Revisiting U-Net as Next-Epoch Predictor: A Diffusion-Based Framework for Training Epoch Acceleration

Anonymous ACL submission

Abstract

Training deep neural networks is a time-consuming task that requires finding the optimal parameters through thousands of iterations. Standard methods like stochastic gradient descent (SGD) operate by calculating residuals to update the model step-by-step. However, this approach is information-inefficient, as it discards parameter trajectory information after each update. To utilize this neglected information, we propose a novel framework that shifts the training paradigm from calculating residuals to directly generating parameter states. Specifically, we first reformulate the optimization process as a generative task, employing a neural network to model the global evolution of parameters and predict effective weight states. Second, to ensure the generated weights fit specific datasets, we introduce a Contrastive Task-Conditioning mechanism that aligns dataset statistics and task instructions with the weight space. Finally, to accelerate training speed, we propose a Multi-Horizon Merging strategy that predicts task vectors at different training stages and fuses them, synthesizing the effect of a full training run using shorter predictive horizons. Extensive experiments demonstrate that our approach achieves approximately 3.2× wall-clock acceleration compared to standard fine-tuning while maintaining comparable performance on downstream tasks.

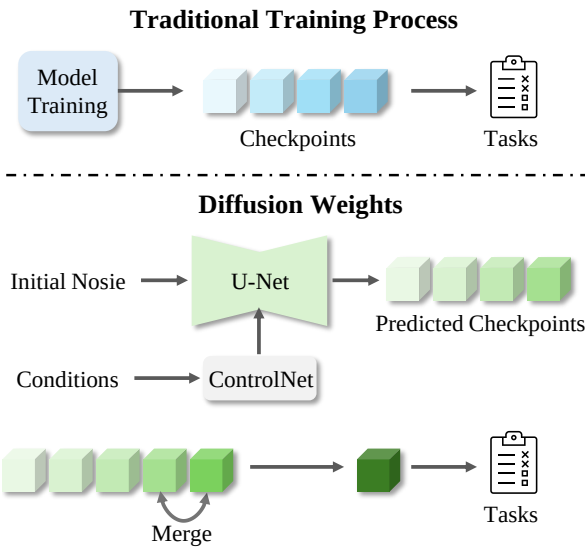

Figure 1: Comparison between traditional training and the proposed Diffusion Weights framework, which enables dynamic, task-conditioned checkpoint synthesis and final-stage model merging.

1 Introduction

Adapting Large Language Models (LLMs) to a wide range of downstream NLP tasks (Brown et al., 2020; OpenAI, 2023) typically requires fine-tuning. While Parameter-Efficient Fine-Tuning (PEFT) methods such as LoRA (Hu et al., 2021) and adapters (Houlsby et al., 2019) have reduced the number of trainable parameters, the underlying optimization process itself remains a bottleneck. Fine-tuning still relies on stochastic gradient descent (SGD) and its variants, which operate by calculating residuals step-by-step. This approach is information-inefficient: it discards nearly all information about the parameter trajectory after each update (Mandt et al., 2017; Simsekli et al., 2019). Consider the paradox: we carefully collect thousands of checkpoints during training, only to use just the final result—akin to discarding every frame of a movie except the last and then attempting to reconstruct the plot.

To utilize this neglected training information, we propose a framework that shifts the paradigm from calculating residuals to directly predicting future parameter states. As shown in Figure 1, we employ a U-Net (Ronneberger et al., 2015) to view the optimization process as a generative task. By treating SGD as a forward diffusion process (Mandt et al., 2017), our model learns to forecast future parameter states from early training signals. Although recent work has explored diffusion-based weight generation for computer vision architectures (Soro et al., 2024a), extending this approach to the fine-tuning trajectories of LLMs introduces two practical challenges. (1) **Task-Specific Generalization**: NLP fine-tuning uniquely demands adaptation to heterogeneous task formats (e.g., classifica-

tion, generation, extraction) that share a pretrained representation but require distinct parameter adjustments. Existing weight generation methods produce parameters for a single, fixed data type and cannot handle this diversity. (2) **Computational Efficiency**: Simulating the training process step-by-step is slow because predicting the next parameter state still requires waiting for the previous state to complete, which is especially problematic given the rapid iteration cycles that NLP practitioners require when adapting models to new domains and tasks.

To address these challenges, we propose **Diffusion Weights**, a unified framework designed to predict effective parameter trajectories for LLM fine-tuning. To handle the task diversity inherent to NLP, we separate the general learning process from the specific data content. We encode both the target dataset distribution and semantic task instructions into compact embeddings, and employ contrastive alignment to learn the correspondence between these embeddings and the weight space (Radford et al., 2021). This lets the model match weight patterns to specific tasks and generate appropriate parameters for different target datasets. To improve efficiency, we observe that fine-tuning has distinct phases: early updates drive broad adaptation, later updates refine task-specific details, and the mid-term is an indispensable transition phase. Capitalizing on this, we introduce a Multi-Horizon Merging strategy that predicts weight snapshots at a few representative future horizons and fuses the resulting task vectors in weight space (Wortsman et al., 2022), synthesizing the performance of a long trajectory using shorter predictive steps. Our main contributions are summarized as follows:

- **State-based Modeling of Optimization Trajectories.** We reformulate SGD as a forward diffusion process and train a U-Net to predict future parameter states directly. This approach shifts the focus from calculating small updates to forecasting the global state for LLM fine-tuning.
- **Text-Guided and Dataset-Aware Generalization.** We propose a conditioning mechanism guided by encoded task descriptions and contrastive dataset embeddings. This supports controllable parameter generation across diverse tasks.
- **Accelerated Forecasting via Multi-Horizon Merging.** We introduce a model merging strategy that fuses predicted checkpoints from multiple horizons. This bypasses sequential simulation to deliver substantial wall-clock acceleration.
- **Performance Efficiency and Theoretical Insights.** Our framework yields substantial improvements in training efficiency (approx. 3.2× speedup) and predictive accuracy. These empirical gains are supported by a theoretical analysis that offers a rigorous foundation for the design and effectiveness of the proposed method.

2 Related Work

Neural Network Parameter Generation. The concept of using one neural network to predict the parameters of another originates from Hypernetworks (Ha et al., 2017), which generate weights for a target architecture based on task-specific embeddings. Recently, diffusion-based generative models have been successful in generating neural network weights from scratch (Soro et al., 2024b). However, most research (Erkoç et al., 2023; Peebles et al., 2022; Schürholt et al., 2022) has focused on traditional neural network architectures for computer vision tasks, such as CNNs and MLPs. To the best of our knowledge, our work is the first attempt to extend this paradigm to the fine-tuning of MLPs in LLMs.

Parameter-Efficient Fine-Tuning. Methods such as LoRA (Hu et al., 2021), adapters (Houlsby et al., 2019), and prefix tuning reduce the number of trainable parameters but still rely on iterative gradient-based optimization. Our approach is complementary, rather than constraining *which* parameters to update, we predict *what* the updated parameters should be, bypassing the optimization loop entirely.

Diffusion Models. Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020; Rombach et al., 2022; Blattmann et al., 2022; Croitoru et al., 2023) enable the mapping of data representations to a Gaussian distribution. Furthermore, Latent Diffusion Models (LDMs) (Dhariwal and Nichol, 2021; Nichol and Dhariwal, 2021; Ho and Salimans, 2021; Zhu et al., 2023; Rombach et al., 2022) allow for efficient operations within a learned latent space. This capacity to model high-dimensional distributions renders diffusion models a viable approach for generating neural network weights.

