Generative Modeling of Weights: Generalization or Memorization?

Boya Zeng[1] Yida Yin[2] Zhiqiu Xu[1] Zhuang Liu[3]

[1]University of Pennsylvania [2]UC Berkeley [3]Princeton University

Abstract

Generative models, with their success in image and video generation, have recently been explored for synthesizing effective neural network weights. These approaches take trained neural network checkpoints as training data, and aim to generate high-performing neural network weights during inference. In this work, we examine four representative methods on their ability to generate ***novel*** *model weights, i.e., weights that are different from the checkpoints seen during training. Surprisingly, we find that these methods synthesize weights largely by memorization: they produce either replicas, or at best simple interpolations, of the training checkpoints. Current methods fail to outperform simple baselines, such as adding noise to the weights or taking a simple weight ensemble, in obtaining different and simultaneously high-performing models. Our findings provide a realistic assessment of what types of data current generative models can model, and highlight the need for more careful evaluation of generative models in new domains.*

1. Introduction

Generative models have advanced significantly in recent years, achieving remarkable results in image synthesis [17, 39, 41]. They demonstrate exceptional photorealism, driving widespread applications in commercial art and graphics. Beyond static images, generative video models [1, 4] have recently gained attention, achieving impressive consistency and coherence in video synthesis.

Building on this success, recent studies [12, 35, 43, 50] have extended the use of generative models to synthesize weights for neural networks. These methods collect network checkpoints trained with standard gradient-based optimization, and apply generative models to learn the weight distributions and produce new checkpoints, without direct access to the training data of the original task [5, 23, 25, 30]. The weights generated by these methods can often perform comparably to conventionally trained weights: they achieve high test accuracy in image classification models and high-quality 3D shape reconstructions in neural field models, across diverse datasets and model architectures.

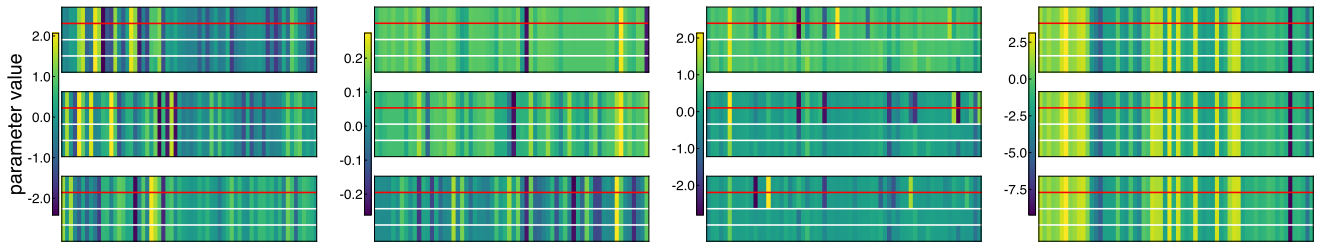
In this study, we seek to answer an important question: have the generative models learned to produce meaningfully distinct weights that *generalize* beyond the training set of checkpoints, or do they merely *memorize* and reproduce the training data? While prior work has focused on evaluating these methods based on the performance of the generated models on the downstream tasks, this question is critical to understanding both the fundamental mechanisms and the practicality of these methods.

To investigate this, we analyze four representative weight generation methods [12, 35, 43, 50], covering different types of generative models and downstream tasks. We first find the nearest training checkpoint to each generated checkpoint, to assess the novelty in the generated weight values. Surprisingly, *almost all* generated checkpoints *closely resemble* specific training checkpoints, showing far less novelty than a new model trained from scratch.

Beyond weight space similarity, we also examine the behaviors of generated models and their nearest training models. We compare the decision boundaries for classification models and the reconstructed 3D shapes for neural field models. In both cases, we find that these generated models, which are very close to training models in weight space, also exhibit highly similar *outputs* as the training ones.

