Task-aware Distributed Source Coding under Dynamic Bandwidth

Anonymous Authors[1]

Abstract

Efficient compression of correlated data is essential to minimize communication overload in multi-sensor networks. Each sensor independently compresses the data and transmits them to a central node due to limited bandwidth. A decoder at the central node decompresses and passes the data to a pre-trained machine learning-based task to generate the final output. Thus, it is important to compress the features that are relevant to the task. Additionally, the final performance depends heavily on the total available bandwidth. In practice, it is common to encounter varying availability in bandwidth, and higher bandwidth results in better performance of the task. We design a novel distributed compression framework composed of independent encoders and a joint decoder, which we call neural distributed principal component analysis (NDPCA). NDPCA flexibly compresses data from multiple sources to any available bandwidth with a single model, reducing computing and storage overhead. NDPCA achieves this by learning low-rank task representations and efficiently distributing bandwidth among sensors, thus providing a graceful trade-off between performance and bandwidth. Experiments show that NDPCA improves the accuracy of object detection tasks on satellite imagery by 14% compared to an autoencoder with uniform bandwidth allocation. [1]

1. Introduction

Efficient data compression plays a pivotal role in multi-sensor networks to minimize communication overload. Due to the limited bandwidth of such networks, it is often impractical to transmit all sensor data to a central server, and compressing data is necessary. In many cases, the sensors, so-called sources, observe correlated data, which are only processed by a downstream task, *e.g.*, an object detection model, but not by human eyes. For example, satellites observe overlapping images and transmit them through limited bandwidth to a central server on Earth. Hence, sources should not transmit redundant information from correlated data and only transmit features relevant to the downstream task. It is important to compress each source independently to reduce the communication overload in the network. Literature refers to this setting as distributed source coding. Together, we name the distributed compression of task-relevant features *task-aware distributed source coding*.

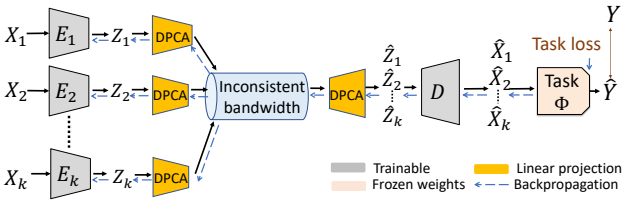


Figure 1: **Task-aware distributed source coding with NDPCA.** $X_1, \ldots, X_k$ are correlated data sources. Encoders $E_1, \ldots, E_k$ independently compress data to latent representations $Z_1, \ldots, Z_k$. The DPCA module allocates the bandwidth of sources based on the importance of the task Φ. The goal is to find the optimal encoders and decoder that minimize the final task loss.

However, existing compression methods fail to combine three aspects: 1. Existing distributed compression methods perform poorly in the presence of a task model. Although neural networks have been shown to be capable of compressing stereo images 2; 3 and correlated images 42, existing methods focus on reconstructing data, but not for downstream tasks. 2. Existing task-aware compression methods cannot take advantage of the correlation of sources. Previous works only consider compressing task-relevant features of single source 10; 23; 12; 31; 9, but not multiple correlated sources. 3. All existing methods, especially those based on neural networks, only compress data to a fixed level of compression but not to multiple levels. Thus, they cannot operate in environments with different demands of compression levels. We use the term bandwidth to indicate the information bottleneck in the dimension of transmitted data, and more related works are discussed in Appendix A.

We present neural distributed principal component analysis (NDPCA), a distributed compression framework that transmits task-relevant features at multiple compression

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]An extended version of the paper is under submission at other conferences.

levels. Fig. 1 illustrates the scenario where the central node requires data from all sources, and network bandwidth varies over time. NDPCA consists of neural encoders $E_1, E_2, \ldots, E_K$ that independently compress correlated data $X_1, X_2, \ldots, X_K$ to latent representations $Z_1, Z_2, \ldots, Z_K$. The distributed principal component analysis (DPCA) module compresses these representations to any dimension based on the current bandwidth. At the central node, a neural decoder reconstructs the representations $\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_k$ to $\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_K$ and feeds them into a task. NDPCA combines a neural autoencoder and the DPCA module to generate task-relevant representations that are compressible in limited bandwidths. It achieves an elegant trade-off between the performance of the task and the bandwidth, enabling efficient transmission of data.

Contributions: First, we formulate the task-aware distributed source coding problem (Sec. 2). Second, we provide a theoretical justification for the framework by analyzing the case of a linear compressor and a task (Sec. 3). Third, we propose a task-aware distributed source coding framework, NDPCA, that learns a single model for different levels of compression (Sec. 4). We validate NDPCA with an object detection task of satellite imagery (Sec. 5), resulting in a 14% increase in accuracy compared to an autoencoder with uniform bandwidth allocation.

2. Problem Formulation

Consider a set of K correlated sources. Let $x_i \in \mathbb{R}^{n_i}$ denote the sample from source i where $i \in \{1, 2, \ldots, K\}$. Samples from each source are compressed independently by encoder E_i to a latent representation $z_i \in \mathbb{R}^{m_i}$ such that $\sum_{i=1}^{K} m_i = m$, where m is the total bandwidth available. A joint decoder D receives the representations $\{z_1, z_2, \ldots, z_k\}$ and reconstructs the sources $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k\} = \{D(E_1(x_1)), D(E_2(x_2)), \ldots, D(E_k(x_k)))\}$. In the presence of a task Φ, it takes the reconstructed inputs to compute the final output $\Phi(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k)$. The goal is to find a set of encoders and a decoder such that the task loss $\mathcal{L}_{\text{task}}$ is minimized. We call this problem as *task-aware* distributed source coding, which is the focus of this paper:

$$\begin{aligned} \underset{E_1,\ldots,E_k,D}{\operatorname{argmin}} \quad & \mathcal{L}_{\text{task}}(Y, \hat{Y}) \\ \text{s.t.} \quad & Y = \Phi(x_1, \ldots, x_k), \hat{Y} = \Phi(\hat{x}_1, \ldots, \hat{x}_k)) \\ & \text{(\textit{Task-aware} distributed source coding),} \end{aligned} \tag{1}$$

where $\mathcal{L}_{\text{task}}$ is the task loss, *e.g.*, the difference of bounding boxes when the task is object detection.

Bandwidth allocation: In the previous formulations, we assume that the output dimensions of encoders are known a priori. However, the dimensions determine the compression of each encoder, which is also a design factor. That is, given the total available bandwidth m, we first need to obtain the optimal m_i for each source i, then, we can design the optimal encoders and decoder accordingly. Finding the optimal set of bandwidths for a given task is a long-standing open problem, even for the simple task of a modulo-two sum of two binary sources 27. Also, existing works 42; 39; 30 largely assume a fixed latent dimension for sources and train different models for different total available bandwidth m, which is, of course, suboptimal. NDPCA provides heuristics to the underlying key challenge of optimally allocating available bandwidth, *i.e.*, deciding m_i, while adapting to different total bandwidths m with a single model.

3. Theoretical Analysis

We start with a motivating example of task-aware distributed source coding under the constraint of linear encoders, a decoder, and a linear task.

