Harnessing Orthogonality to Train Low-Rank Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

In the realm of neural network training, the question of what is truly being learned beyond mathematical optimization has intrigued researchers for decades. This study delves into the essence of neural network weights. By leveraging the principles of singular value decomposition, we explore the hypothesis that the orthogonal bases of the low-rank decomposition of neural network weights stabilize during training, and provide experimental evidence to support this notion. Building upon this insight, we introduce Orthogonality-Informed Adaptive Low-Rank neural network training. Our novel approach seamlessly integrates into existing training workflows with minimal accuracy loss, as demonstrated by benchmarking on various datasets and well-established network architectures. We find that, through standard tuning procedures, our method surpasses the performance of conventional training setups. Finally, we showcase the effectiveness of our tuned low-rank training procedure by applying it to a state-of-the-art transformer model for time series prediction.

1 Introduction

When you train a neural network, what is it learning? From a technical view, its weights are iteratively adjusted based on the loss function's back-propagated gradients so as to minimize the difference between predicted outputs and target values. But what does this process look like from an intuitive standpoint? Does the mathematical optimization of the loss function impose a structure on the weights? Explaining why a neural network has learned something has intrigued researchers for decades.

A neural network's weights are typically represented as numerical values in a tensor. However, the sizes of these network weights has been increasing for a number of years Bernstein et al. (2021). One method to cope with these large networks on resource-constrained hardware is by representing them with low-rank decompositions Hsu et al. (2022). This approximation factorizes a full-rank matrix, $\boldsymbol{M}$, into two or more matrices, where the inner dimension, r, is smaller than the dimensions of the original matrix. It is expressed as $\boldsymbol{M}_{m\times n} = \boldsymbol{A}_{m\times r}\boldsymbol{B}_{r\times n}$, with the constraint that $r < \min(m, n)$. A common method for finding a low-rank approximation of a given matrix is singular value decomposition (SVD). SVD factorizes a matrix into an orthogonal basis $\boldsymbol{U}$, an orthogonal cobasis $\boldsymbol{V}$, and a diagonal matrix of the singular values sorted in descending order $\boldsymbol{\Sigma}$, as $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$.

It has been shown multiple times that a neural network's weights can be effectively approximated by discarding the least significant singular values in their SVD, reducing the inner dimension of the decomposition Psichogios & Ungar (1994); Fontenla-Romero et al. (2018); Xue et al. (2014); Waleffe & Rekatsinas (2020). Many of these methods train on $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{\Sigma}$ but do not maintain the orthogonality of $\boldsymbol{U}$ and $\boldsymbol{V}$; those that do require additional training time Povey et al. (2018); Schotthöfer et al. (2022). Furthermore, low-rank methods often come at the cost of accuracy Ren & Xia (2023).

We posit that the orthogonal bases of the low-rank decomposition of neural network weights stabilize during training, while subsequent training focuses on the linear mixing of said basis. In this paper, we show experimental evidence of this hypothesis. We then demonstrate how we harness the beneficial attributes of orthogonality by carefully structuring the training process and thus ultimately producing more effective, streamlined models compared to the original full-rank models.

We provide implementations of the most common layer types and a method to wrap arbitrary model architectures for any learning task. We use this approach to train low-rank versions of common (state-of-the-art) networks. We find that, at best, our low-rank networks outperform their full-rank counterparts, and at worst show minimal accuracy loss.

Our contributions include:

- We show evidence that the multi-dimensional weights of a neural network stabilize during training.
- We propose a novel method of Orthogonality-Informed Adaptive Low-Rank (OIALR) neural network training and provide the tools for any researcher to use it on most networks.
- We demonstrate that our training approach seamlessly integrates into existing training workflows with minimal accuracy loss by means of benchmarks on multiple datasets, data modalities, well-known network architectures, and training tasks.
- We show that with tuning, OIALR outperforms conventional full-rank training.

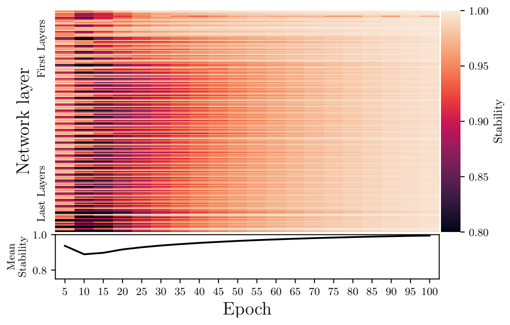
2 Related Work

The objective of network compression is to reduce a network's size and complexity while retaining or improving its performance Xu & McAuley (2023). For example, convolution layers have redundancies which can be removed Tai et al. (2016); Hssayni et al. (2022); Boufssasse et al. (2023). The most common methods of network compression use low-rank approximations. In its simplest form, a low-rank approximation factorizes a matrix $\boldsymbol{M}$ as $\boldsymbol{M}_{m\times n} = \boldsymbol{A}_{m\times r}\boldsymbol{B}_{r\times n}$ where $r < \min(m, n)$. This approach is applied in Cahyawijaya et al. (2021) where network weights are decomposed, then both $\boldsymbol{A}$ and $\boldsymbol{B}$ are trained with standard backpropagation.