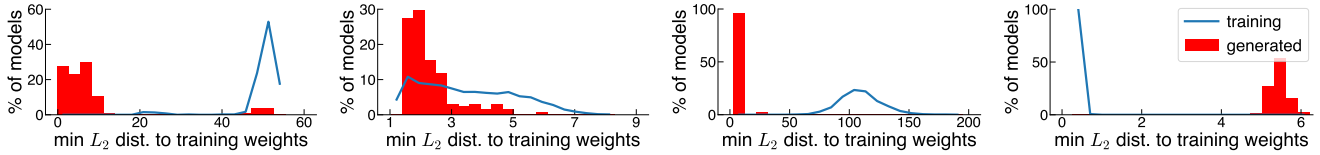
Further, we show that current generative modeling methods offer no advantage over simple baselines for creating new model weights, in terms of producing models that differ from training checkpoints in behavior while maintaining model performance. These baselines generate new weights by adding Gaussian noise to training weights or interpolating between them. To quantify how novel a generated model's behavior is relative to the behaviors of the training models, we compute a similarity metric for models based on their overlap in prediction errors on the test set.

In summary, our analysis reveals *clear patterns of memorization in almost all generated checkpoints* from current methods, both in weight space and model behavior. We find that the generated weights largely replicate or interpolate the training weight data. This stands in stark contrast to the versatility and strong generalization capabilities [33, 45] of image generation models. As generative modeling continues



(a) Hyper-Representations [43] (b) G.pt [35] (c) HyperDiffusion [12] (d) P-diff [50]

Figure 1. **Generated weights highly resemble training weights**. For each method, we display three heatmaps, showing weight values for 64 randomly selected parameter indices. In each heatmap, the top row shows a random generated checkpoint, and the three rows below the red line show its three nearest training checkpoints. At least one training checkpoint is nearly identical to each generated checkpoint.



(a) Hyper-Representations (b) G.pt (c) HyperDiffusion (d) P-diff

Figure 2. **Generated checkpoints are closer to training checkpoints than training checkpoints are to one another**, except for p-diff. This indicates that generated weights have lower novelty than a new model trained from scratch. The red histograms and blue curves represent the distributions of the L_2 distances to nearest training checkpoints (excluding self-comparisons).

to expand into new domains and modalities [9, 40, 54], our findings highlight the importance of evaluating memorization in generated outputs, beyond standard quality metrics.

2. Background

Generative models are recently used to synthesize network weights, producing models that perform comparably to those from standard training, without additional gradient-based optimization. In this study, we analyze four representative methods under their primary experimental setups.

Hyper-Representations [42, 43] generate weights using an autoencoder trained on checkpoints from classification models with identical architectures but different initializations. After training, kernel density estimation (KDE) is applied to the latents of the best-performing checkpoints. New weights are then generated by sampling a latent vector from the KDE-estimated distribution and decoding it.

G.pt [35] is a conditional diffusion model that can generate new weights for a predefined architecture, given input weights and a target loss for the generated weights. It is trained on millions of model checkpoints from tens of thousands of training runs, each paired with corresponding test losses. Once trained, G.pt produces effective weights from randomly initialized weights and a minimal, fixed target loss.

HyperDiffusion [12] trains an unconditional diffusion model on neural field MLPs that represent 3D or 4D shapes [5]. New shapes are generated by sampling a new set of MLP weights from the diffusion model and reconstructing the mesh represented by the MLP.

P-diff [50] trains an unconditional latent diffusion model on 300 neural network checkpoints to generate new weights. These checkpoints are saved at consecutive steps during an additional training epoch of the same base image classification model, after it has converged.

3. Memorization in Weight Generation

To evaluate the novelty of generated model weights, we compare them to the original weights used to train the weight generation models, analyzing both their weight values and model behaviors in comparison with various baselines.