DPCA: We consider a linear task for two sources, defined by the task matrix $\Phi \in \mathbb{R}^{p \times (n_1+n_2)}$, where the sources $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{n_2}$ are of dimensions n_1 and n_2, respectively, and the task output is given by $y = \Phi x \in \mathbb{R}^p$, where $x = [x_1^\top, x_2^\top]^\top$. Without loss of generality, we assume the sources to be zero-mean. Now, we have N observations of two sources $X_1 \in \mathbb{R}^{n_1 \times N}$ and $X_2 \in \mathbb{R}^{n_2 \times N}$ and their corresponding task outputs $Y = \Phi(X) \in \mathbb{R}^{p \times N}$, where $X = [X_1^\top X_2^\top]^\top$. We aim to design the optimal linear encoding matrices (encoders) $E_1 \in \mathbb{R}^{m_1 \times n_1}$, $E_2 \in \mathbb{R}^{m_2 \times n_2}$, and the decoding matrix (decoder) $D \in \mathbb{R}^{(n_1+n_2) \times (m_1+m_2)}$ that minimizes the task loss defined as the Frobenius norm of $\Phi(X) - \Phi(\hat{X})$, where $\hat{X}$ is the reconstructed X. For now, we assume that m_1 and m_2 are given. Letting $Z_1 = E_1 X_1 \in \mathbb{R}^{m_1 \times N}$ and $Z_2 = E_2 X_2 \in \mathbb{R}^{m_2 \times N}$ denote the encoded representations and $M = \Phi D$ denote the product of the task and decoder matrices, we solve the optimization problem:

$$E_1^*, E_2^*, M^* = \underset{E_1, E_2, M}{\operatorname{argmin}} \quad \|Y - MZ\|_2^2 \tag{2a}$$

$$\text{s.t.} \quad Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} E_1 X_1 \\ E_2 \tilde{X}_2 \end{bmatrix}, \tag{2b}$$

$$ZZ^\top = \mathbb{I}_m, \tag{2c}$$

$$\hat{Y} = MZ, \quad Y = \Phi \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}. \tag{2d}$$

Note that solving M is identical to solving the decoder D since we can always convert M to D by the generalized inverse of task Φ. We constrain the representations to be orthonormal vectors in (2c) as in the normalization in principal component analysis (PCA) for the compression of a single source 24.

We discuss the detailed solution of DPCA in Appendix B and provide a brief summary here. First, a preprocessing step removes the correlation part of X_1 from X_2 by subtracting the least-square estimator $\hat{X}_2(X_1)$:

$$\tilde{X}_2 = X_2 - \hat{X}_2(X_1) = X_2 - X_2 X_1^\top (X_1 X_1^\top)^{-1} X_1. \quad (3)$$

The orthogonality principle of least-square estimators 25 ensures that $X_1 \tilde{X}_2^\top = \mathbf{0}_{n_1 \times n_2}$. Then, we can decouple the original problem in (2) into subproblems where each problem is a encoding problem for one encoder, which can be solved by canonical correlation analysis 19.

Dynamic bandwidth: We extend our approach to determine the optimal bandwidth allocation given a total bandwidth m. By solving DPCA with $m_1 = n_1$ and $m_2 = n_2$, we obtain optimal encoders E_1^* and E_2^*, as well as pairs of canonical directions and correlations. Similar to PCA, these pairs can be seen as a generalization of singular vectors and values, and the sums of squares of canonical correlations are the optimal values of (2). Sorting the canonical correlations in descending order, we select the first m pairs of canonical correlations and directions. These canonical correlations determines the optimal encoders E_1^*, E_2^*, and decoder D^*, indirectly solving for m_1 and m_2. The importance of a direction to the task is indicated by the canonical correlations, so we prioritize the transmission of important directions. The same approach can be easily extended to compressing more than 2 sources.

Performance analysis of DPCA: When DPCA compresses new data matrices with encoder E_1^* and E_2^*, the preprocessing step (3) is invalid as the encoders cannot communicate with each other. So for DPCA to perform optimally while skipping the step, the two data matrices need to be uncorrelated, namely, $\hat{X}_2(X_1) = 0$, because in such case the preprocessing step removes nothing from the data sources. Given that correlated sources lead to suboptimality of DPCA, we characterize the performance between the joint compression, PCA, and the distributed compression, DPCA, under the same bandwidth in Lemma C.1 with the simplest case of reconstruction, namely, $\Phi = \mathbb{I}_p$. In this case, the canonical correlation analysis is relaxed to the singular value decomposition, which is later used for NDPCA in Sec. 4. Lemma C.1 concludes that as the covariance decreases, DPCA performs more closely to PCA, the optimal joint compression.

4. Neural Distributed Principal Component Analysis

Theoretical analysis reveals that DPCA has limitations: it optimally compresses data only when sources are uncorrelated and is limited to linear tasks. However, DPCA dynamically allocates bandwidth based on source importance. Neural autoencoders, on the other hand, excel at fixed-dimension compression but lack dynamic bandwidth allocation. To address this contrast, we propose neural distributed principal component analysis (NDPCA), which combines a neural autoencoder and DPCA. NDPCA adapts to any bandwidth and flexibly allocates it based on the importance of the source to the task. This integrated approach enables efficient compression and optimal bandwidth allocation. NDPCA has two encoding stages, as shown in Fig. 1: First, the neural encoder at each i-th source encodes data X_i to a fixed-dimensional representation Z_i for $i \in [K]$. Then, DPCA adapts the dimension of Z_i via linear matrices according to the available bandwidth as per Sec. 3. Similarly, the decoding of NDPCA is also performed in two stages. First, the DPCA linear decoder reconstructs the K fixed-dimensional representations $\hat{Z}_1, \ldots, \hat{Z}_K$, based on which the joint neural decoder generates the estimate of data $\hat{X}_1, \ldots, \hat{X}_K$. These estimates are then passed to the neural task model Φ to obtain the final task output $\hat{Y}$. Since we have a non-linear task model here, DPCA mainly adapts the dimension appropriately as needed; the role of the DPCA here is to reliability reconstruct the embedding $\hat{Z}$, which corresponds to the case described in Lemma C.1 with the task matrix Φ as identity.

Training procedure: During the training of NDPCA, we assume the task model is pre-trained and we do not update its wrights. We aim to learn the K neural encoders and the joint neural decoder which minimize the loss function:

$$\mathcal{L}_{\text{tot}} = \lambda_{\text{task}} \underbrace{\|\hat{Y} - Y\|_F^2}_{\text{task loss}} + \lambda_{\text{rec}} \underbrace{\sum_{i=1}^{K} \|\hat{X}_i - X_i\|_F^2}_{\text{reconstruction loss}}. \quad (4)$$

In the task-aware setting when $\lambda_{\text{rec}} = 0$, the neural autoencoder fully restores task-relevant features, which is the main focus of this paper. When $\lambda_{\text{task}} = 0$, the neural autoencoder learns to reconstruct the data X, which is the task-agnostic setting later compared in Sec. 5.

