GL Equivariant Metanetworks for Learning on Low Rank Weight Spaces

Anonymous Author(s)
Affiliation
Address
email

Abstract

Low-rank adaptations (LoRAs) have revolutionized the finetuning of large foundation models, enabling efficient adaptation even with limited computational resources. The resulting proliferation of LoRAs together with the recent advances of weight-space learning present exciting opportunities for applying machine learning techniques that take these low-rank weights themselves as inputs. In this paper, we investigate the potential of *Learning on LoRAs* (LoL), a setup where machine learning models learn and make predictions on datasets of LoRA weights. Motivated by previous weight-space learning works, we first identify the inherent parameter symmetries of our data – low-rank decompositions of weights – which differ significantly from the parameter symmetries of standard neural networks. To efficiently process LoRA weights, we develop several symmetry-aware invariant or equivariant LoL models. In diverse experiments, we show that our LoL architectures can process LoRA weights to predict CLIP scores, finetuning data attributes, finetuning data membership, and accuracy on downstream tasks. We also show that LoL models trained on LoRAs of one pretrained model can effectively generalize to LoRAs trained on other models from the same model family. As an example of the utility of LoL, our LoL models can accurately estimate CLIP scores of diffusion models and ARC-C test accuracy of LLMs over 50,000 times faster than standard evaluation. As part of this work, we finetuned and will release datasets of more than ten thousand text-to-image diffusion-model and language-model LoRAs. Our anonymized code can be found here.

1 Introduction

Finetuning pretrained models such as large language models [6, 79, 15] and text-to-image generative models [31, 63] for improved performance on target tasks has become an extremely common and successful paradigm in deep learning. While full finetuning of all weights effectively boosts model capabilities, it requires large amounts of memory and computation time. In recent years, Low Rank Adaptation (LoRA) [35], a finetuning method where a learnable low rank decomposition is added to each weight matrix ($W_i \mapsto W_i + U_i V_i^\top$), has been used as an alternative to other finetuning methods thanks to its increased efficiency. LoRA finetuning and its variants have become widespread; there are now software packages [51, 27], paid services [62], and online communities [7] in which countless LoRAs are trained and shared.

Given the ubiquity of LoRA weights, one can imagine treating them as a *data modality*. Recent works in the emerging field of *weight-space learning* train neural networks to operate on the weights of other models. These networks, often termed meta-networks [43], neural functionals [83, 82], or deep weight-space networks [57], have demonstrated promise on diverse parameter-level tasks, including generating or editing neural fields [61], predicting pruning masks [83], merging models [58], learned

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

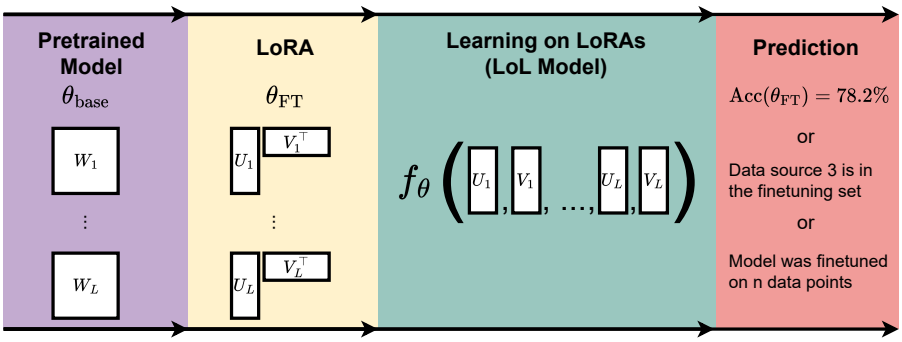


Figure 1: Overview of Learning on LoRAs (LoL). A pretrained model θ_{base} is finetuned to yield LoRA weight matrices $U_1, V_1, \ldots, U_L, V_L$. These LoRA weights are taken as input to an LoL model f_θ, which can make predictions such as the downstream accuracy of the finetuned model.

optimization [56, 82, 39, 25], and predicting performance of neural network checkpoints [76, 68]. However, their reliance on processing the entire weight space limits their applicability to smaller models. Importantly, due to the unique structure of LoRAs, adapting previous weight-space methods to operate on LoRAs poses significant challenges, mainly due to the different symmetry these data types adhere to.

Our approach. In this work, we extend weight-space learning to *LoRA Weight-Spaces*. We introduce and extensively study Learning on LoRAs (LoL), encompassing theory, architectural design, and a variety of applications. When developing LoL architectures for processing LoRA weights, we aim to account for the structure and symmetries of this unique data modality. First, the rank-r decomposition of an $n \times m$ matrix has only $(n+m)r$ parameters compared to nm for the full dense matrix. This parameter efficiency (linear vs quadratic) is one of the main reasons for the popularity of LoRAs. Hence, we opt for using the low-rank decomposition representation (U, V) in the place of the full matrix $UV^\top$, so we can more efficiently learn on LoRA weights. As a consequence of choosing this efficient data representation, taking the geometric deep learning [5] blueprint as a guiding principle, we take into consideration its unique symmetry structure: for any invertible matrix $R \in \text{GL}(r)$, we have that $URR^{-1}V^\top = UV^\top$, so the low rank decompositions (U, V) and $(UR, VR^{-\top})$ are functionally equivalent [1]. Because this transformation does not affect the underlying function represented by the LoRA, almost all relevant LoL problems are invariant to this action of $\text{GL}(r)$. Therefore, an effective LoL model for any such task should be $\text{GL}(r)$-invariant.

To this end, motivated by previous success in invariant and equivariant weight-space learning [57, 39, 43, 82, 38] and geometric deep learning in general [5, 9, 81, 22, 40, 54, 42], we propose several $\text{GL}(r)$-equivariant and invariant neural architectures that can effectively process LoRA weights. These models are designed using multiple techniques from geometric deep learning (canonicalization, invariant featurization, and equivariant linear maps) and offer different trade-offs in terms of efficiency, expressivity, and generalization.

To explore the feasibility of Learning on LoRAs tasks, and to analyze the effectiveness of our proposed architectures, we conduct experiments across various finetuned models. First, we create novel datasets for Learning on LoRAs; we train over ten thousand diverse LoRAs, which are finetuned from text-to-image diffusion generative models and language models. We then train LoL models on these LoRAs to predict various attributes of finetuned models, including: CLIP scores, finetuning data attributes, finetuning data membership, and accuracy on downstream tasks. We find that the $\text{GL}(r)$-invariant LoL models we propose in this paper can often perform these tasks well, and considerably better than standard non-invariant models and other naive approaches. We also show that our LoL models can generalize to LoRAs that were trained on top of other base models from the same model family, opening the opportunity to train single LoL models for entire model families.

As an example application, trained LoL models offer a dramatically faster alternative to standard evaluation methods for metrics like CLIP score and language model common-sense reasoning accuracy. In our experiments, an LoL model can estimate the CLIP score of a diffusion model **53,000 times**

[1] Throughout the paper, we use $R^{-\top}$ to denote $(R^{-1})^\top$.

faster than traditional methods, and assess the downstream LLM common-sense reasoning accuracy **730,000 times faster**. This speedup stems from the fact that standard model evaluation requires thousands of forward passes through a multi-billion parameter model, while an LoL model accomplishes the same task with a single forward pass through a much smaller network. This rapid evaluation can enable faster hyperparameter tuning and real-time performance monitoring during training.

Our contributions. In this paper, we (1) introduce Learning on LoRAs (LoL), extending weight-space learning to low rank weight spaces; (2) characterize the inherent $\mathrm{GL}(r)$ symmetries in LoRAs that pose unique challenges and develop several $\mathrm{GL}(r)$-invariant and equivariant architectures that effectively and efficiently process LoRA weights; (3) create and release datasets of over ten thousand text-to-image and language model LoRAs; (4) demonstrate that our symmetry-aware LoL models perform well across a range of tasks.

2 Background and Related Work

LoRA background and symmetries Hu et al. [35] introduced the LoRA method, which is a parameter-efficient method for finetuning (typically large) models [51, 28]. Consider a weight matrix $W \in \mathbb{R}^{n\times m}$ of some pretrained neural network. Directly finetuning W would require training nm parameters on additional data, which could be quite expensive. LoRA instead trains low-rank matrices $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{m\times r}$ so that the new finetuned weight matrix is given by $W + UV^\top$. This only requires tuning $(n+m)r$ parameters, which is efficient as the rank r is taken to be significantly lower than n or m (for instance a rank of $r = 8$ can be sufficient to finetune a 7B-parameter language model with $n = m = 4096$ [47]).

As mentioned in the introduction, for any invertible matrix $R \in \mathrm{GL}(r)$, $W + (UR)(VR^{-\top})^\top = W + UV^\top$, so the LoRA update given by $(UR, VR^{-\top})$ is functionally equivalent to the one given by (U, V). As a special case, when $Q \in \mathrm{O}(r) < \mathrm{GL}(r)$ is orthogonal, (UQ, VQ) is also functionally equivalent. Many variants of the original LoRA work have been proposed, and they often have comparable symmetries that can be handled similarly in our framework [51, 28]; we discuss LoRA variants and their symmetries in Appendix E.

Prior works in the geometric deep learning community [5] have also studied continuous symmetries [74, 4, 17, 66, 44, 59, 41] such as rotation and scaling symmetries, especially for applications in the chemical and physical sciences. However, they study different classes of symmetries, such as $\mathrm{O}(n)$, $\mathrm{E}(n)$, and $\mathrm{SP}(n)$, none of which contain $\mathrm{GL}(n)$. To the best of our knowledge, we are the first to study $\mathrm{GL}(n)$ equivariant and invariant architectures.

Weight-space learning. Several neural network architectures have been developed that take in neural network weights as input [76, 20, 68, 69, 70, 71, 56, 60, 57, 58, 2, 72]. In many tasks, equivariance or invariance to parameter transformations that leave the network functionally unchanged has been found to be useful for empirical performance of weight-space networks [57, 83, 43, 38, 75]. However, these works are not specialized for LoRA weight spaces, since they consider different symmetry groups: many such works focus only on discrete permutation symmetries [57, 83, 43, 82], or scaling symmetries induced by nonlinearities [38, 75]. As covered in the previous section, LoRA weights have general linear group (invertible matrices) symmetry, which includes as special cases certain permutations and scaling symmetries between U and $V^\top$. While LoRAs contain inner permutation symmetries of the form $UP^\top PV^\top = UV^\top$, we emphasize that they generally do not have outer permutation symmetries of the more recognizable form $P_1UV^\top P_2$, which would resemble those of standard weight spaces (e.g. MLPs or CNNs) more. This is an important difference; for instance, the outer permutation symmetries significantly harm linear merging of models trained from scratch [21, 1, 45], whereas finetuned models can be merged well with simple methods [78, 36].

Recently, two works have explored the weight space of LoRA finetuned models for certain tasks. Salama et al. [65] develop an attack that predicts the size of the dataset used to finetune the model given its finetuned weights. Inspired by correlations between finetuning dataset size and singular value magnitudes, they define a very specific type of LoL model that takes the singular values of dense, multiplied-out LoRA weights $\sigma_1(U_iV_i^\top), \ldots, \sigma_r(U_iV_i^\top)$ as input. In contrast, we study more tasks, consider the problem of general LoL model design, and develop more expressive and efficient LoL models in our paper. In another context, Dravid et al. [13] study the LoRA weight space of finetuned personalized text-to-image diffusion models. They do not define LoL models, but instead study operations such as linear edits in the principal component space of their rank-one LoRA weights.

3 Learning on LoRAs Architectures

Table 1: Properties of LoL architectures proposed in this paper. Runtimes are for one LoRA layer with $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{m\times r}$, so the rank r is generally much lower than n and m. Expressivity refers to a notion of the ability to approximate GL-invariant functions, which is formalized in Definition C.1.

LoL Model	GL-Inv.	O-Inv.	Expressive	Preprocess Time	Forward Time
Naive architectures					
MLP$([U,V])$	✗	✗	✓	$O((m+n)r)$	$O((m+n)r)$
Transformer$([U,V])$	✗	✗	✓	$O((m+n)r)$	$O((m+n)r)$
MLP$(UV^\top)$	✓	✓	✓	$\boldsymbol{O(mnr)}$	$\boldsymbol{O(mn)}$[2]
Efficient symmetry-aware architectures					
MLP(O-Align$([U,V])$)	✗	✓	✓	$O((m+n)r^2)$	$O((m+n)r)$
MLP$(\sigma(UV^\top))$	✓	✓	✗	$O((m+n)r^2)$	$O((m+n)r)$
GL-net	✓	✓	✓	$O((m+n)r)$	$O((m+n)r)$

In this section, we develop several neural network architectures for Learning on LoRAs. These have different trade-offs and properties, which we summarize in Table 1. First, we mathematically formalize the LoL learning problem. Next, we describe three naive methods that apply standard architectures to the LoRA weights and discuss their limitations. We then derive three efficient and symmetry aware architectures. The first two process LoRA weights by canonicalizing the weights or generating invariant features. Finally, we derive a third invariant approach, GL-net, an architecture based on a composition of equivariant and invariant modules. In the experimental section, we demonstrate that symmetry aware architectures significantly outperform the naive baselines.

