Learning Orthonormal Features in Self-Supervised Learning using Functional Maximal Correlation

Bo Hu, Yuheng Bu, José C. Príncipe
Department of Electrical and Computer Engineering
University of Florida
{hubo, buyuheng}@ufl.edu principe@cnel.ufl.edu

Abstract

This paper applies statistical dependence measures to interpret self-supervised learning (SSL). Conventional applications of measures like mutual information commonly use separate procedures for feature extraction and dependence estimation, where the relationship between optimal features and the strength of dependence is unclear. This causes limitations in tasks requiring multivariate feature representations, particularly in SSL. The recently introduced multivariate measure, functional maximal correlation, is a unified framework based on orthonormal decomposition of density ratios, wherein the spectrum and the bases become the measure and the features, respectively. This paper proposes that features in SSL can also be interpreted as basis functions of the density ratio. We introduce the Hierarchical Functional Maximal Correlation Algorithm (HFMCA), a theoretically justified approach that ensures faster convergence, enhanced stability, and prevents feature collapse by learning orthonormal bases as multivariate features.

1 Introduction

Measures of statistical dependence have been instrumental in learning features that maximize information transfer ([1, 2, 3, 4]), which has sparked a multitude of learning principles and algorithms in machine learning ([5, 6, 7, 8, 9]). Improving the diversity and interpretability of multivariate features is important in numerous machine learning tasks, such as one-shot learning, transfer learning, and especially self-supervised learning (SSL) [10, 11, 12]. In this paper, we explore the advances of using multivariate statistical dependence measures for these tasks.

Conventional dependence measures such as mutual information (MI) may encounter limitations in tasks that require multivariate properties. Methods based on these measures ([6, 13, 14, 15, 16, 17, 18]) typically involve a three-step iterative process of projection, estimation, and maximization: first, a network maps data to a feature space; second, a mutual information estimator is optimized to ensure tight variational bounds; and third, the feature extractor is optimized to maximize the estimated MI. The limitation lies in the gap between feature extraction and dependence estimation being two separate procedures, conducted by two separate models, which leaves the link between the optimal multivariate features and the strength of dependence obscure.

The Functional Maximal Correlation (FMC) is a recently introduced multivariate dependence measure based on the concept of orthonormal decomposition of density ratios [19, 20, 21]. The spectrum in this decomposition is the defined dependence measure, and the basis functions are the multivariate features. Together, they create the projection space associated with the density ratio. The measure is accompanied with the Functional Maximal Correlation Algorithm (FMCA), which uses neural networks and log-determinant costs to learn this decomposition directly from empirical data. This framework unifies dependence measurement and feature learning through density ratio decomposition, allowing the learning of multivariate features that are theoretically orthonormal.

This concept may provide a new theoretical approach for explaining SSL: the optimal multivariate features learned through SSL can be interpreted as the basis functions of the density ratio induced by

Preprint. Under review.

a corresponding probabilistic system. Upon formulating the probabilistic system, we propose the Hierarchical Functional Maximal Correlation Algorithm (HFMCA), an algorithm for multiview self-supervised learning that explores the hierarchical relationship between data and their augmentations. HFMCA offers faster convergence, stability that prevents feature collapse, and a theoretical foundation for interpretability.

2 Preliminary: Density Ratio Decomposition

Spectrum, basis functions, and density ratios. FMC is defined by directly applying spectral decomposition to the density ratio [21]. Given any two random processes $\mathbf{X}$ and $\mathbf{Y}$, with a joint distribution $p(X, Y)$ and the marginal product $p(X)p(Y)$, the decomposition follows

$$\rho := \frac{p(X,Y)}{p(X)p(Y)} = \sum_{k=1}^{\infty} \sqrt{\sigma_k}\phi_k(X)\psi_k(Y),\ \mathbb{E}_{\mathbf{X}}[\phi_k(\mathbf{X})\phi_{k'}(\mathbf{X})] = \mathbb{E}_{\mathbf{Y}}[\psi_k(\mathbf{Y})\psi_{k'}(\mathbf{Y})] = \begin{cases} 1, & k = k' \\ 0, & k \neq k' \end{cases}. \tag{1}$$

Each component in this decomposition has a unique role: the spectrum measures dependence (termed FMC), the bases are feature projectors, and the kernel-associated density ratio is a metric distance between two samples. Solving this spectral decomposition problem is an optimization problem.

Neural networks implementation. When dealing with empirical data and lacking the knowledge of *pdf*, spectral decomposition can be achieved through optimization. The empirical studies suggest a log-determinant-based cost function, optimized via paired neural networks, offers superior stability. Two neural networks $\mathbf{f}_\theta : \mathcal{X} \to \mathbb{R}^K$ and $\mathbf{g}_\omega : \mathcal{Y} \to \mathbb{R}^K$ are used to map realizations of $\mathbf{X}$ and $\mathbf{Y}$ respectively, each to a K-dimensional output space. We compute the autocorrelation (ACFs) and crosscorrelation functions (CCFs) between them, defined as

$$\mathbf{R}_F = \mathbb{E}_{\mathbf{X}}[\mathbf{f}_\theta(\mathbf{X})\mathbf{f}_\theta^\intercal(\mathbf{X})],\ \mathbf{R}_G = \mathbb{E}_{\mathbf{Y}}[\mathbf{g}_\omega(\mathbf{Y})\mathbf{g}_\omega^\intercal(\mathbf{Y})],\ \mathbf{P}_{FG} = \mathbb{E}_{\mathbf{X},\mathbf{Y}}[\mathbf{f}_\theta(\mathbf{X})\mathbf{g}_\omega^\intercal(\mathbf{Y})],\ \mathbf{R}_{FG} = \begin{bmatrix} \mathbf{R}_F & \mathbf{P}_{FG} \\ \mathbf{P}_{FG}^\intercal & \mathbf{R}_G \end{bmatrix}. \tag{2}$$

FMCA solves the following minimization problem:

$$\min_{\theta,\omega}\ r(\mathbf{f}_\theta, \mathbf{g}_\omega) := \log\det \mathbf{R}_{FG} - \log\det \mathbf{R}_F - \log\det \mathbf{R}_G. \tag{3}$$

Theoretically, the objective function captures the leading eigenvalues in the spectrum, while the neural networks after applying post-training normalizations approximate leading basis functions. The task of measuring dependence and learning multivariate features are unified by modeling the projection space created by the density ratio.

The defined eigenvalues all range from 0 to 1. The optimal cost approximates their aggregation $r^* = \sum_{k=1}^{K} \log(1 - \sigma_k)$. The dependence can be evaluated using both the spectrum and the cost, where a lower cost and higher eigenvalues indicate a stronger dependence.

3 Density Ratio Decomposition for SSL

We explore the potential of framing SSL as a dependence measurement problem. Regardless of the specified protocols, augmentations of an image describe the common source object. This relationship implies statistical dependence.

Consider an unaugmented image $X \sim \mathbb{P}(\mathbf{X})$, with $\mathbb{P}(\mathbf{X})$ being the given data distribution prior to any augmentation, which is the source data distribution that we are presented with. The augmentation protocols can be modeled as a transformation function $\mathcal{T}(X; v)$, which takes an image $X \in \mathcal{X}$ and a positive integer index $v \in \mathcal{V}$ symbolizing a specific protocol, and produce an augmented version of the image. The set $\mathcal{V}$ is a subset of positive integers $\mathcal{V} \subset \mathbb{Z}^+$, and its cardinality $|\mathcal{V}|$ is the total number of specified protocols. The augmented images will have a distribution $\mathbb{P}(\mathbf{Y})$, assuming they are modeled as a random process $\mathbf{Y} \in \mathcal{Y}$.

For simplicity, consider *pdf*s exist for all defined distributions. We propose that the augmentation procedure can be modeled as a conditional *pdf* $p(Y|\mathbf{X} = X) = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \mathbb{1}\{Y = \mathcal{T}(X; v)\}$, by applying a counting measure to all possible augmented versions of a given image X. Given $X \sim \mathbb{P}(\mathbf{X})$, augmentations of this image can be sampled from this conditional as $Y \sim \mathbb{P}(\mathbf{Y}|\mathbf{X} = X)$. The marginal $p(\mathbf{Y})$ is obtained by marginalizing over all images in the dataset

$$p(\mathbf{X} = X, \mathbf{Y} = Y) = p(X) \cdot \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \mathbb{1}\{Y = \mathcal{T}(X; v)\},\ p(\mathbf{Y} = Y) = \int_{\mathcal{X}} p(\mathbf{X} = X, \mathbf{Y} = Y)dX. \tag{4}$$

Simplify notations to be $p(X,Y)$, $p(X)$ and $p(Y)$. Naturally, we choose to decompose $\rho(X,Y) := \frac{p(X,Y)}{p(X)p(Y)} = \sum_{k=1}^{\infty} \sqrt{\sigma_k}\phi_k(X)\psi_k(Y)$, i.e., the density ratio induced by the joint distribution between the original image and its augmentations.

The remaining question becomes constructing the ACFs and CCFs in the log-determinant cost (3) and apply optimization. Assume that two neural networks $\mathbf{f}_\theta : \mathcal{X} \to \mathbb{R}^K$ and $\mathbf{g}_\omega : \mathcal{Y} \to \mathbb{R}^K$ are given. The two ACFs $\mathbf{R}_F = \mathbb{E}_{\mathbf{X}}[\mathbf{f}_\theta(\mathbf{X})\mathbf{f}_\theta^\intercal(\mathbf{X})]$ and $\mathbf{R}_G = \mathbb{E}_{\mathbf{Y}}[\mathbf{g}_\omega(\mathbf{Y})\mathbf{g}_\omega^\intercal(\mathbf{Y})]$ can be written and estimated empirically by their definitions.

Multiview system. Observe that sampling from this joint is to first sample an image in the dataset, then sample an augmentation from the conditional, indicating that the CCF should also be estimated in a similar way, in terms of the conditional:

$$\begin{aligned}\mathbf{P}_{FG} = \mathbb{E}_{\mathbf{X},\mathbf{Y}}[\mathbf{f}_\theta(\mathbf{X})\mathbf{g}_\omega^\intercal(\mathbf{Y})] &= \iint p(X)p(Y|\mathbf{X}=X)\mathbf{f}_\theta(X)\mathbf{g}_\omega^\intercal(Y)dXdY \\ &= \int p(X)\mathbf{f}_\theta(X)\mathbb{E}_{\mathbf{Y}}[\mathbf{g}_\omega^\intercal(\mathbf{Y})|\mathbf{X}=X]dX.\end{aligned} \tag{5}$$

Therefore, a proper approach is to first estimate the conditional expectation, by averaging over multiple augmentations of each individual image, then estimate the CCF by averaging over all images. This estimation of the conditional mean, which uses multiple views of an image similar to [22, 23], differs from the conventional contrastive learning approach that uses only two views.