To encourage NDPCA to work well under *various available bandwidths* with DPCA during the training phase, we need uncorrelatedness from the limitations of the DPCA. To compress representations with a few singular vectors and make NDPCA more bandwidth efficient, we need *linear compressibility*. That is, encouraging the neural autoencoder to generate low-rank representations. We tried to explicitly encourage the desired properties with additional terms in (4), but they all adversely affect the task performance. We tried to use the cosine similarity to generate uncorrelated representations as per 4; 36; 6; 8, and the convex low-rank approximation, nuclear norm, to increase linear compressibility, as per 35; 16. For the comparison of the resulting performance, see Appendix H.1. In this regard, we propose a novel linear compression module that allows us to adapt to DPCA during training rather than using additional terms in the loss. We introduce a *random-dimension* DPCA projection module to improve performance in lower band-

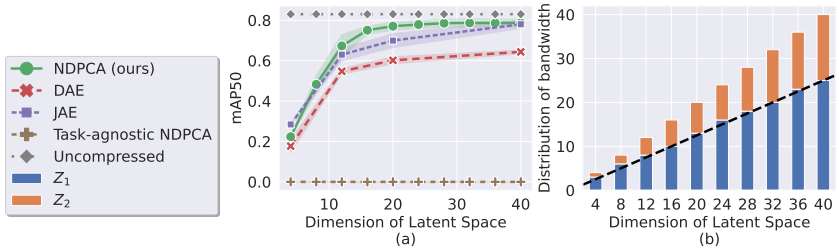


Figure 2: (a) Our method achieves equal or higher performance than other methods. (b) Distribution of total available bandwidth among the two views for NDPCA (ours). The unequal allocation highlights the difference in the importance of the views for a given task.

widths. It projects representations Z to a low dimension randomly chosen, simulating projections in various available bandwidths during inference. It can be interpreted as a differentiable singular value decomposition with a random dimension, described in Alg. 1. Note that no retraining is needed for different bandwidths, and only the storage of a neural autoencoder and a linear matrix at each encoder and decoder is needed.

5. Experiments

We consider three different tasks to test our framework: (a) the denoising of CIFAR-10 images 28, (b) multi-view robotic arm manipulation 41, referred to as the *locate and lift* task, and (c) object detection on satellite imagery 13. Here, we only show the most representative one–object detection from satellite images-and describe other experiments in detail in Appendix E. We assume that there are two data sources, referred to as *views*, each containing partial information relevant to the task. We refer to our proposed method, task-aware NDPCA, as NDPCA for simplicity. NDPCA includes a single autoencoder with a large dimension of representations $Z \in \mathbb{R}^{80}$. The object detection task considers using satellite imagery to locate Airbuses where satellites observe overlapping images of an airport and transmit data to Earth through limited bandwidth. We crop all images in the dataset into smaller pieces (224×224 pixels). The two data sources are the upper 160 pixels (source 1) and the lower 104 pixels of the image (source 2) with 40 pixels overlapped. Our object detection model follows the paper "You Only Look Once" (Yolo) 33. The task loss here is the difference between object detection loss with and without compression.

Baselines: We compare NDPCA against three baselines: (a) Task-aware joint autoencoder (JAE), where a single pair of encoder and decoder compresses both views. JAE is considered an upper bound of NDPCA since it can leverage the correlation between both views while avoiding encoding redundant information. (b) Task-aware vanilla distributed autoencoder (DAE), where two encoders independently encode one view to equal bandwidths and a joint decoder decodes the data. DAE is considered a lower bound of NDPCA since both encoders utilize the same bandwidth regardless of the importance of the views for the task, while NDPCA allocates bandwidths in a task-aware manner. (c) Task-agnostic NDPCA, which differs from NDPCA in the training loss of reconstructing the original views.

Results: Our results are: (1) Task-aware NDPCA outperforms task-agnostic NDPCA, and (2) bandwidth allocation should be related to the importance of the task. In Fig. 2(a), we see that task-aware NDPCA performs much better than task-agnostic NDPCA and DAE, which equally allocates bandwidths. We see from Fig. 2(a) that task-aware NDPCA provides a graceful performance degradation with respect to available bandwidth, with no additional training or storage of multiple models. On the other hand, DAE and JAE require retraining for every level of compression, so every sample point in the plot is a different model. We show the uncompressed upper bound in gray dotted lines. NDPCA results in up to 14% gain in mAP50 compared to DAE. In Fig. 2(b), we plotted the ratio of the areas of both views, while equally splitting the overlapping part, in a dashed black line. Surprisingly, NDPCA's empirical allocation of bandwidth is highly aligned with the theoretical ratio, supporting that it captures the importance of the task and allocates bandwidth according to it. We highlight the performance gap between JAE and NDPCA and our limitations later in Appendix E. Also, we show how the weight in (4) affects the resulting images in Appendix G.5.

6. Conclusion

We proposed a theoretically-grounded linear distributed compressor, DPCA, and analyzed its performance compared to the optimal joint compressor. Then, we designed a distributed compression framework called NDPCA by combining a neural autoencoder and DPCA to allocate bandwidth according to their importance to the task. Experiments on Airbus detection showed that NDPCA near-optimally outperforms task-agnostic or equal-bandwidth compression schemes. Moreover, NDPCA requires only one model and does not need to be retrained for different compression levels, which makes it suitable for settings with dynamic bandwidths.

References

Ultralytics yolov8 docs. https://docs.ultralytics.com/, 2023. [Online; accessed 16-April-2023].

J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

A. Bardes, J. Ponce, and Y. LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022.

T. Berger, Z. Zhang, and H. Viswanathan. The CEO problem. *IEEE Transactions on Information Theory*, 42(3):887–902, 1996. doi: 10.1109/18.490552.

K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, 2016.

A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. ISSN 2078-2489. doi: 10.3390/info11020125.

R. T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018.

J. Cheng, M. Pavone, S. Katti, S. Chinchali, and A. Tang. Data sharing and compression for cooperative networked control. In *Advances in Neural Information Processing Systems*, 2021.

J. Cheng, S. Chinchali, and A. Tang. Task-aware network coding over butterfly network. *arXiv preprint arXiv:2201.11917*, 2022a.

J. Cheng, A. Tang, and S. Chinchali. Task-aware privacy preservation for multi-dimensional data. In *Proceedings of the 39th International Conference on Machine Learning*, 2022b.

J. Choi and B. Han. Task-aware quantization network for jpeg image compression. In *European Conference on Computer Vision*, 2020. ISBN 978-3-030-58565-5.

A. Defense and S. Intelligence. Airbus aircraft detection. https://www.kaggle.com/datasets/airbusgeo/airbus-aircrafts-sample-dataset, 2021. [Online; accessed 03-April-2023].

E. Diao, J. Ding, and V. Tarokh. DRASIC: Distributed recurrent autoencoder for scalable image compression. In *Data Compression Conference*, 2020. doi: 10.1109/DCC47342.2020.00008.

Y. Dubois, B. Bloem-Reddy, K. Ullrich, and C. J. Maddison. Lossy compression for lossless prediction. In *Advances in Neural Information Processing Systems*, 2021.

M. Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

P. han Li, S. P. Chinchali, and U. Topcu. Differentially private timeseries forecasts for networked control. In *American Control Conference*, 2023.

D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004. doi: 10.1162/0899766042321814.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2019.

E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

R. Ji, H. Yao, W. Liu, X. Sun, and Q. Tian. Task-dependent visual-codebook compression. *IEEE Transactions on Image Processing*, 21(4):2282–2293, 2012. doi: 10.1109/TIP.2011.2176950.

I. Jolliffe. *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_455.

S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, Inc., 1993.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

J. Korner and K. Marton. How to encode the modulo-two sum of binary sources. *IEEE Transactions on Information Theory*, 25(2):219–221, 1979. doi: 10.1109/TIT.1979.1056022.

A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. PhD thesis, University of Toronto, 2009.

W. F. Lo, N. Mital, H. Wu, and D. Gündüz. Collaborative semantic communication for edge inference. *IEEE Wireless Communications Letters*, 2023. doi: 10.1109/LWC.2023.3256006.

N. Mital, E. Özyılkan, A. Garjani, and D. Gündüz. Neural distributed image compression with cross-attention feature alignment. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023. doi: 10.1109/WACV56688.2023.00253.

M. Nakanoya, S. S. Narasimhan, S. Bhat, A. Anemogiannis, A. Datta, S. Katti, S. Chinchali, and M. Pavone. Co-design of communication and machine inference for cloud robotics. *Autonomous Robots*, 2023.

