Foundation Models on a Budget: Approximating Blocks in Large Vision Models

Irene Cannistraci [1] **Simone Antonelli** [2] **Emanuele Palumbo** [1] **Thomas M. Sutter** [1] **Emanuele Rodolà** [3]
Bastian Rieck [† 4] **Julia E. Vogt** [† 1]

Abstract

Foundation Models have shown impressive performance in various tasks and domains, yet they require massive computational resources, raising concerns about accessibility and sustainability. In this paper, we propose Transformer Blocks Approximation (TBA), a novel method that leverages intra-network similarities to identify and approximate transformer blocks in large vision models using only a small amount of training data. TBA replaces these blocks using lightweight, closed-form transformations, without any additional training steps. The proposed method reduces the number of parameters while having minimal impact on the downstream task.

1. Introduction

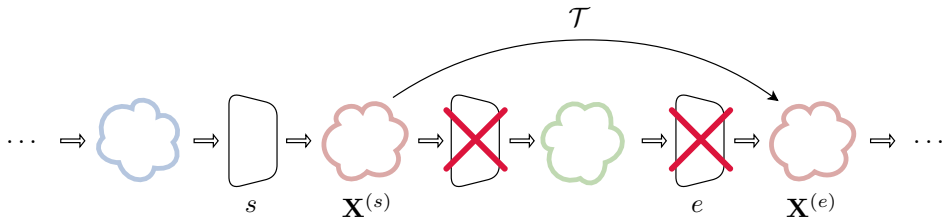

Figure 1: **Framework Description**. Given two latent spaces $\mathbf{X}^{(s)}$ and $\mathbf{X}^{(e)}$ representing the output of blocks s and e for a random subset of n data points from the training set, we define a transformation matrix $\mathcal{T}$ such that: $\mathbf{X}^{(e)} \approx \mathcal{T}(\mathbf{X}^{(s)})$.

As Neural Networks (NNs) grow in size and complexity, their demand for computational resources has become a significant bottleneck. Despite the impressive performance of large models, they often come with substantial trade-offs, such as increased memory and power consumption. This has led to a growing interest in methods that can reduce model complexity without sacrificing performance. However, most existing approaches trying to mitigate these challenges either demand additional training steps or result in substantial performance degradation. Recent research showed that there exist similarities between representation within and between NNs. The observation that many layers or components within these networks perform similar functions or produce highly correlated outputs suggests that some blocks may be approximated, highlighting opportunities for network simplification. In this paper, we investigate two key research questions: (i) how to effectively identify blocks within neural networks that yield similar representations, and (ii) how to approximate these blocks with minimal impact on overall performance without any additional training steps. To address the first question, we provide an extensive analysis showing that the Mean Squared Error (MSE) effectively identifies such blocks, suggesting that their approximation should have a negligible impact. For the second question, we propose Transformer Blocks Approximation (TBA), a novel method that leverages similarities between representations to approximate transformer blocks using lightweight transformations, such as linear mappings. Once the blocks that have minimal impact on model functionality are identified, TBA replaces them with a simpler transformation. This approximation strategy allows for reducing the overall parameter count, ensuring minimal impact on downstream task performance.

[†]Joint senior authorship [1]Department of Computer Science, ETH Zurich [2]CISPA Helmholtz Center for Information Security [3]Sapienza University of Rome [4]University of Fribourg. Correspondence to: Irene Cannistraci <irene.cannistraci@inf.ethz.ch>.

Accepted at Methods and Opportunities at Small Scale (MOSS), ICML 2025, Vancouver, Canada.

2. Related work

While large-scale models with billions or even trillions of parameters continue to achieve state-of-the-art performance, their growth comes with trade-offs, such as slower inference times and significantly higher computational costs. Improving the efficiency of Deep Neural Network (DNN) has been a long-standing area of research. For instance, Veit et al. (2016) shows that removing residual blocks from deep Convolutional Neural Networks (CNNs) only marginally impacts performance, which inspired approaches to reduce inference time by dynamically deciding which layers to execute based on the input (Wu et al., 2018; Veit & Belongie, 2018). Additionally, various techniques to enhance efficiency have emerged, such as early exiting and model pruning. Early exit strategies, which introduce intermediate output layers at different stages of the network, have been shown to reduce inference time (Xin et al., 2020; Zhou et al., 2020; Yu et al., 2022; Tang et al., 2023). However, these approaches require the training of intermediate classifiers to enable exits at predefined layers. Alternatively, model pruning reduces computational load by either removing individual weights based on specific criteria, such as gradient information (Ma et al., 2023), entropy (Liao et al., 2023), or second-order information (Singh & Alistarh, 2020), or by eliminating larger structural components, like channels or residual blocks in ResNets (Bai et al., 2023; Wang & Wu, 2023) and self-attention layers in Transformers (Zhang & He, 2020; Sajjad et al., 2023; Venkataramanan et al., 2024; Zhang et al., 2024). Although effective, these approaches require training the model from scratch and, in the best case, fine-tuning. However, Bai et al. (2023) shows that for CNNs, this additional training step can sometimes be avoided.

Unlike other methods, TBA leverages intra-network similarities to reduce foundation model complexity *without the need for additional training steps* while maintaining competitive performance. A more detailed discussion of related work is provided in Section 6.4.

3. Transformer Blocks Approximation

The core idea of the proposed approach is to detect similar representations within NNs and replace the corresponding blocks with simple transformations. In the context of this work, a "block" refers to a sequence of layers including multi-head self-attention, normalization, and feed-forward layers, functioning as a cohesive unit. Instead of executing the entire NN, using TBA, we approximate these blocks, reducing computational complexity while preserving the network's functionality. To provide a clear understanding, Figure 1 presents an overview of our method.

Identifying similar representations We hypothesize that certain foundation models may contain blocks that produce similar representations. To quantify these similarities, we employ the MSE. A low MSE between the output of an earlier block and a later block indicates that their respective representations are highly similar. This suggests that the intervening sequence of blocks contributes minimally to transforming the representation, highlighting potential redundancy.

Let B represent the total number of blocks in the model. Let $\mathbf{h}^{(k)} \in \mathbb{R}^{d_k}$ denote the output representation (a vector of dimension d_k) of block k for a single input, where $k \in \{1, 2, \ldots, B\}$. For any two blocks, a starting block s and an ending block e (where $1 \leq s < e \leq B$), we compute their output representations $\mathbf{h}^{(s)}(\mathbf{x}^{(s)})$ and $\mathbf{h}^{(e)}(\mathbf{x}^{(e)})$. For each $\mathbf{x} \in \mathcal{D}_{\text{sub}}$, where $\mathcal{D}_{\text{sub}} \subset \mathcal{D}$ is a subset of N points sampled uniformly at random from $\mathcal{D}$, $\mathbf{x}^{(k)}$ defines the input to the corresponding layer $\mathbf{h}^{(k)}$. The MSE between these representations is defined as:

$$\text{MSE}(\mathbf{h}^{(s)}, \mathbf{h}^{(e)}) = \frac{1}{|\mathcal{D}_{\text{sub}}|} \sum_{\mathbf{x} \in \mathcal{D}_{\text{sub}}} \left\| \mathbf{h}^{(e)}(\mathbf{x}^{(e)}) - \mathbf{h}^{(s)}(\mathbf{x}^{(s)}) \right\|_2^2$$

By systematically evaluating the MSE for various pairs of blocks (s, e), we can identify those being good candidates for approximation. This allows for targeted complexity reduction without substantially altering the network's internal representations or downstream-task performance at a very low computational cost.

Approximating transformer blocks Once two blocks s and e are identified as having highly similar output representations, our goal is to replace the intermediate blocks $s+1, \ldots, e$ with a single, lightweight transformation that maps the output of block s directly to an approximation of the output of block e. This approach allows us to skip the computation of blocks $s+1, \ldots, e$, effectively reducing the overall computational costs. This approximation can be repeated for multiple, non-overlapping blocks, i.e., blocks (s_i, e_i) and (s_j, e_j) with $e_i < s_j$.

Let $\mathbf{X}^{(s)} \in \mathbb{R}^{N \times d_s}$ and $\mathbf{X}^{(e)} \in \mathbb{R}^{N \times d_e}$ represent the output representations from block s and e respectively, for the data points in $\mathcal{D}_{\text{sub}}$. Our objective is to find a transformation $\mathcal{T} : \mathbb{R}^{d_s} \to \mathbb{R}^{d_e}$ such that:

$$\mathbf{X}^{(e)} \approx \mathcal{T}(\mathbf{X}^{(s)})$$

In this work, we consider $\mathcal{T}$ to be a *linear* transformation $\mathbf{T}$ that can be estimated by minimizing the squared error between the transformed output $\mathcal{T}(\mathbf{X}^{(s)})$ and the actual $\mathbf{X}^{(e)}$, which can be solved using least squares:

$$\mathbf{T} = \arg\min_{\mathcal{T}} \|\mathbf{X}^{(e)} - \mathcal{T}(\mathbf{X}^{(s)})\|_2^2$$

This optimization problem allows for a closed-form solution that efficiently computes the optimal transformation $\mathbf{T}$. The solution bypasses the computation of *all* layers between any two blocks s and e, replacing them with $\mathbf{T}$. This approximation significantly reduces computational complexity by replacing one or more transformer blocks, comprising multi-head self-attention and feed-forward layers, with a low-cost linear transformation.

4. Experiments

We start by analyzing in Section 4.1 the block-level similarities within pre-trained vision foundation models, then in Section 4.2 we provide a quantitative evaluation of TBA's performance on the image classification tasks. Additional qualitative results can be found in Section 6.3.1 and quantitative results, including standard, zero-shot, and cross-dataset generalization scenarios, in Sections 6.3.2, 6.3.3 and 6.3.7. Details regarding the models, datasets, computational resources, and software tools are provided in Table 2, Table 3, Section 6.2.4, and Section 6.2.3, respectively.

4.1. Latent analysis

We investigate similarities emerging in latent representations using two distinct transformer-based models with different dimensionality (i.e., `DiNO-B`, and `DEiT-S`), and three datasets. Figure 2 presents the MSE matrices between blocks of different models when using different datasets. We conduct the analysis using the mean over the tokens, but in Section 6.3.4 we show that this also applies when using only the CLS token. These similarities are calculated using only a small subset of the training data (i.e., 300 samples). Results reveal that while the patterns of similarity vary across models, they remain consistent across different datasets, suggesting that the similarity structure between computational blocks is predominantly influenced by the model rather than the dataset used. This finding aligns with observations from (Nguyen et al., 2020), where DNNs trained from scratch tend to exhibit a distinctive "block structure" in their representations, linked to model overparameterization. Our results extend this observation to pretrained foundation models, showing that such a structure is primarily influenced by the model. Additionally, Figure 2 shows that the last layers of `DEiT-S` are more similar than those of `DiNO-B`, which suggests that they could be good candidates for the approximation.

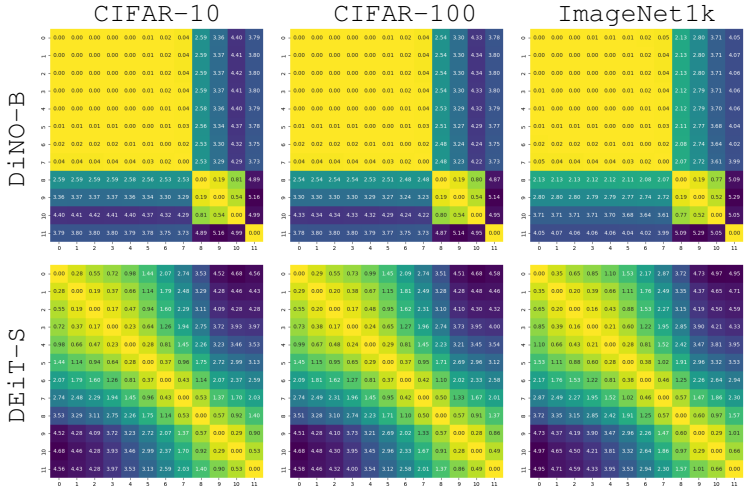

Figure 2: **Block Similarities.** Block-by-block similarities in `DiNO-B`, and `DEiT-S` across three datasets. Each matrix quantifies the MSE between latent representations of different blocks, showing potential blocks for approximation.

In Section 6.3.4 (Figure 6), we present an ablation study to investigate the influence of the similarity metric. We correlate various metrics with downstream task performance using `ViT-S` and `ImageNet1k`. Our findings reveal that using simple metrics such as MSE or cosine similarity to identify block-level similarities exhibits a strong correlation with final accuracy.

These results underscore the flexibility of our proposed method, indicating its compatibility with a range of similarity or distance functions that can capture inter-block relationships. Additional results in Section 6.3.4.

Takeaway The block-wise similarity patterns observed in pretrained foundation models are primarily determined by the model, and remain consistent across different datasets.

