Model-invariant weight distribution descriptors for visual exploration of neural networks en masse

G. Eilertsen, J. Jönsson, J. Unger, and A. Ynnerman

Linköping University, Sweden

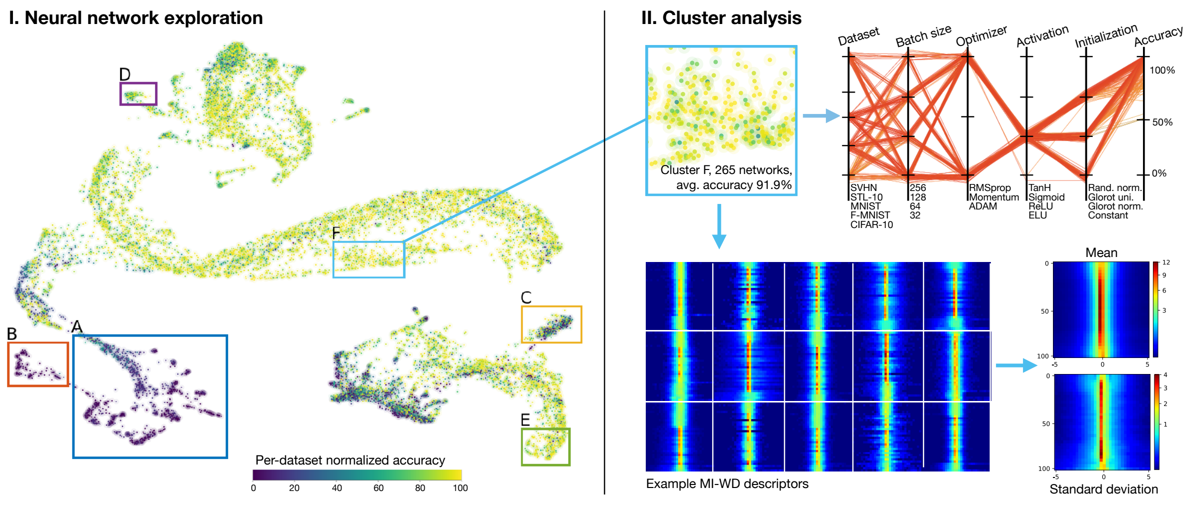

Figure 1: *Visual exploration of a neural network model library composed of 13,000 separately trained models.* ***Left****: UMAP embedding of the weight space where each neural network is represented using a 1024 dimensional model-invariant weight distribution (MI-WD) descriptor. Colors illustrate the test accuracies of the individual models, normalized on a per-dataset basis to be comparable between models trained using different datasets.* ***Right****: Analysis of a high-performing cluster, where parallel coordinates show the combinations of hyper-parameters that are characteristic for the selected cluster. The example MI-WD descriptors show how different architectures generate similar but not equal representations, while the mean and standard deviation MI-WD descriptors illustrate overall cluster properties and variations. We refer to Fig. 5 and the related discussion for an analysis of the different cluster selections.*

Abstract

We present a neural network representation which can be used for visually analyzing the similarities and differences in a large corpus of trained neural networks. The focus is on architecture-invariant comparisons based on network weights, estimating similarities of the statistical footprints encoded by the training setups and stochastic optimization procedures. To make this possible, we propose a novel visual descriptor of neural network weights. The visual descriptor considers local weight statistics in a model-agnostic manner by encoding the distribution of weights over different model depths. We show how such a representation can extract descriptive information, is robust to different parameterizations of a model, and is applicable to different architecture specifications. The descriptor is used to create a model atlas by projecting a model library to a 2D representation, where clusters can be found based on similar weight properties. A cluster analysis strategy makes it possible to understand the weight properties of clusters and how these connect to the different datasets and hyper-parameters used to train the models.

1. Introduction

With the vast number of neural networks trained every day, methods for visualizing the relations between them has become increasingly important. A meaningful visualization of the differences and similarities of a highly diverse set of neural networks can be used for, e.g., explainable AI, providing cues on how different datasets, architectures, and hyper-parameters relate to each other in terms of trained models, and as a general purpose visual analytics tool for conveying large numbers of trained models.