V. Prabhakaran, D. Tse, and K. Ramachandran. Rate region of the quadratic gaussian CEO problem. In *International Symposium on Information Theory*, 2004. doi: 10.1109/ISIT.2004.1365154.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.91.

F. Rellich and J. Berkowitz. *Perturbation theory of eigenvalue problems*. CRC Press, 1969.

M. Salzmann, C. H. Ek, R. Urtasun, and T. Darrell. Factorized orthogonal latent spaces. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

A. Singh, E. Garza, A. Chopra, P. Vepakomma, V. Sharma, and R. Raskar. Decouple-and-sample: Protecting sensitive information in task agnostic data release. In *European Conference on Computer Vision*, 2022. doi: 10.1007/978-3-031-19778-9_29.

D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 1973. doi: 10.1109/TIT.1973.1055037.

F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018. ISBN 9780999241127.

J. Whang, A. Acharya, H. Kim, and A. G. Dimakis. Neural distributed source coding. *arXiv preprint arXiv:2106.02797*, 2021.

M. Ye, C. Gong, L. Nie, D. Zhou, A. Klivans, and Q. Liu. Good subnetworks provably exist: Pruning via greedy forward selection. In *International Conference on Machine Learning*, 2020.

A. Zhan, R. Zhao, L. Pinto, P. Abbeel, and M. Laskin. A framework for efficient robotic manipulation. In *Deep RL Workshop at Advances in Neural Information Processing Systems*, 2022.

X. Zhang, J. Shao, and J. Zhang. LDMIC: Learning-based distributed multi-view image coding. In *The Eleventh International Conference on Learning Representations*, 2023.

R. Zhao, U. Topcu, S. Chinchali, and M. Phielipp. Learning sparse control tasks from pixels by latent nearest-neighbor-guided explorations. *arXiv preprint arXiv: 2302.14242*, 2023.

Appendix

A. Related Work

Information theoretic perspective: Slepian and Wolf *et al.* are the first to obtain the minimum bandwidth of distributed sources to perfectly reconstruct data (Slepian and Wolf, 1973). However, they use exponentially complex compressors while assuming that the joint distribution of sources is known, which is impractical. In the presence of a task, finding the rate region of two binary sources has remained an open problem, even for modulo-two sum tasks 27. In terms of imperfectly reconstructing data with neural autoencoders, previous works consider compression of the original data to a fixed dimension 39; 14, while our work focuses on compressing data to any bandwidth with a task model.

Task-aware compression: Real-world data, such as images or audio, are ubiquitous and high-dimensional, while downstream tasks that input the data only utilize certain features for the output. Task-aware compression aims to compress data while maximizing the performance of a downstream task. Previous works analyze linear task 10, image compression 23; 12; 31; 15, future prediction 9, and data privacy 18; 11, while ours compresses distributed sources under limited bandwidth.

Neural autoencoder: Previous works show the ability of neural autoencoders to generate meaningful and uncorrelated representations. Instead of adding additional loss terms during training like 4; 36; 8; 6; 29, we use a random projection module to help a neural autoencoder learn uncorrelated and linear-compressible representations. Other works focus on designing new neural architectures for multi-view image compression 42; 30, while ours focuses on the framework to compress data to different compression levels. We choose autoencoders instead of variational autoencoders 26; 20 because we focus on the compression of fixed representations rather than generative tasks from latent distributions. Also, autoencoders are more compatible with DPCA than variational autoencoders.

B. Solving DPCA

We now solve the optimization problem in (2). For any given E_1, E_2 (thus, a given Z), we can optimally obtain $M^* = YZ^\top(ZZ^\top)^{-1} = YZ^\top$ by linear regression. Now, we are left to find the optimal encoders E_1, E_2. First, a preprocessing step removes the correlation part of X_1 from X_2 by subtracting the least-square estimator $\hat{X}_2(X_1)$:

$$\tilde{X}_2 = X_2 - \hat{X}_2(X_1) = X_2 - X_2X_1^\top(X_1X_1^\top)^{-1}X_1. \tag{5}$$

The orthogonality principle of least-square estimators 25 ensures that $X_1\tilde{X}_2^\top = \mathbf{0}_{n_1\times n_2}$. We decouple the objective in (2a) with respect to E_1, E_2 by the orthogonality principle and (2c):

$$\begin{aligned}\min_{E1,E2}\|Y - M^*Z\|_2^2 &= \|Y\|_2^2 - \max_{E_1,E_2}\|M^*\|_2^2 \\ &= \|Y\|_2^2 - \max_{E_1}\|Y_1X_1^\top E_1^\top\|_2^2 - \max_{E_2}\|Y_2\tilde{X}_2^\top E_2^\top\|_2^2,\end{aligned} \tag{6}$$

where $Y = \Phi X = \begin{bmatrix}\Phi_1\Phi_2\end{bmatrix}\begin{bmatrix}X_1^\top X_2^\top\end{bmatrix}^\top = Y_1 + Y_2$. We then have two subproblems from (2):

$$\begin{aligned}E_1^* = \underset{E_1}{\operatorname{argmax}} \quad & \|\Phi_1X_1X_1^\top E_1^\top\|_2^2 \\ \text{s.t.} \quad & E_1X_1X_1^\top E_1^\top = \mathbb{I}_{m_1},\end{aligned} \tag{7}$$

$$\begin{aligned}E_2^* = \underset{E_2}{\operatorname{argmax}} \quad & \|\Phi_2\tilde{X}_2\tilde{X}_2^\top E_2^\top\|_2^2 \\ \text{s.t.} \quad & E_2\tilde{X}_2\tilde{X}_2^\top E_2^\top = \mathbb{I}_{m_2}.\end{aligned} \tag{8}$$

The two subproblems are the canonical correlation analysis 19, which can be solved by whitening $E_1X_1, E_2\tilde{X}_2$ and singular value decomposition (see 19 for details).

C. Proof of Lemma

C.1. Bounds of DPCA

Lemma (Bounds of DPCA Reconstruction). Given a zero-mean data matrix and its covariance,

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \in \mathbb{R}^{(n_1+n_2)\times N}, XX^\top = \underbrace{\begin{bmatrix} \mathrm{Cov}_{11} & \mathbf{0} \\ \mathbf{0} & \mathrm{Cov}_{22} \end{bmatrix}}_{X_{\mathrm{diag}}} + \underbrace{\begin{bmatrix} \mathbf{0} & \mathrm{Cov}_{12} \\ \mathrm{Cov}_{21} & \mathbf{0} \end{bmatrix}}_{\Delta X},$$

assume that ΔX is relatively smaller than $XX^\top$, and $XX^\top$ is positive definite with distinct eigenvalues. For PCA's encoding and decoding matrices $E_{\mathrm{PCA}}, D_{\mathrm{PCA}}$ and DPCA's encoding and decoding matrices $E_{\mathrm{DPCA}}, D_{\mathrm{DPCA}}$, the difference of the reconstruction losses is bounded by

$$0 \le \|X - D_{\mathrm{DPCA}}\ E_{\mathrm{DPCA}}(X)\|_2^2 - \|X - D_{\mathrm{PCA}}E_{\mathrm{PCA}}(X)\|_2^2 = -\sum_{i=m+1}^{n_1+n_2} \lambda_i e_i^\top \Delta X e_i.$$

where λ_i and e_i are the i-th largest eigenvalue and eigenvector of $XX^\top$, Tr is the trace function, and m is the dimension of the compression bottleneck.