4.2. Transformer block approximation performance

We perform image classification using various pretrained models with different dimensionalities and datasets, keeping all models frozen. Block approximations are computed using a shared linear transformation applied across all tokens, based on a subset of 3,000 training samples. Then, we train a single linear layer using the Adam optimizer (learning rate 0.001) for 5 epochs, with 3 seeds and a batch size of 256. As presented in Table 1, the proposed TBA method successfully reduces model size while maintaining, and in some cases even improving, image classification performance. These downstream task results support the findings from the analysis in Sections 4.1 and 6.3.1. Specifically, approximating the final block of `DEiT-S` yields a final representation highly similar to the original. This similarity suggests such blocks are optimal candidates for approximation, a conclusion further validated by Table 1. The table shows that even when approximating multiple consecutive blocks (e.g., 9→11), we can reduce parameters while achieving performance superior to the original model. Overall, the consistent or enhanced performance indicates that a linear transformation is sufficient to approximate these transformer blocks, thereby effectively reducing model parameters. It's important to note that this transformation is shared across all tokens, with no additional training steps required. Additional results are presented in Tables 9 to 14. Finally, in Section 6.3.8, we further analyze model behavior after block approximations on the final image-classification task, complementing the quantitative results with qualitative insights into the approximation procedure.

Table 1: **Image Classification Performance Across Architectures.** Classification accuracy scores for `DEiT-S` using multiple datasets, and 3 seeds. The "Approx." column specifies the blocks used for approximation, where the first value represents the block whose output is used to approximate the second block's output. The "Params." column shows the number of parameters removed by the approximation compared to the original model. More results in Tables 9 to 13.

			Accuracy ↑		
	Approx.	Params.	CIFAR-10	CIFAR-100F	ImageNet1k
DEiT-S	1 →5	-6.51M	78.35 ± 0.17 (-13.55%)	50.57 ± 0.29 (-28.90%)	43.70 ± 0.27 (-40.88%)
	2 →5	-4.88M	85.73 ± 0.31 (-5.41%)	60.55 ± 0.16 (-14.87%)	62.04 ± 0.21 (-16.05%)
	7 →10	-4.88M	89.17 ± 0.04 (-1.61%)	69.15 ± 0.33 (-2.78%)	57.48 ± 0.06 (-22.24%)
	2 →4	-3.26M	88.95 ± 0.05 (-1.85%)	66.60 ± 0.50 (-6.37%)	70.00 ± 0.32 (-5.32%)
	9 →11	-3.26M	90.90 ± 0.12 (+0.30%)	71.92 ± 0.17 (+1.11%)	69.95 ± 0.24 (-5.39%)
	1 →2, 4 →5	-3.26M	85.43 ± 0.25 (-5.74%)	61.66 ± 0.13 (-13.31%)	66.04 ± 0.13 (-10.68%)
	0 →1	-1.63M	85.00 ± 0.27 (-6.21%)	61.95 ± 0.39 (-12.91%)	62.55 ± 0.12 (-15.38%)
	3 →4	-1.63M	90.50 ± 0.10 (-0.14%)	70.25 ± 0.20 (-1.24%)	73.03 ± 0.10 (-1.22%)
	9 →10	-1.63M	90.90 ± 0.20 (+0.30%)	71.74 ± 0.09 (+0.86%)	72.34 ± 0.16 (-2.15%)
	10 →11	-1.63M	**91.07** ± 0.18 (**+0.49**%)	**71.95** ± 0.17 (**+1.15**%)	**73.97** ± 0.17 (**+0.05**%)
	original	21.82M	90.63 ± 0.22	71.13 ± 0.26	73.93 ± 0.14

Takeaway Linear transformations learned by TBA generalize effectively across datasets, enabling lightweight model adaptation without additional training.

5. Conclusion

In this work, we first analyze the emergence of consistent block-wise representation similarities within pretrained foundation models and then propose a method to leverage these similarities to obtain smaller yet performant models. To this end, we propose Transformer Blocks Approximation (TBA), a novel method for identifying and efficiently approximating similar transformer blocks using a simple linear transformation and a small subset of the training data, without requiring additional training or fine-tuning. Our extensive empirical evaluations across multiple pretrained vision models and datasets validate that TBA significantly reduces model parameters while maintaining, and sometimes even improving, downstream task performance. TBA thus offers a practical and efficient method for streamlining foundation models, making them more computationally accessible.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Luca Moschella for the insightful discussions. TS and EP are supported by the grant #2021-911 of the Strategic Focal Area "Personalized Health and Related Technologies (PHRT)" of the ETH Domain (Swiss Federal Institutes of Technology). EP is also supported by a fellowship from the ETH AI Center. This work has received funding from the Swiss State Secretariat for Education, Research, and Innovation (SERI).

References

Bai, S., Chen, J., Shen, X., Qian, Y., and Liu, Y. Unified data-free compression: Pruning and quantization without fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5876–5885, 2023.

Ballester, R., Casacuberta, C., and Escalera, S. Topological data analysis for neural network analysis: A comprehensive survey. *arXiv preprint arXiv:2312.05840*, December 2023.

Barannikov, S., Trofimov, I., Balabin, N., and Burnaev, E. Representation topology divergence: A method for comparing neural network representations. *arXiv preprint arXiv:2201.00058*, 2021.

Cannistraci, I., Moschella, L., Maiorca, V., Fumero, M., Norelli, A., and Rodolà, E. Bootstrapping parallel anchors for relative representations. In Maughan, K., Liu, R., and Burns, T. F. (eds.), *The First Tiny Papers Track at ICLR 2023, Tiny Papers @ ICLR 2023, Kigali, Rwanda, May 5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=VBuUL2IWlq.

Cannistraci, I., Moschella, L., Fumero, M., Maiorca, V., and Rodolà, E. From bricks to bridges: Product of invariances to enhance latent space communication. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vngVydDWft.

Crisostomi, D., Cannistraci, I., Moschella, L., Barbiero, P., Ciccone, M., Lio, P., and Rodolà, E. From charts to atlas: Merging latent spaces into one. In *NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations*, 2023. URL https://openreview.net/forum?id=ZFu7CPtznY.

Davari, M., Horoi, S., Natik, A., Lajoie, G., Wolf, G., and Belilovsky, E. Reliability of cka as a similarity measure in deep learning. *arXiv preprint arXiv:2210.16156*, 2022.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Fumero, M., Pegoraro, M., Maiorca, V., Locatello, F., and Rodolà, E. Latent functional maps: a spectral framework for representation alignment. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 66178–66203. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/79be41d858841037987964e3f5caf76d-Paper-Conference.pdf.

Hotelling, H. Relations between two sets of variates. *Breakthroughs in statistics: methodology and distribution*, pp. 162–190, 1992.

Klabunde, M., Schumacher, T., Strohmaier, M., and Lemmerich, F. Similarity of neural network models: A survey of functional and representational measures. *arXiv preprint arXiv:2305.06329*, 2023.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kuprieiev, R., skshetry, Petrov, D., Redzyński, P., Rowlands, P., da Costa-Luis, C., Schepanovski, A., Shcheklein, I., Taskaya, B., Gao, Orpinel, J., de la Iglesia Castro, D., Santos, F., Sharma, A., Berenbaum, D., Zhanibek, Hodovic, D., daniele, Kodenko, N., Grigorev, A., Earl, Dash, N., Vyshnya, G., Lamy, R., maykulkarni, Hora, M., Vera, and Mangal, S. Dvc: Data version control - git for data & models, 2022. URL https://doi.org/10.5281/zenodo.7083378.

Kvinge, H., Jorgenson, G., Brown, D., Godfrey, C., and Emerson, T. Internal representations of vision models through the lens of frames on data manifolds. In *NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations*, 2022.

Lähner, Z. and Moeller, M. On the direct alignment of latent spaces. In Fumero, M., Rodolá, E., Domine, C., Locatello, F., Dziugaite, K., and Mathilde, C. (eds.), *Proceedings of UniReps: the First Workshop on Unifying Representations in Neural Models*, volume 243 of *Proceedings of Machine Learning Research*, pp. 158–169. PMLR, 15 Dec 2024. URL https://proceedings.mlr.press/v243/lahner24a.html.

Liao, Z., Quétu, V., Nguyen, V.-T., and Tartaglione, E. Can unstructured pruning reduce the depth in deep neural networks? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1402–1406, 2023.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Maiorca, V., Moschella, L., Norelli, A., Fumero, M., Locatello, F., and Rodolà, E. Latent space translation via semantic alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

Morcos, A., Raghu, M., and Bengio, S. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31, 2018.

Moschella, L., Maiorca, V., Fumero, M., Norelli, A., Locatello, F., and Rodolà, E. Relative representations enable zero-shot latent space communication. In *Proc. ICLR*, 2023.

Nguyen, T., Raghu, M., and Kornblith, S. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

Norelli, A., Fumero, M., Maiorca, V., Moschella, L., Rodola, E., and Locatello, F. Asif: Coupled data turns unimodal models to multimodal without training. *Advances in Neural Information Processing Systems*, 36:15303–15319, 2023.

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

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Sajjad, H., Dalvi, F., Durrani, N., and Nakov, P. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, 2023.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S., Crowson, K., Schmidt, L., Kaczmarczyk, R., and Jitsev, J. Laion-5b: An open large-scale dataset for training next generation image-text models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 25278–25294. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a1859debfb3b59d094f3504d5ebb6c25-Paper-Datasets_and_Benchmarks.pdf.

Singh, S. P. and Alistarh, D. Woodfisher: Efficient second-order approximation for neural network compression. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18098–18109. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d1ff1ec86b62cd5f3903ff19c3a326b2-Paper.pdf.

Tang, S., Wang, Y., Kong, Z., Zhang, T., Li, Y., Ding, C., Wang, Y., Liang, Y., and Xu, D. You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10781–10791, 2023.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. arxiv 2020. *arXiv preprint arXiv:2012.12877*, 2(3), 2020.

Valeriani, L., Doimo, D., Cuturello, F., Laio, A., Ansuini, A., and Cazzaniga, A. The geometry of hidden representations of large transformer models. *Advances in Neural Information Processing Systems*, 36, 2024.

Veit, A. and Belongie, S. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Veit, A., Wilber, M. J., and Belongie, S. Residual networks behave like ensembles of relatively shallow networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/37bc2f75bf1bcfe8450a1a41c200364c-Paper.pdf.

Venkataramanan, S., Ghodrati, A., Asano, Y. M., Porikli, F., and Habibian, A. Skip-attention: Improving vision transformers by paying less attention. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vI95kcLAoU.

Wang, G.-H. and Wu, J. Practical network acceleration with tiny sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., and Feris, R. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xin, J., Tang, R., Lee, J., Yu, Y., and Lin, J. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

Yu, F., Huang, K., Wang, M., Cheng, Y., Chu, W., and Cui, L. Width & depth pruning for vision transformers. In *Proc. AAAI*, 2022.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Zhang, H., Zhou, Y., and Wang, G.-H. Dense vision transformer compression with few samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15825–15834, June 2024.

Zhang, M. and He, Y. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in neural information processing systems*, 33:14011–14023, 2020.

Zhou, W., Xu, C., Ge, T., McAuley, J., Xu, K., and Wei, F. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020.

6. Appendix

6.1. Limitations and future work

While TBA demonstrates significant promise in efficiently approximating transformer blocks, our current investigation has primarily focused on vision transformer architectures and their application to classification tasks. Future research will explore the applicability of TBA to other modalities (e.g., text) and to diverse downstream tasks (e.g., image reconstruction). Such an expansion will be crucial for testing the universality of the observed block-similarity phenomena and assessing TBA's adaptability. Furthermore, we aim to expand the analysis of these block-level similarities. This involves investigating redundancies at finer granularities, such as within individual attention heads or feed-forward network layers, and potentially at coarser granularities across larger segments of the network. Another important direction is to examine more in detail how these similarity patterns arise depending on token type (e.g., CLS token versus last token). The analysis could lead to even more refined and context-aware approximation strategies, further enhancing model efficiency.

6.2. Implementation details

This section details the experiments conducted in Section 4, providing information to reproduce them. Additionally, we provide the code as a zip file in the supplementary material.

6.2.1. Models and Datasets

Table 2 contains the full list of the pretrained models, while Table 3 contains dataset information.

Table 2: **Pretrained models details.** Details of the pretrained feature extractors with their HuggingFace key, their alias, and their latent space dimensionality.

Modality	HuggingFace Model Name	Alias	Enc. Dim
Vision	WinKawaks/vit-tiny-patch16-224	`ViT-T` (Dosovitskiy et al., 2021)	192
	WinKawaks/vit-small-patch16-224	`ViT-S` (Dosovitskiy et al., 2021)	384
	facebook/dinov2-small	`DiNO-S` (Oquab et al., 2023)	384
	facebook/deit-small-patch16-224	`DEiT-S` (Touvron et al., 2020)	384
	google/vit-base-patch16-224	`ViT-B` (Dosovitskiy et al., 2021)	768
	facebook/dinov2-base	`DiNO-B` (Oquab et al., 2023)	768
	laion/CLIP-ViT-B-16-laion2B-s34B-b88K	`OpenCLIP-ViT-B` (Zhai et al., 2019)	768
	google/vit-large-patch16-224	`ViT-L` (Dosovitskiy et al., 2021)	1024

Table 3: **Dataset details.** Details of the HuggingFace datasets used in the classification and reconstruction experiments, with the associated number of classes.