In this paper, we focus on constructing a tool for screening the

DOI: 10.2312/evs.20241068

space spanned by a large-scale library of trained neural networks. At the heart of the tool is the method used to represent neural networks, which can be used for measuring meaningful distances between models. For this purpose, we introduce a novel descriptor that can extract relevant information suitable for measuring network similarity. Although similarity between layers of different networks can be defined using network activations from some representative set of input samples [RGYSD17, MRB18, KNLH19], this is 1) computationally demanding, 2) cannot be directly used for comparing networks with different architectures, and 3) requires explicit access to architecture specification and representative data. On the other hand, recent work has demonstrated how the trained weights locally encode a wealth of information [EJR*20, UKG*20], and that learned weight information generalizes to unseen architectures and datasets [UKG*20]. However, due to the symmetric nature of neural networks, where the same model can be described with widely different parameterizations, it is not possible to directly compare weights of different models [NSA*23, ZYJ*23]. Thus, we formulate a weight descriptor based on local statistics of the trained weights of a network. This does not rely on access to data or architecture specification, i.e. can be applied to a black-box model with no auxiliary information, and provides a representation that is robust to different parameterizations of the same model.

Through experiments, we demonstrate promising properties of the proposed descriptor, showing how it can successfully untangle information related to how a network is trained. Then, we demonstrate the visually represented model library that we construct for exploration of a diverse set of neural networks, Fig. 1, and discuss how this can be used to extract knowledge on how hyperparameters shape the neural network weight space.

Related work: There are many attempts at providing an understanding of neural networks by leveraging activations at different levels within a network. For example, the similarity between different layers can be formulated by comparing activations [RGYSD17, MRB18, KNLH19, BJR22], and many methods exploit activations in different ways [ZF14, MV15], also for the embedding of many networks using concatenated activation vectors [ECBV10]. Tools for visual analysis using this type of information have seen rapid developments over the last years [LRBB*23, RFFT17, KAKC17, PHVG*17, HKPC18, WTS*20, CL18]. Distilling information from direct analysis of weights, however, is less explored, and mostly limited to understanding learning trajectories [GD97, ASD18, SB20]. There are also some attempts at understanding neural networks by learning from large numbers of sampled weight instances [EJR*20, UKG*20, NSA*23, ZYJ*23]. Previous work has focused on visualization of individual or few networks, and to our knowledge, there are no previous methods that have successfully visualized the relations between large quantities of diverse neural networks purely based on their weights.

2. Method

The central part of our visual exploration tool is the format used to represent each model. We first describe the proposed statistical approach for this purpose, followed by an outline on how this is used to construct a visual model library.

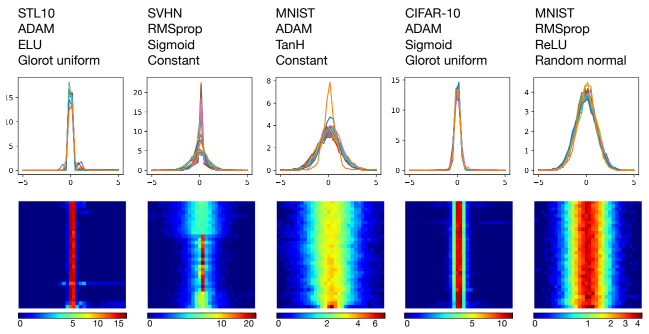

Figure 2: *Descriptors with $B = 32$ histogram bins and evaluated at $S = 32$ local windows of trained weights. The information on the top lists the dataset, optimizer, activation function and initialization scheme, respectively, used for training the individual networks.*

2.1. Model-invariant weight descriptors

We consider the neural network weights $\theta \in \mathbb{R}^N$, represented as a sequence of vectorized and concatenated weight matrices and biases, including batch normalization weights (similar to [EJR*20]). We extract features purely based on the distributions of weights in localized neighborhoods of θ. This is to some extent similar to histograms of oriented gradients (HOG) [DT05], which are popular for image recognition.

