CNSP: CONSISTENT NULL-SPACE PROJECTION FOR PRINCIPLED PROMPT-BASED CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompt-based continual learning has recently shown strong empirical progress, yet its theoretical underpinnings remain incomplete. Prior work such as NSP^2 provides sufficient conditions for performance preservation for visual prompt tuning via null-space projection and achieves strong empirical results, but its reliance on simplifying assumptions in MHSA and LayerNorm undermines robustness and interpretability. In this paper, we revisit the problem from a matrix-level perspective and propose Consistent Null-Space Projection (CNSP). Our framework introduces: (i) rigorous per-head derivations under MHSA; (ii) a matrix-form characterization of LayerNorm; (iii) a relaxed prompt variance constraint that is more stable in practice; and (iv) refined sufficient conditions enforced via null-space projection that extend naturally to classification heads, ensuring end-to-end task performance preservation. Extensive experiments on multiple benchmarks demonstrate that CNSP consistently improves over NSP^2. Our results highlight the importance of principled matrix-level formulations for building robust and interpretable prompt-based continual learning methods.

1 INTRODUCTION

Continual learning seeks to enable models to acquire new knowledge from sequential tasks without forgetting previously learned knowledge. Despite significant progress, catastrophic forgetting (McCloskey & Cohen, 1989), where performance on old tasks deteriorates when learning new ones, remains a major challenge (Wang et al., 2024).

Recently, prompt-tuning methods, particularly Visual Prompt Tuning (VPT) (Jia et al., 2022) with Vision Transformers (ViTs) (Dosovitskiy et al., 2020), have emerged as a promising rehearsal-free solution for continual learning (Wang et al., 2024). By updating only lightweight prompts while freezing the backbone, these methods achieve both computational and storage efficiency. While several prompt-based methods for continual learning (e.g., L2P (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), Coda-Prompt (Smith et al., 2023)) have demonstrated strong empirical performance, a theoretical understanding of how prompts mitigate interference is still limited. In classical continual learning, orthogonal projection methods have provided a principled framework to prevent forgetting in linear and convolutional networks by deriving explicit linear consistency conditions to ensure knowledge retention (Saha et al., 2021; Deng et al., 2021; Wang et al., 2021). Yet, these guarantees do not transfer to ViTs: nonlinear LayerNorm, QKV projections, and softmax attention fundamentally break the linear assumptions.

To bridge this gap, recent efforts have adapted projection ideas to ViTs/VPT (Qiao et al., 2024; Lu et al., 2024). PGP (Qiao et al., 2024) approximates MHSA and LayerNorm as linear mappings and constructs orthogonal subspaces in the sum space of inputs and prompts, but its strong simplification limits generality. NSP^2 (Lu et al., 2024) provides a more systematic analysis of the ViT layer in VPT, deriving sufficient conditions for task performance preservation and proposing a null-space projection algorithm with strong empirical results. However, NSP^2's treatment of LayerNorm and multi-head attention relies on simplifications, leaving its theoretical guarantees incomplete and limiting both robustness and interpretability. In this work, we revisit the problem from a finer-grained matrix-level perspective. Building on NSP^2, we propose **Consistent Null-Space Projection (CNSP)**, which establishes stricter sufficient conditions under multi-head attention and matrix-form LayerNorm, together with feasible optimization strategies. Specifically, through reproducing and

analyzing NSP^2, we identify four key limitations: (i) its extension from single-head to multi-head attention relies on concatenation without rigorous justification; (ii) its derivation treats LayerNorm broadcast operations as scalar quantities, which are then cancelled from both sides of the equations, leading to an oversimplification; (iii) it enforces both mean and variance invariance of prompts, a strong and unstable assumption in practice; and (iv) its derivation overlooks the classification head, leaving analysis incomplete.

We address these issues and make the following contributions:

(i) **Rigorous multi-head analysis.** We derive explicit per-head sufficient conditions, establishing a solid theoretical foundation for performance preservation in MHSA.
(ii) **Matrix-form LayerNorm modeling.** We reformulate LayerNorm's broadcast operations in matrix form, ensuring algebraic rigor.
(iii) **Relaxed distributional constraint.** We show that variance invariance of prompts alone suffices, relaxing NSP^2's stronger mean–variance assumption.
(iv) **Classification head preservation.** We extend task performance preservation beyond attention layers to classification heads, ensuring theoretical and practical end-to-end consistency from representation to decision.
(v) **Empirical gains.** Across benchmarks including CIFAR-100, ImageNet-R, and DomainNet, our method consistently outperforms NSP^2 in both average accuracy and forgetting, and achieves performance that is competitive with leading prompt-based continual learning approaches.

2 Related Works

2.1 Prompt-based Methods for Continual Learning

Prompt-based methods have emerged as a promising paradigm for continual learning, enabling efficient task adaptation through lightweight modules while reducing task interference. Early approaches include L2P (Wang et al., 2022b), which employs a learnable prompt pool with dynamic prompt selection, and DualPrompt (Wang et al., 2022a), which further disentangles prompts into general and expert subsets to balance knowledge sharing and task-specific adaptation. CPrompt (Gao et al., 2024) addresses train–test inconsistency via classifier- and prompt-consistent learning, while CODA-Prompt (Smith et al., 2023) decomposes prompts into fine-grained components and dynamically recombining them through attention. EvoPrompt (Kurniawan et al., 2024) formulates prompt learning as evolutionary search for long-term adaptability.

Beyond purely visual prompts, LGCL (Khan et al., 2023) incorporates semantic guidance by aligning visual representations with language embeddings at both task and class levels, improving interpretability and transferability. ConvPrompt (Roy et al., 2024) leverages convolutional structures to construct hierarchical prompts and employs large language models to estimate task similarity, improving cross-task knowledge transfer and reducing redundancy.

More recent efforts emphasize principled formulations of prompt learning. PGP (Qiao et al., 2024) enforces gradient orthogonality in the joint input–prompt space by linearizing the nonlinear effects of MHSA and LayerNorm, while NSP^2 (Lu et al., 2024) derives null-space projection conditions for VPT by explicitly analyzing attention and LayerNorm. CPG (Lu et al., 2025) further introduces a consistency-constrained Mixture-of-Experts framework to ensure stable and generalizable prompt generation across tasks.

Overall, prompt-based methods have shown strong versatility in continual learning, but their theoretical foundations remain underdeveloped. NSP^2 makes an important step with null-space projection for VPT, yet its simplified treatment of multi-head attention and LayerNorm reduces robustness and interpretability. We address these limitations by re-deriving sufficient conditions for multi-head attention and formalizing LayerNorm in matrix form, yielding a more principled and effective framework.

2.2 Orthogonal Projection Methods for Continual Learning

Orthogonal projection methods have been widely studied in CNNs and MLPs (e.g., OWM (Zeng et al., 2019), GPM (Saha et al., 2021), Adam-NSCL (Wang et al., 2021)) as a principled way

to reduce task interference. The core idea is to preserve past feature responses by constraining parameter updates within subspaces orthogonal to previous tasks' features, formalized as $\boldsymbol{X}^\top(\boldsymbol{\Theta}+\Delta\boldsymbol{\Theta})=\boldsymbol{X}^\top\boldsymbol{\Theta}$, where $\boldsymbol{X}$ denotes input features, $\boldsymbol{\Theta}$ the convolutional or linear parameters, and $\Delta\boldsymbol{\Theta}$ the update. Enforcing $\boldsymbol{X}^\top\Delta\boldsymbol{\Theta}=\mathbf{0}$ guarantees invariance of past intermediate representations, thereby retaining knowledge. This principle, rooted in gradient limitation theory, provides a mathematical explanation of the stability-plasticity trade-off. Representative methods include GPM Saha et al. (2021), which projects gradients onto the null space of previously learned inputs; TRGP (Lin et al., 2022b), which refines GPM by scaling task gradients with a trust region matrix before projection; and Connector (Lin et al., 2022a) interpolates a normally updated model with one constrained by projection.

These methods are effective in linear or convolutional architectures, where the orthogonality condition holds exactly. In transformers, however, nonlinear interactions QKV interactions, softmax normalization, and distributional shifts from LayerNorm collectively violate the linearity, making directly applying classical methods to ViTs/VPT unreliable. To address this, recent works have revisited orthogonal projection in prompt tuning (Qiao et al., 2024; Lu et al., 2024). Notably, NSP^2 (Lu et al., 2024) unfolds the forward propagation of VPT in ViT layers and derives sufficient conditions for performance preservation, enforced approximately through null-space projection. This framework successfully adapts classical projection principles to transformer architectures, but still relies on oversimplifying assumptions. Building on this foundation, we strengthen the theoretical guarantees of projection-based continual learning in VPT by introducing stricter and more general consistency conditions together with stable optimization strategies.

3 PRELIMINARIES

In this section, we introduce the notations and conventions, describe the forward propagation process of transformer blocks under VPT-deep (Jia et al., 2022)—where learnable prompts are inserted into the input token sequence at each block for efficient adaptation with a frozen backbone—and formalize the continual learning problem in this setting.

3.1 NOTATIONS

We use the following notational conventions: (i) Non-bold letters denote positive integers, e.g., $a, A \in \mathbb{N}^+$. (ii) Bold lowercase denote vectors, $\boldsymbol{a} \in \mathbb{R}^n$; bold uppercase denote matrices, $\boldsymbol{A} \in \mathbb{R}^{n\times m}$. (iii) Column concatenation: for $\boldsymbol{A}^{(i)} \in \mathbb{R}^{n\times d}$, $[\boldsymbol{A}^{(i)}]_{i=1}^{N} = \begin{bmatrix}\boldsymbol{A}^{(1)} & \dots & \boldsymbol{A}^{(N)}\end{bmatrix} \in \mathbb{R}^{n\times Nd}$. (iv) Row concatenation: for $\boldsymbol{A}^{(i)} \in \mathbb{R}^{n\times d}$, $\left[\boldsymbol{A}^{(1)};\dots;\boldsymbol{A}^{(N)}\right] \in \mathbb{R}^{Nn\times d}$. (v) Slicing: if $\boldsymbol{A} = \left[\boldsymbol{A}^{(1)};\boldsymbol{A}^{(2)}\right]$ with $\boldsymbol{A}^{(1)} \in \mathbb{R}^{n\times d}$, then $\boldsymbol{A}[:n] = \boldsymbol{A}^{(1)}$.

3.2 FORWARD PROPAGATION IN VPT-DEEP

Let $\boldsymbol{X} \in \mathbb{R}^{(N+1)\times D}$ denote the patch embeddings of an input image, where N is the number of patches, the first row is the pre-trained [CLS] token, and D the embedding dimension. A standard ViT with depth L can be written as $f_{\text{ViT}}\left(\boldsymbol{X} \mid \boldsymbol{\Theta}; \{\boldsymbol{W}, \boldsymbol{b}\}\right)$, where $\boldsymbol{\Theta}$ are frozen backbone parameters and $\{\boldsymbol{W}, \boldsymbol{b}\}$ are learnable classification head parameters. Let $\boldsymbol{X}^{(l)} \in \mathbb{R}^{(N+1)\times D}$ denote the input embeddings of the l-th block.

In VPT-deep, each block l introduces M learnable prompts $\boldsymbol{P}^{(l)} \in \mathbb{R}^{M\times D}$, concatenated with image embeddings to form the input sequences: $\boldsymbol{Z}^{(l)} = \left[\boldsymbol{X}^{(l)};\boldsymbol{P}^{(l)}\right] \in \mathbb{R}^{(N+1+M)\times D}$, $\boldsymbol{X}^{(1)} = \boldsymbol{X}$. The forward propagation in VPT-deep is illustrated in fig. 1. Each transformer block applies LayerNorm (LN), Multi-Head Self-Attention (MHSA), and an MLP with residual connections:

$$\boldsymbol{O}_{\boldsymbol{Z}}^{(l)} = \text{MHSA}(\text{LN}(\boldsymbol{Z}^{(l)})), \quad \boldsymbol{B}_{\boldsymbol{Z}}^{(l)} = \boldsymbol{Z}^{(l)} + \boldsymbol{O}_{\boldsymbol{Z}}^{(l)}, \quad \boldsymbol{E}_{\boldsymbol{Z}}^{(l)} = \boldsymbol{B}_{\boldsymbol{Z}}^{(l)} + \text{MLP}(\text{LN}(\boldsymbol{B}_{\boldsymbol{Z}}^{(l)})). \tag{1}$$

Before the next block, the prompts are removed and replaced with a new set of learnable prompts:

$$\boldsymbol{X}^{(l+1)} = \boldsymbol{E}_{\boldsymbol{Z}}^{(l)}[:N+1], \quad \boldsymbol{Z}^{(l+1)} = [\boldsymbol{X}^{(l+1)};\boldsymbol{P}^{(l+1)}]. \tag{2}$$

Thus, prompts modulate intermediate features at each block but do not persist across blocks. The final [CLS] token is passed into the classification head:

$$f_{\text{ViT}}\left(\boldsymbol{X} \mid \boldsymbol{\Theta}; \{\boldsymbol{W}, \boldsymbol{b}\}; \{\boldsymbol{P}^{(l)}\}_{l=1}^{L}\right) = softmax\left(\boldsymbol{E}_{\boldsymbol{Z}}^{(L)}[:1]\boldsymbol{W} + \boldsymbol{b}\right), \tag{3}$$

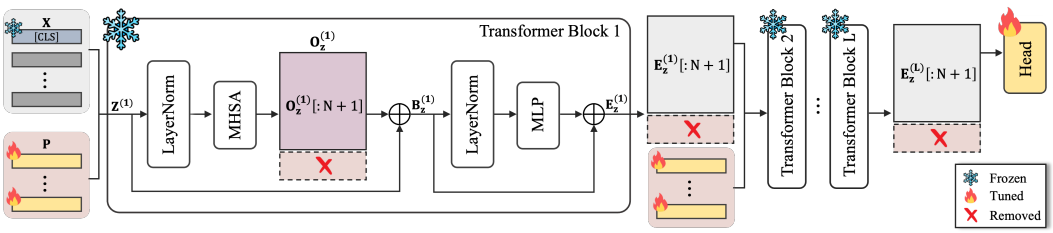

Figure 1: Forward propagation in VPT. Image embeddings ($\boldsymbol{X}$) and learnable prompt embeddings ($\boldsymbol{P}$) are jointly processed by LayerNorm, MHSA, and MLP with residual connections in each transformer block. Modules marked with ❄ are frozen, those with 🔥 are trainable, and those with ✗ indicate information discarded before entering the next layer.

where $\boldsymbol{W} \in \mathbb{R}^{D\times C}$, $\boldsymbol{b} \in \mathbb{R}^{C}$, and C is the number of classes.