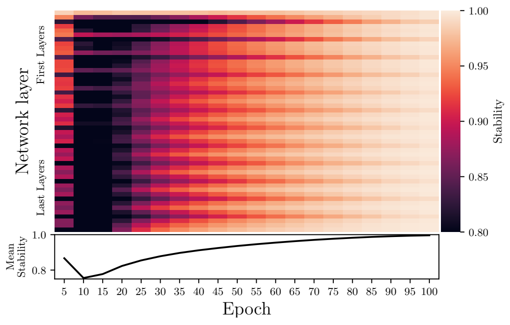
The most common decomposition method for neural networks is SVD, see Wimmer et al. (2023); Cohen & Welling (2016); Nesky & Stout (2020); Psichogios & Ungar (1994); Guo et al. (2023), which comes with pre-existing optimized implementations and has been well-studied outside of the neural network landscape. To the best of our knowledge, the earliest usage of SVD in neural network compression is the SVD-NET Psichogios & Ungar (1994). This method reduces redundant hidden nodes within a simple network by decomposing the network weights with SVD and training $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{V}$. This basic approach has been utilized to great effect in many works, including addressing dataset imbalances Fontenla-Romero et al. (2018), last layer compression Sainath et al. (2013), entire network compression and training Xue et al. (2014), and sparse network training Swaminathan et al. (2020). While many of these methods do not maintain bases orthogonality, those that do show increased network performance Povey et al. (2018); Schotthöfer et al. (2022).

With the plethora of today's pre-trained models, researchers have spent substantial effort to fine-tune them for specific use cases, for which low-rank decompositions have been shown to be effective. In large language models (LLMs), LoRA Hu et al. (2022) demonstrated that adding a low-rank weight component alongside the originally trained weights can adapt LLMs to specialized use cases. DnA Jiang et al. (2022) takes a more extensive approach by reparameterizing the pre-trained model via a "self-supervised alignment step on the target domain using only unlabeled data before conducting the downstream supervised fine-tuning."

A version of the Conformer Gulati et al. (2020) was trained using a specialized low-rank decomposition using two matrices informed by the SVD of the original weight matrix Guo et al. (2023). This method, as well as Fontenla-Romero et al. (2018); Swaminathan et al. (2020); Cohen & Welling (2016); Ceruti et al. (2021); Hsu et al. (2022) and many of the approaches in Xu & McAuley (2023), actively train all low-rank matrices in the decomposition. In DLRT Schotthöfer et al. (2022), each of the three matrices in a model's singular-value decomposed weights is trained in a separate forward-backward pass while maintaining orthogonality. Beginning the training in low rank can be detrimental to network accuracy, as evidenced by previous work from Waleffe & Rekatsinas (2020); Bejani & Ghatee (2020). To correct for this, the corresponding methods opt to transition to a low-rank representation later in the training process. Furthermore, many low-rank methods show better generalization, i.e. reduced overfitting, during training Winata et al. (2020); Phan et al. (2020); Cahyawijaya et al. (2021).



(a) ResNet-RS 101



(b) Vision Transformer B/16

Figure 1: Stability, Equation (1), measurements of two different networks, both trained on ImageNet-2012 Russakovsky et al. (2015) measured at a frequency of five epochs, i.e., the bases of epoch i is compared to the bases of epoch $i - 5$. X-axis is the epoch number, y-axis is the layer of the network with layers closest to the input at the top. Darker colors indicate less stability and lighter colors indicate more stability. Beneath each plot is the mean stability for each epoch shown.

When decomposing a matrix, its orthogonal basis can be viewed as the coordinate system in which the matrix operates. In this vein, it can be extremely useful for explaining the inner working of modern black-box neural networks. Intuitively, orthogonal networks are beneficial from an explainability viewpoint. ExNN Yang et al. (2021) utilized methods to preserve projection orthogonality during training to improve interpretability. Similarly, the Bort optimizer Zhang et al. (2023) aims to improve model explainability using boundedness and orthogonality constraints.

3 Observing Orthogonality Present in Neural Network Training

Like any real-valued, multi-dimensional matrix, each of a neural network's weights can be decomposed into an orthogonal matrix and a linear mixing matrix. We hypothesize that the orthogonal bases in the decomposition of the weights stabilize during training.

To test our hypothesis we determine each weight's orthogonal basis of different neural networks using the semi-orthogonal bases from its compact SVD. Compact SVD loosens the orthogonality constraints on $\boldsymbol{U}$ and $\boldsymbol{V}$ to only semi-orthogonal in the column dimension to reduce the inner dimension r. More precisely, it factorizes a matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ into $\boldsymbol{U\Sigma V}^T$, where $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$ is a diagonal matrix with the singular values along its diagonal, $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ are semi-orthogonal matrices of orthogonal column vectors, and $r \leq \min(m, n)$. To obtain a two-dimensional (2D) representation of a weight tensor with more than two dimensions we maintain the original weight's leading dimension and collapse those remaining. If the resulting 2D weight representation is either not square or not tall and skinny, we use its transpose. We then find a compact SVD of this representation to obtain the orthogonal bases. While the SVD of $\boldsymbol{M}$ is not unique, $\boldsymbol{UV}^T$ is uniquely determined by $\boldsymbol{M}(\boldsymbol{M}^T\boldsymbol{M})^{-1/2}$ and is thus both unique and orthogonal, making it ideal for tracking.

