Towards Principled Prompt-Based Continual Learning via Consistent Preservation

Yunjie HAN[1, 2], Huiyun LI[3, *], Zhengmin JIANG[4, *], Shunran ZHANG[5], Ming SANG[6], Tiantian XU[1], Jia LIU[1, *]
[1]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Shenzhen University of Advanced Technology, Shenzhen, China
[4]City University of Hong Kong, Hong Kong, China
[5]University of Macau, Macau, China
[6]Harbin Institute of Technology (Shenzhen), Shenzhen, China

***Abstract*—Continual Learning (CL) aims to learn from a stream of tasks without forgetting previously acquired knowledge, yet catastrophic forgetting remains a major challenge. Prompt-based CL has recently achieved strong empirical performance, but its theoretical foundations are still limited. In this work, we provide a unified and rigorous characterization of consistent performance preservation for prompt-tuned Transformers in CL. We show that preserving past-task performance can be decomposed into two jointly sufficient requirements, and we derive explicit sufficient consistency conditions under a full prompt-tuned Transformer parameterization. Importantly, these conditions admit a tractable orthogonal null-space projection rule, yielding our Consistent Null-Space Projection (CNSP) method. Extensive experiments across diverse benchmarks and Transformer backbones demonstrate consistent gains in accuracy and reduced forgetting, with particularly pronounced improvements in high-dimensional and domain-shift settings.**

***Index Terms*—continual learning, visual prompt tuning, orthogonal projection, anti-forgetting, vision transformers**

I. Introduction

Continual Learning (CL) aims to equip models with the ability to learn from a stream of tasks while retaining knowledge acquired from earlier ones. Despite substantial progress, catastrophic forgetting [1]—the degradation of performance on previously learned tasks when adapting to new ones—remains a central challenge [2].

Recently, prompt-tuning methods, particularly Visual Prompt Tuning (VPT) [3] with Vision Transformers (ViTs) [4], have emerged as a promising rehearsal-free solution for CL [2]. By updating only lightweight prompts while freezing the backbone, these methods achieve both computational and storage efficiency. While several prompt-based methods for CL(e.g., L2P [5], DualPrompt [6], Coda-Prompt [7]) have demonstrated competitive empirical performance, theoretical foundations of prompt-based CL remain underdeveloped. Classical orthogonal-projection methods [8]–[10] guarantee retention by deriving explicit consistency conditions in linear or convolutional networks, but these guarantees do not transfer to Transformers: nonlinear LayerNorm, learned QKV projections, and multi-head attention invalidate linear assumptions.

To bridge this theoretical gap, recent efforts have adapted projection ideas to ViTs/VPT [11], [12]. PGP [11] approximates Multi-Head Self-Attention (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 [12] advances this line of work by deriving approximate sufficient conditions for preserving representations under prompt updates. However, NSP^2 still leaves several fundamental aspects of the Transformer pipeline theoretically unresolved: the extension from single-head to multi-head attention lacks principled justification; LayerNorm's broadcasted affine structure is simplified in ways that obscure its algebraic properties; strong assumptions on prompt distribution (mean–variance invariance) can introduce instability; and the classification head is omitted from the overall formulation. Consequently, existing analyses offer valuable local insights but fall short of providing an end-to-end account of how prompt updates propagate through the Transformer and preserve past-task performance.

Motivated by this theoretical gap, we revisit prompt-based CL from an end-to-end, matrix-level perspective. We introduce a principled framework that provides a unified performance-consistency formulation for prompt-tuned Transformers in CL, and further refine projection-based analyses such as NSP^2. We prove that performance preservation admits a clean decomposition into two jointly sufficient components: *feature preservation*, ensuring that MHSA+LayerNorm representations for previous tasks remain stable, and *head preservation*, guaranteeing decision-level consistency for previously learned classifier(s). Building on this decomposition, we derive sufficient conditions for preserving past-task performance by jointly accounting for the coupled roles of MHSA, LayerNorm, and classification heads under full prompt-tuned Transformer parameterization. These conditions give rise to our algorithm **Consistent Null-Space Projection (CNSP)**.

Our contributions are summarized as follows:

(i) **Unified theoretical characterization.** We present a

*Corresponding authors: Zhengmin JIANG(zhengmin.jiang@cityu.edu.hk), Jia LIU (jia.liu1@siat.ac.cn), and Huiyun LI (huiyun.li@suat-sz.edu.cn).

unified and mathematically rigorous characterization of performance preservation in prompt-based CL, showing that it can be reduced to a consistent preservation principle that jointly enforces feature and head preservation.

(ii) **Exact sufficient conditions.** Under a full prompt-tuned Transformer parameterization, we provide an algebraically precise matrix-level modeling of MHSA, LayerNorm, and classification heads, and derive exact sufficient conditions for preserving past-task performance. We then convert these theoretical conditions into an orthogonal projection-based update rule, resulting in the algorithm CNSP.

(iii) **Empirical gains.** CNSP consistently mitigates forgetting and improves accuracy across diverse benchmarks and Transformer backbones, with particularly pronounced gains in high-dimensional and domain-shift settings—without architectural heuristics or task-specific prompt designs.

II. Related Works

A. Prompt-based Methods for Continual Learning

Prompt-based methods have emerged as a promising paradigm for CL, offering lightweight task adaptation with reduced interference. Representative approaches include L2P [5] and DualPrompt [6], which employ dynamic prompt selection and disentangle shared versus task-specific prompts, respectively. Subsequent works such as CPrompt [13], CODA-Prompt [7], and EvoPrompt [14] further improve prompt consistency, compositionality, and long-term adaptability. More recently, LGCL [15] and ConvPrompt [16] incorporate semantic guidance and hierarchical prompt structures.

Beyond empirical designs, recent efforts pursue principled formulations. PGP [11] introduces gradient orthogonality in the joint input-prompt space by linearizing MHSA and LayerNorm, while NSP^2 [12] derives null-space projection conditions through explicit analysis of attention and normalization.

Despite their effectiveness, existing theoretical analyses still rely on simplified treatments of multi-head attention and LayerNorm, limiting rigor and robustness. Our work addresses these limitations through an exact matrix-form characterization and principled sufficient conditions for performance preservation.

B. Orthogonal Projection Methods for Continual Learning

Orthogonal projection methods have been widely studied in CNN and MLP (e.g., OWM [18], GPM [8], TRGP [19]) 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 $\mathbf{X}^\top(\mathbf{\Theta}+\Delta\mathbf{\Theta})=\mathbf{X}^\top\mathbf{\Theta}$, where $\mathbf{X}$ denotes input features, $\mathbf{\Theta}$ the convolutional or linear parameters, and $\Delta\mathbf{\Theta}$ the update. Enforcing $\mathbf{X}^\top\Delta\mathbf{\Theta}=\mathbf{0}$ 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 [8], which projects gradients onto the null space of previously learned inputs; TRGP [19], which refines GPM by scaling task gradients with a trust region matrix before projection; and Connector [20] interpolates a normally updated model with one constrained by projection.

These methods work well in linear or convolutional architectures, where orthogonality condition holds exactly. However, in Transformers, nonlinear MHSA, softmax, and LayerNorm interactions violate this linearity, making classical projections unreliable. Recent works have applied orthogonal projection to prompt tuning [11], [12]. NSP^2 [12] unfolds VPT forward propagation and derives approximate sufficient conditions for performance preservation via null-space projection. Building on this foundation, we introduce stricter and more general consistency conditions alongside stable optimization strategies, thereby strengthening projection-based CL in VPT.

