Monomial Matrix Relocation on the Loss Function Level-Set of Feedforward Neural Networks

Ozgur Soysal, Arda Ozdemir, Yigit Yildirim and Orhan Arikan
Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey
E-mail: ozgur.soysal@ug.bilkent.edu.tr, arda.ozdemir@ug.bilkent.edu.tr, yigit.yildirim@bilkent.edu.tr, oarikan@ee.bilkent.edu.tr

***Abstract*—Gradient-based optimizers are crucial for the success of deep learning methods; however, such optimizers frequently suffer from slow convergence and can become trapped in suboptimal local minima or plateaus. To overcome this issue, we propose a novel optimization technique called Monomial Equivalence - Interleaved Training (ME-IT) to augment well-known optimizers. ME-IT periodically applies a monomial matrix transform—a computationally cheap, loss-preserving transformation that leverages the inherent symmetries of the network. This "re-locates" the model to a functionally identical but parametrically different location on the loss level-set, enabling the network to explore the loss surface and find favorable points for continued gradient descent. Experiments on the MNIST dataset demonstrate that ME-IT-applied models converge significantly faster and discover final solutions with lower loss and higher accuracy than their state-of-the-art counterparts. A test accuracy improvement of $\sim 0.3\%$ is achieved on the MNIST dataset with respect to the model with Cosine Annealing with Warm Restarts, and a training speedup of around $2\times$ is reached for shorter training sessions. This work establishes that strategically leveraging network symmetries is a practical tool for improving and accelerating deep learning optimization.**

I. Introduction

Successful Deep Learning (DL) methods rely on the efficacy of gradient-based optimizers. Even though optimizers like Stochastic Gradient Descent (SGD) or Adam [1] have been deployed to a great extent for training complex neural architectures, descending a non-convex landscape with such methods [2] has various important aspects to consider for achieving high-performance in a short time. Namely, the unknown and challenging topology of the loss landscape may force the existing optimizers to converge to suboptimal minima, if not stall them completely. Current optimizers tend to get stuck on wide and flat plateaus and take many iterations to converge. Even so, inability to effectively escape local minima causes models to have poor generalization performance.

Overcoming such limitations is an important challenge in DL. To this end, we propose a novel training scheme, *ME-IT*. ME-IT interleaves parameter updates with loss-preserving network-wide transformations to descend from a loss-equivalent but steeper point on the loss surface. Namely, network weights can be transformed via *monomial groups* that exploit inherent symmetries of feedforward neural networks (FNNs). [3] has previously established that FNNs exhibit equivalences under monomial group actions under certain conditions. Inspired by this, we propose to exploit such equivalences and transform the network weight before parameter updates via computationally inexpensive, loss-preserving monomial transforms. This allows the network to continue its descend from a different point on the same loss level set. Thus, our contributions in this work are threefold:

1) We propose *ME-IT* algorithm, a novel method that interleaves transforms between monomial equivalent networks and parameter updates to effectively escape loss plateaus and suboptimal parameter configurations.
2) We empirically demonstrate that training standard models with ME-IT leads to significantly faster convergence rates and the discovery of better final solutions on benchmark classification tasks using the MNIST [4] dataset.
3) We show that the resulting models achieve not only lower final training loss but also higher test accuracy, indicating that the ability to escape poor local minima translates to improved generalization.

In Section II, we present a review of existing network symmetries and studies on loss surfaces. In Section III, we describe ME-IT algorithm with key observations. Then, we compare ME-IT with *Cosine Annealing with Warm Restarts* (CAWR) learning rate scheduler empirically and conduct ablation studies on ME-IT parameters in Section IV.

II. Related Work

A. Permutation Symmetry in Feedforward Networks

The existence of symmetries in the parameter space of FNNs has long been recognized as a fundamental property. Foundational work [5] shows network parameters to approximate the target input-output mapping are not necessarily unique. One particular and well-established equivalence is *permutation symmetry*, which can be achieved by rearranging n many neurons of a Multi-Layer Perceptron (MLP). By permuting corresponding neurons in adjacent layers, the input-output function of the overall network can be preserved, implying at least $n!$ many equivalent parameter configurations for each such layer, leading to a factorial explosion of functionally identical solutions in the total weight space [6].