To compare a weights's bases at two distinct times during training, i and j, we define their relative stability S_{ij} as the average of the dot products of each basis vector:

$$S_{ij} = \overline{(\boldsymbol{UV}^T)_i \cdot (\boldsymbol{UV}^T)_j} \tag{1}$$

where $(\boldsymbol{UV}^T)_k$ represents the basis at a given time k during training. Figure 1 shows how the layers' stability evolves for two vastly different network architectures, ResNet-RS 101 Bello et al. (2021) and the VisionTransformer (ViT) B/16 Dosovitskiy et al. (2021), both trained on ImageNet-2012 Russakovsky et al. (2015). In both cases, the bases stabilize during training, as shown by the increase in the relative stability S over the training time.

There are general patterns which quickly emerge from these figures: the weights closer to the output stabilize faster than those toward the input, the stabilization rate differs depending on the layer

Algorithm 1: Updating a weight's basis and cobasis, $\boldsymbol{U}$ and $\boldsymbol{V}$, from the weight's actively trained singular value matrix $\boldsymbol{\Sigma}$. $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{V}$ comprise the current low-rank SVD of with weight $\mathbf{W}$, variables with a $'$ are found from the trained values of $\boldsymbol{\Sigma}$.

```
Input: Basis $\boldsymbol{U}$, cobasis $\boldsymbol{V}$, trained $\boldsymbol{\Sigma}$ matrix
1 $\boldsymbol{U}^{'}, \boldsymbol{\Sigma}^{'}, \boldsymbol{V}^{'T} \leftarrow \texttt{SVD}(\boldsymbol{\Sigma})$                    // $\mathbf{W} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
2 $\boldsymbol{U} \leftarrow \boldsymbol{U}\boldsymbol{U}^{'}, \boldsymbol{V}^T \leftarrow \boldsymbol{V}^{'T}\boldsymbol{V}^T, \boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma}^{'}$       // $\mathbf{W} \approx \boldsymbol{U}\boldsymbol{U}^{'}\boldsymbol{\Sigma}^{'}\boldsymbol{V}^{'T}\boldsymbol{V}^T$
```

type, and the stability increases throughout training. These observations intuitively make sense: the updates for weights closer to the network's output will have fewer layer terms thus the weights will get more direct responses for their 'mistakes,' a linear layer behaves differently than a convolution or attention layer, and as we move towards the end of training each layer has learned most of how it responds to inputs.

We theorize that the stabilization of the orthogonal bases during network training is crucial to the optimization process. Intuitively, this process resembles language learning Chomsky (1972). First, you develop a feeling for the language by acquiring a basic vocabulary and some phrases. Then, you delve into more complex grammar. Once you have learned the grammar, there is no need to re-learn it from scratch, you only need to learn how to apply it. Similarly, neural networks first grasp the task's fundamentals, then discern the roles of each layer, and ultimately fine-tune each layer's functionality for its task.

4 Orthogonality-Informed Adaptive Low-Rank Training

To harness the stabilized orthogonal bases in neural network training, we present a novel algorithm that reduces the number of trainable parameters while maintaining accuracy and overall time-to-train.

As shown in Figure 1, most layers' bases do not stabilize before a few epochs have passed. Therefore, we start training in a traditional full-rank scheme. After a specified number of iterations d, we transition the network's eligible weights to low rank via their SVD. Experimentally, we found that the delay should be about one third of the total number of iterations. At this point, we no longer train $\boldsymbol{U}$ and $\boldsymbol{V}^T$ with backpropagation and train only the square matrix $\boldsymbol{\Sigma}$. After a specified number training steps ν, the bases $\boldsymbol{U}$ and $\boldsymbol{V}^T$ are updated by extracting the new bases from the trained $\boldsymbol{\Sigma}$ matrix using an SVD of $\boldsymbol{\Sigma}$, as outlined in Algorithm 1. After the basis $\boldsymbol{U}$ and cobasis $\boldsymbol{V}^T$ are updated, a new inner rank is found by removing all singular values whose absolute magnitude is less than β times the largest singular value in the current $\boldsymbol{\Sigma}$, where β is a hyperparameter that defaults to 0.1. As the first layers of the network are unstable for longer and likely require most of their ranks, the update of $\boldsymbol{U}$ and $\boldsymbol{V}$ is only applied to the last ℓ layers of the network where $\ell = L \cdot \alpha \cdot u$, where L is the number of network layers, α is a hyperparameter defaulting to 0.1, and u is the number of updates which have been completed. This process repeats until the end of training. Optionally, the first or last layers can be excluded from low-rank training depending on the use case. We provide an outline of our Orthogonality-Informed Adaptive Low-Rank (OIALR) training in Algorithm 2.

5 Experiments

To validate our OIALR training approach, we conducted several experiments using different neural network architectures and datasets. We focused on demonstrating its effectiveness in terms of reducing the number of trainable parameters while maintaining, or improving, network performance and training time. We began by naively applying the OIALR method to a standard neural network training setup to explore what a typical researcher would experience in Sections 5.2 to 5.4. However, as OIALR changes the network structure during training, we expect the hyperparameters to vary from those commonly used for full-rank training. For the final two experiments, Sections 5.5 and 5.6, we tuned the hyperparameters for OIALR using `Propulate` Taubert et al. (2023), an asynchronous evolutionary optimization algorithm designed for usage on high performance clusters which has been shown to be effective for neural architecture searches Coquelin et al. (2021).