3.1 Learning Setup

Suppose L matrices in the base model are finetuned via LoRA. The LoRA weights are $(U_1,V_1),\dots,(U_L,V_L)$, where $U_i \in \mathbb{R}^{n_i\times r}$ and $V_i \in \mathbb{R}^{m_i\times r}$. An LoL model with parameters θ and output space $\mathcal{Y}$ is a function $f_\theta : \mathbb{R}^{\sum_i (n_i+m_i)r} \to \mathcal{Y}$. An LoL model can output a scalar prediction ($\mathcal{Y}=\mathbb{R}$) or latent LoRA weight representations $(\tilde{U}_1,\tilde{V}_1),\dots,(\tilde{U}_L,\tilde{V}_L)$ where $\tilde{U}_i \in \mathbb{R}^{n_i'\times r}$ and $\tilde{V}_i \in \mathbb{R}^{m_i'\times r}$ ($\mathcal{Y}=\mathbb{R}^{\sum_i (n_i'+m_i')r}$).

For $(U_1,V_1,\dots,U_L,V_L) \in \mathbb{R}^{\sum_i (n_i+m_i)r}$ and $R_1,\dots,R_L \in \mathrm{GL}(r)$ we denote the action by

$$(R_1,\dots,R_L)\star(U_1,V_1,\dots,U_L,V_L) = (U_1R_1, V_1R_1^{-\top},\dots,U_LR_L,V_LR_L^{-\top}). \tag{1}$$

An LoL model is called GL-*invariant* if for all $R_1,\dots,R_L \in \mathrm{GL}(r)$

$$f_\theta((R_1,\dots,R_L)\star(U_1,V_1,\dots,U_L,V_L)) = f_\theta(U_1,V_1,\dots,U_L,V_L). \tag{2}$$

This represents invariance to the action of the direct product $\mathrm{GL}(r)\times\cdots\times\mathrm{GL}(r)$ of $\mathrm{GL}(r)$ with itself L-times ($\mathrm{GL}(r)^L$). When the output space has decomposition structure, i.e. $f_\theta(U_1,V_1,\dots,U_L,V_L) = (\tilde{U}_1,\tilde{V}_1,\dots,\tilde{U}_L,\tilde{V}_L)$, we say that the model is $\mathrm{GL}(r)$-*equivariant* if for all $R_1,\dots,R_L \in \mathrm{GL}(r)$,

$$f_\theta((R_1,\dots,R_L)\star(U_1,V_1,\dots,U_L,V_L)) = (R_1,\dots,R_L)\star(\tilde{U}_1,\tilde{V}_1,\dots,\tilde{U}_L,\tilde{V}_L). \tag{3}$$

Expressivity is an important property of invariant and equivariant models [81, 53], and LoL models in particular. Informally, an LoL model is universally expressive if it can fit any continuous GL-invariant function to arbitrary accuracy on compact domains. We give a formal definition in Definition C.1, and prove our results in the context of this formal definition.

3.2 Naive Architectures

Simple MLP/transformer on LoRA weights. The simplest LoL model is a standard deep learning model, such as an MLP, that takes the flattened LoRA weights as input: $\mathrm{MLP}_\theta(U_1,V_1,\dots,U_L,V_L)$. This method is efficient (as it doesn't need to form the n-by-m dense matrices $U_iV_i^\top$), and expressive (as a result of the universal approximation theorem [10]), but is *not* invariant to the LoRA parameter symmetries, and thus lacks inductive bias. In the experimental section, we also implement a

[2] In practice, due to memory constraints $UV^\top$ must be recomputed each batch, so forward time is $O(mnr)$.

variant that uses a Transformer instead of an MLP. We refer to these methods as MLP($[U, V]$), and Transformer($[U, V]$). Note that we include these models mainly for comparison purposes. Since they lack the necessary inductive biases to effectively learn from LoRA data, we do not expect them to perform well in practical applications. This observation is confirmed in the experimental section.

Multiplying-out LoRA weights. Another simple and natural method is to first perform matrix multiplications to compute the full dense matrices $U_i V_i^\top$, and then apply a model to these dense matrices. In this paper, we flatten and concatenate these dense matrices, and then apply a simple MLP. This approach is fully GL-invariant, and is universally expressive, but it is *significantly more computationally expensive* since the dense matrices are of size nm as opposed to the size $(n+m)r$ of the low rank decomposition. Furthermore, data pre-processing must be redone for each batch due to memory constraints, so in practice the forward time is $O(mnr)$. Thus, the computational cost of this method makes it inapplicable to real-world LoRAs. This method is denoted by $\mathrm{MLP}(UV^\top)$.

3.3 Efficient Symmetry-Aware Architectures

In this subsection we propose three efficient and symmetry-aware architectures. The first two process the input LoRAs before applying standard architectures. The last architecture, GL-net, is more sophisticated and is a composition of equivariant and invariant building blocks that we suggest.

3.3.1 Symmetry-Aware Methods Based on Standard Architectures

$\mathrm{MLP}(\mathrm{O}$-Align$([U, V]))$: alignment for canonicalization One popular method for designing invariant or equivariant networks is to canonicalize the input by using a symmetry-invariant transformation to convert it into a canonical form [37, 18, 50]. For example, to canonicalize a dataset of point clouds with respect to rotations, one might choose one specific point cloud and then rotate all other point clouds in the dataset to it to maximize their relative similarity. After this alignment, any standard architecture can process the point clouds in an invariant manner without taking rotation into account. This kind of rotation alignment, known as $\mathrm{O}(r)$ canonicalization, admits a closed from solution and is significantly simpler than $\mathrm{GL}(r)$ canonicalization.

We canonicalize LoRA weights U_i, V_i into $\mathrm{O}(r)$-invariant representatives $U_i Q_i, V_i Q_i$ as follows. First, we select template weights $\boldsymbol{U}_i, \boldsymbol{V}_i$ of the same shape as U_i and V_i (in our experiments we randomly select $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ from our training set of LoRAs). Then we align U_i, V_i to the templates by finding the orthogonal matrix Q_i's that most closely match them in Frobenius norm:

$$Q_i = \operatorname*{arg\,min}_{Q_i \in \mathrm{O}(r)} \|U_i Q_i - \boldsymbol{U}_i\|_F^2 + \|V_i Q_i - \boldsymbol{V}_i\|_F^2 \,. \tag{4}$$

This is an instance of the well-known Orthogonal Procrustes problem [67], which has a closed form solution: Q_i can be computed from the singular value decomposition of the r-by-r matrix $U_i^\top \boldsymbol{U}_i + V_i^\top \boldsymbol{V}_i$, which can be computed efficiently when the rank r is low. After computing these Q_i's for each pair of LoRA weights, we input the $U_i Q_i, V_i Q_i$ into an MLP to compute predictions. This architecture, denoted by MLP(O-Align$([U, V])$), is fully expressive but is invariant only to the action of the orthogonal group $\mathrm{O}(r) < \mathrm{GL}(r)$, and does not capture full general linear symmetries.

Singular values as features Similarly to Salama et al. [65], we also consider an LoL architecture that feeds the singular values of the multiplied-out LoRA weights, $\sigma_1(U_i V_i^\top), \ldots, \sigma_r(U_i V_i^\top)$ into an MLP to compute predictions (though Salama et al. [65] mostly use nearest neighbor predictors instead of an MLP). Since $U_i V_i^\top$ is rank r, there are at most r nonzero singular values. This method is denoted by $\mathrm{MLP}(\sigma(UV^\top))$. The singular values, and thus the method, are GL-invariant, but they are not fully expressive. For instance, negating U_i does not affect the singular values, but can completely change the functionality and destroy the performance of the finetuned model.

3.3.2 GL-net: Constructing GL-invariant Models using Equivariant and invariant Layers

A common and effective method for parameterizing invariant neural networks is processing the input with equivariant layers and then making the final prediction with invariant modules [9, 52, 53, 5]. In this vein, we develop GL-net, which consists of a series of GL-equivariant linear layers and nonlinearities followed by a GL-invariant head to predict invariant characteristics of the input, as seen in Figure 2.

Equivariant linear layers We parameterize GL-equivariant linear layers L_{Linear} that map $(U_1, V_1, \ldots, U_L, V_L)$ to $(\tilde{U}_1, \tilde{V}_1, \ldots, \tilde{U}_L, \tilde{V}_L)$ as

$$F_{\text{Linear}}(U_1, V_1, \ldots, U_L, V_L) = (\mathbf{\Phi}_1 U_1, \mathbf{\Psi}_1 V_1, \ldots, \mathbf{\Phi}_L U_L, \mathbf{\Psi}_L V_L), \tag{5}$$

where $\mathbf{\Phi}_i \in \mathbb{R}^{n_i' \times n_i}$ and $\mathbf{\Psi}_i \in \mathbb{R}^{m_i' \times m_i}$ are learnable LoL model parameters. That is, a GL-equivariant linear layer consists of left matrix multiplying each U_i by a learnable $\mathbf{\Phi}_i$, and left matrix multiplying each V_i by a learnable $\mathbf{\Psi}_i$. In practice, we choose the same hidden dimension for every $\tilde{U}_i$ and $\tilde{V}_i$ for simplicity, so $n_1' = m_1' = \ldots = n_L' = m_L'$. It's easy to verify that F_{Linear} is GL-equivariant. For instance, if there is only one layer ($L = 1$), we have

$$F_{\text{Linear}}(R \star (U_1, V_1)) = (\mathbf{\Phi}_1 U_1 R, \mathbf{\Psi}_1 V_1 R^{-\top}) = (\tilde{U}_1 R, \tilde{V}_1 R^{-\top}) = R \star F_{\text{Linear}}(U_1, V_1), \tag{6}$$

where $(\tilde{U}_1, \tilde{V}_1) = F_{\text{Linear}}(U_1, V_1)$, and $R \star (\tilde{U}_1, \tilde{V}_1) = (\tilde{U}_1 R, \tilde{V}_1 R^{-\top})$ is the action of R on the LoRA space. In fact, we show a stronger statement – the linear maps defined by (5) constitute *all possible* GL-equivariant linear maps. See Appendix A for the proof.

Proposition 3.1. ***All*** *linear* GL*-equivariant layers can be written in the form of* (5).

Concurrently with our work, [33] develop linear probing models for learning on model finetunes, including LoRAs. While equivariant, their probing map is less expressive, and does not include GL-equivariant nonlinearities.

Extension to convolution LoRAs Low rank convolution decompositions are generally represented as two consecutive convolutions where C_B projects the input down from m to r channels and C_A projects the input up to n. For our architecture we flatten all dimensions of the convolutions except the hidden channel dimension, and then we apply equivariant linear layers.

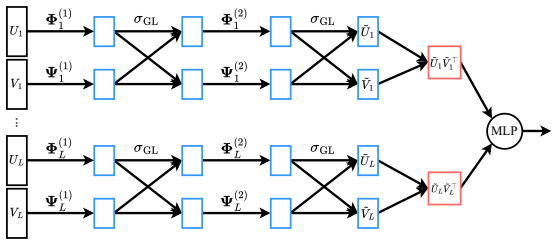
Equivariant non-linearity Equivariant networks often interleave pointwise non-linearities with linear equivariant layers. Unfortunately, non-trivial pointwise non-linearities are not equivariant to our symmetry group. A general recipe for designing equivariant non-linearities is taking $f_{\text{equi}}(\mathbf{x}) = f_{\text{inv}}(\mathbf{x}) \cdot \mathbf{x}$, for some scalar invariant function f_{inv} [74, 77, 3]. In our case, given any non-linearity $\sigma : \mathbb{R} \to \mathbb{R}$, we define a GL-equivariant non-linearity by

$$\sigma_{\text{GL}}(U)_i = \sigma\Big(\sum\nolimits_j (UV^\top)_{ij}\Big) U_i, \quad \sigma_{\text{GL}}(V)_i = \sigma\Big(\sum\nolimits_j (UV^\top)_{ji}\Big) V_i. \tag{7}$$

In other words, we scale the i-th row of U_i by a quantity that depends on the the i-th row sum of $UV^\top$, and scale the i-th row of V_i by a quantity that depends on the i-th column sum of $UV^\top$. Since the representation $UV^\top$ is GL-invariant, σ_{GL} is GL-equivariant (see proof in Appendix B). In our experiments, we often take $\sigma(x) = \text{ReLU}(\text{sign}(x))$, which has the effect of zero-ing out entire rows of U or V in a GL-equivariant way – this is a natural GL-equivariant generalization of ReLU. In our experiments we tune the number of equivariant linear layers and frequently find that only one is required, so we often do not use this equivariant non-linearity; nonetheless, it may be necessary in other applications, such as GL-equivariant tasks.

Invariant head For invariant classification tasks, we use an invariant head which consists of computing the LoRA matrix products, concatenating them, and then applying an MLP. Explicitly, this is given as $f_{\text{inv}}(U_1, V_1, \ldots, U_L, V_L) = \text{MLP}(\text{cat}[U_1 V_1^\top, \ldots, U_L V_L^\top])$, where cat concatenates the entries of the inputs into a flattened vector. For the matrices in the shape of the input, computing $U_i V_i^\top$ would be expensive in both memory and compute. To avoid this, we use our equivariant linear layers to lower the dimension of U_i, V_i to about $32 \times r$, such that $\tilde{U}_i \tilde{V}_i^\top \in \mathbb{R}^{32 \times 32}$ is efficient to compute – see Figure 2 for an illustration.

Figure 2: GL-net architecture. Blue boxes are equivariant representations, red boxes are invariant representations. Features are processed through equivariant linear maps, GL equivariant nonlinearities, and a matrix multiplication head with MLP.



