is 88.00%, compared with 78.75% for DB–C attention-only. Pooling Classic–C and DB–C equally, the four-pair AVG mean increases from 66.69% with full adaptation to 83.38% with attention-only adaptation, a 16.69 pp difference. This pooled comparison focuses on the two class-specific identifier recipes. For Classic–C on $(20, 36)$, class-36 recognition increases

from 24% under full adaptation to 71% under attention-only, while class 20 increases from 96% to 100%. Under DB–C, class 36 rises from 26% to 66% and class 20 from 81% to 100%. These observations support reduced behavioural interference on this pair. Across the selected pairs, attention-only improves AVG recognition for both class-specific identifier recipes.

The classifier also records whether an error is assigned to the other member of the pair or to one of the remaining 41 classes. Figure 5.3 shows this breakdown for every pair and recipe, averaging the two requested prompts. Attention-only reduces opposite-class predictions in all 16 pair–recipe comparisons, although predictions assigned to other labels sometimes increase. Together with the higher requested-class scores, this breakdown shows that attention-only composition better distinguishes the two tasks.

Figure 5.4 compares matched AVG generations across the four recipes, keeping the requested class and seed fixed while using each recipe's own instance prompt.

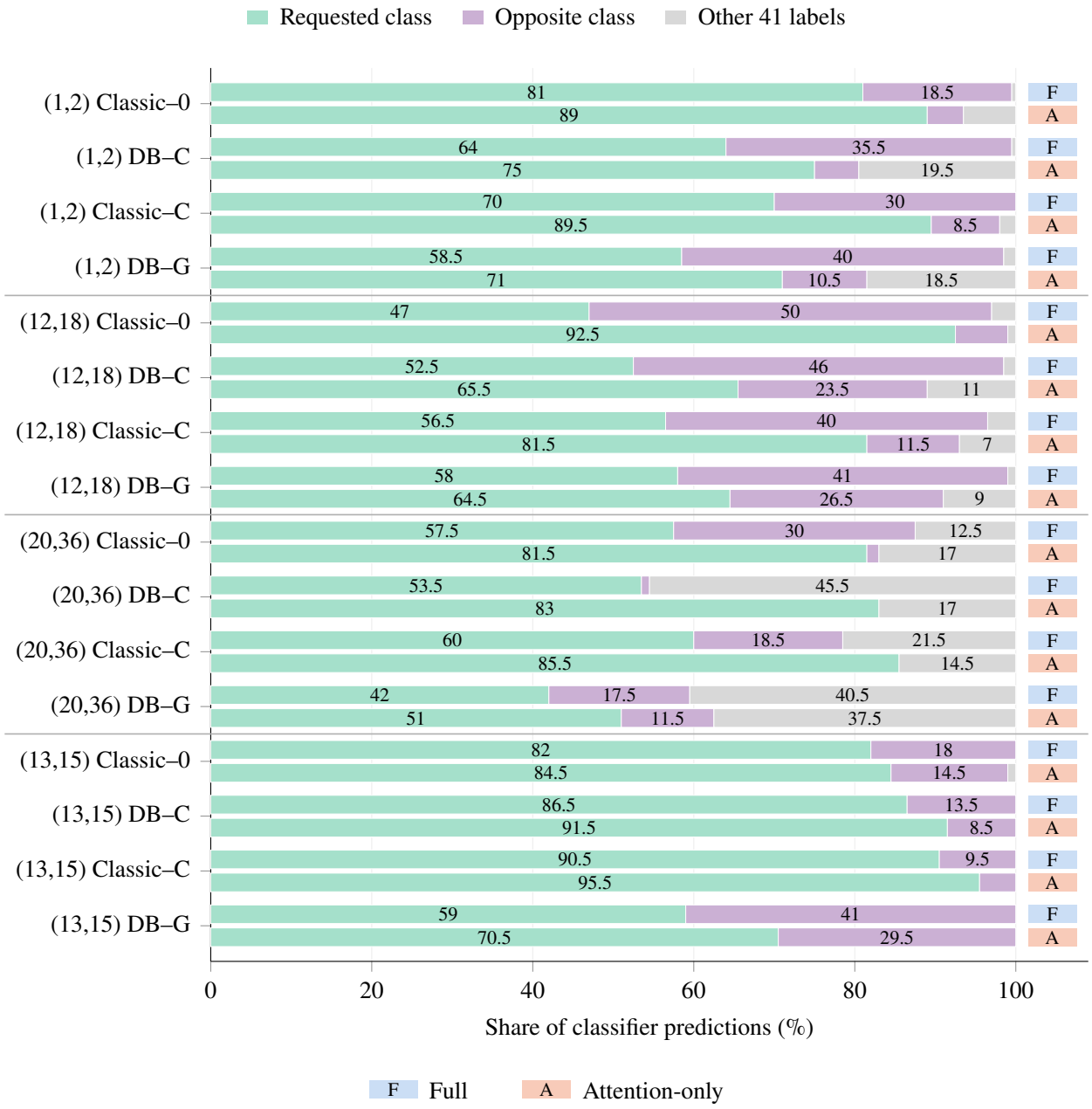


Figure 5.3: Classifier outcomes under AVG composition at 12,000 steps. Each bar pools the two class prompts with equal weight, comprising 200 predictions per pair, recipe and update regime. Green denotes the requested class, purple the opposite member of the pair and grey the remaining 41 labels. F on blue denotes full adaptation (upper bar); A on orange denotes attention-only (lower bar). Values are percentages; segments below 7% are left unlabelled. AVG coefficients are $(\frac{1}{2}, \frac{1}{2})$.

AVG $(\theta_0 + \frac{1}{2}\tau_{20} + \frac{1}{2}\tau_{36})$

	Class 20: right curve				Class 36: straight or right			
Recipe	**Full**		**Attn-only**		**Full**		**Attn-only**	
Seed	1337	1338	1337	1338	1337	1338	1337	1338
Classic–0								
DB–C								
Classic–C								
DB–G								

Figure 5.4: AVG generations on $(20, 36)$ at 12,000 steps. Rows show Classic–0, DB–C, Classic–C and DB–G. Columns show full/class 20, attention-only/class 20, full/class 36 and attention-only/class 36. Each cell contains seeds 1337 and 1338, left to right. Each recipe uses its own instance prompt; AVG coefficients are $(\frac{1}{2}, \frac{1}{2})$.

Table 5.3: Mean negative-vector rates (%) over the same four selected pairs. Retained and negated refer to the requested class under the corresponding prompt. Opposite counts the positively weighted class under the negated prompt. Formulas show A positively weighted and B negated; both directions are averaged.

Metric	Classic–0		DB–C		Classic–C		DB–G	
	Full	**Attn-only**	**Full**	**Attn-only**	**Full**	**Attn-only**	**Full**	**Attn-only**
Retained class (↑) $(\theta_0 + \tau_A - \tau_B)$	68.38	72.25	68.50	74.38	**67.88**	**79.88**	59.50	52.25