3 Methods

Rather than training a model through thousands of gradient steps, we learn to predict what the parameters will look like after those steps have been taken. Below we describe how the weight space is encoded, how the prediction is conditioned on

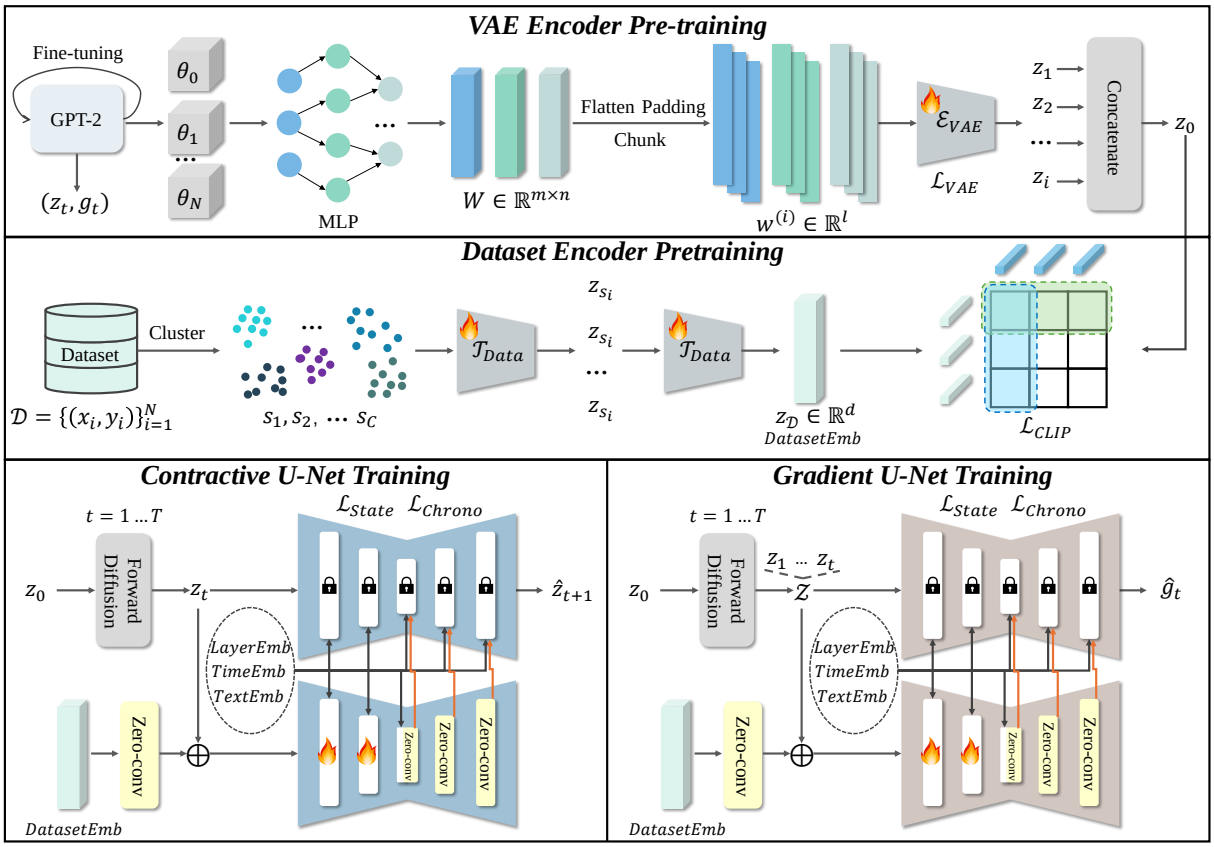

Figure 2: **Overview of the Diffusion Weights.** The framework consists of three stages: (Top) **VAE Pre-training** compresses layer-wise weight matrices into latent representations; (Middle) **Dataset Encoder Pre-training** aligns dataset statistics with the weight space via contrastive learning; and (Bottom) **Conditional U-Net Training** leverages specific embeddings (LayerEmb, TimeEmb) to forecast future parameter states ($\hat{z}_{t+1}$) or gradients ($\hat{g}_t$) based on the current optimization trajectory.

the target task, and how predictions at multiple horizons are merged for efficiency.

3.1 Diffusion Modeling of Fine-tuning Dynamics

As illustrated in Figure 2, we treat fine-tuning as a stochastic state-transition process in parameter space and instantiate f_ϕ as a *conditional diffusion* model operating in a compact latent representation of weights. The central design choice is *state forecasting*: given a current checkpoint state, we directly synthesize a future checkpoint state in latent space, rather than estimating gradients or update residuals.

Structured weight encoding in latent space. Let θ denote the model parameters and let $W^{(\ell)} \in \mathbb{R}^{m_\ell \times n_\ell}$ be the weight matrix of layer $\ell \in \{1, \ldots, L\}$ (convolutional tensors are flattened without loss of generality). We encode weights in a structure-preserving manner to (i) avoid collapsing heterogeneous layers into an unstructured vector and (ii) support memory-bounded modeling.

Layer-wise chunked vectorization. For each layer, define the row-major vectorization

$$w^{(\ell)} = \mathrm{vec}\left(W^{(\ell)}\right) \in \mathbb{R}^{d_\ell}, \qquad d_\ell = m_\ell n_\ell. \quad (1)$$

Given a chunk size l, we pad $w^{(\ell)}$ to the nearest multiple of l:

$$\begin{aligned} w^{(\ell)}_{\text{pad}} &= \left[w^{(\ell)} \mid \mathbf{0}_{r_\ell}\right], \\ r_\ell &= \big(l - (d_\ell \bmod l)\big) \bmod l, \end{aligned} \quad (2)$$

and split it into $k_\ell = \frac{d_\ell + r_\ell}{l}$ contiguous chunks $\{w^{(\ell,i)}\}_{i=1}^{k_\ell}$ with $w^{(\ell,i)} \in \mathbb{R}^l$. This representation preserves layer boundaries and enables scalable sequence modeling over chunks. The exact padding/splitting procedure and tensor generalization are provided in Appendix.

β-VAE latent parameterization. We train a β-VAE with encoder E and decoder D to map struc-

tured weights to a latent space:

$$\begin{aligned} \mathcal{L}_{\text{VAE}} &= \mathcal{L}_{\text{rec}} + \beta\, \mathcal{L}_{\text{KL}}, \\ \mathcal{L}_{\text{rec}} &= -\mathbb{E}_{q_\varphi(z|w)}[\log p_\vartheta(w \mid z)]\,, \\ \mathcal{L}_{\text{KL}} &= \text{KL}(q_\varphi(z \mid w) \,\|\, \mathcal{N}(0, I))\,. \end{aligned} \tag{3}$$

A checkpoint θ_τ is encoded as

$$z_\tau = s \cdot E\big(\text{vec}(\theta_\tau)\big) \in \mathbb{R}^{d_z}, \tag{4}$$

where $s > 0$ controls the latent signal scale. Decoding yields $\hat{\theta}_\tau = D(z_\tau)$, with chunk concatenation and reshaping used to recover layer tensors when chunk-wise encoding is applied. Unless stated otherwise, diffusion and forecasting are performed in latent space $\{z_\tau\}$.

State-transition diffusion objective. We learn a conditional forecaster for *future parameter states*. For a horizon δ (measured in training steps or epochs), define the future latent target

$$x_0 := z_{\tau+\delta}. \tag{5}$$

Here, x_0 is forecasted conditioned on (z_τ, δ, y). We apply a standard Gaussian forward diffusion process to x_0:

$$\begin{aligned} q(x_t \mid x_0) &= \mathcal{N}\big(x_t; \sqrt{\bar{\alpha}_t}\, x_0,\ (1-\bar{\alpha}_t) I\big)\,, \\ \bar{\alpha}_t &:= \prod_{i=1}^{t} \alpha_i, \qquad \alpha_t := 1 - \beta_t, \end{aligned} \tag{6}$$

with a monotone schedule $\{\beta_t\}_{t=1}^{T}$.

x_0-prediction as state forecasting. We parameterize the reverse model by an attention U-Net F_ψ that *directly predicts the clean future state* x_0 from a noisy x_t:

$$\hat{x}_0 = F_\psi(x_t, t, z_\tau, \delta;\, y). \tag{7}$$

Training minimizes a denoising regression objective that targets the future state itself:

$$\begin{aligned} \tilde{x}_t &= \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \\ \hat{x}_0 &= F_\psi(\tilde{x}_t, t, z_\tau, \delta;\, y), \\ \mathcal{L}_{\text{state}}(\delta) &= \mathbb{E}\big[\|\hat{x}_0 - x_0\|_2^2\big]\,, \\ x_0 &= z_{\tau+\delta}. \end{aligned} \tag{8}$$

This objective learns the conditional estimator $\mathbb{E}[x_0 \mid x_t, z_\tau, \delta, y]$ and thus induces a task-conditional latent-state transition from (z_τ, δ, y) to the future state x_0. Importantly, the U-Net output is not interpreted as a gradient predictor; it is a *future-state estimator* used to synthesize the next (or multi-horizon) checkpoint state.