3.4 Theoretical Analysis

Here, we restate the properties of our models as described in Section 3 and Table 1. All proofs are provided in the Appendix.

Theorem 3.2 (Invariance)**.** $\mathrm{MLP}(UV^\top)$*,* $\mathrm{MLP}(\sigma(UV^\top))$*, and* GL*-net are* GL*-Invariant.* $\mathrm{MLP}(\mathrm{O}\text{-}Align([U,V]))$ *is* O*-Invariant, not* GL*-Invariant.* $\mathrm{MLP}([U,V])$ *is not* GL *or* O*-Invariant.*

Theorem 3.3 (Universality)**.** $\mathrm{MLP}([U,V])$*,* $\mathrm{MLP}(\mathrm{O}\text{-}Align([U,V]))$*,* $\mathrm{MLP}(UV^\top)$*, and* GL*-net can arbitrarily approximate any* GL*-invariant continuous function on a compact set of full rank matrices.*

4 Experimental Results

In this section, we present experimental results evaluating our five LoL models across tasks involving finetuned diffusion and language models (Subsections 4.2- 4.3). Our experiments demonstrate the efficacy of LoL models in solving GL-invariant tasks on LoRAs. Additionally, we explore these models' ability to generalize to LoRAs trained for other models and for LoRAs with previously unseen ranks, with detailed findings presented in Subsection F.1.

4.1 Datasets of Trained LoRA Weights

Table 2: Summary of LoRA datasets, base models, number of LoRAs, and LoRA rank.

Dataset Name	Base Model(s)	Total # LoRAs	LoRA Rank
CelebA-LoRA	Stable Diffusion 1.4	4,900	1,2,4,8,16,32
Imagenette-LoRA	Stable Diffusion 1.4, 1.5	6,138	4, 32
Qwen2-ARC-LoRA	Qwen2-1.5B	2,000	4
Llama3.2-ARC-LoRA	Llama3.2-3B	2,000	4

We generate four new datasets of LoRAs for varying tasks and base models. Two of these datasets correspond to finetunes of diffusion models on image datasets, while the other two are finetunes of a language model on text. Our datasets have diversity in terms of LoRA rank, training hyperparameters (three datasets have randomly sampled hyperparameters, whereas one has fixed hyperparameters), and base model architecture.

4.2 Learning on Diffusion Model LoRAs

4.2.1 CLIP Score Prediction

Figure 3: Images generated by diffusion models in the test set. (a), (c), and (e) are images generated by the model with the highest CLIP score predicted by GL-net. (b), (d), and (f) are outputs of the model with the lowest CLIP score predicted by GL-net.

CLIP scores were introduced by [30] as an automated method for measuring text-image-alignment in diffusion models by evaluating the semantic similarity between their prompts and generated images. Higher CLIP scores generally correspond to better diffusion models, so CLIP score is a useful metric for evaluating these models. We calculate the CLIP score of each of our 3,900 rank 4 CelebA-LoRA diffusion models (with 33 fixed prompts) and train LoL models on the task of determining the CLIP score of a model given its finetuned weights; see Appendix D.1.1 for more details.

The results are shown in Table 7. All of the invariant LoL models can effectively predict the CLIP score of diffusion LoRAs using only their decomposed low-rank matrices. The poor performance of vanilla MLP demonstrates the importance of GL symmetries, whereas the relatively good performance of $\mathrm{MLP}(UV^\top)$ suggests "outer" permutation symmetries are less important for LoL models, confirming two of our hypotheses from Section 2. GL-net is able to calculate CLIP score significantly faster than generating images, which requires 660 score-network forward passes per finetuned

model. Other baselines, including $\mathrm{MLP}(\sigma(UV^\top))$, $\mathrm{MLP}(\text{O-Align}([U,V]))$, and $\mathrm{MLP}(UV^\top)$ are less predictive. To demonstrate the utility of this task, we generate images from the two models in our test set predicted by GL-net to have the highest and lowest CLIP scores, respectively. As shown in Figure 3, GL-net's predictions are highly correlated with the actual quality of model output.

4.2.2 Training Data Property Prediction

Table 3: Results for using LoL models to predict CelebA attributes (left) and Imagenette classes (right) of the finetuning data of diffusion models, given only the LoRA weights. The tasks are to predict (left) 5 different binary attributes of the CelebA celebrity that each LoRA was finetuned on and (right) which subset of 10 Imagenette classes appeared in each LoRA's finetuning dataset.

		CelebA Attributes		Imagenette Classes	
	LoL Model	Test Loss (↓)	Test Acc (↑)	Test Loss (↓)	Test Acc (↑)
Naive Models	$\mathrm{MLP}([U,V])$	$.554 \pm .000$	72.4 ± 0.0	$.709 \pm .004$	49.6 ± 1.3
	$\mathrm{Transformer}([U,V])$	$.586 \pm .014$	73.2 ± 0.9	$.695 \pm .001$	50.0 ± 1.3
	$\mathrm{MLP}(UV^\top)$	$.267 \pm .007$	89.1 ± 0.4	$.264 \pm .011$	88.9 ± 0.6
Efficient Invariant	$\mathrm{MLP}(\text{O-Align}([U,V]))$	$.333 \pm .008$	87.2 ± 0.5	$.278 \pm .008$	87.8 ± 0.3
	$\mathrm{MLP}(\sigma(UV^\top))$	$.509 \pm .013$	77.3 ± 1.3	$.638 \pm .013$	65.6 ± 0.6
	GL-net	$\mathbf{.232 \pm .007}$	$\mathbf{91.3 \pm 0.1}$	$\mathbf{.244 \pm .005}$	$\mathbf{90.4 \pm 0.3}$

Dataset attribute prediction On our CelebA-LoRA dataset, we train LoL models to classify attributes of the celebrity that each rank 4 LoRA was finetuned on. Due to the noisy nature of CelebA labels, we follow Lingenfelter et al. [46] and only train and test on the five CelebA attributes they determine to be least noisy. We also train and test LoL models on our Imagenette-LoRA rank 32 dataset to predict which subset of Imagenette classes each LoRA was finetuned on. Our results are in Table 3.

4.3 Learning on Language Model LoRAs

Table 4: LoL model performance on language models LoRAs. The left column is prediction of which data sources the input LoRA was finetuned on, the middle column is prediction of the validation loss for the finetuning task, and the right column is prediction of the LoRA's accuracy on the ARC-C [8] test set. All metrics are reported on the LoL task's test set (on held-out LoRAs). Higher numbers are better on all metrics. OOM refers to out of memory on a 2080 Ti GPU with 11 GB memory.

		Qwen2-ARC-LoRA			Llama3.2-ARC-LoRA	
	LoL Model	Data Memb. (Acc)	Val Loss (R^2)	ARC-C Acc (R^2)	Data Memb. (Acc)	Val Loss (R^2)
Naive Models	$\mathrm{MLP}([U,V])$	$.516 \pm .006$	$.113 \pm .059$	$.107 \pm .035$	$.522 \pm .008$	$.091 \pm .030$
	$\mathrm{Transformer}([U,V])$	$.515 \pm .000$	$.856 \pm .061$	$.630 \pm .045$	$.536 \pm .003$	$.828 \pm .120$
	$\mathrm{MLP}(UV^\top)$	$\mathbf{.625} \pm .008$	$.987 \pm .003$	$.981 \pm .002$	OOM	OOM
Efficient Invariant	$\mathrm{MLP}(\text{O-Align}([U,V]))$	$.550 \pm .016$	$.821 \pm .078$	$.965 \pm .004$	$.620 \pm .007$	$.562 \pm .103$
	$\mathrm{MLP}(\sigma(UV^\top))$	$.551 \pm .001$	$\mathbf{.999} \pm .000$	$.983 \pm .002$	$.560 \pm .008$	$\mathbf{.998} \pm .000$
	GL-net	$.605 \pm .007$	$.998 \pm .000$	$\mathbf{.987} \pm .001$	$\mathbf{.652} \pm .006$	$.995 \pm .000$

In this section, we experiment with LoL models on our Qwen2-ARC-LoRA and Llama3.2-ARC datasets of finetuned language models. For our first task, we consider a type of data membership inference task: we aim to predict whether each of the 19 data sources was used to train a given LoRA (this is a 19-label binary classification task). We also consider two performance prediction tasks: we train LoL models to predict, for each LoRA, the validation loss on the finetuning objective, and the downstream accuracy on the ARC-C test set [8]. Results are in Table 4. Our invariant LoL models are very successful at predicting the finetuning validation loss and ARC-C test accuracy of LoRA-finetuned language models, with some of them achieving .99 R^2 on finetuning validation loss regression and over .98 R^2 on ARC-C test accuracy regression. This could be useful in evaluations of finetuned models, as standard evaluations of language models can require a lot of time and resources, whereas our LoL models can evaluate these models with one quick forward pass. However, data membership inference is a harder task for the LoL models, with the best model achieving only 65.2% test accuracy in predicting which data sources were present in the finetuning dataset. Though our setup is quite different, these results could be related to work showing that model-based membership inference attacks are challenging for LLMs [14, 11].

4.4 OOD Generalization: Different Base Models and LoRA Ranks

In the previous sections, we showed that LoL models succeed at various tasks on Stable Diffusion 1.4, Qwen2, and Llama3.2 LoRAs, where each LoL model is specialized to one base model and LoRAs of a fixed rank r. Now, we discuss several experiments that show promising scalability and broad applicability of LoL models in several senses: generalizing to different base models, processing larger models, and processing LoRAs of different ranks.

Generalization to different base models from the same family To test LoL model generalization across base models, we train LoL models on Stable Diffusion v1.4 (SD1.4) LoRAs and then test them on SD1.5, and vise versa. SD versions 1.4 and 1.5 were each created by continued training of SD1.2 for hundreds of thousands of additional steps (according to the model card). Our results in Table 5 show that, without any further training, LoL models trained on classifying SD1.4 LoRAs generalize to SD1.5 LoRAs and vice versa, with GL-net's performance remaining the highest among all LoL models across all tasks.

Table 5: Results for OOD finetuning dataset membership prediction. LoL Models are trained on LoRA finetunes of Stable Diffusion v1.4 or v1.5, and then evaluated on finetunes of both versions.

		Trained on SD 1.4		Trained on SD 1.5	
	LoL Model	SD 1.4 Acc	SD 1.5 Acc	SD 1.4 Acc	SD 1.5 Acc
Naive Models	MLP($[U, V]$)	50.4 ± 1.0	50.8 ± 0.7	50.1 ± 0.4	50.4 ± 0.8
	Transformer($[U, V]$)	52.8 ± 1.3	52.3 ± 0.9	52.9 ± 1.2	52.8 ± 1.5
	MLP($UV^\top$)	80.4 ± 0.7	81.2 ± 0.7	79.4 ± 1.3	79.9 ± 1.3
Efficient Invariant	MLP(O-Align($[U, V]$))	74.9 ± 1.9	73.3 ± 1.1	72.2 ± 2.1	73.3 ± 1.5
	MLP($\sigma(UV^\top)$)	60.9 ± 0.7	61.0 ± 0.6	61.1 ± 0.6	60.8 ± 0.7
	GL-net	$\mathbf{87.8} \pm 0.4$	$\mathbf{87.6} \pm 0.5$	$\mathbf{87.5} \pm 0.4$	$\mathbf{88.5} \pm 0.2$

Generalization to a different LoRA rank For another type of out of distribution generalization, in Appendix F.1 we show that LoL models trained on LoRAs of rank 4 can generalize to LoRAs of ranks between 1 and 32. MLP($UV^\top$) and GL-net both show little to no performance degradation when tested on different ranks.

4.5 Training on Larger Models

In Appendix F.2, we show that all LoL models–except for MLP($UV^\top$)–can process LoRAs of base models up to GPT-3 scale (175B parameters) within practical runtime and memory limits. Figure 4 reports the combined preprocessing and forward-pass time over 10 epochs for different LoL models and base model sizes. All models run on a single NVIDIA 2080 Ti GPU (11GB memory). Notably, MLP($UV^\top$) runs out of memory when the base model exceeds 7B parameters.

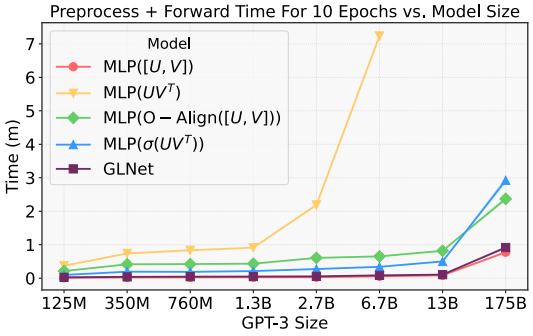


Figure 4: Timing comparison for different LoL models. Each point in the graph represents the total time (in minutes) spent on preprocessing and forward passes when training a LoL model on a dataset of 4096 LoRAs for 10 epochs. The x-axis indicates the size of the base model that the LoRAs were finetuned from, and the y-axis shows the corresponding total runtime.

5 Conclusion

In this work, we introduced the Learning on LoRAs (LoL) framework. We developed different LoL architectures, established theoretical foundations, and demonstrated LoL's practical applications. A key finding is that GL-invariance plays a crucial role in enabling effective learning over low-rank weight spaces. There are many potential applications and future directions for using LoL models to process finetunes of large models. For instance, future work could explore equivariant tasks, such as those that involve editing or merging LoRAs.