III. Preliminaries

A. Notations

We adopt the following notational conventions throughout this paper: (i) Non-boldface letters denote positive integers, e.g., $a, A \in \mathbb{N}^+$. (ii) Bold lowercase letters represent vectors, e.g., $\boldsymbol{a} \in \mathbb{R}^n$; bold upright uppercase letters denote matrices, e.g., $\mathbf{A} \in \mathbb{R}^{n\times m}$. (iii) Row concatenation: for matrices $\mathbf{A}^{(i)} \in \mathbb{R}^{n\times d}$, $\left[\mathbf{A}^{(1)};\ldots;\mathbf{A}^{(N)}\right] \in \mathbb{R}^{Nn\times d}$. (v) Slicing: if $\mathbf{A}=\left[\mathbf{A}^{(1)};\mathbf{A}^{(2)}\right]$ with $\mathbf{A}^{(1)} \in \mathbb{R}^{n\times d}$, then $\mathbf{A}[:n]=\mathbf{A}^{(1)}$. (vi) Bias absorption[1]: for simplicity, we omit bias terms in all linear projections in MLP and MHSA.

B. Forward Propagation in VPT-deep

A ViT with L blocks is denoted by $f_{\mathrm{ViT}}\left(\mathbf{x} \mid \mathbf{\Theta}_0; \mathbf{W}, \mathbf{b}\right)$, where $\mathbf{x}$ is the input image, $\mathbf{\Theta}_0$ denotes the frozen backbone and $\mathbf{W} \in \mathbb{R}^{D\times C}$, $\mathbf{b} \in \mathbb{R}^C$ are learnable classification head parameters, with D the embedding dimension and C the number of classes. Let $\mathbf{X}^{(l)} \in \mathbb{R}^{(N+1)\times D}$ be the input to block l, consisting of N image tokens and one [CLS] token.

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

$$\mathbf{O}_{\mathbf{Z}}^{(l)} = \mathrm{MHSA}(\mathrm{LN}(\mathbf{Z}^{(l)})), \tag{1}$$

$$\mathbf{B}_{\mathbf{Z}}^{(l)} = \mathbf{Z}^{(l)} + \mathbf{O}_{\mathbf{Z}}^{(l)}, \tag{2}$$

$$\mathbf{E}_{\mathbf{Z}}^{(l)} = \mathbf{B}_{\mathbf{Z}}^{(l)} + \mathrm{MLP}(\mathrm{LN}(\mathbf{B}_{\mathbf{Z}}^{(l)})). \tag{3}$$

Before entering the next block, the current prompt tokens are replaced by the next block's learnable prompts:

$$\mathbf{X}^{(l+1)} = \mathbf{E}_{\mathbf{Z}}^{(l)}[:N+1], \quad \mathbf{Z}^{(l+1)} = [\mathbf{X}^{(l+1)};\mathbf{P}^{(l+1)}]. \tag{4}$$

The final [CLS] token is passed into the classification head:

$$\begin{aligned} f_{\mathrm{ViT}}\left(\mathbf{x} \mid \mathbf{\Theta}_0; \mathbf{W}, \mathbf{b}; \{\mathbf{P}^{(l)}\}_{l=1}^{L}\right) = \\ softmax\left(\mathbf{E}_{\mathbf{Z}}^{(L)}[:1]\mathbf{W}+\mathbf{b}\right). \end{aligned} \tag{5}$$

[1]This is without loss of generality: for $\mathbf{y}=\mathbf{W}\mathbf{x}+\mathbf{b}$, define $\tilde{\mathbf{x}}=[\mathbf{x}^\top,1]^\top$ and $\tilde{\mathbf{W}}=[\mathbf{W},\mathbf{b}]$, then $\mathbf{y}=\tilde{\mathbf{W}}\tilde{\mathbf{x}}$.

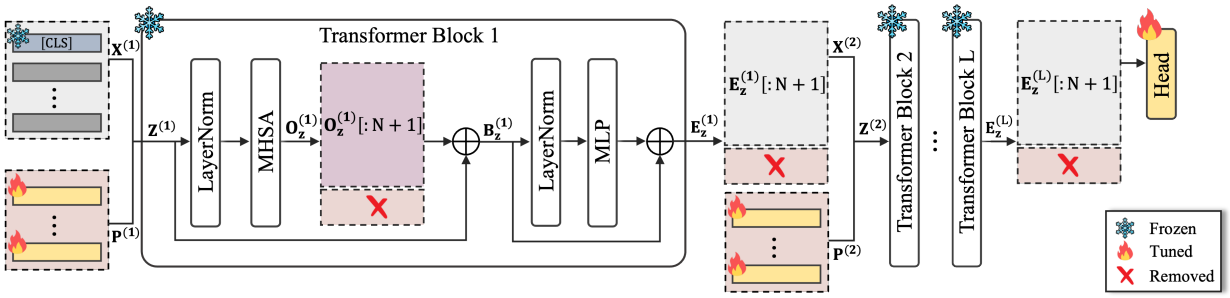

Fig. 1: Forward propagation in VPT-deep. Image tokens and prompt tokens are jointly processed by LayerNorm, MHSA, and MLP with residual connections in each transformer block. ✕ indicates information discarded before entering the next layer.

C. Problem Formulation: Continual Learning with VPT-deep

Consider a CL setting with a sequence of tasks $\{\mathcal{T}_t\}_{t=1}^{T}$. Each task $\mathcal{T}_t$ provides a dataset $\mathcal{D}_t = \{(\mathbf{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{N_t}$, where $\mathbf{x}_t^{(i)}$ is an image and $y_t^{(i)}$ its label. When task $\mathcal{T}_t$ arrives, the model is trained only on $\mathcal{D}_t$ by updating the prompts, yielding $f_{\text{ViT}}\left(\cdot \mid \mathbf{\Theta}_0; \mathbf{W}_t, \mathbf{b}_t; \{\mathbf{P}_t^{(l)}\}_{l=1}^{L}\right)$, while data from previous tasks are unavailable. CL aims to preserve performance on past tasks after learning new ones. Formally, we require:

$$\begin{aligned} &f_{\text{ViT}}(\mathbf{x}_t^{(i)} \mid \mathbf{\Theta}; \mathbf{W}_{t+1}, \mathbf{b}_{t+1}; \{\mathbf{P}_{t+1}^{(l)}\}_{l=1}^{L}) = \\ &f_{\text{ViT}}(\mathbf{x}_t^{(i)} \mid \mathbf{\Theta}; \mathbf{W}_t, \mathbf{b}_t; \{\mathbf{P}_t^{(l)}\}_{l=1}^{L}), \quad (i = 1, \ldots, N_t), \end{aligned} \tag{6}$$

which enforces that model outputs on each input $\mathbf{x}_t^{(i)}$ from $\mathcal{T}_t$ remain unchanged.

IV. Method

Our method builds on (6). We prove that prompt-based CL reduces to *feature* and *head preservation*, enforced via constrained prompt updates and task-specific heads.

A. Consistent Preservation

For tasks $\mathcal{T}_t$ and $\mathcal{T}_{t+1}$, we denote $\mathbf{Z}_{t,t+1}^{(l)} = \left[\mathbf{X}_t^{(l)}; \mathbf{P}_{t+1}^{(l)}\right]$ and $\mathbf{Z}_{t,t}^{(l)} = \left[\mathbf{X}_t^{(l)}; \mathbf{P}_t^{(l)}\right]$. We introduce two key propositions which together constitute sufficient conditions for (6).