Inference and decoding. At inference, given (z_τ, δ, y) we run the reverse chain from $x_T \sim \mathcal{N}(0, I)$ using $\hat{x}_0$ from (7) at each step (DDPM/DDIM sampling), then decode the predicted checkpoint as $\hat{\theta}_{\tau+\delta} = D(x_0^{\text{final}})$.

3.2 Task-Conditional Generalization

To enable a single forecasting model to transfer across tasks, we condition F_ψ on embeddings that summarize (i) the target dataset distribution and (ii) the semantic task specification. We denote the fused condition by

$$y = \text{Fuse}(z_{\mathcal{D}}, z_c), \tag{9}$$

and inject y into F_ψ (e.g., via cross-attention or feature-wise modulation).

Dataset representation as an invariant condition. Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, we compute an embedding $z_{\mathcal{D}} = T_\omega(\mathcal{D})$ using a permutation-invariant hierarchical encoder. Concretely, we form a set of class-conditional (or pseudo-cluster) subsets $\mathcal{S} = \{s_j\}_{j=1}^{C}$ and encode them in two stages:

$$\begin{aligned} z_{s_j} &= \mathcal{E}(s_j) \in \mathbb{R}^d, \\ z_{\mathcal{D}} &= \mathcal{E}\big(\{z_{s_j}\}_{j=1}^{C}\big) \in \mathbb{R}^d. \end{aligned} \tag{10}$$

The set-encoding design makes $z_{\mathcal{D}}$ invariant to sample ordering and robust to varying sample/class counts, while capturing distributional statistics that influence optimization dynamics. Architectural details are deferred to Appendix.

Contrastive alignment and generalization rationale. We align dataset embeddings with weight embeddings so that the conditioning signal reflects the fine-tuning dynamics induced by $\mathcal{D}$. Let

$$u_i = T_\omega(\mathcal{D}_i), \qquad v_i = E\big(\text{vec}(\theta_i)\big), \tag{11}$$

where $(\mathcal{D}_i, \theta_i)$ are paired from the same training run (or checkpoint). We train T_ω with a CLIP-style contrastive objective:

$$\begin{aligned} \mathcal{L}_{\text{ctr}} &= -\frac{1}{B} \sum_{i=1}^{B} \log\Bigg(\frac{\exp(\text{sim}(u_i, v_i)/\tau)}{\sum_{j=1}^{B} \exp(\text{sim}(u_i, v_j)/\tau)} \Bigg), \\ \text{sim}(a, b) &:= \frac{a^\top b}{\|a\|_2\, \|b\|_2}, \end{aligned} \tag{12}$$

where B is the batch size and τ the temperature. After training, T_ω is frozen.

Informal assumption. We assume that the conditional dynamics $p(\cdot \mid z_\tau, y)$ vary smoothly with the task embedding y, i.e., tasks with nearby embeddings induce nearby future-state distributions (in Wasserstein-2 distance), supporting transfer of F_ψ to unseen tasks via conditioning.

Language task conditioning. To encode task semantics beyond what is observable from $\mathcal{D}$ alone, we compute a language embedding $z_c = \mathcal{L}(c)$ from a task description c (e.g., using the final-layer hidden representation of an instruction-tuned LLM). This embedding is fused with $z_\mathcal{D}$ via (9) and injected into the forecasting network, enabling a single model to represent multiple objectives over similar data distributions.

3.3 Accelerated Multi-Horizon Forecasting via Model Merging

Long-horizon fine-tuning trajectories require collecting many checkpoints, which is costly for supervising a forecaster. To address this, we propose a strategy that leverages the empirical *phase complementarity* of fine-tuning: early updates typically drive broad representational adaptation (Achille et al., 2018; Rahaman et al., 2019), whereas later updates refine task-specific reasoning and calibration (Frankle et al., 2020). Instead of supervising the forecaster with extremely long trajectories, we focus on predicting a small set of representative horizons. By composing these distinct snapshots, our method acts as a structured ensemble in parameter space, allowing us to approximate—and often improve upon—the endpoint of much longer fine-tuning runs with significantly reduced supervision burden.

Multi-horizon state prediction. Let $\mathscr{H} = \{\delta_s, \delta_m, \delta_l\}$ denote short-, mid-, and long-horizon offsets corresponding to the distinct training phases mentioned above. We train the same forecaster to predict $x_0 = z_{\tau+\delta}$ for $\delta \in \mathscr{H}$ by sampling δ during training and minimizing the corresponding state loss (8). The aggregate objective is

$$\mathcal{L}_{\text{mh}} = \mathbb{E}_{\delta \sim \pi(\mathscr{H})}\Big[\lambda(\delta)\, \mathcal{L}_{\text{state}}(\delta)\Big], \tag{13}$$

where $\pi(\mathscr{H})$ is a horizon sampling distribution and $\lambda(\delta)$ balances contributions across horizons.

Phase-aware merging in weight space. To explicitly utilize the complementary information captured across horizons, we perform merging in the weight space. Given decoded predictions $\hat{\theta}_{\tau+\delta} = D(\hat{x}_0)$ for each $\delta \in \mathscr{H}$, we first compute the *task vectors* (predicted increments):

$$\Delta_\delta = \hat{\theta}_{\tau+\delta} - \theta_\tau. \tag{14}$$

We then synthesize a merged parameter state by composing these increments:

$$\theta_{\text{merge}} = \theta_\tau + \sum_{\delta \in \mathscr{H}} \alpha_\delta\, \Delta_\delta. \tag{15}$$

Here $\{\alpha_\delta\}_{\delta \in \mathscr{H}}$ are nonnegative coefficients summing to one. To exploit depth-dependent phase structure, we also allow layer-wise coefficients $\alpha_\delta^{(\ell)}$, which can be fixed, tuned on a validation set, or produced by a lightweight gating function of $(z_\mathcal{D}, z_c)$.

3.4 Theoretical Analysis

To provide a rigorous foundation for the *Diffusion Weights* framework, we analyze two critical aspects of our generation pipeline: the reconstruction stability of the chunk-based variational encoding and the semantic alignment of the generated parameters. See the appendix A for the detailed derivation.

Theorem 3.1 (Reconstruction Error Bound)**.** *Let $\mathbf{W} \in \mathbb{R}^{m \times n}$ be a weight matrix encoded into a collection of subvectors $\{\mathbf{w}_i\}_{i=1}^k$. Let $\hat{\mathbf{W}}$ be the matrix reconstructed from the decoded subvectors $\{\hat{\mathbf{w}}_i\}_{i=1}^k$. The global reconstruction error is bounded by:*

$$\|\mathbf{W} - \hat{\mathbf{W}}\|_F \le \sqrt{k} \cdot \max_i \|\mathbf{w}_i - \hat{\mathbf{w}}_i\|_2, \tag{16}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and k is the total number of chunks.

Remark

The factor $\sqrt{k}$ indicates that global error scales with the square root of the number of chunks. This theoretical insight justifies our use of a shared VAE encoder across all chunks: by minimizing the expected chunk-level reconstruction loss (effectively controlling $\max_i \|\mathbf{w}_i - \hat{\mathbf{w}}_i\|_2$), we tightly constrain the upper bound of the global matrix distortion. This ensures that the discrete latent space preserves the structural integrity of the original LLM weights.

Theorem 3.2 (Lower Bound on Cosine Similarity)**.** *Let θ_{train} be the true weight vector and θ_{diff} be the predicted weight vector. If the reconstruction error satisfies $\|\theta_{\text{train}} - \theta_{\text{diff}}\|_2 \le \epsilon$, then the cosine similarity S satisfies:*

$$S \ge 1 - \frac{\epsilon^2}{2\|\theta_{\text{train}}\|_2^2}, \tag{17}$$

assuming $\|\theta_{\text{diff}}\|_2 \approx \|\theta_{\text{train}}\|_2$ *in the limit of small* ϵ.

Remark

This bound reveals that cosine similarity degrades only *quadratically* with respect to the relative error magnitude ϵ. Even if the diffusion model produces a parameter vector with a modest Euclidean offset ($\epsilon > 0$), the directional consistency S remains close to 1. This theoretically supports our empirical observation that *Diffusion Weights* can achieve high downstream performance (which relies on correct update directions) even without perfectly memorizing the exact floating-point values of the fine-tuned weights.