Even though the existence of such equivalences implies global minima points are repeated, it also implies optimization of network parameters initialized from a random point will be challenging. Namely, [7] investigated how these symmetries affect the loss landscape geometrically, and demonstrated that

979-8-3315-7206-8/25/$31.00

permutation symmetries induce first-order **saddle points** located on the paths connecting multiple equivalent minima. [7] proved continuous paths between symmetric solutions exist. This shows that these are critical points located within high-dimensional plateaus of constant loss. The presence of these symmetry-induced saddle points and the associated flat regions can significantly slow or stall the convergence of first-order optimization methods like SGD, which rely on local gradient information to navigate the parameter space.

[3] has characterized **monomial matrix group** action on the weight space of a network. By definition, a monomial matrix is the product of a permutation matrix and a non-singular diagonal matrix. It is a square matrix with exactly one non-zero entry in each row and column. This definition implies that monomial matrices combine discrete permutations with continuous scaling, as pointed out by [3]. This extends discrete permutation symmetry to a broader and more general group of continuous transformations, particularly for networks employing positively homogeneous activation functions, such as the Rectified Linear Unit (ReLU) and its variants. For these networks, the computed function is invariant not only to neuron permutation but also to scaling transformations. Specifically, the incoming weights of any given neuron can be scaled by a positive scalar c, and as long as its outgoing weights are correspondingly scaled by $1/c$, the network's input-output function remains unchanged. [3] established that for many standard feedforward architectures with ReLU-like activations, the monomial matrix group represents the maximal symmetry group of the network function.

Our work is predicated on this insight, using transformations from this group to navigate the parameter space. Specifically, these insights hint that optimal solutions do not exist as isolated points in the weight space. Instead, they form continuous, high-dimensional manifolds or **level-sets** of constant loss. Hence, a parameter vector θ can be transformed via a symmetry operation, e.g., a monomial matrix transformation, into θ' such that the loss $\mathcal{L}(\theta) = \mathcal{L}(\theta')$. Thus, we propose to transform the optimization problem from a search for a single point to the identification of a desirable high-dimensional manifold of solutions.

B. Mode Connectivity

The notion of optimal points lying on continuous manifolds had been empirically and theoretically established with the concept of *mode connectivity*. This phenomenon refers to the ability to find a simple, high-accuracy path through the weight space that connects two distinct solutions found by independent training runs of SGD. Initial work [8] demonstrated that it is possible to find simple curves (e.g., Bezier curves with only one or two control points) connecting two independently trained models, along which the training and test error remain arbitrarily low.

This finding was further investigated by [9], who argued that for sufficiently overparameterized networks, these minima are not just pairwise connected but belong to a single, contiguous basin of attraction. They provided evidence that "essentially no barriers" with high loss exist between these solutions. This body of research strongly suggests that the set of good solutions forms a single, connected component in the weight space. This topological structure provides the geometric foundation for the proposed "relocation" mechanism, which is essentially a discrete method for traversing these level-sets.

C. Implicit Geometric Methods

Some recent work argues that wide, flat minima tend to generalize better than sharp, narrow ones. Accordingly, methods such as **Stochastic Weight Averaging (SWA)** [10] and **Sharpness-Aware Minimization (SAM)** [11] have been proposed to exploit the loss landscape's inherent structure rather than employing algebraic symmetries. SWA maintains a running average of the model weights encountered during the later epochs of training. SAM seeks to find parameter values that lie in neighborhoods with uniformly low loss, effectively finding flat minima by solving a min-max optimization problem. These methods mainly focus on improving the generalization of neural networks and reducing gradient noise. Yet, due to their bias towards wide minima, they tend to take some time to converge fully.

