MODE CONNECTIVITY BEYOND CLASSIFIERS: EVIDENCE FROM GENERATIVE AND CONTRASTIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The loss landscape of Deep Neural Networks (DNNs) exhibits highly complex and non-convex properties. Recent studies have revealed the phenomenon of mode connectivity, demonstrating that independently trained network modes can be connected via a continuous low-loss path. However, existing mode connectivity research is predominantly confined to classifier-based models, leaving it an open question whether similar geometric properties exist in modern complex models. In this paper, we extend the boundaries of mode connectivity to generative and contrastive domains (specifically DDPM and NanoCLIP). Addressing the unique architecture of DDPM and CLIP, we propose an architecture-aware connection building algorithm. Extensive empirical results demonstrate for the first time that we successfully discover mode connectivity between independently trained DDPM and NanoCLIP modes. Our work provides a novel perspective for understanding the geometric properties of the loss landscapes in modern generative and contrastive models.

1 INTRODUCTION

The loss functions of Deep Neural Networks (DNNs) are highly complex and non-convex, and their geometric properties have long been a subject of extensive research. Early visualizations of loss functions (Foret et al., 2021; Li et al., 2018) supported the intuition that well-trained neural networks converge to isolated basins or local minima separated by high-loss barriers in the loss landscape. However, such visualizations are limited by their reliance on low-dimensional projections of the high-dimensional weight space. Further research suggests the basin model is incomplete.

Mode connectivity emerges by asking whether two independently trained neural networks can be connected by a continuous low-loss path in weight space. The endpoints of such a path are commonly referred to as *modes*, i.e., well-trained models that attain low loss. Establishing mode connectivity is significant because traditional loss landscape visualizations typically capture low-dimensional slices and therefore fail to reveal the high-dimensional low-loss regions. In contrast, mode connectivity provides a tool for uncovering deeper geometric and topological properties of the loss landscape beyond the isolated-basin picture.

The study of mode connectivity has both theoretical and empirical roots. Singular learning theory (Watanabe, 2009) provided an algebraic-geometric perspective on this phenomenon: neural networks are singular statistical models, and the degeneracy of their Fisher information matrices (Fukumizu, 1996) naturally gives rise to non-isolated parameter configurations and continuous low-loss subspaces. Empirical evidence for mode connectivity was later established. FGE (Garipov et al., 2018) showed that independently trained modes can be connected by simple curves such as polygonal chains or Bezier curves. AutoNEB (Draxler et al., 2018) iteratively bends linear interpolations into low-loss paths by optimizing intermediate pivots. More recently, LLPF (Tian et al., 2026) proposed a layer-wise algorithm, improving the reliability of connecting independently trained modes across a broader set of neural architectures.

Despite these advances, existing empirical evidence for mode connectivity remains largely confined to classifier-based settings, as shown in Table 1. Although these works cover multiple network architectures, the underlying learning paradigm is still relatively simple: unimodal supervised classi-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

fication on small-scale CIFAR datasets. Whether mode connectivity exists in modern non-classifier architectures, such as generative and contrastive models, remains largely unexplored.

In this paper, we extend the scope of mode connectivity from classifiers to Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020) on Flowers102 (Nilsback & Zisserman, 2008) and NanoCLIP (Ali-bey, 2024) on Flickr30k (Young et al., 2014). These models introduce new challenges absent from classifiers: (1) DDPM uses a U-Net architecture with skip connections, while NanoCLIP relies on cross-modal alignment between image and text encoders. All previous methods (Tian et al., 2026; Adilova et al., 2024) are not compatible with these models; and (2) these models have higher-dimensional parameter spaces than the classifiers, resulting in larger parameter-space distances between two modes. It is difficult to identify an effective optimization direction towards low-loss regions.

Our success in these models is primarily attributed to two key designs in two-stage mode connectivity pipeline (Tian et al., 2026): (1) in *Weighted Movement* stage, we develop a dataflow-based layer moving strategy that determines the moving order from the computational dependencies of each architecture, building on the layer-wise methods; and (2) in *Training Refinement* stage, we first identify that the choice of optimizer is important for higher-dimensional models. Thus, we employ the Adam optimizer (Loshchilov & Hutter, 2019) instead of Stochastic Gradient Descent (SGD), on which previous methods mainly rely.

Our main contributions are summarized as follows:

- To the best of our knowledge, we are the first to extend the scope of mode connectivity from conventional image classifiers to representative non-classifier models, including contrastive and generative models.
- We provide empirical evidence that the mode connectivity property holds for DDPM on the Flowers102 dataset and for NanoCLIP on the Flickr30K dataset.

2 Related Work

In this section, we present prior mode connectivity studies from three distinct paradigms.

2.1 Theoretical Perspectives of Mode Connectivity

Singular learning theory (SLT) (Watanabe, 2009) provides an important theoretical perspective for mode connectivity. In classical regular statistical models, different parameter values typically correspond to locally distinguishable predictive distributions, and the Fisher information matrix is non-degenerate around an optimum. Neural networks, however, are singular statistical models: the parameter-to-function map is generally non-injective due to overparameterization, permutation symmetries (Entezari et al., 2022), scaling invariances (Grigsby et al., 2023), and redundant representations (Fukumizu, 1995). Consequently, the Fisher information matrix may become degenerate near well-trained solutions, and the set of parameters realizing the same or similar functions can form continuous singular structures rather than isolated points.

From the viewpoint of SLT, the existence of flat directions and low-loss subspaces is therefore a natural consequence of the algebraic-geometric structure of neural network parameterizations. This theoretical perspective indicates that trained modes may belong to extended low-loss regions in parameter space. However, SLT does not directly provide algorithms for connecting two given independently trained modes. As a result, empirical methods remain necessary for testing whether a low-loss path exists in neural networks.

2.2 Empirical Studies of Mode Connectivity

The investigation of mode connectivity initially emerged from an empirical approach, AutoNEB (Draxler et al., 2018), which iteratively refines a linear path into a low-loss region by inserting and optimizing a sequence of intermediate pivots. It constructed continuous low-loss paths in architectures like ResNets (He et al., 2016) and DenseNets (Huang et al., 2017). Despite these advancements, previous works were unable to find a low-loss path for any arbitrary mode pairs. To address this limitation, subsequent work (Tian et al., 2026) introduced a Low-Loss Path Finding

Method	Model Architectures	Datasets
FGE	VGG, ResNet, WideResNet	MNIST, CIFAR-10, CIFAR-100
AutoNEB	Basic CNN, ResNet, DenseNet	CIFAR-10, CIFAR-100
LLPF	All above and EfficientNet, MobileNet, RegNet, ShuffleNet, DLA, CCT	MNIST, CIFAR-10, CIFAR-100, ImageNet10
Ours	DDPM, NanoCLIP	Flowers102, Flickr30k

Table 1: Comparison of model architectures and datasets in prior mode connectivity studies.