Negated class (↓) $(\theta_0 + \tau_A - \tau_B)$	**4.88**	**3.63**	1.75	2.50	3.25	3.63	0.50	2.00
Opposite class (↓) $(\theta_0 + \tau_A - \tau_B)$	47.13	21.50	45.00	25.50	**46.75**	**18.13**	53.88	34.13

5.4 Negative Vectors and Selective Control

For $\theta_0 + \tau_A - \tau_B$, selective control should retain A under its prompt and reduce B under its prompt. The reverse composition exchanges the roles. Table 5.3 separates retained-task recognition, recognition of the negated class and replacement by the opposite class.

Across the four pairs, Classic–C retains the positively weighted class in 67.88% of full-adaptation and 79.88% of attention-only generations. DB–C gives 68.50% and 74.38%, respectively. Recognition of the negated class is 3.25% and 3.63% for Classic–C, and 1.75% and 2.50% for DB–C. The principal attention-only gains in these recipes concern retention and reduced replacement by the positively weighted class; negated-class recognition remains slightly higher. The opposite-class rate under the negated prompt decreases from 46.75% to 18.13% for Classic–C and from 45.00% to 25.50% for DB–C.

The Classic–0 four-pair means show retained-class recognition of 68.38%/72.25%, negated-class recognition of 4.88%/3.63% and opposite-class predictions of 47.13%/21.50% (full/attention-only). For DB–G, the four-pair retained-class means are 59.50% for full and 52.25% for attention-only. Negated-class recognition is 0.50% and 2.00%, while opposite-class predictions are 53.88% and

34.13%, respectively. Thus its attention-only AVG advantage of 9.88 pp is accompanied by lower retained-class recognition (−7.25 pp) and higher recognition of the negated class (+1.50 pp) under subtraction.

Figure 5.5 resolves the negated-prompt outcomes by pair. On $(12, 18)$, opposite-class predictions dominate under full adaptation for all recipes. Attention-only reduces this replacement, with most of the difference assigned to the remaining 41 labels rather than the negated class.

On $(20, 36)$, retained-class recognition is 98.5%/99.0% for Classic–C and 98.5%/98.0% for DB–C (full/attention-only). Classic–0 gives 98.5%/95.5%, while DB–G gives 90.5%/78.0%. The negated class scores zero in both regimes for all four recipes. Since base recognition is already low, the main distinction between regimes is the distribution of outputs under the negated prompt. For example, requesting class 36 from $\theta_0 + \boldsymbol{\tau}_{20} - \boldsymbol{\tau}_{36}$ under Classic–C yields 95% class-20 predictions with full adaptation and 13% with attention-only, yet class-36 recognition stays at zero. Attention-only therefore reduces replacement by the positively weighted class, with the remaining predictions assigned to other labels. Pair-level analysis also reveals variation in retained-class recognition, particularly for the visually similar speed-limit pair. Figure 5.6 arranges the two subtraction directions alongside their base and single-vector references, so that retention, replacement and other outputs can be inspected separately.