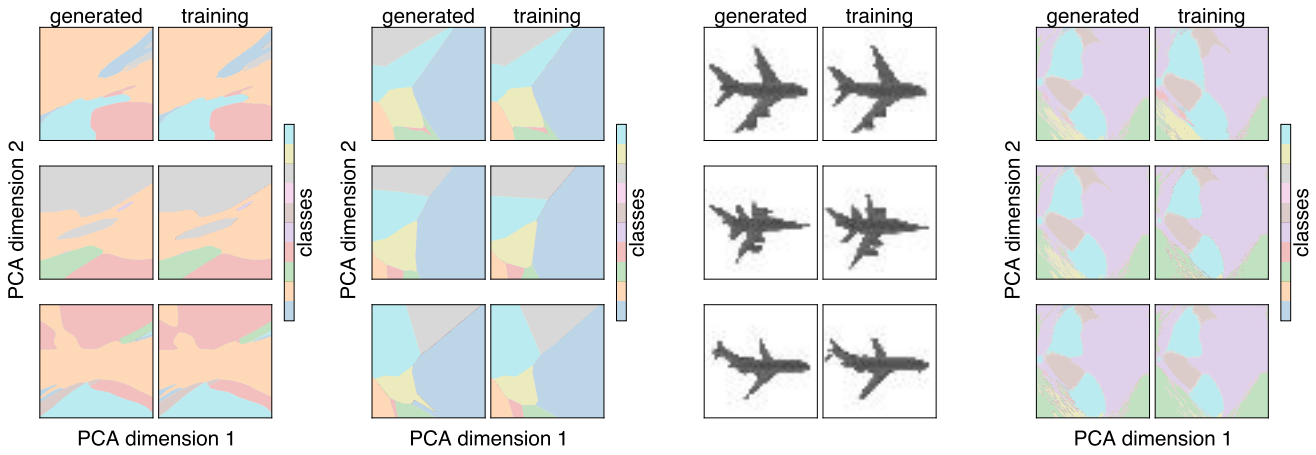
3.1. Memorization in Weight Space

A natural first step in evaluating the novelty of generated weights is to find the nearest training weights to each generated checkpoint under L_2 distance, and check for replications. However, depending on the method, permutations of weight matrices in training checkpoints or autoencoder reconstructions of training weights must also be considered.

For methods (*e.g.*, G.pt) that apply function-preserving weight permutation to augment data during training, we enumerate all possible permutations of training weights to identify the closest match for each generated checkpoint. Meanwhile, Hyper-Representations' autoencoder cannot accurately reconstruct training weights, degrading model accuracy by 13.9% on average. Thus, we compare its generated weights with the reconstructed training weights instead.

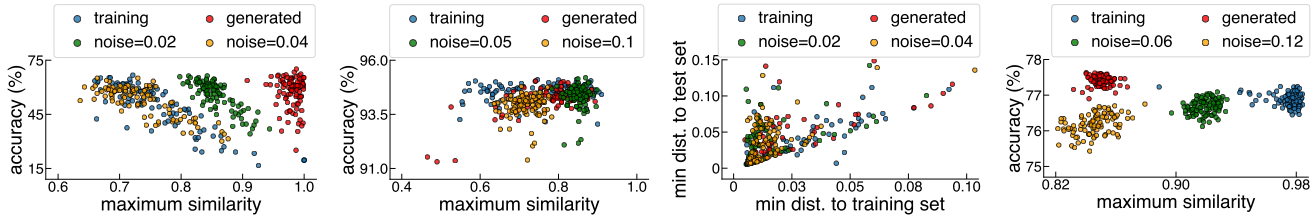
Weight heatmap. For each weight generation method, we visually inspect the three nearest training checkpoints for each of three randomly selected generated checkpoints using a heatmap of weights, shown in Figure 1 (more examples in Appendix C.1). We observe that, for all sampled generated checkpoints across all methods, there is always at least one training checkpoint that closely resembles the generated checkpoint. Further, all of p-diff's training and generated checkpoints have nearly identical weight values. This potentially results from p-diff's training checkpoints being saved consecutively from the same training run.

Distance to training weights. In addition to visually inspecting weight values, we identify quantitative trends that



(a) Hyper-Representations (b) G.pt (c) HyperDiffusion (d) P-diff

Figure 3. **Generated models produce highly similar outputs to their nearest training models**. Each row shows the decision boundaries or reconstructed 3D shapes of a randomly selected generated checkpoint ("generated") and its nearest training checkpoint ("training"). For p-diff models trained on CIFAR-100, decision boundaries are shown for ten randomly selected classes.