(LLPF) algorithm based on layer-wise connectivity. By moving the network layer-by-layer, their method successfully expanded to a wider range of architectures (Tan & Le, 2019; Sandler et al., 2018; Radosavovic et al., 2020; Zhang et al., 2018; Yu et al., 2018; Hassani et al., 2021) as mentioned in Table 1.

However, all of these empirical studies share a common limitation: their explorations are strictly confined to classifier-based models. The mode connectivity of generative and contrastive models remains entirely unexplored, which is the primary focus of our work.

2.3 Mode Connectivity and Linear Mode Connectivity

Although linear mode connectivity appears more direct compared to the nonlinear path, the midpoint of linear interpolation often encounters a severe loss barrier (Draxler et al., 2018). It is crucial to highlight the difference between mode connectivity and linear mode connectivity.

For mode connectivity, two independently trained modes correspond to distant and unaligned points. Finding a continuous path between them is already difficult in such a high-dimensional space. Requiring the entire path to remain inside the low-loss region makes the problem even more restrictive. As a result, existing studies in this category are primarily driven by empirical methods and remain limited in number. *Our work only investigates mode connectivity in independently trained modes.*

Linear mode connectivity is commonly observed under two settings: models either share an early portion of their optimization trajectory or are aligned with respect to parameter-space symmetries. In the former setting, models branched from a common early-stage checkpoint can be connected by a low-loss linear path (Frankle et al., 2020; Fort et al., 2020). In the latter setting, permutation-based weight matching can transform independently trained modes into aligned parameterizations that admit nearly barrier-free linear interpolation (Entezari et al., 2022; Ainsworth et al., 2023). Recently, some works have explored pre-trained language models (Qin et al., 2022; Li & Shen, 2026) and ensemble models (Kanoh & Sugiyama, 2025; Tran et al., 2025), which go beyond the paradigm of classifiers. But they still need shared pre-training trajectories or invasive weight permutations to align models to narrow the distance between two modes in the parameter space, which makes it easier to find a connected path. *In this paper, we explicitly restrict our scope to mode connectivity between independently-trained models without further permutation, i.e., models initialized with different random seeds and trained to convergence.*

3 Terminology and Definitions

To provide a unified notation for describing mode connectivity, we first define the key terms used throughout this paper.

Problem Definition. Mode connectivity can be understood as the problem of determining whether two independently trained modes can be connected by a continuous path in low-loss regions.

A *mode* refers to a well-trained parameter point that attains low loss. In this paper, we focus on independently trained modes, where different modes are obtained from different random initializations without shared early training checkpoints or weight permutation.

Given a neural network architecture $\mathcal{M}$, let $\theta \in \mathbb{R}^D$ denote its trainable parameters, where D is total number of trainable parameters. For geometric discussions, we use P to denote the point in weight space corresponding to θ. Each point $P \in \mathbb{R}^D$ is associated with a loss value $\mathcal{L}(P, \mathcal{D})$ evaluated on dataset $\mathcal{D}$. Given a loss threshold L_{thres}, we define the low-loss region in weight space as:

$$S_{L \leq L_{\text{thres}}} := \{P \in \mathbb{R}^D \mid \mathcal{L}(P, \mathcal{D}) \leq L_{\text{thres}}\} \tag{1}$$

Variance Sphere. Neural networks are composed of multiple layers, which decompose the full parameter point P as $P = [P^{(1)}, P^{(2)}, \ldots, P^{(K)}]$, where $P^{(k)} \in \mathbb{R}^{d_k}$ denotes the parameter vector of the k-th layer. Since the parameters of each layer can be viewed as a distribution of real values, we use $\text{Mean}(P^{(k)})$ and $\text{Var}(P^{(k)})$ to denote the empirical mean and variance of the k-th layer, respectively.

Based on the layer-wise variance, we define the *Variance Sphere* as the set of layer parameters with a fixed variance value. For the k-th layer and a given variance v, the variance sphere is defined as:

$$S^{(k)}_{\text{var}}(v) := \{P^{(k)} \in \mathbb{R}^{d_k} \mid \text{Var}(P^{(k)}) = v\} \tag{2}$$

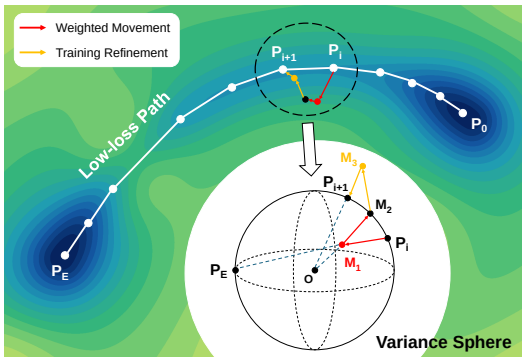


Figure 1: Geometric illustration of architecture-aware connection building algorithm. The panel shows a slice of the loss function, where the endpoint modes are connected by a low-loss path. The variance sphere below zooms into a single iteration of the algorithm. Point labels follow the algorithm's notation. The origin is used as the center of the variance sphere, assuming the mean is approximately zero (Equation 4 and Equation 5).

For independently trained modes obtained under the same model architecture, their layer-wise mean and variance are expected to be close. This is because common initialization schemes for linear, convolutional, and transformer layers usually sample weights from zero-mean distributions. Their variance is determined by the layer size and computing method Orr & Müller (1998); Glorot & Bengio (2010); He et al. (2015). Therefore, two independent initializations of the same architecture satisfy:

$$\theta_0, \theta_0' = \text{Init}(\mathcal{M}) \Longrightarrow \text{Var}(\theta_0) = \text{Var}(\theta_0') \tag{3}$$

As shown by the results in Appendix A.1, Figure 6, independently trained modes trained with identical hyperparameters on the same dataset tend to lie on nearby variance spheres, while the layer-wise means of the trained parameters typically remain near zero. Thus, for the k-th layer, we adopt the approximation:

$$\text{Mean}(P^{(k)}) \approx 0 \quad \text{and} \quad \text{Var}(\theta_n^k) \approx \text{Var}(\theta_n'^k) \tag{4}$$

Under this approximation, the variance of a layer is approximately proportional to the squared Euclidean distance from the corresponding parameter point to the origin in $\mathbb{R}^{d_k}$:

$$\left\| \overrightarrow{OP^{(k)}} \right\|^2 \propto \text{Var}(P^{(k)}) \tag{5}$$

Therefore, fixing the variance of a layer approximately fixes its distance to the origin. This explains why $S^{(k)}_{\text{var}}(v)$ can be interpreted geometrically as a high-dimensional sphere in the layer-wise parameter space.

4 Architecture-aware Connection Building Algorithm

This section presents the proposed architecture-aware connection building algorithm. We first introduce two-stage connection building framework of our algorithm and then detail two key designs in *Weighted Movement* and *Training Refinement* as shown in Figure 1.