To frame this formally, we introduce a series of L i.i.d. categorical r.v., $\mathbf{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_L\}$, denoting the execution of L augmentations. For each image, a set of indices $\{v_1, \cdots, v_L\} \subset \mathcal{V}$ is sampled, generating L views $\mathcal{T}(X;V) = \{\mathcal{T}(X;v), v \in v_1, \cdots, v_L\}$. Then the conditional mean can be estimated by averaging over these L views: $\mathbb{E}_{\mathbf{Y}}[\mathbf{g}_\omega(\mathbf{Y})|\mathbf{X}=X] = \frac{1}{L}\sum_{l=1}^{L}\mathbf{g}_\omega(\mathcal{T}(X;v_l))$. Then the CCF is estimated by averaging over all images.

Hierarchical structure. The second observation is that since augmentations, such as patches, are often part of the original image, this implies a hierarchical relationship that allows the two parameterized networks $\mathbf{f}_\theta$ and $\mathbf{g}_\omega$ to have shared structures. Instead of using two separate networks, the model topology can be simplified to be a cascade of the backbone $\mathbf{f}_\theta^{(1)}$ and the projection head $\mathbf{f}_\theta^{(2)}$ as approximators for basis functions. The backbone $\mathbf{f}_\theta^{(1)}$ is first applied to the L augmentations of an image, extracting L low-level features $\mathbf{Z}_l^{(1)}$, each with K dimensions. These L features are then concatenated in the feature channel, acting as inputs to the projection head, and producing the high-level feature $\mathbf{Z}^{(2)}$. The mapping to $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$ will be considered function approximators $\mathbf{f}_\theta$ and $\mathbf{g}_\omega$.

Combining the two modifications, we introduce HFMCA for SSL as follows.

Proposition 1. *Denote the feature maps produced by the backbone and the projection head as $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$, respectively. Also assign L auxiliary indices to $\mathbf{Z}^{(1)}$ as $\{\mathbf{Z}_l^{(1)}, l = 1, \cdots, L\}$, corresponding to L augmentations of an image. HFMCA solves the optimization problem:*

$$\begin{gathered}\mathbf{R}_1 = \mathbb{E}[\mathbf{Z}^{(1)}\mathbf{Z}^{(1)\intercal}],\ \mathbf{R}_2 = \mathbb{E}[\mathbf{Z}^{(2)}\mathbf{Z}^{(2)\intercal}],\ \mathbf{P}_{1,2} = \frac{1}{L}\mathbb{E}[\sum_{l=1}^{L}\mathbf{Z}_l^{(1)}\mathbf{Z}^{(2)\intercal}],\ \mathbf{R}_{1,2} = \begin{bmatrix}\mathbf{R}_1 & \mathbf{P}_{1,2}\\ \mathbf{P}_{1,2}^\intercal & \mathbf{R}_2\end{bmatrix}, \\ \min_\theta r_H := \log\det\mathbf{R}_{1,2} - \log\det\mathbf{R}_1 - \log\det\mathbf{R}_2.\end{gathered} \tag{6}$$

By the theory of FMCA, the objective function reaches the leading eigenvalues of the density ratio, with neural networks reaching the leading orthonormal basis functions upon normalizations.

4 Experiments

Fast convergence of HFMCA. We compared the performance with multiple benchmark models on CIFAR10 and CIFAR100, with the max accuracy achieved over 20, 200, and 800 epochs reported. All experiments use a consistent setup: a ResNet-18 backbone, batch size of 64, SGD optimizer, a learning rate of 0.06, and momentum of 0.9, following benchmark settings. We use standard SimCLR protocols [11] for augmentation and apply a KNN to embedded training images.

In HFMCA, for a batch of 64 images, we generate 128-dimensional feature maps for $L = 9$ distinct augmentations per image using a ResNet-18 backbone. These feature maps are then reshaped into

a 3×3 grid, forming a tensor of size $(64, 128, 3, 3)$ which is fed into a 3-layer CNN, creating a 128-dimensional feature per source image. HFMCA exhibits faster convergence and better accuracy as shown in Table 1.

Method	Heads	CIFAR10			CIFAR100		
		Epoch 20	Epoch 200	Epoch 800	Epoch 20	Epoch 200	Epoch 800
Methods with two views							
MoCo [24]	128	57.2	83.8	90.0	22.3	45.7	69.8
SimCLR [11]	128	46.7	82.2	87.5	19.6	43.9	65.7
Barlow Twins [25]	2048	45.7	83.5	85.7	28.1	47.1	**70.9**
SimSiam [26]	2048	50.5	83.7	90.0	22.5	39.9	66.0
VICReg [27]	2048	44.8	81.2	90.2	20.3	37.8	68.5
VICRegL [12]	2048	43.2	78.7	89.7	21.5	41.2	67.3
Methods with multiple views							
FastSiam [22]	2048	76.8	87.9	90.1	45.8	62.2	69.9
HFMCA	128	79.3	85.7	90.1	43.3	65.1	67.9

Table 1: Classification accuracy on CIFAR10 and CIFAR100 highlights HFMCA's effectiveness. HFMCA converges fastest among all methods, retaining near-optimal accuracy.

Stabiliy of HFMCA. We demonstrate HFMCA's stability by varying augmentation protocols. We test five distortion strengths across three protocols: random crops, color jitters, and gray scales, used in [11]. Each protocol is tested individually, keeping the other two at default values. Random crop varies from no cropping to the sampling and resizing of any patch from 1×1 to 32×32 as inputs. Color jitter strength refers to the intensity of distortions. Gray scale strength is the likelihood of images converting to gray scales, with the maximum strength making all images colorless. We show the cost as estimated dependence measure in Figure 1 and Table 2 as the estimated dependence level.

Fig. 1 shows learning dynamics of costs as estimated dependence, indicating that random crops impact the most, followed by color jitters, and gray scale. Increased distortion strength reduces dependence level (increases costs), but never reaches strict independence. Intriguingly, the learning settles at a certain level of dependence intrinsic to the dataset even in extreme cases. Modeling this intrinsic level of dependence, which is unaffected by the augmentation's richness, can be fundamental.

Table 2 further supports our argument by showing the classification accuracy (A) and costs (C) for these experiments. The results further support HFMCA's robustness, showing no major accuracy drop or feature collapse with increased distortion in any scenario.

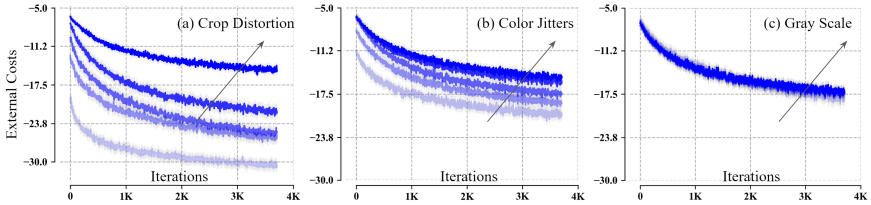


Figure 1: The learning dynamics of costs (dependence level) are displayed for five distortion strengths across three protocols. The arrow's direction indicates an increase in distortion strength. A lower cost value implies a higher dependence level. The figure implies that as distortion strength increases, the dependence level decreases. Even in extreme cases, a consistent level of dependence remains, intrinsic to the dataset.

Strength	A (%)	C
0	24.8	−30.4
0.25	54.5	−26.1
0.5	67.7	−25.4
0.75	71.2	−21.9
1	69.9	−15.1

(a) Crop Distortion

Strength	A (%)	C
0	49.0	−20.4
0.25	69.6	−18.8
0.5	71.0	−17.5
0.75	70.6	−15.7
1	70.8	−15.0

(b) Color Jitters

Strength	A (%)	C
0	61.2	−18.1
0.25	71.2	−17.4
0.5	70.4	−17.3
0.75	71.4	−17.2
1	70.7	−17.1

(c) Grey Scale

Table 2: A comparison of classification accuracy (A) and costs (C) across three protocols. An increase in distortion strength leads to decreased dependence but improves classification accuracy. The external costs never retain zero in all scenarios, suggesting an intrinsic level of dependence within the dataset.

5 Discussion

This paper provides a theoretical interpretation of SSL features as orthonormal basis functions of the density ratio. We propose HFMCA for learning SSL features with fast convergence and enhanced stability. In the appendix, we further discuss how this analysis extends to internal features to provide model interpretabilities. Our study has not yet incorporated local-level supervision, such as patch augmentation [23], which can be explored in future work.

References

[1] Alfréd Rényi. On measures of dependence. *Acta mathematica hungarica*, 10(3-4):441–451, 1959.

[2] Alfréd Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

[3] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6):066138, 2004.

[4] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2 edition, 2006.

[5] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[6] Xiao Wang and Maya R. Gupta. Deep information bottleneck. In *International Conference on Learning Representations (ICLR)*, 2016.

[7] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations (ICLR)*, 2019.

[8] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[9] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations (ICLR)*, 2017.

[10] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.

[12] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicregl: Self-supervised learning of local visual features. *arXiv preprint arXiv:2210.01571*, 2022.

[13] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[14] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R. Devon Hjelm. Mine: Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80, pages 531–540, 2018.

[15] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[16] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, 2020.

[17] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019.

[18] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 875–884. PMLR, 2020.

[19] Shao-Lun Huang, Gregory W Wornell, and Lizhong Zheng. Gaussian universal features, canonical correlations, and common information. In *2018 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2018.

[20] Shao-Lun Huang, Anuran Makur, Gregory W Wornell, and Lizhong Zheng. On universal features for high-dimensional learning and inference. *arXiv preprint arXiv:1911.09105*, 2019.

[21] Bo Hu and Jose C Principe. The cross density kernel function: A novel framework to quantify statistical dependence for random processes. *arXiv preprint arXiv:2212.04631*, 2022.

[22] Daniel Pototzky, Azhar Sultan, and Lars Schmidt-Thieme. Fastsiam: Resource-efficient self-supervised learning on a single gpu. In *Pattern Recognition: 44th DAGM German Conference, DAGM GCPR 2022, Konstanz, Germany, September 27–30, 2022, Proceedings*, pages 53–67. Springer, 2022.

[23] Shengbang Tong, Yubei Chen, Yi Ma, and Yann Lecun. Emp-ssl: Towards self-supervised learning in one training epoch. *arXiv preprint arXiv:2304.03977*, 2023.

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2020.

[25] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[26] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, 2021.

[27] Shang Wang, Zhixuan Liao, Mathilde Caron, and Piotr Bojanowski. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Supplemental Materials

A.1 Extending Analysis to Internal Features

Empirical evidence from our experiments indicates that the use of HFMCA for self-supervision significantly accelerates training and enhances stability. To further interpret our approach, we must investigate the internal structures of images. Fig. 1 presents the full diagram of our proposed approach.

