Harnessing Orthogonality to Train Low-Rank Neural Networks

Daniel Coquelin[a,*], **Katharina Flügel**[a], **Marie Weiel**[a], **Nicholas Kiefer**[a], **Charlotte Debus**[a], **Achim Streit**[a] and
Markus Götz[a]

[a]Scientific Computing Center (SCC), Karlsruhe Institute of Technology (KIT),
76344 Eggenstein-Leopoldshafen, Germany
ORCID (Daniel Coquelin): https://orcid.org/0000-0001-8552-5153

Abstract. This study explores the learning dynamics of neural networks by analyzing the singular value decomposition (SVD) of their weights throughout training. Our investigation reveals that an orthogonal basis within each multidimensional weight's SVD representation stabilizes during training. Building upon this, we introduce Orthogonality-Informed Adaptive Low-Rank (OIALR) training, a novel training method exploiting the intrinsic orthogonality of neural networks. OIALR seamlessly integrates into existing training workflows with minimal accuracy loss, as demonstrated by benchmarking on various datasets and well-established network architectures. With appropriate hyperparameter tuning, OIALR can surpass conventional training setups, including those of state-of-the-art models.

1 Introduction

How does a neural network learn? Mathematically, its weights are adjusted iteratively, most commonly using the back-propagated gradients of the loss function to minimize the difference between predicted outputs and actual targets. Yet, we have only a limited understanding of what characteristics are being learned. In this work, we show that this optimization process imposes a structure on the network and present a way to exploit it. State-of-the-art neural networks have long existed at the edge of what is computationally possible. Hence, there has long been a desire to reduce the size of networks by sparsification or compression. It has been repeatedly shown that both of these methods can be greatly effective given proper training procedures.

As most tensors can be expressed as low-rank approximations, these methods are often used to compress neural networks [8]. A low-rank approximation factorizes a full-rank matrix M into two or more matrices where the inner dimension r is smaller than the original matrix's dimensions. Formally, the factorization is defined as $M_{m \times n} = A_{m \times r} B_{r \times n}$ with $r < \min(m, n)$. Low-rank network representations can have a variety of uses, from reducing the computational complexity of training [34] or validation [35], to fine-tuning large language models in a resource-efficient manner, e.g., LoRA [18].

Singular value decomposition (SVD) is a popular method for finding a, theoretically exact, low-rank representation of a given matrix. SVD factorizes a matrix into an orthogonal basis U, an orthogonal

cobasis V, and a diagonal matrix of the singular values sorted in descending order Σ, as $M_{m \times n} = U_{m \times r} \Sigma_{r \times r} V_{r \times n}^T$. To produce a low-rank approximation, singular values and their corresponding basis vectors can be removed. However, as larger singular values are removed, the approximation quality diminishes.

The power of SVD lies in its ability to identify the principal components of a matrix, i.e., the directions along which it varies most. These principal components are captured by the orthogonal bases U and V. We posit that these components are learned in the initial phases of training, which allows for network compression in later training stages. In this work, we

- show evidence that the orthogonal basis UV^T of each of a network's multidimensional weights stabilizes during training;
- propose Orthogonality-Informed Adaptive Low-Rank (OIALR) neural network training, a novel training method harnessing this finding;
- demonstrate that OIALR seamlessly integrates into existing training workflows with minimal accuracy loss by means of benchmarks on multiple datasets, data modalities, well-known and state-of-the-art network architectures, and training tasks;
- show that OIALR can outperform conventional full-rank and other low-rank training methods.

For simplicity, we will refer to the orthogonal basis UV^T of each of a network's multidimensional weights as the network's orthogonal bases. We provide public implementations of the most common layer types, a method to wrap arbitrary model architectures for any learning task, and the OIALR training method [1]

2 Related work

Several methods exist for obtaining a reduced-size network, including pruning, which involves the removal of unimportant weights in a model; compression, which reduces the size of weights via low-rank approximations; quantization, which reduces the number of bits used to store the weights; and sparse training, where the network is trained to be sparse from initialization [43].

Convolution layers famously contain kernel [16], channel [14], and filter [35] redundancies which can be pruned. These methods are referred to as structured pruning methods. Although convolution

---

* Corresponding Author. Email: daniel.coquelin@kit.edu.

[1] https://github.com/Helmholtz-AI-Energy/oialr

layers are the use case of these methods, other methods have been found for different layer types [39]. In contrast, unstructured pruning methods prune individual weights across all layer types [23].

Low-rank compression methods often employ SVD. To the best of our knowledge, the earliest application of SVD in neural network compression is the SVD-NET [32]. This method decomposes the weights of a simple neural network with SVD and trains the U, Σ, and V matrices. This approach has been successful and has been shown to improve generalization, i.e. reducing overfitting during training [41, 30, 4].

Given the abundance of pre-trained models today, there is a strong inclination to apply a low-rank approximation after training. Although this often leads to significant performance degradation [44], fine-tuning pre-trained models for specific use cases using low-rank approximations has proven to be effective [27]. For instance, LoRA [18] demonstrated that large language models (LLMs) can be adapted to specialized use cases by adding a low-rank weight component alongside the originally trained weights.

The most common approach for training low-rank models from scratch is to train all matrices' low-rank approximations [5, 17, 12]. As starting the training in low rank can be detrimental to network accuracy [38, 1], many methods transition to a low-rank representation later or slowly reduce the inner-rank during training [43].