(a) Hyper-Representations (b) G.pt (c) HyperDiffusion (d) P-diff

Figure 4. **Weight generation methods do not outperform noise addition in the accuracy-novelty trade-off**, except for p-diff. The novelty of a generated model is measured by its maximum error similarity or minimum mesh distance to training checkpoints, where lower maximum similarity and higher minimum distance are preferred. We show 100 random samples for each checkpoint type.

differentiate sampling a generated checkpoint from training a new model using standard gradient-based optimization.

For each training and generated checkpoint, we compute its L_2 distance to the nearest training checkpoint and present the distance distributions in Figure 2. For all methods except p-diff, the generated checkpoints are much closer to the training checkpoints than training checkpoints are to one another. This indicates that the weight generation methods produce outputs with lower novelty than training a new model from scratch. We note that the training checkpoints used in these methods are saved from *independent* training runs.

For p-diff, we observed that the training checkpoints are much closer to each other than generated checkpoints are to their nearest training checkpoints. The low distances between training checkpoints may be expected, since they are saved from the *same* training run at *consecutive* steps.

3.2. Memorization in Model Behaviors

In Section 3.1, we showed that generated weights highly resemble the training weights. However, similar weights can still yield different behaviors. Here, we compare the behaviors of generated models to the behaviors of their nearest training models in weight space. We also assess whether generative modeling methods differ from a simple noise-addition baseline for creating new weights.

Visualizations of model outputs. To understand the behaviors of generated image classification models, we project the high-dimensional image datasets onto two principal components, and then visualize the decision boundaries of these classifiers. For HyperDiffusion, we reconstruct 3D shapes from the neural field models it generates.

For each method, we randomly select three generated checkpoints (additional examples in Appendix C.2) and identify their nearest training checkpoints in weight space using the L_2 metric, as in Section 3.1. Figure 3 presents the corresponding decision boundaries or 3D shapes. We find that generated models and their nearest training models produce highly similar predictions in image classification, as indicated by the nearly identical decision regions across all class labels. Similarly, the neural field models generated by HyperDiffusion also reconstruct to nearly identical 3D shapes as training ones, with visible differences only in minor details.

Metric for novelty. For generated weights to represent generalization, they need to behave sufficiently differently from training weights while maintaining high performance. To quantify the novelty of a generated classification model checkpoint, we adopt the model similarity metric from Wang et al. [50], which measures the Intersection over Union (IoU) of incorrect predictions between two model checkpoints. The formal definition of this metric is in Appendix D.1.

To assess a checkpoint's novelty, we compute its similarity with each training checkpoint and take the *maximum sim-*

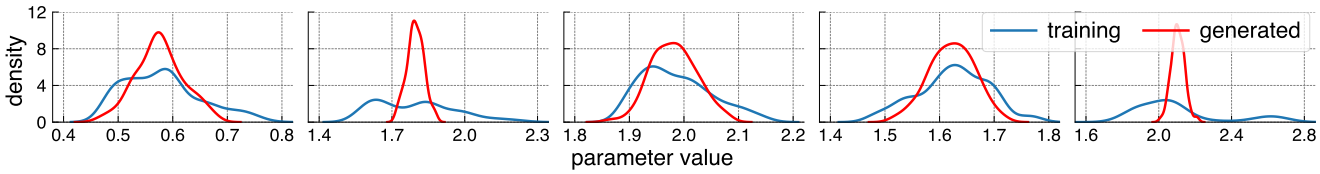


Figure 5. **Distributions of five random parameters** from the weight matrix of the first layer in p-diff checkpoints. The generated weights are centered around the mean of the training weights, suggesting they may be interpolations. More details are in Appendix E.2.

ilarity. A lower *maximum similarity* value indicates greater novelty, as it means the generated model's error patterns differ more from all training models.

For HyperDiffusion, which generates neural field models rather than classifiers, we use Chamfer Distance (CD), a standard metric for 3D shapes. A lower minimum CD to the *test* shapes indicates better shape quality, analogous to higher classification accuracy. A higher minimum CD to the *training* shapes suggests greater novelty, akin to lower maximum similarity in classification models.