A.1.1 Extensions of SSL Propositions

As the main paper proposes modeling the conditional associated with the augmentation protocols with a counting measure considering all possible augmentation protocols. This is possible due to the hierarchical relationship between images and their augmented versions. For any pair of random processes that exhibit a hierarchical structure, not limited to SSL, we propose a generalized proposition of HFMCA.

Proposition 1. *Consider a random process $\mathbf{X}^{(2)} = \{\mathbf{X}^{(2)}(l), l = 1, \cdots, L\}$, composed of L smaller processes that all share the same support. We denote these smaller processes as $\mathbf{X}^{(1)}$. In essence, $\mathbf{X}^{(2)}$ and $\mathbf{X}^{(1)}$ symbolize two hierarchical levels, with $\mathbf{X}^{(2)}$ and $\mathbf{X}^{(1)}$ corresponding to the higher and lower levels, respectively. The lower-level marginal distribution $p(\mathbf{X}^{(1)})$ can be determined by collecting all possible realizations from the lower-level components, independent of $\mathbf{X}^{(2)}$, as $p(\mathbf{X}^{(1)} = X^{(1)}) = \frac{1}{L}\sum_{l=1}^{L} p(\mathbf{X}^{(2)}(l) = X^{(1)})$. Next, for a given higher-level realization $X^{(2)}$ of $\mathbf{X}^{(2)}$, the conditional distribution of its components is characterized by an empirical distribution:*

$$p(X^{(1)}|\mathbf{X}^{(2)} = X^{(2)}) = \frac{1}{L}\sum_{l=1}^{L} \mathbb{1}\{X^{(1)} = X^{(2)}(l)\}. \tag{1}$$

This induces a cross-scale joint distribution $p_H(X^{(1)}, X^{(2)}) = p(X^{(1)}|X^{(2)})p(X^{(2)})$. Combine the joint and the marginal product, we define the induced density ratio $\rho_H(X^{(1)}, X^{(2)}) = \frac{p(X^{(1)}, X^{(2)})}{p(X^{(1)})p(X^{(2)})}$.

Proposition 2. *Assign two neural networks $\mathbf{f}_\theta^{(1)}$ and $\mathbf{f}_\theta^{(2)}$ to two processes $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ respectively. Denote the produced feature maps as $\mathbf{Z}^{(1)}$ (for low-level hierarchy) and $\mathbf{Z}^{(2)}$ (for high-level hierarchy), respectively, and represent features of L augmentations as $\{\mathbf{Z}_l^{(1)}, l = 1, \cdots, L\}$. The optimization problem of HFMCA*

$$\mathbf{R}_1 = \mathbb{E}[\mathbf{Z}^{(1)}\mathbf{Z}^{(1)\intercal}],\ \mathbf{R}_2 = \mathbb{E}[\mathbf{Z}^{(2)}\mathbf{Z}^{(2)\intercal}],\ \mathbf{P}_{1,2} = \frac{1}{L}\mathbb{E}[\sum_{l=1}^{L}\mathbf{Z}_l^{(1)}\mathbf{Z}^{(2)\intercal}],\ \mathbf{R}_{1,2} = \begin{bmatrix}\mathbf{R}_1 & \mathbf{P}_{1,2}\\ \mathbf{P}_{1,2}^\intercal & \mathbf{R}_2\end{bmatrix}, \tag{2}$$
$$\min_\theta r_H := \log\det\mathbf{R}_{1,2} - \log\det\mathbf{R}_1 - \log\det\mathbf{R}_2$$

applies to any pair of processes $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ that exhibit a hierarchical relationship stated in Proposition 1.

A.1.2 Hierarchy in Focus: Pixels, Patches & Images

As multiple views represent one source object, a similar relationship is naturally present within image hierarchies between pixels, patches, and full images. Moreover, our construction also suggests that this relationship exists at local levels. For instance, the composition from pixels to patches is

Preprint. Under review.

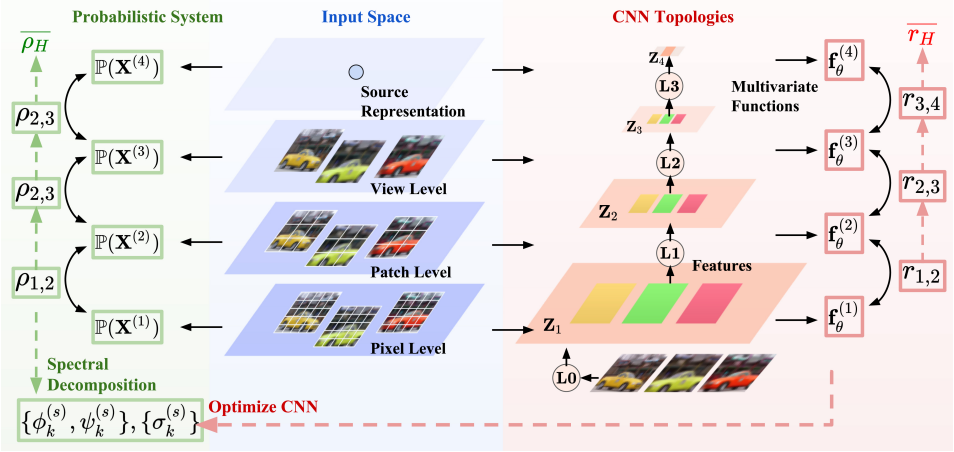


Figure 1: Overview of HFMCA considering both external and internal costs. It comprises three components: Probabilistic system, Input space, and CNN topology. Input space contains pixels, patches, and image augmentations (views), and their collection (source representation). Probabilistic system defines density ratios $\rho_{s,s+1}$ between neighboring scales, exhibiting the telescoping property to form the global density ratio $\overline{\rho_H}$. Each density ratio $\rho_{s,s+1}$ yields a spectral decomposition into bases $\{\phi_k^{(s)}, \psi_k^{(s)}\}$ and spectrum $\{\sigma_k^{(s)}\}$. CNN topology applies downsampling layers $\mathbf{L0}, \mathbf{L1}, \mathbf{L2}, \cdots$, each layer functioning as a universal mapper $\mathbf{f}_\theta^{(s)}$ of the receptive field, measurable against $\mathbb{P}(\mathbf{X}^{(s)})$. Cost functions $r_{s,s+1}$ optimize eigenvalues until identifying leading eigenvalues and bases. Spectrum (dependence measure) and bases (features) are acquired from training CNNs. External costs apply at the top scale, when multiple views compose their respective collections. Internal costs apply at scales within image hierarchies, such as pixels, patches, and full images.

unaffected by and remains independent of how patches compose the full images. Thus, the most precise formulation is based on neighboring hierarchical levels.

We define a sequence of r.p. $\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(S)}\}$, where S denotes the number of hierarchical scales. In this context, $\mathbf{X}^{(1)}$ corresponds to pixels, $\mathbf{X}^{(S)}$ to images, and the intermediate processes $\mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(S-1)}$ to image patches. Each element $\mathbf{X}^{(s)}$ in this sequence has spatial dimensions (H_s, W_s) and is composed of pixels: $\mathbf{X}^{(s)} = \{\mathbf{X}^{(s)}(i,j) \mid 1 \le i \le H_s,\ 1 \le j \le W_s\}$.

Our aim is to replicate the previous procedure to define the joint distribution across two neighboring hierarchical scales. This aligns with the standard patch creation procedure. Given a patch $X^{(s+1)}$ at scale $s+1$, it can be divided into multiple subpatches at a lower scale s. When two scales have nearly identical dimensions, the conditional distribution $p(\mathbf{X}^{(s)} = X^{(s)} | X^{(s+1)})$ has discrete, finite support of subpatches at scale s, modeled by an empirical distribution. Denote the differences in their patch dimensions as $\Delta H_s = H_{s+1} - H_s + 1$ and $\Delta W_s = W_{s+1} - W_s + 1$, we define

$$p(X^{(s)} | X^{(s+1)}) = \frac{1}{\Delta H_s \Delta W_s} \sum_{m=1}^{\Delta H_s} \sum_{n=1}^{\Delta W_s} \mathbb{1}\{X^{(s)} = X^{(s+1)}(m\colon m+H_s, n\colon n+W_s)\}. \quad (3)$$

This induces a joint distribution $p_H(X^{(s)}, X^{(s+1)}) = p(X^{(s)} | X^{(s+1)}) p(X^{(s+1)})$ for every s. To sample from $p(X^{(s)}, X^{(s+1)})$, we first draw a $X^{(s+1)}$ at scale $s+1$, followed by sampling a subpatch $X^{(s)}$ within it, which maintains their dependence. The joint distribution $p_H(X^{(s)}, X^{(s+1)})$ further induces a series of density ratios $\rho_H(X^{(s)}, X^{(s+1)})$ for every scale s.

Telescoping property of density ratios. An important property of our construction is the local existence of dependencies within hierarchies. Pixels first constitute patches, regardless of how these patches eventually combine to form images. Likewise, patches make up images, regardless of the correspondence between augmentations and the source object. This inherent trait uncovers the telescoping nature of dependencies, which can be characterized formally with density ratios:

Proposition 3. *The global-level dependence for the image hierarchical sequence* $\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(S)}\}$ *exists locally between neighboring hierarchical levels, as*

$$\log \frac{p(X^{(1)}, X^{(2)}, \cdots, X^{(S)})}{p(X^{(1)}) p(X^{(2)}) \cdots p(X^{(S)})} = \sum_{s=1}^{S-1} \log \rho_H(X^{(s)}, X^{(s+1)}) := \log \overline{\rho_H}(X^{(1)}, \cdots, X^{(S)}). \quad (4)$$

We name this relationship the telescoping property of density ratios.

This property matches our discussion and addresses the necessity and sufficiency of modeling statistical dependence between neighboring hierarchical levels. It also reveals a crucial characteristic: the global-level dependence can be defined and modeled at local levels.

A.1.3 Functional Maximal Correlation via CNNs: Theoretical Solutions

We have defined a sequence of density ratios $p_H(X^{(s)}, X^{(s+1)})$, ranging from scales $s = 1$ to $S - 1$. The remaining step is to apply HFMCA for decomposition. This follows the procedure detailed in Proposition 2, which includes the implementation of the cost r_H and the subsequent minimization. Remarkedly, the topology for this optimization aligns perfectly with the structure of a vanilla CNN.

Our construction is based on the assumption that each element in the CNN feature map can act as a universal approximator for its corresponding receptive field. This mapping relationship aligns with the functions needed as basis functions in the optimization.

As detailed in Proposition 2, the execution of HFMCA involves first applying a feature network to each component patch, concatenating these lower-level features, and then passing them into another network to generate the higher-level features. This procedure bears a close resemblance to the convolution operation, where kernels are applied to subregions of the preceding layer's feature maps. We illustrate this with Fig. 2 and the following explanation.