Algorithm 2: OIALR training method on an unspecified model M.

```
Input: Model $M$, training steps $t_{\max}$, delay $d$, low-rank update frequency $\nu$, singular value
       cutoff fraction $\beta$, percentage of layers in each low-rank update step size $\alpha$
1  $L \leftarrow$ Number of possible low-rank weights in $M$
2  $\ell \leftarrow L \cdot \alpha$
3  for $t \leftarrow 1$ to $t_{\max}$ do
4      if $t < d$ then
5          Train full-rank network.
6      else if $t = d$ then
7          Convert network to low rank.
8      else if $t \mod \nu = 0$ then
9          for $i \leftarrow L - \ell$ to $L$ do
10             Update weight $i$ with Algorithm 1.
11             Remove singular values $< \beta \cdot \max(\mathbf{\Sigma}_i)$.
12             Reshape $\boldsymbol{U}_i$, $\boldsymbol{V}_i$, $\mathbf{\Sigma}_i$, and optimizer states.
13         $\ell \leftarrow \ell + L \cdot \alpha$
14     else
15         Train low-rank network ($\mathbf{\Sigma}$ for all low-rank weights).
```

Table 1: Baseline and OIALR trainings of ViT-B/16 and ResNet-RS 101 Bello et al. (2021) on ImageNet-2012 Russakovsky et al. (2015). ‘Time’ refers to the time to train in hours, the last row shows the number of trainable parameters in the low-rank model as a percentage of the full-rank model.

	ViT-B/16				ResNet-RS 101			
	Loss	Top-1	Top-5	Time	Loss	Top-1	Top-5	Time
Baseline	**2.16**	**71.64 %**	**89.18 %**	3.29 h	**1.78**	**78.75 %**	**94.21 %**	**5.55 h**
OIALR	2.20	70.30 %	88.73 %	**3.26 h**	1.81	77.95 %	93.95 %	5.92 h
Parameters		**16.56 %**				**15.66 %**		

In an attempt to showcase how our method would perform on real-world use cases, our experiments used state-of-the-art techniques and models, including strong image transforms Touvron et al. (2021), dropout Srivastava et al. (2014), learning rate warm-up Gotmare et al. (2019), and cosine learning rate decay Loshchilov & Hutter (2017), the utilized implementations of which are from Wightman et al. (2023). We trained all networks using the AdamW Loshchilov & Hutter (2018) optimizer. The complete sets of hyperparameters are included in Appendix A. We list results as the average of three runs, all of which have unique random seeds.

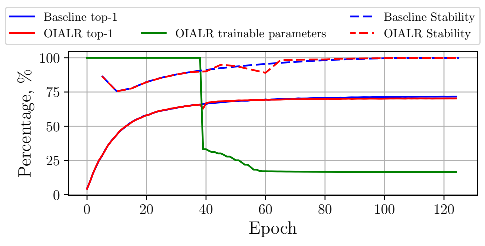
5.1 Computational Environment

We ran all experiments on a distributed-memory, parallel hybrid supercomputer. Each compute node is equipped with two 38-core Intel Xeon Platinum 8368 processors at 2.4 GHz base and 3.4 GHz maximum turbo frequency, 512 GB local memory, a local 960 GB NVMe SSD disk, two network adapters, and four NVIDIA A100-40 GPUs with 40 GB memory connected via NVLink. Inter-node communication uses a low-latency, non-blocking NVIDIA Mellanox InfiniBand 4X HDR interconnect with 200 Gbit/s per port. All experiments used Python 3.10.6 with `CUDA`-enabled `PyTorch` 2.0.0 Paszke et al. (2019). The source code for the implementation is publicly available[1].

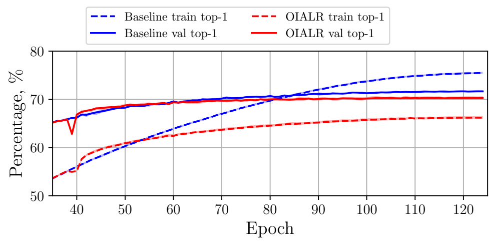
5.2 Vision Transformer on ImageNet-2012

For the first experiment, we trained the Vision Transformer (ViT)-B/16 model Dosovitskiy et al. (2021) in its standard form on the ImageNet-2012 dataset Beyer et al. (2020) using the ReaL validation labels Beyer et al. (2020). The considerable number of parameters in this model provided

[1]Will be released upon publication



(a) Top-1 validation accuracy, percentage of trainable parameters as opposed to the full-rank network, and average stability of the bases.



(b) Top-1 accuracies for training and validation with baseline and OIALR training methods.

Figure 2: Training curves from training a ViT-B/16 network on ImageNet-2012 for 125 epochs.

us with a rigorous testing ground for our OIALR training method. We maintained the same hyper-parameters for both full-rank and adaptive low-rank. To reduce the environmental impact of our experiments, we trained for 125 instead of the original 300 epochs Dosovitskiy et al. (2021). At this point in training, the validation accuracy was observed to be nearly stabilized, see Figure 2b. In this vein, we used an image resolution of 160×160 instead of 224×224 to reduce the training time and energy consumption.

The results are shown in Figure 2 and Table 1. Figure 2a illustrates the top-1 validation score, the number of trainable parameters as a percentage of the full-rank model, and the average network stability (as in Figure 1) throughout the training. We observe that, while the baseline stability increases smoothly throughout the training, OIALR's stability is less consistent as the ranks of the weights are reduced. This is caused by the reduction of ranks themselves as the $\boldsymbol{UV}^T$ from five epochs prior contains more basis vectors than the current $\boldsymbol{UV}^T$. An important observation is the momentary drop in accuracy when the network switches from full-rank to low-rank weights, best seen in Figure 2b, although it quickly rebounds to higher accuracies than before. We theorize that this is caused by the left-over momentum states in the optimizer.