3.3 Problem Formulation: Continual Learning with VPT-deep

We consider a continual learning setting with a sequence of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$. Each task $\mathcal{T}_t$ is associated with a dataset $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{N_t}$, where $\boldsymbol{x}_t^{(i)}$ is an image and $y_t^{(i)}$ its label. At block l, let $\boldsymbol{P}_t^{(l)}$ be the prompts after learning tasks up to $\mathcal{T}_t$. For the next task $\mathcal{T}_{t+1}$, updates are:

$$\boldsymbol{P}_{t+1}^{(l)} = \boldsymbol{P}_t^{(l)} + \Delta\boldsymbol{P}^{(l)}, \quad \boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \Delta\boldsymbol{W}, \quad \boldsymbol{b}_{t+1} = \boldsymbol{b}_t + \Delta\boldsymbol{b}, \tag{4}$$

The continual learning objective requires preserving performance on previous tasks after training on a new one. We formalize the objective as:

$$f_{\text{ViT}}(\boldsymbol{X}_t \mid \boldsymbol{\Theta}; \{\boldsymbol{W}_{t+1}, \boldsymbol{b}_{t+1}\}; \{\boldsymbol{P}_{t+1}^{(l)}\}_{l=1}^{L}) = f_{\text{ViT}}(\boldsymbol{X}_t \mid \boldsymbol{\Theta}; \{\boldsymbol{W}_t, \boldsymbol{b}_t\}; \{\boldsymbol{P}_t^{(l)}\}_{l=1}^{L}), \tag{5}$$

which enforces that model outputs on old inputs $\boldsymbol{X}_t$ from $\mathcal{T}_t$ must remain unchanged. Equivalently, updates $\{\Delta\boldsymbol{P}^{(l)}\}_{l=1}^{L}$ and $\{\Delta\boldsymbol{W}, \Delta\boldsymbol{b}\}$ must not affect the predictions for previously learned tasks. To characterize this objective, we introduce two key propositions that decompose it into two complementary requirements: **feature preservation** at intermediate blocks and **head preservation** at the classification head.

Proposition 1. *Consider two tasks $\mathcal{T}_t$ and $\mathcal{T}_{t+1}$. If the image-token part of attention outputs remains unchanged for all blocks in VPT, i.e.,*

$$\boldsymbol{O}_{\boldsymbol{Z}_{t,t+1}}^{(l)}[:N+1] = \boldsymbol{O}_{\boldsymbol{Z}_{t,t}}^{(l)}[:N+1], \quad l = 1, 2, \ldots, L, \tag{6}$$

then the image-token output of the final block is preserved:

$$\boldsymbol{E}_{\boldsymbol{Z}_{t,t+1}}^{(L)}[:N+1] = \boldsymbol{E}_{\boldsymbol{Z}_{t,t}}^{(L)}[:N+1], \tag{7}$$

*which we refer to as **feature preservation**. Here $\boldsymbol{O}_{\boldsymbol{Z}_{t,t+1}}^{(l)}$, $\boldsymbol{O}_{\boldsymbol{Z}_{t,t}}^{(l)}$, $\boldsymbol{E}_{\boldsymbol{Z}_{t,t+1}}^{(L)}$, and $\boldsymbol{E}_{\boldsymbol{Z}_{t,t}}^{(L)}$ are computed by eq.* (1)*. In particular, the final [CLS] token is preserved:*

$$\boldsymbol{E}_{\boldsymbol{Z}_{t,t+1}}^{(L)}[:1] = \boldsymbol{E}_{\boldsymbol{Z}_{t,t}}^{(L)}[:1]. \tag{8}$$

Proof Sketch. By eq. (2), since prompts are replaced at each block, their effect propagates only through the image-token outputs $\boldsymbol{E}_{\boldsymbol{Z}}^{(l)}[:N+1]$. Row-wise independence of post-MHSA operations ensures invariance of $\boldsymbol{O}_{\boldsymbol{Z}}^{(l)}[:N+1]$ carries forward. Full proof is provided in Appendix A.

Proposition 2. *Consider tasks $\mathcal{T}_t$ and $\mathcal{T}_{t+1}$. If the final [CLS] token remains unchanged (i.e., eq.* (8) *holds) and the head updates satisfy:*

$$\boldsymbol{E}_{\boldsymbol{Z}_{t,t}}^{(L)}[:1]\Delta\boldsymbol{W} + \Delta\boldsymbol{b} = \boldsymbol{0}, \tag{9}$$

*which we refer to as **head preservation**, then the continual learning objective eq.* (5) *is satisfied.*

Proof. From eq. (8) and eq. (9), we have

$$\boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t+1}}[:1]\boldsymbol{W}_{t+1}+\boldsymbol{b}_{t+1}=\boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t+1}}[:1]\left(\boldsymbol{W}_t+\Delta\boldsymbol{W}\right)+\left(\boldsymbol{b}_t+\Delta\boldsymbol{b}\right)=\boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t}}[:1]\boldsymbol{W}_t+\boldsymbol{b}_t, \quad (10)$$

which, by eq. (3), implies eq. (5). □

Together, Propositions 1 and 2 reduce continual learning preservation to two conditions: feature preservation, requiring prompt updates to leave image-token representations invariant for representational stability, and head preservation, requiring the classifier to maintain consistent mappings of the preserved [CLS] token. This decomposition of objective eq. (5) forms the theoretical basis for the sufficient conditions derived in the next section. Particularly, proposition 1 implies intermediate feature preservation reduces to ensuring invariance at the MHSA output. Therefore, in the derivations presented in the next section, we focus on LayerNorm and MHSA, as they are the only operations directly relevant to feature preservation.

4 METHOD

Our method builds on Propositions 1 and 2, enforcing both feature and head preservation. Feature preservation is ensured through explicit constraints on prompt updates and head preservation guaranteed by design. According to proposition 1, the preservation objective can be analyzed blockwise by focusing solely on the MHSA and its preceding LayerNorm operation. Other components (e.g., MLP) are row-wise and do not alter the preservation property. Since eq. (6) must be satisfied at every layer, we omit the superscript (l)) when deriving algebraic sufficient conditions on prompt updates.

4.1 UNFOLDING FORWARD PROPAGATION OF LAYERNORM AND MHSA IN A SINGLE BLOCK

Let the input sequence be $\boldsymbol{Z}_{t,t+1}=[\boldsymbol{X}_t;\boldsymbol{P}_{t+1}]\in\mathbb{R}^{(N+1+M)\times D}$, where $\boldsymbol{P}_{t+1}=\boldsymbol{P}_t+\Delta\boldsymbol{P}$. Figure 2 illustrates the unrolled forward pass of LayerNorm (LN) and MHSA in a single transformer block. In this setting, computations involving $\boldsymbol{Q}^{(h)}_{\boldsymbol{P}_{t+1}}$ can be safely omitted; justification is provided below.

First, $\boldsymbol{Z}_{t,t+1}$ is passed through LayerNorm, we introduce the following lemma:

Lemma 1. *Consider the computation of LayerNorm* $\mathrm{LN}(\cdot)$. *Let* $\boldsymbol{A}=[\boldsymbol{A}^{(1)};\boldsymbol{A}^{(2)}]$ *with* $\boldsymbol{A}^{(1)}\in\mathbb{R}^{n\times d}$ *and* $\boldsymbol{A}^{(1)}\in\mathbb{R}^{m\times d}$. *Then* $\mathrm{LN}\left(\boldsymbol{A}\right)=\left[\mathrm{LN}\left(\boldsymbol{A}^{(1)}\right);\,\mathrm{LN}\left(\boldsymbol{A}^{(2)}\right)\right]$.

Proof of Lemma 1 is in Appendix A. According to lemma 1, we have $\mathrm{LN}\left(\boldsymbol{Z}_{t,t+1}\right)=\left[\mathrm{LN}\left(\boldsymbol{X}_t\right);\,\mathrm{LN}\left(\boldsymbol{P}_t+\Delta\boldsymbol{P}\right)\right]$. Next, the normalized sequence undergoes QKV transformation, including QKV projections and multi-head splitting. For head $h\in\{1,\ldots,H\}$, we have:

$$\boldsymbol{\Phi}^{(h)}_{\boldsymbol{Z}_{t,t+1}}=\mathrm{LN}(\boldsymbol{Z}_{t,t+1})\,\boldsymbol{W}^{(h)}_{\phi}+\mathbf{1}_{N+1+M}\boldsymbol{b}^{(h)\top}_{\phi}=\begin{bmatrix}\mathrm{LN}(\boldsymbol{X}_t)\,\boldsymbol{W}^{(h)}_{\phi}+\mathbf{1}_{N+1}\boldsymbol{b}^{(h)\top}_{\phi}\\ \mathrm{LN}(\boldsymbol{P}_t+\Delta\boldsymbol{P})\,\boldsymbol{W}^{(h)}_{\phi}+\mathbf{1}_{M}\boldsymbol{b}^{(h)\top}_{\phi}\end{bmatrix}, \quad (11)$$

where $\boldsymbol{\Phi}\in\{\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}\}$, ϕ serves as the symbolic index for q,k,v, and $\boldsymbol{W}^{(h)}_{\phi}\in\mathbb{R}^{D\times D_H}$, $\boldsymbol{b}^{(h)}_{\phi}\in\mathbb{R}^{D_H}$, and $D_H=D/H$. Let

$$\boldsymbol{Q}^{(h)}_{\boldsymbol{Z}_{t,t+1}}=\left[\boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t};\,\boldsymbol{Q}^{(h)}_{\boldsymbol{P}_{t+1}}\right],\quad \boldsymbol{K}^{(h)}_{\boldsymbol{Z}_{t,t+1}}=\left[\boldsymbol{K}^{(h)}_{\boldsymbol{X}_t};\,\boldsymbol{K}^{(h)}_{\boldsymbol{P}_{t+1}}\right],\quad \boldsymbol{V}^{(h)}_{\boldsymbol{Z}_{t,t+1}}=\left[\boldsymbol{V}^{(h)}_{\boldsymbol{X}_t};\,\boldsymbol{V}^{(h)}_{\boldsymbol{P}_{t+1}}\right], \quad (12)$$

where $\boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t},\boldsymbol{K}^{(h)}_{\boldsymbol{X}_t},\boldsymbol{V}^{(h)}_{\boldsymbol{X}_t}\in\mathbb{R}^{(N+1)\times D_H}$ and $\boldsymbol{Q}^{(h)}_{\boldsymbol{P}_{t+1}},\boldsymbol{K}^{(h)}_{\boldsymbol{P}_{t+1}},\boldsymbol{V}^{(h)}_{\boldsymbol{P}_{t+1}}\in\mathbb{R}^{M\times D_H}$.