Modality	Name	Alias	# Classes
Vision	MNIST (Deng, 2012)	`MNIST`	10
	Fashion-MNIST (Xiao et al., 2017)	`F-MNIST`	10
	CIFAR-10 (Krizhevsky et al., 2009)	`CIFAR-10`	10
	CIFAR-100 (coarse) (Krizhevsky et al., 2009)	`CIFAR-100C`	20
	CIFAR-100 (fine) (Krizhevsky et al., 2009)	`CIFAR-100F`	100
	Imagenet-1k (Russakovsky et al., 2015)	`ImageNet1k`	1000

6.2.2. Approximators

The first implementation, referred to as the Res-MultiLayer Perceptron (MLP), is composed of two normalization layers and a feedforward submodule. The first layer normalization processes the input tensor, followed by a feedforward submodule comprising a linear transformation, a SiLU activation, a dropout layer, and a final linear transformation. The output of the

feedforward submodule is added to the normalized input via a residual connection. This sum is then passed through the second normalization layer to produce the final output. The second implementation, referred to as the MLP, is a simplified MLP that employs a sequential architecture with a first linear transformation that reduces the input dimensionality to half of the target dimension, followed by a GELU activation function for smooth non-linearity, and a final linear transformation that restores the reduced features to match the target dimensionality. Refer to **??** 1**??** 2 for the code snippet of the two translators.

Listing 1: Python Code Snippet for the Res-MLP translator

```
class ResMLP(nn.Module):
    def __init__(self, num_features: int, dropout_p: float):
        super().__init__()

        self.norm1 = nn.LayerNorm(num_features)
        self.norm2 = nn.LayerNorm(num_features)

        self.ff = nn.Sequential(
            nn.Linear(num_features, num_features),
            nn.SiLU(),
            nn.Dropout(p=dropout_p),
            nn.Linear(num_features, num_features),
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x_normalized = self.norm1(x)
        x_transformed = self.ff(x_normalized)
        return self.norm2(x_transformed + x_normalized)
```

Listing 2: Python Code Snippet for the MLP translator

```
translation = nn.Sequential(
    nn.Linear(x.size(1), y.size(1) // 2),
    nn.GELU(),
    nn.Linear(y.size(1) // 2, y.size(1)),
)
```

6.2.3. Tools & technologies

All the experiments presented in this work employ the following tools:

- *PyTorch Lightning*, to ensure reproducible results while also getting a clean and modular codebase;
- *NN-Template GrokAI (2021)*, to easily bootstrap the project and enforce best practices;
- *Transformers by HuggingFace*, to get ready-to-use transformers for both text and images;
- *Datasets by HuggingFace*, to access most of the datasets;
- *DVC* (Kuprieiev et al., 2022), for data versioning;

6.2.4. Computational resources

Experiments involving larger models, specifically `DiNO-B`, `OpenCLIP-ViT-B`, and `ViT-L`, were conducted on an NVIDIA H100 GPU equipped with 93 GB of memory. All the other experiments utilized an NVIDIA GeForce RTX 5090 GPU with 31 GB of memory.

6.3. Additional Experiments

This section presents supplementary experiments to extend those detailed in Section 4.

6.3.1. Effects of TBA on latent representations

After identifying blocks producing similar representations, we analyze the impact on the latent representation output by the last block when approximating using TBA. These blocks are approximated using a shared linear transformation applied across all tokens based on a subset of 3,000 training samples. For consistency, we employed the same models and datasets used for the results in Figure 2. We compute the MSE between the representations in the last block of the original and the TBA model when approximating the k^{th} block. As illustrated in Figure 3, in most cases, the MSE increases as the block depth increases. This suggests that approximating the final blocks would lead to significant changes in the final

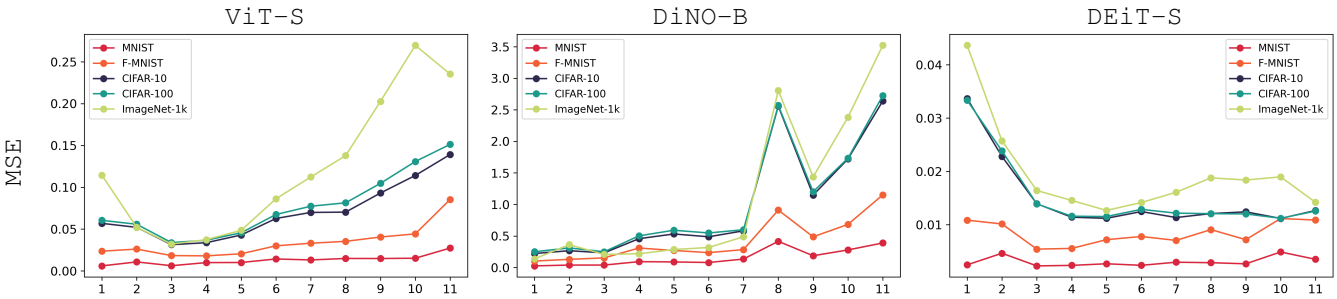

Figure 3: **Approximation vs. Representation Similarity.** MSE between the last block representations of the original and the approximated model when approximating the i^{th} block. Each column is a different model (`ViT-S`, `DiNO-B`, `DEiT-S`), while curves represent different datasets.

representations, indicating their critical role. However, in the case of `DEiT-S`, the trend is different. Here, the MSE decreases toward the final blocks. This is also shown in the similarity matrices illustrated in Figure 2, where the final blocks have a high similarity. These findings reinforce the intuition behind the use of MSE, demonstrating a correlation between the metric and the final representation similarity when approximating blocks. In some instances, such as with the `MNIST` dataset, the MSE scores remain relatively consistent across blocks, indicating that the representations are largely similar to one another. However, for more complex datasets like `ImageNet1k`, the representations in the final or the first blocks become increasingly dissimilar, making it advantageous to approximate intermediate blocks. This suggests that the similarities are primarily driven by the model but also guided by the complexity of the dataset, allowing for targeted approximations that reduce model parameters and complexity without significantly compromising performance. To further investigate the effect of TBA on the latent representations, we visualize, using Principal Component Analysis (PCA), the representations of the TBA model when approximating the final block and its original representation (see Figure 4). The plot shows the representations generated using the `DiNO-B` and `DEiT-S` pretrained encoders on `F-MNIST`. Colors represent classes. The visualization highlights that approximating the final block of `DiNO-B` results in noticeable deviations from the original representations, while in `DEiT-S` the approximated representations remain similar to the original ones. This observation aligns with the results from Figure 3, where approximating blocks can lead to significant changes in representations. For complete results, refer to Section 6.3.5.

6.3.2. Transformer block approximation performance

Zero-shot image classification To further assess the effectiveness of our approach, we evaluate TBA in a zero-shot image classification setting. This evaluation utilizes the `OpenCLIP-ViT-B` model (Radford et al., 2021), which was pretrained on `LAION-2B` (Schuhmann et al., 2022), with `ImageNet1k` serving as the downstream evaluation dataset. As in previous experiments, the model remains frozen, and block approximations are computed using a shared linear transformation applied across all tokens, based on a subset of 3,000 training samples. Importantly, we apply these approximations only to the vision encoder, leaving the text encoder unchanged. We follow the standard `ImageNet1k` prompt templates. As shown in Table 4, when approximating earlier vision blocks (e.g., $1 \rightarrow 2$ or $2 \rightarrow 3$), the model still achieves competitive results, but slightly reduces the number of parameters. The analysis is conducted only on the base version, as larger versions (e.g., `OpenCLIP-ViT-L` or `OpenCLIP-ViT-H`) contain too many parameters and are thus beyond the scope of this paper. Details on the models are provided in Table 2.

Transformation generalization We further evaluate the model's ability to generalize transformations across datasets. Specifically, we analyze the capability of applying a linear transformation learned on one dataset (source) to another (target), using the same underlying pretrained architecture and without any subsequent training or fine-tuning of the approximated

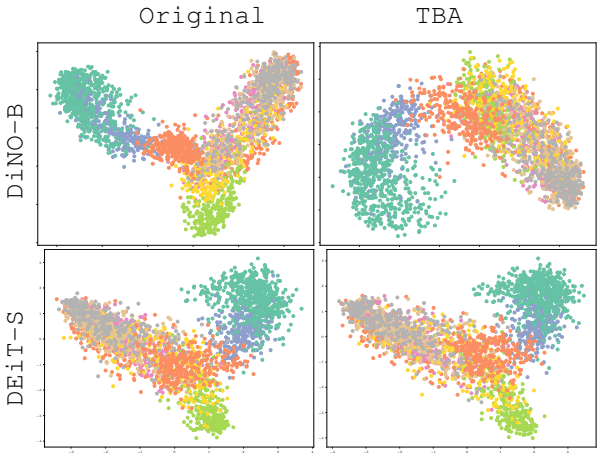

Figure 4: **PCA Visualization.** Final block representations for original and TBA models on F-MNIST reveal DiNO-B's stronger reliance on final block compared to DEiT-S.

Table 4: **Zero-shot image classification.** Accuracy scores for OpenCLIP-ViT-B on ImageNet1k. The "Approx." column specifies the blocks being approximated, where the first value represents the block whose output is used to approximate the second block's output. The "Δ" column indicates the change in accuracy.

Params.	Approx.	Accuracy ↑	Δ
	$0 \to 1$	57.93	-17.41%
	$1 \to 2$	64.20	-8.56%
	$2 \to 3$	**66.35**	**-5.51%**
	$3 \to 4$	64.65	-7.90%
	$4 \to 5$	64.86	-7.60%
-6.49M	$5 \to 6$	58.05	-17.32%
	$6 \to 7$	61.56	-12.31%
	$7 \to 8$	58.53	-16.64%
	$8 \to 9$	52.32	-25.50%
	$9 \to 10$	59.21	-15.68%
	$10 \to 11$	22.64	-67.75%
149.07M	original	<u>70.21</u>	—

model. Table 5 reports accuracy scores for ViT-S and DiNO-S on MNIST, CIFAR-10, and CIFAR-100, where the "Source" column indicates the source dataset used to compute the transformation. The results demonstrate that a single, token-shared linear transformation can generalize effectively across different target datasets, with the exception of MNIST, which may be too simplistic to yield broadly transferable representations. This cross-dataset generalization is particularly valuable in scenarios where fine-tuning is infeasible, such as in privacy-sensitive domains like healthcare or finance. In these contexts, regulatory constraints often restrict training on local data, yet leveraging existing pretrained models remains crucial. Our findings suggest that lightweight adaptation via shared linear transformations offers a promising path for improving model performance on custom datasets without requiring additional training. Additional results are provided in Tables 15 and 16.

Table 5: **Generalization Results.** Classification accuracies for approximating ViT-S blocks with a linear transformation learned on one dataset and applied to others. The "Approx." column specifies the blocks being approximated, where the first value represents the block whose output is used to approximate the second block's output, while the "Source" column names the dataset used to compute the transformation. See Tables 15 and 16 for additional results.

		Accuracy ↑		
Approx.	Source	MNIST	CIFAR-10	CIFAR-100C
$3 \to 4$	MNIST	93.52	10.36	8.97
	CIFAR-10	88.02	95.18	86.14
	CIFAR-100F	88.21	94.82	85.92
$3 \to 5$	MNIST	88.22	15.17	8.52
	CIFAR-10	61.68	93.57	80.24
	CIFAR-100F	64.18	92.77	80.56

6.3.3. Baselines and Comparison

Direct comparisons for TBA are limited, as existing approaches either fine-tune pretrained models or train from scratch. In contrast, TBA maintains the foundation model frozen. We include two types of comparisons: a variant of SkipAt, a method proposed in (Venkataramanan et al., 2024), and an ablation study on transformation complexity.

SkipAt identity (Venkataramanan et al., 2024) explores the effect of skipping self-attention or multi-head self-attention by inserting identity functions in pretrained models, without retraining. We extend this idea by skipping entire blocks, not just attention components. As shown in Table 6, TBA consistently outperforms this SkipAt Identity baseline on CIFAR-10 and CIFAR-100F. The accuracy improvements confirm that approximating is more effective than skipping them. Results with

other datasets and models can be found in Section 6.3.3.

Table 6: **TBA vs. SkipAt.** Accuracy scores for ViT-S on CIFAR-10 and CIFAR-100F using 3 different seeds. The "Approx." column specifies the blocks being approximated, where the first value represents the block whose output is used to approximate the second block's output. More results in Table 7.