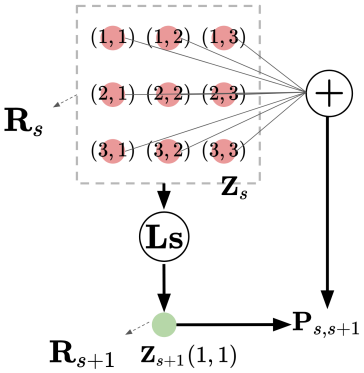


Figure 2: Illustration of HFMCA costs (Eq. (2)). Use region averages in $\mathbf{Z}_s$ and elements in $\mathbf{Z}_{s+1}$ to compute the CCF. Combine the CCF with ACFs of two marginals to construct cost functions r_H at every layer s.

Consider a CNN with S layers. The initial layer consists of 1×1 convolution kernels, yielding feature maps $\mathbf{Z}^{(1)}$, where the receptive fields correspond to individual pixels. Supposedly the second layer consists of $\Delta H_1 \times \Delta W_1$ convolution kernels (as refer to the patch size in Eq. (3)). This layer operates on the concatenation of $\Delta H_1 \times \Delta W_1$ elements from the feature maps of the first layer. We can infer that the receptive field of this second layer aligns with image patches of size $H_2 \times W_2$.

Hence, the receptive fields of first layer align with $\mathbf{X}^{(1)}$ in the hierarchical sequence, while those of the second layer align with $\mathbf{X}^{(2)}$. Following Proposition 2, minimizing the cost r_H between $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$, denoted as $r_H(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)})$, effectively decomposes the density ratio between their receptive fields, $\rho_H(X^{(1)}, X^{(2)})$.

Likewise, for the remaining layers with kernels dimensions $\Delta M_s \times \Delta N_s$, its receptive field will align precisely with the patch size of $\mathbf{X}^{(s)}$. By constructing and minimizing costs $r_H(\mathbf{Z}^{(s)}, \mathbf{Z}^{(s+1)})$ between neighboring CNN layers, we effectively decompose the density ratios $\rho_H(X^{(s)}, X^{(s+1)})$ between neighboring hierarchical levels.

Internal costs for image hierarchies. Suppose this CNN generates S feature maps $\mathbf{Z}^{(1)}, \cdots, \mathbf{Z}^{(S)}$. We introduce internal costs $r_H(\mathbf{Z}^{(s)}, \mathbf{Z}^{(s+1)})$ (per Proposition 2), and minimize the total cost $\min_\theta \sum_{s=1}^{S} r_H(\mathbf{Z}^{(s)}, \mathbf{Z}^{(s+1)})$. This minimization reveals statistical dependencies within image hierarchies. If an external cost $r_H(\mathbf{Z}^{(S)}, \mathbf{Z}^{(S+1)})$ is present, we formulate the task as follows,

$$\min_{\theta} \ \lambda \sum_{s=1}^{S} r_H(\mathbf{Z}^{(s)}, \mathbf{Z}^{(s+1)}) + r_H(\mathbf{Z}^{(S)}, \mathbf{Z}^{(S+1)}). \tag{5}$$

Upon reaching the minimum, we derive an approximation of the sequence of density ratios $\rho_H(X^{(s)}, X^{(s+1)})$. This approximation yields a decomposition at each local level, expressed by the spectrum and the corresponding orthonormal bases at each CNN layer.

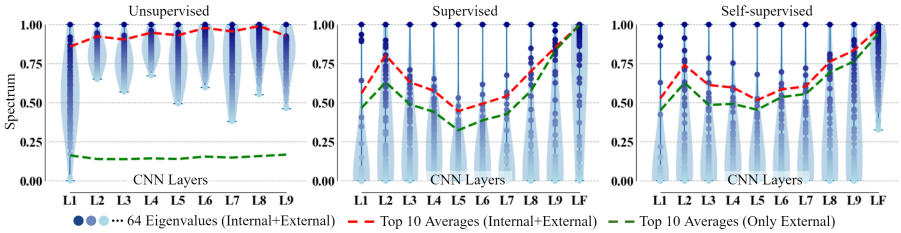


Figure 3: Spectrum across layers under three supervision types. Each plot column represents the spectrum of a specific CNN layer, ranging from **L1** to the last layer **LF**. Each spectrum consists of 64 eigenvalues, described as blue dots. The dots' color intensity reflects their rank in the spectrum, with the blue background showing their distributions. In unsupervised scenarios, we consistently observe the presence of large eigenvalues, indicating the dataset's intrinsic dimensionality. Supervision and self-supervision significantly impact the eigenvalues in the middle layers, creating large null spaces to meet the goal of discrimination.

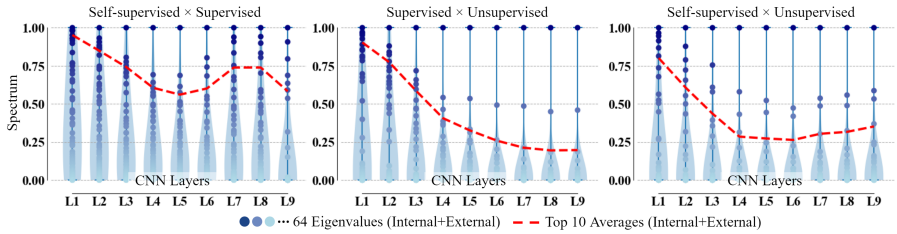


Figure 4: Spectrum across pairs of supervision types. Eigenvalues, by projecting eigenfunctions onto each other, quantify the alignment between supervision pairs across network layers. Supervised and unsupervised learning exhibit different internal representations due to fewer large eigenvalues, particularly at the top layers. Supervised and self-supervised learning settings learn very similar spaces, albeit with some differences in the middle layers where most null space projections occur.

A.2 Experiments for Internal Feature Explainability

A.2.1 Comparison between Supervision & Self-Supervision

Shifting our focus to representations, we investigate HFMCA's behavior under different supervision types using the learned spectrum for interpretability. Additional experiments illustrated in Fig. 3 and Fig. 4, conducted on CIFAR10, include unsupervised (where only internal costs are used) and supervised scenarios (where L distinct views are substituted with L samples from the same class), with and without internal costs. We use a modified CNN backbone and hyperparameters detailed in the appendix. The learned HFMCA eigenspectrum between pairs of layers is very telling. The eigenvalues range between 0 and 1 because of cost normalization and reflect dependence strength. Since dependence is associated with the correlation between the eigenfunctions, we explore this interpretation to discuss the layer effective dimension (different from the number of eigenfunctions that are kept at 64). For the unsupervised case, excluding the first layer, the eigenvalues are basically in the same range, mostly above 0.5, which means that there are minor modifications in the space dimensions due to the nonlinear mappings, but the eigenvalue distribution is always far from 0. Hence, the dynamic range of the eigenvalues across layers shows that the dimensionality of the projection spaces oscillates around the intrinsic dimensionality of the input data set.

This picture changes drastically for the supervised and self-supervised cases, due to the external desired responses that force discrimination in the network input-output map. The dimensionality of the labels in the supervised case is much less than the intrinsic dimension of the input, and so we see a large number of eigenvalues close to zero (light blue regions), which means that the data is being projected to a smaller subspace across the layers. More importantly, notice that the higher density of zero eigenvalues occurs in the middle layers. For the self-supervised case, we have a similar picture, with a notable difference that the spread of eigenvalues in the last layer closely resembles that of the unsupervised case. This alignment is expected since the desired output is based on the source image itself. So we can conclude that the discrimination is affecting mostly the eigenvalues in the middle layers, creating large null spaces to meet the goal of classification. This explains why the feature collapse is so common in these types of applications. We also plot in a red dotted line the average of the 10 largest eigenvalues that corroborate the analysis of the number of zero eigenvalues.

The eigenvalues have another important application because they quantify the solid angle between eigenfunctions. This is particularly important when comparing the learned eigenfunctions across different supervision settings. Fig. 4 shows the eigenvalues between each pair of supervisions at each layer, by projecting eigenfunctions onto each other, to quantify the alignment of the spanned projection spaces (values close to 1/0 mean parallel/orthogonal eigenfunctions, respectively). Notice that there are very few large eigenvalues across the supervised settings versus unsupervised, in particular on the top layers, meaning that their internal representations are quite different. The eigenvalues of the supervised versus self-supervised are much larger, meaning that they learn very similar spaces, particularly at the initial and final layers. This similarity decreases in the intermediate layers, which is precisely where most null space projections occur. This analysis provides a very specific understanding of the internal representations of complex networks across different settings, owing to the proposed methodology.

A.2.2 Visualizing telescoping density ratios

An important component of our hierarchical dependence model is the telescoping property. We demonstrate that the density ratios between two neighboring layers identify their dependencies, which effectively captures the global dependence information by extending to the entire network. Thus, starting from the top layer, we calculate the local density ratios between neighboring layers, passing these density ratios down to the bottom layers, layer by layer. Fig. 5 displays the local response at layer **L2** across three learning setups, revealing the most informative regions. We observe that the boundaries of objects, and the interactions between different parts of an object (such as how wheels are connected to the car body), play a critical role in learning.

This process is quite similar to the effect of backpropagation of errors through gradients, but it is much more efficient and principled because it transmits statistical dependence instead of just the simple gradient of the error, which quantifies only the maximum rate of change. In our opinion, the remarkable performance of HFMCA for SSL can be attributed to its telescoping property. Effectively, the telescoping property could serve as an alternative to error backpropagation training, potentially facilitating the effective layer-by-layer training of deep networks, and mitigating the issue of vanishing gradients while preserving interpretability on the image plane.

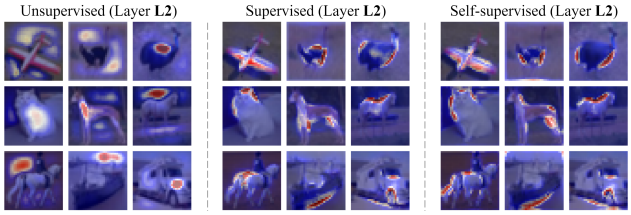


Figure 5: Global-to-local density ratio response of 9 CIFAR10 samples for layer **L2**. The boundaries of objects, and the interactions between different parts of an object, play a critical role in learning. This further demonstrates the resemblances between supervision and self-supervision.

A.3 Pseudocode and algorithm details

In this section, we will further illustrate the proposed methods, including the decomposition scheme with CNNs, obtaining basis functions and the spectrum through normalizations, and producing the local density ratio response.

A.3.1 Details of Decomposition with CNNs

Continuing from Section 4.2 of the paper, we further illustrate the use of CNN topologies in applying density ratio decomposition to image hierarchies.