Proof. The lower bound is intuitive. We know that DPCA cannot outperform PCA since distributed coding cannot outperform joint coding and PCA is the optimal linear encoding. The reconstruction loss of PCA is always not greater than the loss of DPCA, thus the lower bound is 0. Now consider the upper bound:

$$\begin{aligned}
&\|X - D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}X\|_2^2 - \|X - D_{\mathrm{PCA}}E_{\mathrm{PCA}}X\|_2^2 \\
=& \mathrm{Tr}(XX^\top + D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}X\left(D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}X\right)^\top - 2D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}XX^\top) \\
&- \sum_{i=m+1}^{n_1+n_2} \lambda_i(XX^\top) \\
=& \mathrm{Tr}(X_{\mathrm{diag}} + \Delta X + D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}X\left(D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}X\right)^\top - 2D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}XX^\top) \\
&- \sum_{i=m+1}^{n_1+n_2} \lambda_i(XX^\top) \\
=& \mathrm{Tr}(\Delta X + E_{\mathrm{DPCA}}^\top D_{\mathrm{DPCA}}^\top D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}\Delta X - 2D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}\Delta X) \\
&+ \sum_{i=m+1}^{n_1+n_2} \lambda_i(X_{\mathrm{diag}}) - \lambda_i(XX^\top) \\
=& \sum_{i=m+1}^{n_1+n_2} \lambda_i(X_{\mathrm{diag}}) - \lambda_i(XX^\top).
\end{aligned}$$

Finally, we use the matrix perturbation theory 34 to calculate the first-order approximation of the effect of ΔX on the singular values of X_{diag}. The perturbation theory assumes that the perturbation ΔX is relatively small compared to X_{diag}. Then, we know:

$$\begin{aligned}
\|X - D_{\mathrm{DPCA}}E_{\mathrm{DPCA}}X\|_2^2 - \|X - D_{\mathrm{PCA}}E_{\mathrm{PCA}}X\|_2^2 &= \sum_{i=m+1}^{n_1+n_2} \lambda_i(X_{\mathrm{diag}}) - \lambda_i(XX^\top) \\
&\le \sum_{i=m+1}^{n_1+n_2} \lambda_i - \lambda_i - \lambda_i e_i^\top \Delta X e_i \\
&= -\sum_{i=m+1}^{n_1+n_2} \lambda_i e_i^\top \Delta X e_i.
\end{aligned}$$

□

Note that the encoding and decoding matrices of DPCA look like:

$$D_{\text{DPCA}} = \begin{bmatrix} D_1 & \mathbf{0} \\ \mathbf{0} & D_1 \end{bmatrix}, E_{\text{DPCA}} = \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2, \end{bmatrix}$$

where E_1, E_2, D_1, D_2 are matrices obtained from each source with DPCA.

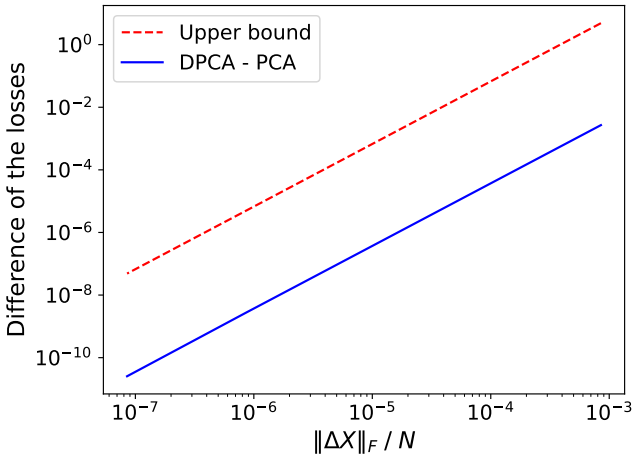


Figure 3: **Bound from Lemma C.1:** The obtained upper bound is always larger than the difference of losses of DPCA and PCA.

We examine the correctness of our bound with random data matrices in Fig. 3. We can see that the gap between DPCA and PCA decreases as the Frobenius norm of ΔX decreases. The upper bound also has the same trend, while it is always larger than the exact value. Note that in Fig. 3, all axes are in log scale.

D. DPCA Module Pseudocode

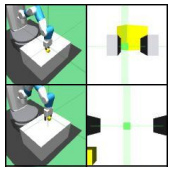
Algorithm 1 Projection into a random low dimension using DPCA

Input: A size b batch of latent representations $Z_i \in \mathbb{R}^{b \times m_i}$ from each source i, min and max bandwidth $m_{\text{min}}, m_{\text{max}}$
Output: Compressed representation Z_i^m of each source, reconstructed representation $\hat{Z}$ for all sources
function ENCODE($Z_i, m_{\text{min}}, m_{\text{max}}$)
  for each source i **do**
    $\bar{Z}_i \leftarrow Z_i - \text{Mean}(Z_i)$ ▷ Normalize representations
    $s_i, V_i, H_i \leftarrow \text{SVD}(\bar{Z}_i)$ ▷ Singular value decomposition
  end for
  $s, V \leftarrow \text{Cat}(s_i), \text{Cat}(V_i)$ ▷ Concatenate singular values and vectors
  $m \leftarrow \text{Rand}(m_{\text{min}}, m_{\text{max}})$ ▷ Randomly choose projection dimension
  $s^m, V^m \leftarrow \arg\max([s, V], m)$ ▷ Select the top m values of s
  for each source i **do**
    $V_i^m \leftarrow \{V | V \in V^m, V \in V_i\}$ ▷ Select m vectors from sources
    $Z_i^m = \bar{Z}_i \times V_i^m$ ▷ Project Z_i to lower dimensions
  end for
  return $Z_{\text{low}} \leftarrow \text{Cat}(Z_i^m)$ ▷ Return Compressed representation
end function
function DECODE(Z_i^m)
  for each source i **do**
    $\hat{\bar{Z}}_i \leftarrow Z_i^m \times \text{Cat}(V_i^m)^\top$ ▷ Decompressed representation
    $\hat{Z}i \leftarrow \hat{\bar{Z}}_i + \text{Mean}(Z_i)$ ▷ Denormalize representations
  end for
  return $\hat{Z} \leftarrow \text{Cat}(\hat{Z}i)$ ▷ Return reconstructed representations
end function

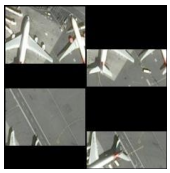
E. Discussion of Other Experiments

We now describe the other two datasets and the corresponding tasks of our additional experiments:

(a) CIFAR-10: view 1 is less corrupted and thus contains more information about the original images.



(b) Locate and lift: Side-view (column 1) faintly captures the absolute position of objects, in contrast to the arm-view (column 2).



(c) Airbus detection: view 1 and view 2 observe different parts of the complete view with overlap.

Figure 4: **Datasets:** (column 1) view 1. (column 2) view 2. In all experiments, both views are correlated, but one view is more important than the other as it contains more information relevant to the task.