OIALR training reduced the training time by approximately $1\,\%$, though the resulting low-rank model showed $1.34 \pm 0.39\,\%$ worse performance in this naive configuration. The OIALR method decreased the number of trainable parameters to $16.56 \pm 0.23\,\%$ of the full-rank parameters. Figure 2b shows that the full-rank model has moved into the overfitting regime, where the training accuracy continues increasing while the validation accuracy does not, and the low-rank model has not.

5.3 ResNet-RS 101 on ImageNet-2012

To show the versatility of our approach, we trained the ResNet-RS 101 Bello et al. (2021) architecture on ImageNet-2012 for image classification. We trained the model for 125 epochs with an image resolution of 160×160 and validate using a resolution of 224×224 on the ReaL labels Beyer et al. (2020). The results are included in Table 1.

At the end of OIALR training, only $15.66 \pm 0.08\,\%$ of the original full-rank weights remain trainable with only a minor $1.03 \pm 0.16\,\%$ reduction in top-1 accuracy. Due to implementation limitations and the conflicting 2D weight representation and 2D convolution operation, low-rank training was $6.67 \pm 0.49\,\%$ slower than full-rank training.

5.4 OneFormer on CityScapes

In the next experiment, we train the OneFormer Jain et al. (2023) model on the CityScapes Cordts et al. (2016) dataset for semantic segmentation. OneFormer is "a universal image segmentation framework that unifies segmentation with a multi-task train-once design" Jain et al. (2023), i.e. this model trains on multiple tasks at once. We trained the model with the configuration provided by the original authors with only slight modifications (reduced batch size to fit into GPU VRAM and model wrapping for OIALR runs). The results are shown in Table 2.

Table 2: Training the OneFormer on the CityScapes dataset. The rightmost column shows the number of trainable parameters as a percentage of the full-rank model.

	IoU	Category IoU	Time to train	Trainable Parameters
Full rank	**72.47 %**	**88.68 %**	**32.76 h**	100 %
OIALR	68.18 %	87.32 %	32.95 h	**28.56 %**

Table 3: A mini ViT similar to that used in Hassani et al. (2022) trained on CIFAR-10. Baseline indicates the full-rank trainings. 'OAILR, tuned' trainings used tuned hyperparameters, while 'OIALR, untuned' used the same hyperparameters as the baseline. Accuracies and loss values are determined on the test dataset. The rightmost column shows the number of trainable parameters as a percentage of the full-rank model.

	Loss	Top-1	Top-5	Time to train	Trainable Parameters
Full rank	0.88	85.17 %	98.34 %	12.14 min	100 %
OIALR, untuned	0.91	83.05 %	98.38 %	11.99 min	30.98 %
OIALR, tuned	**0.85**	**86.33 %**	**98.53 %**	**11.19 min**	**9.97 %**

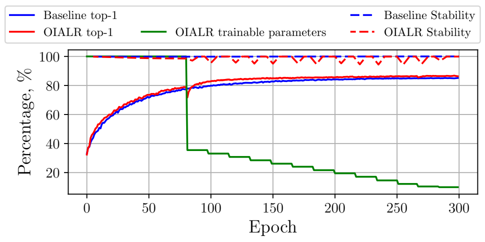
As expected, when using hyperparameters designed for full-rank training, we note a reduction in the class IoU of $4.29 \pm 0.21\,\%$ and a slight reduction in the categorical IoU of $1.02 \pm 0.09\,\%$. While the OIALR model had $28.56 \pm 0.02\,\%$ of the full-rank model's trainable parameters, it required $0.58 \pm 0.02\,\%$ more time to train, this equates to $11.5\,\mathrm{min}$ longer in wallclock time.

5.5 Ablation Study on Mini ViT on CIFAR-10

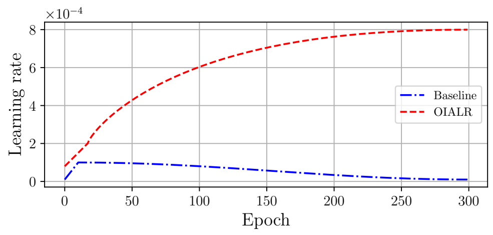
To show how OIALR performs with proper tuning, we trained a reduced-size ViT model on the CIFAR-10 dataset with and without tuning. The runs without tuning utilize the same hyperparameters as the baseline runs. As reduced size ViT models have been shown to perform superbly Hassani et al. (2022) at a fraction of the compute time, we elect to use a ViT-B/16 variant with a patch size of 8, 6 layers, and 6 attention heads in this experiment (original values are a patch size of 16, 12 layers, and 12 attention heads). The results of this experiment are shown in Table 3 and Figure 3.

The top-1 test results are shown in Figure 3a alongside the stability measurements and number of trainable parameters for both the tuned OAILR runs and the baseline runs. We can see here that the baseline stability is nearly constant the entire training, indicating that the network was only tuning the linear mixing of the bases after the fifth epoch. The stability of OIALR runs shows a saw-tooth pattern each time the rank was lowered as the previous bases was composed with basis vectors which had since been removed.