For each head h, the attention weights and value aggregation are computed as

$$\boldsymbol{H}^{(h)}_{\boldsymbol{Z}_{t,t+1}}=softmax\Big(\boldsymbol{Q}^{(h)}_{\boldsymbol{Z}_{t,t+1}}{\boldsymbol{K}^{(h)}_{\boldsymbol{Z}_{t,t+1}}}^{\top}/\sqrt{D_H}\Big)\boldsymbol{V}^{(h)}_{\boldsymbol{Z}_{t,t+1}}, \quad (13)$$

where the softmax function acts on the rows of matrix $\boldsymbol{Q}^{(h)}_{\boldsymbol{Z}_{t,t+1}}{\boldsymbol{K}^{(h)}_{\boldsymbol{Z}_{t,t+1}}}^{\top}$. Finally, concatenating across heads and applying the output projection gives

$$\boldsymbol{O}_{\boldsymbol{Z}_{t,t+1}}=\left[\boldsymbol{H}^{(h)}_{\boldsymbol{Z}_{t,t+1}}\right]_{h=1}^{H}\boldsymbol{W}_o+\mathbf{1}_{N+1+M}\boldsymbol{b}_o^{\top}. \quad (14)$$

By proposition 1, feature preservation requires $\boldsymbol{O}_{\boldsymbol{Z}_{t,t+1}}[:N+1] = \boldsymbol{O}_{\boldsymbol{Z}_{t,t}}[:N+1]$. Note that

$$\left[\boldsymbol{H}^{(h)}_{\boldsymbol{Z}_{t,t+1}}\right]_{h=1}^{H}[:N+1] = \left[softmax(\boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}{\boldsymbol{K}^{(h)}_{\boldsymbol{Z}_{t,t+1}}}^{\top}/\sqrt{D_H})\boldsymbol{V}^{(h)}_{\boldsymbol{Z}_{t,t+1}}\right]_{h=1}^{H}. \tag{15}$$

Therefore, only $\boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}$ contributes to the image-token outputs, terms from $\boldsymbol{Q}^{(h)}_{\boldsymbol{P}_{t+1}}$ can be omitted in the preservation analysis.

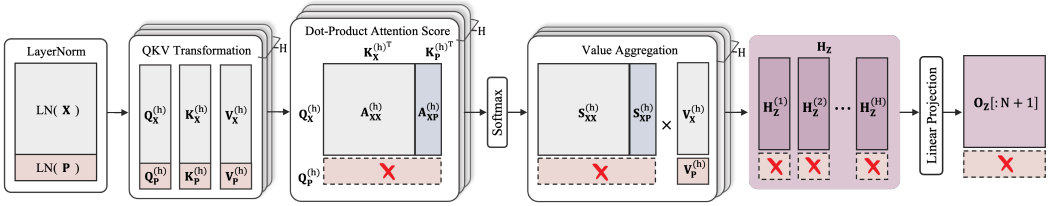

Figure 2: Detailed illustration of LayerNorm and the multi-head self-attention mechanism in a VPT Transformer block, showing how image and prompt embeddings are jointly processed. Those with ✗ indicate information discarded before entering the next layer.

4.2 Sufficient Conditions for Feature Preservation in VPT

Building on the above expansion, we restrict attention to the $\boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}$-dependent pathways. Define

$$\boldsymbol{A}^{(h)}_{\boldsymbol{Z}_{t,t+1}} = \frac{\boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}{\boldsymbol{K}^{(h)}_{\boldsymbol{Z}_{t,t+1}}}^{\top}}{\sqrt{D_H}} = \frac{1}{\sqrt{D_H}}\boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}\left[{\boldsymbol{K}^{(h)}_{\boldsymbol{X}_t}}^{\top} \quad {\boldsymbol{K}^{(h)}_{\boldsymbol{P}_{t+1}}}^{\top}\right] \triangleq \left[\boldsymbol{A}^{(h)}_{\boldsymbol{X}_t\boldsymbol{X}_t} \quad \boldsymbol{A}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_{t+1}}\right], \tag{16}$$

$$\boldsymbol{S}^{(h)}_{\boldsymbol{Z}_{t,t+1}} = softmax\left(\left[\boldsymbol{A}^{(h)}_{\boldsymbol{X}_t\boldsymbol{X}_t} \quad \boldsymbol{A}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_{t+1}}\right]\right) \triangleq \left[\boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{X}_t} \quad \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_{t+1}}\right], \tag{17}$$

where $\boldsymbol{A}^{(h)}_{\boldsymbol{X}_t\boldsymbol{X}_t}, \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{X}_t} \in \mathbb{R}^{(N+1)\times(N+1)}$ and $\boldsymbol{A}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_{t+1}}, \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_{t+1}} \in \mathbb{R}^{(N+1)\times M}$. (Symbol–module correspondence is depicted in 2.) By proposition 1, eq. (14) and eq. (15), feature preservation requires

$$\left[\boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{X}_t}\boldsymbol{V}^{(h)}_{\boldsymbol{X}_t} + \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_{t+1}}\boldsymbol{V}^{(h)}_{\boldsymbol{P}_{t+1}}\right]_{h=1}^{H}\boldsymbol{W}_o = \left[\boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{X}_t}\boldsymbol{V}^{(h)}_{\boldsymbol{X}_t} + \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_t}\boldsymbol{V}^{(h)}_{\boldsymbol{P}_t}\right]_{h=1}^{H}\boldsymbol{W}_o \tag{18}$$

In MHSA, $\boldsymbol{W}_o$ fuses per-head outputs into a unified representation (Vaswani et al., 2017; Horn & Johnson, 2012). Since $\boldsymbol{W}_o$ may in principle be singular, we impose stronger sufficient conditions by requiring per-head equality:

$$\boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_{t+1}}\boldsymbol{V}^{(h)}_{\boldsymbol{P}_{t+1}} - \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_t}\boldsymbol{V}^{(h)}_{\boldsymbol{P}_t} = \boldsymbol{0}. \tag{19}$$

Direct algebraic characterization is intractable due to the nonlinearity of softmax. Following NSP^2 (Lu et al., 2024), we instead require invariance of the attention scores:

$$\boldsymbol{A}^{(h)}_{\boldsymbol{Z}_{t,t}} = \boldsymbol{A}^{(h)}_{\boldsymbol{Z}_{t,t+1}} \Leftrightarrow \boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}\left({\boldsymbol{K}^{(h)}_{\boldsymbol{P}_{t+1}}}^{\top} - {\boldsymbol{K}^{(h)}_{\boldsymbol{P}_t}}^{\top}\right) = \boldsymbol{0}. \tag{20}$$

That is, the attention scores from image tokens to prompts remain unchanged before and after the prompt update. Under eq. (20), condition eq. (19) reduces to

$$\boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_t}\left(\boldsymbol{V}^{(h)}_{\boldsymbol{P}_{t+1}} - \boldsymbol{V}^{(h)}_{\boldsymbol{P}_t}\right) = \boldsymbol{0}. \tag{21}$$

By substituting eq. (11), we obtain per-head sufficient conditions for feature preservation:

$$\begin{cases} \boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}\left({\boldsymbol{K}^{(h)}_{\boldsymbol{P}_{t+1}}}^{\top} - {\boldsymbol{K}^{(h)}_{\boldsymbol{P}_t}}^{\top}\right) = \boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}{\boldsymbol{W}^{(h)}_k}^{\top}\left(\mathrm{LN}\left(\boldsymbol{P}_t + \Delta\boldsymbol{P}\right) - \mathrm{LN}\left(\boldsymbol{P}_t\right)\right)^{\top} = \boldsymbol{0} \\ \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_t}\left(\boldsymbol{V}^{(h)}_{\boldsymbol{P}_{t+1}} - \boldsymbol{V}^{(h)}_{\boldsymbol{P}_t}\right) = \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t\boldsymbol{P}_t}\left(\mathrm{LN}\left(\boldsymbol{P}_t + \Delta\boldsymbol{P}\right) - \mathrm{LN}\left(\boldsymbol{P}_t\right)\right)\boldsymbol{W}^{(h)}_v = \boldsymbol{0} \end{cases} \tag{22}$$

To reduce the conditions to direct dependence on $\Delta \boldsymbol{P}$, we expand the LayerNorm:

$$\mathrm{LN}\left(\boldsymbol{P}_t + \Delta \boldsymbol{P}\right) = \frac{\boldsymbol{P}_t + \Delta \boldsymbol{P} - \boldsymbol{\mu}_{\boldsymbol{P}_t + \Delta \boldsymbol{P}}}{\boldsymbol{\sigma}_{\boldsymbol{P}_t + \Delta \boldsymbol{P}}} \cdot \boldsymbol{\gamma} + \boldsymbol{\beta}, \tag{23}$$

where $\boldsymbol{\mu}_{\boldsymbol{P}_t+\Delta \boldsymbol{P}}, \boldsymbol{\sigma}_{\boldsymbol{P}_t+\Delta \boldsymbol{P}} \in \mathbb{R}^M$ are row-wise statistics and $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^D$ are shared affine parameters (broadcast in implementation). To avoid broadcast ambiguity, we introduce LayerNorm in matrix form: let $\boldsymbol{C} = \boldsymbol{I}_D - \frac{1}{D}\mathbf{1}_D\mathbf{1}_D^\top (\in \mathbb{R}^{D\times D})$ be the centering matrix, $\boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{P}_t+\Delta \boldsymbol{P}}} (\in \mathbb{R}^{M\times M})$ [1] the diagonal scaling matrix from $\boldsymbol{\sigma}_{\boldsymbol{P}_t+\Delta \boldsymbol{P}}$, and $\boldsymbol{\Gamma} (\in \mathbb{R}^{D\times D})$ the diagonal matrix from $\boldsymbol{\gamma}$. Then

$$\mathrm{LN}\left(\boldsymbol{P}_t + \Delta \boldsymbol{P}\right) = \boldsymbol{D}^{-1}_{\boldsymbol{\sigma}_{\boldsymbol{P}_t+\Delta \boldsymbol{P}}} \left(\boldsymbol{P}_t + \Delta \boldsymbol{P}\right) \boldsymbol{C}\boldsymbol{\Gamma} + \mathbf{1}_M \boldsymbol{\beta}^\top. \tag{24}$$

Note that $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are frozen. If we introduce a variance-invariance assumption

$$\boldsymbol{\sigma}_{\boldsymbol{P}_t} = \boldsymbol{\sigma}_{\boldsymbol{P}_t+\Delta \boldsymbol{P}} \triangleq \boldsymbol{\sigma}, \tag{25}$$

then

$$\mathrm{LN}\left(\boldsymbol{P}_t + \Delta \boldsymbol{P}\right) - \mathrm{LN}\left(\boldsymbol{P}_t\right) = \boldsymbol{D}_{\boldsymbol{\sigma}}^{-1} \Delta \boldsymbol{P} \boldsymbol{C} \boldsymbol{\Gamma}. \tag{26}$$

Substituting eq. (26) into eq. (22), the sufficient constraints reduce to the following linear form:

$$\begin{cases} \boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t} {\boldsymbol{W}^{(h)}_k}^\top \boldsymbol{\Gamma} \boldsymbol{C} \Delta \boldsymbol{P}^\top = \mathbf{0} \\ \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t \boldsymbol{P}_t} \boldsymbol{D}_{\boldsymbol{\sigma}}^{-1} \Delta \boldsymbol{P} \boldsymbol{C} \boldsymbol{\Gamma} \boldsymbol{W}^{(h)}_v = \mathbf{0} \end{cases} \quad h = 1, 2, \ldots, H. \tag{27}$$

The first constraint ensures that, after centering and scaling, prompt updates do not perturb the scoring pattern from image tokens to prompts. The second ensures that they do not perturb the value aggregation pathway. Therefore, under the assumption eq. (25), eq. (27) provides a set of per-head sufficient conditions for feature preservation eq. (7).

4.3 Optimization under Feature Preservation Constraints

To enforce the variance-invariance assumption eq. (25), we adopt a soft constraint by introducing a standard deviation alignment loss:

$$\mathcal{L} = \mathcal{L}_{\mathrm{ce}} + \lambda \mathcal{L}_{\mathrm{std}}, \quad \mathcal{L}_{\mathrm{std}} = \|\boldsymbol{\sigma}_{\boldsymbol{P}_{t+1}} - \boldsymbol{\sigma}_{\boldsymbol{P}_t}\|_1, \tag{28}$$

where $\mathcal{L}$ is the total loss, $\mathcal{L}_{\mathrm{ce}}$ is the cross-entropy loss, and λ balances the alignment term. Under this assumption, we obtain the sufficient conditions in eq. (27), constraining $\Delta \boldsymbol{P}$. We adopt a right-side nullification form on the second constraint $\boldsymbol{S}^{(h)}_{\boldsymbol{X}_t \boldsymbol{P}_t} \boldsymbol{D}_{\boldsymbol{\sigma}}^{-1} \Delta \boldsymbol{P} \boldsymbol{C} \boldsymbol{\Gamma} \boldsymbol{W}^{(h)}_v = \mathbf{0}$. The rationale is to avoid circular dependence: if $\boldsymbol{D}_{\boldsymbol{\sigma}}^{-1}$ remains on the solving side, then $\Delta \boldsymbol{P}$ becomes entangled with the current statistic $\boldsymbol{\sigma}$, complicating implementation and hurting numerical stability. In contrast, the right-sided formulation $\Delta \boldsymbol{P} \boldsymbol{C} \boldsymbol{\Gamma} \boldsymbol{W}^{(h)}_v = \mathbf{0}$ removes explicit dependence on $\boldsymbol{\sigma}$, yielding more stable behavior. Moreover, it only involves frozen pretrained parameters, reducing the computational overhead. Let $\boldsymbol{R}^{(h)}_k = \boldsymbol{C} \boldsymbol{\Gamma} \boldsymbol{W}^{(h)}_k {\boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t}}^\top$, $\boldsymbol{R}^{(h)}_v = \boldsymbol{C} \boldsymbol{\Gamma} \boldsymbol{W}^{(h)}_v$, and define

