A Appendix

This supplementary material provides in-depth information on the following topics:

- More experiments and ablation studies.
- Impact of mini-batch variability on network modularization.
- Architecture of Swin-Nano and Swin-Pico.
- Optimization.
- Related works - Architectures.
- Principal Component Analysis (PCA).
- Principal Component Analysis for Mean Variance Computation.

Each section offers detailed insights into the respective topic for a comprehensive understanding.

B More Experiments and Ablation Studies

B.1 Knowledge Transfer in Consistent Architectures.

We employ the widely recognized pairing of a ResNet34 teacher and a ResNet18 student on the ImageNet-1K dataset to further showcase the efficacy of our method within consistent architectures. As depicted in Table 4, the PKD technique yields results comparable to the top-performing distillation baseline.

	T.	S.	KD	OFD	Review	CRD	DKD	DIST	OFA	PKD (ours)
Accuracy	73.62	69.90	70.66	70.81	71.61	71.17	71.70	72.07	72.10	73.51 ± 0.18

Table 4: KD methods with homogeneous architectures on ImageNet-1K. *T*: ResNet34, *S*: ResNet18.

Teacher	T.	S.(ResNet50)	RKD	Review	CRD	DKD	DIST	OFA	PKD (ours)
ResNet152	82.83	79.86	79.53	80.06	79.33	80.49	80.55	80.64	82.75 ± 0.30
ViT-B	86.53	79.86	79.38	79.32	79.48	80.76	80.90	81.33	84.67 ± 0.22

Table 5: Comparison of homogeneous and heterogeneous teacher on ImageNet-1K.

B.2 Consistent *vs.* Diverse Teacher Models

To evaluate the influence of employing a larger diverse teacher model, we train a ResNet50 student with both a ResNet152 teacher (consistent architecture) and a ViT-B teacher (diverse architecture). As illustrated in Table 5, our PKD approach achieves a significant improvement in performance when utilizing the ViT-B teacher compared to the ResNet152 teacher. This finding underscores the importance of architecture-agnostic knowledge distillation (KD) in striving for enhanced performance gains.

B.3 KD with Last Layer Module-Specific Features

In this experiment, we utilize the OFA method to examine the impact of using only module-specific features from the last layer, specifically extracting the most important features from this layer. For modularization, we divided the student and teacher networks into four modules, as outlined in the OFA method by (Hao et al., 2024). Initially, we trained the first two modules by leveraging the most important features while keeping the last two modules (the deeper layers) frozen. The rationale is that the shallow layers of the network learn the most critical features and semantics (class-specific features).

After training the first two modules, we proceeded to train the deeper modules using data where the indices corresponding to the most important features were masked by zero. The experimental results, presented in Table 6 labeled as OFA+MS, demonstrate an improvement over the OFA method, highlighting the significance of utilizing the appropriate features to train different layers or modules of the network.

B.4 Enhanced DKD Performance with PKD

We conduct further experiments to assess the effectiveness of our PKD method, particularly the progressive modular alignment, on other knowledge distillation techniques. Specifically, we integrate our method with DKD. In this setup, we adhere to the primary DKD methodology and incorporate modularization along with progressive modular training. The experimental outcomes are presented as (DKD+PKD) in Table 6, highlighted as gray. As shown in the table, our approach enhances the performance of DKD.

Teacher	Student	From Scratch		Logits-based						
		T.	S.	KD	DKD	DKD +PKD	DIST	OFA	OFA+MS	PKD
CNN-based students										
DeiT-T	ResNet18	72.17	69.75	70.22	69.39	73.56	70.64	71.34	72.11	71.97 ± 0.23
Swin-T	ResNet18	81.38	69.75	71.14	71.10	78.46	70.91	71.85	74.09	78.31 ± 0.31
DeiT-T	MobileNetV2	72.17	68.87	70.87	70.14	71.28	71.08	71.39	71.63	72.00 ± 0.17
ViT-based students										
ResNet50	DeiT-T	80.38	72.17	75.10	75.60	78.02	75.13	76.55	77.23	78.08 ± 0.19
ConvNeXt-T	DeiT-T	82.05	72.17	74.00	73.95	75.94	74.07	74.41	74.83	77.62 ± 0.10
MLP-based students										
ResNet50	ResMLP-S12	80.38	76.65	77.41	78.23	79.61	77.71	78.53	79.17	79.83 ± 0.32
ConvNeXt-T	ResMLP-S12	82.05	76.65	76.84	77.23	80.38	77.24	77.53	76.68	80.29 ± 0.20

Table 6: Applying our PKD method on top of the DKD method improves performance on ImageNet-1K. Applying MS procedures on OFA last layer , OFA+MS, improves its performance.