Proposition 1 (Feature Preservation)**.** *If the image-token part of attention outputs remains unchanged for all blocks in VPT, i.e.,*

$$\mathbf{O}_{\mathbf{Z}_{t,t+1}}^{(l)}[:N+1] = \mathbf{O}_{\mathbf{Z}_{t,t}}^{(l)}[:N+1], \quad l = 1, 2, \ldots, L, \tag{7}$$

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

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

Proof. See Appendix A

Proposition 2 (Head Preservation)**.** *If the final [CLS] token remains unchanged and the head updates satisfy:*

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

where $\Delta\mathbf{W}_t$ and $\Delta\mathbf{b}_t$ are the parameter updates of the classification head from $\mathcal{T}_t$ to $\mathcal{T}_{t+1}$, then (6) *is satisfied.*

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

$$\begin{aligned} &\mathbf{E}_{\mathbf{Z}_{t,t+1}}^{(L)}[:1]\mathbf{W}_{t+1} + \mathbf{b}_{t+1} \\ &= \mathbf{E}_{\mathbf{Z}_{t,t+1}}^{(L)}[:1]\left(\mathbf{W}_t + \Delta\mathbf{W}_t\right) + \left(\mathbf{b}_t + \Delta\mathbf{b}_t\right) \\ &= \mathbf{E}_{\mathbf{Z}_{t,t}}^{(L)}[:1]\mathbf{W}_t + \mathbf{b}_t, \end{aligned} \tag{10}$$

which implies (6) by (5). □

Together, Propositions 1 and 2 reduce (6) to *feature preservation* (invariant image-token representations under prompt updates) and *head preservation* (consistent classifier mapping on the preserved [CLS] token).

B. Unfolding Forward Propagation of LayerNorm and MHSA

By Proposition 1, feature preservation can be analyzed blockwise, involving only LayerNorm and MHSA modules. As (7) must holds for all layers, we omit the layer index (l). For the current block, let the input sequence be $\mathbf{Z} = [\mathbf{X}; \mathbf{P}]$. Fig. 2 illustrates the unrolled forward pass of LayerNorm and MHSA in one transformer block. We will show that computations involving prompt queries can be ignored.

First, $\mathbf{Z}$ is passed through LayerNorm, we introduce Lemma 1 (proof in Appendix A):

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

By Lemma 1, $\text{LN}(\mathbf{Z}) = [\text{LN}(\mathbf{X}); \text{LN}(\mathbf{P})]$. Next, the normalized sequence undergoes QKV transformation, including QKV projections and multi-head splitting. For head $h \in \{1, \ldots, H\}$, we have[2]:

$$\mathbf{\Phi}_{\mathbf{Z}}^{(h)} = \text{LN}(\mathbf{Z})\,\mathbf{W}_{\phi}^{(h)} = \begin{bmatrix} \text{LN}(\mathbf{X})\,\mathbf{W}_{\phi}^{(h)} \\ \text{LN}(\mathbf{P})\,\mathbf{W}_{\phi}^{(h)} \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{\Phi}_{\mathbf{X}}^{(h)} \\ \mathbf{\Phi}_{\mathbf{P}}^{(h)} \end{bmatrix}, \tag{11}$$

where $\mathbf{\Phi} \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}\}$ denotes the Q/K/V matrix, and $\phi \in \{q, k, v\}$ is its corresponding index. Here $\mathbf{W}_{\phi}^{(h)} \in \mathbb{R}^{D\times D_H}$ is the projection matrix for head h, with $D_H = D/H$. Moreover, $\mathbf{\Phi}_{\mathbf{X}}^{(h)} \in \mathbb{R}^{(N+1)\times D_H}$ and $\mathbf{\Phi}_{\mathbf{P}}^{(h)} \in \mathbb{R}^{M\times D_H}$ are the submatrices of $\mathbf{\Phi}_{\mathbf{Z}}^{(h)}$ corresponding to the input tokens and prompt tokens, respectively.

[2]As explained in the Section III-A, we omit bias terms for simplicity.

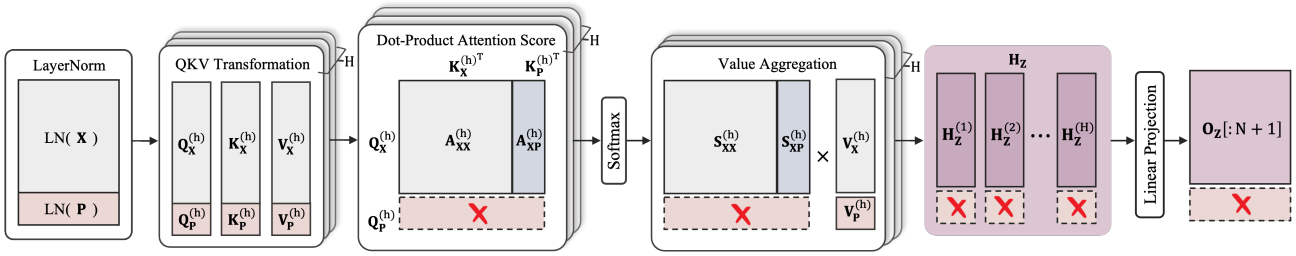

Fig. 2: Detailed illustration of LayerNorm and the multi-head self-attention mechanism in a VPT transformer block, showing how image and prompt embeddings are jointly processed. ✗ indicates information discarded before entering the next layer.

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

$$\mathbf{H}_{\mathbf{Z}}^{(h)} = softmax\Big(\mathbf{Q}_{\mathbf{Z}}^{(h)}{\mathbf{K}_{\mathbf{Z}}^{(h)}}^{\top}/\sqrt{D_H}\Big)\mathbf{V}_{\mathbf{Z}}^{(h)}. \tag{12}$$

Finally, concatenating across heads and applying the output projection gives

$$\mathbf{O}_{\mathbf{Z}} = \big[\mathbf{H}_{\mathbf{Z}}^{(h)}\big]_{h=1}^{H}\mathbf{W}_o. \tag{13}$$

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

$$\begin{aligned}\Big(\big[\mathbf{H}_{\mathbf{Z}}^{(h)}\big]_{h=1}^{H}\mathbf{W}_o\Big)[:N+1] &= \big[\mathbf{H}_{\mathbf{Z}}^{(h)}\big]_{h=1}^{H}[:N+1]\mathbf{W}_o \\ &= \big[softmax\Big(\mathbf{Q}_{\mathbf{Z}}^{(h)}{\mathbf{K}_{\mathbf{Z}}^{(h)}}^{\top}/\sqrt{D_H}\Big)\mathbf{V}_{\mathbf{Z}}^{(h)}\big]_{h=1}^{H}\mathbf{W}_o.\end{aligned} \tag{14}$$

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

C. Sufficient Conditions for Feature Preservation in VPT

We further derive sufficient conditions for feature preservation, such that they can be implemented as constraints on the prompt updates during optimization.