Noise-addition baseline. We compare the accuracy and maximum similarity of the generated checkpoints against a baseline that simply adds Gaussian noise to training weights. The weight generation methods are considered superior if, at the same level of novelty relative to training models, they produce models with better performances than noise addition. Figure 4 shows the accuracy and maximum similarity distributions for training, generated, and noise-added models. For each weight generation method, the noise amplitudes are chosen so that the maximum similarity of noise-added models matches that of generated models.

Accuracy-novelty trade-off. As shown in Figure 4, for G.pt and Hyper-Representations, noise-added models often achieve comparable or even higher accuracy than generated models at the same maximum similarity to training models. Similarly, for HyperDiffusion, the distributions of the minimum CD to training and test shapes show no significant difference between generated and noise-added MLPs. These results suggest that the weight generation methods may *not* offer further benefits than simply adding noise to the training weights. An exception is p-diff, where generated models achieve a better trade-off between maximum similarity and accuracy than noise-added models.

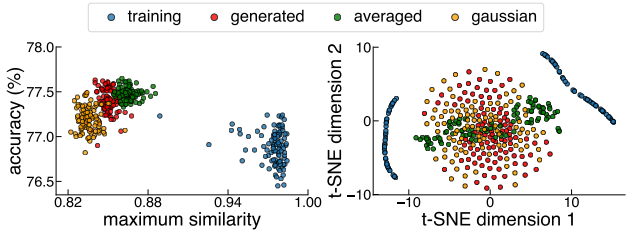


Figure 6. **P-diff generates weights with model behaviors (left) and values (right) similar to interpolations of training weights**. We compare them to two baselines: averaging training weights ("averaged") and sampling from a Gaussian distribution fitted to training weights ("gaussian"). Model behavior is evaluated via accuracy and maximum similarity; t-SNE visualizes weight values.

3.3. Understanding P-diff's Accuracy-Novelty Trade-off

As observed in Section 3.2, different from the other methods, p-diff achieves a better trade-off between the novelty and accuracy of generated models compared to the noise-addition baseline. The generated weights even surpass the training weights in accuracy (Figure 4d).

Observations. To investigate this, we examine the distribution of parameter values in generated and training models in Figure 5. The generated weight values for parameters tend to concentrate around the average of the training values. Averaging the weights of multiple models fine-tuned from the same base model is known to lead to improved accuracy [51]. Thus, p-diff may achieve higher accuracy in its generated models by interpolating its training checkpoints.

Interpolation baselines. To explore this hypothesis, we generate new models using two approaches that approximate the interpolations of the training checkpoints: (1) averaging the weights of 16 randomly selected training checkpoints ("average") and (2) fitting a Gaussian distribution to the training weight values in each parameter dimension and sampling from these distributions ("gaussian").

Behaviors and weights. The left subplot of Figure 6 shows that the accuracy and maximum similarity of the interpolation models closely match those of p-diff. The right subplot of Figure 6 visualizes the weight distributions using t-SNE [49]. The weights generated by p-diff are close to weights from the interpolation baselines, further suggesting that p-diff may primarily interpolate between training checkpoints. We note that this interpolation occurs within a very narrow range, as detailed in Appendix E.

4. Discussion and Conclusion

We provide evidence that current generative modeling methods for weights primarily memorize the training data rather than generating truly novel network weights. Our analysis shows that generated checkpoints are close replications or interpolations of training checkpoints, exhibiting similar weight values and model behaviors. Notably, our analysis of model behaviors shows that the generation methods offer no clear advantage over simple baselines to create new models.