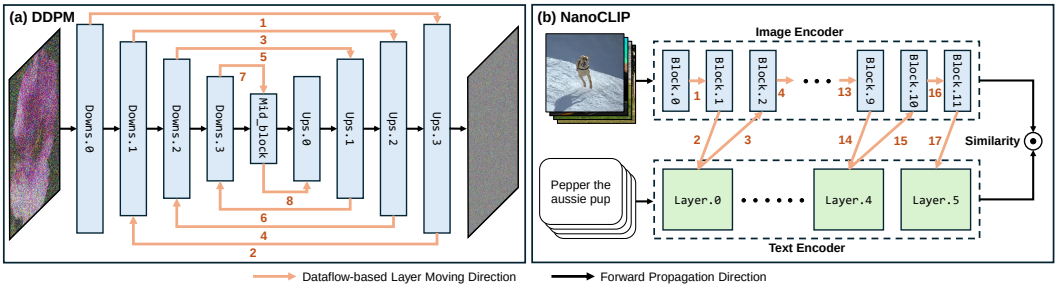


Figure 2: Dataflow-based layer moving strategies for DDPM and NanoCLIP. *Left*: DDPM layers are added to the moving sequence following a symmetrical head-to-tail schedule. *Right*: NanoCLIP layers are added through an alternating visual-textual schedule. The Layer Moving Direction indicates the order in which layer groups are introduced into layer moving sequence $\mathcal{U}_i$.

4.1 Mode-to-mode Connection Building Framework

The core objective of connection building is to iteratively transition from P_0 toward P_E while ensuring all anchor points remain within the low-loss regions, as presented by the low-loss path in Figure 1. The corresponding procedure is formalized in Algorithm 1 and consists of two stages:

Weighted Movement ($P_i \rightarrow M_2$). We first move the active layers of P_i to obtain M_1 toward the destination P_E, controlled by (α, β, γ) and $\mathcal{U}_i$. Because averaging independently trained weights reduces their layer-wise variance, it can hinder subsequent training process (Tian et al., 2024). Variance correction rescales M_1 to M_2, which lies on the reference variance sphere of P_0 as shown in Figure 1.

Training Refinement ($M_2 \rightarrow P_{i+1}$). Restoring the layer-wise variance does not by itself guarantee that M_2 lies in low-loss regions, so we then train M_2 for several rounds, yielding M_3. The variance correction is applied once more to define the next anchor P_{i+1}. Repeating this procedure for T_{total} iterations generates a sequence from P_0 to $P_T \approx P_E$, with consecutive points forming the desired low-loss path.

Algorithm 1: Architecture-aware Connection Building Algorithm

Require: Endpoint modes P_0 and P_E, dataset $\mathcal{D}$, step-size hyperparameters (α, β, γ), training hyperparameters ξ, layer moving sequence $\mathcal{U}$, total iterations T_{total}.

$P_i = P_0, \mathcal{P}.append(P_i)$
for $i = 0$ **to** $T_{total} - 1$ **do**
  $step = \alpha \left|\overrightarrow{P_i P_E}\right| + \beta \left|\overset{\frown}{P_0 P_E}\right| + \gamma$
  $M_1 = P_i + step \cdot \Pi_{\mathcal{U}_i}\left(\overrightarrow{P_i P_E}\right)$
  $M_2 = \text{VarianceCorrection}(M_1, S_{Var})$
  $M_3 = \text{Train}(M_2, \xi, \mathcal{D})$.
  $P_{i+1} = \text{VarianceCorrection}(M_3, S_{Var})$
  $\mathcal{P}.append(P_{i+1})$.
end for
return $\mathcal{P}$

Function VarianceCorrection(Q, S_{Var})
for $W = Q^{(k)}$ in Q **do**
  $\bar{W} = \frac{1}{n}\sum_{j=1}^{n} W[j]$
  $\sigma_W^2 \leftarrow \frac{1}{n}\sum_{j=1}^{n}(W[j] - \bar{W})^2$
  for $j = 1$ to len(W) **do**
    $W'[j] = \bar{W} + \sqrt{\frac{v}{\sigma_W^2}}(W[j] - \bar{W})$
  end for
  $Q'^{(k)} = W'$
end for
return Q'

4.2 Dataflow-based Layer Moving Strategy

The most critical hyperparameter of *Weighted Movement* is the layer moving sequence $\mathcal{U}_i$, which decides the subset of layers needed to move at a stage. We construct layer moving strategy that follows two rules: (1) layers should be introduced in the direction of data flow according to model architectures; and (2) attention modules should be processed individually before proceeding to subsequent layers. The dataflow direction is interpreted through the feature-processing hierarchy and structural correspondences within the architecture, rather than simple order of forward execution.

For standard acyclic sequential architectures, these rules allow a moving sequence to be generated automatically from the computational graph. For more complex architectures, the same rules are applied by identifying the corresponding feature-processing stages, as illustrated by DDPM and

NanoCLIP below. Our empirical evaluation focuses on DDPM and NanoCLIP, and validating the effectiveness of the strategy on additional architectures is beyond the scope of this work.

The U-Net used in DDPM contains corresponding downsampling and upsampling blocks that are linked by skip connections and handle features at identical spatial resolutions. Applying Rule (1), we organize these corresponding blocks into shared resolution levels of the feature-processing hierarchy. At each level, encoder and decoder blocks are introduced successively before proceeding to the next, lower-resolution level. Attention modules are introduced individually according to Rule (2). This produces an outside-in order that alternates between the encoder and decoder, progressing from the outermost layers toward the innermost layers, as shown in Figure 2.

NanoCLIP consists of image and text encoders whose outputs are compared in a shared embedding space. Applying Rule (1) to these parallel feature-processing streams, we advance both encoders through their respective depths rather than complete one encoder before starting the other. Rule (2) requires attention modules to be introduced individually, so this coordinated progression is implemented by interleaving blocks from the two encoders instead of introducing them simultaneously. In the evaluated configuration, two visual encoder blocks are introduced individually, followed by one text encoder layer, and this pattern repeats across the available layers, as shown in Figure 2.

4.3 Adam-based Refinement for Distant Modes

The purpose of *Training Refinement* is to return the variance-corrected point M_2 to the low-loss region after each weighted movement. In the classifier settings, the movements are typically short and localized. Under this condition, a SGD optimizer is often sufficient to identify a nearby descent direction and recover low loss. The DDPM and NanoCLIP models operate in substantially higher-dimensional parameter spaces than the classifier-based models. As a result, independently trained checkpoints exhibit larger parameter distances, including larger layer-wise separations. Empirically, we find that the single global learning-rate scaling of SGD becomes unreliable in this regime: SGD may reduce the loss for small endpoint separations, but frequently fails to restore low loss when the checkpoint separation is large.

We address this limitation by implementing training refinement with Adam. Its first and second moment estimates provide coordinate-wise adaptive scaling, which is useful when different modules exhibit substantially different gradient magnitudes. The momentum state also accumulates a stable refinement direction across successive updates. During each refinement phase, Adam keeps updating the parameters until the training loss falls below the threshold or the maximum number of refinement rounds is reached. The ablation in Figure 5 supports this design: replacing Adam with SGD while retaining other components prevents the refinement step from maintaining a low-loss path.