	CIFAR-10		CIFAR-100F	
Approx.	SkipAt	TBA	SkipAt	TBA
$1 \rightarrow 5$	58.08 ± 0.44	84.93 ± 0.62	32.68 ± 0.70	58.98 ± 0.19
$2 \rightarrow 5$	64.43 ± 2.00	90.97 ± 0.30	41.78 ± 0.45	69.85 ± 0.18
$7 \rightarrow 10$	73.94 ± 0.34	85.81 ± 1.03	45.00 ± 0.31	60.33 ± 0.85
$1 \rightarrow 3$	66.27 ± 0.76	92.09 ± 0.30	42.76 ± 0.75	72.13 ± 0.37
$2 \rightarrow 4$	71.56 ± 1.62	93.03 ± 0.10	50.19 ± 0.38	74.65 ± 0.59
$9 \rightarrow 11$	89.65 ± 0.52	89.16 ± 1.10	70.75 ± 0.39	68.25 ± 0.57
$2 \rightarrow 3$	81.24 ± 0.48	94.87 ± 0.20	60.22 ± 0.75	79.16 ± 0.43
$9 \rightarrow 10$	93.40 ± 0.32	94.23 ± 0.12	76.32 ± 0.30	76.69 ± 0.36
original	95.87 ± 0.08	95.87 ± 0.08	81.29 ± 0.20	81.29 ± 0.20

Table 7: **SkipAt.** Accuracy scores for ViT-S on MNIST, F-MNIST, CIFAR-10, CIFAR-100C, and CIFAR-100F using 3 different seeds. The "Skip" column specifies the blocks being skipped, where the first value represents the starting block (excluded from the skip) and the second value represents the final (included) block.

	Accuracy ↑				
Skip	MNIST	F-MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F
$1 \rightarrow 5$	92.74 ± 0.58	82.25 ± 0.93	58.08 ± 0.44	43.43 ± 0.79	32.68 ± 0.70
$2 \rightarrow 5$	93.78 ± 0.55	84.99 ± 0.51	64.43 ± 2.00	51.39 ± 0.57	41.78 ± 0.45
$7 \rightarrow 10$	91.56 ± 0.46	85.02 ± 1.15	73.94 ± 0.34	59.99 ± 0.73	45.00 ± 0.31
$1 \rightarrow 3$	94.41 ± 0.33	82.82 ± 0.46	66.27 ± 0.76	52.52 ± 0.48	42.76 ± 0.75
$3 \rightarrow 5$	93.96 ± 0.25	86.10 ± 0.15	74.79 ± 1.56	62.53 ± 0.32	54.62 ± 0.52
$2 \rightarrow 4$	94.31 ± 0.48	85.22 ± 0.67	71.56 ± 1.62	59.40 ± 0.38	50.19 ± 0.38
$8 \rightarrow 10$	94.82 ± 0.21	87.77 ± 0.43	85.74 ± 0.32	72.39 ± 0.41	63.79 ± 0.66
$9 \rightarrow 11$	94.80 ± 0.15	88.32 ± 0.46	89.65 ± 0.52	76.40 ± 0.08	70.75 ± 0.39
$0 \rightarrow 1$	95.98 ± 0.13	84.91 ± 0.36	70.90 ± 0.09	57.16 ± 0.41	47.54 ± 0.37
$1 \rightarrow 2$	95.79 ± 0.16	87.07 ± 0.70	83.21 ± 0.52	70.66 ± 0.69	62.23 ± 0.21
$2 \rightarrow 3$	95.14 ± 0.39	85.50 ± 0.62	81.24 ± 0.48	68.63 ± 0.33	60.22 ± 0.75
$3 \rightarrow 4$	95.34 ± 0.58	87.62 ± 1.18	88.25 ± 0.23	77.58 ± 0.46	69.79 ± 0.02
$4 \rightarrow 5$	95.75 ± 0.20	87.26 ± 0.86	86.23 ± 0.63	74.52 ± 0.63	66.69 ± 0.48
$5 \rightarrow 6$	95.77 ± 0.22	86.99 ± 0.33	83.42 ± 0.52	69.62 ± 0.32	61.96 ± 0.55
$6 \rightarrow 7$	95.33 ± 0.08	86.64 ± 1.14	87.57 ± 0.24	75.91 ± 0.20	68.70 ± 0.31
$7 \rightarrow 8$	95.76 ± 0.20	87.50 ± 0.85	88.70 ± 0.46	76.80 ± 0.09	69.33 ± 0.39
$8 \rightarrow 9$	96.28 ± 0.04	88.38 ± 0.83	89.98 ± 0.48	76.45 ± 0.65	71.80 ± 0.22
$9 \rightarrow 10$	95.56 ± 0.47	88.74 ± 1.09	93.40 ± 0.32	82.44 ± 0.44	76.32 ± 0.30
$10 \rightarrow 11$	95.22 ± 0.29	89.39 ± 0.30	93.77 ± 0.69	82.39 ± 0.06	78.68 ± 0.29
original	95.95 ± 0.40	89.01 ± 0.63	95.87 ± 0.08	87.60 ± 0.15	81.28 ± 0.20

Trained approximators We also evaluate whether more complex approximations improve performance. Specifically, we compare TBA to deeper MLP-based translators (MLP and Res-MLP), which are trained for 300 steps using Adam with a learning rate of 1e-3. As shown in Table 8, employing a simple linear transformation to approximate blocks is the optimal choice in most cases. While deeper translators occasionally yield slightly higher accuracy (e.g., 1→5), they incur significant training overhead and require gradient-based optimization. In contrast, TBA is entirely training-free and operates in closed form; no backpropagation or updates are required. This highlights the efficiency–performance trade-off and supports the design choice behind TBA. Refer to Section 6.2.2 for approximator details.

6.3.4. Block Similarities

The analysis in Section 4.1 reveals distinct block-wise similarity patterns within pretrained foundation models. The analysis utilizes the mean of token representations within each block to compute similarities. However, as detailed in Figure 5, these characteristic similarity patterns persist and are qualitatively very similar when using only the CLS token's representation

Table 8: **Approximators Comparison.** Classification accuracy on `ImageNet1k` using `ViT-S` across three seeds. The "Approx." column specifies the blocks being approximated, where the first value represents the block whose output is used to approximate the second block's output. MLP and Res-MLP are trained approximators, while TBA uses a closed-form linear map.

Approx.	TBA	MLP	Res-MLP
$1 \rightarrow 5$	44.04 ± 0.42	**45.79** ± 0.19	45.44 ± 0.12
$2 \rightarrow 5$	**60.38** ± 0.12	60.22 ± 0.08	60.02 ± 0.34
$7 \rightarrow 10$	**35.80** ± 0.11	22.85 ± 0.10	33.01 ± 0.76
$1 \rightarrow 3$	65.31 ± 0.14	**65.45** ± 0.31	64.54 ± 0.25
$2 \rightarrow 4$	**67.52** ± 0.16	67.30 ± 0.12	66.91 ± 0.09
$9 \rightarrow 11$	**46.17** ± 0.25	34.70 ± 0.68	39.01 ± 0.34
$3 \rightarrow 4$	**71.40** ± 0.22	70.78 ± 0.42	70.78 ± 0.10
$9 \rightarrow 10$	**61.82** ± 0.24	53.78 ± 0.19	58.06 ± 0.43

from each block. This suggests a degree of interchangeability for this type of analysis.

Finally, to assess the impact of the similarity measure itself, we conducted an ablation study correlating various metrics with downstream task performance on `ViT-S` and `ImageNet1k` (Figure 6). Our findings reveal that even simple metrics, such as MSE or cosine similarity, when used to identify block-level similarities, exhibit a strong correlation with final accuracy. These results underscore the robustness and flexibility of leveraging such block similarities, indicating compatibility with a range of similarity or distance functions capable of capturing these inter-block relationships.

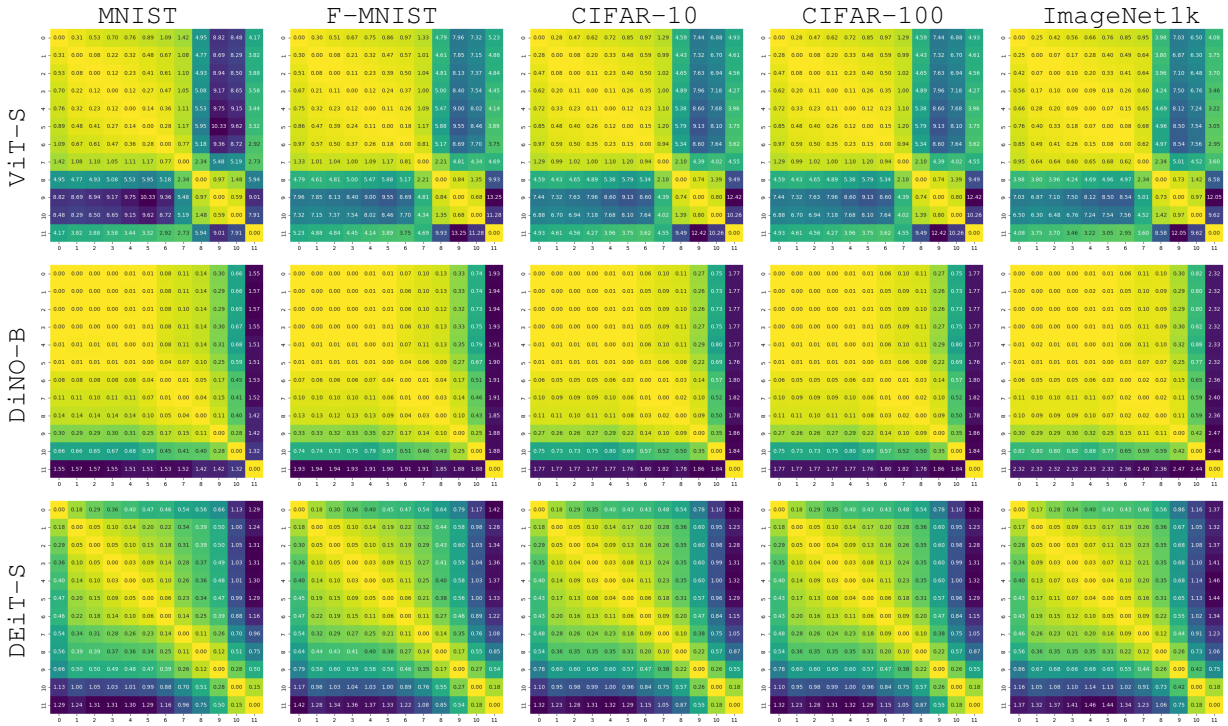

Figure 5: **CLS Block Similarities.** Block-by-block similarities in `ViT-S`, `DiNO-B`, and `DEiT-S` models across five datasets: `MNIST`, `F-MNIST`, `CIFAR-10`, `CIFAR-100` and `ImageNet1k`. Each matrix quantifies the MSE between latent representations of different blocks, showing potential candidates for approximation in pretrained foundation models. The matrices reveal that the similarity structure between computational blocks is predominantly influenced by the model rather than the specific dataset.

Approx.	Params.	`ImageNet1k`
$1 \rightarrow 5$	`15.31M`	44.04 ± 0.42
$2 \rightarrow 5$	`16.94M`	60.38 ± 0.12
$7 \rightarrow 10$	`16.94M`	35.80 ± 0.11
$1 \rightarrow 3$	`18.56M`	64.99 ± 0.29
$3 \rightarrow 5$	`18.56M`	67.27 ± 0.45
$2 \rightarrow 4$	`18.56M`	67.52 ± 0.16
$9 \rightarrow 11$	`18.56M`	47.23 ± 0.24
$2 \rightarrow 3$	`20.19M`	71.26 ± 0.03
$3 \rightarrow 4$	`20.19M`	$\mathbf{71.40} \pm 0.22$
$4 \rightarrow 5$	`20.19M`	70.98 ± 0.16
$9 \rightarrow 10$	`20.19M`	61.82 ± 0.24
-	`21.82M`	$\underline{73.24} \pm 0.13$

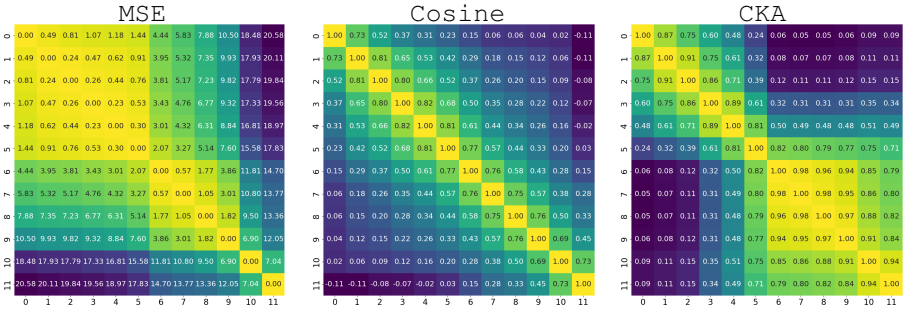

Figure 6: **Correlation Between Similarity Metrics and Accuracy Approximation.** The table on the *Left* reports the accuracy performance of the `ViT-S` encoder is shown with different approximation strategies applied on `ImageNet1k`. While *right* matrices report block-by-block similarities using various similarity metrics. The findings reveal that using MSE or cosine similarity enhances the emergence of the block structure, making it easier to identify similar blocks.

6.3.5. PCA VISUALIZATION

In Section 4.1 we show the effect of TBA on latent representations. In this section, we provide additional visualization using PCA for DiNO-S, DEiT-S, ViT-S, with different datasets and approximating both early and late blocks (see Figures 7 to 11).