A matrix's orthogonal basis can be viewed as the coordinate system that the matrix operates in, making it extremely useful for explaining the inner working of modern black-box neural networks. Intuitively, orthogonal networks are beneficial from an explainability viewpoint. ExNN [45] utilized methods to preserve projection orthogonality during training to improve interpretability. Similarly, the Bort optimizer [46] aims to improve model explainability using boundedness and orthogonality constraints.

While many low-rank approximations utilize orthogonality, low-rank training methods frequently do not maintain it. Methods that respect orthogonality, show increased network performance [31, 34]. In DLRT [34], each of the three matrices in a model's singular-value decomposed weights is trained in a separate forward-backward pass while maintaining orthogonality.

3 Observing orthogonality in neural network training

The work of Schotthöfer et al. [34] formulates the training of a low-rank neural network as a continuous-time gradient flow. They use low-rank numerical integrators for matrix Ordinary Differential Equations (ODEs) to create a training process which uses three forward and backward passes for each batch to train a low-rank neural network. This formulation of training with ODEs is quite appealing as it can accurately show the impact factorizations have on the gradients of the parameters that they represent.

We begin from the same formulation but significantly simplify the low-rank training process by showing that a network parameter's orthogonal component stabilizes during the training. Let us consider a single weight matrix of network layer, k, of shape $m \times n$ at step t of training, $W_k(t) \in \mathcal{M}_{r_k}$ where $\mathcal{M}_{r_k}$ is the manifold of matrices with rank r_k. The other weights are fixed in time and treated as constants for gradient calculations. From Schotthöfer et al. [34], the training can then be formulated as a continuous process:

$$\min\{\|\dot{W}_k(t) + \nabla_{W_k}\mathcal{L}(W_k(t))\| : \dot{W}_k(t) \in \mathcal{T}_{W_k(t)}\mathcal{M}_{r_k}\} \quad (1)$$

where $\mathcal{L}$ is the loss function, $\mathcal{T}_{W_k(t)}\mathcal{M}_{r_k}$ is the tangent space of $\mathcal{M}_{r_k}$ at position $W_k(t)$, and $\dot{W}_k(t)$ denotes the temporal deriva-

tive. This can be expressed as a Galerkin condition:

$$\langle \dot{W}_k(t) + \nabla_{W_k}\mathcal{L}(W_k(t)), \delta W_k \rangle = 0 \quad \forall \delta W_k \in \mathcal{T}_{W_k(t)}\mathcal{M}_{r_k} \quad (2)$$

where δW_k is an element of the tangent space $\mathcal{T}_{W_k(t)}\mathcal{M}_{r_k}$.

Using the SVD decomposition, $W_k = U_k \Sigma_k V_k^\top$, this element can be described as:

$$\delta W_k = \delta U_k \Sigma_k V_k^\top + U_k \delta \Sigma_k V_k^\top + U_k \Sigma_k \delta V_k^\top \quad (3)$$

where δU_k and δV_k are elements of the tangent space of the Stiefel manifold with r_k orthonormal columns at the points U_k and V_k, and Σ_k is a matrix of shape $r_k \times r_k$.

Lets consider the QR decomposition of W_k. A small change to W_k can result in changes to the orthogonal basis Q_k, the mixing matrix R_k, or both. However, given that SGD makes small iterative steps, we postulate that large-scale changes to the basis happen predominately at low step counts. In short, we hypothesise that the orthogonal component of W_k stabilizes during the early stages of training.

Most of a network's weights can be represented by tall-and-skinny matrices, i.e. $m > n$. For these weights, the full Q matrix is of size $\mathcal{O}(m^2)$, making it memory inefficient to track Q over multiple training steps. To test our hypothesis, we use the semi-orthogonal bases $U_k V_k^\top$ as determined by the compact SVD of W_k as the orthogonal component. This semi-orthogonal matrix is more memory efficient, $\mathcal{O}(mn)$, and uniquely determined by $W_k(W_k^T W_k)^{-1/2}$. To track how the orthogonal component of W_k changes throughout training we define the *Stability* between timesteps i and j as

$$S_{k,ij} = \frac{\mathrm{tr}\left((U_k V_k^\top)_i (V_k U_k^\top)_j\right)}{m} \quad (4)$$

where tr is the trace and the product $(U_k V_k^\top)_i$ is orthogonal component of W_k at timestep i. This metric ranges from zero to one, where a *Stability* of zero indicates two completely different bases and a *Stability* of one indicates two identical bases. To track the mixing matrix, R_k, we use a form of Euclidean similarity

$$D_{k,ij} = 1 - \sqrt{\frac{(R_{k,i} - R_{k,j})^2}{mn}} \quad (5)$$

where $R_{k,i}$ and $R_{k,i}$ are the R matrices for W_k at two different timsteps i and j. This similarity metric is chosen to maintain the same scale and behavior as *Stability*.

Figure 1 shows how both the mixing matrix and the orthogonal component of the weights evolve during the training of two vastly different network architectures, ResNet-RS 101 [2] and the Vision-Transformer (ViT) B/16 [9], on ImageNet-2012 [33]. In both cases (Figures 1a and 1b) the *Stability* decreases in the first ten epochs, i.e. the parameters' basis vectors are changing, as the networks move away from their random initialization. Then, the *Stability* converges towards its maximum value of one, indicating that the current basis and the previous iteration are becoming more aligned. This is in sharp contrast to the similarity plots (Figures 1c and 1d), where the largest changes to the mixing occur towards the middle of training, while at the beginning and end of training the changes to the mixing are smaller. This finding aligns itself with fact that gradients converge in direction during training [20].