References

[1] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

[2] Bruno Andreis, Soro Bedionita, and Sung Ju Hwang. Set-based neural network encoding. *arXiv preprint arXiv:2305.16625*, 2023.

[3] Ben Blum-Smith and Soledad Villar. Machine learning and invariant theory. *Notices of the American Mathematical Society*, 2022.

[4] Alexander Bogatskiy, Brandon M. Anderson, Jan T. Offermann, Marwah Roussi, David W. Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, 2020. URL https://api.semanticscholar.org/CorpusID:219531086.

[5] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[7] Civitai, 2024. URL https://civitai.com/.

[8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[9] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[10] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[11] Debeshee Das, Jie Zhang, and Florian Tramèr. Blind baselines beat membership inference attacks for foundation models. *arXiv preprint arXiv:2406.16201*, 2024.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Amil Dravid, Yossi Gandelsman, Kuan-Chieh Wang, Rameen Abdal, Gordon Wetzstein, Alexei A Efros, and Kfir Aberman. Interpreting the weight space of customized diffusion models. *arXiv preprint arXiv:2406.09413*, 2024.

[14] Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841*, 2024.

[15] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[16] Nadav Dym and Steven J Gortler. Low-dimensional invariant embeddings for universal geometric learning. *Foundations of Computational Mathematics*, pages 1–41, 2024.

[17] Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. *arXiv preprint arXiv:2010.02449*, 2020.

[18] Nadav Dym, Hannah Lawrence, and Jonathan W. Siegel. Equivariant frames and the impossibility of continuous canonicalization. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=4iy0q0carb.

[19] Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022.

[20] Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: dissecting the weight space of neural networks. In *ECAI 2020*, pages 1119–1126. IOS Press, 2020.

[21] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

[22] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[23] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International conference on machine learning*, pages 3318–3328. PMLR, 2021.

[24] William Fulton and Joe Harris. *Representation Theory: A First Course*, volume 129 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1991. ISBN 978-0-387-97495-8.

[25] Yoav Gelberg, Yam Eitan, Aviv Navon, Aviv Shamsian, Theo Putterman, and Haggai Maron. Gradmetanet: An equivariant architecture for learning on gradients. In *Workshop on Neural Network Weights as a New Data Modality*, 2025. URL https://openreview.net/forum?id=vLq615Cpmn.

[26] Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *Advances in Neural Information Processing Systems*, 35: 22911–22924, 2022.

[27] Daniel Han and Michael Han. Unsloth, 2023.

[28] Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

[29] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=NEv8YqBROO.

[30] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning, 2022. URL https://arxiv.org/abs/2104.08718.

[31] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[32] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[33] Eliahu Horwitz, Bar Cavia, Jonathan Kahana, and Yedid Hoshen. Learning on model weights using tree experts. 2024. URL https://arxiv.org/abs/2410.13569.

[34] Jeremy Howard. Imagenette dataset, 2019.

[35] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

[36] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

[37] Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *International Conference on Machine Learning*, pages 15546–15566. PMLR, 2023.

[38] Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. *arXiv preprint arXiv:2406.10685*, 2024.

[39] Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J Burghouts, Efstratios Gavves, Cees GM Snoek, and David W Zhang. Graph neural networks for learning equivariant representations of neural networks. *arXiv preprint arXiv:2403.12143*, 2024.

[40] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International conference on machine learning*, pages 2747–2755. PMLR, 2018.

[41] Hannah Lawrence and Mitchell Tong Harris. Learning polynomial problems with $SL(2, \mathbb{R})$-equivariance. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gyfXuRfxW2.

[42] Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

[43] Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures, 2023. URL https://arxiv.org/abs/2312.04501.

[44] Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[45] Derek Lim, Moe Putterman, Robin Walters, Haggai Maron, and Stefanie Jegelka. The empirical impact of neural parameter symmetries, or lack thereof, 2024. URL https://arxiv.org/abs/2405.20231.

[46] Bryson Lingenfelter, Sara R Davis, and Emily M Hand. A quantitative analysis of labeling issues in the celeba dataset. In *International Symposium on Visual Computing*, pages 129–141. Springer, 2022.

[47] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=3d5CIRG1n2.

[48] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[49] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[50] George Ma, Yifei Wang, Derek Lim, Stefanie Jegelka, and Yisen Wang. A canonization perspective on invariant and equivariant learning. *arXiv preprint arXiv:2405.18378*, 2024.

[51] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

[52] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Syx72jC9tm.

[53] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pages 4363–4371. PMLR, 2019.

[54] Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. In *International conference on machine learning*, pages 6734–6744. PMLR, 2020.

[55] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024.

[56] Luke Metz, James Harrison, C Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, et al. Velo: Training versatile learned optimizers by scaling up. *arXiv preprint arXiv:2211.09760*, 2022.

[57] Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces, 2023. URL https://arxiv.org/abs/2301.12780.

[58] Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment. *arXiv preprint arXiv:2310.13397*, 2023.

[59] Edward Pearce-Crump. Brauer's group equivariant neural networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 27461–27482. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/pearce-crump23a.html.

[60] William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A Efros, and Jitendra Malik. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2022.

[61] Pierluigi Zama Ramirez, Luca De Luigi, Daniele Sirocchi, Adriano Cardace, Riccardo Spezialetti, Francesco Ballerini, Samuele Salti, and Luigi Di Stefano. Deep learning on object-centric 3d neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[62] Replicate, 2024. URL https://replicate.com/docs/get-started/fine-tune-with-flux.

[63] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

[64] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.

[65] Mohammad Salama, Jonathan Kahana, Eliahu Horwitz, and Yedid Hoshen. Dataset size recovery from lora weights, 2024. URL https://arxiv.org/abs/2406.19395.

[66] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks, 2022. URL https://arxiv.org/abs/2102.09844.

[67] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[68] Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. *Advances in Neural Information Processing Systems*, 34:16481–16493, 2021.

[69] Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. *Advances in Neural Information Processing Systems*, 35:27906–27920, 2022.

[70] Konstantin Schürholt, Diyar Taskiran, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Model zoos: A dataset of diverse populations of neural network models. *Advances in Neural Information Processing Systems*, 35:38134–38148, 2022.

[71] Konstantin Schürholt, Michael W Mahoney, and Damian Borth. Towards scalable and versatile weight space learning. *arXiv preprint arXiv:2406.09997*, 2024.

[72] Aviv Shamsian, Aviv Navon, David W Zhang, Yan Zhang, Ethan Fetaya, Gal Chechik, and Haggai Maron. Improved generalization of weight space networks via augmentations. *arXiv preprint arXiv:2402.04081*, 2024.

[73] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[74] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018. URL https://arxiv.org/abs/1802.08219.

[75] Hoang V Tran, Thieu N Vo, Tho H Tran, An T Nguyen, and Tan Minh Nguyen. Monomial matrix group equivariant neural functional networks. *arXiv preprint arXiv:2409.11697*, 2024.

[76] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020.

[77] Soledad Villar, David W Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems*, 34:28848–28863, 2021.

[78] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022. URL `https://arxiv.org/abs/2203.05482`.

[79] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[80] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation, 2022. URL `https://arxiv.org/abs/2206.10789`.

[81] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[82] Allan Zhou, Chelsea Finn, and James Harrison. Universal neural functionals. *arXiv preprint arXiv:2402.05232*, 2024.

[83] Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. *Advances in neural information processing systems*, 36, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In Sections B and 4, we include theoretical proofs, a description of the methods and experimental setup, and experimental results to support the claims.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Appendix G, we detail empirical and theoretical limitations of our work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We give the full set of assumptions and proofs for each result in Appendix B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We give high-level experimental setup information in the main paper (Section 4), and more detailed information about experiments in the Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We have published our code here, but have not yet published the data we use. We commit to publishing the data once anonymity constraints are no longer relevant.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We describe high-level experimental setup in Section 4, and fine-grained experimental details in Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We generally do several runs of experiments, and report error bars that measure the standard deviation in the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include details on compute in F.2 and D.5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and complied with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes, we discuss both potential positive and negative societal impacts in Appendix H.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not train models or use data that have a high risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit code and data used throughout the paper. For other types of models, we cite papers in which they were introduced.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have released our code, which in turn provides source code for generating the datasets which we use.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing or human subjects were used.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No crowdsourcing or human subjects were used.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We did not use LLMs for non-standard purposes.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A GL Equivariant Linear Maps Characterization

Here, we characterize the form of GL-equivariant linear maps, and provide the proof of Proposition 3.1. We use basic representation theory techniques, which are similar to the ones used to characterize equivariant linear maps in the geometric deep learning literature [52, 23, 57].

A.1 Proof of Proposition 3.1

First, we prove a lemma, that allows us to analyze equivariant linear maps between direct sums of spaces in terms of a direct sum of equivariant linear maps between the constituent spaces. For a group G, we denote the vector space of G-equivariant linear maps from $\mathcal{V}$ to $\mathcal{W}$ by $\mathrm{Hom}_G(\mathcal{V}, \mathcal{W})$. This is closely related to a result from Navon et al. [58] that characterizes the matrices underlying linear maps between direct sums.

Lemma A.1. *Let G be a group and let $\mathcal{V}_1, \dots, \mathcal{V}_n$, $\mathcal{W}_1, \dots, \mathcal{W}_m$ be G-representations. If $\mathcal{V} = \mathcal{V}_1 \oplus \cdots \oplus \mathcal{V}_n$ and $\mathcal{W} = \mathcal{W}_1 \oplus \cdots \oplus \mathcal{W}_m$ are direct sums of representations then*

$$\mathrm{Hom}_G(\mathcal{V}, \mathcal{W}) \cong \bigoplus_{i,j} \mathrm{Hom}_G(\mathcal{V}_i, \mathcal{W}_j). \tag{8}$$

Proof. We construct an explicit isomorphism $F : \mathrm{Hom}_G(\mathcal{V}, \mathcal{W}) \to \bigoplus_{i,j} \mathrm{Hom}_G(\mathcal{V}_i, \mathcal{W}_j)$. Let $\mathrm{inc}_i : \mathcal{V}_i \to \mathcal{V}$ denote the inclusion of $\mathcal{V}_i$ in $\mathcal{V}$, and $\mathrm{proj}_j : \mathcal{W} \to \mathcal{W}_j$ denote the projection from $\mathcal{W}$ to $\mathcal{W}_j$. For $\phi \in \mathrm{Hom}_G(\mathcal{V}, \mathcal{W})$, define:

$$F(\phi) = (\mathrm{proj}_j \circ \phi \circ \mathrm{inc}_i)_{i,j} \tag{9}$$

F is clearly linear, being defined by composition of linear maps. Moreover, since ϕ, inc_i, and proj_j are all G-equivariant, their composition $\phi_{i,j} = \mathrm{proj}_j \circ \phi \circ \mathrm{inc}_i$ is also G-equivariant. To prove injectivity, suppose $F(\phi) = F(\psi)$ for some $\phi, \psi \in \mathrm{Hom}_G(\mathcal{V}, \mathcal{W})$. Then $\forall v = (v_1, \dots, v_n) \in \mathcal{V}$

$$\begin{aligned}
\phi(v) &= (\mathrm{proj}_1(\phi(v)), \dots, \mathrm{proj}_m(\phi(v))) \\
&= \left(\sum_i \mathrm{proj}_1(\phi(\mathrm{inc}_i(v_i))), \dots, \sum_i \mathrm{proj}_m(\phi(\mathrm{inc}_i(v_i))) \right) \\
&= \left(\sum_i \mathrm{proj}_1(\psi(\mathrm{inc}_i(v_i))), \dots, \sum_i \mathrm{proj}_m(\psi(\mathrm{inc}_i(v_i))) \right) \\
&= \psi(v).
\end{aligned}$$

For surjectivity, given $(\phi_{i,j})_{i,j} \in \bigoplus_{i,j} \mathrm{Hom}_G(\mathcal{V}_i, \mathcal{W}_j)$, define $\phi : \mathcal{V} \to \mathcal{W}$ by

$$\phi(v_1, \dots, v_n) = \left(\sum_i \phi_{i,1}(v_i), \dots, \sum_i \phi_{i,m}(v_i) \right). \tag{10}$$

ϕ is linear by construction; to show G-equivariance, let $g \in G$ and $(v_1, \dots, v_n) \in \mathcal{V}$

$$\begin{aligned}
\phi(g \cdot (v_1, \dots, v_n)) &= \phi(g \cdot v_1, \dots, g \cdot v_n) \\
&= \left(\sum_i \phi_{i,1}(g \cdot v_i), \dots, \sum_i \phi_{i,m}(g \cdot v_i) \right) \\
&= \left(\sum_i g \cdot \phi_{i,1}(v_i), \dots, \sum_i g \cdot \phi_{i,m}(v_i) \right) \\
&= g \cdot \phi(v_1, \dots, v_n),
\end{aligned}$$

where we use the G-equivariance of each $\phi_{i,j}$. Thus, $\phi \in \mathrm{Hom}_G(\mathcal{V}, \mathcal{W})$, and by construction $F(\phi) = (\phi_{i,j})_{i,j}$. Therefore, F is a linear bijection. □