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}^{(1)}_k & \cdots & \boldsymbol{R}^{(H)}_k & \boldsymbol{R}^{(1)}_v & \cdots & \boldsymbol{R}^{(H)}_v \end{bmatrix} \in \mathbb{R}^{D \times 2D}. \tag{29}$$

Our implementation enforces, at every optimization step, $\Delta \boldsymbol{P} \boldsymbol{R} = \mathbf{0}$. Equivalently, each row of $\Delta \boldsymbol{P}$ lies in the left null space of $\boldsymbol{R}$. Since the effective constraint size grows with tasks, handling $\boldsymbol{R}$ directly can be expensive. We thus use the following basic result:

Lemma 2. *For $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ with left null space $\mathcal{N}_L\left(\boldsymbol{A}\right) = \{\boldsymbol{x}\boldsymbol{A} = \mathbf{0} \mid \boldsymbol{x}^\top \in \mathbb{R}^n\}$, it holds that $\mathcal{N}_L\left(\boldsymbol{A}\right) = \mathcal{N}_L\left(\boldsymbol{A}\boldsymbol{A}^\top\right)$.*

Proof of lemma 2 is in Appendix A. Lemma 2 implies that $\Delta \boldsymbol{P} \boldsymbol{R} = \mathbf{0}$ is equivalent to requiring each row of $\Delta \boldsymbol{P}$ to lie in $\mathcal{N}_L\left(\boldsymbol{R}\boldsymbol{R}^\top\right)$, where $\boldsymbol{R}\boldsymbol{R}^\top \in \mathbb{R}^{D\times D}$ has fixed dimension during training. In practice, we perform Singular Value Decomposition (SVD) on $\boldsymbol{R}\boldsymbol{R}^\top$, collect the left singular vectors corresponding to zero singular values to form an orthogonal basis $\boldsymbol{U}_0$, so that $\mathcal{N}_L\left(\boldsymbol{R}\boldsymbol{R}^\top\right) = \mathrm{span}\left(\boldsymbol{U}_0\right)$ and its null-space projector is $\boldsymbol{B}_{\boldsymbol{R}} = \boldsymbol{U}_0\boldsymbol{U}_0^\top$. Let $\Delta \boldsymbol{P}_{\mathrm{raw}}$ denote the raw update from back-propagation. We apply the feature-preservation projection

$$\Delta \boldsymbol{P} \leftarrow \Delta \boldsymbol{P}_{\mathrm{raw}} \boldsymbol{B}_{\boldsymbol{R}}, \tag{30}$$

which guarantees that the feature-preservation sufficient conditions (i.e., eq. (27)) are satisfied.

[1] In practice, a small constant $\epsilon > 0$ is added to the variance when computing each row's standard deviation, ensuring positivity and numerical stability. Hence all standard deviations are strictly greater than zero, and the corresponding diagonal matrix is invertible.

4.4 Classification Head Preservation by Design

Proposition 2 requires that the classification head preserve its mapping of the [CLS] token across tasks. In our framework, this property is inherently guaranteed by design through task-specific classification heads, a strategy commonly adopted in prompt-based continual learning (Wang et al., 2022b; Lu et al., 2024; Wang et al., 2022a). During training, only the head for the current task is updated, while all previously learned heads remain frozen. Consequently, their mappings from the [CLS] token remain unchanged, directly satisfying the head preservation requirement. At inference, the outputs of all heads are concatenated, and prediction is made by selecting the class with the highest confidence, thereby eliminating the need for task identity. An illustration of this head design (fig. 5), together with a more detailed discussion of head preservation, is provided in Appendix E.1.

Summary. Our method enforces feature preservation through null-space projected updates and ensures head preservation by design, achieving end-to-end consistency across tasks. Detailed algorithmic descriptions are provided in Appendix B, and a comprehensive comparison with NSP^2 is given in Appendix C.

5 Experiments

Benchmarks and Evaluation Metrics. We evaluate CNSP on four benchmarks widely used in prompt-based continual learning: 10/20-Split CIFAR-100 (Krizhevsky et al., 2009), 10-Split ImageNet-R (Hendrycks et al., 2021), and 10-Split DomainNet (Peng et al., 2019). Performance is measured by Last Average Accuracy (ACC, ↑) and Forgetting (↓). Details of benchmarks and metric definitions are provided in Appendix D.1 and D.2

Implementation. We use the ImageNet-21K pretrained ViT-B/16 (Dosovitskiy et al., 2020) as the backbone, inserting four prompt tokens into each layer. Experiments are conducted under the class-incremental setting with disjoint classes and unknown task identity at inference. Models are trained with Adam (Kingma & Ba, 2015) for 10 epochs per task, using a loss that combines cross-entropy with prompt alignment regularizers (eq. (28)). Complete implementation details are provided in Appendix D.3.

Competitors. We compare CNSP against the state-of-the-art prompt-based continual learning methods, including L2P (Wang et al., 2022b), Dual-Prompt (Wang et al., 2022a), CodaPrompt (Smith et al., 2023), CPrompt (Gao et al., 2024), EvoPrompt (Kurniawan et al., 2024), PGP (Qiao et al., 2024), LGCL (Khan et al., 2023), ConvPrompt (Roy et al., 2024), CPG (Lu et al., 2025), and our direct predecessor, NSP^2 (Lu et al., 2024). The results are shown in table 1.

Table 1: Comparison with existing prompt-based methods using ImageNet-21K pretrained backbones. Results (%) are reported as mean ± standard deviation over three runs. The highest accuracies are in **bold**, and the second highest accuracies are underlined. Results marked with †, ‡, and § are reproduced by Qiao et al. (2024), Gao et al. (2024), and Lu et al. (2025), respectively, due to lack of official results.

Method	10-Split-CIFAR100		20-Split-CIFAR100		10-Split-ImageNet-R		10-Split-DomainNet	
	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)
L2P	$83.83_{\pm 0.04}$	$7.63_{\pm 0.30}$	$81.29_{\pm 0.43}$ †	$8.96_{\pm 0.38}$ †	$61.57_{\pm 0.66}$	$9.73_{\pm 0.47}$	$81.17_{\pm 0.83}$ ‡	$8.98_{\pm 1.25}$ ‡
DualPrompt	$86.51_{\pm 0.33}$	$5.16_{\pm 0.09}$	$82.98_{\pm 0.47}$ †	$8.20_{\pm 0.08}$ †	$68.13_{\pm 0.49}$	$4.68_{\pm 0.20}$	$81.70_{\pm 0.78}$ ‡	$8.04_{\pm 0.31}$ ‡
CodaPrompt	$86.25_{\pm 0.74}$	$1.67_{\pm 0.26}$	-	-	$75.45_{\pm 0.56}$	$1.64_{\pm 0.10}$	$80.04_{\pm 0.79}$ ‡	$10.16_{\pm 0.35}$ ‡
CPrompt	$87.82_{\pm 0.21}$	$5.06_{\pm 0.50}$	$83.97_{\pm 0.31}$ §	$6.85_{\pm 0.43}$ §	$77.14_{\pm 0.11}$	$5.97_{\pm 0.68}$	$82.97_{\pm 0.34}$	$7.45_{\pm 0.93}$
EvoPrompt	$87.97_{\pm 0.30}$	$2.60_{\pm 0.42}$	$84.64_{\pm 0.14}$	$3.98_{\pm 0.24}$	$76.83_{\pm 0.08}$	$2.78_{\pm 0.06}$	$79.50_{\pm 0.29}$ §	$3.81_{\pm 0.36}$ §
PGP	$86.92_{\pm 0.05}$	$5.35_{\pm 0.19}$	$83.74_{\pm 0.01}$	$7.91_{\pm 0.15}$	$69.34_{\pm 0.05}$	$4.53_{\pm 0.04}$	$80.41_{\pm 0.25}$ §	$8.39_{\pm 0.18}$ §
LGCL	$87.23_{\pm 0.21}$	$5.10_{\pm 0.15}$	-	-	$69.46_{\pm 0.04}$	$4.20_{\pm 0.06}$	-	-
ConvPrompt	$88.87_{\pm 0.33}$	$4.75_{\pm 0.15}$	$87.22_{\pm 0.42}$ §	$5.43_{\pm 0.29}$ §	$77.86_{\pm 0.25}$	$4.33_{\pm 0.24}$	$79.47_{\pm 0.35}$ §	$6.49_{\pm 0.43}$ §
CPG	$90.63_{\pm 0.44}$	$3.98_{\pm 0.65}$	$88.08_{\pm 0.77}$	$5.20_{\pm 0.64}$	$78.63_{\pm 0.52}$	$7.18_{\pm 0.62}$	$83.21_{\pm 0.67}$	$7.09_{\pm 0.82}$
NSP^2	$\underline{90.70}_{\pm 0.17}$	$4.20_{\pm 0.35}$	$\underline{88.59}_{\pm 0.46}$	$4.45_{\pm 0.60}$	$\underline{79.17}_{\pm 0.63}$	$5.06_{\pm 1.07}$	$\underline{84.17}_{\pm 0.54}$	$7.33_{\pm 0.94}$
CNSP (Ours)	**91.01** $_{\pm 0.10}$	$3.91_{\pm 0.47}$	**89.42**$_{\pm 0.58}$	$4.23_{\pm 0.51}$	**79.69**$_{\pm 0.32}$	$4.62_{\pm 0.66}$	**84.51**$_{\pm 0.41}$	$6.62_{\pm 0.26}$

Main Results and Analysis. Table 1 shows that CNSP consistently achieves the highest ACC across all benchmarks while maintaining competitive or lower forgetting, validating the effectiveness of our matrix-form analysis and right-side null-space projection. Specifically, on 10-Split CIFAR100, CNSP reaches 91.01±0.10, exceeding NSP^2 (90.70±0.17) by +0.31% and reducing forgetting (3.91 vs. 4.20). On 20-Split CIFAR100, it improves ACC by +0.83% (89.42 vs. 88.59) with slightly lower forgetting (4.23 vs. 4.45). On 10-Split ImageNet-R, it raises ACC to 79.69±0.32 (+0.52%) and reduces forgetting (4.62 vs. 5.06). On 10-Split DomainNet, it achieves 84.51±0.41 (+0.34%) and lowers forgetting (6.62 vs. 7.33). Task-wise curves in fig. 3 further reveal that CNSP maintains higher or comparable ACC throughout training, with consistently lower forgetting, particularly on cross-domain benchmark DomainNet where forgetting is more severe. For completeness, we also report task-wise curves for 20-Split CIFAR100 (fig. 4), and additional comparisons with fine-tuning (lower bound) and joint training (oracle upper bound) in Appendix D.5, which exhibit the same consistent advantage of CNSP over NSP^2.

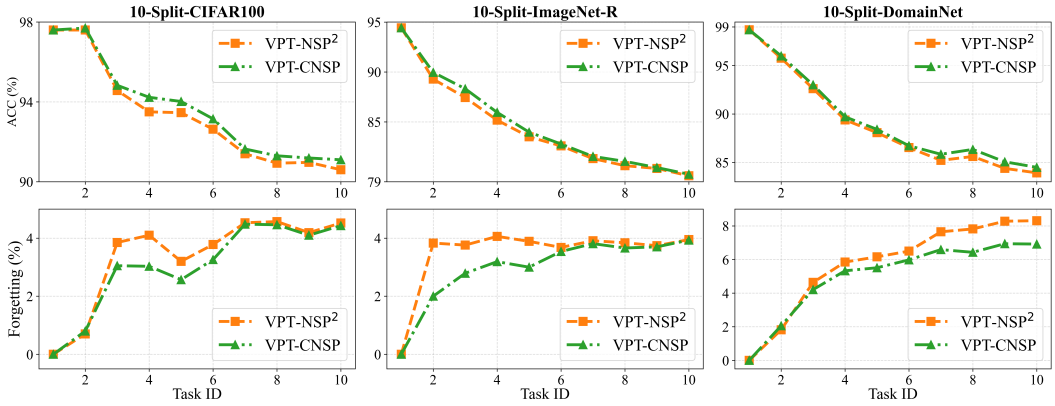

Figure 3: Results on three continual learning benchmarks. The first row shows the last average accuracy (ACC, %) on 10-Split-CIFAR100, 10-Split-ImageNet-R, and 10-Split-DomainNet, while the second row reports the corresponding forgetting (%), where lower values indicate better retention of prior knowledge. Compared methods are NSP^2 and our proposed CNSP.

Ablation Study. We assess the contribution of each component in CNSP (Table 2, Appendix D.5). Removing the variance-preservation loss ($\mathcal{L}_{\text{std}}$) yields moderate drops (e.g., CIFAR100 ACC 91.1→90.6; forgetting 4.43→5.17), showing its stabilizing effect. Eliminating the null-space projection ($\boldsymbol{B_R}$) causes severe degradation (ACC 85.0, forgetting 14.4), confirming its necessity for feature preservation. Removing the entire module ("Backbone only") further reduces ACC below 84% and raises forgetting above 16–25%. These results demonstrate that both $\mathcal{L}_{\text{std}}$ and $\boldsymbol{B_R}$ are indispensable for robust knowledge retention.