Using the information from Figure 1, we can assume that in the later stages of training δU_k and δV_k tend to zero; and Equation (3) reduces to:

$$\delta W_k = U_k \delta \Sigma_k V_k^\top \quad (6)$$

(a) ResNet-RS 101, *Stability*, see Equation (4)

(b) Vision Transformer B/16, *Stability*, see Equation (4)

(c) ResNet-RS 101, Euclidean similarity, see Equation (5)

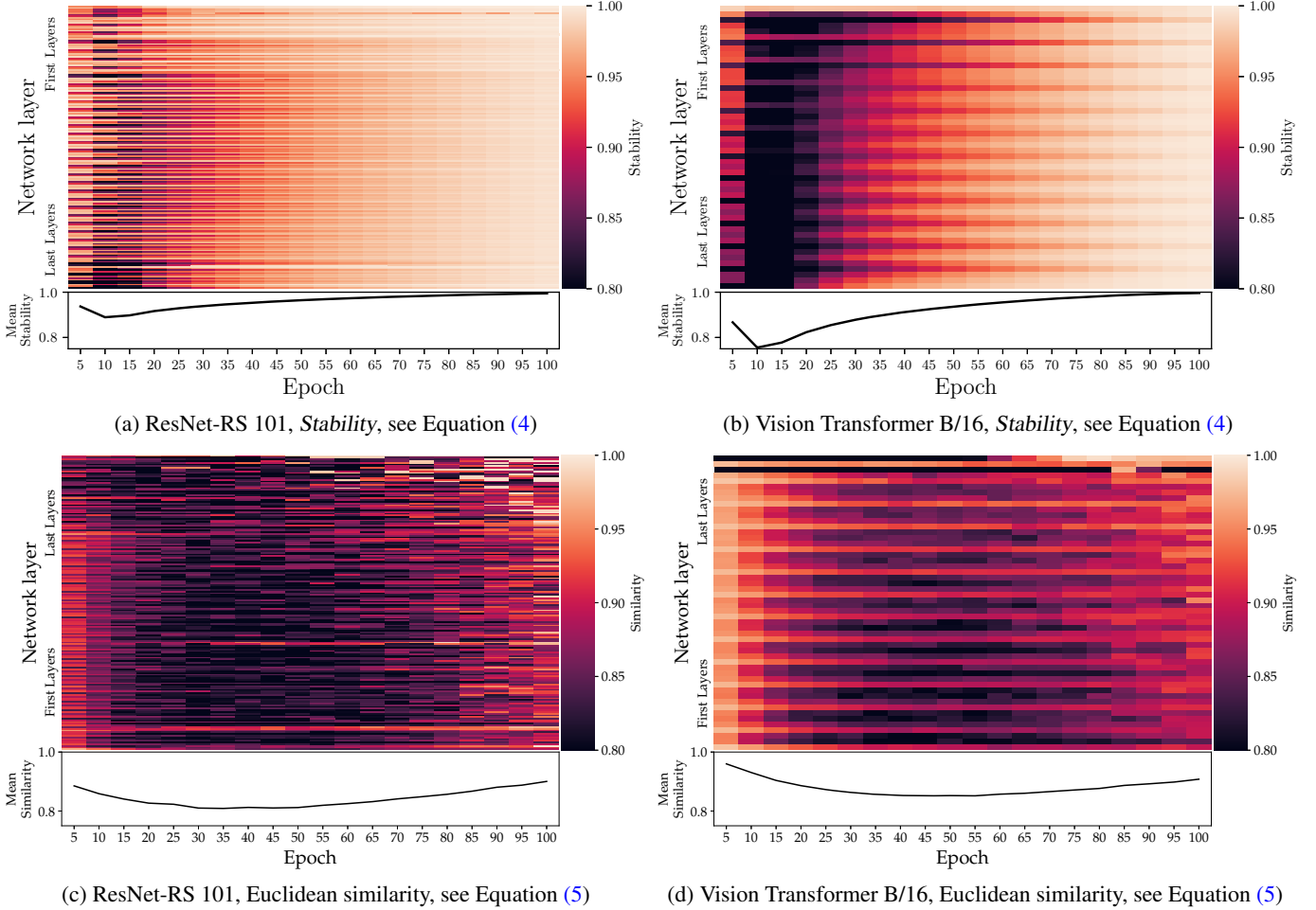(d) Vision Transformer B/16, Euclidean similarity, see Equation (5)

Figure 1: Analysis of the linear mixing Euclidean similarity and orthogonal basis *Stability* for ResNet and ViT models during ImageNet-2012 training. *Stability* is defined by Equation (4) and Euclidean similarity is defined by Equation (5), higher denotes less changes between steps for both. Both metrics compare the network's current parameters with those of five epochs prior. The x-axis denotes the training epoch, and the y-axis denotes the network layer (input layers at the top). Mean stability and similarity are shown below each heatmap.

and the time derivative of $\boldsymbol{W}_k$ becomes

$$\dot{\boldsymbol{W}}_k = \frac{d}{dt}\left\{\boldsymbol{U}_k\boldsymbol{\Sigma}\boldsymbol{V}_k^{\top}\right\} = \boldsymbol{U}_k\dot{\boldsymbol{\Sigma}}_k\boldsymbol{V}_k^{\top}. \qquad (7)$$