If we denote the weight at location i by θ_i, for $i = \{1,...,N\}$, we can evaluate histograms on chunks of weights $\theta_{i_0:i_1}$, between indices i_0 and i_1. We define S non-overlapping windows using $i_0 = \lfloor N(s-1)/S \rfloor + 1$ and $i_1 = \lfloor Ns/S \rfloor$, for $s = \{1,...,S\}$. For each window, we evaluate the local normalized histogram with B bins and arrange the histograms as rows in an $S \times B$ matrix. This forms a 2D descriptor describing the local distribution of weights for different depths of a network, with depth increasing towards the lower part of the matrix. Some examples of descriptors are illustrated in Fig. 2, visualized both using plots for each local histogram as well as color-mapped image representations. We can see that the local weight distribution is often similar across depths, but this is not always the case, especially in the early and deep layers (top and bottom, respectively, in the descriptor matrix).

The benefit of our descriptor definition is that it can be applied on θ of different sizes N without knowledge of how different layers are defined. The only requirement is to have θ constructed consistently, with different layers in increasing order. Since the descriptor relies only on the distribution of weights it is likely to be robust to different formats of exported weights, e.g., if weight matrices are stored in row- or column-wise order. As default, we use $S = B = 32$, for a representation described with $K = 1024$ dimensions, but we also test other dimensions in our experiments. For the evaluation of histograms, we define bins between weight values -5 and 5. On average, this corresponds approximately to the 5th and 95th percentiles of a weight vector θ, making the descriptor robust against extreme values.

2.2. A model library for visual exploration

While MI-WD descriptors have the potential to be used in many different contexts, we focus on the problem of visualizing the relations between a large number of neural networks.

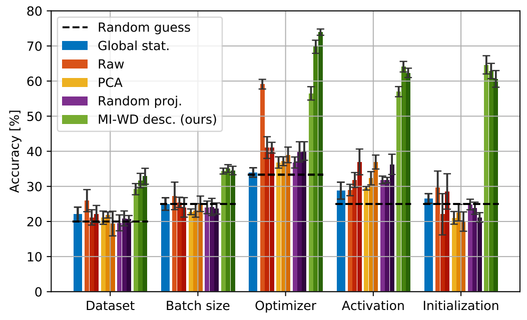

Figure 3: *Linear SVMs trained for hyper-parameter classification of different weight representations. Error bars show standard deviation over a 5-fold cross-validation. Except for the global statistics, representations have been tested with 64, 256, and 1024 dimensions (left, middle, and right bars, respectively) for each method.*

Model atlas: The base for the method is a UMAP embedding [MHSG18] for visually screening the space of available models, formulating the projection $f : \mathbb{R}^{S\times B} \rightarrow \mathbb{R}^2$ from the $K = SB$ dimensional MI-WD space (where each descriptor is first vectorized). With the resulting atlas of 2D coordinates, it is possible to find clusters with similar weight properties, see Fig. 1(left).

Cluster analysis: To build an understanding of the cluster characteristics, we implement a selection-based summary of properties, illustrated in Fig. 1(right). The summary allows for exploring the MI-WD descriptors of a cluster, providing both visual comparisons of the individual descriptors as well as the descriptor mean and standard deviation over the cluster. As the descriptors describe the distributions of weights over different model depths, it is possible to pinpoint where and how much the models typically differ within a cluster. This focuses on *direct* understanding on what distributions of weights form the cluster, as well as the variations between models. As a means of *indirect* understanding of similarities and variations between models in terms of the datasets and hyper-parameters that were used in training, we connect a parallel coordinate representation that shows this information for the models in the cluster. Fig. 1(right) shows an example, where a selection of six parameters have been plotted, including the test accuracy of the models.

Pipeline: The tool for model library exploration can be composed as: 1) Construct MI-WD descriptors for each model in the library. 2) Perform UMAP projection on the collection of vectorized descriptors, and present the 2D embedding result, e.g. colorized to convey test accuracy of individual models. 3) Based on chosen areas of interest in the embedding, present information on a) individual descriptors together with mean and standard deviation over the selected area, and b) parallel coordinates plot for fast visual feedback on the combination of hyper-parameters used in training.

We emphasize that the model library embedding does not require any knowledge of the architecture, training data, or parameters used by the different networks, i.e. it could also relate incoming black-box models to an already existing library. However, for the analysis of cluster configurations, we link information on training setup to the visualization pipeline, to enable an understanding of how hyper-parameters indirectly shape the relations between network weights.