Our findings emphasize the need for careful evaluation of memorization in generative modeling, particularly as these models expand to new modalities and tasks. We hope this work can inspire future research to address the memorization issues, and further explore the practical applications of generative models for weight data and beyond.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 1

[2] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *NeurIPS*, 2019. 20, 21

[3] Andrew Brock, Theo Lim, J.M. Ritchie, and Nick Weston. SMASH: One-shot model architecture search through hypernetworks. In *ICLR*, 2018. 8

[4] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators, 2024. 1

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 2

[6] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural computation*, 1993. 8

[7] Minshuo Chen, Kaixuan Huang, Tuo Zhao, and Mengdi Wang. Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data. In *ICML*, 2023. 21

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 10

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *IJRR*, 2023. 2

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshops*, 2020. 10, 15

[11] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 9

[12] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *ICCV*, 2023. 1, 2

[13] Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *TMLR*, 2025. 9

[14] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 8

[15] Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*. 1990. 8

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 19

[17] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 1

[18] Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *ICLR*, 2024. 9

[19] Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. In *NeurIPS*, 2024. 10, 11

[20] Boris Knyazev, Michal Drozdzal, Graham W Taylor, and Adriana Romero Soriano. Parameter prediction for unseen deep architectures. In *NeurIPS*, 2021. 8

[21] Boris Knyazev, Doha Hwang, and Simon Lacoste-Julien. Can we scale transformers to predict parameters of diverse imagenet models? In *ICML*, 2023. 8

[22] Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, Efstratios Gavves, Cees G. M. Snoek, and David W. Zhang. Graph neural networks for learning equivariant representations of neural networks. In *ICLR*, 2024. 10, 11

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1

[24] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. In *NeurIPS*, 1989. 10

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 1

[26] Elizaveta Levina and Peter Bickel. Maximum likelihood estimation of intrinsic dimension. In *NeurIPS*, 2004. 20

[27] Derek Lim, Haggai Maron, Marc T Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. In *ICLR*, 2024. 10, 11

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 15

[29] David J.C. MacKay and Zoubin Ghahramani. Comments on 'maximum likelihood estimation of intrinsic dimension' by e. levina and p. bickel (2004), 2005. 20

[30] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011. 1

[31] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 9

[32] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian conference on computer vision, graphics & image processing*, 2008. 19

[33] Maya Okawa, Ekdeep S Lubana, Robert Dick, and Hidenori Tanaka. Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task. In *NeurIPS*, 2024. 1

[34] Kazusato Oko, Shunta Akiyama, and Taiji Suzuki. Diffusion models are minimax optimal distribution estimators. In *ICML*, 2023. 21

[35] 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. 1, 2, 10

[36] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017. 9

[37] Ed Pizzi, Sreya Dutta Roy, Sugosh Nagavara Ravindra, Priya Goyal, and Matthijs Douze. A self-supervised descriptor for image copy detection. In *CVPR*, 2022. 19

[38] Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *ICLR*, 2021. 20

[39] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 1

[40] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, et al. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 2021. 2

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1

[42] Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. In *NeurIPS*, 2021. 2, 20

[43] Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. In *NeurIPS*, 2022. 1, 2, 10, 20

[44] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Understanding and mitigating copying in diffusion models. In *NeurIPS*, 2023. 9

[45] Gowthami Somepalli, Anubhav Gupta, Kamal Gupta, Shramay Palta, Micah Goldblum, Jonas Geiping, Abhinav Shrivastava, and Tom Goldstein. Measuring style similarity in diffusion models. *arXiv preprint arXiv:2404.01292*, 2024. 1

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 9

[47] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 15

[48] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 9

[49] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008. 4

[50] Kai Wang, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024. 1, 2, 3, 14

[51] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022. 4

[52] Fan Yin, Jayanth Srinivasa, and Kai-Wei Chang. Characterizing truthfulness in large language model generations with local intrinsic dimension. In *ICML*, 2024. 20, 21

[53] TaeHo Yoon, Joo Young Choi, Sehyun Kwon, and Ernest K Ryu. Diffusion probabilistic models generalize when they fail to memorize. In *ICML workshop on structured probabilistic inference & generative modeling*, 2023. 9

[54] Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Zilong Wang, Aliaksandra Shysheya, Jonathan Crabbé, Shoko Ueda, et al. A generative model for inorganic materials design. *Nature*, 2025. 2

[55] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *ICLR*, 2019. 8