Proof of Proposition 3.1. Let $\mathcal{I} := (\mathcal{U}_1 \oplus \mathcal{V}_1^*) \oplus \cdots \oplus (\mathcal{U}_L \oplus \mathcal{V}_L^*)$ and $\mathcal{O} := (\tilde{\mathcal{U}}_1 \oplus \tilde{\mathcal{V}}_1^*) \oplus \cdots \oplus (\tilde{\mathcal{U}}_L \oplus \tilde{\mathcal{V}}_L^*)$ be the input and output spaces of an equivariant LoL model. Denote $\dim(\mathcal{U}_i) = n_i \cdot r$, $\dim(\mathcal{V}_i^*) =$

$r \cdot m_i$, $\dim(\tilde{\mathcal{U}}_i) = n_i' \cdot r$, $\dim(\tilde{\mathcal{V}}_i^*) = r \cdot m_i'$, and $G := \mathrm{GL}(r)^L = \mathrm{GL}(r) \times \cdots \times \mathrm{GL}(r)$. We are interested in characterizing the vector space of G-equivariant linear maps, which we denote by $\mathrm{Hom}_G(\mathcal{I}, \mathcal{O})$. In the main text we show that the maps $F_{\text{Linear}}^{\mathbf{\Phi}_1, \mathbf{\Psi}_1, \ldots, \mathbf{\Phi}_L, \mathbf{\Psi}_L}$, defined by

$$F_{\text{Linear}}^{\mathbf{\Phi}_1, \mathbf{\Psi}_1, \ldots, \mathbf{\Phi}_L, \mathbf{\Psi}_L}(U_1, V_1, \ldots, U_L, V_L) = (\mathbf{\Phi}_1 U_1, \mathbf{\Psi}_1 V_1, \ldots, \mathbf{\Phi}_L U_L, \mathbf{\Psi}_L V_L) \tag{11}$$

are all G-equivariant. In other words, we showed that

$$\mathcal{L} := \left\{ F_{\text{Linear}}^{\mathbf{\Phi}_1, \mathbf{\Psi}_1, \ldots, \mathbf{\Phi}_L, \mathbf{\Psi}_L} \mid \mathbf{\Phi}_i \in \mathbb{R}^{n_i' \times n_i}, \mathbf{\Psi}_i \in \mathbb{R}^{m_i' \times m_i} \right\} \subseteq \mathrm{Hom}_G(\mathcal{I}, \mathcal{O}). \tag{12}$$

$\mathcal{L}$ is a linear subspace of dimension $\sum_{i=1}^L n_i n_i' + \sum_{i=1}^L m_i m_i'$, so in order to prove that $\mathcal{L} = \mathrm{Hom}_G(\mathcal{I}, \mathcal{O})$, it's enough to show that $\dim(\mathrm{Hom}_G(\mathcal{I}, \mathcal{O})) = \sum_{i=1}^L n_i n_i' + \sum_{i=1}^L m_i m_i'$. Since $\mathcal{I}$ and $\mathcal{O}$ are direct sums of representations, Lemma A.1 implies that the dimension of $\mathrm{Hom}_G(\mathcal{I}, \mathcal{O})$ is the sum of the dimensions of the constituents:

$$\begin{aligned} \dim(\mathrm{Hom}_G(\mathcal{I}, \mathcal{O})) = \sum_{i=1}^{L} \sum_{i'=1}^{L} \Bigg(& \dim\left(\mathrm{Hom}_G(\mathcal{U}_i, \tilde{\mathcal{U}}_{i'})\right) + \dim\left(\mathrm{Hom}_G(\mathcal{U}_i, \tilde{\mathcal{V}}_{i'}^*)\right) + \\ & \dim\left(\mathrm{Hom}_G(\mathcal{V}_i^*, \tilde{\mathcal{U}}_{i'})\right) + \dim\left(\mathrm{Hom}_G(\mathcal{V}_i^*, \tilde{\mathcal{V}}_{i'}^*)\right) \Bigg). \end{aligned} \tag{13}$$

Next, note that $\mathcal{U}_i$, $\mathcal{V}_i^*$, $\tilde{\mathcal{U}}_i$, and $\tilde{\mathcal{V}}_i^*$ all decompose into irreducible representations

$$\mathcal{U}_i = \bigoplus_{j=1}^{n_i} \mathcal{U}_i^j, \; \mathcal{V}_i^* = \bigoplus_{j=1}^{m_i} (\mathcal{V}_i^j)^*, \; \tilde{\mathcal{U}}_i = \bigoplus_{j=1}^{n_i'} \tilde{\mathcal{U}}_i^j, \; \tilde{\mathcal{V}}_i^* = \bigoplus_{j=1}^{m_i'} (\tilde{\mathcal{V}}_i^j)^*, \tag{14}$$

each of which is isomorphic to the standard representation of the i-th copy of $\mathrm{GL}(r)$ on either $\mathbb{R}^r$ (for $\mathcal{U}_i^j$ and $\tilde{\mathcal{U}}_i^j$) or $(\mathbb{R}^r)^*$ (for $(\mathcal{V}_i^j)^*$ and $(\tilde{\mathcal{V}}_i^j)^*$). We can think of $\mathcal{U}_i^j$ as the space spanned by the j-th row in the input U_i matrix and $(\mathcal{V}_i^j)^*$ as the space spanned by the j-th column of the input $V_i^\top$ matrix. This decomposition gives us

$$\begin{aligned} \dim\left(\mathrm{Hom}_G(\mathcal{U}_i, \tilde{\mathcal{U}}_{i'})\right) &= \sum_{j=1}^{n_i} \sum_{j'=1}^{n_i'} \dim\left(\mathrm{Hom}_G(\mathcal{U}_i^j, \tilde{\mathcal{U}}_{i'}^{j'})\right), \\ \dim\left(\mathrm{Hom}_G(\mathcal{U}_i, \tilde{\mathcal{V}}_{i'}^*)\right) &= \sum_{j=1}^{n_i} \sum_{j'=1}^{m_i'} \dim\left(\mathrm{Hom}_G(\mathcal{U}_i^j, (\tilde{\mathcal{V}}_{i'}^{j'})^*)\right), \\ \dim\left(\mathrm{Hom}_G(\mathcal{V}_i^*, \tilde{\mathcal{U}}_{i'})\right) &= \sum_{j=1}^{m_i} \sum_{j'=1}^{n_i'} \dim\left(\mathrm{Hom}_G((\mathcal{V}_i^j)^*, \tilde{\mathcal{U}}_{i'}^{j'})\right), \\ \dim\left(\mathrm{Hom}_G(\mathcal{V}_i^*, \tilde{\mathcal{V}}_{i'}^*)\right) &= \sum_{j=1}^{m_i} \sum_{j'=1}^{m_i'} \dim\left(\mathrm{Hom}_G((\mathcal{V}_i^j)^*, (\tilde{\mathcal{V}}_{i'}^{j'})^*)\right). \end{aligned} \tag{15}$$

Since these are all irreducible representations, Schur's lemma [24] implies that the dimension of the space of G-equivariant maps is either 1 (if the representations are isomorphic) or 0 (if they are not). Notice that,

- $\mathcal{U}_i^j$ and $\tilde{\mathcal{U}}_{i'}^{j'}$ are isomorphic as G-representations if and only if $i = i'$ (if $i \neq i'$ different copies of $\mathrm{GL}(r)$ act on $\mathcal{U}_i^j$ and $\tilde{\mathcal{U}}_{i'}^{j'}$).
- $(\mathcal{V}_i^j)^*$ and $(\tilde{\mathcal{V}}_{i'}^{j'})^*$ are isomorphic as G-representations if and only if $i = i'$ (if $i \neq i'$ different copies of $\mathrm{GL}(r)$ act on $(\mathcal{V}_i^j)^*$ and $(\tilde{\mathcal{V}}_{i'}^{j'})^*$).
- $\mathcal{U}_i^j$ is never isomorphic to $(\tilde{\mathcal{V}}_{i'}^{j'})^*$ and $(\mathcal{V}_i^j)^*$ is never isomorphic to $\tilde{\mathcal{U}}_{i'}^{j'}$.

Therefore, together with Equation 13 and Equation 15 we get

$$\begin{aligned}\dim\left(\mathrm{Hom}_G(\mathcal{I},\mathcal{O})\right) &= \sum_{i=1}^{L}\sum_{i'=1}^{L}\left(n_i n'_{i'}\cdot \mathbf{1}_{i=i'} + m_i m'_{i'}\cdot \mathbf{1}_{i=i'}\right)\\ &= \sum_{i=1}^{L} n_i n'_i + \sum_{i=1}^{L} m_i m'_i\end{aligned} \tag{16}$$

concluding the proof. □

B Proof of Invariances: Theorem 3.2

We prove invariance (or lack thereof) of each model one by one. For this section, consider arbitrary inputs $\mathbf{UV} = (U_1, V_1, \ldots, U_L, V_l)$, where $U_i \in \mathbb{R}^{n_i \times r}$ and $V_i \times \mathbb{R}^{m_i \times r}$. Further, choose invertible $\mathbf{R} = (R_1, \ldots, R_L) \in \mathrm{GL}(r)^L$. Denote the application of $\mathbf{R}$ on $\mathbf{UV}$ by $\mathbf{R} \star \mathbf{UV} = (U_1 R_1, V_1 R_1^{-\top}, \ldots, U_L R_L, V_L R_L^{-\top})$ To show that a function f is GL invariant, we will show that $f(\mathbf{R} \star \mathbf{UV}) = f(\mathbf{UV})$. Note that, since $\mathrm{O}(r) \subset \mathrm{GL}(r)$, if we show that a function is GL-invariant, then it is also O-invariant.

$\mathrm{MLP}([U, V])$**.** We will show that the simple MLP is neither O-invariant or GL-invariant. It suffices to show that it is not O-invariant. As a simple example, let $\mathrm{MLP}(U, V) = U_{1,1}$ output the top-left entry of U. Then let U be a matrix with 1 in the top-left corner and 0 elsewhere. Further, let P be the permutation matrix that swaps the first and second entries of its input. Then $\mathrm{MLP}(UP, VP) = 0 \neq \mathrm{MLP}(U, V) = 1$. As permutation matrices are orthogonal, $P \in \mathrm{O}(r)$, so the $\mathrm{MLP}([U, V])$ is indeed not $\mathrm{O}(r)$ invariant.

$\mathrm{MLP}($O-**Align**$([U, V]))$**.** We will show that the O-alignment approach is O-invariant on all but a Lebesgue-measure-zero set. For simplicity, let $L = 1$, $U \in \mathbb{R}^{n \times r}$, and $V \in \mathbb{R}^{m \times r}$. Let $\mathbf{U} \in \mathbb{R}^{n \times r}$ and $\mathbf{V} \in \mathbb{R}^{m \times r}$ be the template matrices, which we assume are full rank (the full rank matrices are a Lebesgue-dense set, so this is an allowed assumption). Recall that we canonicalize U, V as $\rho(U, V) = (UQ, VQ)$, where:

$$Q = \underset{Q \in O(r)}{\arg\min} \|UQ - \mathbf{U}\|_F^2 + \|VQ - \mathbf{V}\|_F^2 . \tag{17}$$

We call any solution to this problem a *canonicalizing matrix* for (U, V). This can be equivalently written as

$$Q = \underset{Q \in O(r)}{\arg\min} \left\| \begin{bmatrix} U \\ V \end{bmatrix} Q - \begin{bmatrix} \mathbf{U} \\ \mathbf{V} \end{bmatrix} \right\|_F^2 . \tag{18}$$

Let $M = U^\top \mathbf{U} + V^\top \mathbf{V}$. Then a global minimum of this problem is $Q = AB^\top$, where $A\Sigma B^\top$ is an SVD of M. If M has distinct singular values, then A and B are unique up to sign flips, and $AB^\top$ is in fact unique. Thus, if M has unique singular values, (UQ, VQ) is unique. We assume from here that M has distinct singular values, as the set of all M that do form a Lebesgue-dense subset of $\mathbb{R}^{r \times r}$ (and thus this is satisfies for Lebesgue-almost-every U and V).

Now, to show O-invariance, let $\tilde{Q} \in \mathrm{O}(r)$. We will show that $\rho(U\tilde{Q}, V\tilde{Q}) = \rho(U, V)$.

$$\underset{Q \in O(r)}{\arg\min} \left\| \begin{bmatrix} U\tilde{Q} \\ V\tilde{Q} \end{bmatrix} Q - \begin{bmatrix} \mathbf{U} \\ \mathbf{V} \end{bmatrix} \right\|_F^2 = \underset{Q' \in O(r)}{\arg\min} \left\| \begin{bmatrix} U \\ V \end{bmatrix} Q' - \begin{bmatrix} \mathbf{U} \\ \mathbf{V} \end{bmatrix} \right\|_F^2 \tag{19}$$

because the orthogonal matrices are closed under multiplication. Thus, if Q is a canonicalizing matrix for (U, V), then $\tilde{Q}^\top Q$ is a canonicalizing matrix for $(U\tilde{Q}, V\tilde{Q})$. Moreover, we have that $\tilde{Q}^\top U^\top \mathbf{U} + \tilde{Q}^\top V^\top \mathbf{V} = \tilde{Q}^\top M$, so this has the same singular values as M (as orthogonal matrices don't affect singular values); this means that the singular values are distinct, so there is a unique canonicalizing matrix for $(U\tilde{Q}, V\tilde{Q})$. This means that the canonicalization matrix must be equal to $\tilde{Q}^\top Q$, so that

$$\rho(U\tilde{Q}, V\tilde{Q}) = (U\tilde{Q}\tilde{Q}^\top Q, V\tilde{Q}\tilde{Q}^\top Q) = (UQ, VQ) = \rho(U, V). \tag{20}$$

As this argument holds for Lebesgue-almost-every U and V, we have shown O-invariance of $\mathrm{MLP}(\text{O-Align}([U, V]))$.