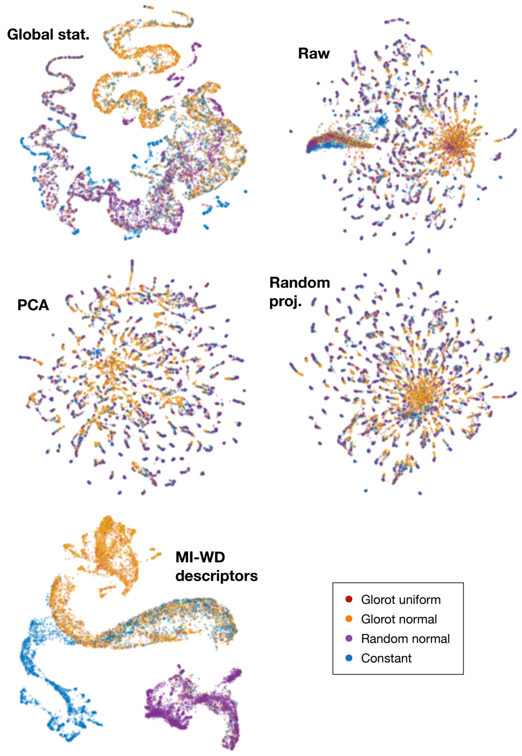

Figure 4: *UMAP embeddings of different weight representations, color coded to visualize the initialization schemes used in training.*

3. Results

We verify the quality of the MI-WD descriptors and compare to a selection of baseline methods. We then demonstrate the visual exploration of a model library. For additional examples and analysis of the descriptors, e.g. in terms of invariance to different parameterizations of a network, we refer to the supplementary material.

Baselines: A naive baseline, which we term *raw*, re-samples the raw weights θ in K dimensions using linear interpolation. A second simple baseline considers *global statistical features* extracted from raw weights. These are specified as in [EJR*20], from 8 different statistical measures of a weight vector: mean, variance, skewness (third standardized moment), and five-number summary (1, 25, 50, 75, and 99 percentiles). The measures are applied directly on the weights θ and weight gradients $\nabla\theta_i = \theta_{i+1} - \theta_i$, for a total of 16 features. Furthermore, as an example of a commonly used dimensionality reduction method we apply *PCA* on re-sampled raw weights $\hat{\theta} \in \mathbb{R}^{\hat{N}}$ represented in $\hat{N} = 65{,}536$ dimensions, and then use the K first principal components as a weight representation. Finally, we also test randomly projecting $\hat{\theta}$ (in 65,536 dimensions as with PCA) to a K-dimensional space, $z = \Phi\hat{\theta}$, where Φ is a $K \times \hat{N}$ matrix populated by normally distributed random numbers ($\mu = 0, \sigma = 1$).

Dataset: We use the neural weight space (NWS) dataset from [EJR*20], which is composed of 13,000 trained CNNs with a wide variety of architectures, training data, and hyper-parameters. For the numerical evaluations, we use the curated training dataset of 8,035 models for producing PCA projection and perform evaluations on the 2448 models of the test set. For testing the visual exploration tool, we include all 13,000 models in a model library.