The speed-limit pair $(1, 2)$ is more challenging under subtraction. Under attention-only Classic–C, its AVG score is 89.5%, but retained recognition after subtraction is only 11% for class 1 and 40% for class 2. Figure 5.7 compares the single-vector, AVG and retained-task conditions for these two classes.

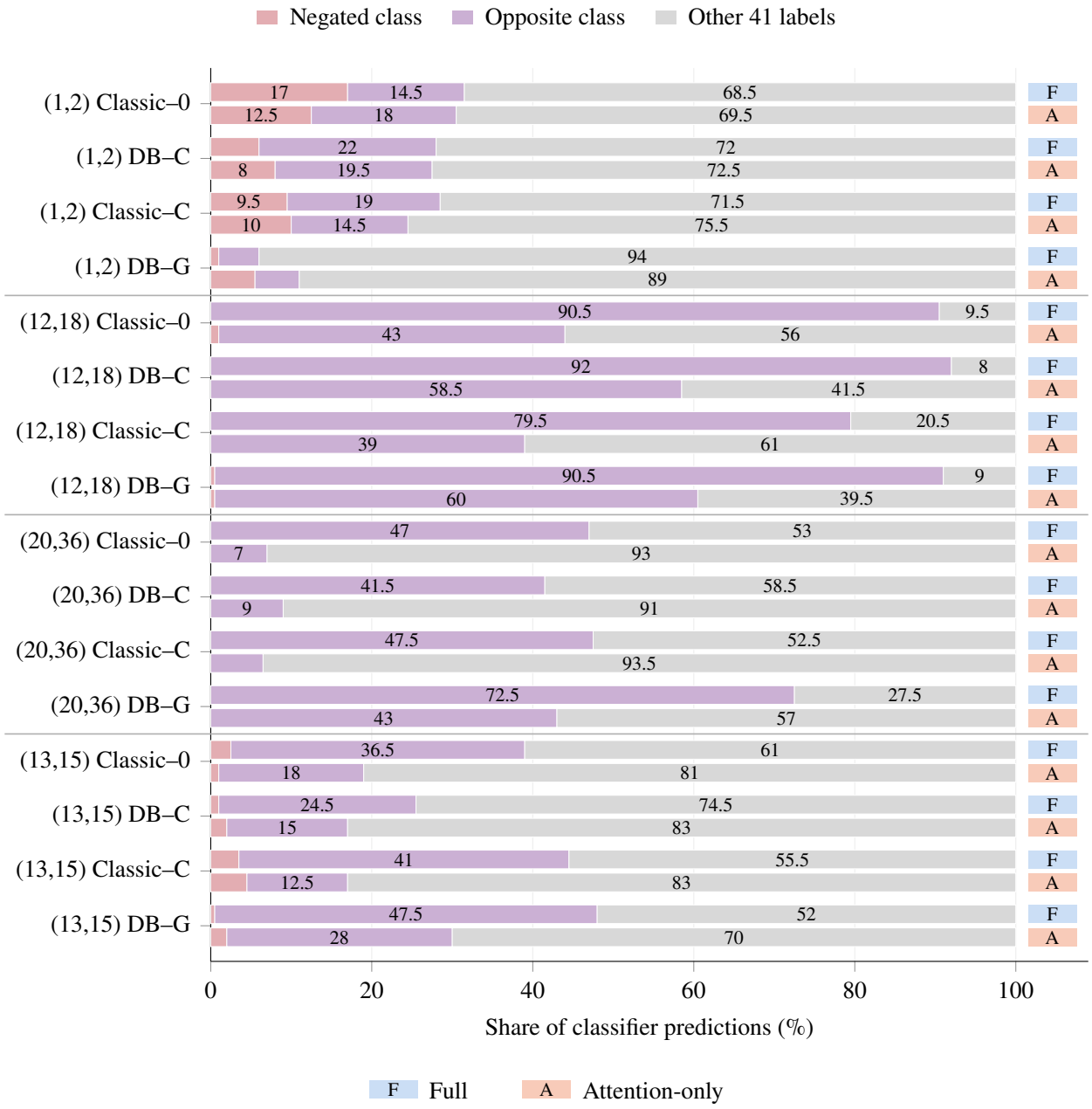
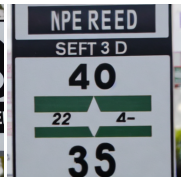


Figure 5.5: Subtraction outcomes under the negated-class prompt at 12,000 steps. Both subtraction directions are pooled equally, giving 200 predictions per pair, recipe and update regime. Red denotes recognition of the requested, negated class; purple denotes replacement by the positively weighted, opposite class; grey denotes the remaining 41 labels. F on blue denotes full adaptation (upper bar); A on orange denotes attention-only (lower bar). Values are percentages; segments below 7% are left unlabelled. Retained-task recognition is reported separately in Table 5.3.