Notations. To make the illustration more clear, we first clarify the notations that we use:

- $\mathcal{F}_\theta^{(1)}, \mathcal{F}_\theta^{(2)}, \cdots, \mathcal{F}_\theta^{(S)}$: Convolution layers of the CNN, each a function receiving patches from preceding feature maps.
- $\widehat{\mathcal{F}_\theta^{(1)}}, \widehat{\mathcal{F}_\theta^{(2)}}, \cdots, \widehat{\mathcal{F}_\theta^{(S)}}$: Each CNN layer treated as a function of the corresponding receptive field in input images.
- $\mathbf{Z}_1, \mathbf{Z}_2, \cdots, \mathbf{Z}_S$: Feature maps at each layer.

- $\mathbf{R}_{\phi}^{(s)}, \mathbf{R}_{\psi}^{(s)}, \mathbf{P}_{\phi,\psi}^{(s)}, \mathbf{R}_{\phi,\psi}^{(s)}, 1 \leq s \leq S$: Autocorrelation functions (ACFs) and crosscorrelation functions (CCFs) defining the optimization cost at each layer.

CNN as function approximators of feature maps. Consider a CNN with S layers. The first layer consists of 1×1 convolution kernels (denote as $\mathcal{F}_{\theta}^{(1)}$). The remaining layers have kernels of dimensions $\Delta M_s \times \Delta N_s$ (denote as $\mathcal{F}_{\theta}^{(2)}, \cdots, \mathcal{F}_{\theta}^{(S)}$). We assume a fixed channel number K, and denote each layer's feature map, a tensor with dimensions (H_s, W_s, K), as $\mathbf{Z}_s = \{\mathbf{Z}^{(s)}(i,j) \mid 1 \leq i \leq H_s,\ 1 \leq j \leq W_s \in \mathbb{R}^{H_s \times W_s \times K}\}$.

Every convolutional layer can be treated as a mapping function $\mathcal{F}_{\theta}^{(s)} : \Delta M_s \times \Delta N_s \to \mathbb{R}^K$, which operates on the outputs of its preceding layer, as $\mathbf{Z}_{s+1}(i,j) = \mathcal{F}_{\theta}^{(s+1)}(\mathbf{Z}_s(i : i+\Delta M_s, j : j+\Delta N_s))$. It follows that each element $\mathbf{Z}_s(i,j)$ in the feature map corresponds to a receptive field of dimensions (H_s, W_s) in the input images, which we denote as $\mathbf{Z}_s(i,j) = \widehat{\mathcal{F}_{\theta}^{(s+1)}}(\mathbf{X}(i : i+H_s, j : j+W_s))$. This sequence of functions $\{\widehat{\mathcal{F}_{\theta}^{(s)}} : \mathcal{X}^{(s)} \to \mathbb{R}^K\}$ can be directly used as basis functions for density ratio decomposition proposed in Proposition 2 of the main paper.

Minimization criterion. Taking into account the defined forms of joint distributions, the ACFs and CCFs can be defined at each layer and between two neighboring layers, taking the following forms:

$$\begin{aligned} \mathbf{R}_{\phi}^{(s)} &= \frac{1}{M_{\phi}^{(s)}} \mathbb{E}[\sum_{i,j} \sum_{i'=i}^{i+\Delta M_s} \sum_{j'=j}^{j+\Delta N_s} \mathbf{Z}_s(i',j') \mathbf{Z}_s^{\intercal}(i',j')],\ \mathbf{R}_{\psi}^{(s)} = \frac{1}{M_{\psi}^{(s)}} \mathbb{E}[\sum_{i,j} \mathbf{Z}_{s+1}(i,j) \mathbf{Z}_{s+1}^{\intercal}(i,j)] \\ \mathbf{P}_{\phi,\psi}^{(s)} &= \frac{1}{M_{\phi}^{(s)}} \mathbb{E}[\sum_{i,j} \sum_{i'=i}^{i+\Delta M_s} \sum_{j'=j}^{j+\Delta N_s} \mathbf{Z}_s(i',j') \mathbf{Z}_{s+1}^{\intercal}(i,j)], \mathbf{R}_{\phi,\psi}^{(s)} = \begin{bmatrix} \mathbf{R}_{\phi}^{(s)} & \mathbf{P}_{\phi,\psi}^{(s)} \\ \mathbf{P}_{\phi,\psi}^{(s)\intercal} & \mathbf{R}_{\psi}^{(s)} \end{bmatrix}, \end{aligned} \tag{6}$$

where $M_{\phi}^{(s)}$ and $M_{\psi}^{(s)}$ simply denote the count of additions. The definition of $\mathbf{R}_{\phi}^{(s)}$ arises from how each layer is applied to the boundaries. For example, only one element in $\mathbf{Z}_{s+1}$ has a mapping relationship with the boundaries of $\mathbf{Z}_s$. Therefore, we examine each element in $\mathbf{Z}_{s+1}$, identify its corresponding inputs in $\mathbf{Z}_s$, compute the ACFs for all such related elements, and repeat this process for every element in $\mathbf{Z}_{s+1}$.

The internal costs and the minimization task can be written as

$$r_{s,s+1} = \log\det \mathbf{R}_{\phi,\psi}^{(s)} - \log\det \mathbf{R}_{\phi}^{(s)} - \log\det \mathbf{R}_{\psi}^{(s)},\ 1 \leq s \leq S-1;\ \ \overline{r} = \sum_{s=1}^{S-1} r_{s,s+1};\ \ \underset{\theta}{\text{minimize}}\ \overline{r}. \tag{7}$$

We provide the pseudocode for optimizing the internal costs in Algorithm 1.

Gradient estimation. In our implementation, an adaptive filter can be added for gradient estimation, similar to a conventional Adam optimizer [1]. Note that the gradient of $r_{s,s+1}$ has the form

$$\frac{\partial r_{s,s+1}}{\partial \theta} = \mathbf{Tr}((\mathbf{R}_{\phi,\psi}^{(s)})^{-1} \frac{\partial \mathbf{R}_{\phi,\psi}^{(s)}}{\partial \theta}) - \mathbf{Tr}((\mathbf{R}_{\phi}^{(s)})^{-1} \frac{\partial \mathbf{R}_{\phi}^{(s)}}{\partial \theta}) - \mathbf{Tr}((\mathbf{R}_{\psi}^{(s)})^{-1} \frac{\partial \mathbf{R}_{\psi}^{(s)}}{\partial \theta}). \tag{8}$$

Thus, we use adaptive filters to estimate the three ACFs, and substitute the argument within the inverse function with these estimated values. We provide the pseudocode for this procedure in Algorithm 2.

A.3.2 Pseudocode for HFMCA SSL

Continuing from Section 3 of the paper, we provide the pseudocode for HFMCA with self-supervision in Algorithm 3.

A.3.3 Retrieve Eigenspectrum and Basis Functions.

After training, we apply a standard normalization scheme to obtain the spectrum and basis functions from neural network outputs. This includes the first step by enforcing orthonormality with $\overline{\phi_{\theta}^{(s)}} = \mathbf{R}_{\phi}^{(s)-\frac{1}{2}} \widehat{\mathcal{F}_{\theta}^{(s)}}, \overline{\psi_{\theta}^{(s)}} = \mathbf{R}_{\psi}^{(s)-\frac{1}{2}} \widehat{\mathcal{F}_{\theta}^{(s+1)}}$. The second step is to apply the singular-value decomposition such that the functions are invariant to the conditional mean operator, which follows

$$\mathbb{E}[\overline{\phi_{\theta}^{(s)}}\ \overline{\psi_{\theta}^{(s)}}^{\intercal}] = \mathbf{U}_s \widehat{\mathbf{\Sigma}_s}^{\frac{1}{2}} \mathbf{V}_s^{\intercal},\ \widehat{\mathbf{\Sigma}_s} = \text{diag}([\widehat{\sigma_1^{(s)}}, \cdots, \widehat{\sigma_K^{(s)}}]),\ \widehat{\phi_{\theta}^{(s)}} = \mathbf{U}_s^{\intercal} \overline{\phi_{\theta}^{(s)}},\ \widehat{\psi_{\theta}^{(s)}} = \mathbf{V}_s^{\intercal} \overline{\psi_{\theta}^{(s)}}. \tag{9}$$

Algorithm 1 HFMCA - Internal Costs

Initialize CNN with downsampling blocks $\mathcal{F}_\theta^{(1)}, \mathcal{F}_\theta^{(2)}, \mathcal{F}_\theta^{(3)}, \cdots, \mathcal{F}_\theta^{(S)}$
while convergence is not reached **do**
  Sample a batch of images $\mathbf{X}$; $\mathbf{Z}_1 = \mathcal{F}_\theta^{(1)}(\mathbf{X})$; $\overline{r} = 0$
  for $s = 1, \cdots, S-1$ **do**
    $\mathbf{Z}_{s+1} = \mathcal{F}_\theta^{(s+1)}(\mathbf{Z}_s)$
    $\mathbf{R}_\psi^{(s)} = \text{mean}(\mathbf{Z}_{s+1}(i,j)\mathbf{Z}_{s+1}^\intercal(i,j))$
    For every $\mathbf{Z}_{s+1}^\intercal(i,j)$, find its receptive field in $\mathbf{Z}_s$: $\{\mathbf{Z}_s(i',j'), i' \in I, j' \in J\}$
    $\mathbf{R}_\phi^{(s)} = \text{mean}\left((\frac{1}{|I||J|}\sum_{i'}\sum_{j'}\mathbf{Z}_s(i',j')\mathbf{Z}_s^\intercal(i',j')\right)$
    $\mathbf{P}_{\phi,\psi}^{(s)} = \text{mean}\left(\frac{1}{|I||J|}\sum_{i'}\sum_{j'}\mathbf{Z}_s(i',j')\mathbf{Z}_{s+1}^\intercal(i,j)\right)$
    $\mathbf{R}_{\phi,\psi}^{(s)} = \mathbf{R}_{\phi,\psi}^{(S)} = \begin{bmatrix} \mathbf{R}_\phi^{(S)} & \mathbf{P}_{\phi,\psi}^{(S)} \\ \mathbf{P}_{\phi,\psi}^{(S)\intercal} & \mathbf{R}_\psi^{(S)} \end{bmatrix}$
    $r_{s,s+1} = \log\det\mathbf{R}_{\phi,\psi}^{(s)} - \log\det\mathbf{R}_\phi^{(s)} - \log\det\mathbf{R}_\psi^{(s)}$; $\overline{r} \leftarrow \overline{r} + r_{s,s+1}$
  end for
  SGD: $\theta \leftarrow \theta + \partial\overline{r}/\partial\theta$
end while