5 Experimental Evaluation

We evaluate the proposed architecture-aware connection building algorithm on DDPM trained on Flowers102 and NanoCLIP trained on Flickr30k. For each model, we consider three distinct pairs of independently trained endpoint modes, with the two modes in each pair trained using different random seeds. Figure 3 presents the result for Pair 1 with other pairs in Appendix A.5, Figure 9. Table 2 summarizes results for the three mode pairs. The layer-moving schedules and experimental configurations are described in Appendix A.4, Table 4.

Overall Performance. We evaluate the validity of the constructed paths from three complementary perspectives: (1) the layer-wise L_2 distance between the current point P_i and the end mode P_E should decrease progressively and eventually approach the zero. Small deviations are acceptable because each stage only approximates the ideal movement; (2) the training loss should remain within the low-loss threshold throughout the connection process; and (3) the test evaluation metrics at the end of the path should converge to those of the starting mode P_0, indicating that the constructed path not only reaches the target parameter region but also recovers its generalization performance.

The first row of Figure 3 shows the evolution of the layer-wise L_2 distances. For both DDPM and NanoCLIP, the distances decrease stage by stage following the strategy and reduced by at least one order of magnitude compared with their initial values. It is expected that the final distances of several layers do not converge exactly to zero, because different layers contain substantially different

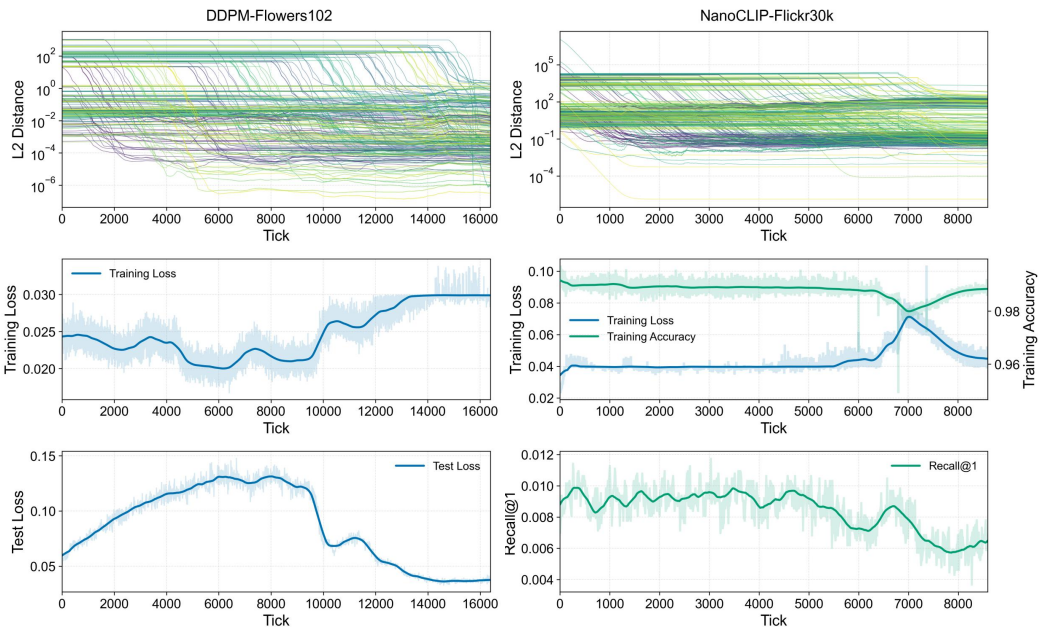


Figure 3: Results of the architecture-aware connection building algorithm on DDPM (first column) and NanoCLIP (second column). The first row shows the layer-wise L_2 distance from the current point P_i to the end mode P_E, and the legend of layers is omitted due to excessive quantity. The second and third rows respectively report the training and test metrics along the path. The curves show the smoothed trends, and the shaded regions indicate local fluctuation envelopes of the raw metrics. Due to the learning objectives of the models, DDPM is evaluated only with training and test losses, whereas NanoCLIP is evaluated without a test loss.

numbers of parameters, leading to different distance scales. To make it comparable across pairs, Table 2 reports the final relative L_2 distance, $R_{L_2} = \|P_T - P_E\|_2 / \|P_0 - P_E\|_2$, where P_T is the final constructed checkpoint and P_0 and P_E are the original starting and end modes, respectively. Across the three pairs, the final relative distances of every reported are below 0.2% of its initial separation, providing evidence of close approach to the target.

Table 2 summarizes the training-loss behavior across the three pairs. For both DDPM and NanoCLIP, mean training losses maintain steady low level. Although the maximum loss for NanoCLIP Pair 1 appears to be higher than the other two pairs, it is only a special point with its second largest loss being 0.0828. The third row of Figure 3 also shows the test loss and $Recall@1$ at the end converge to those of the starting mode. These results demonstrate that the proposed algorithm is capable of constructing low-loss paths while preserving their generation capability.

Because the procedure produces a discrete sequence of checkpoints, we further verify that the line segments between consecutive checkpoints do not introduce hidden loss barriers. As detailed in Appendix A.2, dense linear interpolation along the DDPM and NanoCLIP trajectory reveals no observable barrier, providing empirical evidence that the path remains within the low-loss region.

Table 2: Path statistics for three mode pairs per model. All mode pairs are all different from each other. R_{L_2} is the residual distance ratio. Smaller values indicate closer proximity to the target. All the TL in table represents training loss.

	DDPM			NanoCLIP		
Pair	R_{L_2} (%)	Mean TL	Max TL	R_{L_2} (%)	Mean TL	Max TL
1	0.1794	0.0247	0.0339	0.1163	0.0441	0.1038
2	0.1071	0.0264	0.0330	0.1083	0.0455	0.0896
3	0.1385	0.0261	0.0337	0.1065	0.0444	0.0879

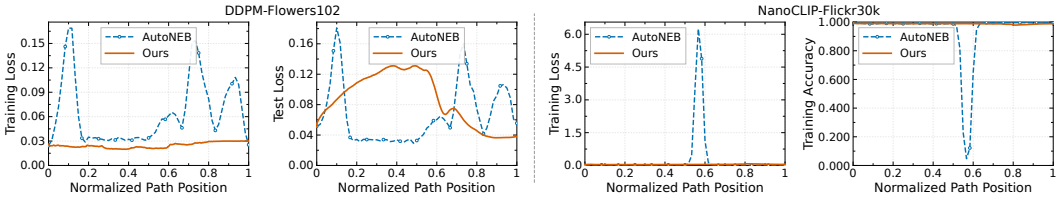


Figure 4: Comparison of AutoNEB and our method along constructed paths on DDPM and NanoCLIP. From left to right, the four panels report DDPM training loss, DDPM test loss, NanoCLIP training loss, and NanoCLIP training accuracy, respectively. The horizontal axis denotes the normalized path position, where 0 and 1 correspond to the starting and end modes. The dots on the dashed line are the pivots optimized by AutoNEB. Between adjacent pivots, we also evaluate the metrics of interpolation points along each segment. For visual clarity, the densely sampled curves of our method are smoothed.