6 Conclusion

In this paper, we introduced CNSP, a principled framework that strengthens the theoretical foundation of prompt-based continual learning via refined null-space projection. Building on NSP^2, our approach addresses its key limitations with more rigorous treatments of LayerNorm and multi-head attention. As a result, CNSP offers a more stable and optimization-friendly formulation with end-to-end consistency guarantees. Empirically, CNSP consistently outperforms NSP^2 and other strong baselines across benchmarks, achieving higher accuracy with lower or comparable forgetting. Looking ahead, CNSP opens promising directions such as extending to multi-modal continual learning, adaptive prompt allocation, and tighter theoretical guarantees for nonlinear components (e.g., softmax). Further discussions and limitations are provided in Appendix E.

Reproducibility Statement

Our experiments are conducted on publicly available datasets (e.g., CIFAR-100, ImageNet). We report the details of our training setup, including hyperparameters (batch size, learning rate, optimizer), number of epochs, and evaluation metrics, in Appendix D. To ensure reproducibility, we will release the complete source code and training/evaluation scripts upon acceptance of the paper.

References

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 113–123, 2019.

Danruo Deng, Guangyong Chen, Jianye Hao, Qiong Wang, and Pheng-Ann Heng. Flattening sharpness for dynamic gradient projection memory benefits continual learning. *Advances in Neural Information Processing Systems*, 34:18710–18721, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Zhanxin Gao, Jun Cen, and Xiaobin Chang. Consistent prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28463–28473, 2024.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8340–8349, 2021.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision*, pp. 709–727. Springer, 2022.

Muhammad Gul Zain Ali Khan, Muhammad Ferjad Naeem, Luc Van Gool, Didier Stricker, Federico Tombari, and Muhammad Zeshan Afzal. Introducing language guidance in prompt-based continual learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11463–11473, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32, 2019.

Muhammad Rifki Kurniawan, Xiang Song, Zhiheng Ma, Yuhang He, Yihong Gong, Yang Qi, and Xing Wei. Evolving parameterized prompt memory for continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13301–13309, 2024.

Guoliang Lin, Hanlu Chu, and Hanjiang Lai. Towards better plasticity-stability trade-off in incremental learning: A simple linear connector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 89–98, 2022a.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Trgp: Trust region gradient projection for continual learning. *arXiv preprint arXiv:2202.02931*, 2022b.

Yue Lu, Shizhou Zhang, De Cheng, Yinghui Xing, Nannan Wang, Peng Wang, and Yanning Zhang. Visual prompt tuning in null space for continual learning. *Advances in neural information processing systems*, 37:7878–7901, 2024.

Yue Lu, Shizhou Zhang, De Cheng, Guoqiang Liang, Yinghui Xing, Nannan Wang, and Yanning Zhang. Training consistent mixture-of-experts-based prompt generator for continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19152–19160, 2025.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019.

Jingyang Qiao, zhizhong zhang, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, and Yuan Xie. Prompt gradient projection for continual learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=EH2O3h7sBI.

Anurag Roy, Riddhiman Moulick, Vinay K Verma, Saptarshi Ghosh, and Abir Das. Convolutional prompting meets language models for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 23616–23626, 2024.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *CoRR*, abs/2103.09762, 2021. URL https://arxiv.org/abs/2103.09762.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11909–11919, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Cheng Wang. Calibration in deep learning: A survey of the state-of-the-art. *arXiv preprint arXiv:2308.01222*, 2023.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 184–193, 2021.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European conference on computer vision*, pp. 631–648, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 139–149, 2022b.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

APPENDIX

A PROOFS OF LEMMAS AND PROPOSITIONS

Lemma 1. *Consider the computation of LayerNorm* $\mathrm{LN}(\cdot)$*. Let* $\boldsymbol{A} = [\boldsymbol{A}^{(1)}; \boldsymbol{A}^{(2)}]$ *with* $\boldsymbol{A}^{(1)} \in \mathbb{R}^{n\times d}$ *and* $\boldsymbol{A}^{(1)} \in \mathbb{R}^{m\times d}$*. Then* $\mathrm{LN}(\boldsymbol{A}) = \left[\mathrm{LN}\left(\boldsymbol{A}^{(1)}\right); \mathrm{LN}\left(\boldsymbol{A}^{(2)}\right)\right]$.

Proof. Expanding the LayerNorm operations gives

$$\mathrm{LN}(\boldsymbol{A}) = \frac{\boldsymbol{A} - \boldsymbol{\mu}_{\boldsymbol{A}}}{\boldsymbol{\sigma}_{\boldsymbol{A}}} \cdot \boldsymbol{\gamma} + \boldsymbol{\beta}, \quad \boldsymbol{\mu}_{\boldsymbol{A}}, \boldsymbol{\sigma}_{\boldsymbol{A}} \in \mathbb{R}^{n+m}, \quad \boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{d}, \tag{31}$$

where broadcasting is applied to match the dimensions $(n+m)\times d$. By matrix reformulation of the broadcast, let

$$\boldsymbol{C} = \boldsymbol{I}_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top \in \mathbb{R}^{d\times d}, \quad \boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}}} = \mathrm{diag}\left(\boldsymbol{\sigma}_{\boldsymbol{A}}\right) \in \mathbb{R}^{(n+m)\times(n+m)}, \quad \boldsymbol{\Gamma} = \mathrm{diag}\left(\boldsymbol{\gamma}\right) \in \mathbb{R}^{d\times d}. \tag{32}$$

Then

$$\mathrm{LN}(\boldsymbol{A}) = \boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}}}^{-1}\boldsymbol{A}\boldsymbol{C}\boldsymbol{\Gamma} + \mathbf{1}_{n+m}\boldsymbol{\beta}^\top. \tag{33}$$

In practice, the standard deviation of the i-th row is computed as

$$\sqrt{\frac{1}{d}\mathrm{Var}\left(\boldsymbol{A}[i]\right) + \epsilon}, \tag{34}$$

where $\mathrm{Var}(\cdot)$ is the operation to compute variance, $\epsilon > 0$ ensures non-degeneracy. Hence, $\boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}}}$ is invertible. Since $\boldsymbol{\sigma}_{\boldsymbol{A}}$ corresponds row-wise to $\boldsymbol{A}$, we may write

$$\boldsymbol{\sigma}_{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{\sigma}_{\boldsymbol{A}^{(1)}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\sigma}_{\boldsymbol{A}^{(2)}} \end{bmatrix}, \quad \boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}}} = \begin{bmatrix} \boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}^{(1)}}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}^{(2)}}} \end{bmatrix}. \tag{35}$$

Therefore,

$$\mathrm{LN}(\boldsymbol{A}) = \begin{bmatrix} \boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}^{(1)}}}\boldsymbol{A}^{(1)}\boldsymbol{C}\boldsymbol{\Gamma} + \mathbf{1}_n\boldsymbol{\beta}^\top \\ \boldsymbol{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}^{(2)}}}\boldsymbol{A}^{(2)}\boldsymbol{C}\boldsymbol{\Gamma} + \mathbf{1}_m\boldsymbol{\beta}^\top \end{bmatrix} = \begin{bmatrix} \mathrm{LN}\left(\boldsymbol{A}^{(1)}\right) \\ \mathrm{LN}\left(\boldsymbol{A}^{(2)}\right) \end{bmatrix}. \tag{36}$$

□

Lemma 2. *For* $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ *with left null space* $\mathcal{N}_L(\boldsymbol{A}) = \{\boldsymbol{x}\boldsymbol{A} = \mathbf{0} \mid \boldsymbol{x}^\top \in \mathbb{R}^n\}$*, it holds that* $\mathcal{N}_L(\boldsymbol{A}) = \mathcal{N}_L\left(\boldsymbol{A}\boldsymbol{A}^\top\right)$.

Proof. $\forall \boldsymbol{x} \in \mathcal{N}_L(\boldsymbol{A})$, we have $\boldsymbol{x}\boldsymbol{A} = \mathbf{0}$, which implies $\boldsymbol{x}\boldsymbol{A}\boldsymbol{A}^\top = \mathbf{0}$, i.e., $\boldsymbol{x} \in \mathcal{N}_L\left(\boldsymbol{A}\boldsymbol{A}^\top\right)$. Conversely, $\forall \boldsymbol{x} \in \mathcal{N}_L\left(\boldsymbol{A}\boldsymbol{A}^\top\right)$, we have $\boldsymbol{x}\boldsymbol{A}\boldsymbol{A}^\top = \mathbf{0}$. Then, $0 = \boldsymbol{x}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{x}^\top = \|\boldsymbol{A}^\top\boldsymbol{x}^\top\|^2$, which implies $\boldsymbol{x}\boldsymbol{A} = \mathbf{0}$, i.e., $\boldsymbol{x} \in \mathcal{N}_L(\boldsymbol{A})$. □

Lemma 3. *Consider the operation* $\mathrm{MLP}(\cdot)$ *in a standard ViT MLP (Dosovitskiy et al., 2020), containing two layers with a GELU non-linearity, defined as a linear projection followed by a GELU activation and another linear projection. For a block matrix* $\boldsymbol{A} = \left[\boldsymbol{A}^{(1)}; \boldsymbol{A}^{(2)}\right]$*, where* $\boldsymbol{A}^{(1)} \in \mathbb{R}^{n\times d}$ *and* $\boldsymbol{A}^{(2)} \in \mathbb{R}^{m\times d}$*, we have*

$$\mathrm{MLP}(\boldsymbol{A}) = \begin{bmatrix} \mathrm{MLP}\left(\boldsymbol{A}^{(1)}\right) \\ \mathrm{MLP}\left(\boldsymbol{A}^{(2)}\right) \end{bmatrix} \tag{37}$$

Proof. By definition,

$$\mathrm{MLP}(\boldsymbol{A}) = \phi\left(\boldsymbol{A}\boldsymbol{W}_1 + \mathbf{1}_{n+m}\boldsymbol{b}_1^\top\right)\boldsymbol{W}_2 + \mathbf{1}_{n+m}\boldsymbol{b}_2^\top, \tag{38}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d\times d_h}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_h\times d}$ are weight matrices of two linear projections respectively, $\boldsymbol{b}_1 \in \mathbb{R}^{d_h}$, $\boldsymbol{b}_2 \in \mathbb{R}^d$ are bias vectors, and ϕ denotes the element-wise GELU.

Since linear transformations, element-wise nonlinearities, and bias additions act independently on each row, the block structure is preserved. Explicitly,

$$\phi\left(\boldsymbol{A}\boldsymbol{W}_1 + \mathbf{1}_{n+m}\boldsymbol{b}_1^\top\right) = \phi\left(\begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \end{bmatrix}\boldsymbol{W}_1 + \begin{bmatrix} \mathbf{1}_n \\ \mathbf{1}_m \end{bmatrix}\boldsymbol{\beta}_1^\top\right) = \begin{bmatrix} \phi\left(\boldsymbol{A}^{(1)}\boldsymbol{W}_1 + \mathbf{1}_n\boldsymbol{b}_1^\top\right) \\ \phi\left(\boldsymbol{A}^{(2)}\boldsymbol{W}_1 + \mathbf{1}_m\boldsymbol{b}_1^\top\right) \end{bmatrix}. \tag{39}$$

Similarly, multiplying by $\boldsymbol{W}_2$ and adding bias yields

$$\phi\left(\boldsymbol{A}\boldsymbol{W}_1 + \mathbf{1}_{n+m}\boldsymbol{b}_1^\top\right)\boldsymbol{W}_2 + \mathbf{1}_{n+m}\boldsymbol{b}_2^\top = \begin{bmatrix} \phi\left(\boldsymbol{A}^{(1)}\boldsymbol{W}_1 + \mathbf{1}_n\boldsymbol{b}_1^\top\right)\boldsymbol{W}_2 + \mathbf{1}_n\boldsymbol{b}_2^\top \\ \phi\left(\boldsymbol{A}^{(2)}\boldsymbol{W}_1 + \mathbf{1}_m\boldsymbol{b}_1^\top\right)\boldsymbol{W}_2 + \mathbf{1}_m\boldsymbol{b}_2^\top \end{bmatrix}, \tag{40}$$

which implies that

$$\mathrm{MLP}\left(\boldsymbol{A}\right) = \begin{bmatrix} \mathrm{MLP}\left(\boldsymbol{A}^{(1)}\right) \\ \mathrm{MLP}\left(\boldsymbol{A}^{(2)}\right) \end{bmatrix} \tag{41}$$

□

Proposition 1. *Consider two tasks $\mathcal{T}_t$ and $\mathcal{T}_{t+1}$. If the image-token part of attention outputs remains unchanged for all blocks in VPT, i.e.,*

$$\boldsymbol{O}^{(l)}_{\boldsymbol{Z}_{t,t+1}}[:N+1] = \boldsymbol{O}^{(l)}_{\boldsymbol{Z}_{t,t}}[:N+1], \quad l = 1, 2, \ldots, L, \tag{6}$$

then the image-token output of the final block is preserved:

$$\boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t+1}}[:N+1] = \boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t}}[:N+1], \tag{7}$$