Algorithm 2 Adaptive Filters for Gradient Estimation

$k = 1$; Initiate ACFs/CCFs estimators $\{\tilde{\mathbf{R}}_\phi^{(s)}, \tilde{\mathbf{R}}_\psi^{(s)}, \tilde{\mathbf{R}}_{\phi,\psi}^{(s)}\}_{s=1}^S$
while convergence is not reached **do**
  for $s = 1, \cdots, S$ **do**
    $\tilde{\mathbf{R}}_\phi^{(s)} \leftarrow \beta\tilde{\mathbf{R}}_\phi^{(s)} + (1-\beta)\mathbf{R}_\phi^{(s)}$
    $\tilde{\mathbf{R}}_\psi^{(s)} \leftarrow \beta\tilde{\mathbf{R}}_\psi^{(s)} + (1-\beta)\mathbf{R}_\psi^{(s)}$
    $\tilde{\mathbf{R}}_{\phi,\psi}^{(s)} \leftarrow \beta\tilde{\mathbf{R}}_{\phi,\psi}^{(s)} + (1-\beta)\mathbf{R}_{\phi,\psi}^{(s)}$
    $\widehat{\mathbf{R}}_\phi^{(s)} = \tilde{\mathbf{R}}_\phi^{(s)}/(1-\beta^k)$; $\widehat{\mathbf{R}}_\psi^{(s)} = \tilde{\mathbf{R}}_\psi^{(s)}/(1-\beta^k)$; $\widehat{\mathbf{R}}_{\phi,\psi}^{(s)} = \tilde{\mathbf{R}}_{\phi,\psi}^{(s)}/(1-\beta^k)$
    Estimate gradients:
    $\frac{\partial r_{s,s+1}}{\partial\theta} \approx \text{pinv}(\widehat{\mathbf{R}}_{\phi,\psi}^{(s)})\ \frac{\partial\mathbf{R}_{\phi,\psi}^{(s)}}{\partial\theta}$ - $\text{pinv}(\widehat{\mathbf{R}}_\phi^{(s)})\ \frac{\partial\mathbf{R}_\phi^{(s)}}{\partial\theta}$ - $\text{pinv}(\widehat{\mathbf{R}}_\psi^{(s)})\ \frac{\partial\mathbf{R}_\psi^{(s)}}{\partial\theta}$
  end for
  $k \leftarrow k+1$
end while

Algorithm 3 HFMCA - External Costs

Choose number of views L; Initialize CNN $\mathcal{F}_\theta^{(1)}, \cdots, \mathcal{F}_\theta^{(S)}$; Initialize $\mathcal{F}_\theta^{(S+1)} : K \times L \to \mathbb{R}$
while convergence is not reached **do**
  Sample a batch of images $\mathbf{X}$
  if self-supervised:
    Create L augmentations of $\mathbf{X}$, resulting in $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$
  else if supervised:
    Sample $L-1$ batches with samples of the same class as $\mathbf{X}$, resulting in $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$
  $\mathbf{Z}_{1,l} = \mathcal{F}_\theta^{(1)}(\mathbf{X}_l)$ for all l ; $\overline{r} = 0$
  for $s = 1, \cdots, S-1$ **do**
    $\mathbf{Z}_{s+1,l} = \mathcal{F}_\theta^{(s+1)}(\mathbf{Z}_{s,l})$ for all l
    if internal costs: Build $r_{s,s+1}$; $\overline{r} \leftarrow \overline{r} + r_{s,s+1}$
  end for
  Concatenate feature maps as inputs: $\mathbf{Z}_{S+1} = \mathcal{F}_\theta^{(S+1)}([\mathbf{Z}_{S,1}, \mathbf{Z}_{S,2}, \cdots, \mathbf{Z}_{S,L}]^\intercal)$
  $r_{S,S+1} = \text{mean}\left(\frac{1}{L}\sum_{l=1}^L \mathbf{Z}_{S,l}\mathbf{Z}_{S,L+1}^\intercal\right)$; $\overline{r} \leftarrow \overline{r} + r_{S,S+1}$
  SGD: $\theta \leftarrow \theta + \partial\overline{r}/\partial\theta$
end while

After the normalization, we obtain the leading K eigenvalues $\{\widehat{\sigma_K^{(s)}}\}$ and the corresponding basis functions with $\{\widehat{\psi_\theta^{(s)}}, \widehat{\psi_\theta^{(s)}}\}$. Since they form a decomposition of density ratios, with a sufficiently large bandwidth K, local-level density ratios can be approximated by

$$\widehat{\rho_{s,s+1}} = \widehat{\phi_\theta^{(s)}}^{\intercal} \widehat{\mathbf{\Sigma}_s}^{\frac{1}{2}} \widehat{\psi_\theta^{(s)}} \to \rho_{s,s+1}, \;\; 1 \le s \le S-1; \;\; \sum_{s=1}^{S-1} \log \widehat{\rho_{s,s+1}} \to \log \overline{\rho}. \tag{10}$$

This procedure will produce the spectrum, the basis functions, and the approximated density ratio $\widehat{\rho_{s,s+1}}$ at each scale s. A thorough pseudocode for this procedure is presented in Algorithm 4.

A.3.4 Generate Local Response of Density Ratios

Here, we further illustrate the procedure for producing figures associated with telescoping density ratios, as shown in Section 5 of the paper. Based on the estimated density ratios, we are able to capture top-down information from the upper layers down to the bottom layers, by propagating density ratios layer by layer. To accomplish this, we establish a sequence of functions $\varrho_s : \mathcal{X}^{(s)} \to \mathbb{R}$ through a recursive procedure. Each function in this series is tasked with evaluating global-to-local dependence relationships at respective local scales.

As we apply each convolution layer to the feature maps derived from its preceding layer, we fix $\mathbf{Z}_s(i,j)$ at the lower scale s and identify the corresponding elements at the higher scale $s+1$ that share a mapping relationship with this element $\mathbf{Z}_s(i,j)$. For this purpose, we define four coordinates:

$$I_s = [\max(0, i - \Delta H_s + 1), \min(i, H_{s+1} - 1)], J_s = [\max(0, j - \Delta W_s + 1), \min(j, W_{s+1} - 1)]. \tag{11}$$

Elements at the higher scale $s+1$ that share a mapping relationship with this element from the lower scale are located within the bounding box defined by these coordinates. Denote the density ratio between receptive fields of $\mathbf{Z}^{(s)}(i',j')$ and $\mathbf{Z}^{(s+1)}(i,j)$ as $\widehat{\rho_{s,s+1}}((i',j'),(i,j))$. Starting with $\varrho_S = 1$, we then implement a recursive procedure:

$$\varrho_s(i',j') = \sum_{i \in I_s} \sum_{j \in J_s} \varrho_{s+1}(i,j) \widehat{\rho_{s,s+1}}((i',j'),(i,j)), 1 \le i' \le H_s, 1 \le j' \le W_s. \tag{12}$$

For each $1 \le s \le S-1$, ϱ_s is a heatmap with dimensions $W_s \times H_s$. It localizes the pattern with the most significant statistical dependence to the global scale, as viewed from local scales. We provide the pseudocode for this procedure in Algorithm 5.

Algorithm 4 Retrieve eigenspectrum and basis functions

Given any input $\mathbf{X}$; $\mathbf{Z}_1 = \mathcal{F}_\theta^{(1)}(\mathbf{X})$
for $s = 1, \cdots, S-1$ **do**
  $\mathbf{Z}_{s+1} = \mathcal{F}_\theta^{(s+1)}(\mathbf{Z}_s)$
  SVD: $(\widehat{\mathbf{R}}_\phi^{(s)})^{-\frac{1}{2}} \widehat{\mathbf{P}}_{\phi,\psi}^{(s)} (\widehat{\mathbf{R}}_\psi^{(s)})^{-\frac{1}{2}} = \mathbf{U}_s \widehat{\mathbf{\Sigma}_s}^{\frac{1}{2}} \mathbf{V}_s^{\intercal}$
  $\widehat{\phi_\theta^{(s)}}(\mathbf{X}) := \mathbf{U}_s^{\intercal} (\widehat{\mathbf{R}}_\phi^{(s)})^{-\frac{1}{2}} \mathbf{Z}_s$
  $\widehat{\psi_\theta^{(s)}}(\mathbf{X}) := \mathbf{V}_s^{\intercal} (\widehat{\mathbf{R}}_\psi^{(s)})^{-\frac{1}{2}} \mathbf{Z}_{s+1}$
  $\widehat{\rho_{s,s+1}} = \widehat{\phi_\theta^{(s)}}^{\intercal} \widehat{\mathbf{\Sigma}_s}^{\frac{1}{2}} \widehat{\psi_\theta^{(s)}}$
  Eigenspectrum (dependence measure): $\widehat{\mathbf{\Sigma}_s} = \text{diag}([\widehat{\sigma_1^{(s)}}, \cdots, \widehat{\sigma_K^{(s)}}])$
  Basis functions (features): $\{\widehat{\phi_\theta^{(s)}}, \widehat{\psi_\theta^{(s)}}\}$
  Density ratio approximations: $\widehat{\rho_{s,s+1}}$
end for

A.4 Additional Visualization Results

In the main body of the paper, we illustrated the null-space projections caused by adding supervision, and the similarity between supervised and self-supervised scenarios, by visualizing the cross-layer

Algorithm 5 Generate local response of density ratios

Given any input $\mathbf{X}$; $\mathbf{Z}_1 = \mathcal{F}_\theta^{(1)}(\mathbf{X})$
for $s = 1, \cdots, S-1$ **do**
  $\mathbf{Z}_{s+1} = \mathcal{F}_\theta^{(s+1)}(\mathbf{Z}_s)$
  Obtain density ratio approximations: $\widehat{\rho_{s,s+1}} : \mathcal{X}_s \times \mathcal{X}_{s+1} \to \mathbb{R}$.
end for
Initialize $\varrho_S = 1$; Initialize each ρ_s to be a (H_s, W_s) heatmap.
for $s = S-1, \cdots, 1$ **do**
  For every $\mathbf{Z}_s(i,j)$, find its mapped counterparts in $\mathbf{Z}_{s+1}$: $\{\mathbf{Z}_{s+1}(i',j'), i' \in I, j' \in J\}$
  $\varrho_s(i,j) = \sum_{i \in I} \sum_{j \in J} \varrho_{s+1}(i',j') \widehat{\rho_{s,s+1}}((i,j),(i',j')), 1 \le i \le H_s, 1 \le j \le W_s$
end for

and cross-supervision spectrum. We have also depicted local density ratio responses for nine CIFAR10 samples at layer L2. In addition to these two experiments, we include additional figures as supplementary.

A.4.1 Learning Dynamics of Eigenvalues

In addition to the spectrum obtained after training the model (corresponding to Fig. 4 and Fig. 5 in the main paper), we also visualize the learning dynamics during training, as seen in Fig. 6. In these figures, we represent the learning dynamics of the cross-layer spectrum as heatmaps between all neighboring layers ($\mathbf{L1}, \cdots, \mathbf{L9}, \mathbf{LF}$) for three types of supervision. The x-axis of each figure represents $50,000$ training iterations, and the y-axis represents the top 20 eigenvalues. This tracks the evolution of eigenvalues over $50,000$ iterations.