D. Explicit Symmetry-Based Methods

In contrast, other approaches have sought to directly exploit the known algebraic symmetries of the weight space. A prominent application is in model merging and ensembling. [12] developed a technique, "Git Re-Basin," that can merge two independently trained models by finding an optimal permutation that aligns their neurons. By solving the assignment problem to match neurons between the models before averaging their weights, this method places them into an equivalent basin of attraction, enabling effective merging where naive weight averaging would fail. This work directly addresses permutation invariance to achieve its goal. Yet, it requires two independently trained models so that their parameters can be averaged. Concurrently, other lines of research have explored the **dynamic rewiring** of network connectivity during training, using learnable permutations to achieve structural plasticity, particularly for tasks in non-stationary environments [13].

E. Learning Rate Schedulers

To increase model generalization and decrease the training time, learning rate schedulers have been proposed to adaptively tune the learning rate during training. In [14], the classical momentum technique has been proposed, in which the optimizer accumulates a velocity vector parallel to a persistent reduction and increases the learning rate when moving in that direction. This allows for faster learning at steep loss surfaces. In [15], *Cosine Annealing with Warm Restarts* (CAWR) has been proposed, and it has been the preference of DL applications since. CAWR reduces the learning rate until a predefined threshold in a $\cos$ wave and jumps back to the initial rate. [15] has shown that much faster convergence and better generalization is possible with CAWR. In Section IV, we compare ME-IT and CAWR empirically.

III. Monomial Equivalence - Interleaved Training (ME-IT) Method

In this section, we describe the **ME-IT** method that aims to augment the gradient descent with loss-preserving structured monomial transforms to hopefully continue the descent from a more favorable location on the loss surface. We begin by formally defining monomial matrices in Definition III.1.

Definition III.1. A matrix A of size $n \times n$ is called a *monomial matrix* (or *generalized permutation matrix*) if A has exactly one non-zero entry in each row and column. We will use $\mathcal{G}_n$ to denote the set of such matrices.

Note that Definition III.1 implies $\mathcal{G}_n$ can also be written as

$$\mathcal{G}_n = \{PD \mid P \in \mathcal{P}_n,\ D \in \mathcal{D}_n\} \ , \tag{1}$$

where $\mathcal{P}_n$ and $\mathcal{D}_n$ are the sets of $n \times n$ permutation matrices and invertible diagonal matrices on $\mathbb{R}^{n\times n}$, respectively. Now, suppose $f : \mathbb{R}^n \to \mathbb{R}^m$ represents an FNN with weight matrices $\{W_1,\ W_2,\ \ldots,\ W_{L-1},\ W_L\}$, bias vectors $\{b_1, \ldots, b_L\}$, and piecewise linear activation functions $\{\sigma_1, \ldots, \sigma_L\}$ at each layer. Then the adjusted FNN $\hat{f}$ with the same structure as f, but with weights $\{M_1W_1,\ M_2W_2M_1^{-1},\ \ldots,\ M_{L-1}W_{L-1}M_{L-2}^{-1},\ W_LM_{L-1}^{-1}\}$ and bias vectors $\{M_1b_1,\ M_2b_2,\ \ldots,\ M_{L-1}b_{L-1},\ b_L\}$ is functionally equivalent to f, $\forall M_i \in \mathcal{G}_n^+$ (i.e. $f(x) = \hat{f}(x) \quad \forall x \in \mathbb{R}^n$). This can be seen by expanding the feedforward operation of an input x as

$$\begin{aligned}
&\sigma_L\Big[W_LM_{L-1}^{-1}\sigma_{L-1}\Big(M_{L-1}W_{L-1}M_{L-2}^{-1}\sigma_{L-2}(\cdots) \\
&\quad + M_{L-1}b_{L-1}\Big) + b_L\Big] \\
&\quad = \sigma_L\Big[W_L\sigma_{L-1}\big(W_{L-1}\sigma_{L-2}(\cdots) + b_{L-1}\big) + b_L\Big] \ ,
\end{aligned} \tag{2}$$

since activations σ_l, $\forall l \in [1, L]$ are piecewise linear. This observation leads to the following lemma.