B.5 Impact of the Threshold in Equation 5

To better understand the impact of the threshold parameter ϵ_{th} in Equation 5 on performance, we conducted an ablation study. This experiment systematically evaluated different values of ϵ_{th} and their effect on the accuracy of our method. The results of this ablation are presented in Table 7, where we report the performance of the PKD model with varying threshold values.

As shown in Table 7, our method consistently outperforms the OFA baseline across all tested thresholds. Notably, ϵ_{th} values of $1e-4$ and $5e-4$ yield the best performance, suggesting that fine-tuning the threshold in this range is critical for maximizing model accuracy. These results validate the robustness of our approach and demonstrate the importance of selecting an appropriate threshold value for optimal knowledge distillation performance.

C Impact of Mini-batch Variability on Network Modularization

In our PKD framework, network modularization is not based on individual mini-batches, but rather on a subset of the data to manage computational costs efficiently. Specifically, we randomly select a limited number of samples per class (e.g., 25) for CKA (Centered Kernel Alignment) computations. Through extensive experimentation, we found that this approach produces modularization results comparable to using the entire dataset.

To account for variability when considering all training data, we compute the average CKA score across mini-batches. This ensures consistency in the modularization process while maintaining computational efficiency, as the use of a subset of data for CKA calculation reduces the overall computational burden without sacrificing accuracy in the modularization results.

Teacher	Student	PKD considering different ϵ_{th}							OFA
		$2e-5$	$5e-5$	$1e-4$	$5e-4$	$2e-3$	$5e-3$	$1e-3$	OFA
DeiT-T	ResNet18	71.91	72.04	71.97	71.92	71.86	71.80	71.81	71.34
Swin-T	ResNet18	78.00	78.03	78.31	78.26	78.17	78.14	78.09	71.85
ResNet50	DeiT-T	77.86	77.99	78.08	78.03	77.91	77.83	77.80	76.55

Table 7: Performance comparison of PKD using different threshold values ϵ_{th} in Equation 5, with varying teacher-student model pairs on the ImageNet dataset. Our method consistently outperforms the OFA baseline across all tested values.

D Architecture of Swin-Nano and Swin-Pico

To ensure that the teacher models surpass the performance of the student model, (Hao et al., 2024) presented two modified versions of Swin-Tiny (Liu et al., 2021), named Swin-Nano and Swin-Pico. Swin-Nano features an embedding dimension of 64, while Swin-Pico has an embedding dimension of 48, in contrast to the original Swin-Tiny's embedding dimension of 96. Additionally, Swin-Tiny has layer depths of (2, 2, 6, 2) and numbers of heads of (3, 6, 12, 24), whereas the two modified models share the same configurations for depths and numbers of heads, which are (2, 2, 2, 2) and (2, 4, 8, 16), respectively.

E Optimization

For training models with diverse architectures on the ImageNet-1K and CIFAR-100 datasets, we employ distinct optimization settings. The comprehensive settings are provided in Table 8.

	ImageNet-1k		CIFAR-100	
	CNN	ViT/ MLP	CNN	ViT/MLP
Epochs	100	300	300	300
Batch size	512	1024	300	300
Initial LR	0.1	5e-4	5e-2	5e-4
Minimum LR	1e-6	1e-6	1e-3	1e-5
Optimizer	SGD	AdamW	SGD	AdamW
Weight decay	1e-4	5e-2	2e-3	5e-2
LR schedule	×0.1 at [30,60,90]	Cosine	Cosine	Cosine
Warmup	3	20	3	20
EMA	-	0.99996	-	-
RandAugment	-	9/0.5	-	9/0.5
Mixup	-	0.8	-	0.8
Cutmix	-	1.0	-	1.0
RE prob	-	0.25	-	0.25

Table 8: Optimization settings details.