The visualized learning dynamics reveal additional insights. Note that in the heatmap, the color intensity represents the magnitude of eigenvalues. Notably, in both supervised and self-supervised scenarios, the middle layers ($\mathbf{L3}$ through $\mathbf{L8}$) consistently display lighter color at the beginning of training, which indicates there is an initial rise in leading eigenvalues. This increase indicates the stage of capturing the dataset's intrinsic dimensions in the middle layers.

However, as training advances, these leading eigenvalues decrease, suggesting the occurrence of null space projections triggered by the added supervision. Interestingly, this phenomenon is not observed in the unsupervised scenario. This pattern of an initial increase followed by a decrease in dependencies indicates the two phases in the learning dynamics, corresponding to the internal and external sources of dependencies. This behavior is also absent in the initial and final layers since they directly interact with inputs and targets. This effect could be crucial for a better understanding of neural networks.

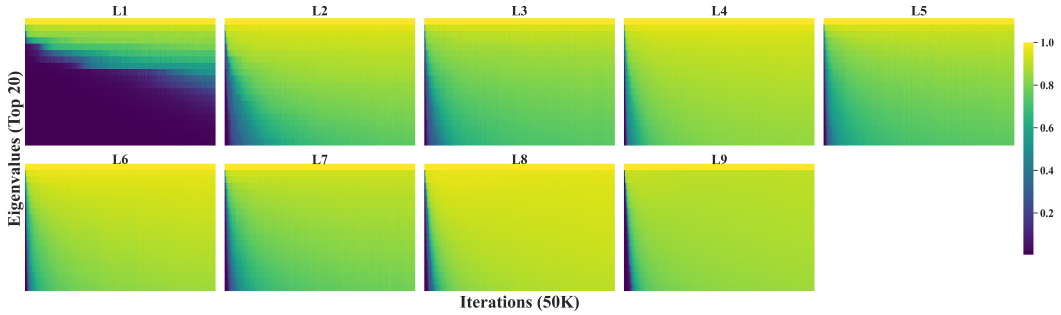
The figure also highlights the differences between supervised and self-supervised learning. In the middle layers, observe that self-supervised learning only has a marginal decrease after the initial stage. In contrast, supervised learning exhibits a more significant and enduring decrease in eigenvalues, indicating a greater influence of null space projections and a diminished ability to capture the intrinsic dimensions of the data, compared to self-supervised learning.

Furthermore, this visualization also uncovers insights beyond the scope of a single-variate dependence measure. Observe that in the supervised scenario, the leading eigenvalues at the final layer are nearly the maximal value 1, and exhibit more instability with noise. These leading eigenvalues at the final layer appear to propagate to lower layers, suppressing the smaller eigenvalues in the spectrum. This causes the discrimination between small and large eigenvalues in the middle layer, which is necessary for the network to perform the classification task. The spectrum for the self-supervised scenario is much more stable, smooth, and does not exhibit such a discrimination effect between large and small eigenvalues.

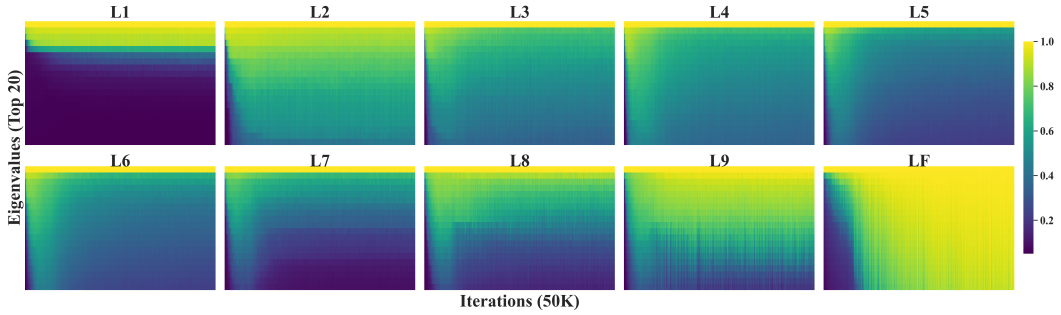
A.4.2 Basis Functions

Another critical component in our proposed decomposition scheme is the eigenfunctions associated with the visualized spectrum, obtained through Algorithm 4.

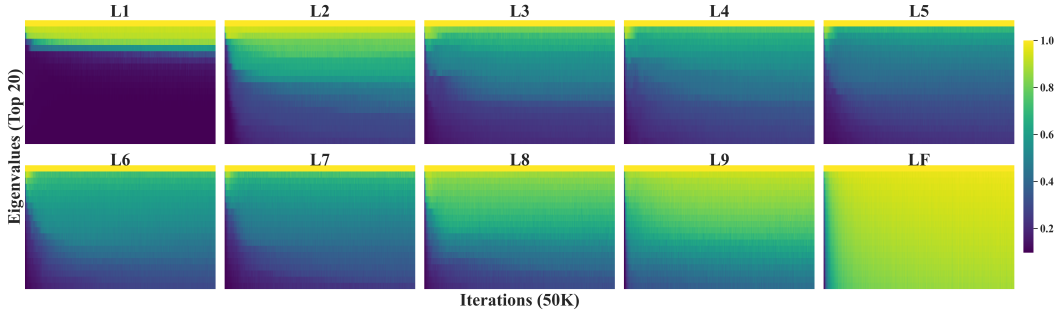
By normalizing the feature maps of each layer, each element in the normalized feature maps can be viewed as the corresponding eigenfunction on the respective field in the images, at the particular hierarchical scale. Visualizing these normalized feature maps allows us to observe the evolution of basis functions across the spatial domain. In Fig. 7, we visualize the evolution of the full 64 eigenfunctions for one selected car sample from CIFAR10 at layer $\mathbf{L2}$, $\mathbf{L4}$ and $\mathbf{L6}$ in the unsupervised scenario.



(a) Unsupervised scenario

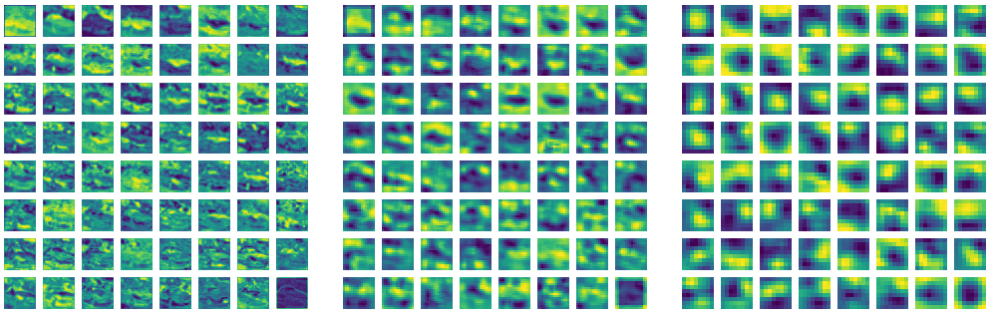


(b) Supervised scenario



(c) Self-supervised scenario

Figure 6: Visualizing cross-layer learning dynamics. Each supervision method comprises 10 figures, corresponding to one of 10 layers (from **L1** through **L9**, and **LF**), except for the unsupervised case without the final layer. The x-axis of each figure represents $50,000$ training iterations, and the y-axis represents the top 20 eigenvalues. The presented heatmaps illustrate the evolution of these 20 eigenvalues throughout $50,000$ iterations. The colors in the heatmap correspond to the eigenvalue magnitudes, with lighter areas indicating larger values. In both supervised and self-supervised scenarios, the middle layers display an initial rise in leading eigenvalues, followed by a decrease, suggesting a two-phase learning dynamic: initially capturing the dataset's intrinsic dimensions and then executing null-space projections essential for classification discrimination.



(a) Layer **L2** (b) Layer **L4** (c) Layer **L6**

Figure 7: Visualizing basis functions for one car sample from CIFAR10, at layers **L2**, **L4** and **L6**, in the unsupervised scenario, each representing a set of orthonormal features at a hierarchical scale.

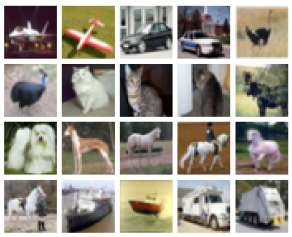
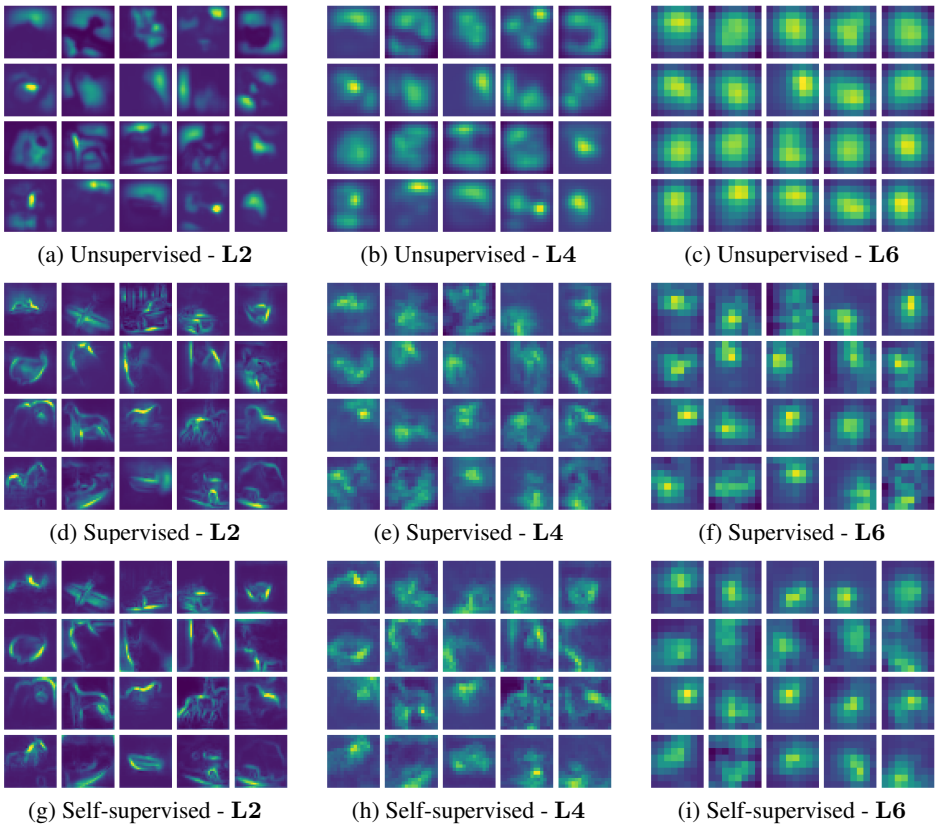


Figure 8: The 20 samples from CIFAR10 we use for comparing local density ratio responses in Fig. 4.