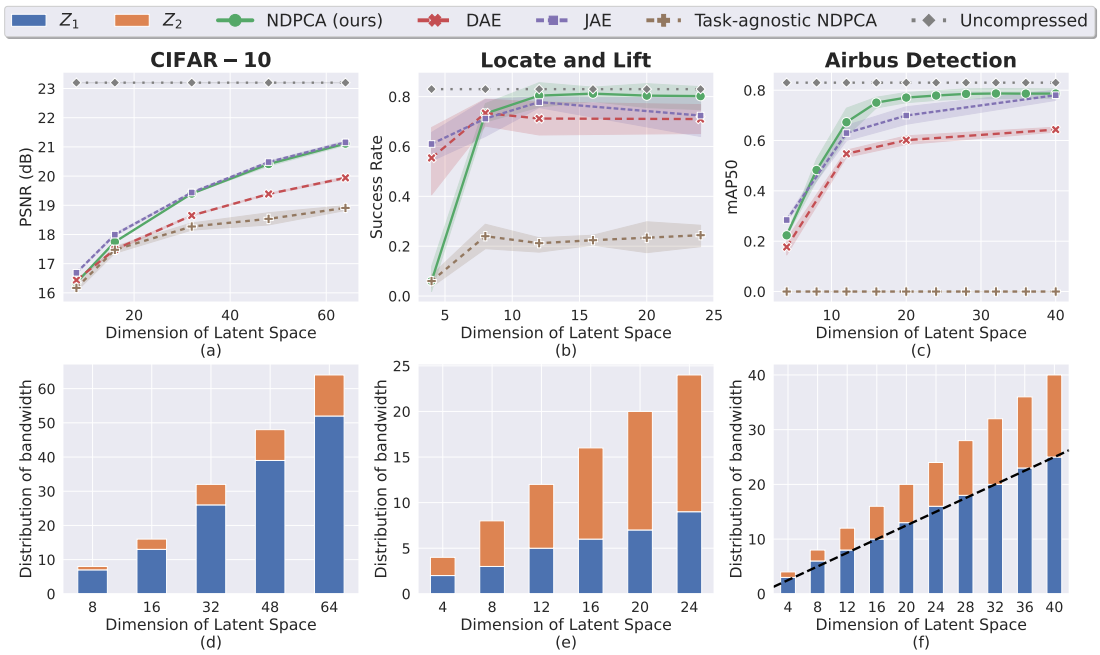


Figure 5: **Top:** Performance Comparison for 3 different tasks. Our method achieves equal or higher performance than other methods. **Bottom:** Distribution of total available bandwidth (latent space) among the two views for NDPCA (ours). The unequal allocation highlights the difference in the importance of the views for a given task.

CIFAR-10 denoising: We first consider a simple task of denoising CIFAR-10 images using two noisy observations of the same image, shown in Fig. 4 (a). Here, the importance of each observation, or view, for the task is simply the noise level. For view 1, we consider an image corrupted with additive white Gaussian noise (AWGN) with a variance of 0.1^2. And view 2 is highly corrupted by AWGN with a variance of 1. All the images were normalized to $[0, 1]$ before adding the noise. We compressed the noisy observations and passed the reconstructed images through a pre-trained denoising network. We then computed the final peak signal-to-noise ratio (PSNR) with respect to the clean image. Since the noise levels of both views are unequal, the importance of the task is unequal as well. The optimal bandwidth allocation should not be equal, thus showing the advantage of NDPCA. Although view 1 contains more information, not all bandwidth should be allocated to view 1. This problem is called the CEO problem 5; 32. In fact, even if one view is highly corrupted, we should still leverage that view and never allocate 0 bandwidth to it.

Locate and lift: For the manipulation task, we consider a scenario in which a simulated 6 degrees-of-freedom robotic arm controlled by a reinforcement learning agent inputs two camera views to locate and lift a yellow brick. We call the view from the robotic arm "arm-view" and the one recording the whole desk "side-view", as shown in Fig. 4 (b). The two views are complementary to completing the task, details discussed in Appendix H.3. We trained the agent in a supervised-learning manner. We collected a dataset of observation and action pairs 43 and trained an agent from the dataset. Then, we defined task loss as the L_2 norm of actions from images with and without compression and trained NDPCA to minimize the task loss

through the agent. Literature calls this training method "behavior cloning" 38 as it learns from demonstrations. Behavior cloning causes a drop in performance, but this paper only focuses on the performance degradation caused by compression, so we treat the behavior cloning agent with uncompressed views as the upper bound of our method.

The results of the other two experiments are:

Fig. 5(a) shows the results of denoising CIFAR-10 with NPDCA trained at $(m_{\min}, m_{\max}) = (8, 64)$. Although view 1 is more important than view 2, DAE can only equally allocates bandwidth to both sources. NDPCA compresses the data and flexibly allocates bandwidths, as shown in 5(d), where we can see that Z_1 has more bandwidth than Z_2. NDPCA results in 1.2 dB gain in PSNR compared to DAE when $m = 64$.

Fig. 5(b) shows the results of the locate and lift task with NPDCA trained at $(m_{\min}, m_{\max}) = (8, 48)$. We set the length of an episode as 50 time steps and measure the success rate in 100 episodes. We show the upper bound, a behavior cloning agent without compression, in gray dotted lines. The arm view is more important as it captures the precise location of the brick, and as expected, NDPCA allocates more bandwidth to the arm-view (Z_2), as seen in Fig. 5(e). We see that NDPCA has a 9% higher success rate compared to DAE when $m = 24$.

Comparison of NDPCA with JAE: JAE uses the information from both views simultaneously to capture the best joint embedding for the task. In an ideal scenario, JAE will be the upper bound for the performance and hence easily performs better than DAE across all the experiments. Interestingly, in Fig. 5(b) and (c), we see that NDPCA outperforms not only DAE but also JAE as well. We attribute it to the better representations present in higher-dimension latent space. It turns out that learning a high-dimensional representation and then projecting to a lower dimension space, like NDPCA, is more efficient compared to directly learning a low-dimensional representation, like JAE. This projection from higher dimensional to lower dimensional is similar to pruning large neural networks to identify effective sparse sub-networks. 17; 40. We also note that Low-Rank Adaptation (LoRA) 22 technique for large language models can be thought of as a similar approach.

Limitations: In general, autoencoders are poor at generalizing to out-of-distribution data and the drawback translates to NDPCA as well. When the testing set is noticeably different from the training set, the performance of NDPCA can get noticeably lower. Additionally, during training, DPCA performs the singular value decomposition in the training set. The decomposition operation can become ill-conditioned and unstable if the batch size is too small. An alternative approach could be a parametric low-rank decomposition such as LoRA 22 or using adapter networks 21, although the complexity increases and the compatibility with DPCA remains to be explored.

F. Details of the Datasets

F.1. CIFAR-10 denoising:

We started with the standard CIFAR-10 dataset and normalized the images to $[0, 1]$. Two different views are created by adding different levels of Gaussian noise, $\mathcal{N}(0, 0.1^2)$ and $\mathcal{N}(0, 1)$. The pre-trained task model is created by training a denoising autoencoder that takes both views, concatenates them along the channel dimension, and produces a clean image. The autoencoders need to learn features that are important for this task model.

F.2. Locate and lift:

We collected $20,000$ pairs of actions and the corresponding images of both views for our training set. The actions are 4 dimensional, controlling the x, y, z coordinate movements and the gripper of the robotic arm. We randomly cropped the images from 128×128 to 112×112 pixels to make our autoencoder more robust. The expert agent is pre-trained by the same data augmentation as well.

F.3. Airbus detection:

We first cropped all original images of 2560×2560 pixels (Fig. 6) into 224×224 pixels with 28 pixels overlapping between each cropped image. We then eliminated the bounding boxes that are less than 30% left after cropping.

Figure 6: **Original image of airbus detection**. The original images are 2560×2560 pixels, and we cropped them into smaller pieces in 224×224.