Thus, later in training, the Galerkin condition, Equation (2), is now

$$\langle\boldsymbol{U}_k\dot{\boldsymbol{\Sigma}}_k\boldsymbol{V}_k^{\top} + \nabla_{\boldsymbol{W}_k}\mathcal{L}(\boldsymbol{W}_k(t)), \delta\boldsymbol{W}_k\rangle = 0 \quad \forall\delta\boldsymbol{W}_k \in \mathcal{T}_{\boldsymbol{W}_k(t)}\mathcal{M}_{r_k} \quad (8)$$

and as we can treat $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ as constants,

$$\langle\dot{\boldsymbol{\Sigma}}_k + \boldsymbol{U}_k^{\top}\nabla_{\boldsymbol{W}_k}\mathcal{L}(\boldsymbol{W}_k(t))\boldsymbol{V}_k, \delta\boldsymbol{\Sigma}_k\rangle = 0 \qquad (9)$$

finally, we arrive at

$$\dot{\boldsymbol{\Sigma}}_k = -\boldsymbol{U}_k^{\top}\nabla_{\boldsymbol{W}_k}\mathcal{L}(\boldsymbol{W}_k(t))\boldsymbol{V}_k. \qquad (10)$$

This allows for a massive simplification of low-rank training: only training on the low-rank matrix $\boldsymbol{\Sigma}_k$ while using low-rank numerical integrators for matrix ODEs as shown by Schotthöfer et al. [34].

4 Orthogonality-Informed Adaptive Low-Rank Training

To harness the stabilization of the orthogonal bases in neural network training, we present a novel algorithm that reduces the number

of trainable parameters while maintaining both accuracy and overall time-to-train, unlike most previous methods which focused on either one or the other [43].

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 number of iterations d, a hyperparameter of the algorithm, we transition the network's multidimensional weights to their $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ representation via their SVD. If a weight has more than two dimensions, we maintain the first dimension and flatten those trailing. After this, if the second dimension is larger than the first, we transpose it before finding its SVD. Experimentally, we found that the delay should be one third of the total number of iterations. At this point, we no longer train $\boldsymbol{U}$ and $\boldsymbol{V}^T$ with backpropagation but *train only the square matrix* $\boldsymbol{\Sigma}$. After a specified number of 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 the 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 generally unstable for longer, the update of $\boldsymbol{U}$ and $\boldsymbol{V}$ is only applied to the network's last $\ell = L \cdot \alpha \cdot u$ layers, where L is the number of network layers, α is a hyperparameter de-

---

Algorithm 1 Updating a weight matrix's basis and cobasis, U and V, from its actively trained singular value matrix Σ. U, Σ, and V comprise the current $U\Sigma V^T$ representation of a weight matrix.

Inputs: Frozen bases U and V, trained Σ matrix
1: $U', \Sigma', V'^T \leftarrow \text{SVD}(\Sigma)$
2: $U \leftarrow UU', V^T \leftarrow V'^T V^T, \Sigma \leftarrow \Sigma'$

Algorithm 2 OIALR training method

Inputs: Model M, training steps $t_{\max}$, delay steps d, low-rank update frequency ν, singular value cutoff fraction β, percentage of layers in each low-rank update step size α
Parameter: $L \leftarrow$ Number of possible low-rank weights in M
Parameter: $\ell \leftarrow L \cdot \alpha$
1: **for** $t \leftarrow 1$ **to** $t_{\max}$ **do**
2:   **if** $t < d$ **then**
3:     Train full-rank network.
4:   **else if** $t = d$ **then**
5:     Convert network to $U\Sigma V^T$ representation.
6:   **else if** $t \mod \nu = 0$ **then**
7:     **for** $i \leftarrow L - \ell$ **to** L **do**
8:       Update $U\Sigma V^T$ rep. i with Algorithm 1.
9:       Remove singular values $< \beta \cdot \max(\Sigma_i)$
10:       Reshape U_i, V_i, Σ_i, and optimizer states.
11:     $\ell \leftarrow \ell + L \cdot \alpha$
12:   **else**
13:     Train network's $U\Sigma V^T$ representations (Σ).

---

faulting to 0.1, and u is the number of already completed updates. 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.

The first transition to the $U\Sigma V^T$ representation will almost certainly utilize more memory than in its traditional state. For example, a traditional weight matrix has $\mathcal{O}(mn)$ elements while the $U\Sigma V^T$ representation in OIALR has $\mathcal{O}(r(m + r + n))$ elements, where r is the inner rank. As training progresses, the number of 'useful' basis vectors is expected to decrease, resulting in a reduction in the network's size, given that the network is trained appropriately.

5 Experiments

To evaluate the effectiveness of our OIALR training approach, we conducted extensive experiments using different neural network architectures and datasets. Our primary focus was to demonstrate its effectiveness in terms of reducing the number of trainable parameters while maintaining or enhancing network performance and training time.

In our first experiment, we aim to understand what a typical researcher would experience by applying OIALR directly to a conventional and well-known neural network setup for a computer vision problem (see Section 5.2). Next, we compare OIALR to other popular low-rank and sparse methods (see Section 5.3). To see how OIALR performs on a real-world application, we investigate its performance in time-series forecasting with the Autoformer [42], which was deployed at the 2022 Winter Olympics.