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

$$\mathbf{A}_{\mathbf{Z}}^{(h)} = \frac{\mathbf{Q}_{\mathbf{X}}^{(h)}\Big[{\mathbf{K}_{\mathbf{X}}^{(h)}}^{\top} \quad {\mathbf{K}_{\mathbf{P}}^{(h)}}^{\top}\Big]}{\sqrt{D_H}} \triangleq \Big[\mathbf{A}_{\mathbf{XX}}^{(h)} \quad \mathbf{A}_{\mathbf{XP}}^{(h)}\Big], \tag{15}$$

$$\mathbf{S}_{\mathbf{Z}}^{(h)} = softmax\left(\Big[\mathbf{A}_{\mathbf{XX}}^{(h)} \quad \mathbf{A}_{\mathbf{XP}}^{(h)}\Big]\right) \triangleq \Big[\mathbf{S}_{\mathbf{XX}}^{(h)} \quad \mathbf{S}_{\mathbf{XP}}^{(h)}\Big], \tag{16}$$

where $\mathbf{A}_{\mathbf{XX}}^{(h)}, \mathbf{S}_{\mathbf{XX}}^{(h)} \in \mathbb{R}^{(N+1)\times(N+1)}$ and $\mathbf{A}_{\mathbf{XP}}^{(h)}, \mathbf{S}_{\mathbf{XP}}^{(h)} \in \mathbb{R}^{(N+1)\times M}$ (see Fig. 2 for the blockwise notation).

Consider $\mathcal{T}_t$ and $\mathcal{T}_{t+1}$. By Proposition 1, (13) and (14), feature preservation requires

$$\begin{aligned}&\Big[\mathbf{S}_{\mathbf{X}_t\mathbf{X}_t}^{(h)}\mathbf{V}_{\mathbf{X}_t}^{(h)} + \mathbf{S}_{\mathbf{X}_t\mathbf{P}_{t+1}}^{(h)}\mathbf{V}_{\mathbf{P}_{t+1}}^{(h)}\Big]_{h=1}^{H}\mathbf{W}_o \\ &= \Big[\mathbf{S}_{\mathbf{X}_t\mathbf{X}_t}^{(h)}\mathbf{V}_{\mathbf{X}_t}^{(h)} + \mathbf{S}_{\mathbf{X}_t\mathbf{P}_t}^{(h)}\mathbf{V}_{\mathbf{P}_t}^{(h)}\Big]_{h=1}^{H}\mathbf{W}_o\end{aligned} \tag{17}$$

In MHSA, $\mathbf{W}_o$ aggregates per-head outputs [21]. Since $\mathbf{W}_o$ may be singular, we impose stronger sufficient conditions by requiring per-head equality:

$$\mathbf{S}_{\mathbf{X}_t\mathbf{P}_{t+1}}^{(h)}\mathbf{V}_{\mathbf{P}_{t+1}}^{(h)} - \mathbf{S}_{\mathbf{X}_t\mathbf{P}_t}^{(h)}\mathbf{V}_{\mathbf{P}_t}^{(h)} = \mathbf{0}. \tag{18}$$

Direct algebraic characterization is intractable due to the nonlinearity of softmax. Following NSP^2 [12], we instead require invariance of the attention scores:

$$\begin{aligned}\mathbf{A}_{\mathbf{Z}_{t,t}}^{(h)} = \mathbf{A}_{\mathbf{Z}_{t,t+1}}^{(h)} \Leftrightarrow \mathbf{Q}_{\mathbf{X}_t}^{(h)}\left({\mathbf{K}_{\mathbf{P}_{t+1}}^{(h)}}^{\top} - {\mathbf{K}_{\mathbf{P}_t}^{(h)}}^{\top}\right) = \mathbf{0} \\ \Rightarrow \mathbf{S}_{\mathbf{Z}_{t,t}}^{(h)} = \mathbf{S}_{\mathbf{Z}_{t,t+1}}^{(h)}.\end{aligned} \tag{19}$$

That is, the attention scores from image tokens to prompts remain unchanged before and after the prompt update. Under (19), condition (18) reduces to

$$\mathbf{S}_{\mathbf{X}_t\mathbf{P}_t}^{(h)}\left(\mathbf{V}_{\mathbf{P}_{t+1}}^{(h)} - \mathbf{V}_{\mathbf{P}_t}^{(h)}\right) = \mathbf{0}. \tag{20}$$

We obtain the following per-head sufficient conditions for feature preservation:

$$\begin{cases}\mathbf{Q}_{\mathbf{X}_t}^{(h)}\left({\mathbf{K}_{\mathbf{P}_{t+1}}^{(h)}}^{\top} - {\mathbf{K}_{\mathbf{P}_t}^{(h)}}^{\top}\right) = \mathbf{0} \\ \mathbf{S}_{\mathbf{X}_t\mathbf{P}_t}^{(h)}\left(\mathbf{V}_{\mathbf{P}_{t+1}}^{(h)} - \mathbf{V}_{\mathbf{P}_t}^{(h)}\right) = \mathbf{0}\end{cases} \tag{21}$$

Using (11), they can be rewritten in terms of the prompt representations as:

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

where $\mathbf{P}_{t+1} = \mathbf{P}_t + \Delta\mathbf{P}$.

To obtain conditions depending explicitly on $\Delta\mathbf{P}$, we expand LayerNorm:

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

where $\boldsymbol{\mu}_{\mathbf{P}_t+\Delta\mathbf{P}}, \boldsymbol{\sigma}_{\mathbf{P}_t+\Delta\mathbf{P}} \in \mathbb{R}^M$ are row-wise statistics and $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^D$ are shared affine parameters (broadcast in implementation). To enable an algebraically exact characterization of prompt interactions, we explicitly reformulate LayerNorm in matrix form. Let $\mathbf{C} = \mathbf{I}_D - \frac{1}{D}\mathbf{1}_D\mathbf{1}_D^{\top} (\in \mathbb{R}^{D\times D})$ be the centering matrix, $\mathbf{D}_{\boldsymbol{\sigma}_{\mathbf{P}_t+\Delta\mathbf{P}}} = \mathrm{diag}\left(\boldsymbol{\sigma}_{\mathbf{P}_t+\Delta\mathbf{P}}\right) \in \mathbb{R}^{M\times M}$ [3], and $\boldsymbol{\Gamma} = \mathrm{diag}\left(\boldsymbol{\gamma}\right) \in \mathbb{R}^{D\times D}$. Then

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

[3] 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.

Note that pretrained $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are frozen. We introduce a variance-preservation condition

$$\sigma_{\mathbf{P}_t} = \sigma_{\mathbf{P}_t + \Delta\mathbf{P}} \triangleq \sigma, \tag{25}$$

then

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

Substituting (26) into (22), the sufficient conditions reduce to

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

Therefore, under (25), (27) provides a set of per-head sufficient conditions for feature preservation (8).

D. Optimization under Feature Preservation Constraints

To enforce the variance-preservation condition (25), we adopt a soft constraint by introducing a standard deviation alignment loss:

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

where $\mathcal{L}$ is the total loss, $\mathcal{L}_{\mathrm{ce}}$ is the cross-entropy loss, and λ balances the alignment term. Under this condition, we obtain the sufficient conditions in (27), constraining $\Delta\mathbf{P}$.