Finally, we have to show that this LoL model is not GL-invariant. To do this, let the MLP approximate the Frobenius norm function on its first input, so $\mathrm{MLP}(U, V) \approx \|U\|$. Consider any U that is nonzero, and let $a > 2$ be a scalar. Then $aI \in \mathrm{GL}(r)$. Also, we have that $\rho(U, V) = (UQ, VQ)$ for some $Q \in \mathrm{O}(r)$, so this canonicalization does not affect the Frobenius norm of the first entry. Thus, we have that

$$\mathrm{MLP}(\rho(U, V)) \approx \|U\| \neq a \|U\| \approx \mathrm{MLP}(\rho(aU, (1/a)V)). \tag{21}$$

So $\mathrm{MLP}(\text{O-Align}([U, V]))$ is not invariant under GL.

$\mathrm{MLP}(\sigma(UV^\top))$. We show this is GL-invariant. The output of this model f on $\mathbf{UV}$ can be written as

$$\mathrm{MLP}(\sigma_1(U_1V_1^\top), \ldots, \sigma_r(U_1V_1^\top), \ldots, \sigma_1(U_LV_L^\top), \ldots, \sigma_r(U_LV_L^\top). \tag{22}$$

Where $\sigma_j(U_iV_i^\top)$ is the jth singular value of $U_iV_i^\top$. Thus, we can compute that:

$$\begin{aligned} f(\mathbf{R} \star \mathbf{UV}) &= \mathrm{MLP}(\sigma_1(U_1R_1R_1^{-1}V_1^\top), \ldots, \sigma_r(U_LR_LR_L^{-1}V_L^\top)) && (23)\\ &= \mathrm{MLP}(\sigma_1(U_1V_1^\top), \ldots, \sigma_r(U_LV_L^\top)) && (24)\\ &= f(\mathbf{UV}). && (25) \end{aligned}$$

$\mathrm{MLP}(UV^\top)$. We show this is GL-invariant. We can simply see that

$$\begin{aligned} \mathrm{MLP}(\mathbf{R} \star \mathbf{UV}) &= \mathrm{MLP}(U_1R_1R_1^{-1}V_1^\top, \ldots, U_LR_LR_L^{-1}V_L^\top) && (26)\\ &= \mathrm{MLP}(U_1V_1^\top, \ldots, U_LV_L^\top) && (27)\\ &= \mathrm{MLP}(\mathbf{UV}). && (28) \end{aligned}$$

GL-net. We show this is GL-invariant. First, assume that the equivariant linear layers are GL-equivariant, and the equivariant nonlinearities are GL-equivariant. Note that the invariant head of GL-net is a special case of $\mathrm{MLP}(UV^\top)$, which we have already proven to be invariant. Further, an invariant function composed with an equivariant function is invariant, so we are done.

We only need to prove that the nonlinearities are GL-equivariant, because we have already proven that the equivariant linear layers are GL-equivariant in the main text. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be any real function, and recall that the GL-equivariant nonlinearity takes the following form on $U \in \mathbb{R}^{n\times r}$, $V \in \mathbb{R}^{m\times r}$:

$$\sigma_{\mathrm{GL}}(U, V) = (\tilde{U}, \tilde{V}), \qquad \tilde{U}_i = \sigma\Big(\sum\nolimits_j (UV^\top)_{ij}\Big)U_i, \qquad \tilde{V}_i = \sigma\Big(\sum\nolimits_j (UV^\top)_{ji}\Big)V_i, \tag{29}$$

where $\tilde{U} \in \mathbb{R}^{n\times r}$, $\tilde{V} \in \mathbb{R}^{m\times r}$, and for example $U_i \in \mathbb{R}^r$ denotes the ith row of U_i.

Lemma B.1. *For any real function $\sigma : \mathbb{R} \to \mathbb{R}$, the function σ_{GL} defined in* (29) *is* GL *equivariant.*

Proof. Let $U \in \mathbb{R}^{n\times r}, V \in \mathbb{R}^{m\times r}$, and $R \in \mathrm{GL}(r)$ be arbitrary. Denote the output of the nonlinearity on the transformed weights as $\sigma_{\mathrm{GL}}(UR, VR^{-\top}) = (\tilde{U}^{(R)}, \tilde{V}^{(R)})$. Then we have that

$$\tilde{U}_i^{(R)} = \sigma\Big(\sum_j (URR^{-1}V^\top)_{ij}\Big)R^\top U_i = R^\top \left[\sigma\Big(\sum_j (UV^\top)_{ij}\Big)U_i\right] = R^\top U_i \tag{30}$$

$$\tilde{V}_i^{(R)} = \sigma\Big(\sum_j (URR^{-1}V^\top)_{ji}\Big)R^{-1}V_i = R^{-1} \left[\sigma\Big(\sum_j (UV^\top)_{ji}\Big)V_i\right] = R^{-1}V_i. \tag{31}$$

In matrix form, this means that $\tilde{U}_i^{(R)} = \tilde{U}_iR$ and $\tilde{V}_i^{(R)} = \tilde{V}_iR^{-\top}$. In other words,

$$\sigma_{\mathrm{GL}}(UR, VR^{-1}) = R \star \sigma_{\mathrm{GL}}(U, V), \tag{32}$$

which is the definition of GL-equivariance, so we are done. □

C Proof of Expressivity: Theorem 3.3

In this section we formally restate and prove Theorem 3.3. We start by defining full rank GL-universality for LoL models.

Definition C.1 (Full rank GL-universality)**.** Let $\mathcal{D} = \{(U_1, V_1, \dots, U_L, V_L) \mid U_i \in \mathbb{R}^{n_i \times r}, V_i \in \mathbb{R}^{m_i}, \operatorname{rank}(U_i) = \operatorname{rank}(V_i) = r\}$ be the set of LoRA updates of full rank. A LoL architecture is called full rank GL-universal if for every GL-invariant function $f : \mathcal{D} \to \mathbb{R}$, every $\epsilon > 0$, and every compact set $K \subset \mathcal{D}$, there is a model f^{LoL} of said architecture that approximates f on K up to ϵ:

$$\sup_{\boldsymbol{X} \in K} |f^{\mathrm{LoL}}(\boldsymbol{X}) - f(\boldsymbol{X})| < \epsilon. \tag{33}$$

Note that the set $\mathcal{D}$ of full-rank LoRA updates is Lebesgue-dense (its compliment has measure 0).

Theorem C.2 (Formal restatement of Theorem 3.3)**.** *The MLP,* $\mathrm{MLP}(\text{O-}\textit{Align}([U, V]))$*,* $\mathrm{MLP}(UV^\top)$*, and* GL*-net LoL architectures are all full rank* GL *universal.*

The universality of MLP and $\mathrm{MLP}(\text{O-Align}([U, V]))$ models follows from the universal approximation theorem for MLPs [32]. To prove Theorem C.2 for $\mathrm{MLP}(UV^\top)$ and GL-net we use the following result from Dym and Gortler [16].

Proposition C.3 (Proposition 1.3 from Dym and Gortler [16])**.** *Let $\mathcal{M}$ be a topological space, and G a group which acts on $\mathcal{M}$. Let $K \subset \mathcal{M}$ be a compact set, and let $f^{\mathrm{inv}} : \mathcal{M} \to \mathbb{R}^N$ be a continuous G-invariant map that separates orbits. Then for every continuous invariant function $f : \mathcal{M} \to \mathbb{R}$ there exists some continuous $f^{\mathrm{general}} : \mathbb{R}^N \to \mathbb{R}$ such that $f(x) = f^{\mathrm{general}}(f^{\mathrm{inv}}(x))$, $\forall x \in K$.*

In order to use the proposition above for $\mathrm{MLP}(UV^\top)$ LoL models, we need to show that multiplying out the LoRA updates separates GL orbits.

Lemma C.4. *The function*

$$f^{\mathrm{mul}}(U_1, V_1, \dots, U_L, V_L) = (U_1 V_1^\top, \dots, U_L V_L^\top),$$

separates GL*-orbits in $\mathcal{D}$. That is, if $(U_1, V_1, \dots, U_L, V_L)$ and $(U_1', V_1', \dots, U_L', V_L')$ are in different G-orbits then $f^{\mathrm{mul}}(U_1, V_1, \dots, U_L, V_L) \neq f^{\mathrm{mul}}(U_1', V_1', \dots, U_L', V_L')$.*

Proof. We prove the contrapositive. Let $(U_1, V_1, \dots, U_L, V_L)$ and $(U_1', V_1', \dots, U_L', V_L')$ be LoRA updates such that $f^{\mathrm{mul}}(U_1, V_1, \dots, U_L, V_L) = f^{\mathrm{mul}}(U_1', V_1', \dots, U_L', V_L')$, i.e. $\forall i \in \{1, \dots, L\}$

$$U_i V_i^\top = U_i' V_i'^\top. \tag{34}$$

Since U_i, V_i, U_i', and V_i' are of rank r, their corresponding Gram matrices $U_i^\top U_i$, $V_i^\top V_i$, $U_i'^\top U_i', V_i'^\top V_i' \in \mathbb{R}^{r \times r}$, are also of rank r and are thus invertible. Multiplying both sides of Equation 34 by $V_i (V_i^\top V_i)^{-1}$ from the right, we get

$$U_i \cancel{V_i^\top V_i (V_i^\top V_i)^{-1}} = U_i' \underbrace{V_i'^\top V_i (V_i^\top V_i)^{-1}}_{R_i}. \tag{35}$$

Substituting $U_i' R_i$ back to Equation 34 we get

$$U_i' R_i V_i^\top = U_i' V_i'^\top.$$

Multiplying by $(U_i'^\top U_i')^{-1} U_i'^\top$ from the left gives

$$\cancel{(U_i'^\top U_i')^{-1} U_i'^\top U_i'} R_i V_i^\top = \cancel{(U_i'^\top U_i')^{-1} U_i'^\top U_i'} V_i'^\top.$$

Therefore, to prove $(U_1, V_1, \dots, U_L, V_L)$ and $(U_1', V_1', \dots, U_L', V_L')$ are in the same orbit all we need to do is show that $R_i \in \mathrm{GL}(r)$. To do so it's enough to show that $V_i'^\top V_i$ is invertible. And indeed,

starting from Equation34

$$
\begin{aligned}
U_i V_i^\top &= U_i' V_i'^\top \\
&\textit{(Multiply both sides by } (U_i^\top U_i)^{-1} U_i^\top \textit{ from the left)} \\
(U_i^\top U_i)^{-1} U_i^\top U_i V_i^\top &= (U_i^\top U_i)^{-1} U_i^\top U_i' V_i'^\top \\
&\textit{(Simplify left side: } (U_i^\top U_i)^{-1} U_i^\top U_i = I\textit{)} \\
V_i^\top &= (U_i^\top U_i)^{-1} U_i^\top U_i' V_i'^\top \\
&\textit{(Multiply both sides by } V_i \textit{ from the right)} \\
V_i^\top V_i &= (U_i^\top U_i)^{-1} U_i^\top U_i' V_i'^\top V_i \\
&\textit{(Multiply both sides by } (V_i^\top V_i)^{-1} \textit{ from the left)} \\
I &= (V_i^\top V_i)^{-1} (U_i^\top U_i)^{-1} U_i^\top U_i' V_i'^\top V_i.
\end{aligned}
\tag{36}
$$

Therefore, $R_1, \ldots, R_L \in \mathrm{GL}(r)$, $U_i = U_i' R$, and $V_i = V_i' R_i^{-\top}$, implying that $(U_1, V_1, \ldots, U_L, V_L)$ and $(U_1', V_1', \ldots, U_L', V_L')$ are in the same G-orbit. □

We are now ready to prove Theorem C.2.

Proof of theorem C.2. Let $f : \mathcal{D} \to \mathbb{R}$ be a continuous GL-invariant function, let $K \subset \mathcal{D}$ be a compact set and fix $\epsilon > 0$.

1. **MLP.** Since K is compact and f is continuous, universality follows from the universal approximation theorem for MLPs [32].

2. $\mathrm{MLP}(\text{O-}\textbf{Align}([U, V]))$**.** Let f^{align} be the O-canonicalization function. f^{align} is continuous so $f^{\mathrm{align}}(K)$ is compact. and let f^{MLP} be an MLP that approximates f up-to ϵ on $f^{\mathrm{align}}(K)$.

$$
\begin{aligned}
\sup_{\boldsymbol{X} \in K} |f^{\mathrm{MLP}}(f^{\mathrm{align}}(\boldsymbol{X})) - f(\boldsymbol{X})| &= \sup_{\boldsymbol{X} \in K} |f^{\mathrm{MLP}}(f^{\mathrm{align}}(\boldsymbol{X})) - f(f^{\mathrm{align}}(\boldsymbol{X}))| \\
&= \sup_{\boldsymbol{Y} \in f^{\mathrm{align}}(K)} |f^{\mathrm{MLP}}(\boldsymbol{Y}) - f(\boldsymbol{Y})| < \epsilon.
\end{aligned}
$$

   The first equality holds since f is GL-invariant, and in particular O-invariant.