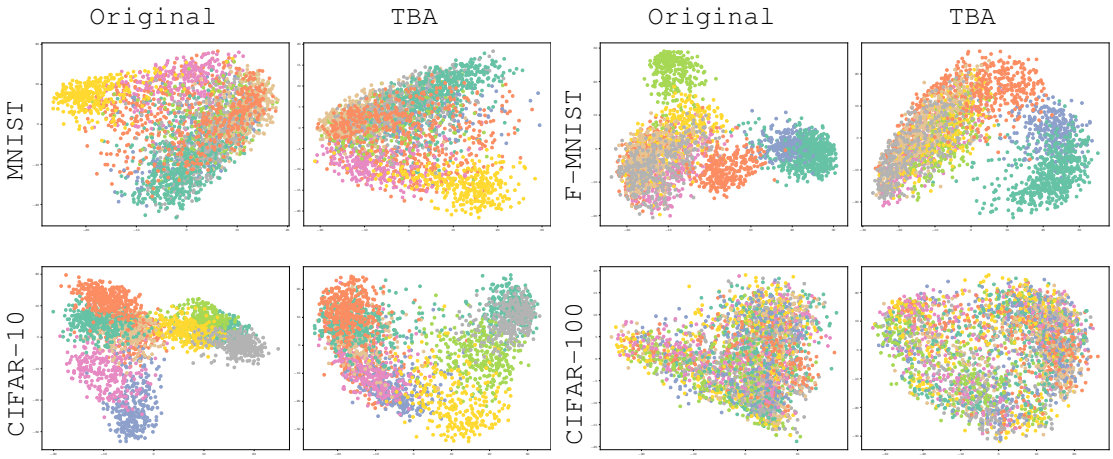

Figure 7: **Last Block Approximation.** PCA visualization of the final layer representations for both the original model and the model with its last block approximated from the preceding one. The representations are generated using the DiNO-S model across four datasets. The plots highlight that the last layer representations in this model are crucial, making it more effective to approximate earlier blocks instead. Note that for CIFAR-100 (bottom right), only the overall structure of the space can be observed, as the 100 classes make it challenging to distinguish labels based on color.

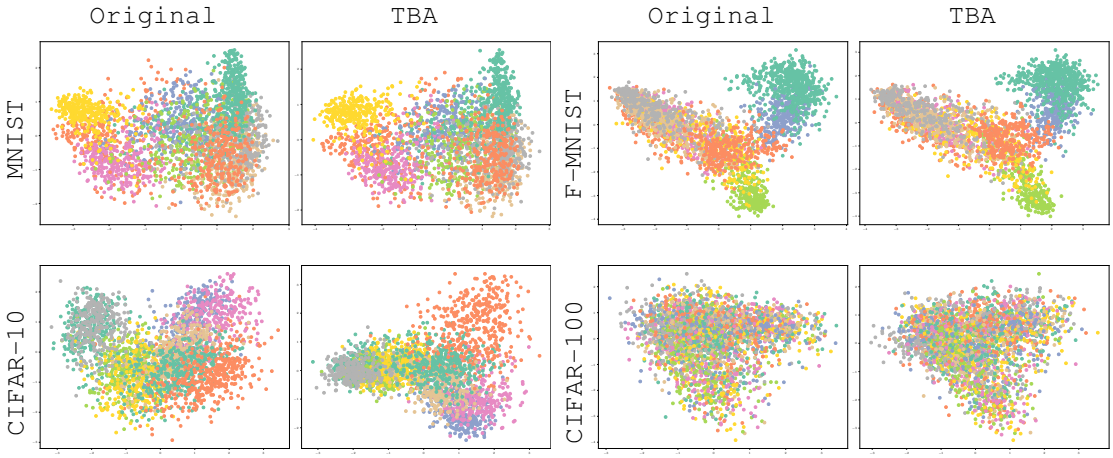

Figure 8: **Last Block Approximation.** PCA visualization of the final layer representations for both the original model and the model with its last block approximated by the preceding one. The representations are generated using the DEiT-S model across four datasets. The plots highlight that in this model, the representations in the last layer are redundant and can be effectively approximated, offering potential performance improvements while reducing model complexity and parameter count. Note that for CIFAR-100 (bottom right), only the overall structure of the space can be observed, as the 100 classes make it challenging to distinguish labels based on color.

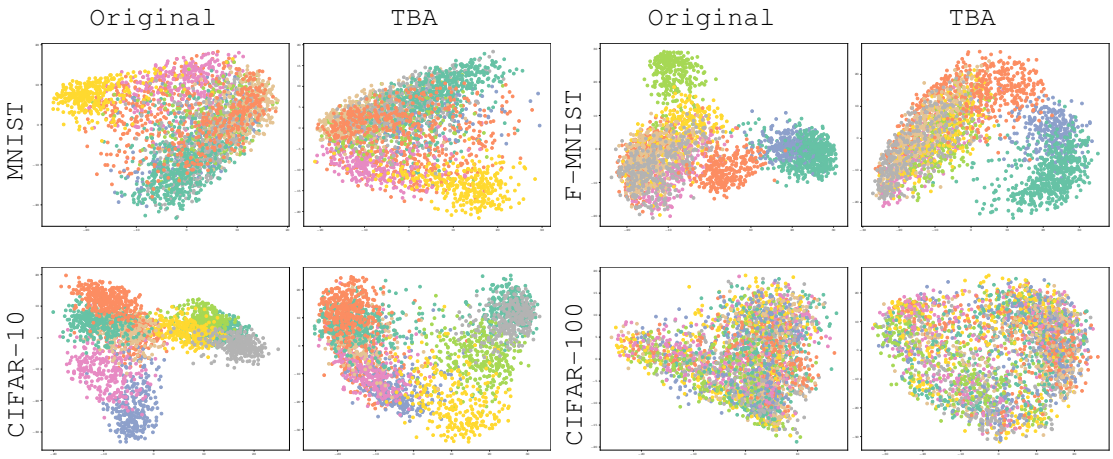

Figure 9: **Last Block Approximation.** PCA visualization of the final layer representations for both the original model and the model with its second block approximated by the preceding one. The representations are generated using the `DiNO-S` model across four datasets. Note that for `CIFAR-100` (bottom right), only the overall structure of the space can be observed, as the 100 classes make it challenging to distinguish labels based on color.

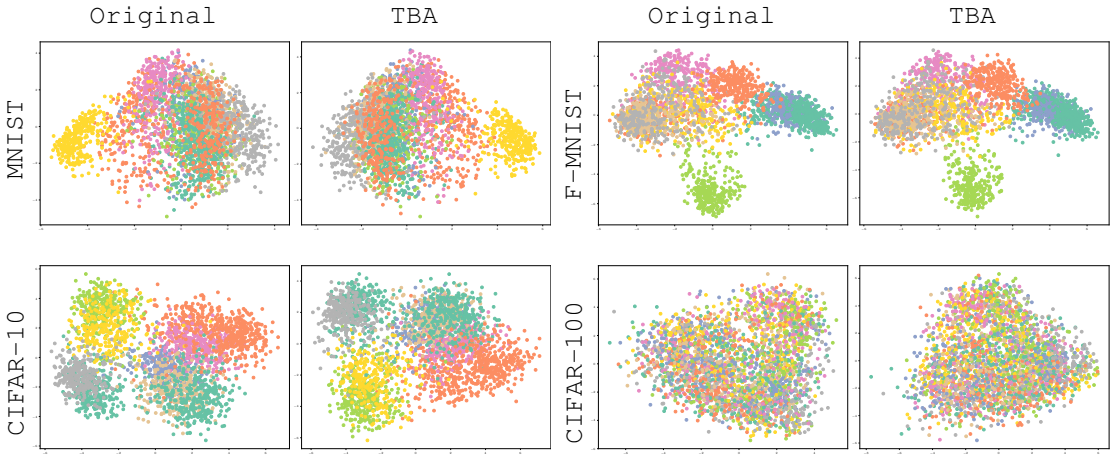

Figure 10: **Last Block Approximation.** PCA visualization of the last layer representations for both the original model and the model with its second block approximated using the previous one. Representations refer to the using `ViT-S` model across four datasets.

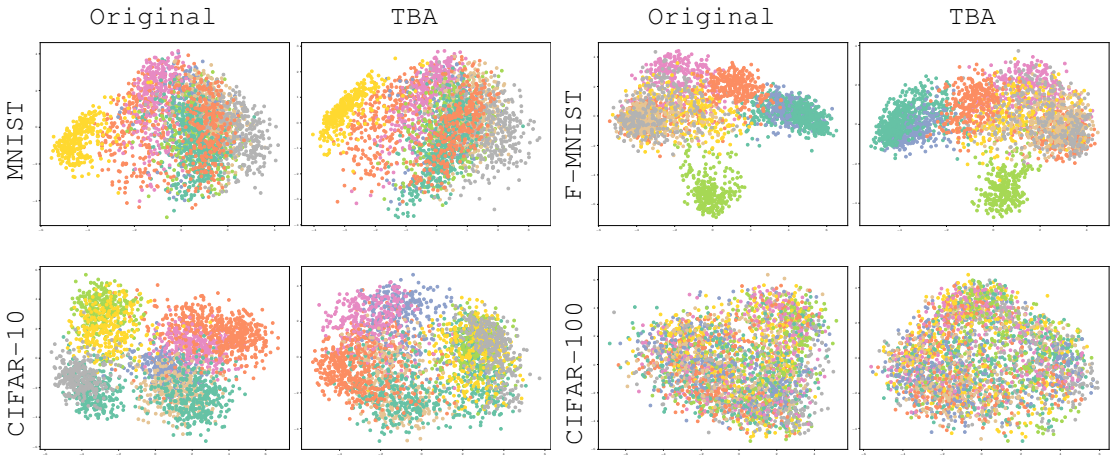

Figure 11: **Last Block Approximation.** PCA visualization of the last layer representations for both the original model and the model with its last block approximated from the previous one. Representations refer to the using `ViT-S` model across four datasets.

6.3.6. IMAGE CLASSIFICATION

This section presents additional experiments that complement and extend those detailed in Section 4.2. Datasets and models are the ones detailed in Tables 2 and 3.

Table 9: **`ViT-S` Image Classification Performance Across Seeds.** Classification accuracy scores for `ViT-S` using multiple datasets, and 3 seeds. The "Approx." column specifies the blocks used for approximation, where the first value represents the block whose output is used to approximate the second block's output, while the "Params." column shows the number of parameters removed by the approximation compared to the original model.

Approx.	Params.	MNIST	F-MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F	ImageNet1k
1 →5	15.31M	92.28 ± 0.81	86.90 ± 0.72	85.07 ± 0.55	68.01 ± 0.31	59.21 ± 0.12	44.04 ± 0.42
2 →5	16.94M	94.76 ± 0.20	88.57 ± 0.31	91.01 ± 0.37	77.77 ± 0.22	69.75 ± 0.36	60.38 ± 0.12
7 →10	16.94M	94.58 ± 0.28	88.44 ± 0.35	87.36 ± 0.17	72.58 ± 0.69	62.03 ± 0.56	35.80 ± 0.11
1 →3	18.56M	94.60 ± 0.78	88.36 ± 0.44	91.97 ± 0.16	79.36 ± 0.54	72.41 ± 0.08	64.99 ± 0.29
2 →4	18.56M	95.08 ± 0.18	88.83 ± 0.21	92.86 ± 0.11	81.45 ± 0.44	74.43 ± 0.27	67.52 ± 0.16
3 →5	18.56M	94.75 ± 0.57	88.81 ± 0.19	94.09 ± 0.06	83.16 ± 0.34	76.17 ± 0.45	67.27 ± 0.45
1 →2, 3 →4	18.56M	94.68 ± 0.69	88.30 ± 0.25	91.91 ± 0.25	79.72 ± 0.16	72.17 ± 0.15	65.38 ± 0.03
1 →2, 4 →5	18.56M	94.58 ± 0.77	88.95 ± 0.07	92.29 ± 0.28	80.14 ± 0.10	72.45 ± 0.35	64.42 ± 0.24
0 →1	20.43M	**95.69** ± 0.29	88.81 ± 0.19	93.68 ± 0.22	83.55 ± 0.23	76.49 ± 0.29	65.11 ± 0.27
1 →2	20.43M	95.40 ± 0.57	88.53 ± 0.63	93.90 ± 0.11	83.98 ± 0.22	76.99 ± 0.26	70.32 ± 0.38
2 →3	20.43M	95.43 ± 0.45	88.93 ± 0.62	94.90 ± 0.26	85.72 ± 0.48	78.96 ± 0.05	71.26 ± 0.03
3 →4	20.43M	95.43 ± 0.39	88.77 ± 0.36	95.05 ± 0.17	85.99 ± 0.37	79.49 ± 0.32	**71.40** ± 0.22
4 →5	20.43M	95.39 ± 0.35	89.18 ± 0.51	**95.41** ± 0.12	86.27 ± 0.27	**79.61** ± 0.14	70.98 ± 0.16
5 →6	20.43M	95.14 ± 0.56	89.30 ± 0.54	94.89 ± 0.27	**86.49** ± 0.33	79.29 ± 0.19	69.25 ± 0.09
6 →7	20.43M	95.11 ± 0.42	88.94 ± 0.66	94.81 ± 0.26	85.33 ± 0.30	78.06 ± 0.17	67.41 ± 0.08
7 →8	20.43M	95.64 ± 0.46	89.41 ± 0.45	94.50 ± 0.34	85.30 ± 0.50	78.03 ± 0.12	66.22 ± 0.10
8 →9	20.43M	95.36 ± 0.47	**89.64** ± 0.37	94.36 ± 0.14	84.66 ± 0.25	77.88 ± 0.20	64.03 ± 0.29
9 →10	20.43M	95.52 ± 0.41	89.57 ± 0.10	94.58 ± 0.27	81.76 ± 0.34	76.45 ± 0.22	61.82 ± 0.24
10 →11	20.43M	94.83 ± 0.20	89.11 ± 0.43	94.08 ± 0.27	82.13 ± 0.70	77.45 ± 0.29	63.92 ± 0.25
original	22.06M	95.59 ± 0.42	89.04 ± 0.85	95.68 ± 0.24	87.61 ± 0.39	81.50 ± 0.39	73.24 ± 0.13

Table 10: **DiNO-S Image Classification Performance Across Seeds.** Classification accuracy scores for DiNO-S using multiple datasets, and 3 seeds. The "Approx." column specifies the blocks used for approximation, where the first value represents the block whose output is used to approximate the second block's output, while the "Params." column shows the number of parameters removed by the approximation compared to the original model.