Interestingly, the best learning rate schedule for the OIALR trainings which was found in the hyper-parameter search increases the learning rate as the number of parameters is reduced, see Figure 3b.

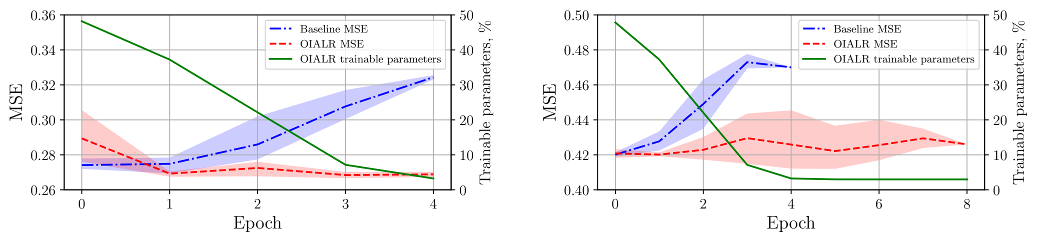


(a) Top-1 test accuracy, trainable parameters as the percentage of the full-rank model, and average stability of the bases with a five epoch frequency.



(b) Learning rate schedules for baseline training and OIALR training. OIALR training learning rate schedule determined by hyperparameter search.

Figure 3: Training curves for a mini ViT on CIFAR-10 for tuned OIALR and baseline trainings.



(a) Prediction length 192

(b) Prediction length 720

Figure 4: MSE and percentage of trainable parameters as opposed to the full-rank network for the Autoformer trained on the ETTm2 dataset using two different prediction lengths in $15\,\mathrm{min}$ time steps.

This result makes intuitive sense: as the number of trainable parameters decreases, the learning rate applied to the gradients of the remaining parameters can be increased without the model degrading.

The tuned OIALR model reduced the number of trainable parameters by $90.03 \pm 0.13\,\%$ while increasing predictive performance over the baseline from $85.17 \pm 0.42\,\%$ to $86.33 \pm 0.71\,\%$. Furthermore, training time was reduced by $8.52 \pm 0.82\,\%$ over the baseline. Although the untuned OIALR model reduced the number of trainable parameters by $69.02 \pm 0.07\,\%$, the top-1 test accuracy drops by over $2\,\%$.

5.6 Ablattion Study on Autoformer on ETTm2

This use case serves as a crucial test for OIALR training as it demonstrates the method's versatility by applying it to a model in a radically different domain. Furthermore, it serves to validate that the findings depicted in Figure 1 remain applicable in non-image scenarios.

The Electricity Transformer Dataset Zhou et al. (2021) (ETT) measures load and oil temperature of electrical transformers. It contains 70,000 measurements, available in different levels of granularity, each with seven different features and is primarily used for time series forecasting. We focus on the ETTm2 dataset, which uses a 15-minute resolution. Common prediction lengths for this dataset are 96, 192, 336, and 720 time steps. The Autoformer Wu et al. (2021) is a well-known transformer model contained in Hugging Face's repository. Unlike the other tested transformers, this model uses auto-correlation layers and one-dimensional convolutions. Due to its success at the time, it was deployed at the 2022 Winter Olympics for weather forecasting.

As the baseline for this experiment suffers quickly from overfitting, see Figure 4, we start in low rank instead of transitioning during training. Although we do see some overfitting in the OIALR results, it is much less severe than in the baseline. As Table 4 indicates, the tuned OIALR models

Table 4: Training of the Autoformer model on the ETTm2 dataset. Baseline and untuned OIALR hyperparameters were chosen as the default parameters in the original source. Tuned OIALR hyperparamters found via `Propulate`. Prediction lengths are in $15\,\mathrm{min}$ time steps. The optimal values for the mean squared error (MSE) and mean absolute error (MAE) are both zero. The last two rows show the number of trainable parameters as a percentage of the full-rank model for untuned and tuned OIALR training respectively.

Prediction length	96		192		336		720	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Base	0.2145	0.2994	0.2737	0.3356	0.3277	0.3640	0.4194	0.4157
OIALR, untuned	0.2140	0.2974	0.2773	0.3336	0.3253	0.3632	0.4213	0.4186
OIALR, tuned	**0.2112**	**0.2942**	**0.2686**	**0.3305**	**0.3212**	**0.3591**	**0.4120**	**0.4147**
Parameters, untuned	49.09 %		43.59 %		45.67 %		51.33 %	
Parameters, tuned	**27.53 %**		**8.85 %**		**4.88 %**		**4.47 %**	

were more accurate for all prediction lengths with a drastically decreased number of parameters, The untuned OIALR models outperformed the baseline in some cases, and succeeded in reducing the number of trainable parameters by $47.42\,\%$ on average. Interestingly, the tuned OIALR model required more trainable parameters for predicting shorter time-spans.

In contrast to the previous experiment, the best learning rate scheduler found for this use case more closely resembles a traditional scheduler, featuring a warm-up phase followed by a gradual decay. This may be related to the fact that the networks, both low-rank and full-rank, overfit the training dataset quickly.

6 CONCLUSION

There has long been curiosity about what is being learned within a neural network during training. This study aimed to shed light on this question by exploring the nature of neural network weights during training through their singular value decomposition. Our findings revealed that the orthogonal component of the a neural network's weights stabilize early in the training process. Building on this discovery, we introduced Orthogonality-Informed Adaptive Low-Rank (OIALR) training.