Comparison Study. Among a limited number of mode connectivity methods, FGE, AutoNEB and LLPF are the relevant baselines for comparison. However, the released implementation of FGE contains an error, and its procedure does not necessarily yield a valid low-loss path (Tian et al., 2026). Our method is based on the layer-wise paradigm of LLPF, so the comparative results with LLPF can actually refer to the ablation results combined with forward order and SGD. We therefore select AutoNEB as the primary baseline. For a controlled comparison, we follow the original AutoNEB training configuration and adapt its pipeline to accommodate DDPM and NanoCLIP as shown in Appendix A.3, Table 3.

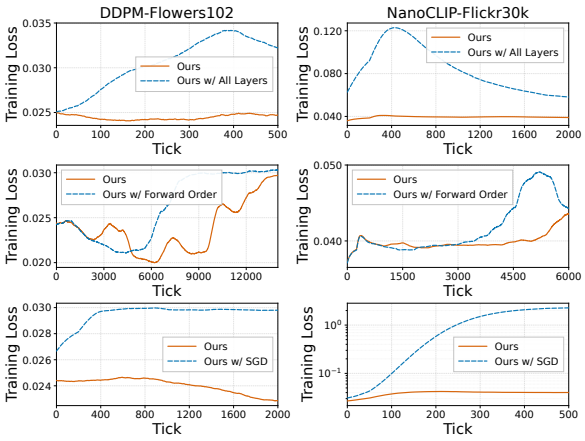


Figure 5: Ablation study of the proposed architecture-aware connection building algorithm on DDPM and NanoCLIP. The left and right columns report the results on DDPM and NanoCLIP, respectively. The first row compares the proposed dataflow-based layer moving strategy with simultaneous movement of all layers. The second row compares the proposed strategy with the forward propagation order. The third row replaces Adam with SGD while keeping the remaining components unchanged. All curves are smoothed.

Figure 4 compares AutoNEB with our method along the constructed paths. On DDPM, the maximum training and test losses obtained by AutoNEB are 0.17 and 0.18, respectively, whereas our method limits them to 0.03 and 0.15. The difference is substantially larger on NanoCLIP: AutoNEB reaches a maximum training loss of 6.69 and a minimum training accuracy of only 0.05, while our method achieves a maximum training loss of 0.09 and maintains a minimum training accuracy of 0.95. These results show that AutoNEB fails to construct a low-loss path for generative and contrastive models. Our method maintains consistently low training loss throughout the path.

Ablation Study. We examine two key designs of our method: dataflow-based layer moving strategy and Adam-based training refinement. Figure 5 compares the complete method with three variants: *Ours w/ All Layers*, which moves all model layers simultaneously; *Ours w/ Forward Order*, which retains staged movement but activates layers according to the forward propagation order; and *Ours w/ SGD*, which replaces Adam with SGD. All remaining components and hyperparameters are kept unchanged unless otherwise specified. It is worth noting that we did not construct a complete path in all experiments, as sufficient differences could already be observed within limited ticks.

The first row of Figure 5 investigates whether our moving strategy can be replaced by moving all layers simultaneously. On DDPM, the all-layers variant yields an average training loss of 0.031,

compared with 0.025 of our method. On NanoCLIP, the average training loss of the variant is more than twice that of ours. It indicates moving all layers increases the optimization difficulty.

The second row compares the proposed strategy with the forward propagation order schedule. Our strategy outperforms the forward-order variant on both DDPM and NanoCLIP, reducing the average training loss by 0.002 in each case. This indicates that our dataflow-based order provides a more effective movement schedule than computational order for movement in weight space.

The third row evaluates the optimizer used during training refinement. On DDPM, the complete method achieves an average training loss of 0.024, compared with 0.030 for the SGD variant. The difference is more pronounced on NanoCLIP, where its training loss can increase by two orders of magnitude. These results indicate that the coordinate-wise adaptive updates of Adam improve refinement stability when different models exhibit substantially different gradient magnitudes. Overall, the ablation results show that Adam-based refinement and dataflow-based layer moving strategy provide complementary benefits for preserving low training loss along the constructed paths.

6 Discussion

Beyond validating the effectiveness of the proposed method, our empirical results also reveal several observations which is worthy of further discussion.

Low training loss does not imply high generation quality. Although mode connectivity is defined primarily by low training loss, examining the image generation capability of intermediate DDPM models provides a complementary assessment. We uniformly sample 41 checkpoints along the path, including both endpoints. Each checkpoint generates 1000 images under an identical sampling configuration and random seed, so corresponding samples differ only in model parameters. Figure 10 in Appendix A.6 shows one representative image from each checkpoint together with its Fréchet Inception Distance (FID). Quantitatively, the intermediate checkpoint generally produces higher FID values than the endpoints, while the training loss of the checkpoints remains basically the same as that of the endpoint. This behavior is expected because the intermediate models are optimized to satisfy the low-loss constraint rather than to minimize FID directly. The results may also indicate that a low-loss region is not necessarily a high-quality generative region.

Endpoint parameter proximity does not imply a flat basin. Prior studies describe well-trained minima as lying within locally flat manifolds. Under this interpretation, once a constructed trajectory approaches the end mode, the remaining neighborhood should be sufficiently flat that direct interpolation introduces little additional loss. We test this expectation around the DDPM endpoint. As shown in the left panel of Figure 11 in Appendix A.7, the checkpoint at tick 16400 is already strongly aligned with the end mode: layer-wise cosine similarities are concentrated near one across all DDPM modules, indicating that the checkpoint is close to the endpoint in weight space. Nevertheless, the right panel shows that linear interpolation between them creates a maximum training loss of 0.038, compared with a maximum of 0.034 along the previous low-loss path. One possible interpretation is that the low-loss region near the end mode is highly anisotropic: even near a trained mode, the local geometry of foundation-model loss landscapes may be more intricate than the conventional picture of a uniformly flat basin.

7 Conclusion

We investigated whether mode connectivity extends beyond classifiers to independently trained models with generative and contrastive objectives. To address the architectural and optimization challenges posed by DDPM and NanoCLIP, we proposed an architecture-aware connection building algorithm that combines dataflow-based layer movement with Adam-based refinement. Experiments on DDPM trained on Flowers102 and NanoCLIP trained on Flickr30k show that the proposed method constructs continuous low-loss paths while progressively reducing layer-wise parameter distances and maintaining low training loss. Overall, our work is the first to extend mode connectivity to generative and contrastive domains and motivate further studies of loss-landscape geometry in modern non-classifier models.

AI USE STATEMENT

In this work, we used generative AI tools (specifically ChatGPT / Gemini) for text polishing, grammar checking, and assisting in basic script formatting. We have not used generative AI tools for core research idea generation, mathematical proofs, experimental design, or result analysis, and human-subject data collection disclosures are not applicable to this work. Additionally, we used generative AI tools for improving the readability of manuscript drafts. We have manually reviewed, verified, and edited all AI-assisted text and code to ensure technical correctness and accuracy. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we have taken the following measures:

Code and Datasets: The complete source code, preprocessing scripts, and environment setups are provided in the supplementary materials (and an anonymous GitHub link will be made public upon publication).