Approx.	Params.	MNIST	F-MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F	ImageNet1k
1 →5	15.31M	96.25 ± 0.30	86.50 ± 1.42	80.11 ± 0.95	59.15 ± 0.45	51.24 ± 0.51	18.70 ± 0.09
2 →5	16.94M	95.86 ± 0.52	87.99 ± 0.30	85.28 ± 0.99	67.50 ± 1.02	59.57 ± 0.45	40.63 ± 0.59
7 →10	16.94M	96.05 ± 1.44	88.28 ± 1.25	91.00 ± 0.82	78.47 ± 0.61	70.56 ± 0.25	45.66 ± 0.69
1 →3	18.56M	96.61 ± 0.34	88.48 ± 0.61	91.73 ± 0.36	78.62 ± 0.87	72.33 ± 0.37	56.85 ± 0.21
2 →4	18.56M	96.79 ± 0.58	88.34 ± 0.33	91.31 ± 0.16	76.41 ± 0.44	69.71 ± 0.31	60.16 ± 0.41
3 →5	18.56M	96.76 ± 1.02	88.65 ± 0.92	91.00 ± 0.49	75.51 ± 0.45	69.31 ± 0.05	57.47 ± 0.11
1 →2, 3 →4	18.56M	96.71 ± 0.62	88.69 ± 0.46	92.57 ± 0.54	79.16 ± 1.02	72.88 ± 0.57	59.79 ± 0.19
1 →2, 4 →5	18.56M	96.81 ± 0.31	88.67 ± 1.23	93.50 ± 0.26	79.35 ± 1.00	73.55 ± 0.38	58.62 ± 0.25
0 →1	20.43M	96.71 ± 0.79	88.97 ± 1.12	**95.67** ± 0.12	**85.89** ± 0.56	**80.15** ± 0.35	61.25 ± 0.22
1 →2	20.43M	96.69 ± 0.90	88.26 ± 1.10	95.38 ± 0.09	84.86 ± 0.84	79.38 ± 0.23	64.86 ± 0.36
2 →3	20.43M	96.42 ± 0.36	88.31 ± 1.20	94.71 ± 0.33	84.15 ± 0.94	77.74 ± 0.85	65.16 ± 0.69
3 →4	20.43M	96.82 ± 0.68	88.77 ± 0.78	94.87 ± 0.30	83.96 ± 0.62	77.71 ± 0.08	**65.35** ± 0.56
4 →5	20.43M	96.82 ± 0.60	89.15 ± 0.72	94.63 ± 0.26	83.04 ± 0.62	77.13 ± 0.17	64.28 ± 0.24
5 →6	20.43M	96.81 ± 0.85	88.75 ± 0.86	95.33 ± 0.19	84.83 ± 0.04	79.37 ± 0.25	64.88 ± 0.43
6 →7	20.43M	**96.99** ± 0.88	**89.42** ± 0.68	95.21 ± 0.10	83.82 ± 0.53	78.54 ± 0.64	63.61 ± 0.62
7 →8	20.43M	96.76 ± 0.38	89.05 ± 1.29	95.37 ± 0.14	84.57 ± 0.42	78.95 ± 0.37	61.59 ± 0.31
8 →9	20.43M	96.62 ± 0.85	88.45 ± 1.21	95.21 ± 0.36	84.98 ± 0.22	79.35 ± 0.22	61.73 ± 0.43
9 →10	20.43M	96.66 ± 0.33	88.53 ± 0.71	94.55 ± 0.25	83.97 ± 1.25	77.06 ± 0.36	58.56 ± 0.25
10 →11	20.43M	94.61 ± 0.66	86.96 ± 1.18	92.11 ± 0.32	79.85 ± 0.26	73.01 ± 0.51	50.76 ± 0.33
original	22.06M	96.57 ± 0.64	88.07 ± 1.40	96.24 ± 0.08	87.53 ± 0.45	82.04 ± 0.42	67.45 ± 0.45

Table 11: **DEiT-S Image Classification Performance.** Classification accuracy scores for DEiT-S using multiple datasets, and 3 seeds. The "Approx." column specifies the blocks used for approximation, where the first value represents the block whose output is used to approximate the second block's output, while the "Params." column shows the number of parameters removed by the approximation compared to the original model.

Approx.	Params.	MNIST	F-MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F	ImageNet1k
1 →5	15.31M	93.84 ± 0.61	85.70 ± 0.16	78.35 ± 0.17	60.12 ± 0.17	50.57 ± 0.29	43.70 ± 0.27
2 →5	16.94M	95.29 ± 0.02	87.33 ± 0.19	85.73 ± 0.31	69.90 ± 0.16	60.55 ± 0.16	62.04 ± 0.21
7 →10	16.94M	95.85 ± 0.20	88.13 ± 0.13	89.17 ± 0.04	75.69 ± 0.07	69.15 ± 0.33	57.48 ± 0.06
1 →3	18.56M	95.51 ± 0.14	87.14 ± 0.14	85.19 ± 0.19	70.42 ± 0.16	61.76 ± 0.13	66.63 ± 0.14
2 →4	18.56M	95.74 ± 0.13	87.90 ± 0.26	88.95 ± 0.05	75.93 ± 0.23	66.60 ± 0.50	70.00 ± 0.32
3 →5	18.56M	95.98 ± 0.10	87.90 ± 0.21	89.16 ± 0.21	75.70 ± 0.08	66.76 ± 0.06	68.78 ± 0.07
1 →2, 3 →4	18.56M	95.14 ± 0.17	87.11 ± 0.26	85.98 ± 0.07	70.76 ± 0.20	62.33 ± 0.24	66.82 ± 0.07
1 →2, 4 →5	18.56M	95.28 ± 0.04	86.96 ± 0.08	85.43 ± 0.25	70.26 ± 0.14	61.66 ± 0.13	66.04 ± 0.13
0 →1	20.43M	95.97 ± 0.26	87.34 ± 0.22	85.00 ± 0.27	70.81 ± 0.29	61.95 ± 0.39	62.55 ± 0.12
1 →2	20.43M	95.59 ± 0.09	87.31 ± 0.31	86.66 ± 0.15	72.77 ± 0.27	64.21 ± 0.20	70.35 ± 0.17
2 →3	20.43M	**96.22** ± 0.21	88.03 ± 0.14	90.19 ± 0.13	78.17 ± 0.20	69.89 ± 0.25	73.27 ± 0.06
3 →4	20.43M	96.00 ± 0.32	88.19 ± 0.05	90.50 ± 0.10	78.40 ± 0.08	70.25 ± 0.20	73.03 ± 0.10
4 →5	20.43M	95.93 ± 0.12	87.96 ± 0.13	90.38 ± 0.26	77.89 ± 0.27	69.52 ± 0.32	72.19 ± 0.20
5 →6	20.43M	95.96 ± 0.10	88.14 ± 0.13	90.18 ± 0.14	77.48 ± 0.06	69.40 ± 0.20	71.24 ± 0.09
6 →7	20.43M	95.90 ± 0.28	88.38 ± 0.03	90.86 ± 0.06	78.35 ± 0.25	70.45 ± 0.53	71.01 ± 0.19
7 →8	20.43M	96.07 ± 0.23	87.93 ± 0.16	90.48 ± 0.13	78.25 ± 0.20	70.87 ± 0.16	69.88 ± 0.19
8 →9	20.43M	95.97 ± 0.18	88.43 ± 0.24	90.78 ± 0.17	78.31 ± 0.35	71.07 ± 0.17	70.12 ± 0.09
9 →10	20.43M	96.12 ± 0.26	**88.46** ± 0.19	90.90 ± 0.20	79.33 ± 0.25	71.74 ± 0.09	72.34 ± 0.16
10 →11	20.43M	96.00 ± 0.28	88.18 ± 0.11	**91.07** ± 0.18	**79.72** ± 0.18	**71.95** ± 0.17	**73.97** ± 0.17
original	22.06M	96.02 ± 0.20	88.05 ± 0.12	90.63 ± 0.22	78.90 ± 0.25	71.13 ± 0.26	73.93 ± 0.14

Table 12: **ViT-T Image Classification Performance.** Classification accuracy scores for ViT-T using multiple datasets, and 3 seeds. The "Approx." column specifies the blocks used for approximation, where the first value represents the block whose output is used to approximate the second block's output, while the "Params." column shows the number of parameters removed by the approximation compared to the original model.

Approx.	Params.	MNIST	F-MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F	ImageNet1k
1 →5	15.31M	87.66 ± 0.57	85.10 ± 0.42	73.68 ± 0.46	53.46 ± 0.29	44.61 ± 0.42	22.21 ± 0.39
2 →5	16.94M	90.59 ± 0.79	85.84 ± 0.18	82.41 ± 0.11	62.87 ± 0.21	54.68 ± 0.21	35.14 ± 0.38
7 →10	16.94M	92.41 ± 0.47	86.50 ± 0.19	82.48 ± 0.85	69.26 ± 0.65	61.15 ± 0.28	39.03 ± 0.13
1 →3	18.56M	90.55 ± 1.04	85.91 ± 0.22	80.48 ± 0.29	63.43 ± 0.25	54.57 ± 0.32	43.68 ± 0.26
2 →4	18.56M	92.81 ± 0.56	86.58 ± 0.05	86.85 ± 0.17	70.49 ± 0.30	63.53 ± 0.23	49.94 ± 0.27
3 →5	18.56M	91.84 ± 0.69	86.80 ± 0.04	88.00 ± 0.04	72.67 ± 0.30	65.66 ± 0.14	48.48 ± 0.37
1 →2, 3 →4	18.56M	91.94 ± 0.78	86.71 ± 0.20	83.43 ± 0.41	66.92 ± 0.42	60.07 ± 0.48	45.14 ± 0.15
1 →2, 4 →5	18.56M	90.86 ± 0.66	86.57 ± 0.24	84.61 ± 0.14	68.07 ± 0.55	60.11 ± 0.61	44.84 ± 0.26
0 →1	20.43M	91.74 ± 0.48	86.22 ± 0.23	83.32 ± 0.22	68.58 ± 0.41	61.05 ± 0.36	44.12 ± 0.20
1 →2	20.43M	91.65 ± 0.61	86.26 ± 0.24	85.84 ± 0.08	71.12 ± 0.06	63.85 ± 0.37	54.34 ± 0.44
2 →3	20.43M	92.89 ± 0.18	86.49 ± 0.06	88.89 ± 0.08	74.90 ± 0.25	68.03 ± 0.37	57.83 ± 0.07
3 →4	20.43M	93.10 ± 0.43	**87.34** ± 0.03	89.73 ± 0.37	76.45 ± 0.17	70.04 ± 0.35	57.55 ± 0.14
4 →5	20.43M	92.43 ± 0.20	87.22 ± 0.10	90.11 ± 0.32	76.40 ± 0.42	69.97 ± 0.37	55.91 ± 0.10
5 →6	20.43M	**93.57** ± 0.11	86.80 ± 0.13	90.17 ± 0.27	76.47 ± 0.35	70.69 ± 0.49	55.43 ± 0.38
6 →7	20.43M	92.13 ± 0.37	86.77 ± 0.02	87.73 ± 0.22	72.35 ± 0.31	66.73 ± 0.45	47.39 ± 0.45
7 →8	20.43M	93.20 ± 0.06	86.90 ± 0.30	88.58 ± 0.26	75.80 ± 0.29	69.28 ± 0.41	53.48 ± 0.24
8 →9	20.43M	92.76 ± 0.11	87.18 ± 0.17	89.57 ± 0.33	76.43 ± 0.50	71.07 ± 0.33	56.07 ± 0.77
9 →10	20.43M	92.39 ± 0.10	86.74 ± 0.18	89.86 ± 0.31	77.34 ± 0.04	71.70 ± 0.37	57.45 ± 0.29
10 →11	20.43M	90.92 ± 0.48	86.89 ± 0.12	**90.98** ± 0.21	**78.85** ± 0.38	**72.29** ± 0.42	**58.94** ± 0.22
original	22.06M	93.22 ± 0.18	86.99 ± 0.29	91.29 ± 0.06	79.27 ± 0.23	73.45 ± 0.38	63.02 ± 0.22

Table 13: **ViT-B Image Classification Performance.** Classification accuracy scores for ViT-B using multiple datasets, and 3 seeds. The "Approx." column specifies the blocks used for approximation, where the first value represents the block whose output is used to approximate the second block's output, while the "Params." column shows the number of parameters removed by the approximation compared to the original model.