G. Implementation Details

G.1. CIFAR-10 denoising:

For the CIFAR-10 dataset, we used the standard CIFAR-10 dataset and applied different levels of AWGN noise to create two correlated datasets. We used the CIFAR-10 experiments as a proof of concept to try different architectures and loss functions and other techniques to finalize our framework. We choose $\lambda_{\text{task}} = 1$ for the task-aware setting and $\lambda_{\text{rec}} = 1$ for the task-agnostic setting. We run 4 random seeds on NDPCA and all baselines to evaluate the performance.

G.2. Locate and lift:

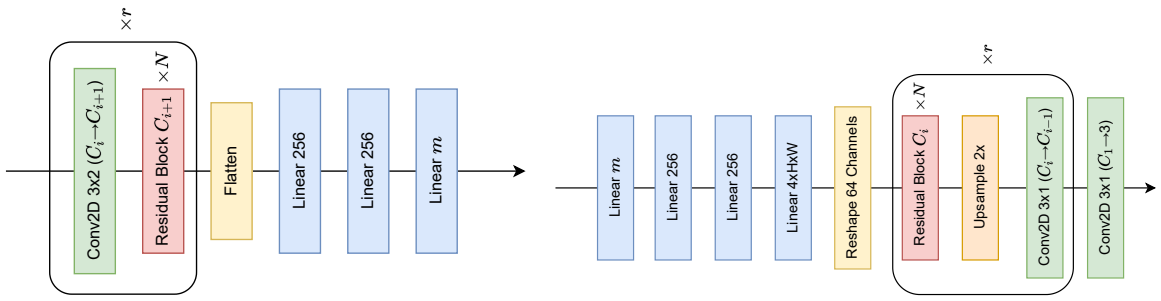
For the locate and lift experiment, we trained our autoencoder with the same random cropping setting as in Sec. F, which cropped the images from 128×128 to 112×112 pixels. During testing, we randomly initialized the location of the brick and center-cropped the images from 128×128 to 112×112 pixels. We scaled all images to 0 to 1 and ran 5 random seeds on NDPCA and all baselines to evaluate the performance. For the task-aware setting, $\lambda_{\text{task}} = 500$, and $\lambda_{\text{rec}} = 1000$ for the task-agnostic. setting

G.3. Airbus detection:

For the Airbus detection task, we used the original Yolo paper for our object detection model together with the detection loss 33. Our experiments with the latest state-of-the-art Yolo v8 model 1 showed that there is no big difference in the Airbus detection dataset in terms of run time and accuracy. Since the size of the original dataset is not enough to train an object detection model, we used the data augmentation proposed in Yolo v8, mosaic, to increase the size of the dataset. Mosaic randomly crops 4 images and merges them to generate a new image. We used random resized crop, blur, median blur, and CLAHE enhancement during training, each with probability 0.05 by functions in the Albumentations package 7. We increased the size of the Airbus dataset from 5904 to 21808 with mosiac and trained the Yolo detection model. Finally, we trained our autoencoder with the same dataset, but downsample the images to 112×112 pixels so that the autoencoder is faster to train. For the task-aware setting, $\lambda_{\text{task}} = 0.1$, and $\lambda_{\text{rec}} = 0.5$ for the task-agnostic setting. We run 2 random seeds

on NDPCA and all baselines to evaluate the performance.

G.4. Neural Autoencoder Architecture and Hyperparameters



(a) Encoder architecture. (b) Decoder architecture.

Figure 7: **ResNet Autoencoer:** The encoder processes inputs through r convolution layers and $r \times N$ residual blocks, followed by 3 fully connected layers with ReLU activation. The decoder processes latent representations in the reverse order from the encoder with $2\times$ upsamplings.

We used the ResNet encoder shown in Fig. 7a and the decoder in Fig. 7b for all experiments. We used different numbers of filters and numbers of residual blocks for our experiments, shown as C and r. We denote m as the number of latent dimensions. The numbers of filters are $C_1 = 32,\ C_2 = 64,\ C_3 = 128$, $C_1 = 8,\ C_2 = 16,\ C_3 = 32,\ C_4 = 64$, and $C_1 = 16,\ C_2 = 32,\ C_3 = 64,\ C_4 = 128$, and the numbers of residual blocks are $r = 0$, $r = 1$, $r = 1$ for CIFAR-10 denoising, locate and lift, and Airbus detection. For CIFAR-10 denoising, we use the Adam optimizer with a learning rate of 0.0002, and for the other two experiments, we use the Adam optimizer with a learning rate of 0.0001. For the sake of training speed, when training DAE and JAE, we first trained a large network with $m_{\max}$ with each random seed. Then, we fixed the network parameters and trained concatenate 3 fully connected layers on each encoder and decoder network to compress and decompress the data to smaller m.

G.5. Balancing Task-aware and Task-agnostic Loss

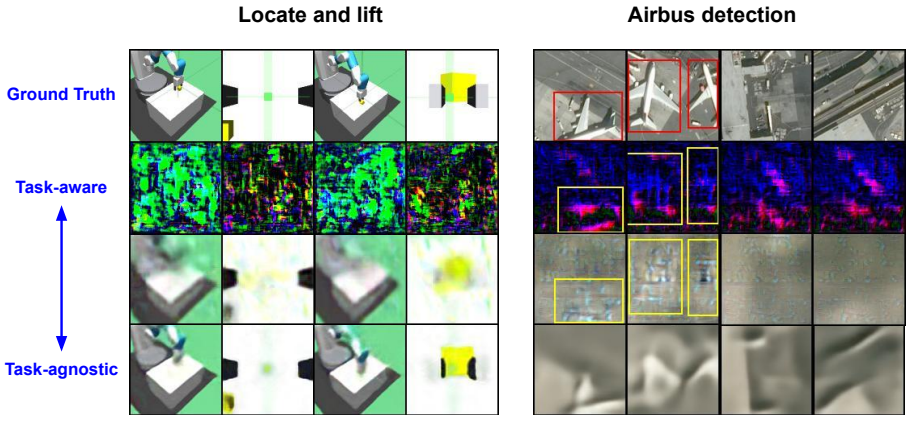


Figure 8: **Weighted task-loss:** Weighted task-aware images faintly reconstruct the original images while restoring task-relevant features with high-frequency noise. In Airbus detection, location of Airbuses is captured with shiny high-frequency pixels in row 3.

NPDCA has a loss function consisting of 2 terms, as shown in (4):

$$\mathcal{L}_{\text{tot}} = \lambda_{\text{task}} \underbrace{\|\hat{Y} - Y\|_F^2}_{\text{task loss}} + \lambda_{\text{rec}} \underbrace{\left(\|\hat{X}_1 - X_1\|_F^2 + \|\hat{X}_2 - X_2\|_F^2 + \ldots \|\hat{X}_K - X_K\|_F^2 \right)}_{\text{reconstruction loss}}. \qquad \text{(4 revisited)}$$

Previous work 31 tested cases of (4), such as task-aware when $\lambda_{\text{task}} > 0, \lambda_{\text{rec}} = 0$, and task-agnostic when $\lambda_{\text{task}} = 0, \lambda_{\text{rec}} > 0$. Of course, one can use different weighted sums of the 2 terms in (4), which we call weighted task-aware. We

show the resulting reconstructed image in Fig. 8, whose weights are a mixture of half of the two other methods. Weighted task-aware images have both blurry reconstructions of the original images and task-relevant features. Unsurprisingly, the task loss and the reconstructed loss of weighted task-aware images are between pure task-aware and task-agnostic, that is, we can use the weights in the loss function to trade off compressing human perception features against task-relevant features. Interestingly, we can see that the task-aware images look similar to the images without Airbuses (last 2 columns), and when there are Airbuses, the task-aware images look different. It means that the features of no Airbuses are pretty much the same in the latent space, thus resulting in similar images in pixel space. Hence we can conclude that task-aware features are not random noise, they are meaningful features only to the task model but not to our eyes.