Fine-tuning / prompt	**Base** (θ_0)		**Single +1** ($\theta_0 + \tau_c$)		**Add 20, subtract 36** ($\theta_0 + \tau_{20} - \tau_{36}$)		**Add 36, subtract 20** ($\theta_0 + \tau_{36} - \tau_{20}$)	
Seed	1337	1338	1337	1338	1337	1338	1337	1338
Full Class 20								
Attn-only Class 20								
Full Class 36								
Attn-only Class 36								

c: requested class. Class 20: right curve; class 36: straight or right.

Figure 5.6: Classic–C subtraction on $(20, 36)$ at 12,000 steps. Rows show full/class 20, attention-only/class 20, full/class 36 and attention-only/class 36. Columns show the base, the requested class's single-vector model, $\theta_0 + \tau_{20} - \tau_{36}$ and $\theta_0 + \tau_{36} - \tau_{20}$. Seeds 1337 and 1338 appear left to right within each cell. The same edited model is tested under both class prompts.

Fine-tuning / prompt	**Single +1** $(\theta_0 + \tau_c)$		**AVG** $(\theta_0 + \frac{1}{2}\tau_1 + \frac{1}{2}\tau_2)$		**Retained class** $(\theta_0 + \tau_c - \tau_d)$	
Seed	1337	1338	1337	1338	1337	1338
Full Class 1 30 km/h						
Attn-only Class 1 30 km/h						
Full Class 2 50 km/h						
Attn-only Class 2 50 km/h						

c: requested class; *d*: the other class in the pair.

Figure 5.7: Classic–C generations for speed-limit classes 1 (30 km/h) and 2 (50 km/h) at 12,000 steps. Rows show full/class 1, attention-only/class 1, full/class 2 and attention-only/class 2. Columns show the requested class's single-vector model, AVG and subtraction with the requested class positively weighted: $\theta_0 + \tau_1 - \tau_2$ for class 1, and the reverse for class 2. Seeds 1337 and 1338 appear left to right within each cell.

Table 5.4: Paired LPIPS distance from the base on five control prompts and 100 seeds per prompt using Classic–C vectors. All eight selected vectors are included, with positive and negative edits shown separately. Each class averages 500 distances per regime and sign; the overall mean averages 4,000. Δ is attention-only minus full; relative reduction is computed from unrounded means.

Class	**Positive edit: LPIPS (↓)** $(\theta_0 + \tau_c)$				**Negative edit: LPIPS (↓)** $(\theta_0 - \tau_c)$			
	Full	**Attn-only**	**Δ (↓)**	**Reduction (↑) (%)**	**Full**	**Attn-only**	**Δ (↓)**	**Reduction (↑) (%)**
1	0.5692	0.4464	-0.1229	21.6	0.4099	0.3534	-0.0565	13.8
2	0.5447	0.4410	-0.1037	19.0	0.4374	0.3494	-0.0880	20.1
12	0.5540	0.4204	-0.1336	24.1	0.4189	0.3559	-0.0630	15.0
13	0.5400	0.4095	-0.1305	24.2	0.4176	0.3286	-0.0890	21.3
15	0.6402	0.4550	-0.1852	28.9	0.4458	0.3734	-0.0724	16.2
18	0.5484	0.4763	-0.0720	13.1	0.4491	0.3619	-0.0871	19.4
20	0.5771	0.4595	-0.1176	20.4	0.4662	0.3564	-0.1098	23.5
36	0.5870	0.4723	-0.1147	19.5	0.4504	0.3624	-0.0880	19.5
Mean (8 vectors)	0.5701	0.4476	-0.1225	21.5	0.4369	0.3552	-0.0817	18.7

5.5 Control Prompts Outside the Traffic-Sign Task

LPIPS compares base and single-vector edited generations at matching prompt and seed. Lower whole-image distance means closer agreement with the base. The five control prompts are dog, car, airplane, house and elephant. Table 5.4 reports Classic–C vector-level diagnostics and the mean over all eight selected classes, 1, 2, 12, 13, 15, 18, 20 and 36, for positive and negative edits separately. Each regime–sign mean gives equal weight to class and prompt, using 4,000 paired distances. The complete control contains 16,000 distances.