We used the OIALR training method to train low-rank versions of common and state-of-the-art networks across various data modalities and tasks. While our method may not improve upon traditional training techniques per se, it can outperform them in terms of accuracy and time to train when tuned appropriately. Notably, its true potential lies in significantly reducing the number of trainable parameters, enabling model fine-tuning and production on resource-constrained devices and reducing the amount of data to communicate during distributed training.

Integrating orthogonality-based training methods into the deep learning researcher's toolkit offers promising possibilities for a wide range of applications. With this work, we hope to inspire further exploration and refinement of orthogonality-informed methods, ultimately advancing the field of machine learning and its practicality across diverse domains.

REFERENCES

Mohammad Mahdi Bejani and Mehdi Ghatee. Adaptive Low-Rank Factorization to regularize shallow and deep neural networks, May 2020. URL http://arxiv.org/abs/2005.01995. arXiv:2005.01995 [cs, stat].

Irwan Bello, William Fedus, Xianzhi Du, et al. Revisiting ResNets: Improved Training and Scaling Strategies, March 2021. URL http://arxiv.org/abs/2103.07579. arXiv:2103.07579 [cs].

Liane Bernstein, Alexander Sludds, Ryan Hamerly, et al. Freely scalable and reconfigurable optical hardware for deep learning. *Scientific Reports*, 11(1):3144, February 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-82543-3. URL https://www.nature.com/articles/s41598-021-82543-3. Number: 1 Publisher: Nature Publishing Group.

Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, et al. Are we done with ImageNet?, June 2020. URL http://arxiv.org/abs/2006.07159. arXiv:2006.07159 [cs].

Ali Boufssasse, El houssaine Hssayni, Nour-Eddine Joudar, and Mohamed Ettaouil. A Multi-objective Optimization Model for Redundancy Reduction in Convolutional Neural Networks. *Neural Processing Letters*, March 2023. ISSN 1573-773X. doi: 10.1007/s11063-023-11223-2. URL https://doi.org/10.1007/s11063-023-11223-2.

Samuel Cahyawijaya, Genta Indra Winata, Holy Lovenia, et al. Greenformer: Factorization Toolkit for Efficient Deep Neural Networks, October 2021. URL http://arxiv.org/abs/2109.06762. arXiv:2109.06762 [cs].

Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation, April 2021. URL http://arxiv.org/abs/2104.05247. arXiv:2104.05247 [cs, math].

Carol Chomsky. Stages in Language Development and Reading Exposure. *Harvard Educational Review*, 42(1):1–33, April 1972. ISSN 0017-8055. doi: 10.17763/haer.42.1.h78l676h28331480. URL https://doi.org/10.17763/haer.42.1.h78l676h28331480.

Taco Cohen and Max Welling. Group Equivariant Convolutional Networks. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/cohenc16.html.

Daniel Coquelin, Rocco Sedona, Morris Riedel, and Markus Götz. Evolutionary Optimization of Neural Architectures in Remote Sensing Classification Problems. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pp. 1587–1590, July 2021. doi: 10.1109/IGARSS47720.2021.9554309. ISSN: 2153-7003.

Marius Cordts, Mohamed Omran, Sebastian Ramos, et al. The Cityscapes Dataset for Semantic Urban Scene Understanding. pp. 3213–3223, 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Cordts_The_Cityscapes_Dataset_CVPR_2016_paper.html.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. URL http://arxiv.org/abs/2010.11929. arXiv:2010.11929 [cs] version: 2.

Oscar Fontenla-Romero, Beatriz Pérez-Sánchez, and Bertha Guijarro-Berdiñas. LANN-SVD: A Non-Iterative SVD-Based Learning Algorithm for One-Layer Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3900–3905, August 2018. ISSN 2162-2388. doi: 10.1109/TNNLS.2017.2738118. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A Closer Look at Deep Learning Heuristics: Learning rate restarts, Warmup and Distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r14EOsCqKX.

Anmol Gulati, James Qin, Chung-Cheng Chiu, et al. Conformer: Convolution-augmented Transformer for Speech Recognition, May 2020. URL http://arxiv.org/abs/2005.08100. arXiv:2005.08100 [cs, eess].

Ting Guo, Nurmemet Yolwas, and Wushour Slamu. Efficient Conformer for Agglutinative Language ASR Model Using Low-Rank Approximation and Balanced Softmax. *Applied Sciences*, 13(7):4642, January 2023. ISSN 2076-3417. doi: 10.3390/app13074642. URL https://www.mdpi.com/2076-3417/13/7/4642. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

Ali Hassani, Steven Walton, Nikhil Shah, et al. Escaping the Big Data Paradigm with Compact Transformers, June 2022. URL http://arxiv.org/abs/2104.05704. arXiv:2104.05704 [cs] version: 4.

El houssaine Hssayni, Nour-Eddine Joudar, and Mohamed Ettaouil. KRR-CNN: kernels redundancy reduction in convolutional neural networks. *Neural Computing and Applications*, 34(3):2443–2454, February 2022. ISSN 1433-3058. doi: 10.1007/s00521-021-06540-3. URL https://doi.org/10.1007/s00521-021-06540-3.

Yen-Chang Hsu, Ting Hua, Sungen Chang, et al. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uPv9Y3gmAI5.

Edward J Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jitesh Jain, Jiachen Li, MangTik Chiu, et al. OneFormer: One Transformer to Rule Universal Image Segmentation. 2023.