F Related works- Architectures

In recent years, significant advancements have been made in the evolution of model architectures for computer vision tasks. This section offers a succinct overview of two notable architectures: Transformer and MLP.

Vision Transformer Vaswani and colleagues (Vaswani et al., 2017) initially introduced the transformer architecture for tasks in natural language processing (NLP). Due to the utilization of the attention mechanism, this framework adeptly captures prolonged dependencies and attains remarkable performance. Motivated by its considerable success, endeavors have been made to devise transformer-based models for computer vision (CV) tasks. (Dosovitskiy et al., 2021) partition an image into non-overlapping patches and map these patches into embedding tokens. Subsequently, these tokens undergo processing by the transformer model akin to NLP tasks. Their design achieves state-of-the-art performance and stimulates the creation of a sequence of subsequent architectures.

MLP For an extended period, MLP has exhibited inferior performance compared to CNN in the domain of computer vision. To explore the potential of MLP, (Tolstikhin et al., 2021) proposed MLP-Mixer, exclusively based on the MLP structure. MLP-Mixer takes embedding tokens of an image patch as input and interleaves channel and spatial information mixing at each layer. This architecture performs comparably to the leading CNN and ViT models. Touvron and colleagues (Touvron et al., 2021) proposed another MLP architecture termed ResMLP.

The most advanced CNN, Transformer, and MLP models achieve analogous performance. Nonetheless, these architectures possess distinct inductive biases, leading to disparate preferences in representation learning. Generally, *noticeable distinctions exist between features acquired by diverse architectures*.

G Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical method aimed at reducing the dimensionality of high-dimensional data while preserving as much variance as possible. This is accomplished by identifying principal components, which are orthogonal vectors that indicate the directions of maximum variance (Shlens, 2014; Jolliffe, 2002).

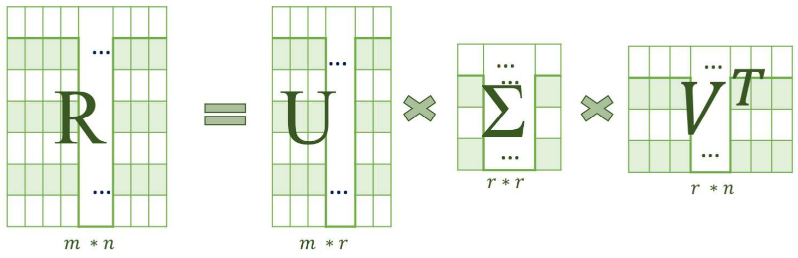

Figure 4: Singular Value Decomposition

G.1 Procedure

This section details the steps involved in PCA to extract the most informative features.

Data Standardization: Standardize the dataset features to have a mean of zero and a unit variance.

Covariance Matrix: Calculate the covariance matrix of the standardized data, which captures the relationships between different features.

SVD of Covariance Matrix: Perform Singular Value Decomposition (SVD) on the covariance matrix. The SVD of the covariance matrix yields the principal components.

Selecting Principal Components: Sort the singular values in descending order. The corresponding singular vectors are the principal components. Select the top k principal components to form a reduced-dimensional space.

Projection: Project the original data onto the selected principal components to obtain the lower-dimensional representation.

Benefits: - Dimensionality reduction simplifies data visualization and interpretation. - Reduced dimensionality often enhances computational efficiency. - Principal components encapsulate the most significant patterns in the data.

G.2 More Explanation Regarding SVD Computation

Consider an $m \times n$ matrix R, where m denotes the number of rows and n represents the number of columns. The primary objective of Singular Value Decomposition (SVD) is to decompose matrix R into three distinct matrices: U, Σ, and V^T (the transpose of matrix V). This decomposition is expressed as $R = U\Sigma V^T \in \mathbb{R}^{m\times n}$, as illustrated in Figure 4.

- U: An $m \times m$ orthogonal matrix, whose columns are the left singular vectors of R. - Σ: An $m \times n$ diagonal matrix containing the singular values of R (non-negative and arranged in descending order). - V^T: An $n \times n$ orthogonal matrix, with columns representing the right singular vectors of R.