*which we refer to as **feature preservation**. Here $\boldsymbol{O}^{(l)}_{\boldsymbol{Z}_{t,t+1}}$, $\boldsymbol{O}^{(l)}_{\boldsymbol{Z}_{t,t}}$, $\boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t+1}}$, and $\boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t}}$ are computed by eq.* (1)*. In particular, the final [CLS] token is preserved:*

$$\boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t+1}}[:1] = \boldsymbol{E}^{(L)}_{\boldsymbol{Z}_{t,t}}[:1]. \tag{8}$$

Proof. Without loss of generality, we first omit the task indices and focus on the l-th transformer block. As shown in fig. 1, within a block, $\boldsymbol{Z}^{(l)}$ is added to the output $\boldsymbol{O}^{(l)}_{\boldsymbol{Z}}$ via a residual connection, followed by LayerNorm and an MLP with an additional residual connection:

$$\boldsymbol{B}^{(l)}_{\boldsymbol{Z}} = \boldsymbol{Z}^{(l)} + \boldsymbol{O}^{(l)}_{\boldsymbol{Z}}, \quad \boldsymbol{E}^{(l)}_{\boldsymbol{Z}} = \boldsymbol{B}^{(l)}_{\boldsymbol{Z}} + \mathrm{MLP}\left(\mathrm{LN}\left(\boldsymbol{B}^{(l)}_{\boldsymbol{Z}}\right)\right). \tag{42}$$

Since

$$\boldsymbol{E}^{(l)}_{\boldsymbol{Z}}[:N+1] = \boldsymbol{B}^{(l)}_{\boldsymbol{Z}}[:N+1] + \mathrm{MLP}\left(\mathrm{LN}\left(\boldsymbol{B}^{(l)}_{\boldsymbol{Z}}\right)\right)[:N+1], \tag{43}$$

by lemmas 1 and 3, we obtain

$$\mathrm{MLP}\left(\mathrm{LN}\left(\boldsymbol{B}^{(l)}_{\boldsymbol{Z}}\right)\right)[:N+1] = \mathrm{MLP}\left(\mathrm{LN}\left(\boldsymbol{B}^{(l)}_{\boldsymbol{Z}}[:N+1]\right)\right). \tag{44}$$

Moreover,

$$\boldsymbol{B}^{(l)}_{\boldsymbol{Z}}[:N+1] = \boldsymbol{Z}^{(l)}[:N+1] + \boldsymbol{O}^{(l)}_{\boldsymbol{Z}}[:N+1] = \boldsymbol{X}^{(l)} + \boldsymbol{O}^{(l)}_{\boldsymbol{Z}}[:N+1], \tag{45}$$

thus,

$$\boldsymbol{E}^{(l)}_{\boldsymbol{Z}}[:N+1] = \boldsymbol{X}^{(l)} + \boldsymbol{O}^{(l)}_{\boldsymbol{Z}}[:N+1] + \mathrm{MLP}\left(\mathrm{LN}\left(\boldsymbol{X}^{(l)} + \boldsymbol{O}^{(l)}_{\boldsymbol{Z}}[:N+1]\right)\right). \tag{46}$$

In VPT, before passing to the next block , the prompt embeddings are discarded and replaced by new learnable prompts:

$$\boldsymbol{X}^{(l+1)} = \boldsymbol{E}^{(l)}_{\boldsymbol{Z}}[:N+1], \quad \boldsymbol{Z}^{(l+1)} = [\boldsymbol{X}^{(l+1)}; \boldsymbol{P}^{(l+1)}]. \tag{47}$$

Now consider two tasks $\mathcal{T}_t$ and $\mathcal{T}_{t+1}$. We prove the statement by induction on l:

(i) When $l = 1$, by assumption and setting, we have

$$\boldsymbol{O}^{(1)}_{\boldsymbol{Z}_{t,t+1}}[:N+1] = \boldsymbol{O}^{(1)}_{\boldsymbol{Z}_{t,t}}[:N+1], \quad \boldsymbol{X}^{(1)}_{t,t+1} = \boldsymbol{X}^{(1)}_{t,t} = \boldsymbol{X}_t, \tag{48}$$

where $\boldsymbol{X}_t$ is the input image tokens of VPT from task $\mathcal{T}_t$ dataset. According to eq. (46), it follows that

$$\boldsymbol{E}^{(1)}_{\boldsymbol{Z}_{t,t+1}}[:N+1] = \boldsymbol{E}^{(l)}_{\boldsymbol{Z}_{t,t}}[:N+1], \quad \boldsymbol{X}^{(2)}_{t,t+1} = \boldsymbol{X}^{(2)}_{t,t}. \tag{49}$$

(ii) Suppose that for some $l = k(< L)$,

$$\boldsymbol{E}^{(k)}_{\boldsymbol{Z}_{t,t+1}}[:N+1] = \boldsymbol{E}^{(k)}_{\boldsymbol{Z}_{t,t}}[:N+1], \tag{50}$$

which implies

$$\boldsymbol{X}_{t,t+1}^{(k+1)} = \boldsymbol{X}_{t,t}^{(k+1)}. \tag{51}$$

For $l = k+1$, we know

$$\boldsymbol{O}_{\boldsymbol{Z}_{t,t+1}}^{(k+1)}[:N+1] = \boldsymbol{O}_{\boldsymbol{Z}_{t,t}}^{(k+1)}[:N+1]. \tag{52}$$

By eqs. (46) and (51), it follows that

$$\boldsymbol{E}_{\boldsymbol{Z}_{t,t+1}}^{(k+1)}[:N+1] = \boldsymbol{E}_{\boldsymbol{Z}_{t,t}}^{(k+1)}[:N+1]. \tag{53}$$

(iii) By induction, for all $l = 1, 2, \ldots, L$, we have

$$\boldsymbol{E}_{\boldsymbol{Z}_{t,t+1}}^{(l)}[:N+1] = \boldsymbol{E}_{\boldsymbol{Z}_{t,t}}^{(l)}[:N+1]. \tag{54}$$

Especially, the final [CLS] token fed into the classification head is unchanged, i.e.,

$$\boldsymbol{E}_{\boldsymbol{Z}_{t,t+1}}^{(L)}[:1] = \boldsymbol{E}_{\boldsymbol{Z}_{t,t}}^{(L)}[:1]. \tag{55}$$

□

B Consistent Null-Space Projection in Practice

In the main text (section 4.3), we derived sufficient conditions for feature preservation, requiring that each prompt update $\Delta\boldsymbol{P}$ lies in the left null space of the constraint matrix $\boldsymbol{R}$. Theoretically, this is enforced by projecting $\Delta\boldsymbol{P}$ onto the null space of $\boldsymbol{R}\boldsymbol{R}^\top$ (see eq. (30)). However, in practice, due to finite-precision arithmetic, exact zero singular values rarely occur, making it nontrivial to determine the effective nullity. To address this issue, following Lu et al. (2024), we estimate the nullity of $\boldsymbol{R}\boldsymbol{R}^\top$ using an *adaptive thresholding criterion* based on the inflection point of the singular value spectrum. Specifically, we compute the discrete second derivative of sorted singular values and use its maximizer to determine the cut-off between the null and non-null components. Formally, let λ_j denote the j-th singular value of $\boldsymbol{R}\boldsymbol{R}^\top$ (sorted in descending order). The adaptive estimate of the nullity is given by

$$\text{Nullity}\left(\boldsymbol{R}\boldsymbol{R}^\top\right) = D - \arg\max_j \left\{\lambda_{j+1} - 2\lambda_j + \lambda_{j-1}\right\}_{j=2}^{D-1}, \tag{56}$$

where the criterion detects the point of maximum curvature change in the spectrum. Empirically, this approach has proven more robust than fixed thresholds in VPT Lu et al. (2024), providing a numerically stable approximation of the null space dimension.

With the estimated null space, we then construct an orthogonal basis $\boldsymbol{U}_0$ for $\mathcal{N}_L(\boldsymbol{R}\boldsymbol{R}^\top)$ and form the projection matrix $\boldsymbol{B}_{\boldsymbol{R}} = \boldsymbol{U}_0\boldsymbol{U}_0^\top$. During training, the raw prompt update $\Delta\boldsymbol{P}_{\text{raw}}$ from back-propagation is consistently projected onto this null space, ensuring that the sufficient conditions for feature preservation are satisfied at every optimization step. This procedure constitutes our *Consistent Null-Space Projection (CNSP)* mechanism.

The detailed training algorithm is summarized in Algorithm 1, and the computation of the null-space projection matrix is given in Algorithm 2.

C Comparison and Discussion with NSP^2

Our method builds on the theoretical framework of NSP^2, but introduces critical improvements in terms of multi-head rigor, LayerNorm treatment, and practical feasibility.

Multi-head Derivation. In NSP^2, the sufficient conditions were first established under the single-head setting, and then directly extended to multi-head attention by concatenation. While this extension is formally valid, it does not come with a rigorous derivation and leaves open the possibility of inter-head coupling. By contrast, our derivation explicitly unfolds the forward pass of multi-head self-attention, proving that the feature-preservation sufficient conditions can be satisfied per head. This provides a mathematically sound basis for the transition from single-head to multi-head analysis.

Algorithm 1 CNSP for Visual Prompt Tuning

Inputs: Datasets $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{|\mathcal{T}_t|}$ for task $\mathcal{T}_t \in \{\mathcal{T}_1, \mathcal{T}_2, \dots\}$; ViT model $f_{\text{ViT}}(\cdot \mid \boldsymbol{P}_t)$ with the prompts $\boldsymbol{P}_t$ to be optimized (the updates of classification head is omitted for simplicity) ; the gram matrix of constraints $\boldsymbol{G}$; the projection matrix $\boldsymbol{B}_{\boldsymbol{R}}$
Outputs: Optimized prompts $\boldsymbol{P}_t$

Initialization: Randomly initialize $\boldsymbol{P}_t$; set $\boldsymbol{G} = \boldsymbol{0}$, $\boldsymbol{B}_{\boldsymbol{R}} = \boldsymbol{I}$
for task $\mathcal{T}_t \in \{\mathcal{T}_1, \mathcal{T}_2, \dots\}$ **do**
  repeat
    Sample a mini-batch $\boldsymbol{x}_t, y_t \sim \mathcal{D}_t$
    $\hat{y}_t \leftarrow f_{\text{ViT}}(\boldsymbol{x}_t \mid \boldsymbol{P}_t)$
    $\mathcal{L}_{\text{total}} \leftarrow \text{CrossEntropy}(\hat{y}_t, y_t)$ ▷ the classification loss
    if $t > 1$ **then**
      Compute $\mathcal{L}_{\text{std}}$ by eq. (28) ▷ loss of prompts standard deviation
      $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{total}} + \mathcal{L}_{\text{std}}$
    end if
    Get raw prompts update $\Delta \boldsymbol{P}_{\text{raw}}$ from optimizer using $\mathcal{L}_{\text{total}}$
    if $t > 1$ **then**
      $\Delta \boldsymbol{P} \leftarrow \Delta \boldsymbol{P}_{\text{raw}} \boldsymbol{B}_{\boldsymbol{R}}$ ▷ consistent null-space projection, eq. (30)
    else
      $\Delta \boldsymbol{P} \leftarrow \boldsymbol{P}_{\text{raw}}$
    end if
    $\boldsymbol{P}_t \leftarrow \boldsymbol{P}_t - lr \times \Delta \boldsymbol{P}$
  until convergence
  if $t = 1$ **then**
    $\boldsymbol{R}_v \leftarrow \boldsymbol{C}\boldsymbol{\Gamma}\boldsymbol{W}_v$ ▷ cache the second constraint matrix
  end if
  for $\boldsymbol{x}_t^{(i)} \in \mathcal{D}_t$ **do**
    $\boldsymbol{R}_k \leftarrow \boldsymbol{C}\boldsymbol{\Gamma}\boldsymbol{Q}_{\boldsymbol{X}_t^{(i)}}\boldsymbol{W}_k^\top$ ▷ by forward propagation $f_{\text{ViT}}(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{P}_t)$
    $\boldsymbol{G} \leftarrow \boldsymbol{G} + \boldsymbol{R}_k \boldsymbol{R}_k^\top + \boldsymbol{R}_v \boldsymbol{R}_v^\top$
  end for
  $\boldsymbol{B}_{\boldsymbol{R}} \leftarrow \text{ComputingNullSpaceProjection}(\boldsymbol{G})$ ▷ using algorithm 2
end for

Algorithm 2 Computing Null-Space Projection Matrix

Inputs: Gram matrix $\boldsymbol{G}$
Outputs: Null-space projection matrix $\boldsymbol{B}_{\boldsymbol{R}}$

$\boldsymbol{U}, \boldsymbol{\Sigma}, \_ \leftarrow \text{SVD}(\boldsymbol{G})$ ▷ sorted by the singular values in descending order
Computing the nullity $N_{\boldsymbol{G}}$ of $\boldsymbol{G}$ by eq. (56)
$\boldsymbol{U}_0 \leftarrow \boldsymbol{U}[D - N_{\boldsymbol{G}} : D]$ ▷ get the left singular vectors of zero singular values
$\boldsymbol{B}_{\boldsymbol{R}} \leftarrow \frac{\boldsymbol{U}_0 \boldsymbol{U}_0^\top}{\|\boldsymbol{U}_0 \boldsymbol{U}_0^\top\|_F}$ ▷ improve numerical stability by its Frobenius norm

LayerNrom Treatment. NSP^2 handled broadcast operations in LayerNorm by simply canceling them as scalars in equations through derivation, ignoring the centering and scaling mechanism inherent in LayerNorm. This leads to deviations in its conclusions. By contrast, we fully matrixize all broadcast operations in our derivation. Specifically, we explicitly represent the centering matrix $\boldsymbol{C}$, the row-wise scaling matrix $\boldsymbol{D}_{\boldsymbol{\sigma}}$ (standard deviation normalization), and the affine scaling matrix $\boldsymbol{\Gamma}$. This guarantees rigor in the derivation and allows us to assume only variance invariance of prompt embeddings across tasks (i.e., eq. (25)), rather than both variance and mean invariance as required in NSP^2, thereby relaxing the distributional restrictions.