We adopt a right-side nullification form on the second constraint $\mathbf{S}_{\mathbf{X}_t\mathbf{P}_t}^{(h)}\mathbf{D}_{\boldsymbol{\sigma}}^{-1}\Delta\mathbf{P}\mathbf{C}\boldsymbol{\Gamma}\mathbf{W}_v^{(h)} = \mathbf{0}$. The rationale is to avoid circular dependence: if $\mathbf{D}_{\boldsymbol{\sigma}}^{-1}$ remains on the solving side, then $\Delta\mathbf{P}$ becomes entangled with the current statistic $\boldsymbol{\sigma}$, complicating implementation and hurting numerical stability. In contrast, the right-sided formulation $\Delta\mathbf{P}\mathbf{C}\boldsymbol{\Gamma}\mathbf{W}_v^{(h)} = \mathbf{0}$ removes explicit dependence on $\boldsymbol{\sigma}$, yielding more stable behavior. Moreover, it involves only frozen pretrained parameters, thereby reducing computational overhead. Let $\mathbf{R}_k^{(h)} = \mathbf{C}\boldsymbol{\Gamma}\mathbf{W}_k^{(h)}{\mathbf{Q}_{\mathbf{X}_t}^{(h)}}^\top$, $\mathbf{R}_v^{(h)} = \mathbf{C}\boldsymbol{\Gamma}\mathbf{W}_v^{(h)}$, and define

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

Our implementation enforces $\Delta\mathbf{P}\mathbf{R} = \mathbf{0}$ at every optimization step, equivalently constraining each row of $\Delta\mathbf{P}$ to the left null space of $\mathbf{R}$. Since $\mathbf{R}$ grows with tasks, we avoid handling it directly via the following lemma:

Lemma 2. *For $\mathbf{A} \in \mathbb{R}^{n\times m}$ with left null space $\mathcal{N}_L(\mathbf{A}) = \{\mathbf{x} \mid \mathbf{x}\mathbf{A} = \mathbf{0}\}$, it holds that $\mathcal{N}_L(\mathbf{A}) = \mathcal{N}_L\left(\mathbf{A}\mathbf{A}^\top\right)$ [22].*

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

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

which guarantees (27).

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

Algorithm 1 CNSP for VPT in Continual Learning

Inputs: Datasets $\{\mathcal{D}_t\}_{t=1}^T$; ViT $f_{\mathrm{ViT}}(\cdot \mid \{\mathbf{P}^{(l)}\}_{l=1}^L)$ with learnable prompts $\mathbf{P}^{(l)}(l = 1, \ldots, L)$;
Outputs: Optimized prompts $\mathbf{P}^{(l)}$

Initialize $\mathbf{P}^{(l)}$; set $\mathbf{G}^{(l)} = \mathbf{0}$, $\mathbf{B}_{\mathbf{R}}^{(l)} = \mathbf{I}$
for $t = 1, 2, \ldots, T$ **do**
  for each training iteration on $\mathcal{D}_t$ **do**
    Sample a mini-batch $(\mathbf{x}, y) \sim \mathcal{D}_t$
    $\hat{y} \leftarrow f_{\mathrm{ViT}}(\mathbf{x}_t \mid \{\mathbf{P}_t^{(l)}\}_{l=1}^L)$
    $\mathcal{L}_{\text{total}} \leftarrow \mathrm{CrossEntropy}(\hat{y}, y)$
    $\mathcal{L} \leftarrow \mathrm{CE}(\hat{y}, y) + \lambda \cdot \mathbf{1}_{[t>1]} \cdot \mathcal{L}_{\mathrm{std}}$ ▷ $\mathcal{L}_{\mathrm{std}}$: see (28)
    Get raw update $\Delta\mathbf{P}_{\text{raw}}^{(l)}$ from the optimizer on $\mathcal{L}$
    $\Delta\mathbf{P}^{(l)} \leftarrow \Delta\mathbf{P}_{\mathrm{raw}}^{(l)}\mathbf{B}_{\mathbf{R}}^{(l)}$ if $t > 1$ else $\Delta\mathbf{P}_{\mathrm{raw}}^{(l)}$
    $\mathbf{P}^{(l)} \leftarrow \mathbf{P}^{(l)} - \mathrm{learning\_rate} \times \Delta\mathbf{P}^{(l)}$
  end for
  // update projection matrice for next task
  for $\mathbf{x}_t^{(i)} \in \mathcal{D}_t$ **do**
    $\mathbf{R}_v^{(l)} \leftarrow \mathbf{C}\boldsymbol{\Gamma}^{(l)}\mathbf{W}_v^{(l)}$, $\mathbf{R}_k^{(l)} \leftarrow \mathbf{C}\boldsymbol{\Gamma}^{(l)}\mathbf{W}_k^{(l)}{\mathbf{Q}_{\mathbf{X}_t^{(i)}}^{(l)}}^\top$
    $\mathbf{G}^{(l)} \leftarrow \mathbf{G}^{(l)} + \mathbf{R}_k^{(l)}{\mathbf{R}_k^{(l)}}^\top + \mathbf{R}_v^{(l)}{\mathbf{R}_v^{(l)}}^\top$
  end for
  $\mathbf{B}_{\mathbf{R}}^{(l)} \leftarrow \mathrm{Alg2}(\mathbf{G}^{(l)})$ ▷ using algorithm 2
end for

Algorithm 2 Computing Null-Space Projection Matrix

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

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

E. Ensuring Classification-Head Preservation

Proposition 2 requires the classification head to preserve its mapping from the [CLS] token across tasks. This requirement is naturally satisfied by task-specific heads: only the head of the current task is updated, while all previous heads are kept frozen. Under the feature-preservation constraint, the representations fed into earlier heads remain unchanged. Consequently, the head-preservation condition (9) is automatically satisfied when adopting task-specific classification heads in prompt-based continual learning.

V. Experiments

A. Experimental Settings

Benchmarks. We evaluate CNSP on three benchmarks commonly used in prompt-based CL: (i) 10/20-Split CIFAR-

TABLE I: Comparison with existing prompt-based methods using ImageNet-21K pretrained backbones. Results (%) are reported as mean ± standard deviation over three runs. Results marked with †, ‡, and § are reproduced by [11], [13], and [17], respectively.

Method	10S-CIFAR100		20S-CIFAR100		10S-ImageNet-R		10S-DomainNet	
	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
L2P [5]	$83.83_{\pm 0.04}$	$7.63_{\pm 0.30}$	$81.29_{\pm 0.43}$ †	$8.96_{\pm 0.38}$ †	$61.57_{\pm 0.66}$	$9.73_{\pm 0.47}$	$81.17_{\pm 0.83}$ ‡	$8.98_{\pm 1.25}$ ‡
DualPrompt [6]	$86.51_{\pm 0.33}$	$5.16_{\pm 0.09}$	$82.98_{\pm 0.47}$ †	$8.20_{\pm 0.08}$ †	$68.13_{\pm 0.49}$	$4.68_{\pm 0.20}$	$81.70_{\pm 0.78}$ ‡	$8.04_{\pm 0.31}$ ‡
CodaPrompt [7]	$86.25_{\pm 0.74}$	$1.67_{\pm 0.26}$	-	-	$75.45_{\pm 0.56}$	$1.64_{\pm 0.10}$	$80.04_{\pm 0.79}$ ‡	$10.16_{\pm 0.35}$ ‡
CPrompt [13]	$87.82_{\pm 0.21}$	$5.06_{\pm 0.50}$	$83.97_{\pm 0.31}$ §	$6.85_{\pm 0.43}$ §	$77.14_{\pm 0.11}$	$5.97_{\pm 0.68}$	$82.97_{\pm 0.34}$	$7.45_{\pm 0.93}$
EvoPrompt [14]	$87.97_{\pm 0.30}$	$2.60_{\pm 0.42}$	$84.64_{\pm 0.14}$	$3.98_{\pm 0.24}$	$76.83_{\pm 0.08}$	$2.78_{\pm 0.06}$	$79.50_{\pm 0.29}$ §	$3.81_{\pm 0.36}$ §
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 [15]	$87.23_{\pm 0.21}$	$5.10_{\pm 0.15}$	-	-	$69.46_{\pm 0.04}$	$4.20_{\pm 0.06}$	-	-
ConvPrompt [16]	$88.87_{\pm 0.33}$	$4.75_{\pm 0.15}$	$87.22_{\pm 0.42}$ §	$5.43_{\pm 0.29}$ §	$77.86_{\pm 0.25}$	$4.33_{\pm 0.24}$	$79.47_{\pm 0.35}$ §	$6.49_{\pm 0.43}$ §
CPG [17]	$90.63_{\pm 0.44}$	$3.98_{\pm 0.65}$	$88.08_{\pm 0.77}$	$5.20_{\pm 0.64}$	$78.63_{\pm 0.52}$	$7.18_{\pm 0.62}$	$83.21_{\pm 0.67}$	$7.09_{\pm 0.82}$
NSP^2 [12]	$90.70_{\pm 0.17}$	$4.20_{\pm 0.35}$	$88.59_{\pm 0.46}$	$4.45_{\pm 0.60}$	$79.17_{\pm 0.63}$	$5.06_{\pm 1.07}$	$84.17_{\pm 0.54}$	$7.33_{\pm 0.94}$
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**$_{\pm 0.26}$