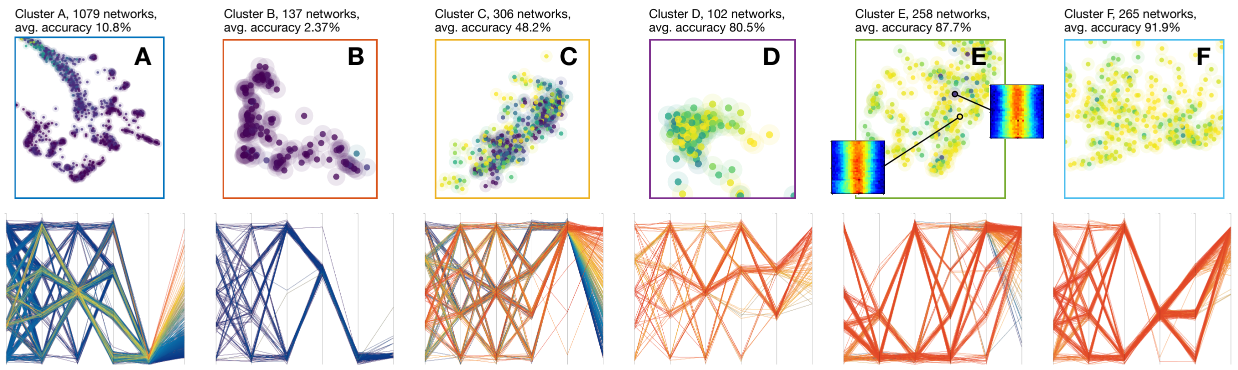

Figure 5: *Cluster properties of the selections in Fig. 1, with color encoding and parallel coordinate labels as in Fig. 1. As visible from the color encoding, and the last parallel coordinate axis, the clusters differ in performance of the clustered networks, where **A**-**B** are low-performing, **C** is medium-performing, and **D**-**F** are high-performing. The example descriptors in cluster **E** demonstrate closely located good and bad local minima in the visualized weight space.*

Untangling hyper-parameters: A common strategy for quality evaluation of representations is to train a linear classifier in the representation space, showing how successful the representation is at separating/untangling different types of variations. We make a similar evaluation using weight descriptors and baselines, by considering classification of hyper-parameters from the weights. This is similar to the experiments on raw weights in [EJR*20], but here we consider weight representations. The results for five different hyper-parameter classifications are shown in Fig. 3, where linear SVMs have been trained with 5-fold cross-validation on the NWS test set. We also evaluate representations with different dimensionality, $K = \{64, 256, 1024\}$. It is clear how the MI-WD descriptors overall are significantly better than the baseline methods, where most baselines are incapable of separating the variations.

Embeddings: Given the results in Fig. 3, we expect that MI-WD descriptors will generate a better structured UMAP embedding for visualizing the model library. This is also confirmed by comparing the projections in Fig. 4. Re-sampled raw weights, PCA, and random projections generate embeddings with little structure, while the global statistical features have problems forming meaningful clusters. The weight descriptor embedding, on the other hand, shows distinct clusters based on differences in network setups.

Model library: To provide an understanding of the relations between models, Fig. 1 highlights a set of selections that have been made using the cluster analysis tool. In Fig. 5, we focus on analyzing the clusters based on the combinations of training settings. The embedding is colored to show the normalized test performance of the different networks. Normalization has been done per dataset according to $(p_d - min(p_d))/(max(p_d) - min(p_d))$, where p are the performances of models trained on dataset d. With the normalization, it is possible to align model performances for a cross-dataset comparison. Based on accuracy, we classify clusters **A**-**B** as low-performing, **C** as medium-performing, and **D**-**F** as high-performing. Clusters **A** and **B** use constant initialization (also visible in Fig. 4), which is highly correlated with low performance. However, there are low-performing models in clusters **C** which do not use constant initialization, and high-performing models in cluster **E** and **F** which have models with constant initialization. Cluster **B** shows how branches within a larger cluster have different setups – in this case with Sigmoid activation function and ADAM or RMSprop optimizers. Clusters **D**-**F** show good performance with different constellations of parameters. **D** uses mostly Glorot uniform or normal initialization together with Sigmoid or TanH activation. **E** has many models with constant or random normal initialization, ADAM or RMSprop initializers, and batch sizes of only 32 or 64. **F** uses all but random normal initialization together with ReLU activation and ADAM or RMSprop optimizers. Neither **E** or **F** include models trained on STL-10.

From inspecting the combination of hyper-parameters of different clusters it is possible to see how the embedding has a clear structure which allows for reasoning around how models relate to each other. It is also interesting to see how models with similar statistics can result in very different performances, e.g. in cluster **E**, where the illustrated descriptors are both trained on MNIST but where one has test accuracy 99.0% and the other 19.6%. This points to how good and bad local minima can be close in the weight space.

4. Conclusion

We have introduced a novel method for comparison of the weight properties of large quantities of diverse neural networks without access to information about their design or training setup. The tool is built around a descriptor for weight representation, which has been demonstrated successful for efficient model-invariant comparison of neural networks. The descriptor has tractable properties in terms of untangling the variations of the weight space and is robust to different parameterizations of the same model. We have verified that the proposed descriptor can aid in visual analysis of large quantities of networks. Future work includes extending the analysis to larger networks and datasets, as well as exploring possibilities in terms of visual analytics. With refined visualization and filtering options, we believe that weight space analysis has the potential to become a powerful tool for understanding and exploiting the relations between a wide variety of different models.