Implementation of Sufficient Conditions. The difference in LayerNorm treatment directly afffects the final form of the sufficient conditions. Under the single-head formulation, NSP^2 gives rise to the

following constraints:

$$\begin{cases} \boldsymbol{Q}_{\boldsymbol{X}_t} \boldsymbol{W}_k{}^\top \Delta \boldsymbol{P}^\top = \mathbf{0} \\ \boldsymbol{S}_{\boldsymbol{P}_t} \Delta \boldsymbol{P} \boldsymbol{W}_v = \mathbf{0} \end{cases}, \tag{57}$$

while our rigorous derivation leads to the per-head feature-preservation sufficent conditions

$$\begin{cases} \boldsymbol{Q}^{(h)}_{\boldsymbol{X}_t} \boldsymbol{W}^{(h)}_k{}^\top \boldsymbol{\Gamma} \boldsymbol{C} \Delta \boldsymbol{P}^\top = \mathbf{0} \\ \boldsymbol{S}^{(h)}_{\boldsymbol{X}_t \boldsymbol{P}_t} \boldsymbol{D}_{\boldsymbol{\sigma}}^{-1} \Delta \boldsymbol{P} \boldsymbol{C} \boldsymbol{\Gamma} \boldsymbol{W}^{(h)}_v = \mathbf{0} \end{cases}. \tag{58}$$

The key difference arises from LayerNorm: by omitting broadcast operations, NSP^2 makes its second condition appear solvable via left-sided nullification. However, as discussed in section 4.3, left nullification inevitably couples $\boldsymbol{D}_{\boldsymbol{\sigma}}$ (which depends on current prompt statistics) with the solution side. In contrast, our formulation adopts right-sided nullification, which decouples this dependency and thereby ensures both feasibility and stability in practice.

Classification Head Preservation. NSP^2 focuses solely on attention-layer preservation conditions (corresponding to feature preservation in our work), without considering the classification head. Consequently, the interpretability of its results remain limited to intermediate layers. By contrast, our framework extends beyond feature preservation to also formalize head preservation, yielding an end-to-end solution that maintains consistency from representation to decision. This extension ensures both theoretical completeness and practical implementability.

Summary. In summary, our method not only address the theoretical gaps of NSP^2, but also relaxes distributional assumptions and provide a more stable, implementable form of the task-performance preservation constraints.

D Experiments

D.1 Datasets and Benchmarks

To thoroughly evaluate the proposed CNSP method in prompt-based continual learning scenarios, we conduct experiments on four widely used benchmark datasets:

- **10/20-Split CIFAR-100** (Krizhevsky et al., 2009): CIFAR-100 consists of 60,000 32×32 color images across 100 classes, with 500 training images and 100 test images per class. Following common practice, it is randomly partitioned into 10 or 20 tasks, where each task contains 10 or 5 classes, respectively.
- **10-Split ImageNet-R** (Hendrycks et al., 2021): ImageNet-R contains 30,000 images from 200 ImageNet classes, covering diverse styles such as art, cartoons, and sketches. It is randomly divided into 10 tasks, each consisting of 20 classes.
- **10-Split DomainNet** (Peng et al., 2019): DomainNet is a cross-domain dataset covering 345 everyday object classes across six domains (clipart, real, sketch, infograph, painting, and quickdraw). Since the number of images per class varies significantly, we follow the exsiting setting (Gao et al., 2024; Lu et al., 2024) by selecting the 200 largest classes and randomly splitting them into 10 tasks, each containing 20 classes with samples drawn from multiple domains.

D.2 Metrics

In continual learning, evaluation is typically based on the accuracy matrix $\boldsymbol{A} \in \mathbb{R}^{T \times T}$, where T is the total number of tasks and the entry $A_{i,j}$ denotes the accuracy on task j after training on task i. We report two standard metrics: Last Average Accuracy and Last Average Forgetting.

- **Last Average Accuracy (ACC)** measures the overall performance on all tasks after completing the training sequence:

$$\text{ACC} = \frac{1}{T} \sum_{j=1}^{T} A_{T,j},$$

- **Last Average Forgetting (Forgetting)** quantifies the average performance drop on past tasks after learning subsequent ones:

$$\text{Forgetting} = \frac{1}{T-1}\sum_{j=1}^{T-1}\max_{i\in\{1,\ldots,T-1\}}(A_{i,j} - A_{T,j}).$$

D.3 Implementation Details

We follow the class-incremental learning protocol, where task classes are disjoint and the task identity is unknown at inference. All experiments are conducted with ViT-B/16 (Dosovitskiy et al., 2020) pretrained on ImageNet-21K as the backbone. Each Transformer layer is augmented with four learnable prompts,yielding 48 prompts across the 12 layers. The prompt embeddings are shared across tasks and initialized from a uniform distribution in $[-1, 1]$. Input images are resized to 224×224, and data augmentation is applied using AutoAugment (Cubuk et al., 2019).

Models are trained with Adam (Kingma & Ba, 2015) (learning rate of 0.01, a batch size of 256, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 5×10^{-5}). Each task is trained for 10 epochs on all benchmarks, which is sufficient for convergency (as shown in Appendi D.4). NSP^2 is reproduced from its official codebase and hyperparameters, with the sole modification of training for 10 epochs per task instead of the originally reported 100. Empirical evidence in Appendix D.4 confirms fully convergence, disentangling forgetting from underfitting. The training objective combines cross-entropy loss with a feature standard-deviation alignment loss for prompt embeddings (see eq. (28)), using an alignment weight $\lambda = 1.0$. The temperature parameter in cross-entropy is tuned via cross-validation and set to 28, 25, 30, and 30 for 10-Split CIFAR-100, 20-Split CIFAR-100, 10-Split ImageNet-R, and 10-Split DomainNet, respectively.

Our method is implemented in Python 3.11.5 using PyTorch 2.1.0 (Paszke et al., 2019). Experiments are performed on a server with 256 GB RAM and four NVIDIA GeForce RTX 4090 GPUs. To ensure stability and reproducibility, each experiment is repeated three times with different random seeds, and each run takes complete within three hours.

D.4 Convergence Guarantee

To justify the training protocol adopted in our experiments, we report the training accuracy and loss curves of both NSP^2 and CNSP across tasks on four benchmarks (10-Split-CIFAR100, 20-Split-CIFAR100, 10-Split-DomainNet, and 10-Split-ImageNet-R). As shown in figs. 6 to 9, all curves exhibit rapid convergence within the first 5 epochs: training accuracy quickly increases and stabilizes, while training loss consistently decreases and plateaus. By epoch 10, both methods have fully converged across tasks and datasets, as further confirmed by the shaded regions indicating the last four epochs and the black curve representing the average accuracy/loss across all tasks at each epoch. These results provide empirical evidence that training each task for 10 epochs is sufficient to ensure convergence, without sacrificing performance or stability.

D.5 Supplementary Results

Table 2 provides both overall comparison and ablation of CNSP. The Backbone only baseline (equivalent to naïve fine-tuning without any continual learning strategy) suffers from severe catastrophic forgetting, representing the natural lower bound. At the opposite extreme, Joint training with access to all task data simultaneously provides an oracle upper bound. Between these two, NSP^2 delivers strong results, yet CNSP consistently improves upon it across all four benchmarks, achieving higher accuracy and lower forgetting. This shows that CNSP substantially narrows the gap toward the oracle performance while significantly surpassing naïve fine-tuning.

The ablation results reveal the contribution of individual components. Removing the variance preservation loss ($\mathcal{L}_{\text{std}}$) leads to only moderate drops compared to full CNSP, while already yielding performance that clearly surpasses NSP^2. This indicates that the core strength of CNSP comes from the null-space formulation itself, and that $\mathcal{L}_{\text{std}}$ mainly plays a stabilizing role. In contrast, removing the right-side null-space projection ($\boldsymbol{B_R}$) causes severe degradation in both accuracy and forgetting across all benchmarks, confirming it as an indispensable mechanism for effective feature preser-

vation. Finally, the Backbone only baseline performs worst overall, underscoring the substantial improvements CNSP provides beyond a frozen backbone.

Table 2: Overall comparison and ablation study of CNSP on four benchmarks by removing the variance preservation loss ($\mathcal{L}_{\text{std}}$), the right-side null-space projection ($\boldsymbol{B_R}$), and the entire CNSP module ("Backbone only"). Joint indicates training with access to all task data simultaneously (oracle upper bound).

Method	**10-Split-CIFAR100**		**20-Split-CIFAR100**		**10-Split-ImageNet-R**		**10-Split-DomainNet**	
	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)
Joint (oracle)	93.59	0	93.59	0	84.62	0	89.52	0
NSP^2	90.6	4.52	88.12	5.11	79.6	3.95	83.92	8.31
CNSP(Ours)	91.1	4.43	88.76	4.79	79.75	3.93	84.49	6.92
w/o $\mathcal{L}_{\text{std}}$	90.58	5.17	88.22	6.13	79.72	4.3	84.29	7.58
w/o $\boldsymbol{B_R}$	85.01	14.42	84.8	14.84	73.98	17.62	78.29	19.85
Backbone only	83.72	16.17	83.88	15.55	73.33	18.33	74.3	24.94

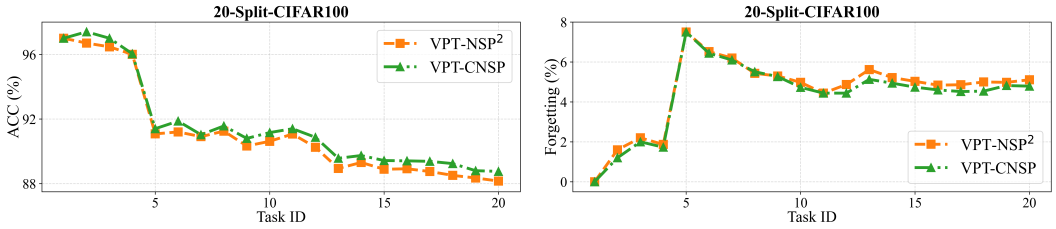

Figure 4: Results on 20-Split-CIFAR100. The first subplot shows the last average accuracy (ACC, %) and the second reports the corresponding forgetting (%), where lower values indicate better retention of prior knowledge. Compared methods are VPT-NSP^2 and our proposed VPT-CNSP.

E Discussion and Limitations

E.1 Discussion on Head Preservation

In our framework, head preservation (eq. (9)) is guaranteed by design through the use of task-specific classification heads. Figure 5 illustrates the training and inference procedures of task-specific heads. During training, only the head corresponding to the current task is updated, while all previously learned heads remain frozen. As a result, their mappings from the preserved [CLS] token remain unchanged across tasks, directly satisfying proposition 2. At inference, we concatenate the logits of all task-specific heads and perform a global prediction by selecting the class with the highest confidence.

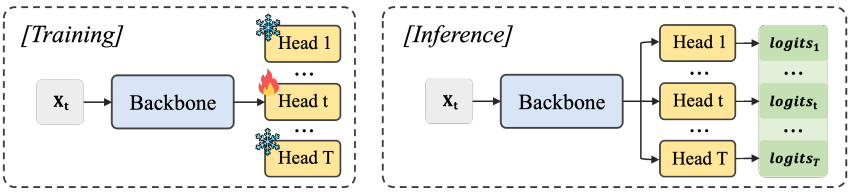

Figure 5: Training and inference of task-specific heads. During training, only the current task head is updated while others remain frozen. During inference, all heads run in parallel and their logits are concatenated, with the final prediction given by the most confident class, eliminating the need for task identity.

This design is grounded in the fact that the softmax outputs of a neural classifier are commonly interpreted as conditional probability estimates over class labels (Guo et al., 2017; Kull et al., 2019; Wang, 2023). Concatenating logits across all heads is therefore approximately equivalent to forming a unified probability distribution over the joint label space, upon which a single global decision can be made by $\arg\max$ (Guo et al., 2017; Kull et al., 2019; Wang, 2023). This provides a clear probabilistic foundation for our approach.