G.3 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors are also fundamental to understanding matrix properties. An eigenvalue λ and its corresponding eigenvector $\mathbf{v}$ of a square matrix R satisfy the equation $R\mathbf{v} = \lambda\mathbf{v}$. Eigenvectors denote directions in the vector space that are scaled by the matrix R, while eigenvalues represent the scaling factors for these eigenvectors.

G.4 SVD and Its Relationship to Eigenvalues and Eigenvectors

SVD establishes a crucial relationship between eigenvalues and eigenvectors and the singular values and singular vectors of a matrix. The singular values of R are the square roots of the eigenvalues of either RR^T or R^TR, and the left and right singular vectors are the eigenvectors of RR^T and R^TR, respectively.

G.5 Rank and Matrix Approximation

The rank of a matrix R is determined by the number of non-zero singular values in Σ. By retaining only the largest singular values and their corresponding singular vectors, it is possible to approximate the original matrix R with a lower-rank approximation. This technique is valuable for tasks such as dimensionality reduction and noise reduction, and it is utilized in our approach.

G.6 Properties of SVD

- The singular values in Σ are non-negative and arranged in descending order. - The columns of U and V are orthonormal, forming an orthogonal basis for their respective vector spaces. - The SVD decomposition is unique, except for the sign of the singular values and the order of the singular vectors.

SVD is a powerful matrix factorization technique, offering a concise representation of a matrix while preserving essential structural properties. Its applications span various fields, including data analysis, image processing, recommendation systems, and more (Deisenroth et al., 2020).

H Principal Component Analysis for Mean Variance Computation

In this section, we outline the procedure for computing the mean variance of high-dimensional data features using Principal Component Analysis (PCA). Consider a dataset $X \in \mathbb{R}^{n\times d}$, where n is the number of data samples and d is the number of features.

H.1 Standardization of Features

PCA is sensitive to the scale of the input data, so we begin by standardizing the features. This ensures that each feature has zero mean and unit variance.

Let the dataset $X = \{X_1, X_2, \dots, X_n\}$, where each $X_i \in \mathbb{R}^d$, represent the set of n data samples, each having d features. The standardized data matrix $X_{\text{standardized}}$ is computed as:

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} X_{ij}, \quad \forall j = 1, 2, \dots, d, \tag{3}$$

$$X_{\text{centered}} = X - \mu, \tag{4}$$

$$X_{\text{standardized}} = \frac{X_{\text{centered}}}{\sigma}, \quad \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_{ij} - \mu_j)^2}. \tag{5}$$

Here, μ_j represents the mean of the j-th feature, and σ_j is the standard deviation of the j-th feature.

H.2 Covariance Matrix Computation

Once the data is standardized, the covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ can be computed to measure the pairwise dependencies between features. The covariance matrix is defined as:

$$\Sigma = \frac{1}{n-1} X_{\text{standardized}}^{\top} X_{\text{standardized}}, \tag{6}$$

where Σ_{jk} represents the covariance between the j-th and k-th features.

H.3 Eigenvalue Decomposition of the Covariance Matrix

We perform eigenvalue decomposition on the covariance matrix Σ, which gives us the principal components and the amount of variance explained by each. The decomposition is expressed as:

$$\Sigma = V \Lambda V^\top, \tag{7}$$

where $V \in \mathbb{R}^{d\times d}$ is the matrix of eigenvectors (principal components) and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d) \in \mathbb{R}^{d\times d}$ is the diagonal matrix of eigenvalues λ_j, where λ_j corresponds to the variance explained by the j-th principal component.

H.4 Explained Variance

The eigenvalues λ_j provide the variance explained by each corresponding principal component. The proportion of variance explained by the j-th principal component is computed as:

$$\text{Explained Variance Ratio} = \frac{\lambda_j}{\sum_{k=1}^{d} \lambda_k}. \tag{8}$$

H.5 Mean Variance Explained

The mean variance explained by the principal components can be computed by averaging the explained variance ratio across all components:

$$\text{Mean Variance} = \frac{1}{d} \sum_{j=1}^{d} \frac{\lambda_j}{\sum_{k=1}^{d} \lambda_k}. \tag{9}$$

This metric represents the average amount of variance explained by each principal component in the transformed space.