3. $\mathrm{MLP}(UV^\top)$**.** From Lemma C.4 we know that f^{mul} separates orbits. It's additionally clear that f^{mul} is continuous and G-invariant. Therefore, using Proposition C.3 there exists a function $f^{\mathrm{general}} : \mathbb{R}^N \to \mathbb{R}$ such that $f \equiv f^{\mathrm{general}} \circ f^{\mathrm{mul}}$ on K. Since f^{mul} is continuous, $f^{\mathrm{mul}}(K)$ is also compact and we can use the universal approximation theorem of MLPs for f^{genral} on $f^{\mathrm{mul}}(K)$. Therefore, there exists an MLP f^{MLP} such that

$$
\begin{aligned}
\sup_{\boldsymbol{X} \in K} |f^{\mathrm{MLP}}(f^{\mathrm{mul}}(\boldsymbol{X})) - f(\boldsymbol{X})| &= \sup_{\boldsymbol{X} \in K} |f^{\mathrm{MLP}}(f^{\mathrm{mul}}(\boldsymbol{X})) - f^{\mathrm{general}}(f^{\mathrm{mul}}(\boldsymbol{X}))| \\
&= \sup_{\boldsymbol{Y} \in f^{\mathrm{mul}}(K)} |f^{\mathrm{MLP}}(\boldsymbol{Y}) - f^{\mathrm{general}}(\boldsymbol{Y})| < \epsilon
\end{aligned}
\tag{37}
$$

4. **GL-net.** Since we proved $\mathrm{MLP}(UV^\top)$ is universal, it's enough to show that GL-net can implement $\mathrm{MLP}(UV^\top)$. If we take a GL-net with no equivariant layers (or equivalently a single equivariant layer that implements the identity by setting $\boldsymbol{\Phi}_i = I_{n_i}$, $\boldsymbol{\Psi}_i = I_{m_i}$) and apply the invariant head directly to the input, the resulting model is exactly $\mathrm{MLP}(UV^\top)$.

□

D Experimental Details

D.1 Diffusion Model LoRA Datasets

CelebA-LoRA We train a dataset of 3,900 LoRA finetuned Stable Diffusion 1.4 models [63] using the PEFT library [51]. Each LoRA is rank 4, and is finetuned via DreamBooth personalization [64] on 21 images of a given celebrity in the CelebA dataset [48]. Further, each LoRA is trained with randomly sampled hyperparameters (gradient accumulation steps, train steps, learning rate, prompt) and initialization.

Imagenette-LoRA We also use Dreambooth to finetune another 2,046 Stable Diffusion 1.4 models and 2,046 Stable Diffusion 1.5 models with LoRA rank 4 on different subsets of the Imagenette dataset [34] — a subset of ImageNet [12] consisting of images from ten dissimilar classes. Unlike CelebA LoRA, we use the same training hyperparameters for each finetuning run (but vary initialization, random seed, and finetuning dataset). We also finetune 2,046 SD1.4 rank 32 LoRAs.

Qwen2-ARC-LoRA and Llama3.2-ARC-LoRA We create two datasets totalling 4,000 language model LoRAs by finetuning Qwen2-1.5B [79] and Llama3.2-3B [15] on subsets of the training set of the commonly used ARC dataset [8], which is a dataset for testing question answering and science knowledge. The ARC dataset consists of questions from many data sources. For each LoRA, we randomly sample a subset of 19 data sources, and omit data from the unsampled data sources. Also, each LoRA is randomly initialized and trained with randomly sampled hyperparameters. See Appendix D.2 for more details.

D.1.1 CelebA Finetuning

Table 6: Hyperparameter distributions for the 3,900 different LoRA diffusion model finetunes we trained. $U(S)$ denotes the uniform distribution over a set S.

Hyperparameter	Distribution
Learning rate	$U(\{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\})$
Train Steps	$U(\{100, 133, 167, 200\})$
Batch Size	$U(\{1, 2\})$
Prompt	$U(\{$“Celebrity", “Person", “thing", “skd"$\})$
Rank	4

We finetune 3,900 models on various celebrities in the CelebA dataset [48] using the DreamBooth personalization method [64]. Each LoRA is personalized to one celebrity by finetuning on 21 images of that celebrity. Every LoRA is trained starting from a different random initialization, with hyperparameters randomly sampled from reasonable distributions — see Table 6 for the distributions.

CLIP score prediction. For CLIP score prediction, we use the following prompts, the first thirty of which are from PartiPrompts Yu et al. [80], and the last three of which are written to include words relevant to prompts the models are finetuned on. We use a fixed random seed of 42 for image generation. For predicting CLIP scores, GL-net takes the mean of each matrix product $U_i V_i^\top$ in the invariant head. This is equivalent to using an equivariant hidden dimension of 1.

1. ‘a red sphere on top of a yellow box’,
2. ‘a chimpanzee sitting on a wooden bench’,
3. ‘a clock tower’,
4. ‘toy cars’,
5. ‘a white rabbit in blue jogging clothes doubled over in pain while a turtle wearing a red tank top dashes confidently through the finish line’,
6. ‘a train going to the moon’,
7. ‘Four cats surrounding a dog’,
8. ‘the Eiffel Tower in a desert’,
9. ‘The Millennium Wheel next to the Statue of Liberty. The Sagrada Familia church is also visible.’,

10. ‘A punk rock squirrel in a studded leather jacket shouting into a microphone while standing on a stump and holding a beer on dark stage.’,
11. ‘The Statue of Liberty surrounded by helicopters’,
12. ‘A television made of water that displays an image of a cityscape at night.’,
13. ‘a family on a road trip’,
14. ‘the mona lisa wearing a cowboy hat and screaming a punk song into a microphone’,
15. ‘a family of four posing at Mount Rushmore’,
16. ‘force’,
17. ‘an oil surrealist painting of a dreamworld on a seashore where clocks and watches appear to be inexplicably limp and melting in the desolate landscape. a table on the left, with a golden watch swarmed by ants. a strange fleshy creature in the center of the painting’,
18. ‘Downtown Austin at sunrise. detailed ink wash.’,
19. ‘A helicopter flies over Yosemite.’,
20. ‘A giraffe walking through a green grass covered field’,
21. ‘a corgi’s head’,
22. ‘portrait of a well-dressed raccoon, oil painting in the style of Rembrandt’,
23. ‘a volcano’,
24. ‘happiness’,
25. ‘the words ’KEEP OFF THE GRASS’ on a black sticker’,
26. ‘A heart made of wood’,
27. ‘a pixel art corgi pizza’,
28. ‘two wine bottles’,
29. ‘A funny Rube Goldberg machine made out of metal’,
30. ‘a horned owl with a graduation cap and diploma’,
31. ‘A celebrity in a park’,
32. ‘A person on the beach’,
33. ‘A thing in a city’

Table 7: We report train and test mean squared error (MSE) for predicting normalized CLIP score of CelebA-LoRA models. We also show test Kendall's τ and R^2 coefficients. GL-net has significantly lower test MSE than any other LoL model, whereas a standard MLP does no better than random guessing. Transformers, MLPs with O-Align, SVD, or Dense featurization all perform similarly.

	LoL Model	Train MSE	Test MSE (↓)	τ (↑)	R^2 (↑)
Naive Models	$\mathrm{MLP}([U, V])$	$.047 \pm .004$	$.988 \pm .005$	$.226 \pm .016$	$.333 \pm .022$
	$\mathrm{Transformer}([U, V])$	$.027 \pm .005$	$.158 \pm .013$	$.671 \pm .003$	$.872 \pm .002$
	$\mathrm{MLP}(UV^\top)$	$.013 \pm .002$	$.169 \pm .011$	$.677 \pm .006$	$.871 \pm .005$
Efficient Invariant	$\mathrm{MLP}(\text{O-Align}([U, V]))$	$.001 \pm .001$	$.175 \pm .006$	$.654 \pm .004$	$.856 \pm .004$
	$\mathrm{MLP}(\sigma(UV^\top))$	$.071 \pm .003$	$.148 \pm .005$	$.667 \pm .008$	$.863 \pm .006$
	GL-net	$.009 \pm .001$	$\mathbf{.111} \pm .004$	$\mathbf{.695} \pm .005$	$\mathbf{.884} \pm .003$

CelebA Attribute Prediction. As mentioned in Section 4.2.2, we only predict the five least noisy CelebA attributes, as measured by Lingenfelter et al. [46]. Recall that each LoRA in CelebA-LoRA is trained on 21 images of a given celebrity. The attribute labels can vary across these 21 images, so we say the ground truth attribute label of the LoRA is the majority label across these 21 images.

D.1.2 ImageNette Finetuning

For Imagenette [34], we finetune $3 \times 2{,}046$ models. In particular, for each nonempty subset of the 10 Imagenette classes, we finetune 6 models using 1 image from each present class. Two are rank-32 finetunes on Stable Diffusion V1.4, two are rank-4 finetunes on Stable Diffusion V1.4, and two are rank-4 finetunes of Stable Diffusion V1.5. We use rank 32 in part so that we can test how well LoL models perform on larger ranks than the other experiments.

Table 8: Hyperparameters for the dataset of 2,046 different LoRA diffusion model finetunes we trained on Imagenette (Imagenette-LoRA).

Hyperparameter	Value
Learning rate	$3 \cdot 10^{-4}$
Train Steps	150
Batch Size	1
Prompt	`sks_photo`
Rank	4, 32

D.2 Language Model LoRA Dataset

Here, we describe the details of our Qwen2-ARC-LoRA and Llama3.2-ARC-LoRA datasets of trained language model LoRAs.

For training data, we use the ARC training set [8], using both the easy and challenge splits. First, we hold out a fixed validation set sampled from this set to compute the validation loss on. Each training data point originates from one of 20 sources (e.g. some questions come from Ohio Achievement Tests, and some come from the Virginia Standards of Learning). For each LoRA finetuning run, we sample a random subset of these sources to use as training data (except we do not ever omit the Mercury source, which contains many more data points than the other sources). To do this, we sample a random integer in $s \in [1, 19]$, then choose a random size-s subset of the 19 possibly-filtered sources to drop.

Table 9: Hyperparameter distributions for the 2,000 different LoRA language model finetunes we trained (Qwen2-ARC-LoRA and Llama3.2-ARC-LoRA). $U(S)$ denotes the uniform distribution over a set S.

Hyperparameter	Distribution
Learning rate	$10^{U([-5,-3])}$
Weight decay	$10^{U([-6,-2])}$
Epochs	$U(\{2, 3, 4\})$
Batch Size	$U(\{32, 64, 128\})$
LoRA Dropout	$U([0, .1])$
Filtered sources	$U(\text{sources})$

For each LoRA finetuning run, we sample random hyperparameters: learning rate, weight decay, number of epochs, batch size, LoRA dropout, and the data sources to filter. The distributions from which we sample are shown in Table 9, and were chosen to give reasonable but varied performance across different runs. All trained LoRAs were of rank 4, and we only applied LoRA to tune the key and value projection matrices of each language model.

D.3 Language Model LoRA Experiments

Here, we provide further details on the LLM experiments in Section 4.3. For each dataset and LoL model, we train for one run with each of the learning rates $5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}$. Then for the data membership task we choose the learning rate of highest validation accuracy, and for the validation loss and ARC-C regression tasks we choose the learning rate of highest validation R^2. With this optimal learning rate, we conduct 3 training runs, and report the mean and one standard deviation in Section 4.3. For the regression tasks, we train with an MSE loss, and for the data membership task we train with a cross entropy loss. For each dataset, we train on 80% of the data, validate on 10% of the data, and test on 10%. We use the AdamW optimizer [49] with weight decay of 10^{-2}, and evaluate the test performance on the checkpoint with lowest validation loss.

D.4 Dataset size prediction

Dataset size prediction. The task of predicting the size of the finetuning dataset given LoRA weights was first studied recently by Salama et al. [65]. The success of this task implies a privacy leak, since model developers may sometimes wish to keep the size of their finetuning dataset private. Moreover, the dataset size is a useful quantity to know for data membership inference attacks and model inversion attacks, so accurately predicting dataset size could improve effectiveness of these attacks too [73, 26].

Table 10: Results for finetuning dataset size prediction with LoL models. We use our Imagenette-LoRA dataset, and predict the number of classes (or equivalently images) that are used to finetune each model.

LoL Model	Train Loss	Test Acc
MLP	$.547 \pm .026$	25.9 ± 0.6
$\mathrm{MLP}(\text{O-Align}([U, V]))$	$.020 \pm .001$	33.7 ± 1.5
$\mathrm{MLP}(\sigma(UV^\top))$	$.153 \pm .013$	$\mathbf{58.8} \pm 2.7$
$\mathrm{MLP}(UV^\top)$	$.547 \pm .099$	39.0 ± 4.4
GL-net	$.011 \pm .002$	44.3 ± 3.3

In Table 10, we show results for finetuning-dataset-size prediction using LoL models. The task is to take the LoRA weights of one of the diffusion models from our Imagenette-LoRA dataset, and predict the number of unique images that it was finetuned on. Although it struggles on some other tasks, we see that $\mathrm{MLP}(\sigma(UV^\top))$ is able to effectively predict dataset size, in line with observations from Salama et al. [65].