Up to this point, the hyperparameters (HPs) of the training methods are the same for both the baseline and OIALR methods. Given that OIALR dynamically alters the network structure during training, we expect the optimal HPs for OIALR training to vary from those used for full-rank training. For the final two experiments in Sections 5.4 and 5.5, we also determine more well-suited HPs for OIALR using Propulate [36], an asynchronous evolutionary optimization package shown to be effective for neural architecture searches [6]. We report 'Compression' and 'Trainable parameters' as percentages relative to the conventional model. For example, a compression of 80% signifies 20% fewer total parameters than the traditional model. Non-trainable parameters for OIALR-trained models are dominated by the U and V bases, while traditional models tend to have fewer non-trainable parameters like the running average in a batch normalization layer.

To demonstrate how our method would perform on real-world use cases, our experiments used state-of-the-art techniques and models, including strong image transforms [37], dropout [28], learning rate warm-up [11], and cosine learning rate decay [25], implemented as per [40]. All networks were trained using the AdamW [21] optimizer. Complete sets of HPs are included in the supplementary material [7]. Results represent the average of three runs with distinct 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 Infini-Band 4X HDR interconnect with 200 Gbit/s per port. All experiments used Python 3.10.6 with CUDA-enabled PyTorch 2.0.0 [29].

5.2 Vision Transformer on ImageNet-2012

For the first experiment, we trained the Vision Transformer (ViT)-B/16 model [9] on the ImageNet-2012 dataset [3] using the ReaL validation labels [3]. The considerable parameter count of this model provided a rigorous test for the OIALR training method. We maintained identical HPs for both full-rank and adaptive low-rank training. To reduce the environmental impact, we trained for 125 epochs instead of the original 300 [9]. By this point, validation accuracy had nearly stabilized, as shown in Figure 2b. Additionally, we used an image resolution of 160×160 instead of 224×224 to further reduce the energy consumption.

The results are shown in Figure 2 and Table 1. Figure 2a presents the top-1 validation score, the percentage of trainable parameters relative to the full-rank model, and the average network *Stability* (as shown in Figure 1) throughout training. Notably, the baseline *Stability* increases smoothly throughout training, while OIALR's *Stability* is less consistent due to the reductions in the weights' ranks. This arises from the fact that the UV^T from five epochs prior contains more basis vectors than the current UV^T.

As evident in Figure 2b, there is a momentary accuracy drop when the network transitions from full-rank to its $U\Sigma V^T$ representation, but it swiftly rebounds, surpassing previous performance. We theorize that this is caused by the residual momentum states in the optimizer 'pushing' the network in different directions.

In this untuned case, OIALR training required approximately the same amount of time to train (1 % longer) while the models maintained similar performance ($1.34 \pm 0.39\,\%$ decrease in top-1 accuracy) than traditional full-rank training. OIALR reduced the number

Table 1: Training ViT-B/16 on ImageNet-2012 with and without OIALR. Hyperparameters are identical in both cases.

	Training method	Loss	Top-1 acc.	Top-5 acc.	Time to train	Compression	Trainable parameters
ViT-B/16	Baseline	**2.16**	**71.64 %**	**89.18 %**	3.29 h	—	—
	OIALR	2.20	70.30 %	88.73 %	**3.26 h**	98.95 %	16.56 %
ResNet-RS 101	Baseline	**1.78**	**78.75 %**	**94.21 %**	**5.55 h**	—	—
	OIALR	1.81	77.95 %	93.95 %	5.92 h	104.66 %	**15.66 %**
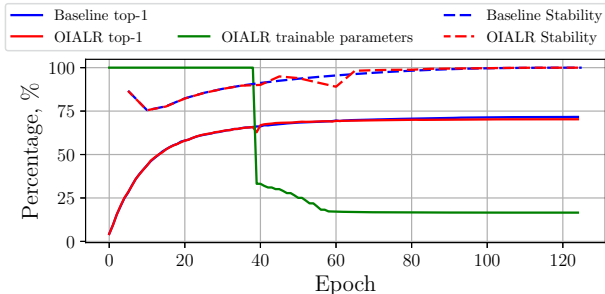
(a) Top-1 validation accuracy, percentage of trainable parameters as compared to the traditional network, and average *Stability* measured with a five-epoch frequency.
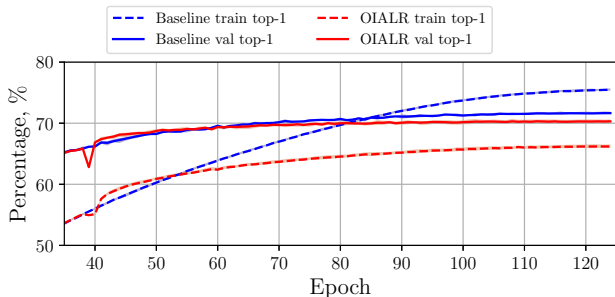
(b) Top-1 accuracies for training and validation with baseline and OIALR training methods.

Figure 2: Training of a ViT-B/16 network on ImageNet-2012 over 125 epochs.

of trainable parameters to $16.56 \pm 0.23 \%$ of the full-rank parameters. Figure 2b shows that the full-rank model has entered the overfitting regime, where training accuracy continues increasing while validation accuracy plateaus, whereas the low-rank model has not.

## 5.3 Comparison with related low-rank and sparse training methods

To show where OIALR fits into the landscape of full-to-low-rank, low-rank, and sparse training methods, we performed a comparative analysis shown in Table 2. In these experiments, the baseline and compression methods use the same HPs.