Nevertheless, a potential limitation of this design is that logits from different heads may be poorly calibrated, leading to unfair comparisons across tasks. To mitigate this, we adopt temperature scaling (Guo et al., 2017) during training, which smooths the predicted distribution, prevents overconfident logits, stabilize cross-entropy optimization, and improves calibration. At inference, a properly chosen temperature τ helps align the output scales of different heads, making the concatenated logits more comparable across tasks. For future work, more advanced calibration or normalization techniques (e.g., Dirichlet calibration (Kull et al., 2019) or other recent approaches (Wang, 2023)) could be implemented in our setting to further enhance fairness and robustness.

Another limitation arises when tasks contain highly similar classes (e.g., visually similar categories). In such cases, the corresponding heads may both assign high confidence to the same input and the global $\arg\max$ will attribute solely based on raw logit magnitudes. This can lead to inconsistent decisions across tasks. Future extension could explore logit normalization or gating mechanisms to explicitly handle such conflicts, building on the joint guarantees of our proposed propositions 1 and 2.

E.2 Prompt Position

In practice, we adopt the strategy of appending prompts after the image patches, following NSP^2 (Lu et al., 2024), to ensure direct comparability. From lemmas 1 and 3, it follows that the preservation conditions in propositions 1 and 2 are invariant to the relative position of prompt tokens within the input sequence. In other words, whether prompts are prepended before the [CLS] token or appended after the patch embeddings, the algebraic form of the preservation constraints remains unchanged. This invariance arises from the row-wise independence of LayerNorm and MLP operations, which treat each token identically regardless of ordering. While these properties hold in theory, a systematic empirical investigation into how different prompt insertion strategies influence performance preservation is beyond the scope of this work. We consider this an interesting and important direction for future research.

E.3 Future Directions and Limitations

Beyond the scope of this work, several limitations remain: (i) our theoretical guarantees cover LayerNorm and MHSA but do not fully address nonlinearities from softmax attention; (ii) task-specific heads may not scale well to highly heterogeneous tasks; and (iii) our experiments focus on single-modality vision tasks.

Promising extensions include multi-modal continual learning with vision–language models, adaptive prompt allocation under task heterogeneity, and tighter theoretical analysis of softmax and other nonlinear components.

F LLM Usage Statement

We acknowledge the use of Large Language Models (LLMs) as an assistive tool in the preparation of this work. Their role was limited to grammar refinement, LaTeX formatting assistance, language polishing, and minor code debugging/modification. No part of the research methodology, experiments, or data analysis was delegated to LLMs. All conceptual contributions, design of experiments, and interpretation of results were carried out by the authors. The authors take full responsibility for all content presented in this paper, including any text or code that may have been refined with the assistance of LLMs.

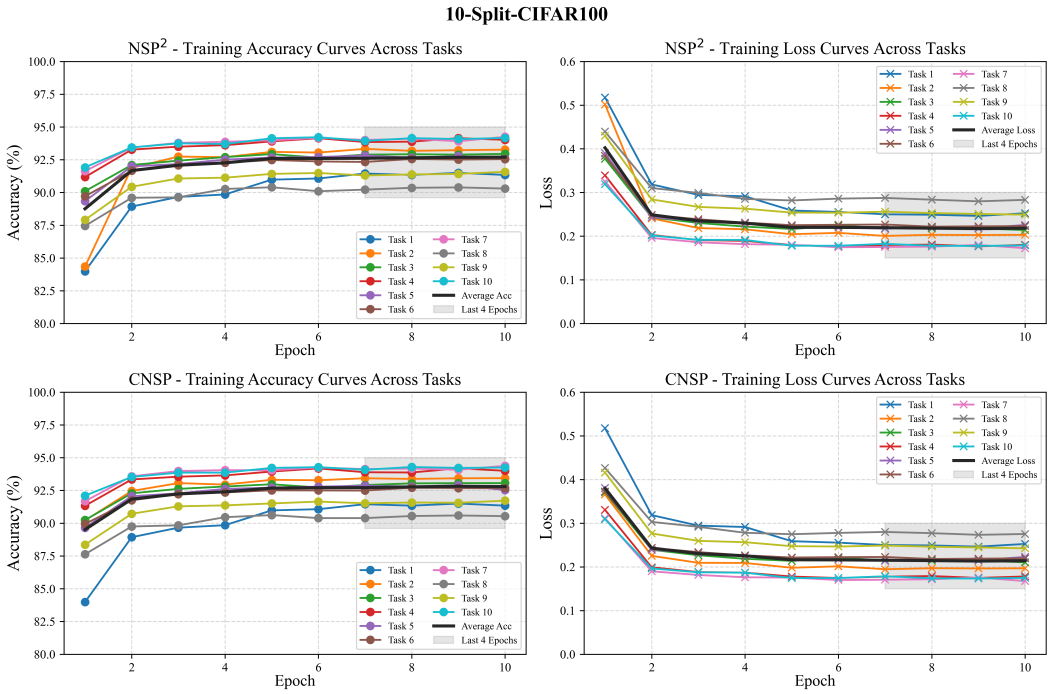

Figure 6: Training accuracy and loss curves across 10 tasks on the 10-Split-CIFAR100 benchmark. The top row reports results of NSP^2, and the bottom row reports CNSP. Left: training accuracy across tasks. Right: training loss across tasks. Each colored curve corresponds to one task, while the black curve denotes the average across all tasks at each epoch.

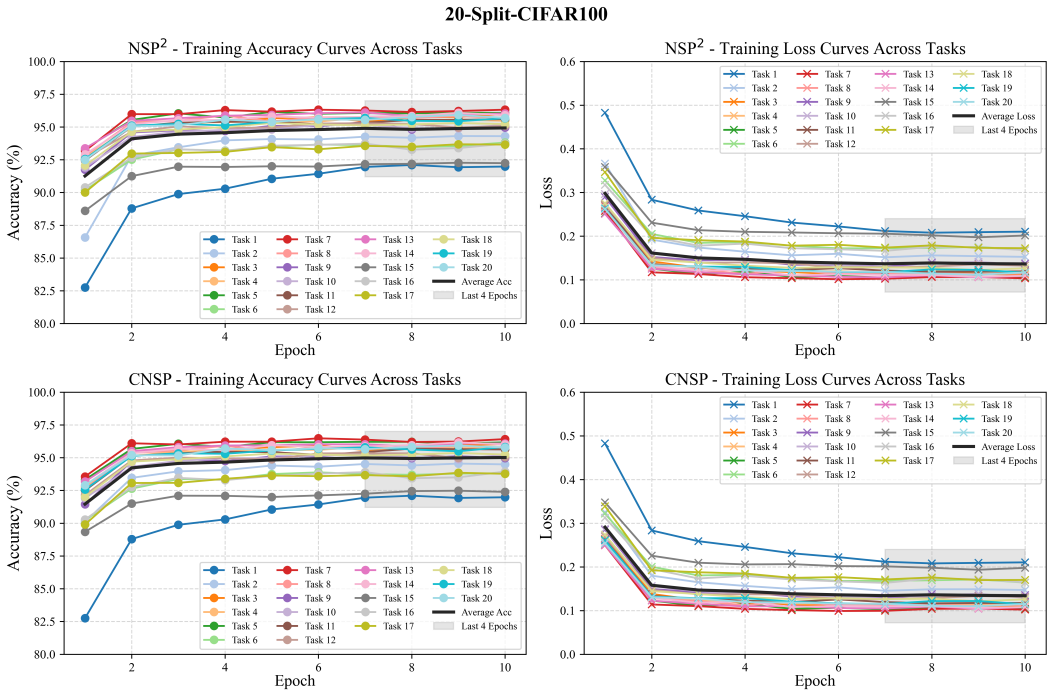

Figure 7: Training accuracy and loss curves across 20 tasks on the 20-Split-CIFAR100 benchmark. The top row reports results of NSP^2, and the bottom row reports CNSP. Left: training accuracy across tasks. Right: training loss across tasks. Each colored curve corresponds to one task, while the black curve denotes the average across all tasks at each epoch.

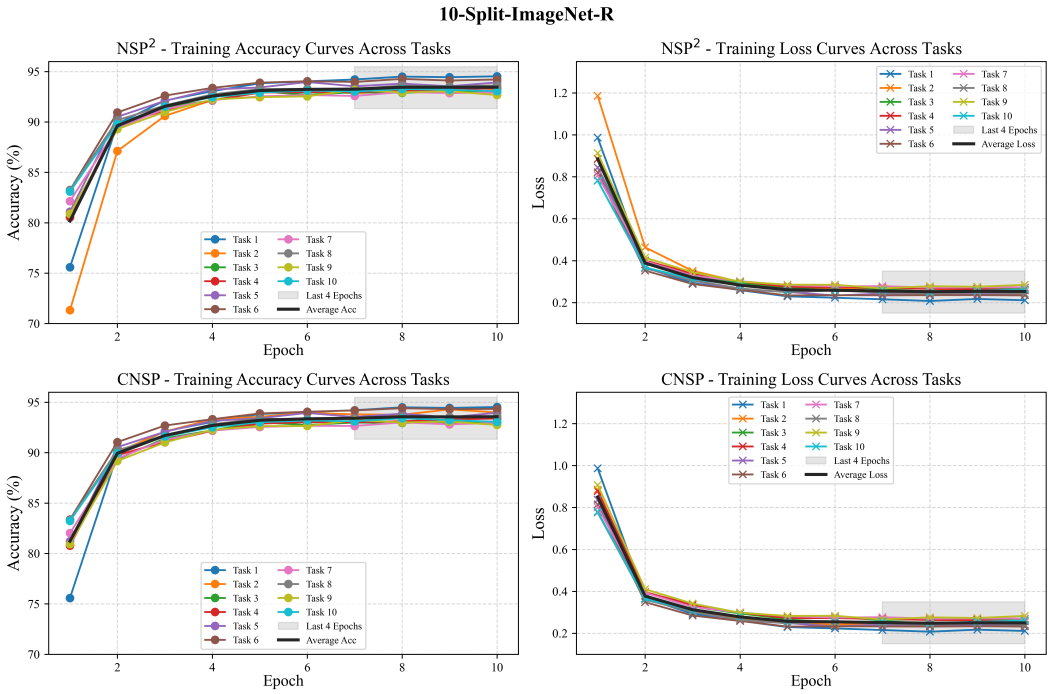

Figure 8: Training accuracy and loss curves across 10 tasks on the 10-Split-ImageNet-R benchmark. The top row reports results of NSP^2, and the bottom row reports CNSP. Left: training accuracy across tasks. Right: training loss across tasks. Each colored curve corresponds to one task, while the black curve denotes the average across all tasks at each epoch.

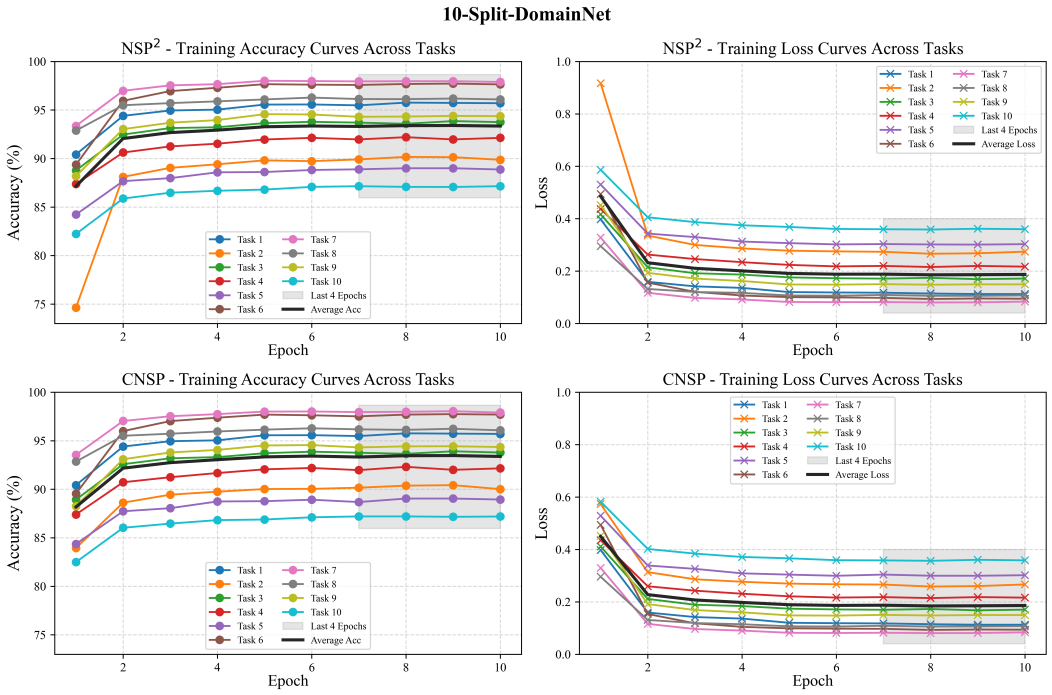

Figure 9: Training accuracy and loss curves across 10 tasks on the 10-Split-DomainNet benchmark. The top row reports results of NSP^2, and the bottom row reports CNSP. Left: training accuracy across tasks. Right: training loss across tasks. Each colored curve corresponds to one task, while the black curve denotes the average across all tasks at each epoch.