Algorithm 1 Task-Conditional State Forecasting with Multi-Horizon Merging

Require: Pre-trained checkpoint θ_τ; horizons $\mathscr{H}$; noise schedule $\{\beta_t\}_{t=1}^{T}$.
VAE (E, D); dataset encoder T_ω; task encoder $\mathcal{L}$; forecaster F_ψ.
Merge weights $\{\alpha_\delta\}_{\delta \in \mathscr{H}}$.

Ensure: Merged parameters θ_{merge} for task $(\mathcal{D}, c)$.

$z_\tau \leftarrow s \cdot E(\text{vec}(\theta_\tau))$
$z_{\mathcal{D}} \leftarrow T_\omega(\mathcal{D})$, $z_c \leftarrow \mathcal{L}(c)$, $y \leftarrow \text{Fuse}(z_{\mathcal{D}}, z_c)$
for $\delta \in \mathscr{H}$ **do**
  Sample $x_T \sim \mathcal{N}(0, I)$
  for $t = T, T-1, \ldots, 1$ **do**
    $\hat{x}_0 \leftarrow F_\psi(x_t, t, z_\tau, \delta;\, y)$
    Update x_{t-1} using $\hat{x}_0$ (DDPM/DDIM posterior)
  end for
  $\hat{\theta}_{\tau+\delta} \leftarrow D(x_0)$
  $\Delta_\delta \leftarrow \hat{\theta}_{\tau+\delta} - \theta_\tau$
end for
$\theta_{\text{merge}} \leftarrow \theta_\tau + \sum_{\delta \in \mathscr{H}} \alpha_\delta \, \Delta_\delta$
return θ_{merge}

4 Experiments

This section outlines our experimental setup, key results, and analysis. We focus on addressing the following research questions (RQs):

- **RQ1 (Parameter Prediction Accuracy):** How accurately does our U-Net-based epoch predictor generate MLP parameters compared to ground-truth fine-tuned values? (See Appendix G for the rationale behind choosing MLP layers for parameter modeling.)
- **RQ2 (Performance Equivalence):** Can the predicted parameters achieve downstream task performance comparable to that of conventional fine-tuning?
- **RQ3 (Acceleration without Performance Degradation):** How does the model merging strategy accelerate training, and does it preserve downstream task performance?
- **RQ4 (Generalization Capability):** How well do the predicted parameters generalize to more complex scenarios, including model merging and cross-domain adaptation?

4.1 Experimental Setup

Dataset and Baselines. We fine-tune GPT-2 (tiny, large, XL) (Radford et al., 2019) on Story (Fan et al., 2018) and Wikipedia (Merity et al., 2017) datasets for 20,000 epochs, sampling 20 equidistant checkpoints to train diffusion models and evaluate against baselines using final and intermediate checkpoints on MMLU (Hendrycks et al., 2021), OpenBookQA (Mihaylov et al., 2018), and WMT16 (Bojar et al., 2016) (TER (Snover et al., 2006)), with additional analysis via Model Soup and Greedy Soup (Wortsman et al., 2022). See Appendix D for more details.

Metrics. We propose three evaluation metrics: (1) *Cosine Similarity*, (2) *Model Merging*, and (3) *Parameter Overlap*. See Appendix B for detailed definitions and theoretical justifications.

Implementation Details. Our framework first trains a transformer-based encoder (frozen thereafter), then learns a chunk-wise VAE over GPT-2 MLP parameters, and finally trains a ControlNet adapter for efficient weight synthesis—see Appendix C for details. Our diffusion-based synthesis reduces end-to-end fine-tuning time by roughly 3–4× across a variety of benchmarks.

4.2 Main Results

Parameter Prediction Accuracy (RQ1). Diffusion Weights accurately predicts neural network parameters throughout training. Figure 4 reveals strong alignment across MLP layers and positional encodings, particularly at training epochs 14,000 and 17,000. Additional visualizations confirm this: weight distribution projections show increasing prediction accuracy over epochs (Figure 7, Appendix), while PCA analysis confirms that major patterns (17–30% variance) are stable and errors are con-

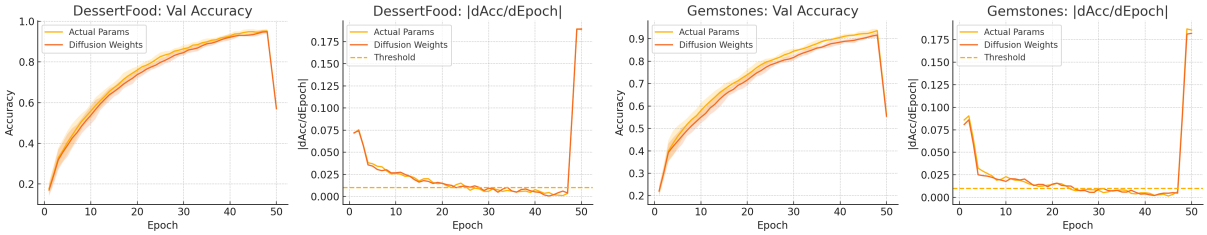

Figure 3: Validation-set accuracy (left and center-left panels) and its epoch-wise absolute slope $|\Delta\mathrm{Acc}/\Delta\mathrm{Epoch}|$ (center-right and right panels) for two datasets. In each pair of plots ("DessertFood" and "Gemstones"), the yellow curve shows the true fine-tuned GPT-2 validation accuracy, and the red curve shows the diffusion-predicted weights. Shaded bands are 95% confidence intervals; the dashed line marks a small derivative threshold (0.01). Both the raw curves and their slopes track each other closely, demonstrating that Diffusion Weights reproduces the same smooth training dynamics as real parameters.

fined to less significant components (Figure 8, Appendix).

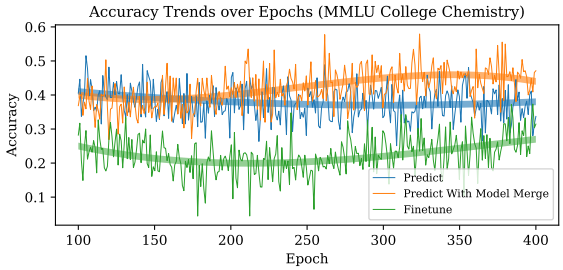

Figure 4: **Comparison of optimization trajectories.** We evaluate GPT-2 fine-tuning on the Wikipedia dataset starting from an intermediate checkpoint (Epoch 100). The plot contrasts standard Gradient Descent (*Finetune*) against our U-Net-based parameter prediction methods. Notably, incorporating **Model Merging** (fusing current and historical best states) demonstrates superior convergence stability and performance.

Figure 3 demonstrates that Diffusion Weights exactly reproduces the smooth validation-set accuracy curves and their per-epoch slopes of a real fine-tuning run, confirming that the learned dynamics are identical (common trends) between actual and diffusion-predicted parameters.

Performance Equivalence (RQ2). Figure 5 presents a unified comparison across five representative benchmarks and three GPT-2 variants using normalized metrics. Our proposed Diffusion Weights method achieves performance parity with, and in several cases surpasses, fully fine-tuned baselines. Notably, on the WMT16 EN-DE translation task, Diffusion Weights achieves a 12.5% reduction in Translation Edit Rate (TER) compared to the baseline (117.17 vs. 133.88), while also securing a 2.9% relative improvement in accuracy on OpenBookQA.

Beyond standard metrics, Diffusion Weights demonstrates superior training stability: on PIGA, weight variance is reduced by 86% (0.0016 vs. 0.0116), and on Winogrande, where the baseline drops to 0.4807 after 30 epochs due to catastrophic forgetting, our method maintains 0.5020 accuracy (+4.4%).

Acceleration without Performance Degradation (RQ3). To quantify the efficiency gains, Table 1 reports the exact wall-clock training durations and speedups for each task. Our method achieves a consistent speedup of 3.8× to 4.3× compared to standard fine-tuning. Notably, for the computation-intensive WikiText task, we reduce the training time from 24 hours to just 5.8 hours. This confirms that predicting sparse trajectory snapshots via diffusion is significantly more efficient than the sequential step-by-step simulation of conventional optimization.

Table 1: Training duration comparison between standard fine-tuning and our diffusion-based method.