Lemma III.1. A set of generalized permutation matrices $\{M_i\}$ acting on a feedforward neural network f transforms f onto a new network $\hat{f}$ on the level set of the loss surface.

The proposed ME-IT technique builds upon Lemma III.1, and augments existing gradient-based methods. Thus, the core principle is to hybridize the training process by introducing a new operation, the "relocate," to complement the standard descent step. The training loop alternates between two primary states:

- **DESCEND:** In this state, the model parameters W are updated using the chosen base optimizer according to the local gradient of the loss function $\mathcal{L}(W)$.
- **RELOCATE:**A loss-preserving transformation is applied to the weights, moving the model to a new point W' such that $\mathcal{L}(W') = \mathcal{L}(W)$.

This allows the optimizer to escape unfavorable local geometries from which the **DESCEND** operation alone cannot easily recover.

A. The Monomial Relocation Transformation

The key mechanism enabling the **RELOCATE** operation is a symmetry transformation based on monomial matrices. A monomial matrix $M \in \mathcal{G}_n$ is a square matrix having exactly one non-zero entry in each row and column.

Given a hidden layer l with weight matrix W_l and bias b_l and the subsequent layer $l+1$ with weights W_{l+1}, the monomial relocation is defined by the following updates:

$$W_l \to MW_l \ , \tag{3}$$

$$W_{l+1} \to W_{l+1}M^{-1} \ , \tag{4}$$

$$b_l \to Mb_l \ . \tag{5}$$

The permutation matrix P effectively reorders the neurons in layer l, while the diagonal matrix D scales them. The compensating transformation M^{-1} applied to the next layer's weights perfectly neutralizes this effect, given the activation function is homogeneous, ensuring that the input to layer $l+2$ and all subsequent network outputs remain unchanged. Consequently, the value of the loss function $\mathcal{L}(W)$ is exactly preserved. The above transformation is applied to each layer in the network. An important aspect to consider is when to apply the ME-IT transformation to the network weights. The following lemma demonstrates that transforming before and after gradient steps would yield different results.

Lemma III.2. Let $M(\cdot)$ denote the ME-IT transformation and SGD$(\cdot)$ denote one step of gradient descent. Then, operators $M(\cdot)$ and SGD$(\cdot)$ do not commute, i.e.,

$$\text{SGD}(M(\theta)) \neq M(\text{SGD}(\theta)) \ . \tag{6}$$

Proof: Suppose SGD$(M(\theta)) = M(\text{SGD}(\theta))$. Observe that transforming network weights after gradient descent yields the following parameter updates for layer i:

$$W_i \leftarrow MW_i - \eta M\left(\nabla_{W_i}L\right)_\theta \ , \tag{7}$$

$$b_i \leftarrow Mb_i - \eta M\left(\nabla_{b_i}L\right)_\theta \ , \tag{8}$$

$$W_{i+1} \leftarrow W_{i+1}M^{-1} - \eta\left(\nabla_{W_{i+1}}L\right)_\theta M^{-1} \ , \tag{9}$$

where $(\nabla_{W_i}L)_\theta$, $(\nabla_{b_i}L)_\theta$, $\left(\nabla_{W_{i+1}}L\right)_\theta$ denote the gradient of the loss L at point θ w.r.t. parameters W_i, b_i, and W_{i+1}, respectively, and η is the learning rate. On the other hand, transformed parameters $W_i' := MW_i$, $b_i' := Mb_i$, and $W_{i+1}' := W_{i+1}M^{-1}$ will yield parameter updates:

$$W_i' \leftarrow W_i' - \eta\left(\nabla_{W_i'}L\right)_{\theta'} \ , \tag{10}$$

$$b_i' \leftarrow b_i' - \eta\left(\nabla_{b_i'}L\right)_{\theta'} \ , \tag{11}$$

$$W_{i+1}' \leftarrow W_{i+1}' - \eta\left(\nabla_{W_{i+1}'}L\right)_{\theta'} \ . \tag{12}$$