Experimental Details: Detailed hyperparameter configurations, dataset split details, and training schedules are elaborated in Appendix A.3 and A.4.

REFERENCES

Linara Adilova, Maksym Andriushchenko, Michael Kamp, Asja Fischer, and Martin Jaggi. Layer-wise linear mode connectivity. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LfmZh91tDI.

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Amar Ali-bey. nanoCLIP: A lightweight text-to-image retrieval model. https://github.com/amaralibey/nanoCLIP, 2024. GitHub repository, accessed July 14, 2026.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1309–1318. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/draxler18a.html.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3259–3269. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/frankle20a.html.

Kenji Fukumizu. Active learning in multilayer perceptrons. In D. Touretzky, M.C. Mozer, and M. Hasselmo (eds.), *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995. URL https://proceedings.neurips.cc/paper_files/paper/1995/file/8248a99e81e752cb9b41da3fc43fbe7f-Paper.pdf.

Kenji Fukumizu. A regularity condition of the information matrix of a multilayer perceptron network. *Neural Networks*, 9(5):871–879, 1996. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(95)00119-0. URL https://www.sciencedirect.com/science/article/pii/0893608095001190.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 8803–8812, Red Hook, NY, USA, 2018. Curran Associates Inc.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

J. Elisenda Grigsby, Kathryn Lindsey, and David Rolnick. Hidden symmetries of relu networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *ArXiv*, abs/2104.05704, 2021. URL https://api.semanticscholar.org/CorpusID:233210459.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015. doi: 10.1109/ICCV.2015.123.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017. doi: 10.1109/CVPR.2017.243.

Ryuichi Kanoh and Mahito Sugiyama. Linear mode connectivity in differentiable tree ensembles, 2025. URL https://arxiv.org/abs/2405.14596.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf.

Tianyi Li and Zhiqiang Shen. Scaling linear mode connectivity and merging to billion parameter pretrained transformers, 2026. URL https://arxiv.org/abs/2606.23607.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Genevieve B. Orr and Klaus-Robert Müller (eds.). *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS workshop*, Berlin, Heidelberg, 1998. Springer-Verlag. ISBN 3540653112.

Yujia Qin, Cheng Qian, Jing Yi, Weize Chen, Yankai Lin, Xu Han, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Exploring mode connectivity for pre-trained language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6726–6746, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.451. URL https://aclanthology.org/2022.emnlp-main.451/.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing Network Design Spaces . In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10425–10433, Los Alamitos, CA, USA, June 2020. IEEE Computer Society. doi: 10.1109/CVPR42600.2020.01044. URL https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01044.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks . In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, Los Alamitos, CA, USA, June 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00474. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00474.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ArXiv*, abs/1905.11946, 2019. URL https://api.semanticscholar.org/CorpusID:167217261.

Yongding Tian, Zaid Al-Ars, Maksim Kitsak, and Peter Hofstee. Vanishing variance problem in fully decentralized neural-network systems, 2024. URL https://arxiv.org/abs/2404.04616.

Yongding Tian, Zaid Al-Ars, Maksim Kitsak, and H Peter Hofstee. Connecting independently trained modes via layer-wise connectivity. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4VOTzpH9MO.

Viet-Hoang Tran, Van-Hoan Trinh, Khanh Vinh Bui, and Tan Minh Nguyen. On linear mode connectivity of mixture-of-experts architectures. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=RF3miSqdXa.

Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, USA, 2009. ISBN 0521864674.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep Layer Aggregation . In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2403–2412, Los Alamitos, CA, USA, June 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00255. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00255.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices . In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6848–6856, Los Alamitos, CA, USA, June 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00716. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00716.

A Appendix

A.1 Empirical Validation of Equation 4

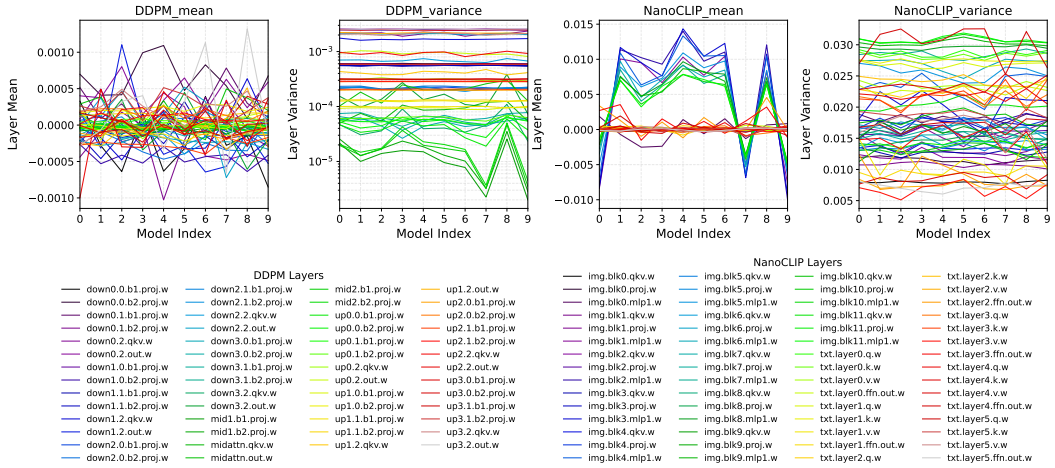


Figure 6: Layer-wise weights mean and variance across independently trained NanoCLIP on Flickr30k and DDPM on Flowers102 models with different random seeds. The x-axis denotes the model index, and the y-axis represents the mean or variance of layer parameters. Each curve corresponds to one layer following the PyTorch naming convention. Batch normalization layers are excluded, as their statistics are highly sensitive to the training batches. The results show that layer-wise variances remain consistent across independent training runs and layer-wise means remain centered around zero.

A.2 Mode-Connection Continuity Check via Linear Interpolation

Figure 7 and 8 presents the resulting loss trajectory. Despite the substantially denser sampling, no observable loss barrier appears along the constructed path. Compared with the interpolated paths with original paths, the maximum training loss increases by less than 9.7% and 5.3% for DDPM and NanoCLIP, respectively. The result indicates that the interpolated models remain within the same low-loss region as the optimization trajectory.

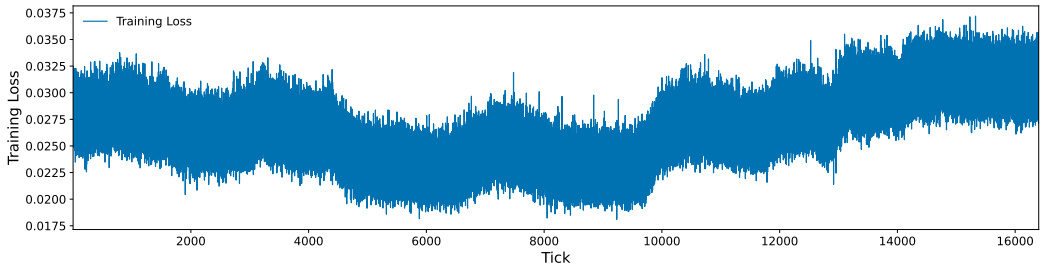


Figure 7: Mode-connection continuity check for DDPM. Twenty uniformly spaced interpolation points are inserted between every pair of consecutive optimization ticks. All interpolation points also remain within the low-loss region.