(a) Unsupervised - **L2** (b) Unsupervised - **L4** (c) Unsupervised - **L6**

(d) Supervised - **L2** (e) Supervised - **L4** (f) Supervised - **L6**

(g) Self-supervised - **L2** (h) Self-supervised - **L4** (i) Self-supervised - **L6**

Figure 9: Local density ratio responses are compared for 20 samples from CIFAR10 at layers **L2**, **L4** and **L6**, under all three supervision scenarios. The responses reveal consistent patterns across different network layers. The procedure of propagating density ratios effectively allows bottom layers, which are conventionally viewed as having high-resolution but low-level information, to reflect high-level information. This comparison also further confirms the close resemblance between supervised and self-supervised learning, not only at the bottom layers but across all internal layers.

Fig. 7 validates the diversity of the learned basis functions, demonstrating their effectiveness as features at various hierarchical scales. Another observation, which aligns with conventional FMCA, is that the decomposition involving density ratios also adheres to the principles of any orthonormal decomposition. The spatial domain evolution incorporates both high and low frequencies. Generally, basis functions for large eigenvalues tend to represent low-frequency components, and those for small eigenvalues represent high-frequency components.

The features at different hierarchy levels also correspond to the range of dependency relationships. Observations indicate that lower hierarchies capture short-range and higher hierarchies capture long-range relationships. Together, they form a diverse image representation.

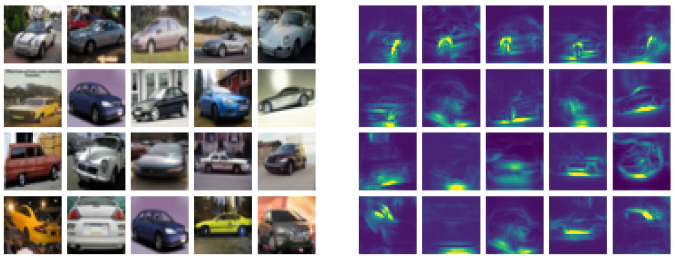
A.4.3 Local Density Ratio Responses

In addition to the 9 samples at layer **L2** shown in the main paper, we illustrate local density ratio responses for 20 additional CIFAR10 samples (Fig. 3) across layers **L2**, **L4** and **L6** in Fig. 4. Their generation procedure has been thoroughly described in Algorithm 5, which involves constructing density ratio approximations with spectrum (Fig. 1) and eigenfunctions (Fig. 2), and propagating density ratios layer by layer from top to bottom.

As discussed in the main paper, adding supervision at the lower layer **L2** effectively highlights the most informative regions in the images, such as object boundaries and interactions between different parts of an object. By examining responses across various layers, we can clearly see consistency in the patterns. The key difference is that upper layers tend to represent high-level object information, capturing long-range dependencies, while lower layers focus more on the details. This approach, instead of relying solely on the network's final layer and ignoring all internal outputs, generates a multi-scale representation. Propagating density ratios from upper to lower layers improves the representation by adding details that upper layers, due to their lower resolution, cannot capture.

Inherent in the conventional neural network topology is the property that the bottom layers have high resolution but only contain low-level information, while the top layers have low resolution, despite capturing high-level information. On the contrary, our generated density ratio responses effectively enable the bottom layers to also contain high-level information.

In addition, we present 20 more car samples visualized at layer **L2** for self-supervised learning in Fig. 5. Interestingly, we find that the most frequent pattern for the car class is the boundary of the wheels and the interactions between the car wheels and the ground. Fig. 5 demonstrates that these patterns are indeed consistent within the class.



(a) Randomly Picked 20 Car Samples (b) Self-supervised - **L2**

Figure 10: Comparing local density ratio responses for 20 samples from CIFAR10 car class, at layer **L2**, in self-supervised scenario.

A.5 Implementation Details

Now we move to the implementation details for the experiments, particularly the used neural network structures and the important hyperparameters for reproducing the experiments.

A.5.1 Network Structure

To obtain the optimal performance for SSL, we adopt the commonly-used ResNet-18 backbone, following the same implementations in [2]. All experiments utilize this ResNet-18 backbone with a consistent batch size of 64, an SGD optimizer, a learning rate of 0.06, and a momentum of 0.9. These settings stay consistent with benchmark models.

To obtain the best visualization for analysis purposes, we modify the backbone to a CNN with blocks. The first block **L0** (see Table 1), employs only 1×1 kernels, transforming input images $\mathbf{X}$ into feature maps $\mathbf{Z}_1$. The blocks **L1** through **L8** (see Table 2), include two layers with 1×1 kernels and one layer with a 3×3 kernel in between, making it a universal approximator for 3×3 patches. The block **L9** (see Table 3), has an additional average pool at the end, such that the output dimension is 1.

The block responsible for constructing the external cost, pertinent to both performance and spectrum analysis, utilizes the network described in Table 4. The input for this block is the concatenation (in the channel dimensions) of feature maps for a group of 9 samples. For the best performance in SSL, we enhance the feature dimensions from 64 to 128.

Blocks $\mathbf{L0}$ to $\mathbf{LF}$ for spectrum analysis are optimized using an Adam optimizer with a learning rate of 0.0001, $\beta_1 = 0.5$, $\beta_2 = 0.9$, and a batch size of 32.

Layer	In Ch.	Out Ch.	Kernel Size	Padding	Input	Output
PaddingLayer	64	64	-	1	$\mathbf{X}$	-
NoiseChannel	64	84	-	-	-	-
Conv2D	84	200	1x1	0	-	-
BatchNorm2d	200	200	-	-	-	-
ReLU	200	200	-	-	-	-
Conv2D	200	200	1x1	0	-	-
BatchNorm2d	200	200	-	-	-	-
ReLU	200	200	-	-	-	-
Conv2D	200	64	1x1	0	-	-
BatchNorm2d	64	64	-	-	-	-
Sigmoid	64	64	-	-	-	$\mathbf{Z}_1$

Table 1: Blocks $\mathbf{L0}$. All convolution kernels are 1×1. No internal costs.

Layer	In Ch.	Out Ch.	Kernel Size	Padding	Input	Output
PaddingLayer	84	84	-	1	-	$\mathbf{Z}_s$
NoiseChannel	64	84	-	-	$\mathbf{Z}_s$	-
Conv2D	84	200	1x1	0	-	-
BatchNorm2d	200	200	-	-	-	-
ReLU	200	200	-	-	-	-
Conv2D	200	200	3x3	0	-	-
BatchNorm2d	200	200	-	-	-	-
ReLU	200	200	-	-	-	-
Conv2D	200	64	1x1	0	-	-
BatchNorm2d	64	64	-	-	-	-
Sigmoid	64	64	-	-	-	$\mathbf{Z}_{s+1}$

Table 2: Blocks $\mathbf{L1}$ through $\mathbf{L8}$. The second convolution has a kernel of 3×3. Internal costs between $\mathbf{Z}_s$ and $\mathbf{Z}_{s+1}$. Average pooling between every two blocks.

Layer	In Ch.	Out Ch.	Kernel Size	Padding	Input	Output
PaddingLayer	64	64	-	1	–	$\mathbf{Z}_8$
NoiseChannel	64	84	-	-	$\mathbf{Z}_8$	-
Conv2D	84	200	1x1	0	-	-
BatchNorm2d	200	200	-	-	-	-
ReLU	200	200	-	-	-	-
Conv2D	200	200	1x1	0	-	-
BatchNorm2d	200	200	-	-	-	-
ReLU	200	200	-	-	-	-
Conv2D	200	64	1x1	0	-	-
BatchNorm2d	64	64	-	-	-	-
Sigmoid	64	64	-	-	-	–
AvgPool	64	64	4x4	-	-	$\mathbf{Z}_9$

Table 3: Blocks $\mathbf{L9}$. The final layer is an average pooling such that the output dimension is 1. Internal costs between $\mathbf{Z}_8$ and $\mathbf{Z}_9$.

Layer	In Ch.	Out Ch.	Kernel Size	Padding	Input	Output
NoiseChannel	64	84	-	-	$\mathbf{Z}_{9,1}, \mathbf{Z}_{9,2}, \cdots, \mathbf{Z}_{9,9}$	-
Conv2D	84	200	1x1	0	-	-
BatchNorm2d	200	200	-	-	-	-
ReLU	200	200	-	-	-	-
Conv2D	200	200	3x3	0	-	-
BatchNorm2d	200	200	-	-	-	-
ReLU	200	200	-	-	-	-
Conv2D	200	64	1x1	0	-	-
BatchNorm2d	64	64	-	-	-	-
Sigmoid	64	64	-	-	-	$\mathbf{Z}_{10}$

Table 4: External block $\mathbf{LF}$. Its input is the concatenation of feature maps for a group of 9 samples. A batch contains multiple such groups. External costs between $\mathbf{Z}_{9,1}, \cdots, \mathbf{Z}_{9,9}$ and $\mathbf{Z}_{10}$.

Here we address several important considerations:

- Noise Channel: At the beginning of each block, we add an additional 20 uniform noise channels, matching the size of the input feature maps. This significantly stabilizes the learning dynamics of eigenvalues.
- Padding Layer: We assume all 3×3 convolutions are performed without padding. Padding is applied to the image at the start of each block, and the output of the padding layer is treated as $\mathbf{Z}_s$, used to calculate the costs with the output of this block $\mathbf{Z}s + 1$. The calculations of ACFs and CCFs take the paddings into account.
- Average Pooling: The input size is retained within each block. Average pooling with a kernel size of 2 is applied every two blocks from $\mathbf{L1}$ to $\mathbf{L8}$. Layer $\mathbf{L9}$ contains a 4×4 average pooling operation at the end of the network. We take the output of this pooling operation, which is one-dimensional, as $\mathbf{Z}_9$.

A.5.2 Other Hyperparameters

There are two additional hyperparameters that are crucial for our experiments:

- Regularization parameter: Each time we compute the inverse of any ACFs (e.g., gradient estimation in Algorithm 2), similar to the pseudo-inverse, we add a small diagonal matrix, scaled by a regularization parameter, denoted as $\lambda\mathbf{I}$. This ensures the invertibility of the matrices. We found this constant to be important, and it may impact the learned spectrum. To achieve optimal performance in SSL, we select $\lambda = 0.1$. For spectrum analysis, we choose $\lambda = 0.001$.
- Smoothness of adaptive filters (as referred to Algorithm 2): In addition to Adam, we also add adaptive estimators for the ACFs. As the gradient of the log-determinant involves the matrix inverse, we replace the inverse matrices with the estimated ones. Similarly to Adam, we control the smoothness through a parameter β, as shown in Algorithm 2. To achieve optimal performance in SSL, we simply set $\beta = 0$. For spectrum analysis, we choose $\beta = 0.5$.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[2] Igor Susmelj, Matthias Heller, Philipp Wirth, Jeremy Prescott, and Malte Ebner et al. Lightly. *GitHub. Note: https://github.com/lightly-ai/lightly*, 2020.