Now, observe that gradients w.r.t. transformed parameters can be shown as

$$\left(\nabla_{W_i'}L\right)_{\theta'} = M^{-T}\left(\nabla_{W_i}L\right)_\theta \ , \tag{13}$$

$$\left(\nabla_{b_i'}L\right)_{\theta'} = M^{-T}\left(\nabla_{b_i}L\right)_\theta \ , \tag{14}$$

$$\left(\nabla_{W_{i+1}'}L\right)_{\theta'} = \left(\nabla_{W_{i+1}}L\right)_\theta M^T \ . \tag{15}$$

Thus, $\text{SGD}(M(\theta)) = M(\text{SGD}(\theta))$ implies $M^T = M^{-1}$, yet this forms a contradiction as $M^T M = D^2 \neq I$ for a general monomial matrix. This completes the proof. ∎

Lemma III.2 shows that the order of ME-IT and parameter updates matters. Thus, we apply ME-ITs before each parameter update so that the training process can be sped up significantly when the transformation carries the network parameters to a much steeper point. In the next section, a method to ensure this is presented.

Remark. A critical implementation detail is the handling of the base optimizer's internal state. When a relocation is performed on weights W_l and W_{l+1}, their local geometric context is completely changed. Therefore, the historical information stored by the optimizer, such as momentum and variance estimates in Adam [1], becomes invalid for these specific parameters. One can choose to either reset such variables or adapt them using the relocation matrices.

B. Generating Relocation Matrices

In this section, we propose a heuristic for generating $M = PD$ such that the relocation operation yields maximal effect on training, and the transformed network parameters are at a steeper point. The proposed heuristic aims to maintain training stability by setting a maximal benefit factor r. Using the update equations (10), (11), and (12) with the known fact that $\Delta\mathcal{L} = -\eta \text{Tr}(\nabla\mathcal{L}^T \nabla\mathcal{L})$ for gradient descent, the expected loss change of doing a descent operation (under the linearity assumption) without and with a relocation can be written, respectively, as:

$$(\Delta\mathcal{L})_{GD} = -\eta \sum_{j=1}^{N} \left(\|(\nabla_{W_{i+1}}\mathcal{L})^j\|_2^2 + \|(\nabla_{W_i}\mathcal{L})_j\|_2^2 + \|\nabla_{b_i}\mathcal{L}\|_2^2 \right) ,$$

$$(\Delta\mathcal{L})_R = -\eta \sum_{j=1}^{N} \Bigg(d_j^4 \|(\nabla_{W_{i+1}}\mathcal{L})^j\|_2^2 + \frac{1}{d_j^4} \left(\|(\nabla_{W_i}\mathcal{L})_j\|_2^2 + \|\nabla_{b_i}\mathcal{L}\|_2^2 \right) \Bigg) . \tag{16}$$