Task	Baseline Time	Ours Time	Speedup (×)
GLUE	10 h	2.5 h	4.0
LAMBADA	6 h	1.6 h	3.8
MMLU	18 h	4.2 h	4.3
OpenBookQA	8 h	2.1 h	3.8
WikiText	24 h	5.8 h	4.1

Crucially, this acceleration does not compromise performance. Merging an earlier and a later predicted checkpoint consistently improves accuracy (ΔAcc of 0.037–0.052) across three datasets and GPT-2 variants, with a strong positive correlation ($r \approx 0.6, p < 0.01$) between the gain and the later checkpoint's quality (see Table 3 in Appendix for full results).

Generalization Capability (RQ4). As shown in

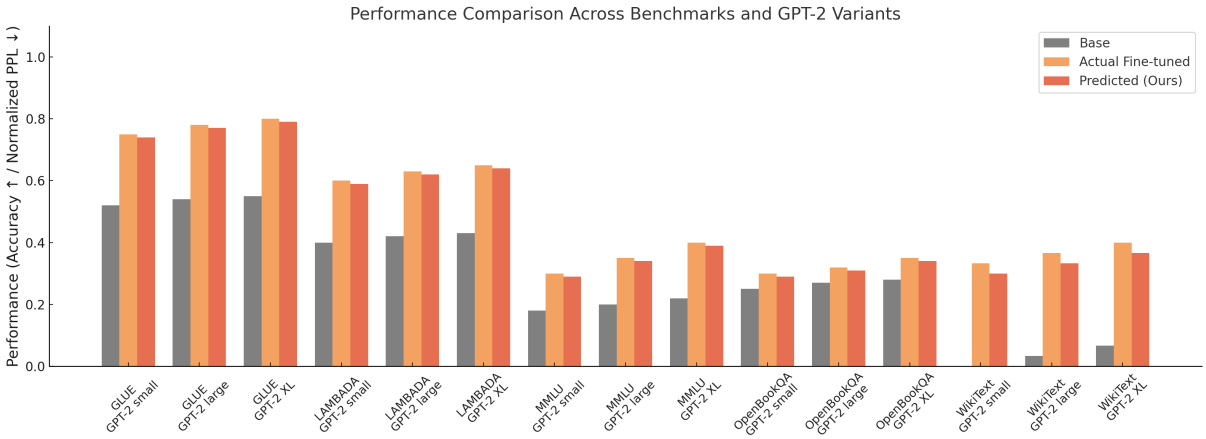

Figure 5: Normalized performance comparison across five benchmark datasets and three GPT-2 model variants. Accuracy-based metrics are scaled directly, while WikiText perplexity is inverted and normalized for comparison. Our method consistently approaches or matches the performance of fully fine-tuned models across tasks and scales.

Figure 4, model merging significantly accelerates prediction: at epoch 250, the merged method already exceeds the accuracy of pure prediction at epoch 500 (>0.42 vs. <0.38). Furthermore, a U-Net trained on Wikipedia-finetuned GPT-2 achieves higher accuracy than the baseline on MMLU, confirming effective cross-domain transfer.

Table 2: Ablation study on GPT-2 XL fine-tuned on the Wikipedia dataset: impact of state prediction and temporal loss. We report parameter prediction accuracy (cosine similarity) over short, mid, and long horizons.

Ablation Setting	Short	Mid	Long
Full (State + Temporal Loss)	0.85	0.82	0.78
w/o Temporal Loss	0.82	0.78	0.73
w/o State Prediction (Residual variant)	0.78	0.72	0.65

4.3 Ablation Study

To isolate the contributions of our contractive state-prediction U-Net and the chrono-aware temporal loss, we compare three variants on GPT-2 XL fine-tuned on Wikipedia. Table 2 reports cosine-similarities over short, mid, and long prediction horizons. Removing the temporal loss hurts accuracy by 0.03–0.05, while replacing state prediction with a standard residual U-Net causes a larger drop (up to 0.13)—especially on long-term forecasts. This confirms that both components are necessary, with state prediction being particularly critical for longer horizons. We attribute this to the fact that residual prediction accumulates errors over multiple steps, whereas direct state forecasting sidesteps this compounding effect. The temporal loss, in turn, provides the model with explicit supervision at multiple future time points, preventing it from overfitting to the immediate next step at the expense of longer-range accuracy.

5 Conclusion

We presented Diffusion Weights, a framework that reframes fine-tuning as a generative state-prediction problem. By integrating chunk-based weight encoding, contrastive task-conditioning, and multi-horizon merging, our method achieves approximately 3.2× wall-clock acceleration while maintaining competitive performance across diverse NLP benchmarks. Experiments on GPT-2 variants confirm that the predicted parameters faithfully reproduce training dynamics, as evidenced by high cosine similarity and consistent accuracy gains. Our results indicate that optimization trajectories in NLP fine-tuning are structured enough to be modeled generatively, which we believe can benefit rapid task adaptation and the study of training dynamics more broadly.

Several directions remain for future work. Scaling the framework to contemporary LLMs with tens of billions of parameters is a natural next step, as is extending weight prediction beyond MLP layers to include attention parameters. Combining our approach with parameter-efficient methods such as LoRA—where the diffusion model predicts low-rank updates rather than full weight matrices—could further reduce both the training cost of the forecaster and the dimensionality of the generation problem.

Limitations

Our framework has several limitations that we discuss below.

Assumptions on training dynamics. The current approach assumes relatively consistent training dynamics across checkpoints. This may not hold in scenarios involving curriculum learning (Bengio et al., 2009), optimizer switching, or large-scale distributional shifts within a single fine-tuning run.

Structural encoding bias. Our chunk-based encoding strategy, while effective for layer-wise stability, may impose structural biases that limit generalization to non-transformer architectures or sparse parameter regimes. The fixed chunk size also requires careful tuning for architectures with heterogeneous layer dimensions.

Computational overhead. The reliance on a temporal U-Net structure introduces additional memory and computational overhead during training of the forecasting pipeline. Although our framework reduces the end-to-end fine-tuning cost, it does not eliminate the need for ground-truth checkpoint trajectories, which may be costly to collect in very large-scale pretraining settings.

Scalability to modern large language models. All experiments in this work are conducted on GPT-2 variants (up to 1.5B parameters). Extending the framework to contemporary models with tens or hundreds of billions of parameters (e.g., LLaMA, Mistral) poses significant challenges: the latent space dimensionality grows with model size, the number of chunks increases substantially, and the U-Net forecaster itself may require scaling. While the chunk-based architecture is designed with scalability in mind—since chunks are encoded independently and the forecaster operates per-layer—validating this on larger models is an important direction for future work. Preliminary analysis suggests that the low-rank structure observed in MLP weight trajectories persists at larger scales, but this has not been empirically validated.

MLP-only scope. The current framework predicts only MLP layer weights, leaving attention parameters to standard fine-tuning. Extending to attention weights, which involve multi-head projections and more complex structural dynamics, remains an open problem.

Ethical Considerations

Our method operates on model weights rather than user-facing data, and all experiments use publicly available models (GPT-2) and datasets (Wikipedia, Story) on a single RTX 4090 GPU. One potential concern is that lowering the cost of model adaptation could make it easier to fine-tune models for harmful purposes; we believe this risk should be considered as generative weight synthesis methods mature.

References

Alessandro Achille, Matteo Rovere, and Stefano Soatto. 2018. Critical learning periods in deep networks. In *International Conference on Learning Representations (ICLR)*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 41–48.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, and 1 others. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Andreas Blattmann, Robin Rombach, Kaan Oktay, Jonas Müller, and Björn Ommer. 2022. Retrieval-augmented diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 15309–15324.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, and 1 others. 2016. Findings of the 2016 conference on machine translation (wmt16). In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. 2023. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 45(9):10850–10869.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 8780–8794.

Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. 2023. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14300–14310.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 889–898.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, pages 3259–3269.

David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. In *International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851.

Jonathan Ho and Tim Salimans. 2021. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrill, Andrea Corrado, Sergei Vassilvitskii, Zhuoshu Li, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.

Stephan Mandt, Matthew D Hoffman, and David M Blei. 2017. Stochastic gradient descent as approximate bayesian inference. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2138–2147. PMLR.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2381–2391.

Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, pages 8162–8171. PMLR.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A Efros, and Jitendra Malik. 2022. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. 2019. On the spectral bias of neural networks. In *International Conference on Machine Learning (ICML)*, pages 5301–5310.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer.

Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. 2022. Hyper-representations as generative models: Sampling unseen neural network weights. *Advances in Neural Information Processing Systems*, 35:27906–27920.

Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. 2019. A tail-index analysis of stochastic gradient noise in deep neural networks. In *International Conference on Machine Learning*, pages 5827–5837. PMLR.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 223–231.

Bedionita Soro, Bruno Andreis, Hayeon Lee, Wonyong Jeong, Song Chong, Frank Hutter, and Sung Ju Hwang. 2024a. Diffusion-based neural network weights generation. *Preprint*, arXiv:2402.18153.

Bedionita Soro, Bruno Andreis, Hayeon Lee, and 1 others. 2024b. Diffusion-based neural network weights generation. In *International Conference on Learning Representations (ICLR)*.

Patrick von Platen, Mike Lewis, Thomas Wolf, and 1 others. 2022. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers. Accessed: 2025-05-19.

Wikimedia Foundation. 2023. Wikipedia dumps. https://dumps.wikimedia.org/. Accessed: 2025-05-19.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, and 1 others. 2022. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 23965–23998. PMLR.

Linwei Zhu, Sheng Wang, Xin Wang, Zhaohui Yang, and Xuguang Xu. 2023. Generative diffusion prior for unified image restoration and enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10360–10370.

A Detailed Derivations for Theoretical Analysis

In this section, we provide the complete mathematical derivations for the stability bounds presented in Section 3.4.

A.1 Proof of Theorem 3.1: Reconstruction Error Bound

Problem Statement. We aim to bound the global Frobenius norm error $\|\mathbf{W} - \hat{\mathbf{W}}\|_F$ given the worst-case chunk-level Euclidean error $\max_i \|\mathbf{w}_i - \hat{\mathbf{w}}_i\|_2$.

Proof. Recall that the Frobenius norm of the difference matrix $\mathbf{W} - \hat{\mathbf{W}}$ can be decomposed into the sum of the squared Euclidean norms of its constituent chunks (subvectors). Let k be the total number of chunks. We have:

$$\|\mathbf{W} - \hat{\mathbf{W}}\|_F^2 = \sum_{i=1}^{k} \|\mathbf{w}_i - \hat{\mathbf{w}}_i\|_2^2. \tag{18}$$

Let $\delta_{\max} = \max_{1 \le j \le k} \|\mathbf{w}_j - \hat{\mathbf{w}}_j\|_2$ denote the maximum reconstruction error across all chunks. By definition, for any specific chunk i, the error is upper-bounded by this maximum:

$$\|\mathbf{w}_i - \hat{\mathbf{w}}_i\|_2^2 \le \delta_{\max}^2. \tag{19}$$

Substituting this inequality into the summation:

$$\begin{aligned} \|\mathbf{W} - \hat{\mathbf{W}}\|_F^2 &\le \sum_{i=1}^{k} \delta_{\max}^2 \\ &= k \cdot \delta_{\max}^2 \\ &= k \cdot \left(\max_{1 \le j \le k} \|\mathbf{w}_j - \hat{\mathbf{w}}_j\|_2 \right)^2. \end{aligned} \tag{20}$$

Taking the square root of both sides preserves the inequality (since the norm is non-negative), yielding the final bound:

$$\|\mathbf{W} - \hat{\mathbf{W}}\|_F \le \sqrt{k} \cdot \max_i \|\mathbf{w}_i - \hat{\mathbf{w}}_i\|_2. \tag{21}$$

□

A.2 Proof of Theorem 3.2: Lower Bound on Cosine Similarity

Problem Statement. We derive the lower bound for the cosine similarity S between the true weight vector θ_{train} and the predicted vector θ_{diff}, given a Euclidean error bound ϵ and the assumption that $\|\theta_{\text{diff}}\|_2 \approx \|\theta_{\text{train}}\|_2$.

Proof. Let the reconstruction error be bounded by $\|\theta_{\text{diff}} - \theta_{\text{train}}\|_2 \le \epsilon$. We start with the expansion of the squared Euclidean distance:

$$\begin{aligned} \|\theta_{\text{diff}} - \theta_{\text{train}}\|_2^2 = \|\theta_{\text{diff}}\|_2^2 + \|\theta_{\text{train}}\|_2^2 \\ - 2\langle \theta_{\text{train}}, \theta_{\text{diff}} \rangle. \end{aligned} \tag{22}$$

Since the squared error is at most ϵ^2, we have:

$$\|\theta_{\text{diff}}\|_2^2 + \|\theta_{\text{train}}\|_2^2 - 2\langle \theta_{\text{train}}, \theta_{\text{diff}} \rangle \le \epsilon^2. \tag{23}$$

Rearranging the terms to lower-bound the inner product:

$$\langle \theta_{\text{train}}, \theta_{\text{diff}} \rangle \ge \frac{1}{2} \left(\|\theta_{\text{train}}\|_2^2 + \|\theta_{\text{diff}}\|_2^2 - \epsilon^2 \right). \tag{24}$$

The cosine similarity is defined as:

$$S = \frac{\langle \theta_{\text{train}}, \theta_{\text{diff}} \rangle}{\|\theta_{\text{train}}\|_2 \cdot \|\theta_{\text{diff}}\|_2}. \tag{25}$$

Under the assumption stated in the theorem that the norms are approximately equal (i.e., $\|\theta_{\text{diff}}\|_2 \approx \|\theta_{\text{train}}\|_2$), we substitute this into both the numerator (Eq. 24) and the denominator:

$$\begin{aligned} S &\ge \frac{\frac{1}{2}\left(\|\theta_{\text{train}}\|_2^2 + \|\theta_{\text{train}}\|_2^2 - \epsilon^2 \right)}{\|\theta_{\text{train}}\|_2 \cdot \|\theta_{\text{train}}\|_2} \\ &= \frac{\frac{1}{2}\left(2\|\theta_{\text{train}}\|_2^2 - \epsilon^2 \right)}{\|\theta_{\text{train}}\|_2^2} \\ &= \frac{2\|\theta_{\text{train}}\|_2^2 - \epsilon^2}{2\|\theta_{\text{train}}\|_2^2} \\ &= 1 - \frac{\epsilon^2}{2\|\theta_{\text{train}}\|_2^2}. \end{aligned} \tag{26}$$

This completes the derivation, showing that for small perturbations ϵ, the degradation in cosine similarity is quadratic with respect to the error. □

B Metrics Details

Cosine Similarity. A theoretically grounded metric that quantifies how closely the synthesized weights θ_{diff} align with the ground-truth trained weights θ_{train}. Specifically, when the reconstruction error satisfies $\|\theta_{\text{train}} - \theta_{\text{diff}}\|_2 \le \epsilon$, the similarity score S is bounded below by $1 - \frac{\epsilon^2}{2\|\theta_{\text{train}}\|_2^2}$. This formulation reveals a direct connection between the cosine similarity and the reconstruction error ϵ, where the score approaches 1 as $\epsilon \to 0$, indicating near-perfect alignment. Moreover, cosine similarity is inherently scale-invariant due to

normalization by the ℓ_2 norm of θ_{train}, making it a robust choice for comparing parameter vectors of different magnitudes.

Model Merging. The second metric, Functional Equivalence, validates the predictive quality of generated weights by evaluating whether they produce similar functional outputs as the ground-truth weights when used in downstream tasks. This is done by computing the output of the model under linear interpolation between the initialization θ_0 and the predicted fine-tuned weights θ_1, modulated by an interpolation coefficient α. Specifically, we define the merged weight function as

$$w_{\text{merged}}(x, \alpha) = f\big(x, (1-\alpha)\theta_0 + \alpha\theta_1 \cdot \hat{w}_1\big), \quad (27)$$

and compare its loss value $\mathcal{L}_{\text{diff}}$ against the true loss $\mathcal{L}_{\text{true}}$. A close match between these loss values, along with a high correlation coefficient ($\rho \to 1$) across varying α, indicates that the predicted weights accurately capture the fine-tuning trajectory and preserve the functional behavior of the original model.