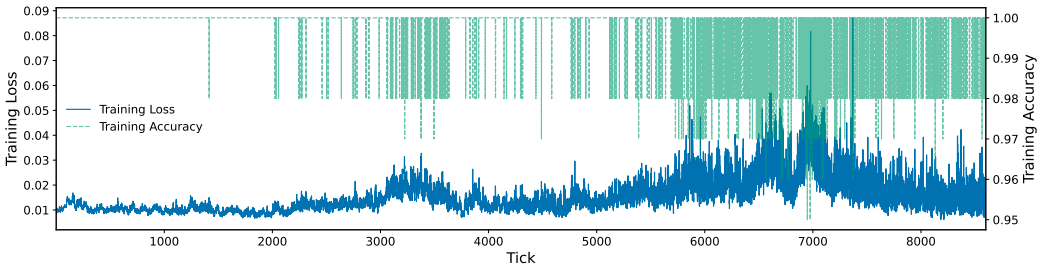


Figure 8: Mode-connection continuity check for NanoCLIP. Twenty uniformly spaced interpolation points are inserted between every pair of consecutive optimization ticks. All interpolation points also remain within the low-loss region.

A.3 Hyperparameters and Configurations of AutoNEB

Cycle	Updates per Pivot	Trainable Internal Pivots	Optimizer and Hyperparameter
Model: DDPM-Flowers102		Runtime: 23 hours (NVIDIA A6000)	
Model: NanoCLIP-Flickr30k		Runtime: 41 hours (NVIDIA A6000)	
1	1000	1	
2	1000	3	
3	1000	7	AdamW
4	1000	11	(lr=$1e^{-3}$, wd=0)
5	2000	11	
6	2000	11	
7	1000	11	
8	1000	11	AdamW
9	1000	11	(lr=$1e^{-4}$, wd=0)
10	1000	11	
11	1000	11	
12	1000	11	AdamW
13	1000	11	(lr=$1e^{-5}$, wd=0)
14	1000	11	

Table 3: Hyperparameters used for the AutoNEB baseline. Both DDPM and NanoCLIP use the same 14-cycle optimization schedule. Updates per pivot denotes the number of AdamW updates applied to each trainable internal pivot in a cycle. The two endpoint pivots remain fixed and are excluded from the pivot counts and the final path contains 13 pivots in total. AdamW states are reset at the beginning of each cycle. We also report the total runtime.

A.4 Hyperparameters and Configurations of Architecture-aware Connection Building Algorithm

Stage	Layer Moving Sequence $\mathcal{U}$	Iteration T	Training Refinement Round r	Optimizer and Hyperparameter	Step-size Hyperparameters (α, β, γ)
	Model: DDPM-Flowers102		Runtime: 59 hours (NVIDIA RTX 5090)		
1	init_conv.{weight+bias}	400	Train until loss< 0.03 or rounds> 500	AdamW (lr=$1e^{-6}$, wd=0)	(0.001, 0.002, 0)
2	Stage 1 + mlp + downs.0.0	400			
3	Stage 2 + downs.0.1	400			
4	Stage 3 + downs.0.2	400			
5	Stage 4 + downs.0.3	400			
6	Stage 5 + ups.3.0	400			
7	Stage 6 + ups.3.1	400			
8	Stage 7 + ups.3.2	400			
9	Stage 8 + ups.3.3	400			
10	Stage 9 + final_res_block	400			
11	Stage 10 + final_conv	400			
12	Stage 11 + downs.1.0	400			
13	Stage 12 + downs.1.1	400			
14	Stage 13 + downs.1.2	400			
15	Stage 14 + downs.1.3	400			
16	Stage 15 + ups.2.0	400			
17	Stage 16 + ups.2.1	400			
18	Stage 17 + ups.2.2	400			
19	Stage 18 + ups.2.3	400			
20	Stage 19 + downs.2.0	400			
21	Stage 20 + downs.2.1	400			
22	Stage 21 + downs.2.2	400			
23	Stage 22 + downs.2.3	400			
24	Stage 23 + ups.1.0	400			
25	Stage 24 + ups.1.1	400			
26	Stage 25 + ups.1.2	400			
27	Stage 26 + ups.1.3	400			
28	Stage 27 + downs.3.0	400			
29	Stage 28 + downs.3.1	400			
30	Stage 29 + downs.3.2	400			
31	Stage 30 + downs.3.3	400			
32	Stage 31 + mid_block1	400			
33	Stage 32 + mid_attn	400			
34	Stage 33 + mid_block2	400			
35	Stage 34 + ups.0.0	400			
36	Stage 35 + ups.0.1	400			
37	Stage 36 + ups.0.2	400			
38	Stage 37 + ups.0.3	400			
39	All layers	1200			
	Model: NanoCLIP-Flickr30k		Runtime: 77 hours (NVIDIA RTX 5090)		
1	txt.{embeddings+pooler+fc} img.{patch_embed+cls+pos+mask+fc}	400	Train until loss< 0.04 or rounds> 200	AdamW (lr=$1e^{-4}$, wd=0)	(0.005, 0, 0)
2	Stage 1 + blocks.0	400			
3	Stage 2 + blocks.1	400			
4	Stage 3 + layer.0	400			
5	Stage 4 + blocks.2	400			
6	Stage 5 + blocks.3	400			
7	Stage 6 + layer.1	400			
8	Stage 7 + blocks.4	400			
9	Stage 8 + blocks.5	400			
10	Stage 9 + layer.2	400			
11	Stage 10 + blocks.6	400			
12	Stage 11 + blocks.7	400			
13	Stage 12 + layer.3	400			
14	Stage 13 + blocks.8	400			
15	Stage 14 + blocks.9	400			
16	Stage 15 + layer.4	400			
17	Stage 16 + blocks.10	400			
18	Stage 17 + blocks.11	400			
19	Stage 18 + layer.5	400			
20	All layers	1000			

Table 4: Hyperparameters used for the architecture-aware connection building algorithm. For each model, the whole process is divided into multiple stages, where only the parameter subsets specified in the layer moving sequence are optimized in each stage. The layer names are abbreviated in the table, following the PyTorch implementation of the corresponding architecture. We also report the total runtime for constructing the complete low-loss path.