		Accuracy ↑				
Approx.	Params.	MNIST	F-MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F
1 → 5	-25.99M	87.06 ± 0.53	84.33 ± 0.61	73.54 ± 0.57	51.67 ± 1.10	38.98 ± 0.72
2 → 5	-19.49M	94.20 ± 0.21	87.80 ± 0.24	87.10 ± 0.83	71.68 ± 0.50	61.19 ± 0.37
1 → 3	-13M	96.51 ± 0.42	88.72 ± 0.41	93.71 ± 0.13	83.05 ± 0.23	74.74 ± 0.29
3 → 5	-13M	95.59 ± 0.09	88.28 ± 0.20	93.11 ± 0.06	83.50 ± 0.17	74.35 ± 0.47
2 → 4	-13M	96.21 ± 0.33	89.21 ± 0.64	94.59 ± 0.32	85.13 ± 0.24	76.82 ± 0.41
8 → 10	-13M	96.54 ± 0.21	**89.72** ± 0.52	95.05 ± 0.26	85.78 ± 0.37	79.62 ± 0.14
9 → 11	-13M	95.59 ± 0.52	89.49 ± 0.26	93.22 ± 0.56	82.23 ± 0.44	76.33 ± 0.10
3 → 4	-6.5M	96.86 ± 0.35	89.69 ± 1.09	**96.18** ± 0.09	**89.18** ± 0.06	**82.50** ± 0.17
4 → 5	6.5M	96.55 ± 0.23	89.13 ± 0.50	95.39 ± 0.23	87.43 ± 0.15	80.30 ± 0.16
0 → 1	-6.5M	96.75 ± 0.29	88.97 ± 0.26	93.74 ± 0.15	84.49 ± 0.20	76.54 ± 0.29
1 → 2	-6.5M	96.88 ± 0.01	89.29 ± 0.24	95.63 ± 0.11	87.46 ± 0.20	80.64 ± 0.23
2 → 3	-6.5M	**96.91** ± 0.17	89.69 ± 0.61	96.00 ± 0.18	88.38 ± 0.13	81.59 ± 0.35
-	86.39M	95.61 ± 0.22	89.64 ± 0.57	96.25 ± 0.17	89.52 ± 0.23	83.41 ± 0.20

Table 14: **ViT-L Image Classification Performance.** Classification accuracy scores for ViT-L using multiple datasets, and 3 seeds. The "Approx." column specifies the blocks used for approximation, where the first value represents the block whose output is used to approximate the second block's output, while the "Params." column shows the number of parameters removed by the approximation compared to the original model.

		Accuracy ↑					
Approx.	Params.	MNIST	F-MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F	ImageNet1k
1→12	-127.02M	87.87 ± 0.59	81.54 ± 0.07	60.79 ± 1.15	42.70 ± 0.52	30.12 ± 0.41	3.70 ± 0.17
0→4	-46.19M	97.09 ± 0.41	87.54 ± 0.79	89.94 ± 0.43	74.88 ± 0.59	65.79 ± 0.09	44.97 ± 0.10
1→5	-46.19M	83.51 ± 2.49	81.05 ± 0.51	77.57 ± 0.92	59.05 ± 2.30	47.15 ± 0.75	38.31 ± 0.13
2→5	-34.64M	85.40 ± 1.19	78.15 ± 1.00	78.60 ± 1.25	60.42 ± 1.72	46.59 ± 1.23	21.21 ± 0.39
7→10	-34.64M	**97.91** ± 0.04	**90.13** ± 0.67	97.17 ± 0.11	91.13 ± 0.36	84.97 ± 0.15	74.52 ± 0.19
0→2, 3→4	-34.64M	96.91 ± 0.29	88.88 ± 0.74	94.61 ± 0.11	84.16 ± 0.35	76.20 ± 0.28	67.98 ± 0.26
1→2, 3→4	-23.09M	97.22 ± 0.16	89.57 ± 0.70	97.06 ± 0.09	90.45 ± 0.08	84.53 ± 0.25	76.55 ± 0.15
1→3	-23.09M	97.18 ± 0.09	89.32 ± 0.54	96.64 ± 0.17	88.86 ± 0.07	82.67 ± 0.10	75.63 ± 0.18
2→4	-23.09M	97.32 ± 0.34	89.51 ± 0.90	97.26 ± 0.02	90.71 ± 0.17	84.44 ± 0.17	76.66 ± 0.21
3→5	-23.09M	87.04 ± 0.51	81.96 ± 0.65	83.09 ± 0.73	64.90 ± 2.29	50.10 ± 1.63	26.53 ± 0.42
5→7	-23.09M	95.86 ± 0.40	88.19 ± 0.51	78.17 ± 0.49	60.02 ± 0.45	40.53 ± 0.25	6.41 ± 0.21
0→1	-11.55M	96.83 ± 0.19	89.28 ± 0.95	96.94 ± 0.06	90.15 ± 0.35	84.15 ± 0.11	76.85 ± 0.03
1→2	-11.55M	96.98 ± 0.13	89.25 ± 0.54	97.17 ± 0.12	91.12 ± 0.19	85.45 ± 0.43	77.63 ± 0.25
2→3	-11.55M	97.20 ± 0.14	89.65 ± 0.61	97.42 ± 0.24	91.53 ± 0.22	86.07 ± 0.09	78.07 ± 0.03
3→4	-11.55M	97.20 ± 0.18	89.83 ± 0.63	97.51 ± 0.09	91.76 ± 0.30	86.17 ± 0.21	**78.43** ± 0.08
4→5	-11.55M	94.16 ± 0.23	85.36 ± 0.18	83.66 ± 0.41	67.76 ± 1.74	54.60 ± 0.29	18.46 ± 0.60
5→6	-11.55M	95.82 ± 0.37	89.34 ± 0.56	82.35 ± 1.51	67.17 ± 1.13	50.65 ± 0.84	14.36 ± 0.07
6→7	-11.55M	97.20 ± 0.09	89.78 ± 0.36	97.58 ± 0.11	91.60 ± 0.06	85.96 ± 0.22	78.05 ± 0.36
7→8	-11.55M	97.43 ± 0.10	90.17 ± 0.44	97.58 ± 0.06	**92.03** ± 0.11	86.13 ± 0.14	78.23 ± 0.10
8→9	-11.55M	97.14 ± 0.15	89.98 ± 0.77	97.72 ± 0.17	91.98 ± 0.36	86.46 ± 0.09	78.01 ± 0.22
9→10	-11.55M	97.20 ± 0.19	89.80 ± 0.94	97.63 ± 0.20	91.68 ± 0.23	86.13 ± 0.32	77.68 ± 0.06
10→11	-11.55M	97.06 ± 0.01	89.99 ± 0.60	**97.80** ± 0.14	91.82 ± 0.28	86.62 ± 0.18	77.63 ± 0.22
11→12	-11.55M	97.04 ± 0.20	89.81 ± 0.84	97.58 ± 0.09	91.51 ± 0.23	86.05 ± 0.32	77.69 ± 0.31
12→13	-11.55M	96.98 ± 0.05	89.68 ± 0.86	97.53 ± 0.05	91.52 ± 0.39	85.70 ± 0.25	77.93 ± 0.28
13→14	-11.55M	96.96 ± 0.18	89.79 ± 1.04	97.43 ± 0.04	91.44 ± 0.09	85.99 ± 0.34	77.99 ± 0.40
14→15	-11.55M	97.04 ± 0.09	89.56 ± 0.58	97.38 ± 0.02	91.27 ± 0.37	85.76 ± 0.25	77.74 ± 0.37
15→16	-11.55M	97.11 ± 0.19	89.93 ± 0.76	97.35 ± 0.12	91.37 ± 0.21	86.12 ± 0.23	78.18 ± 0.60
16→17	-11.55M	96.98 ± 0.08	89.80 ± 0.83	97.55 ± 0.08	91.55 ± 0.13	86.28 ± 0.18	78.08 ± 0.06
17→18	-11.55M	97.26 ± 0.17	89.87 ± 0.68	97.44 ± 0.06	91.34 ± 0.11	85.98 ± 0.54	77.88 ± 0.08
18→19	-11.55M	97.00 ± 0.18	89.90 ± 0.63	97.70 ± 0.10	91.35 ± 0.26	85.83 ± 0.05	77.37 ± 0.14
19→20	-11.55M	97.28 ± 0.13	89.92 ± 0.57	97.54 ± 0.08	91.45 ± 0.45	86.27 ± 0.31	77.13 ± 0.35
20→21	-11.55M	97.12 ± 0.03	89.92 ± 0.67	97.67 ± 0.09	91.37 ± 0.27	86.62 ± 0.03	77.34 ± 0.08
21→22	-11.55M	96.79 ± 0.19	89.83 ± 1.49	97.65 ± 0.06	91.54 ± 0.32	**86.70** ± 0.13	77.09 ± 0.14
22→23	-11.55M	97.03 ± 0.13	89.57 ± 1.18	97.32 ± 0.15	91.56 ± 0.31	86.55 ± 0.02	78.21 ± 0.08
original	304.35M	96.92 ± 0.11	89.79 ± 0.84	97.52 ± 0.08	91.68 ± 0.32	86.48 ± 0.12	78.37 ± 0.21

6.3.7. TRANSFORMATION GENERALIZATION

In this section, we extend the analysis to evaluate the model's ability to generalize transformations across datasets to different approximation blocks and models (i.e., DiNO-S).

Table 15: **ViT-S Generalization Results.** Classification accuracies for approximating ViT-S blocks with a linear transformation learned on one dataset and applied to others. Datasets are MNIST, CIFAR-10, CIFAR-100C and CIFAR-100F. The "Approx." column specifies the blocks being approximated, where the first value represents the block whose output is used to approximate the second block's output, while the "Source" column names the dataset used to compute the transformation.

			Accuracy ↑			
Encoder	Approx.	Source	MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F
ViT-S	2 → 3	MNIST	94.11	57.13	41.89	28.50
		CIFAR-10	89.58	95.08	85.32	77.92
		CIFAR-100	89.63	95.00	85.50	77.74
	3 → 4	MNIST	93.52	10.36	8.97	3.09
		CIFAR-10	88.02	95.18	86.14	78.52
		CIFAR-100	88.21	94.82	85.92	78.09
	4 → 5	MNIST	93.96	38.40	25.56	16.52
		CIFAR-10	78.36	95.31	85.84	78.20
		CIFAR-100	80.11	94.98	86.01	78.14
	9 → 10	MNIST	89.73	74.41	59.78	44.40
		CIFAR-10	82.28	92.39	71.63	57.17
		CIFAR-100	54.12	85.60	77.37	61.81
	1 → 3	MNIST	92.79	16.17	11.09	3.84
		CIFAR-10	80.41	90.63	75.59	65.98
		CIFAR-100	81.24	89.98	76.27	66.26
	3 → 5	MNIST	88.22	15.17	8.52	2.03
		CIFAR-10	61.68	93.57	80.24	71.76
		CIFAR-100	64.18	92.77	80.56	72.43
	2 → 4	MNIST	92.74	17.24	12.27	4.27
		CIFAR-10	63.52	92.14	79.80	70.52
		CIFAR-100	66.05	91.21	79.57	70.16
	8 → 10	MNIST	86.77	36.61	30.79	15.10
		CIFAR-10	24.29	80.81	48.73	31.74
		CIFAR-100	38.89	59.12	64.07	43.20
	9 → 11	MNIST	77.19	31.40	18.79	4.32
		CIFAR-10	49.65	76.61	50.48	25.57
		CIFAR-100	35.61	68.40	55.67	31.59
	2 → 5	MNIST	81.11	13.09	6.74	2.24
		CIFAR-10	37.16	88.70	67.99	57.24
		CIFAR-100	39.60	86.75	70.00	58.90
	7 → 10	MNIST	85.04	33.28	19.26	4.59
		CIFAR-10	20.67	69.49	34.65	17.18
		CIFAR-100	30.00	48.19	53.16	26.97
	1 → 5	MNIST	69.44	10.36	5.38	1.56
		CIFAR-10	39.49	76.98	48.11	36.38
		CIFAR-100	36.94	72.48	51.03	38.75

Table 16: **DiNO-S Generalization Results.** Classification accuracies for approximating DiNO-S blocks with a linear transformation learned on one dataset and applied to others. Datasets are MNIST, CIFAR-10, CIFAR-100C and CIFAR-100F. The "Approx." column specifies the blocks being approximated, where the first value represents the block whose output is used to approximate the second block's output, while the "Source" column names the dataset used to compute the transformation.