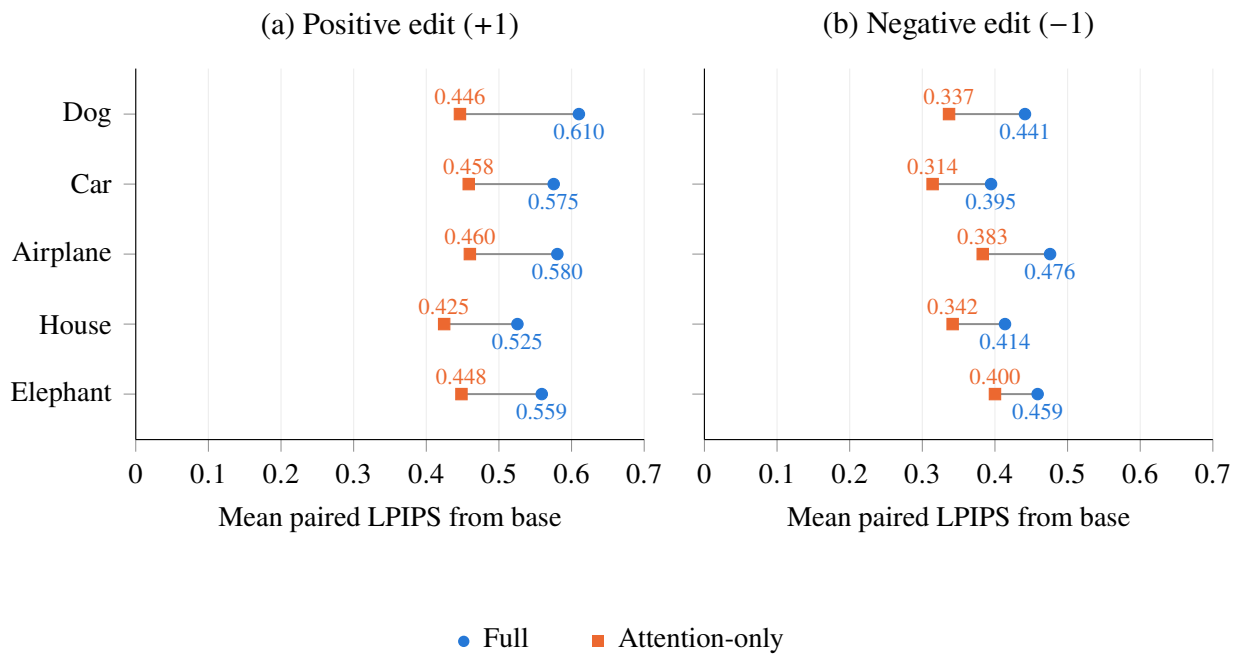


Figure 5.8: Mean paired LPIPS distance from the base for Classic–C single-vector edits at 12,000 steps. Panels separate positive and negative edits. Each point averages the eight selected class vectors and 100 matched seeds per prompt, giving 800 distances per update regime, prompt and sign. Blue circles show full adaptation and orange squares attention-only; grey lines connect the corresponding means. Lower distance indicates closer whole-image agreement with the base.

The positive-edit mean is 0.5701 for full and 0.4476 for attention-only, a relative reduction of 21.5%. Negative edits give 0.4369 and 0.3552, a relative reduction of 18.7%.

Attention-only has a lower mean distance in all 16 class–sign combinations and all 80 class–sign–prompt cells. The paired-bootstrap 95% confidence intervals for the mean differences are $[-0.1295, -0.1156]$ for positive edits and $[-0.0891, -0.0744]$ for negative edits, conditional on the evaluated tasks and checkpoints.

Figure 5.8 reports the same control by prompt, averaging all eight vectors separately for each edit sign. Attention-only has lower mean LPIPS for every prompt under both signs.

Figure 5.9 compares base and edited outputs for the class-15 vector on two of the control prompts, with the same seed and prompt across update regimes and vector signs.