where $(\nabla_{W_{i+1}}\mathcal{L})^j$ denotes the j'th row of $\nabla_{W_{i+1}}\mathcal{L}$. Let $f_j(d_j) \coloneqq (\Delta\mathcal{L})_R / (\Delta\mathcal{L})_{GD}$. Maximizing $f_j(d_j)$ would imply the maximal benefit and the fastest descent is achieved under the linear approximation assumption. Yet, observe that $f_j(\cdot)$ is a concave function and $f_j(d_j)$ are unbounded, thus it can endanger training stability. Furthermore, the Hessian of the parameters grows with d_j^8, invalidating the benefit functions above by changing linearity around the point θ. This necessitates imposing some limits on d_j. When $d_j = 1$, we have $f_j(d_j) = 1$, implying ME-IT is equivalent to usual gradient descent. Observe that d_j^4 appears as a multiplicative and divisive term in (16). To avoid gradients blowing up or diminishing, we propose to set d_j to conservative values around 1. Namely, for $d_j > 1$, the term with $\|\nabla_{W_{i+1}}\mathcal{L}\|_2^2$ would dominate the sum in (16). Likewise, when $d_j < 1$, the sum in (16) is being dominated by $\|(\nabla_{W_i}\mathcal{L})_j\|_2^2 + \|\nabla_{b_i}\mathcal{L}\|_2^2$. Consider the case when $f_j(d_j) = r$, for some $r \in \mathbb{R}$. The corresponding $\hat{d}_j = d_j|_{f_j(d_j)=r}$ can be either $\hat{d}_j > 1$ or $\hat{d}_j < 1$. When $\hat{d}_j \ll 1$, we clamp $\hat{d}_j$ to $1 - \sigma$, so that (16) does not blow to infinity. Similarly, due to unbounded nature of f_j, we clamp $\hat{d}_j$ to $1 + \sigma$ when $\hat{d}_j \gg 1$. Hence, for a desired benefit factor r, the final value for $D = \text{diag}(\overline{d}_1, \ldots, \overline{d}_N)$ is given by

$$\overline{d}_j = \text{clamp}(\hat{d}_j, [1 - \sigma, 1 + \sigma]) , \tag{17}$$

where σ and r are the hyper-parameters for ME-IT. By tuning σ and r, it is possible to speed up the training process significantly while maintaining conservative jumps so that the resulting point from which SGD will be conducted is still stable.

By Lemma III.1 and selecting D such that $f_j(d_j) \geq 1$, the transformed point is at least as favorable as the original parameters θ in terms of gradient descent performance. This way, we aim to continue descending on a steeper surface until the next relocation phase.

IV. Results

All experiments are conducted on an MLP with ReLU activations shown in Fig. 1, trained on the MNIST [4] dataset with 60K training images and 10K test images. The hyper-parameters for the benchmark CAWR are: number of iterations before the first restart $T_0 = 10$, increasing factor of epoch interval between two restarts $T_{mult} = 1$, minimum learning rate $\eta_{min} = 0$.

A. Training and Test Performance

We begin our empirical analysis with a direct comparison of the training dynamics between a standard SGD optimizer with the addition of CAWR and our proposed ME-IT optimizer. Fig. 2 illustrates a representative training run on the target dataset over 100 epochs, demonstrating the decay of the training loss and the rise of test accuracy for various baseline models along with ME-IT trained model.

The results demonstrate that ME-IT can make the training loss descend rapidly around epoch 30, and converge to a lower value than all baseline models. Furthermore, this sharp decrease also affects the test accuracy, obtaining a test accuracy 0.3% higher than the highest baseline. This demonstrates that ME-IT optimizer poses as an efficient optimization method that can escape local minima to quickly converge.

ME-IT optimizer tends to fall behind the baseline until epoch 30 during training. This slow-start problem can be overcome by using learning rate scheduling whenever the jump strategy falls on the conservative side, where monomial matrices have condition numbers very close to 1. This regime is close to just using SGD, and is suitable for learning rate schedulers to speed up the training process.

B. Robustness and Interaction with Learning Rate

In this section, we conduct a grid search to investigate how σ and r choices change the performance and generalization of an ME-IT trained model. We evaluate the performance of the ME-IT optimizer against the baseline across five σ values (ranging from 0.3 down to 0.1). For each learning rate, we varied r between 5 and 40. The results, averaged over multiple runs, are presented in Fig. 3.