100 [23], where CIFAR-100 is randomly partitioned into 10 or 20 tasks (10 or 5 classes per task); (ii) 10-Split ImageNet-R [24], which is split into 10 tasks (20 classes per task); and (iii) 10-Split DomainNet [25], where the 200 largest classes are selected and split into 10 tasks (20 classes per task). We further consider a domain-incremental setting, 6-Split DomainNet, in which each task corresponds to one of distinct six domains.

Evaluation Metrics. We report Last Average Accuracy (ACC, ↑) and Forgetting (FOR, ↓). ACC measures the overall performance on all tasks after completing the training sequence: $\mathrm{ACC} = \frac{1}{T}\sum_{j=1}^{T} A_{T,j}$. FOR quantifies the average performance drop on past tasks: $\mathrm{FOR} = \frac{1}{T-1}\sum_{j=1}^{T-1} \max_{i\in\{1,\dots,T-1\}}(A_{i,j} - A_{T,j})$.

Implementation Details. We follow the rehearsal-free class-incremental setting with disjoint classes and unknown task identity at inference. Experiments use ImageNet-21K pretrained ViT-B/16 [4] and AugReg ViT-L/16 [27]. Each block is equipped with four learnable prompts, shared across tasks and initialized uniformly in $[-1, 1]$. Models are trained for 10 epochs per task using Adam [28] (lr=0.01, batch size = 256, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay = 5×10^{-5}). NSP^2 is reproduced from its official codebase and hyperparameters. The training objective combines cross-entropy and standard-deviation alignment loss (28).

B. Results and Analysis

Overall Comparison on ImageNet-21K ViT-B/16. As shown in Table I, CNSP consistently outperforms prior prompt-based CL methods on all benchmarks using the ImageNet-21K pretrained ViT-B/16 backbone. On 10-Split DomainNet, for example, CNSP achieves a 9.7% relative reduction in FOR (7.33% → 6.62%) while also improving accuracy from 84.17% to 84.51% over NSP^2, indicating that the proposed matrix-form analysis and right-side projection yields measurable gains even in accuracy-saturated regimes.

Scaling to Larger Backbone. We further evaluate CNSP on the larger ViT-L/16 (AugReg) [27], with results in Table II. CNSP consistently outperforms NSP^2. The improvement is especially pronounced on ViT-L/16, where approximation errors accumulate in high-dimensional token spaces. For instance, on 10-Split ImageNet-R, CNSP reduces FOR from 6.50% to 4.29% (34% relative) while improving ACC from 83.52% to 84.78%. Overall, these results demonstrate that CNSP scales robustly to larger ViT backbones.

TABLE II: Results on four benchmarks with ViT-L/16 (AugReg, IN-21k) backbone.

Method	10S-ImageNet-R		10S-DomainNet		10S-CIFAR100		20S-CIFAR100	
	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
Joint	88.58	0.00	90.48	0.00	95.06	0.00	95.06	0.00
Sequential	79.92	15.51	80.85	15.88	88.91	10.11	87.40	11.32
NSP^2	83.52	6.50	84.38	10.28	92.20	4.14	90.83	6.34
CNSP	**84.78**	**4.29**	**85.63**	**8.03**	**92.45**	**4.01**	**91.02**	**5.24**

Robustness to Domain Shifts. To evaluate robustness under task heterogeneity, we test CNSP on 6-Split DomainNet, where each task corresponds to a distinct visual domain. As shown in Table III, FOR decreases from 6.67% to 5.84% on run 1 (a 12% relative reduction) with an ACC gain (83.55% → 84.32%). These results indicate that CNSP maintains stable performance under substantial domain shifts without relying on assumptions about invariant prompt distributions.

TABLE III: Results on the 6-Split DomainNet benchmark across four runs with ViT-B/16-21K.

Method	Run 1		Run 2		Run 3		Run 4	
	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
Joint	91.68	0.00	91.94	0.00	91.88	0.00	91.24	0.00
Sequential	63.34	34.11	79.17	18.52	78.22	17.65	67.55	28.49
NSP^2	83.55	6.67	84.85	7.91	84.76	3.36	83.61	6.85
CNSP	**84.32**	**5.84**	**85.34**	**6.44**	**85.28**	**3.07**	**84.88**	**5.77**

Analysis of Prompt Expressive Capacity. To assess whether null-space projection restricts prompt expressiveness as tasks accumulate, we track the update subspace by the rank of projection matrices $\{\mathbf{B}_t^{(l)}\}_{l=1}^{L}$, which determine allowable prompt-update directions. We quantify this via the average

rank ratio $\rho_t = \frac{1}{L}\sum_{l=1}^{L}\frac{\operatorname{rank}\left(\mathbf{B}_t^{(l)}\right)}{D}$, where higher ρ_t indicates a less constrained update space. As shown in Table IV, ρ_t remains above 97.52% across benchmarks and does not decrease over tasks, indicating that CNSP preserves prompt expressiveness by removing only feature-violating directions.

TABLE IV: Average rank ratio (%) of projection matrices across layers for each task.

Benchmark	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8	ρ_9	ρ_{10}
10S-ImageNet-R	97.53	97.53	97.53	97.54	97.53	97.53	97.54	97.53	97.53
10S-CIFAR100	97.53	97.54	97.54	97.54	97.54	97.54	97.54	97.55	97.55
10S-DomainNet	97.52	97.54	97.53	97.53	97.53	97.54	97.53	97.53	97.53

VI. Conclusion

In this work, we introduced a principled framework that provides an end-to-end theoretical formulation for prompt-based continual learning. By modeling MHSA, LayerNorm, and classification heads in an exact matrix form, we derived unified sufficient conditions for jointly preserving features and classifier decisions across tasks. Based on these conditions, we proposed our algorithm CNSP. Extensive experiments across diverse benchmarks and Transformer backbones show that CNSP consistently improves accuracy and mitigates catastrophic forgetting. Looking forward, this framework opens several promising directions, including theoretically grounded head designs, extensions to multimodal continual learning, adaptive prompt allocation, and tighter analyses of nonlinear components.

Acknowledgment

This work was partially supported by the Shenzhen Research Project (No. KCXST20221021111210023), the Shenzhen Science and Technology Program (No. ZDCY20250901103002003), and the Natural Science Foundation of China (NSFC) under Grants 62473357 and 92473112.