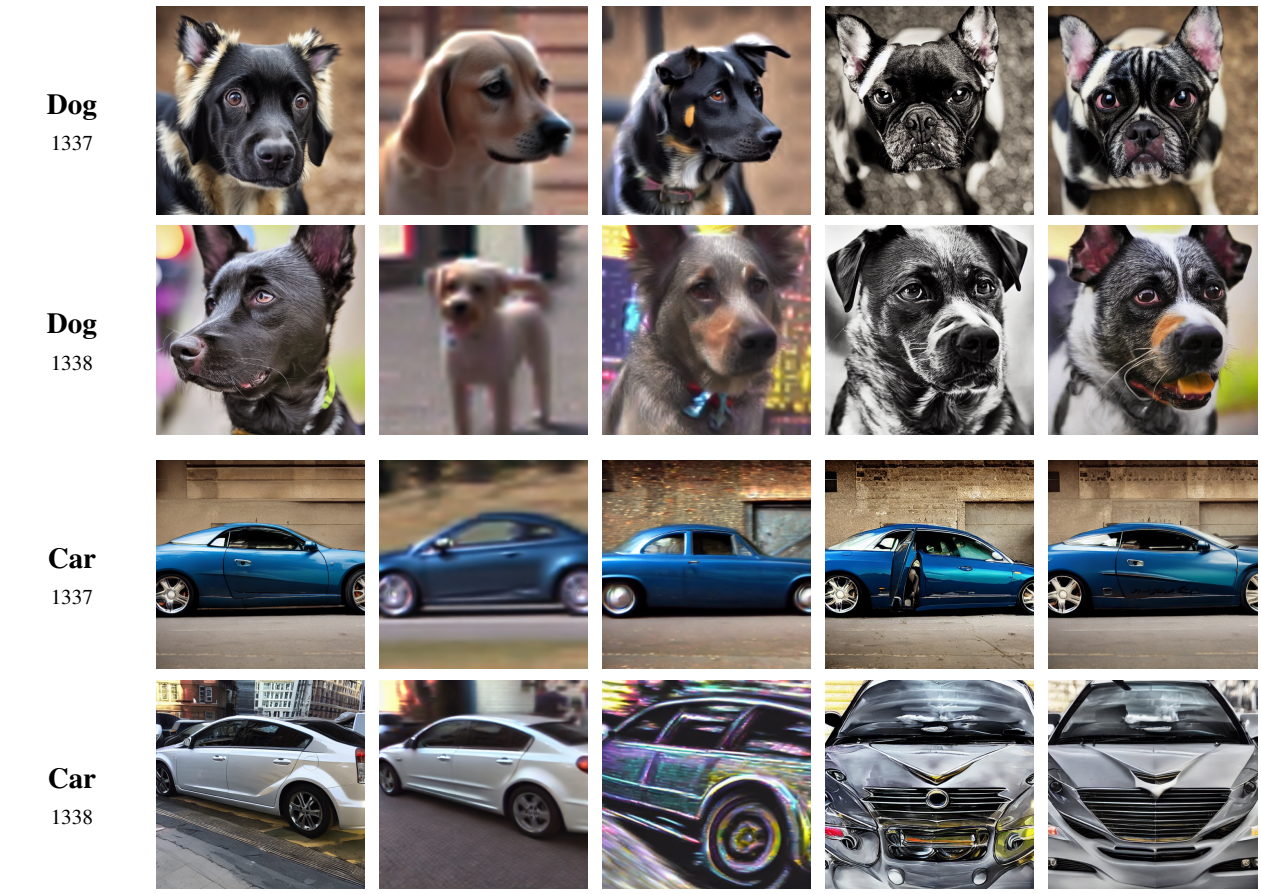


Figure 5.9: Paired LPIPS-control generations using the Classic–C class-15 vector at 12,000 steps. Rows show dog/seed 1337, dog/seed 1338, car/seed 1337 and car/seed 1338. Columns show the base, full +1, attention-only +1, full −1 and attention-only −1. Images come from the float32 control and share the same initial seed within a row. These two prompts illustrate single-vector changes; the quantitative means cover all five control prompts.

5.6 Discussion

The most consistent result is the attention-only advantage under averaged composition: all 16 pair–recipe comparisons favour this regime. The advantage holds with and without identifiers and under both training objectives. Classic–C achieves the highest four-pair mean in both regimes, showing that strong composition is attainable with the instance-only loss, class-specific descriptions and distinct identifiers. The lower DB–G scores highlight the role of descriptive conditioning in keeping related classes distinguishable.

Subtraction identifies a complementary benefit. With class-specific prompts, attention-only improves mean retention of the positively weighted class and reduces its appearance under the negated-class prompt. The speed-limit results and the lower DB–G retention show that this benefit depends on the task and conditioning. Negated-class recognition remains low in both regimes. Because the base already rarely generates some target classes, low recognition alone is insufficient to establish concept erasure; outputs assigned to other labels may also be malformed or unrelated.

The LPIPS control complements these class-level findings. Attention-only edits yield lower mean distance from the base for every class–sign–prompt combination, under both positive and negative single-vector edits. Together, stronger composition and smaller changes on control prompts support greater behavioural task disentanglement under the evaluated conditions.

The comparisons concern the specified fine-tuning configurations, including their different initial learning rates. Likewise, the gap between AVG and the positive single-vector reference reflects both combination with another task and the halved vector coefficient. Classifier scores measure class recognition rather than visual validity, and LPIPS measures single-vector changes on the five control prompts. These define the scope of the behavioural evidence; geometric separation of the vectors and selective concept erasure require separate tests.