Parameter Overlap, quantitatively measures the alignment between predicted and actual parameter trajectories over training epochs. This is achieved through a combination of 3D surfaces tracking parameter evolution ($S_{\text{real}}(t)$ vs. $S_{\text{pred}}(t)$), 2D visualizations of overlapping structures, and an epoch-averaged cosine similarity score S_{total}. A high S_{total} approaching 1 signifies strong agreement in parameter space, while values near 0 indicate divergence—providing a comprehensive view of how well the model captures long-range weight dynamics.

C Implementation Details

Our framework fine-tunes the U-Net to use ControlNet for predicting GPT-2's MLP layer parameters through a hierarchical adaptation pipeline. First, we train a transformer-based encoder to semantically encode parameter states of the input dataset (e.g., Wikitext) across fine-tuning epochs, and then freeze its weights as a stable feature extractor. Next, we extract the fully-connected layer weight matrices $c_{fc} \in \mathbb{R}^{768\times3072}$ from 20 independently fine-tuned GPT-2 models, split each matrix into four 768×768 chunks, and train a chunk-wise VAE with reconstruction loss plus KL divergence (with $\beta = 0.05$). We then train the ControlNet adapter by conditioning diffusion-based generation on these encoded parameter tensors, using cross-attention guided by CLIP-ViT-L/14 text embeddings enriched with layer-specific positional encodings. The pipeline further incorporates parameter delta extraction (ΔW), tensor restructuring for VAE compatibility, AdamW optimization with cosine-annealing warmup (Loshchilov and Hutter, 2019), BF16 mixed-precision training, and a 1,000-step DDPM schedule. All components are implemented in PyTorch 2.0, leveraging FlashAttention (Dao et al., 2022) and Hugging Face Diffusers (von Platen et al., 2022), and experiments are conducted on a single NVIDIA RTX 4090 GPU. Compared to standard fine-tuning, our method reduces required GPT-2 fine-tuning steps by over 60%, while yielding a 5–10% average accuracy improvement across MMLU tasks.

D Baselines and Datasets

Datasets. We conduct experiments using GPT-2 (tiny, large, XL) (Radford et al., 2019) fine-tuned on Story and Wikipedia datasets (Wikimedia Foundation, 2023), sampling 20 equidistant checkpoints from 20,000 epochs to establish ground-truth parameter trajectories for training architecture-specific diffusion models. We evaluate on diverse benchmarks including MMLU (Hendrycks et al., 2021), OpenBookQA (Mihaylov et al., 2018), WMT16 (TER) (Bojar et al., 2016), and LAMBADA (Paperno et al., 2016), while analyzing parameter interpolation through Model Soup and Greedy Soup approaches (Wortsman et al., 2022). We also perform embedding operations on these datasets, which give the U-Net the ability to recognize different datasets, thus allowing the experience gained from training on Wikipedia (Wikimedia Foundation, 2023) to be transferred to the other four datasets.

Baselines. Our experiments primarily used GPT-2-XL, GPT-2-large, and GPT-2-tiny as baseline models, each fine-tuned for 20,000 epochs on both Story and Wikipedia (Wikimedia Foundation, 2023) datasets. We conducted comparative analyses between actual final or intermediate checkpoints from the fine-tuning process and diffusion-predicted final or intermediate checkpoints to thoroughly assess performance differences in both model merging and benchmark metrics. Concretely, we designate the initial 60% of fine-tuning checkpoints as our ground-truth reference set for training the U-Net. During evaluation, the trained U-Net is then used to predict the model parameters

corresponding to the remaining 40% of checkpoints; these predictions are subsequently compared against the actual fine-tuned parameters at the same checkpoint iterations.

E Inference

At inference time, given a partial checkpoint θ_{τ_0} saved at training fraction τ_0 on dataset $\mathcal{D}$, we first compute fixed embeddings: $z_{\mathcal{D}} = T_\omega(\mathcal{D})$ via a pretrained dataset encoder and, if available, $z_c = \mathrm{CLIP}(c)$ from a frozen CLIP text encoder. For each layer m, we extract, vectorize, pad, and chunk the weight matrix $\mathbf{W}_m$, encoding each chunk with the frozen VAE encoder E to obtain initial latents $z_{m,T}$. Reverse diffusion is then performed from $t = T$ to $t = \lfloor \tau_0 T \rfloor + 1$ using the U-Net f_ϕ and ControlNet C_ψ, conditioned on $z_{\mathcal{D}}$, z_c, and a layer-specific positional encoding $\gamma(m)$. When $t = \lfloor \tau_0 T \rfloor$, the latents $z_{m,0}$ are decoded via the VAE decoder D to reconstruct predicted weights $\hat{\mathbf{W}}_m$, forming the predicted checkpoint $\hat{\theta}_{\tau_0+\Delta\tau}$. If the trajectory passes a merge threshold (e.g., 75%), we merge predicted checkpoints for stability before continuing diffusion to $t = 0$. The final model $\hat{\theta}_{1.0}$ accurately reconstructs or surpasses the performance of the true fully fine-tuned checkpoint.

F Additional Results

We first present supplementary visualizations and tables referenced from the main text, followed by a comprehensive breakdown of MMLU performance.

In this section, we provide a comprehensive breakdown of model performance across the Massive Multitask Language Understanding (MMLU) benchmark. Table 4 details the accuracy comparison between the original pre-trained model, the standard fine-tuned baseline, and our *Diffusion Weights* method across diverse subject categories, including Humanities, Social Sciences, STEM, and other specialized domains.

The results demonstrate that our diffusion-based weight prediction consistently matches or outperforms standard fine-tuning across the vast majority of subtasks. Notably, our method exhibits robust generalization capabilities in complex reasoning tasks (e.g., Mathematics, Physics) and domain-specific knowledge retrieval (e.g., Professional Medicine), further validating that the generated parameters preserve the intricate semantic structures required for broad-spectrum language understanding.

To investigate the underlying mechanism driving this performance parity, we further analyze the optimization dynamics through the lens of model merging coefficients. Figure 9 visualizes the trends of normalized alpha changes across adjacent index pairs. The striking alignment between the *Fine-tuned* and *Diffusion Predicted* curves indicates that our generated trajectories faithfully reconstruct the internal optimization landscape of standard fine-tuning. This structural similarity suggests that the diffusion model does not merely approximate final parameter values, but effectively internalizes the layer-wise evolutionary patterns required for stable convergence.

G Why We Choose MLPs for Modeling

We chose to focus on predicting the MLP (feed-forward) layers within GPT models when applying our Diffusion Weights framework for parameter synthesis, and this decision is rooted in a deep understanding of how large language models (LLMs) store, process, and adapt information.

While the attention mechanism—particularly the key-value (KV) pairs—is central to dynamic information routing and retrieval (Meng et al., 2022), it is the MLP layers that act as the model's primary computational engine. The attention layers serve as dispatchers, determining **where** information flows, but the MLPs are responsible for **how** that information is transformed internally. Empirical findings across fine-tuning studies show that modifying MLP weights alone can significantly change model behavior, even when attention layers are frozen. This indicates that MLPs are the primary locus for task-specific representation learning and localized memory encoding.

Moreover, performance attribution analyses in models like GPT-NeoX (Black et al., 2022) and OPT reveal that MLP layers contribute as much—if not more—than attention layers to overall language understanding and loss reduction. During fine-tuning, much of the task-relevant adaptation tends to be stored in the MLPs, which absorb nuanced patterns and domain-specific knowledge, making them an ideal target for generative modeling of parameters.