References

[1] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[2] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 8, pp. 5362–5383, 2024.

[3] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim, "Visual prompt tuning," in *European conference on computer vision*. Springer, 2022, pp. 709–727.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[5] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister, "Learning to prompt for continual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 139–149.

[6] Z. Wang, Z. Zhang, S. Ebrahimi, R. Sun, H. Zhang, C.-Y. Lee, X. Ren, G. Su, V. Perot, J. Dy *et al.*, "Dualprompt: Complementary prompting for rehearsal-free continual learning," in *European conference on computer vision*, 2022, pp. 631–648.

[7] J. S. Smith, L. Karlinsky, V. Gutta, P. Cascante-Bonilla, D. Kim, A. Arbelle, R. Panda, R. Feris, and Z. 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*, 2023, pp. 11 909–11 919.

[8] G. Saha, I. Garg, and K. Roy, "Gradient projection memory for continual learning," *CoRR*, vol. abs/2103.09762, 2021. [Online]. Available: https://arxiv.org/abs/2103.09762

[9] D. Deng, G. Chen, J. Hao, Q. Wang, and P.-A. Heng, "Flattening sharpness for dynamic gradient projection memory benefits continual learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 710–18 721, 2021.

[10] S. Wang, X. Li, J. Sun, and Z. 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*, 2021, pp. 184–193.

[11] J. Qiao, zhizhong zhang, X. Tan, C. Chen, Y. Qu, Y. Peng, and Y. Xie, "Prompt gradient projection for continual learning," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=EH2O3h7sBI

[12] Y. Lu, S. Zhang, D. Cheng, Y. Xing, N. Wang, P. Wang, and Y. Zhang, "Visual prompt tuning in null space for continual learning," *Advances in neural information processing systems*, vol. 37, pp. 7878–7901, 2024.

[13] Z. Gao, J. Cen, and X. Chang, "Consistent prompting for rehearsal-free continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 28 463–28 473.

[14] M. R. Kurniawan, X. Song, Z. Ma, Y. He, Y. Gong, Y. Qi, and X. Wei, "Evolving parameterized prompt memory for continual learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 12, 2024, pp. 13 301–13 309.

[15] M. G. Z. A. Khan, M. F. Naeem, L. Van Gool, D. Stricker, F. Tombari, and M. Z. Afzal, "Introducing language guidance in prompt-based continual learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 11 463–11 473.

[16] A. Roy, R. Moulick, V. K. Verma, S. Ghosh, and A. Das, "Convolutional prompting meets language models for continual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 23 616–23 626.

[17] Y. Lu, S. Zhang, D. Cheng, G. Liang, Y. Xing, N. Wang, and Y. Zhang, "Training consistent mixture-of-experts-based prompt generator for continual learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 18, 2025, pp. 19 152–19 160.

[18] G. Zeng, Y. Chen, B. Cui, and S. Yu, "Continual learning of context-dependent processing in neural networks," *Nature Machine Intelligence*, vol. 1, no. 8, pp. 364–372, 2019.

[19] S. Lin, L. Yang, D. Fan, and J. Zhang, "Trgp: Trust region gradient projection for continual learning," *arXiv preprint arXiv:2202.02931*, 2022.

[20] G. Lin, H. Chu, and H. 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*, 2022, pp. 89–98.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[23] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[24] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. 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*, 2021, pp. 8340–8349.

[25] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1406–1415.

[26] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[27] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your vit? data, augmentation, and regularization in vision transformers," *arXiv preprint arXiv:2106.10270*, 2021.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

APPENDIX

Proof of Lemma 1. Expanding the LayerNorm operations gives

$$\mathrm{LN}(\mathbf{A}) = \frac{\mathbf{A} - \boldsymbol{\mu}_{\mathbf{A}}}{\boldsymbol{\sigma}_{\mathbf{A}}} \cdot \boldsymbol{\gamma} + \boldsymbol{\beta}, \tag{31}$$

where $\boldsymbol{\mu}_{\mathbf{A}}, \boldsymbol{\sigma}_{\mathbf{A}} \in \mathbb{R}^{n+m}$, $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^d$, and broadcasting is used to match the dimensions. By matrix reformulation of the broadcast operation, let $\mathbf{C} = \mathbf{I}_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top \in \mathbb{R}^{d\times d}$, $\mathbf{D}_{\boldsymbol{\sigma}_{\mathbf{A}}} = \mathrm{diag}(\boldsymbol{\sigma}_{\mathbf{A}}) \in \mathbb{R}^{(n+m)\times(n+m)}$, and $\boldsymbol{\Gamma} = \mathrm{diag}(\boldsymbol{\gamma}) \in \mathbb{R}^{d\times d}$. Then[4]

$$\mathrm{LN}(\mathbf{A}) = \mathbf{D}_{\sigma_{\mathbf{A}}}^{-1}\mathbf{A}\mathbf{C}\boldsymbol{\Gamma} + \mathbf{1}_{n+m}\boldsymbol{\beta}^\top. \tag{32}$$

Since $\boldsymbol{\sigma}_{\mathbf{A}}$ corresponds row-wise to $\mathbf{A}$, we may write

$$\boldsymbol{\sigma}_{\mathbf{A}} = \begin{bmatrix} \boldsymbol{\sigma}_{\mathbf{A}^{(1)}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\sigma}_{\mathbf{A}^{(2)}} \end{bmatrix}, \ \mathbf{D}_{\boldsymbol{\sigma}_{\mathbf{A}}} = \begin{bmatrix} \mathbf{D}_{\boldsymbol{\sigma}_{\mathbf{A}^{(1)}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\boldsymbol{\sigma}_{\mathbf{A}^{(2)}}} \end{bmatrix}. \tag{33}$$

Therefore,

$$\mathrm{LN}(\mathbf{A}) = \begin{bmatrix} \mathbf{D}_{\boldsymbol{\sigma}_{\mathbf{A}^{(1)}}}^{-1}\mathbf{A}^{(1)}\mathbf{C}\boldsymbol{\Gamma} + \mathbf{1}_n\boldsymbol{\beta}^\top \\ \mathbf{D}_{\boldsymbol{\sigma}_{\mathbf{A}^{(2)}}}^{-1}\mathbf{A}^{(2)}\mathbf{C}\boldsymbol{\Gamma} + \mathbf{1}_m\boldsymbol{\beta}^\top \end{bmatrix} = \begin{bmatrix} \mathrm{LN}\left(\mathbf{A}^{(1)}\right) \\ \mathrm{LN}\left(\mathbf{A}^{(2)}\right) \end{bmatrix}. \tag{34}$$

□

Lemma 3. *Let* $\mathrm{MLP}(\cdot)$ *denote the two-layer GELU MLP in ViT [4]. For any row-partitioned matrix* $\mathbf{A} = \left[\mathbf{A}^{(1)}; \mathbf{A}^{(2)}\right]$, *where* $\mathbf{A}^{(1)} \in \mathbb{R}^{n\times d}$ *and* $\mathbf{A}^{(2)} \in \mathbb{R}^{m\times d}$, *we have*

$$\mathrm{MLP}(\mathbf{A}) = \begin{bmatrix} \mathrm{MLP}\left(\mathbf{A}^{(1)}\right) \\ \mathrm{MLP}\left(\mathbf{A}^{(2)}\right) \end{bmatrix} \tag{35}$$