5.7 Findings

The experiments support two broader findings within the diffusion setting and training conditions studied in this thesis:

- **Attention-only improves behavioural measures of task disentanglement.**
  - **Class composition.** After task-vector averaging, attention-only more often generates the requested class under its corresponding prompt than full fine-tuning. The improved recognition of the composed tasks is consistent with less interference between closely related concepts.
  - **Selective subtraction.** With class-specific prompts, attention-only improves mean recognition of the positively weighted class and reduces its generation under the negatively weighted class's prompt.
  - **Generation outside the target dataset.** Lower LPIPS distances for both vector signs show that attention-only edits preserve greater perceptual agreement with the base on the control prompts. This supports less disruption of generation behaviour outside the adapted task.
- **Specific class conditioning supports composition without requiring the DreamBooth-style loss.** Classic training with distinct identifiers and detailed class descriptions gives the strongest mean composition result. The DreamBooth-style prior term does not improve this result, while adding identifiers to Classic training provides a modest mean improvement. Identifiers with generic wording perform worse than those with class-specific descriptions. Together, these comparisons identify class-specific descriptions with distinct identifiers and the instance-only loss as the most effective recipe for averaged composition in this study.

6. Conclusions

This thesis asks whether task vectors learned independently for traffic-sign classes can be combined or subtracted in Stable Diffusion 1.5 while preserving the requested class. The experiments compare full U-Net and attention-only adaptation across four training recipes to assess the contribution of the update regime, class descriptions, identifiers and the DreamBooth-style prior term.

The comparison gives equal weight to four selected pairs: $(1, 2)$, $(12, 18)$, $(20, 36)$ and $(13, 15)$. Across these pairs, the 12,000-step AVG means are 69.25% versus 88.00% for full and attention-only Classic–C, and 64.13% versus 78.75% for DB–C. The Classic–0 runs give 66.88% versus 86.88% over the same four pairs. DB–G also covers all four pairs, with AVG means of 54.38% versus 64.25%, a 9.88 pp difference. Attention-only improves AVG recognition on every evaluated pair within each of the four recipes.

On $(20, 36)$, Classic–C reaches 60.0% with full adaptation and 85.5% with attention-only, the highest pair scores among the four recipes. DB–G reaches 42.0% and 51.0%, with class 36 recognised in only 1% and 5% of AVG generations. Classic–C therefore leads both on this pair and in the four-pair mean.

Under subtraction with class-specific prompts, attention-only improves mean retention of the positively weighted class and reduces its generation under the negated-class prompt. Retention is particularly strong on $(20, 36)$, while the visually similar speed-limit pair remains challenging. These results support improved control over the interaction between tasks. Establishing selective concept erasure additionally requires a baseline capable of generating the negated class and an assessment of the validity of the remaining outputs.

The Classic–C LPIPS control averages all eight selected vectors and five non-traffic-sign prompts. Positive-edit distances decrease from 0.5701 to 0.4476 with attention-only (21.5%); negative-edit distances decrease from 0.4369 to 0.3552 (18.7%). This supports closer agreement with the base under single-vector edits on the evaluated classes, prompts and seeds, complementing the stronger class-composition results.

Further experiments with matched learning rates, isolated prompt changes and repeated training seeds would help attribute the observed differences more precisely.

Generative artificial intelligence tools were used to support the drafting and linguistic revision of this thesis; the author assumes full responsibility for the accuracy of its content and the final version of the work.

Bibliography

[1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. “Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2021.

[2] Mikołaj Bińkowski et al. “Demystifying MMD GANs”. In: *International Conference on Learning Representations (ICLR)*. 2018.

[3] Yinzhi Cao and Junfeng Yang. “Towards Making Systems Forget with Machine Unlearning”. In: *IEEE Symposium on Security and Privacy*. 2015, pp. 463–480. DOI: 10.1109/SP.2015.35.

[4] Prafulla Dhariwal and Alexander Quinn Nichol. “Diffusion Models Beat GANs on Image Synthesis”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[5] Patrick Esser, Robin Rombach, and Björn Ommer. “Taming Transformers for High-Resolution Image Synthesis”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[6] Patrick Esser et al. “Scaling Rectified Flow Transformers for High-Resolution Image Synthesis”. In: *International Conference on Machine Learning (ICML)*. 2024.

[7] Jonathan Frankle et al. “Linear Mode Connectivity and the Lottery Ticket Hypothesis”. In: *International Conference on Machine Learning (ICML)*. 2020.

[8] Robert M. French. “Catastrophic Forgetting in Connectionist Networks”. In: *Trends in Cognitive Sciences* 3.4 (1999), pp. 128–135.

[9] Rinon Gal et al. “An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion”. In: *International Conference on Learning Representations (ICLR)*. 2023.

[10] Rohit Gandikota et al. “Erasing Concepts from Diffusion Models”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023.

[11] Ian J. Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[12] Yuchao Gu et al. “Mix-of-Show: Decentralized Low-Rank Adaptation for Multi-Concept Customization of Diffusion Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[13] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[14] Amir Hertz et al. “Prompt-to-Prompt Image Editing with Cross-Attention Control”. In: *International Conference on Learning Representations (ICLR)*. 2023.