Other LoL models struggle to generalize well on this task. Interestingly, GL-net performs approximately as well as expected if using the following strategy: first predict which classes are present in the finetuning set of the LoRA (as in Table 5), and then sum the number of classes present to predict the dataset size. MLP+SVD is clearly using different predictive strategies for this task, as it cannot predict which individual classes are present (Table 5), but it can predict the number of classes present. See Appendix D.4 for more analysis on the learned prediction strategies of the LoL models on this task. Here we describe theoretical and experimental details relating to the implicit strategies that GL-net and $\mathrm{MLP}(\sigma(UV^\top))$ may employ in dataset size prediction as in Table 10.

Suppose that a model f_{class} trained to predict Imagenette class presence of LoRAs (as in Table 5) has test accuracy p, while each of the 10 classes is present with probability $.5$. For LoRA weights x, $f_{\text{class}}(x)_i = 1$ if the model predicts that class i is in the finetuning dataset of x. Define the corresponding dataset-size prediction model $f_{\text{size}}(x) = \sum_{i=1}^{10} f_{\text{class}}(x)_i$ That is, f_{size} predicts whether each class is in the dataset, sums up these predictions, and then outputs the sum as the expected dataset size.

Assuming f_{class} is equally likely to provide false-negative or false-positive predictions for each class, each of its 10 outputs has probability $1 - p$ of being incorrect. Consider an input LoRA x with a dataset size s. Then $f_{\text{size}}(x) = \hat{y}_1 + \hat{y}_2 + \ldots \hat{y}_{10}$, where $\hat{y}_i$ is equal to y_i with probability p, and $1 - y_i$ with probability $1 - p$. So,

$$f_{\text{size}}(x) = s \iff |\{i \mid y_i = 1 \wedge \hat{y}_i = 0\}| = |\{i \mid y_i = 0 \wedge \hat{y}_i = 1\}| \tag{38}$$

The cardinality of the left set is distributed as $\mathrm{binom}(s, 1-p)$, while the cardinality of the right set is distributed as $\mathrm{binom}(10-s, 1-p)$. So, $\mathbb{P}(f_{\text{size}}(x) = s|s) = \mathbb{P}(\beta_1 = \beta_2)$, for $\beta_1 \sim \mathrm{binom}(s, 1-p)$ and $\beta_2 \sim \mathrm{binom}(10-s, 1-p)$. Thus,

$$\mathbb{P}_{(x,s)\sim\text{data}}(f_{\text{size}}(x) = s) = \sum_{\eta=0}^{10} [\mathbb{P}\left(\mathrm{binom}(\eta, 1-p) = \mathrm{binom}(10-\eta, 1-p) \mid \eta\right) \cdot \mathbb{P}(\eta)] \tag{39}$$

Table 5 shows that for GL-net, f_{class} is correct with probability $p = .878$. So, we have $1 - p = .122$. On the otherhand, for rank-32 data membership on Imagenette, GL-net is correct with $p = .904$.

Using a Python script to evaluate (39), we find that the probability that $f_{\text{size}}(x)$ is correct is 39.9% in the rank 4 case, which is near the observed probability of 44.3%. In the rank 32 case, the theoretical accuracy is 46.1%, which almost exactly matches the observed probability of 46.8%.

We further test our hypothesis empirically by training GL-net on Imagenette-32 class prediction (as in Table **??**) and then summing up its class predictions to predict dataset size. On the Imagenette size-prediction test set, GL-net with this sum strategy achieves an accuracy of $43.5 \pm 2.1\%$. This means GL-net is likely learning a function with underlying mechanism similar to f_{size}, where it predicts the presence of each class and then sums those predictions to output the total dataset size.

On the other hand, $\text{MLP}(\sigma(UV^\top))$ correctly predicts class presence in Imagenette-4 with probability $p = .61$. Predicting classes and summing up its individual predictions would give $\text{MLP}(\sigma(UV^\top))$ a theoretical accuracy of 20% by equation 39, which is significantly worse than its true performance of 58.8%. This suggests that $\text{MLP}(\sigma(UV^\top))$ likely uses higher level characteristics than class presence to determine dataset size.

D.5 Compute Used

We used a total of 144 NVIDIA 3090 (24GB memory) hours to generate diffusion LoRA datasets, and 832 NVIDIA 2080 (11GB memory) hours to generate language LoRA finetunes. Per-dataset compute is listed below. Most other uses of compute, including training and hyperparameter tuning LoL models were negligible and account for < 48 GPU hours.

Dataset Name	Training Time	GPU Used
CelebA-LoRA	∼48 hrs	NVIDIA 3090
CelebA-LoRA (Compute CLIP scores)	∼48 hrs	NVIDIA 3090
Imagenette-LoRA	∼72 hrs	NVIDIA 3090
Qwen2-ARC-LoRA	∼384 hrs	NVIDIA 2080 Ti
Llama3.2-ARC-LoRA	∼448 hrs	NVIDIA 2080 Ti

Table 11: Type of GPU and number of GPU hours to generate each dataset.

E LoRA Variants and Their Symmetries

In this section, we describe various LoRA variants that have been proposed for parameter-efficient finetuning. We also discuss their symmetries, and how one may process these LoRA variants with LoL models.

Training Modifications. Several LoRA variants such as PiSSA [55] and LoRA+ [29] have the same type of weight decomposition as the original LoRA, but with different initialization or training algorithms. These variants have the same exact symmetries as standard LoRA, so they can be processed in exactly the same way with our LoL models.

DoRA (Weight-Decomposed Low-Rank Adaptation). Liu et al. [47] decompose the parameter updates into magnitude and directional components. For base weights $W \in \mathbb{R}^{n\times m}$, DoRA finetuning learns the standard low rank weights $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{m\times r}$ along with a magnitude vector $\mathbf{m} \in \mathbb{R}^m$ so that the new finetuned weights are given by

$$(W + UV^\top)\text{Diag}\left(\frac{\mathbf{m}}{\|W + UV^\top\|_c}\right), \tag{40}$$

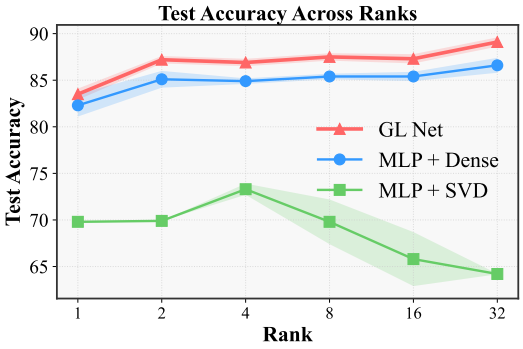
where $\|\cdot\|_c : \mathbb{R}^{n\times m} \to \mathbb{R}^m$ is the norm of each column of the input matrix, the division $\frac{\mathbf{m}}{\|W+UV^\top\|_c}$ is taken elementwise, and Diag takes vectors in $\mathbb{R}^m$ to diagonal matrices in $\mathbb{R}^{m\times m}$. The vector $\mathbf{m}$ represents the norm of each column of the finetuned matrix. DoRA weights $(U, V, \mathbf{m})$ also have the same invertible matrix symmetry as standard LoRA, where $(UR, VR^{-\top}, \mathbf{m})$ is functionally equivalent to $(U, V, \mathbf{m})$. The $\mathbf{m}$ vector cannot in general be changed without affecting the function. Thus, an LoL model could take as input $\mathbf{m}$ as an invariant feature, for instance by concatenating it to features in an invariant head of GL-net, or concatenating it to the input of an MLP in the other LoL models.

KronA (Kronecker Adapter). Edalati et al. [19] use a Kronecker product structure to decompose the learned weight matrix in a parameter-efficient way. For a weight matrix W of shape $n \times m$, LoKr learns $U \in \mathbb{R}^{n'\times m'}, V \in \mathbb{R}^{n''\times m''}$ such that the finetuned weight is given by $W + U \otimes V$. Here,

we no longer have a large general linear symmetry group. Instead, there is a scale symmetry, where (U, V) is equivalent to $(sU, \frac{1}{s}V)$ for a scalar $s \in \mathbb{R} \setminus \{0\}$, since $(sU) \otimes (\frac{1}{s}V) = U \otimes V$. For LoL tasks, we can use a scale-invariant architecture, as for instance explored by Kalogeropoulos et al. [38]. We can also use GL-net on the flattened $\text{vec}(U) \in \mathbb{R}^{n'm' \times 1}$ and $\text{vec}(V) \in \mathbb{R}^{n''m'' \times 1}$, since $\mathbb{R} \setminus \{0\} = \text{GL}(1)$ is the general linear symmetry group in the special case of rank $r = 1$.

F More Experiments

F.1 Generalization to Unseen Ranks



Rank	MLP+SVD	MLP+Dense	GL-net
1	.714 ± .03	.424 ± .02	.418 ± .01
2	1.01 ± .06	.357 ± .02	.335 ± .01
4	.571 ± .02	.369 ± .02	.323 ± .01
8	.604 ± .04	.349 ± .02	.324 ± .01
16	.700 ± .07	.343 ± .02	.317 ± .01
32	.789 ± .01	.328 ± .01	.284 ± .01

Figure 5: Performance of LoL models across inputs of varying ranks. Each model is only trained on rank 4 LoRA weights from CelebA-LoRAs. (Left) Test accuracy on CelebA attribute prediction. (Right) Test loss on CelebA attribute prediction. $\text{MLP}(UV^\top)$ and GL-net generalize well to ranks that are unseen during training, but do face degradation at rank one. On the other hand, $\text{MLP}(\sigma(UV^\top))$ does not generalize well.

In Section 4, each LoL model was trained on LoRA weights of one fixed rank ($r = 4$). In practice, model developers can choose different LoRA ranks for finetuning their models, even when they are finetuning the same base model for similar tasks. Thus, we may desire LoL models that are effective for inputs of different ranks. In this section, we explore an even more difficult problem — whether LoL models can generalize to ranks that are unseen during training time.

The $\text{MLP}(UV^\top)$ and GL-net models can directly take as input models of different ranks. We also use $\text{MLP}(\sigma(UV^\top))$ on inputs of different ranks, by parameterizing the model for inputs of rank r, then truncating to top r singular values for inputs of rank greater than r, and zero-padding to length r for inputs of rank less than r.

In Figure 5, we train LoL models on inputs of rank 4, and then test performance on inputs of ranks between 1 and 32. On the CelebA attribute prediction task, we see that $\text{MLP}(UV^\top)$ and GL-net mostly generalize very well to different ranks that are unseen during training (though not as well for $r = 1$), while $\text{MLP}(\sigma(UV^\top))$ does not generalize as well.

F.2 Runtime and Scaling to Large Models

Previous weight-space models generally take in small networks as inputs, e.g. 5,000 parameters [76], 80,000 parameters [43], or sometimes up to 4 million parameters [58]. However, many of the most impactful neural networks have orders of magnitude more parameters. Thus, here we consider the runtime and scalability of LoL models as we increase the model size. We will consider all of the language model sizes in the GPT-3 family [6], which range from 125M to 175B parameters. We assume that we finetune rank 4 LoRA weights for one attention parameter matrix for each layer.

In Figure 6, we show the time it takes for data preprocessing of 64 input networks, and forward passes for 512 input networks (with batch size up to 64). Even at the largest scales, it only takes at most seconds to preprocess and compute LoL model forward passes for each input LoRA. Every LoL model besides $\text{MLP}(UV^\top)$ scales well with the model size: forward passes barely take longer at larger model sizes, and data preprocessing is still limited to less than a second per input network.

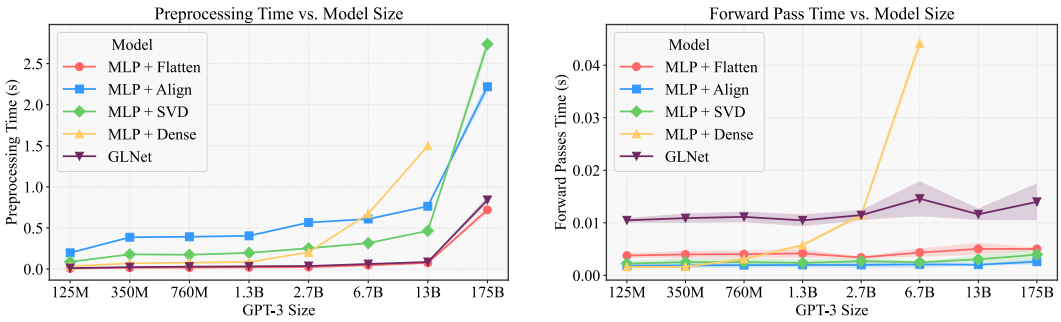


Figure 6: (Left) data preprocessing time for LoL models across 64 inputs of varying sizes. (Right) forward pass time for LoL models across 512 inputs of varying sizes. $\mathrm{MLP}(UV^\top)$ runs out of memory for largest inputs.

G Limitations

One possible limitation of our work is that in general, many thousands of networks need to finetuned in order to create a sufficient amount of data to effectively learn on low rank weight-spaces. However, during one instance of model merging, models might be evaluated thousands of times, so LoL can still be used to speed up evaluation on average.

H Impact Statement

Our work develops the LoL framework, which could be used for diverse tasks involving processing LoRA weights of finetuned models. This includes tasks that are likely to be societally positive, such as detecting harmful tendencies in finetuned models or speeding up model evaluation to reduce computational costs in certain settings. However, it could also be used for potentially negative tasks, for instance predicting private properties such as the size of a finetuning dataset. Still, our work is largely foundational, and we do not believe that there is overall much potential for negative impacts from our work.