OIALR and DLRT [34] are SVD-based low-rank factorization methods. LRNN [19] uses a traditional two-matrix representation ($\boldsymbol{W} \approx \boldsymbol{AB}$). CP [14], SFP [15], PP-1 [35], and ThiNet [26] are structured pruning methods which use channel or filter pruning for convolution layers. GAL [23] and RNP [22] are unstructured pruning methods; RigL [10] is a sparse training method.

Overall, OIALR demonstrates competitive performance across different architectures and datasets. Despite producing subpar results on ResNet-50, it showed a marginal accuracy improvement over the baseline for VGG16 [24] on CIFAR-10. We theorize that since

Table 2: Comparison of OIALR with various compression methods. 'Diff. to baseline' refers to the difference in top-1 validation (ImageNet-2012) or test (CIFAR-10) accuracy between the baseline and the listed methods. Positive values indicate that the listed method outperforms the traditionally trained network. Absence of data indicated by '—'. For non-OIALR results see Schotthöfer et al. [34], Evci et al. [10].

	Training method	Diff. to baseline	Compression	Trainable parameters
ResNet-50 ImageNet-2012	OIALR	-1.72 %	82.55 %	15.15 %
	DLRT	-0.56 %	54.10 %	**14.20 %**
	PP-1	**-0.20 %**	44.20 %	—
	CP	-1.40 %	50.00 %	—
	SFP	**-0.20 %**	41.80 %	—
	ThiNet	-1.50 %	36.90 %	—
	RigL	-2.20 %	**20.00 %**	—
VGG16 ImageNet-2012	OIALR	-1.53 %	**36.52 %**	**4.77 %**
	DLRT	-2.19 %	86.00 %	78.40 %
	PP-1	**-0.19 %**	80.20 %	—
	CP	-1.80 %	80.00 %	—
	ThiNet	-0.47 %	69.04 %	—
	RNP(3X)	-2.43 %	66.67 %	—
VGG16 CIFAR-10	OIALR	**0.10 %**	**27.05 %**	**13.88 %**
	DLRT	-1.89 %	56.00 %	77.50 %
	GAL	-1.87 %	77.00 %	—
	LRNN	-1.90 %	60.00 %	—
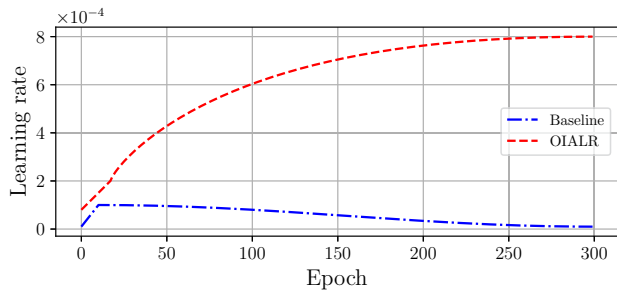
Figure 3: Learning rate schedules for baseline and OIALR training for a mini ViT on CIFAR-10. OIALR training learning rate schedule determined by HP search.

VGG16 is known to be over-parameterized, there are more basis vectors that can be removed. By eliminating these less useful basis vectors, the network can focus on those remaining to train a performant low-rank network.

## 5.4 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 use the same HPs as the baseline runs. As reduced-size ViT models have been shown to perform superbly [13] at a fraction of the compute time, we elect to use a ViT-B/16 variant with a patch size of eight, six layers, and six attention heads in this experiment (original values are a patch size of 16, 12

Table 3: A mini ViT trained on CIFAR-10. 'OIALR, tuned' training runs used tuned HPs, while 'OIALR' used the same HPs as the baseline. Accuracies and loss values are determined on the test dataset.

Training method	Loss	Top-1 accuracy	Top-5 accuracy	Time to train	Compression	Trainable parameters
Traditional	0.88	85.17 %	98.34 %	12.14 min	—	—
OIALR	0.91	83.05 %	98.38 %	11.99 min	146.78 %	30.98 %
OIALR, tuned	**0.85**	**86.33 %**	**98.53 %**	**11.19** min	**55.06 %**	**9.97 %**
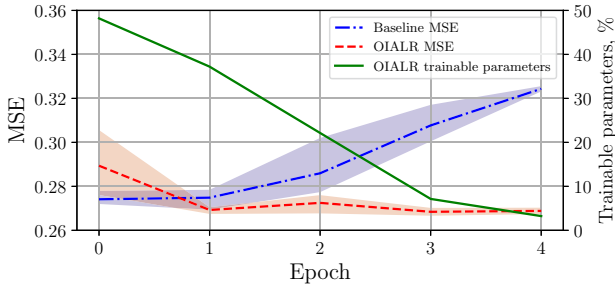
(a) Prediction length 192
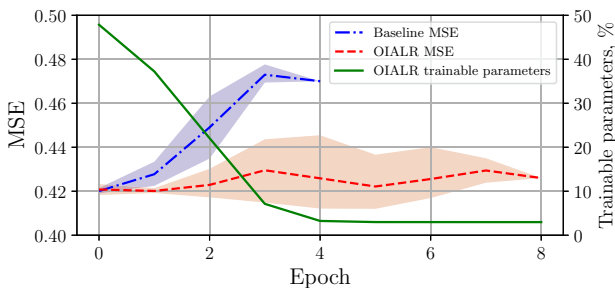
(b) Prediction length 720

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

layers, and 12 attention heads). The results of this experiment are shown in Table 3 and Figure 3.