[15] Martin Heusel et al. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[16] Irina Higgins et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[18] Jonathan Ho and Tim Salimans. “Classifier-Free Diffusion Guidance”. In: *NeurIPS Workshop on Deep Generative Models and Downstream Applications*. 2021.

[19] Edward J. Hu et al. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *International Conference on Learning Representations (ICLR)*. 2022.

[20] Gabriel Ilharco et al. “Editing Models with Task Arithmetic”. In: *International Conference on Learning Representations (ICLR)*. 2023.

[21] Arthur Jacot, Franck Gabriel, and Clément Hongler. “Neural Tangent Kernel: Convergence and Generalization in Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[22] Ruochen Jin et al. “Fine-Tuning Attention Modules Only: Enhancing Weight Disentanglement in Task Arithmetic”. In: *International Conference on Learning Representations (ICLR)*. 2025.

[23] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations (ICLR)*. 2014.

[24] Nupur Kumari et al. “Ablating Concepts in Text-to-Image Diffusion Models”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023.

[25] Nupur Kumari et al. “Multi-Concept Customization of Text-to-Image Diffusion”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[26] Nan Liu et al. “Compositional Visual Generation with Composable Diffusion Models”. In: *European Conference on Computer Vision (ECCV)*. 2022.

[27] Michael Matena and Colin Raffel. “Merging Models with Fisher-Weighted Averaging”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[28] Michael McCloskey and Neal J. Cohen. “Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem”. In: *Psychology of Learning and Motivation*. Vol. 24. 1989, pp. 109–165.

[29] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. “What is Being Transferred in Transfer Learning?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[30] Alexander Quinn Nichol and Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. In: *International Conference on Machine Learning (ICML)*. 2021.

[31] Maria-Elena Nilsback and Andrew Zisserman. “Automated Flower Classification over a Large Number of Classes”. In: *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*. 2008.

[32] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. “Neural Discrete Representation Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[33] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. “Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[34] Aniello Panariello et al. “Accurate and Efficient Low-Rank Model Merging in Core Space”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 38. 2025. DOI: 10.52202/085713-2066. URL: https://proceedings.neurips.cc/paper_files/paper/2025/hash/5913278289e8de3e8f9ac22b47f3ae06-Abstract-Conference.html.

[35] Aniello Panariello et al. “Modular Embedding Recomposition for Incremental Learning”. In: *British Machine Vision Conference (BMVC)*. BMVA, 2025. URL: https://bmvc2025.bmva.org/proceedings/825/.

[36] Dustin Podell et al. “SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis”. In: *International Conference on Learning Representations (ICLR)*. 2024.

[37] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *International Conference on Machine Learning (ICML)*. 2021.

[38] Aditya Ramesh et al. “Hierarchical Text-Conditional Image Generation with CLIP Latents”. In: *arXiv preprint arXiv:2204.06125* (2022).

[39] Filippo Rinaldi et al. “Gradient-Sign Masking for Task Vector Transport Across Pre-Trained Models”. In: *International Conference on Learning Representations (ICLR)*. 2026. URL: https://proceedings.iclr.cc/paper_files/paper/2026/hash/59e4f5d009f95324a1a1aa9971b51e25-Abstract-Conference.html.

[40] Filippo Rinaldi et al. “Transporting Task Vectors across Different Architectures without Training”. In: *International Conference on Machine Learning (ICML)*. Vol. 306. Proceedings of Machine Learning Research. PMLR, 2026, pp. 105193–105211. URL: https://proceedings.mlr.press/v306/rinaldi26a.html.

[41] Robin Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015.

[43] Nataniel Ruiz et al. “DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[44] Chitwan Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[45] Christoph Schuhmann et al. “LAION-5B: An Open Large-Scale Dataset for Training Next Generation Image-Text Models”. In: *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*. 2022.

[46] Viraj Shah et al. “ZipLoRA: Any Subject in Any Style by Effectively Merging LoRAs”. In: *European Conference on Computer Vision (ECCV)*. 2024.

[47] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *International Conference on Machine Learning (ICML)*. 2015.

[48] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising Diffusion Implicit Models”. In: *International Conference on Learning Representations (ICLR)*. 2021.

[49] Yang Song et al. “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *International Conference on Learning Representations (ICLR)*. 2021.

[50] Johannes Stallkamp et al. “Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition”. In: *Neural Networks* 32 (2012), pp. 323–332.

[51] Ashish Vaswani et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[52] Mitchell Wortsman et al. “Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy without Increasing Inference Time”. In: *International Conference on Machine Learning (ICML)*. 2022.

[53] Prateek Yadav et al. “TIES-Merging: Resolving Interference When Merging Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[54] Le Yu et al. “Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch”. In: *International Conference on Machine Learning (ICML)*. 2024.

[55] Richard Zhang et al. “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.