A.5 Additional Experimental Results of DDPM and NanoCLIP with Different Mode Pairs

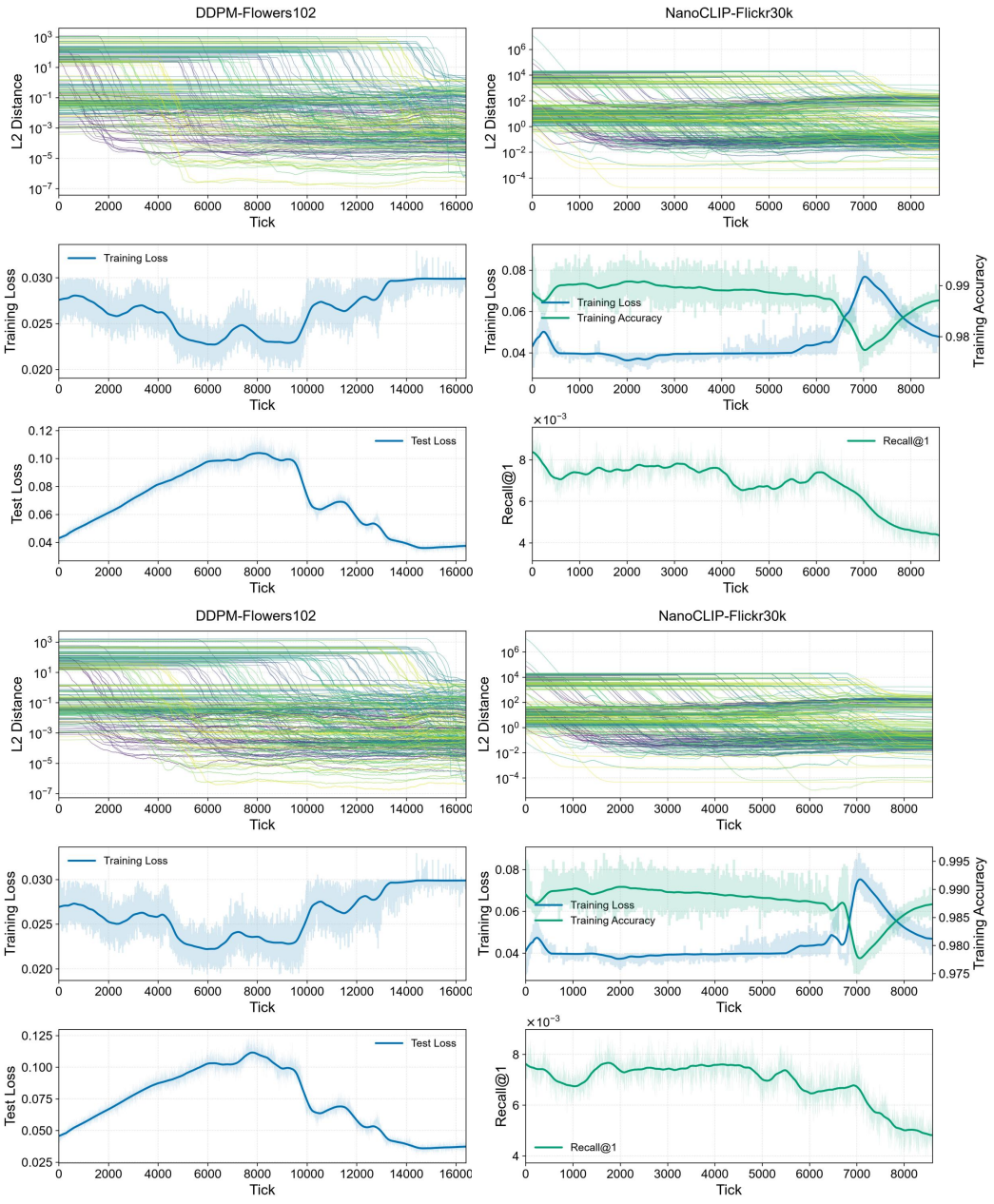


Figure 9: Additional results of the architecture-aware connection building algorithm on DDPM (first column) and NanoCLIP (second column). The first three rows correspond to the second mode pair, and the last three rows correspond to the third mode pair. Both endpoint modes are replaced relative to each other.

A.6 Experimental Results of DDPM Generated Images and FID Analysis

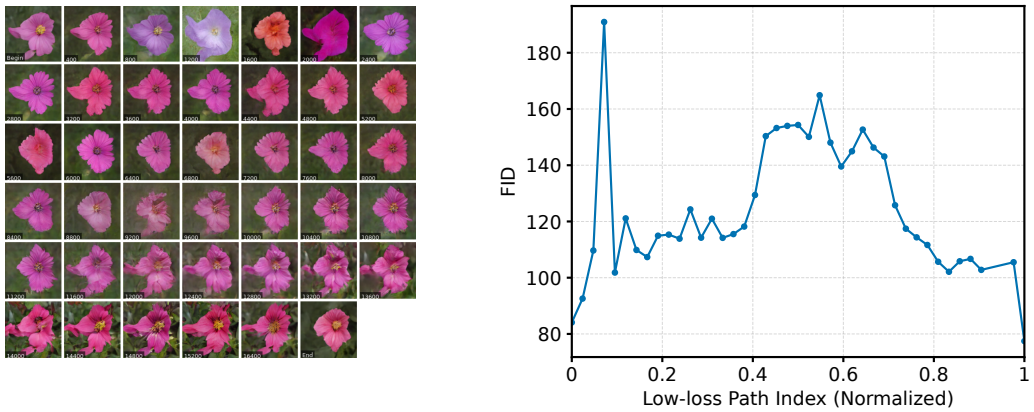


Figure 10: Generation quality evaluation of checkpoints along the constructed low-loss path. *Left*: representative synthesized images sampled from identical initial Gaussian noise. *Right*: the corresponding FID computed across the 41 sampled checkpoints.

A.7 Experimental Results of Endpoint Parameter Proximity and Linear Interpolation

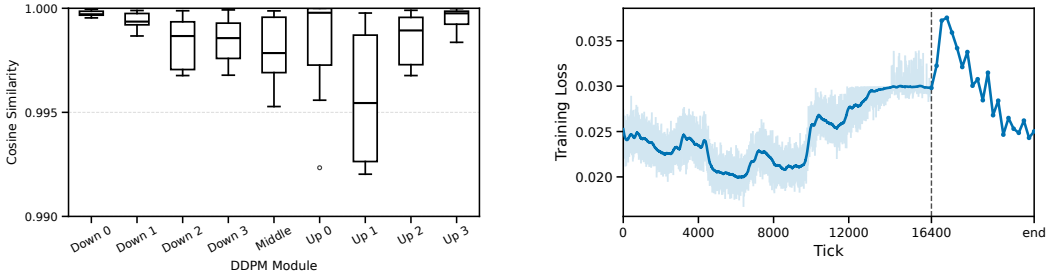


Figure 11: Parameter-space proximity and training loss near the DDPM end mode. *Left*: layer-wise cosine similarity between the DDPM checkpoint at tick 16400 and the end mode. Each box summarizes the distribution of cosine similarity values for multiple layers within one DDPM module, indicating they are highly close in parameter space. *Right*: training loss near the end mode. The left part shows the training loss along the original DDPM low-loss path. The right part shows the training loss obtained by linearly interpolating between the checkpoint at tick 16400 and the end mode using uniformly sampled interpolation points.