From a modeling perspective, MLP weights are structurally more regular and predictable than those of attention modules, which involve complex tensor reshaping, head projections, and positional dynam-

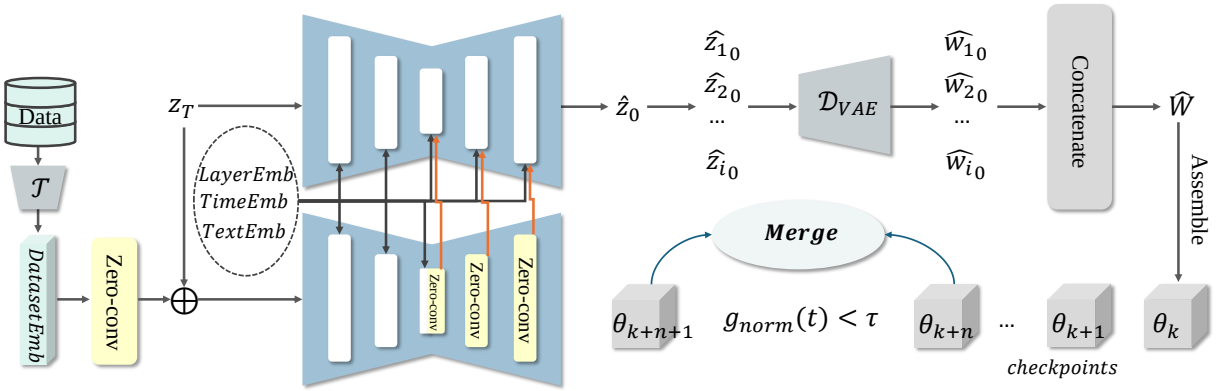

Figure 6: Inference process with dataset-conditioned diffusion: (1) encode dataset and prompt; (2) initialize layer-wise latents from the intermediate checkpoint; (3) run conditional reverse diffusion with ControlNet; (4) decode weights via VAE; (5) merge checkpoints if the update stabilizes, yielding the final model.

Table 3: Generalization of model merging across datasets, architectures, and merge strategies. Each row reports the average accuracy gain (ΔAcc) of a merged checkpoint over the later checkpoint alone, along with standard deviation, Pearson correlation r between ΔAcc and the later checkpoint's accuracy, and significance p.

Dataset	Model	Merge Strategy	Mean ΔAcc	Std ΔAcc	Pearson r	p-value
Wikipedia	GPT-2 small	Uniform Soup	0.048	0.012	0.62	0.003
Wikipedia	GPT-2 small	Greedy Soup	0.052	0.015	0.65	0.001
Story	GPT-2 large	Uniform Soup	0.037	0.010	0.57	0.005
Story	GPT-2 large	Greedy Soup	0.041	0.013	0.60	0.004
News	GPT-2 XL	Uniform Soup	0.045	0.014	0.59	0.002
News	GPT-2 XL	Greedy Soup	0.049	0.016	0.63	0.001

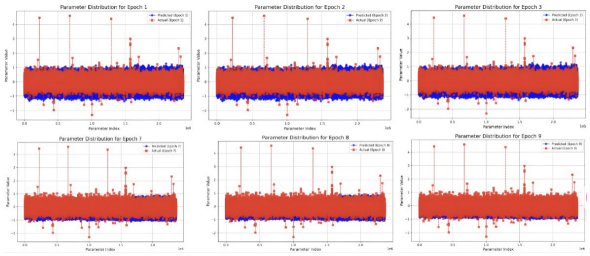

Figure 7: **Weight distribution visualization.** A 2D projection comparing finetuned versus diffusion-predicted weight distributions. The visualization highlights the structural similarities and distribution shifts between the two optimization paths.

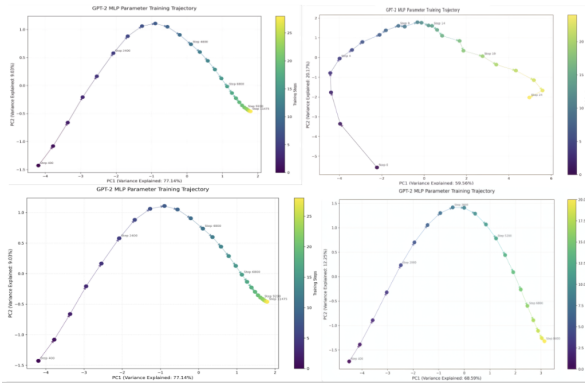

Figure 8: Training trajectory of GPT-2 MLP parameters visualized across principal components, with varying variance explained by PC1 (17.14%-30.33%).

ics. This makes MLPs far more suitable for learning with generative models like diffusion, VAE encoders, or U-Net-style predictors, as their shapes and transformations are consistent and easier to reconstruct.

In summary, MLP layers serve as a critical backbone of both representation and adaptability in LLMs. They offer the ideal balance of semantic richness, memory encoding, and structural regularity—making them the most strategic and effective target for parameter prediction in our diffusion-based generative framework. This design choice is not only theoretically motivated but also empirically validated by the improvements we observe in downstream performance.

Table 4: MMLU Task Accuracy Comparison

Category	Subtask	Dir.	Original	Finetuned	Ours	Delta (Ours - Orig)	Gap (Ours - Fine)
Humanities	Formal Logic	↑	0.2778	0.3680	0.3702	+0.0924	+0.0022
	High School European History	↑	0.2182	0.2560	0.2584	+0.0402	+0.0024
	High School US History	↑	0.2500	0.2561	0.2577	+0.0077	+0.0016
	Logical Fallacies	↑	0.2209	0.2350	0.2371	+0.0162	+0.0021
	Moral Scenarios	↑	0.2380	0.2741	0.2760	+0.0380	+0.0019
	Philosophy	↑	0.1897	0.2430	0.2454	+0.0557	+0.0024
	Prehistory	↑	0.2160	0.2268	0.2280	+0.0120	+0.0012
Other	Clinical Knowledge	↑	0.2075	0.2995	0.3020	+0.0945	+0.0025
	College Medicine	↑	0.2081	0.3365	0.3384	+0.1303	+0.0019
	Management	↑	0.1748	0.3799	0.3821	+0.2073	+0.0022
	Nutrition	↑	0.2157	0.2957	0.2978	+0.0821	+0.0021
	Professional Accounting	↑	0.2305	0.2420	0.2432	+0.0127	+0.0012
	Professional Medicine	↑	0.1949	0.4498	0.4520	+0.2571	+0.0022
Social Sciences	High School Geography	↑	0.1818	0.3548	0.3564	+0.1746	+0.0016
	Government & Politics	↑	0.1969	0.3691	0.3709	+0.1740	+0.0018
	Macroeconomics	↑	0.2026	0.3655	0.3674	+0.1648	+0.0019
	Microeconomics	↑	0.2101	0.3499	0.3516	+0.1415	+0.0017
	Psychology	↑	0.1908	0.3498	0.3512	+0.1604	+0.0014
	Human Sexuality	↑	0.2595	0.2835	0.2851	+0.0256	+0.0016
	Security Studies	↑	0.1878	0.4015	0.4033	+0.2155	+0.0018
	Sociology	↑	0.2438	0.2700	0.2722	+0.0284	+0.0022
STEM	Anatomy	↑	0.2000	0.2310	0.2333	+0.0333	+0.0023
	Astronomy	↑	0.1776	0.3369	0.3390	+0.1614	+0.0021
	College Biology	↑	0.2569	0.2650	0.2668	+0.0099	+0.0018
	College Chemistry	↑	0.1900	0.4115	0.4137	+0.2237	+0.0022
	Computer Science	↑	0.2500	0.3313	0.3330	+0.0830	+0.0017
	Mathematics	↑	0.2100	0.3115	0.3136	+0.1036	+0.0021
	Physics	↑	0.2157	0.3739	0.3760	+0.1603	+0.0021
	Elementary Mathematics	↑	0.2090	0.2688	0.2705	+0.0615	+0.0017
	High School Biology	↑	0.1774	0.3175	0.3196	+0.1422	+0.0021
	High School Chemistry	↑	0.1527	0.2821	0.2845	+0.1318	+0.0024
	High School Physics	↑	0.1987	0.3325	0.3347	+0.1360	+0.0022
	High School Statistics	↑	0.1528	0.4735	0.4759	+0.3231	+0.0024

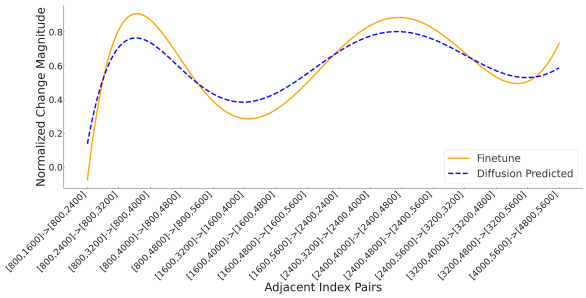

Figure 9: **Analysis of merging coefficients.** Trends of normalized alpha changes between finetuned and diffusion-predicted weights across adjacent index pairs. We sample 20 candidate coefficients (0.05–0.95) via Bayesian methods, selecting optimal values based on accuracy.