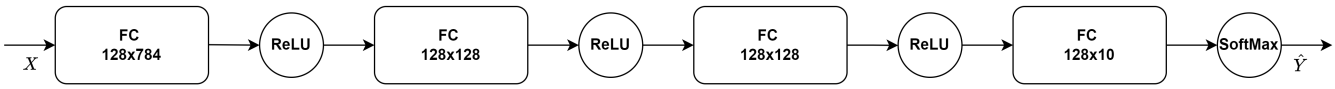

Fig. 1. MLP Architecture on which Experiments are Conducted

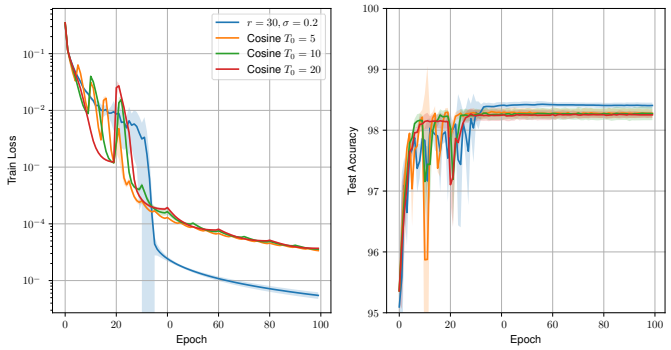

Fig. 2. Training dynamics comparing our ME-IT augmented optimizer (blue) against a standard baselines. **(a)** Training Loss over 200 epochs. **(b)** Test Accuracy (%). The base learning rate is 0.1 unless otherwise stated.

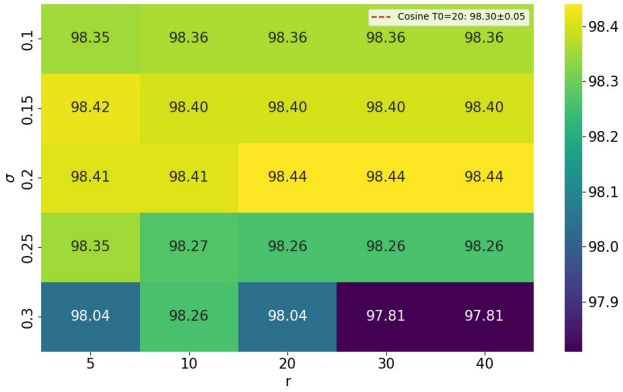

Fig. 3. Final test accuracy after 100 epochs as a function of jump variance, evaluated across five different σ and r values. The baseline can be seen at the upper right corner.

The most salient observation from Fig. 3 is that the ME-IT-enhanced optimizer consistently outperforms the baseline optimizer across a σ region of $[0.1, 0.25]$. For $\sigma = 0.3$, however, we observe that the ME-IT method starts to perform worse. For each $\sigma \in [0.1, 0.25]$, the performance of the ME-IT model is statistically higher than the baseline, as indicated by the non-overlapping standard deviations. This strongly suggests that the benefits of the jump mechanism are not contingent on a finely-tuned hyperparameter but are instead available for a wide range of σ and r choices.

Lastly, we investigate learning rates $\eta = [0.05, 0.1, 0.15, 0.25, 0.3]$ to see how the benchmark and proposed ME-IT perform. We test with $\sigma = 0.05, 0.1, 0.2$ for each learning rate. The resulting test accuracies averaged over 5 different seeds are given in Fig. 4.

In Fig. 4, ME-IT with $\sigma = 0.05$ obtains the highest peak accuracy, whereas ME-IT with $\sigma = 0.1$ comes second. Under high learning rates, $\sigma > 0.05$ creates stability issues and causes

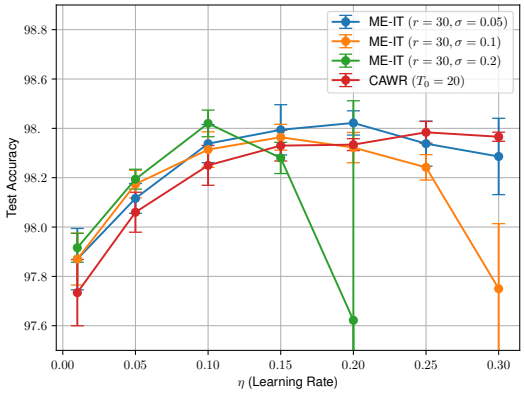

Fig. 4. Relation between Learning Rate and Test Accuracy of Two Optimization Methods