Proof. By definition,

$$\mathrm{MLP}(\mathbf{A}) = \phi\left(\mathbf{A}\mathbf{W}_1 + \mathbf{1}_{n+m}\mathbf{b}_1^\top\right)\mathbf{W}_2 + \mathbf{1}_{n+m}\mathbf{b}_2^\top, \tag{36}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times d_h}$, $\mathbf{W}_2 \in \mathbb{R}^{d_h\times d}$ are weight matrices of two linear projections respectively, $\mathbf{b}_1 \in \mathbb{R}^{d_h}$, $\mathbf{b}_2 \in \mathbb{R}^d$ 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,

$$\begin{aligned} \phi\left(\mathbf{A}\mathbf{W}_1 + \mathbf{1}_{n+m}\mathbf{b}_1^\top\right) &= \phi\left(\begin{bmatrix} \mathbf{A}^{(1)} \\ \mathbf{A}^{(2)} \end{bmatrix}\mathbf{W}_1 + \begin{bmatrix} \mathbf{1}_n \\ \mathbf{1}_m \end{bmatrix}\boldsymbol{\beta}_1^\top\right) \\ &= \begin{bmatrix} \phi\left(\mathbf{A}^{(1)}\mathbf{W}_1 + \mathbf{1}_n\mathbf{b}_1^\top\right) \\ \phi\left(\mathbf{A}^{(2)}\mathbf{W}_1 + \mathbf{1}_m\mathbf{b}_1^\top\right) \end{bmatrix}. \end{aligned} \tag{37}$$

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

$$\begin{aligned} &\phi\left(\mathbf{A}\mathbf{W}_1 + \mathbf{1}_{n+m}\mathbf{b}_1^\top\right)\mathbf{W}_2 + \mathbf{1}_{n+m}\mathbf{b}_2^\top \\ &= \begin{bmatrix} \phi\left(\mathbf{A}^{(1)}\mathbf{W}_1 + \mathbf{1}_n\mathbf{b}_1^\top\right)\mathbf{W}_2 + \mathbf{1}_n\mathbf{b}_2^\top \\ \phi\left(\mathbf{A}^{(2)}\mathbf{W}_1 + \mathbf{1}_m\mathbf{b}_1^\top\right)\mathbf{W}_2 + \mathbf{1}_m\mathbf{b}_2^\top \end{bmatrix}, \end{aligned} \tag{38}$$

which implies that

$$\mathrm{MLP}(\mathbf{A}) = \begin{bmatrix} \mathrm{MLP}\left(\mathbf{A}^{(1)}\right) \\ \mathrm{MLP}\left(\mathbf{A}^{(2)}\right) \end{bmatrix}. \tag{39}$$

□

Proof of Proposition 1. 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, $\mathbf{Z}^{(l)}$ is added to the output $\mathbf{O}_{\mathbf{Z}}^{(l)}$ via a residual connection, followed by LayerNorm and an MLP with an additional residual connection:

$$\mathbf{B}_{\mathbf{Z}}^{(l)} = \mathbf{Z}^{(l)} + \mathbf{O}_{\mathbf{Z}}^{(l)}, \quad \mathbf{E}_{\mathbf{Z}}^{(l)} = \mathbf{B}_{\mathbf{Z}}^{(l)} + \mathrm{MLP}\left(\mathrm{LN}\left(\mathbf{B}_{\mathbf{Z}}^{(l)}\right)\right). \tag{40}$$

Since

$$\mathbf{E}_{\mathbf{Z}}^{(l)}[:N+1] = \mathbf{B}_{\mathbf{Z}}^{(l)}[:N+1] + \mathrm{MLP}\left(\mathrm{LN}\left(\mathbf{B}_{\mathbf{Z}}^{(l)}\right)\right)[:N+1], \tag{41}$$

by Lemmas 1 and 3, we obtain

$$\mathrm{MLP}\left(\mathrm{LN}\left(\mathbf{B}_{\mathbf{Z}}^{(l)}\right)\right)[:N+1] = \mathrm{MLP}\left(\mathrm{LN}\left(\mathbf{B}_{\mathbf{Z}}^{(l)}[:N+1]\right)\right). \tag{42}$$

Moreover,

$$\begin{aligned} \mathbf{B}_{\mathbf{Z}}^{(l)}[:N+1] &= \mathbf{Z}^{(l)}[:N+1] + \mathbf{O}_{\mathbf{Z}}^{(l)}[:N+1] \\ &= \mathbf{X}^{(l)} + \mathbf{O}_{\mathbf{Z}}^{(l)}[:N+1], \end{aligned} \tag{43}$$

$$\begin{aligned} \mathbf{E}_{\mathbf{Z}}^{(l)}[:N+1] &= \mathbf{X}^{(l)} + \mathbf{O}_{\mathbf{Z}}^{(l)}[:N+1] \\ &\quad + \mathrm{MLP}\left(\mathrm{LN}\left(\mathbf{X}^{(l)} + \mathbf{O}_{\mathbf{Z}}^{(l)}[:N+1]\right)\right). \end{aligned} \tag{44}$$

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

$$\mathbf{X}^{(l+1)} = \mathbf{E}_{\mathbf{Z}}^{(l)}[:N+1], \quad \mathbf{Z}^{(l+1)} = [\mathbf{X}^{(l+1)}; \mathbf{P}^{(l+1)}]. \tag{45}$$

Consider 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

$$\mathbf{O}_{\mathbf{Z}_{t,t+1}}^{(1)}[:N+1] = \mathbf{O}_{\mathbf{Z}_{t,t}}^{(1)}[:N+1], \ \mathbf{X}_{t,t+1}^{(1)} = \mathbf{X}_{t,t}^{(1)} = \mathbf{X}_t, \tag{46}$$

where $\mathbf{X}_t$ is the input image tokens of VPT from task $\mathcal{T}_t$ dataset. According to (44), it follows that

$$\mathbf{E}_{\mathbf{Z}_{t,t+1}}^{(1)}[:N+1] = \mathbf{E}_{\mathbf{Z}_{t,t}}^{(l)}[:N+1], \quad \mathbf{X}_{t,t+1}^{(2)} = \mathbf{X}_{t,t}^{(2)}. \tag{47}$$

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

$$\mathbf{E}_{\mathbf{Z}_{t,t+1}}^{(k)}[:N+1] = \mathbf{E}_{\mathbf{Z}_{t,t}}^{(k)}[:N+1], \tag{48}$$

which implies

$$\mathbf{X}_{t,t+1}^{(k+1)} = \mathbf{X}_{t,t}^{(k+1)}. \tag{49}$$

For $l = k+1$, we know

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

By (44) and (49), it follows that

$$\mathbf{E}_{\mathbf{Z}_{t,t+1}}^{(k+1)}[:N+1] = \mathbf{E}_{\mathbf{Z}_{t,t}}^{(k+1)}[:N+1]. \tag{51}$$

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

$$\mathbf{E}_{\mathbf{Z}_{t,t+1}}^{(l)}[:N+1] = \mathbf{E}_{\mathbf{Z}_{t,t}}^{(l)}[:N+1]. \tag{52}$$

□

[4] In practice, the standard deviation of the i-th row is computed as $\sqrt{\frac{1}{d}\mathrm{Var}(\mathbf{A}[i]) + \epsilon}$, where $\mathrm{Var}(\cdot)$ is the operation to compute variance, $\epsilon > 0$ ensures non-degeneracy. Hence, $\mathbf{D}_{\boldsymbol{\sigma}_{\boldsymbol{A}}}$ is invertible.