Acknowledgments

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP, NTU, NEST __main__) and the Zenith career development program at Linköping University.

References

[ASD18] Antognini J., Sohl-Dickstein J.: Pca of high dimensional random walks with comparison to neural network training. *Advances in Neural Information Processing Systems 31* (2018), 10307–10316. 2

[BJR22] Bäuerle A., Jönsson D., Ropinski T.: Neural activation patterns (naps): Visual explainability of learned concepts. *arXiv preprint arXiv:2206.10611* (2022). 2

[CL18] Choo J., Liu S.: Visual analytics for explainable deep learning. *IEEE computer graphics and applications 38*, 4 (2018), 84–92. 2

[DT05] Dalal N., Triggs B.: Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (2005), vol. 1, Ieee, pp. 886–893. 2

[ECBV10] Erhan D., Courville A., Bengio Y., Vincent P.: Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), JMLR Workshop and Conference Proceedings, pp. 201–208. 2

[EJR*20] Eilertsen G., Jönsson D., Ropinski T., Unger J., Ynnerman A.: Classifying the classifier: dissecting the weight space of neural networks. In *Proceedings of the European Conference on Artificial Intelligence (ECAI 2020)* (2020), pp. 1119–1126. 2, 3, 4

[GD97] Gallagher M., Downs T.: Visualization of learning in neural networks using principal component analysis. In *International Conference on Computational Intelligence and Multimedia Applications* (1997), pp. 327–331. 2

[HKPC18] Hohman F., Kahng M., Pienta R., Chau D. H.: Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics 25*, 8 (2018), 2674–2693. 2

[KAKC17] Kahng M., Andrews P. Y., Kalro A., Chau D. H.: A cti v is: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics 24*, 1 (2017), 88–97. 2

[KNLH19] Kornblith S., Norouzi M., Lee H., Hinton G.: Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414* (2019). 2

[LRBB*23] La Rosa B., Blasilli G., Bourqui R., Auber D., Santucci G., Capobianco R., Bertini E., Giot R., Angelini M.: State of the art of visual analytics for explainable deep learning. In *Computer graphics forum* (2023), vol. 42, Wiley Online Library, pp. 319–355. 2

[MHSG18] McInnes L., Healy J., Saul N., Grossberger L.: Umap: Uniform manifold approximation and projection. *Journal of Open Source Software 3*, 29 (2018), 861. 3

[MRB18] Morcos A., Raghu M., Bengio S.: Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems* (2018), pp. 5727–5736. 2

[MV15] Mahendran A., Vedaldi A.: Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 5188–5196. 2

[NSA*23] Navon A., Shamsian A., Achituve I., Fetaya E., Chechik G., Maron H.: Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning* (2023), PMLR, pp. 25790–25816. 2

[PHVG*17] Pezzotti N., Höllt T., Van Gemert J., Lelieveldt B. P., Eisemann E., Vilanova A.: Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE transactions on visualization and computer graphics 24*, 1 (2017), 98–108. 2

[RFFT17] Rauber P. E., Fadel S. G., Falcão A. X., Telea A. C.: Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 101–110. doi:10.1109/TVCG.2016.2598838. 2

[RGYSD17] Raghu M., Gilmer J., Yosinski J., Sohl-Dickstein J.: Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems* (2017), pp. 6076–6085. 2

[SB20] Schürholt K., Borth D.: An investigation of the weight space for version control of neural networks. *arXiv preprint arXiv:2006.10424* (2020). 2

[UKG*20] Unterthiner T., Keysers D., Gelly S., Bousquet O., Tolstikhin I.: Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448* (2020). 2

[WTS*20] Wang Z. J., Turko R., Shaikh O., Park H., Das N., Hohman F., Kahng M., Chau D. H.: Cnn explainer: Learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). 2

[ZF14] Zeiler M. D., Fergus R.: Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833. 2

[ZYJ*23] Zhou A., Yang K., Jiang Y., Burns K., Xu W., Sokota S., Kolter J. Z., Finn C.: Neural functional transformers. *Advances in Neural Information Processing Systems 36* (2023). 2