the network to diverge. Indicating η and σ relation is important to fine-tune. Specifically, one needs to decrease σ to increase η, indicating an important trade-off. Study of this trade-off is left open for further research. CAWR benchmark tends to perform better with increasing learning rate until $\eta = 0.25$ where is peaks at 98.36%, then a small performance decrease is observed. Yet, we note that due to the cos nature, the effective learning rate is $\eta/2$. **We conclude that CAWR has more tolerance to η choice, while ME-IT achieves the highest peak performance.** Note that for a larger η, CAWR would also diverge as well.

V. Conclusions

In this work, we addressed the persistent challenge of gradient-based optimizers becoming trapped in suboptimal regions of the neural network loss landscape. We introduced ME-IT, a novel optimization technique that improves standard models by leveraging the inherent monomial symmetries of feedforward networks. By periodically applying computationally inexpensive, loss-preserving jumps, our method effectively relocates the model to new regions of the parameter space, allowing it to escape plateaus and poor local minima to find more effective descent paths.

We empirically demonstrated that ME-IT optimizer leads to better generalization and higher test accuracy with shorter converge times. This implies less data will be required to train the network. We proposed a heuristic method for calculating such jumps and investigated the introduced hyper-parameters. We have shown the benefits of ME-IT optimizer are available for a wide-range of hyper-parameter configurations. Yet, we have observed that setting too large a learning rate may cause the ME-IT optimizer to diverge. How to choose diagonal matrices D so that divergence is prevented with maximal benefit is still an open question, and left as a further study.

This study was conducted on feedforward neural networks, and the direct extension of this specific monomial jump to architectures with different symmetry groups, such as Convolutional Neural Networks, remains another open question. While we have shown that ME-IT finds "better" regions of the loss landscape, a deeper theoretical understanding of the geometric properties that make these regions more favorable for optimization is a subject for further investigation. Finally, beyond finding a single superior model, ME-IT could be used as a powerful tool to generate ensembles of diverse, high-performing models from a single training run, potentially at a negligible additional computational cost.

References

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

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0.

[3] V.-H. Tran, T. N. Vo, T. H. Tran, A. T. Nguyen, and T. M. Nguyen, "Monomial matrix group equivariant neural functional networks," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 48 628–48 665.

[4] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[5] R. Hecht-Nielsen, *Neurocomputing*. Reading, MA, USA: Addison-Wesley, 1990.

[6] R. Hecht-Nielsen, "On the algebraic structure of feedforward network weight spaces," in *Advanced Neural Computers*, R. ECKMILLER, Ed., Amsterdam: North-Holland, 1990, pp. 129–135, ISBN: 978-0-444-88400-8. DOI: https://doi.org/10.1016/B978-0-444-88400-8.50019-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444884008500194.

[7] J. Brea, B. Simsek, B. Illing, and W. Gerstner, "Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape," *arXiv preprint arXiv:1907.02911*, 2019. arXiv: 1907.02911.

[8] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," in *Advances in Neural Information Processing Systems*, 2018, pp. 8799–8808.

[9] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht, "Essentially no barriers in neural network energy landscapes," in *International Conference on Machine Learning*, 2018, pp. 1309–1318.

[10] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Uncertainty in Artificial Intelligence*, 2018, pp. 876–885.

[11] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," in *International Conference on Learning Representations*, 2021.

[12] M. K. Ainsworth, J. Hayase, and S. S. Srinivasa, "Git re-basin: Merging models modulo permutation symmetries," in *International Conference on Learning Representations*, 2023.

[13] Z. Sun and Y. Mu, "Rewiring neurons in non-stationary environments," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 28 173–28 186.

[14] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964, ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(64)90137-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0041555364901375.

[15] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with restarts," *CoRR*, vol. abs/1608.03983, 2016. arXiv: 1608.03983. [Online]. Available: http://arxiv.org/abs/1608.03983.