Interestingly, the best learning rate schedule for the OIALR training method discovered through HP search increases the learning rate as the number of parameters decreases, see Figure 3. 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.

Although the untuned OIALR model reduced the trainable parameters by $69.02\pm0.07\,\%$, the top-1 test accuracy dropped by over $2\,\%$. In contrast, 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.

5.5 Ablation study on Autoformer on ETTm2

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

The Electricity Transformer Dataset [47] (ETT) measures load and oil temperature of electrical transformers. It contains 70,000 measurements at various levels of granularity, each with seven features, and is primarily used for time series forecasting. We focus on the

ETTm2 dataset, which has a 15-minute resolution. Common prediction lengths for this dataset are 96, 192, 336, and 720 time steps. The Autoformer [42], a well-known transformer model in Hugging Face's repository, differs from the other tested transformers by using auto-correlation layers and one-dimensional convolutions. Due to its success, it was deployed at the 2022 Winter Olympics for weather forecasting.

Given the baseline's susceptibility to overfitting (see Figure 4), we initiate training in low rank instead of transitioning during training. Although some overfitting is observed in the OIALR results, it is considerably less severe than in the baseline. As Table 4 indicates, the tuned OIALR models were more accurate across 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 to $45.72\,\%$ on average. As explained in Section 4, these models required more parameters than the baseline model due to the shapes of the model's $U\Sigma V^T$ representation. The tuned OIALR measurements generally showed a much more successful compression percentage. 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 how a neural network learns. 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 a neural network's weights stabilizes early in the training process. Building on this discovery, we introduced Orthogonality-Informed Adaptive Low-Rank (OIALR) training.

We evaluated OIALR by training low-rank versions of widely used and state-of-the-art neural networks across diverse data modalities and tasks. OIALR demonstrated superior performance when compared against other low-rank methods, even when employing the default hyperparameter settings used in traditional model training. While our approach may not directly surpass traditional training techniques in all scenarios, it can outperform them in terms of both accuracy and training time when tuned appropriately.

OIALR's true strength lies in substantially reducing the number of trainable parameters of the final model, thereby facilitating model fine-tuning, transfer learning, and deployment on resource-constrained devices. This reduction also contributes to reducing the data transfer requirements during distributed training, reducing the gap between expensive, top-tier clusters and more affordable options.

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, ul-

Table 4: Training of the Autoformer model on the ETTm2 dataset. Baseline and untuned OIALR HPs were the default parameters from Wu et al. [42]. Tuned OIALR HPs were found via `Propulate`. Prediction lengths (PL) in the leftmost column are in 15 min time steps. The optimal value for mean squared error (MSE) and mean absolute error (MAE) is zero.

PL	Training method	MSE	MAE	Compression	Trainable parameters
96	Baseline	0.2145	0.2994	—	—
	OIALR	0.2140	0.2974	182.06 %	46.16 %
	OIALR, tuned	**0.2112**	**0.2942**	**47.84 %**	**12.19** %
192	Baseline	0.2737	0.3356	—	—
	OIALR	0.2773	0.3336	163.35 %	105.31 %
	OIALR, tuned	**0.2686**	**0.3305**	105.31 %	**27.15 %**
336	Baseline	0.3277	0.3640	—	—
	OIALR	0.3253	0.3863	179.87 %	45.67 %
	OIALR, tuned	**0.3212**	**0.3591**	**27.30 %**	**7.14 %**
720	Baseline	0.4194	0.4157	—	—
	OIALR	0.4213	0.4186	194.13 %	51.33 %
	OIALR, tuned	**0.4120**	**0.4147**	**13.55 %**	**4.46 %**

timately advancing the field of machine learning and its practicality across diverse domains.

Acknowledgements

References

[1] M. M. Bejani and M. 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].

[2] I. Bello, W. Fedus, X. Du, et al. Revisiting ResNets: Improved Training and Scaling Strategies, Mar. 2021. URL http://arxiv.org/abs/2103.07579. arXiv:2103.07579 [cs].

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

[4] S. Cahyawijaya, G. I. Winata, H. Lovenia, et al. Greenformer: Factorization Toolkit for Efficient Deep Neural Networks, Oct. 2021. URL http://arxiv.org/abs/2109.06762. arXiv:2109.06762 [cs].

[5] G. Ceruti, J. Kusch, and C. Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation, Apr. 2021. URL http://arxiv.org/abs/2104.05247. arXiv:2104.05247 [cs, math].

[6] D. Coquelin, R. Sedona, M. Riedel, and M. Götz. Evolutionary Optimization of Neural Architectures in Remote Sensing Classification Problems. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 1587–1590, July 2021. doi: 10.1109/IGARSS47720.2021.9554309. ISSN: 2153-7003.

[7] D. Coquelin, K. Flügel, M. Weiel, N. Kiefer, C. Debus, A. Streit, and M. Götz. Harnessing orthogonality to train low-rank neural networks, 2024. URL https://arxiv.org/abs/2401.08505.

[8] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. *Proceedings of the IEEE*, 108(4):485–532, Apr. 2020. ISSN 1558-2256. doi: 10.1109/JPROC.2020.2976475. Conference Name: Proceedings of the IEEE.

[9] A. Dosovitskiy, L. Beyer, A. 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.

[10] U. Evci, T. Gale, J. Menick, et al. Rigging the Lottery: Making All Tickets Winners. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[11] A. Gotmare, N. S. Keskar, C. Xiong, and R. 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.