Ziyu Jiang, Tianlong Chen, Xuxi Chen, et al. DnA: Improving Few-Shot Transfer Learning with Low-Rank Decomposition and Alignment. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, et al. (eds.), *Computer Vision – ECCV 2022*, pp. 239–256, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20044-1.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts, May 2017. URL `http://arxiv.org/abs/1608.03983`. arXiv:1608.03983 [cs, math].

Ilya Loshchilov and Frank Hutter. Fixing Weight Decay Regularization in Adam, 2018. URL `https://openreview.net/forum?id=rk6qdGgCZ`.

Amy Nesky and Quentin F. Stout. Neural networks with block diagonal inner product layers: a look at neural network architecture through the lens of random matrices. *Neural Computing and Applications*, 32(11):6755–6767, June 2020. ISSN 1433-3058. doi: 10.1007/s00521-019-04531-z. URL `https://doi.org/10.1007/s00521-019-04531-z`.

Adam Paszke, Sam Gross, Francisco Massa, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, et al. Stable Low-Rank Tensor Decomposition for Compression of Convolutional Neural Network. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pp. 522–539, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58526-6. doi: 10.1007/978-3-030-58526-6_31.

Daniel Povey, Gaofeng Cheng, Yiming Wang, et al. Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks. In *Interspeech 2018*, pp. 3743–3747. ISCA, September 2018. doi: 10.21437/Interspeech.2018-1417. URL `https://www.isca-speech.org/archive/interspeech_2018/povey18_interspeech.html`.

D.C. Psichogios and L.H. Ungar. SVD-NET: an algorithm that automatically selects network structure. *IEEE Transactions on Neural Networks*, 5(3):513–515, May 1994. ISSN 1941-0093. doi: 10.1109/72.286929. Conference Name: IEEE Transactions on Neural Networks.

Jianfeng Ren and Dong Xia. *Deep Learning Model Optimization*, pp. 183–199. Springer Nature Singapore, Singapore, 2023. ISBN 978-981-99-2897-2. doi: 10.1007/978-981-99-2897-2_8. URL `https://doi.org/10.1007/978-981-99-2897-2_8`.

Olga Russakovsky, Jia Deng, Hao Su, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y. URL `https://doi.org/10.1007/s11263-015-0816-y`.

Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6655–6659, May 2013. doi: 10.1109/ICASSP.2013.6638949. ISSN: 2379-190X.

Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, et al. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. *Advances in Neural Information Processing Systems*, 35:20051–20063, December 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/7e98b00eeafcdaeb0c5661fb9355be3a-Abstract-Conference.html`.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frederic Andres. Sparse low rank factorization for deep neural network compression. *Neurocomputing*, 398:185–196, July 2020. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.02.035. URL https://www.sciencedirect.com/science/article/pii/S0925231220302253.

Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, and Weinan E. Convolutional neural networks with low-rank regularization, February 2016. URL http://arxiv.org/abs/1511.06067. arXiv:1511.06067 [cs, stat].

Oskar Taubert, Marie Weiel, Daniel Coquelin, et al. Massively parallel genetic optimization through asynchronous propagation of populations. In *International Conference on High Performance Computing*, pp. 106–124. Springer, 2023.

Hugo Touvron, Matthieu Cord, Matthijs Douze, et al. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

Roger Waleffe and Theodoros Rekatsinas. Principal Component Networks: Parameter Reduction Early in Training. *CoRR*, abs/2006.13347, 2020. URL https://arxiv.org/abs/2006.13347.

Ross Wightman, Nathan Raw, Alexander Soare, et al. rwightman/pytorch-image-models: v0.8.10dev0 Release, February 2023. URL https://zenodo.org/record/4414861.

Paul Wimmer, Jens Mehnert, and Alexandru Paul Condurache. Dimensionality reduced training by pruning and freezing parts of a deep neural network: a survey. *Artificial Intelligence Review*, May 2023. ISSN 1573-7462. doi: 10.1007/s10462-023-10489-1. URL https://doi.org/10.1007/s10462-023-10489-1.

Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, and Pascale Fung. Lightweight and Efficient End-To-End Speech Recognition Using Low-Rank Transformer. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6144–6148, May 2020. doi: 10.1109/ICASSP40776.2020.9053878. ISSN: 2379-190X.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*, 2021.

Canwen Xu and Julian McAuley. A Survey on Model Compression and Acceleration for Pretrained Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):10566–10575, June 2023. ISSN 2374-3468. doi: 10.1609/aaai.v37i9.26255. URL https://ojs.aaai.org/index.php/AAAI/article/view/26255. Number: 9.

Jian Xue, Jinyu Li, Dong Yu, Mike Seltzer, and Yifan Gong. Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6359–6363, May 2014. doi: 10.1109/ICASSP.2014.6854828. ISSN: 2379-190X.

Zebin Yang, Aijun Zhang, and Agus Sudjianto. Enhancing explainability of neural networks through architecture constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(6): 2610–2621, 2021. doi: 10.1109/TNNLS.2020.3007259.

Borui Zhang, Wenzhao Zheng, Jie Zhou, and Jiwen Lu. Bort: Towards Explainable Neural Networks with Bounded Orthogonal Constraint. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=My57qBufZWs.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, et al. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting, March 2021. URL http://arxiv.org/abs/2012.07436. arXiv:2012.07436 [cs].