			Accuracy ↑			
Encoder	Approx.	Source	MNIST	CIFAR-10	CIFAR-100C	CIFAR-100F
DiNO-S	$2 \rightarrow 3$	MNIST	93.04	58.24	37.95	27.62
		CIFAR-10	86.16	94.11	82.37	75.26
		CIFAR-100	86.39	93.78	82.28	75.29
	$3 \rightarrow 4$	MNIST	92.33	62.78	38.18	27.52
		CIFAR-10	84.70	94.37	81.93	74.69
		CIFAR-100	83.72	94.10	82.02	74.59
	$4 \rightarrow 5$	MNIST	91.64	57.39	36.97	26.02
		CIFAR-10	70.87	93.65	80.38	73.84
		CIFAR-100	71.51	92.98	79.96	73.54
	$9 \rightarrow 10$	MNIST	83.39	38.85	20.20	13.10
		CIFAR-10	45.69	88.70	61.71	50.46
		CIFAR-100	60.57	76.58	76.77	61.29
	$1 \rightarrow 3$	MNIST	90.60	22.30	11.76	5.47
		CIFAR-10	78.51	89.72	74.58	65.04
		CIFAR-100	79.80	89.28	74.75	64.92
	$3 \rightarrow 5$	MNIST	87.54	24.55	11.93	6.67
		CIFAR-10	63.66	87.17	66.16	58.36
		CIFAR-100	64.26	84.40	66.43	58.51
	$2 \rightarrow 4$	MNIST	90.54	19.14	9.99	4.99
		CIFAR-10	62.32	88.03	68.53	59.23
		CIFAR-100	64.89	86.98	68.54	59.15
	$8 \rightarrow 10$	MNIST	80.88	22.27	10.30	6.25
		CIFAR-10	25.67	85.07	48.44	35.42
		CIFAR-100	29.81	67.51	67.59	47.97
	$9 \rightarrow 11$	MNIST	27.79	9.93	7.30	1.67
		CIFAR-10	15.94	59.66	19.22	7.62
		CIFAR-100	15.71	40.73	32.06	12.17
	$2 \rightarrow 5$	MNIST	82.67	10.77	5.85	2.85
		CIFAR-10	49.78	73.83	46.89	38.80
		CIFAR-100	48.24	67.62	46.85	38.36
	$7 \rightarrow 10$	MNIST	75.50	15.89	10.43	4.24
		CIFAR-10	17.75	76.55	36.68	21.94
		CIFAR-100	19.13	53.86	55.80	33.79
	$1 \rightarrow 5$	MNIST	68.07	11.29	6.29	1.74
		CIFAR-10	49.25	56.93	31.06	22.86
		CIFAR-100	47.81	47.83	30.78	21.78

6.3.8. Analysis of misclassifications

In this section, we examine changes in per-class accuracy and misclassification patterns. As shown in Figure 12, models behave differently at block approximations. `DiNO-S` remains remarkably stable across blocks and classes, with the only degradation appearing for classes dog (when approximating blocks 10 or 11) and deer (for block 10 approximation). `ViT-S` shows a similar drop for class dog on its final block. Instead, the most noticeable hit occurs for class cat when the earlier blocks are approximated. For `DEiT-S`, several mid-to-late block approximations improve accuracy for various classes, whereas the very first block causes a clear relative decline in nearly every class. These observations suggest strategies like preferring late-block approximation for `DEiT-S`, or reserving extra samples for the linear transformation in order to recover the accuracy of difficult classes for the model.

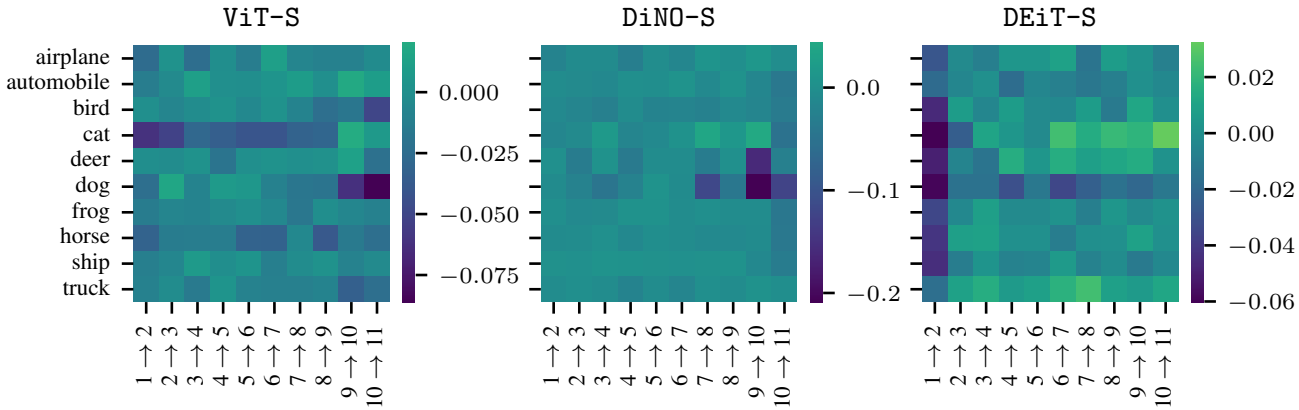

Figure 12: **Per-class accuracy delta on `CIFAR-10` when a single block is approximated in `ViT-S`, `DiNO-S` and `DEiT-S`**. Cell values indicate the relative change in the accuracy with respect to the original model. Brighter (green) cells indicate an accuracy gain for the class, while darker (blue) cells indicate an accuracy drop.

In order to further investigate how the predictions change while approximating blocks, we plot the difference in the normalized confusion matrix before and after the approximation. In Figure 13, we show the delta confusion matrix for `DEiT-S` on `CIFAR-100C`. Also, here we can see how approximating the very first block makes the model puzzling and lose per-class accuracy (i.e., negative delta along the diagonal). On the other hand, approximating the last block acts as a regularizer, resulting in an overall gain in the per-class accuracy and, as a consequence, fewer misclassifications (negative deltas off-diagonal). This supports results shown in Figure 12 and Table 1.

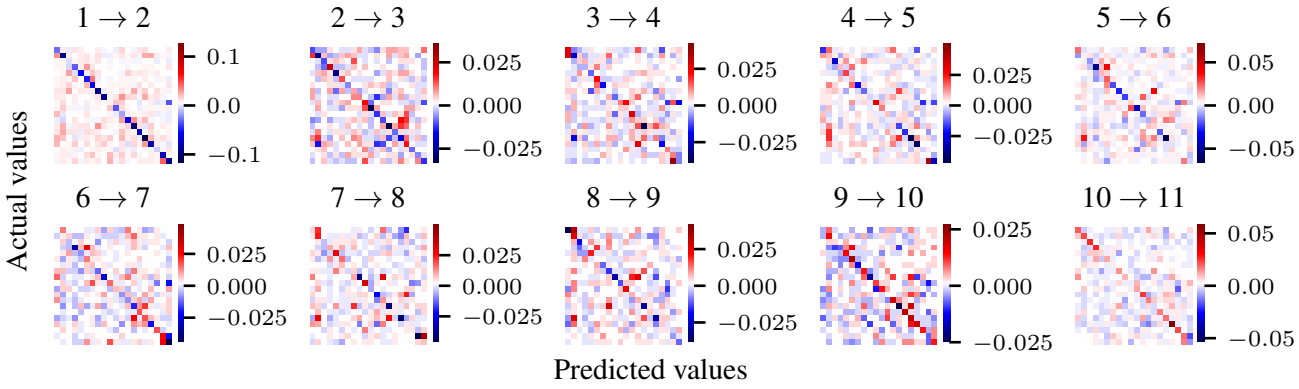

Figure 13: **Normalized relative confusion matrix when single blocks are approximated for `DEiT-S` on `CIFAR-100C`**. Diagonal cells capture the per-class change in accuracy, whereas off-diagonal cells capture changes in misclassifications between classes. Red (positive) values on the diagonal mean the approximation improves that class's accuracy. Red off-diagonal values mean more misclassifications. Conversely, blue (negative) off-diagonal values indicate fewer misclassifications, and blue values on the diagonal indicate a drop in per-class accuracy.

Additionally, Figure 14 shows representative CIFAR-10 images that become misclassified after approximating a block of ViT-S. The patterns we observe mirror the trends in Figures 12 and 13: when approximating earlier blocks, we observe many images belonging to class cat to be misclassified. Instead, when approximating later blocks, we observe images of the class dog to be misclassified. Together, these qualitative examples show that understanding these block-specific vulnerabilities allows us to steer the approximation procedure, informing choices about which blocks to approximate based on the observed impact on the final model's class-wise performance.

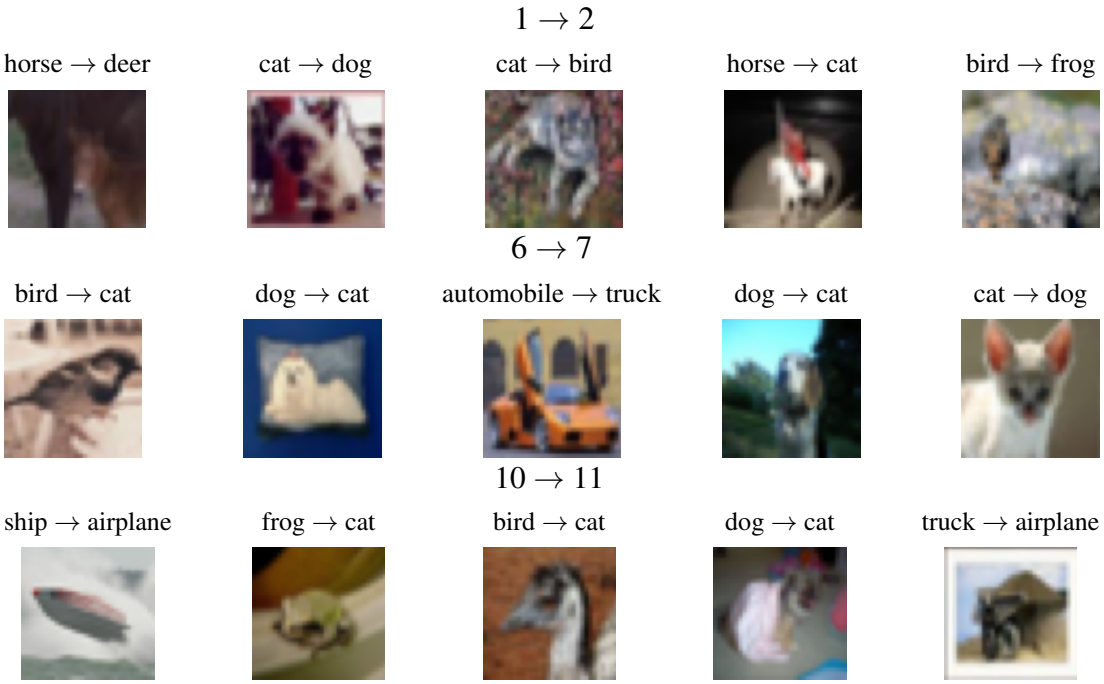

Figure 14: **Misclassified samples after approximating blocks of ViT-S**. Images from CIFAR-10 whose label *flips from correct to incorrect* when specific blocks are approximated. The title reports the true class followed by the wrong prediction.

6.4. Extended related work

Measuring similarities A range of metrics have been introduced to assess the similarity between latent spaces generated by different NNs (Klabunde et al., 2023; Ballester et al., 2023). One established approach is Canonical Correlation Analysis (CCA) (Hotelling, 1992), known for its invariance to linear transformations. Variants of CCA, such as Singular Value CCA (SVCCA) (Raghu et al., 2017), aim to enhance robustness, while techniques like Projection Weighted CCA (PWCCA) (Morcos et al., 2018) mitigate sensitivity to small perturbations. Another widely used metric, Centered Kernel Alignment (CKA) (Kornblith et al., 2019), captures the similarity between latent spaces while ignoring orthogonal transformations. However, recent work (Davari et al., 2022) highlights that this metric can be sensitive to shifts in the latent space. Additionally, Barannikov et al. (2021) proposes a method to compare two data representations by measuring the multi-scale topological dissimilarity, while Fumero et al. (2024) leverages the principles of spectral geometry to model and analyze the relationships between distinct latent spaces.

Leveraging similarities Valeriani et al. (2024) examine the intrinsic dimensionality and neighbor compositions of representations in transformer models. Kvinge et al. (2022) investigate how models process variations in data points across layers, while (Nguyen et al., 2020) assess the impact of network depth and width on hidden representations. Additionally, Crisostomi et al. (2023) study the conditions under which two latent spaces can be merged into a unified one. Moschella et al. (2023) construct a unified space shared by different NNs, enabling zero-shot stitching of independently trained models across different modalities (Norelli et al., 2023). More recently, Cannistraci et al. (2024) enable model stitching without explicit assumptions about the transformation class connecting the latent manifold embeddings, or with only partial correspondence between latent spaces (Cannistraci et al., 2023). Finally, Lähner & Moeller (2024); Maiorca et al. (2024) demonstrate that representations learned by distinct NNs can be aligned using simple transformations.