G.6. Storage and Training Complexity

Model	CIFAR-10		Locate and lift		Airbus detection	
	Storage (MB)	Train (hr)	Storage (MB)	Train (hr)	Storage (MB)	Train (hr)
NDPCA	8.3	0.25	16.4	5.0	33.0	13.0
DAE	5×8.4	5×0.21	4×16.3	4×5.0	4×22.5	4×11.5
JAE	5×10.2	5×0.22	4×11.4	4×3.5	4×32.9	4×10.5

Table 1: **Storage and training complexity:** NDPCA has slightly more storage and training overload than other models for a single bandwidth but can operate across different bandwidths. We multiply the number of bandwidths tested in Fig. 5 to the storage size and training time of DAE and JAE as they require different models for different compression levels.

One key feature of NDPCA is that it only needs one model to operate in different bandwidths. Therefore, we only need to train and store one model at the edge devices and the central node. We compare the complexity of storage and training in Table 1. Although NDPCA has a larger storage size and longer training time than other models, it can operate across different bandwidths. According to Table 1, if all models operate in more than 1 bandwidths, NDPCA saves more storage and training overload because other models have more than 50% of NPDCA's overload. For CIFAR-10 denoising, we tested the training time on an RTX 4090, and for the locate and lift and Airbus detection experiments, we tested the training time on an NVIDIA RTX A5000.

H. Ablation Study

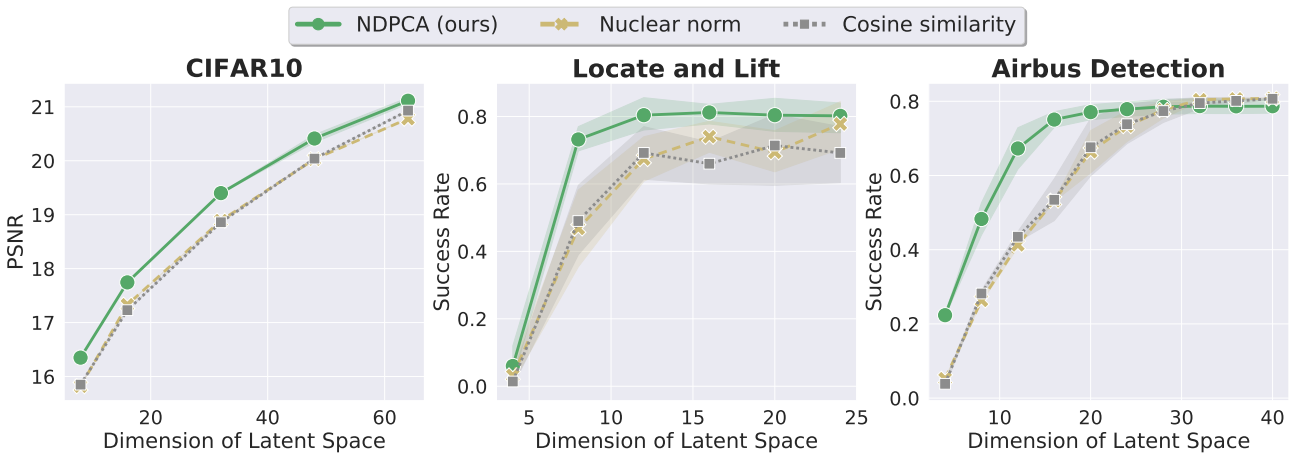


Figure 9: **Ablation study of the nuclear norm and cosine similarity:** Adding the nuclear norm or cosine similarity to the loss function does not improve the performance of the model when compressing latent representations to lower dimensions.

H.1. Cosine similarity and nuclear norm

In Fig. 9, we show that adding nuclear norm or cosine similarity in the training loss (4) does not help the model perform when we use DPCA to project latent representations into lower dimensions. We compared our proposed NDPCA with

the DPCA module against NDPCA without the DPCA module but with the penalization of the nuclear norm and cosine similarity added. The weights of all the additional terms are 0.1. From Fig. 9, we conclude that the DPCA module can increase the performance better than the other two.

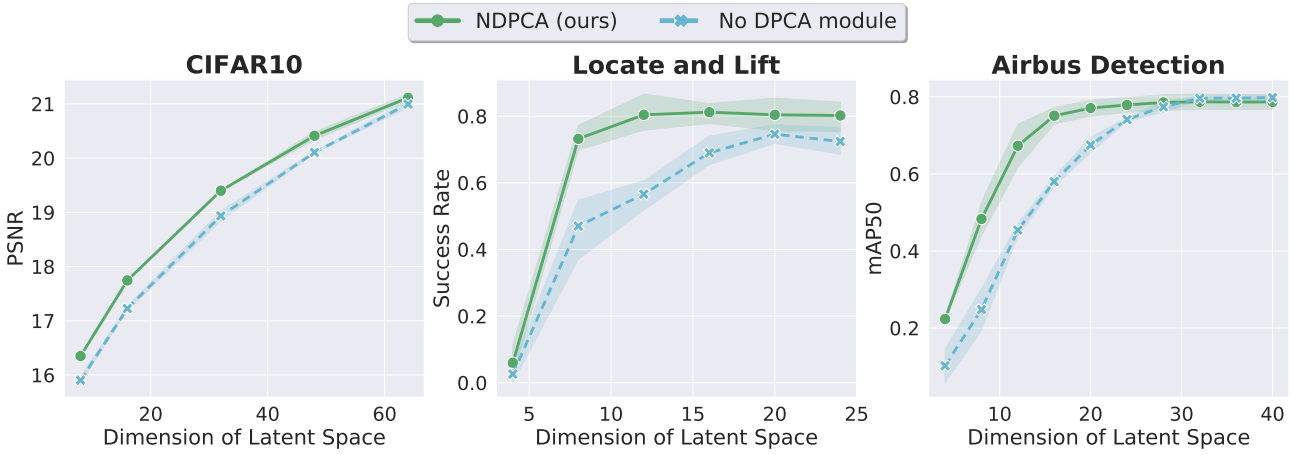


Figure 10: **Ablation study of DPCA module:** The proposed DPCA module effectively increases the performance in lower bandwidths, while achieving the same performance at larger bandwidths.

H.2. DPCA module

In Fig. 10, we show that the proposed DPCA module can help the neural autoencoder learn linear compressible representations, as described in Sec. 4. We see that with the DPCA module, NDPCA can increase the performance in lower bandwidths, while saturating at the performance close to the model without the module. We conclude that with the DPCA module, NDPCA learns to generate low-rank representations, so the performance is better in lower bandwidths. However, when the bandwidth is higher, the bandwidth can almost fully restore the representations, so the two methods perform similarly.

H.3. Single view performance of locate and lift

In the locate and lift experiments, the reinforcement learning agent leverages information from both views as input to manipulate. Here, we detail why the 2 views are complementary to accomplish the task. The success rate of an agent is 76% with only the arm-view and 45% with the side-view. When combining both, the success rate is 83%. The reason why the views are complementary is that the side-view provides global information on the position of the arm and the brick, but sometimes the brick is hidden behind the arm. The arm-view captures detailed information from a narrow view of the desk. Once the arm-view captures the brick, it is straightforward to move toward it and lift it. The arm view is more important because with only the arm-view, the agent can randomly explore the brick, but with only the side-view, the brick might be vague to see and thus harder to lift. Of course, with both views, the robotic arm can easily move toward the vague position of the brick and use arm-view to lift it.