[12] T. Guo, N. Yolwas, and W. Slamu. Efficient Conformer for Agglutinative Language ASR Model Using Low-Rank Approximation and Balanced Softmax. *Applied Sciences*, 13(7):4642, Jan. 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.

[13] A. Hassani, S. Walton, N. 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.

[14] Y. He, X. Zhang, and J. Sun. Channel Pruning for Accelerating Very Deep Neural Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1398–1406, 2017. doi: 10.1109/ICCV.2017.155.

[15] Y. He, G. Kang, X. Dong, et al. Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2234–2240. AAAI Press, 2018. ISBN 9780999241127.

[16] E. h. Hssayni, N.-E. Joudar, and M. Ettaouil. KRR-CNN: kernels redundancy reduction in convolutional neural networks. *Neural Computing and Applications*, 34(3):2443–2454, Feb. 2022. ISSN 1433-3058. doi: 10.1007/s00521-021-06540-3. URL https://doi.org/10.1007/s00521-021-06540-3.

[17] Y.-C. Hsu, T. Hua, S. 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.

[18] E. J. Hu, Y. Shen, P. 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.

[19] Y. Idelbayev and M. A. Carreira-Perpinan. Low-Rank Compression of Neural Nets: Learning the Rank of Each Layer. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8046–8056, June 2020. doi: 10.1109/CVPR42600.2020.00807. ISSN: 2575-7075.

[20] Z. Ji and M. Telgarsky. Directional convergence and alignment in deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17176–17186. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c76e4b2fa54f8506719a5c0dc14c2eb9-Paper.pdf.

[21] I. L. and F. H. Fixing Weight Decay Regularization in Adam, 2018. URL https://openreview.net/forum?id=rk6qdGgCZ.

[22] J. Lin, Y. Rao, J. Lu, and J. Zhou. Runtime Neural Pruning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[23] S. Lin, R. Ji, C. Yan, et al. Towards Optimal Structured CNN Pruning via Generative Adversarial Learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2785–2794, 2019. doi: 10.1109/CVPR.2019.00290.

[24] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015. doi: 10.1109/ACPR.2015.7486599.

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

[26] J.-H. Luo, J. Wu, and W. Lin. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5068–5076, 2017. doi: 10.1109/ICCV.2017.541.

[27] R. K. Mahabadi, J. Henderson, and S. Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=bqGK5PyI6-N.

[28] S. Nitish, G. Hinton, A. 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.

[29] A. Paszke, S. Gross, F. 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.

[30] A.-H. Phan, K. Sobolev, K. Sozykin, et al. Stable Low-Rank Tensor Decomposition for Compression of Convolutional Neural Network. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 522–539, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58526-6. doi: 10.1007/978-3-030-58526-6_31.

[31] D. Povey, G. Cheng, Y. Wang, et al. Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks. In *Interspeech 2018*, pages 3743–3747. ISCA, Sept. 2018. doi: 10.21437/Interspeech.2018-1417. URL https://www.isca-speech.org/archive/interspeech_2018/povey18_interspeech.html.

[32] D. Psichogios and L. 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.

[33] O. Russakovsky, J. Deng, H. Su, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec. 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

[34] S. Schotthöfer, E. Zangrando, J. 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, Dec. 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/7e98b00eeafcdaeb0c5661fb9355be3a-Abstract-Conference.html.

[35] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Play and Prune: Adaptive Filter Pruning for Deep Model Compression. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 3460–3466. AAAI Press, 2019. ISBN 9780999241141.

[36] O. Taubert, M. Weiel, D. Coquelin, et al. Massively parallel genetic optimization through asynchronous propagation of populations. In *International Conference on High Performance Computing*, pages 106–124. Springer, 2023.

[37] H. Touvron, M. Cord, M. Douze, et al. Training data-efficient image transformers & distillation through attention. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

[38] R. Waleffe and T. Rekatsinas. Principal Component Networks: Parameter Reduction Early in Training. *CoRR*, abs/2006.13347, 2020. URL https://arxiv.org/abs/2006.13347.

[39] Z. Wang, J. Wohlwend, and T. Lei. Structured pruning of large language models. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.496. URL https://aclanthology.org/2020.emnlp-main.496.

[40] R. Wightman, N. Raw, A. Soare, et al. rwightman/pytorch-image-models: v0.8.10dev0 Release, Feb. 2023. URL https://zenodo.org/record/4414861.

[41] G. I. Winata, S. Cahyawijaya, Z. Lin, Z. Liu, and P. 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)*, pages 6144–6148, May 2020. doi: 10.1109/ICASSP40776.2020.9053878. ISSN 2379-190X.

[42] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*, 2021.

[43] C. Xu and J. McAuley. A Survey on Model Compression and Accelera-tion 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.

[44] Y. Xu, Y. Li, S. Zhang, W. Wen, B. Wang, W. Dai, Y. Qi, Y. Chen, W. Lin, and H. Xiong. Trained Rank Pruning for Efficient Deep Neural Networks. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, pages 14–17, 2019. doi: 10.1109/EMC2-NIPS53020.2019.00011.

[45] Z. Yang, A. Zhang, and A. 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.

[46] B. Zhang, W. Zheng, J. Zhou, and J. 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.

[47] H. Zhou, S. Zhang, J. Peng, et al. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting, Mar. 2021. URL http://arxiv.org/abs/2012.07436. arXiv:2012.07436 [cs].