Self-Supervised Representation Learning on Neural Network Weights for Model Characteristic Prediction

Anonymous Author(s)
Affiliation
Address
email

Abstract

Self-supervised learning approaches proved to be particularly valuable for extracting task-relevant, useful and information-preserving representations. Neural Networks (NNs) are widely applied, yet their weight space is still not fully understood. Therefore, we propose self-supervised representation learning on the weights of populations of trained NNs from which we predict NN characteristics. To that end, we introduce domain specific data augmentations and an adapted attention architecture. Our empirical evaluation demonstrates that self-supervised representation learning in this domain helps to faithfully recover useful NN model properties. We show that learned representations for trained NNs outperform prior work for the tasks of hyper-parameter, test accuracy, and generalization gap prediction and transfer to out-of-distribution settings.

1 Introduction

Self-supervised and unsupervised representation learning approaches offer task-relevant, function-serving and information-preserving data representations [Bengio et al., 2015, Chen et al., 2020]. Such methods proved to be particularly valuable in helping to gain insights, uncovering latent factors, finding grouping, or unveiling intrinsic structure within the data. Vincent et al. [2010], Bengio et al. [2012], Lee et al. [2015], Bengio et al. [2015], LeCun et al. [2015] used them for data exploration, dimensionality reduction, reconstruction and prediction tasks in image, video, audio, and text data.

Only recently, researchers scratched the surface on related studies for populations of trained NNs (referred to as *model zoos* [McInnes et al., 2018, Unterthiner et al., 2020, Eilertsen et al., 2020]). Insights about populations of trained NNs weights and biases might help gain a better understanding of the fundamental characteristics that make NN successful, their learning dynamics, and their performance. Thus, we could improve NN hyper-parameters selection, perform test accuracy prediction without seeing the test data, estimate the generalization gap, and give a more reliable out-of-distribution estimate [Jiang et al., 2019, Yak et al., 2019, Hsu et al., 2020, Eilertsen et al., 2020, Unterthiner et al., 2020, Corneanu et al., 2020]. However, the weight space of a NNs is high-dimensional and, in does not offer straightforward and intuitive understanding of its latent factors. The space of trained NN is not well understood and little is known about the most valuable characteristic in trained NN models or how to obtain useful representations for them.

Most of the recent efforts in this direction focused on manually finding a set of features, carefully designing summaries and measures with discriminative and predictive power over a) NN activation responses triggered by data, or b) trained NNs weights and biases. Raghu et al. [2017], Morcos et al. [2018] and Kornblith et al. [2019] compared NN models by correlating their activations. Dinh et al. [2017] link model properties to characteristics of the loss surface around the minimum solution. Such methods provide insights about the learning process and have their merits, but they are expensive to compute, and can only compare two NN models at a time. Jiang et al. [2019] proposed a measure

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

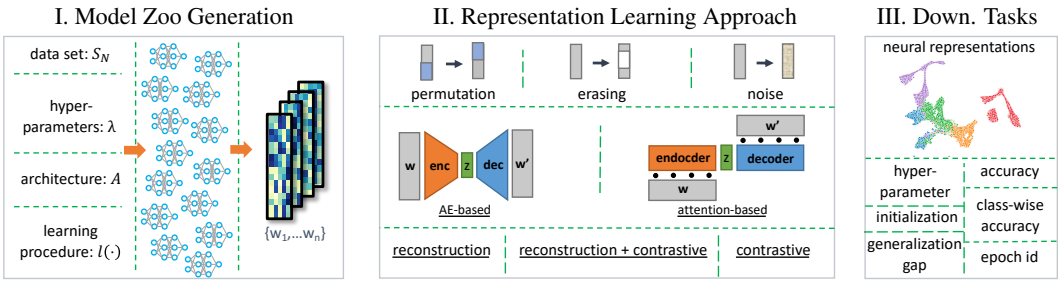

Figure 1: An overview of the proposed self-supervised representation learning approach. I. Populations of trained NNs form model zoos. NNs weights are vectorized. II. Neural representations are learned from the model zoos. III. Neural representations are evaluated on downstream tasks.

based on the concept of margin distribution, manually summarised over data points and across NN layers. Corneanu et al. [2020] used connectivity patterns in the NN activation, and computed topological summaries. Jiang et al. [2019], Corneanu et al. [2020] showed that such measures correlate with the generalization gap, but estimating them requires data and computation. Eilertsen et al. [2020] predicted NN hyper-parameters using weight statistics or 1D convolutional neural networks (CNN) directly from the weights. Unterthiner et al. [2020] proposed layer-wise statistics derived from the NN models to predict the test accuracy of NN model. Unfortunately, such features are limited by design as they disregard the order of weights and neglect the structural relations among weights.

In this paper, we introduce a novel approach. Instead of using hand-crafted features, or summaries, we propose self-supervised representation learning on populations of trained NNs weights and biases and predict the NN properties from its learned representation. We hypothesize that learned representations over populations of NNs can faithfully unveil properties about the NNs and the data on which they were trained. It is reasonable to assume that a self-supervised method can find latent variables common for diverse populations of NNs given sufficient capacity and efficient training of the learning model. Also, similarly to image, video, or audio data, a self-supervised approach could capture the relevant correlations in the populations of trained NNs with very little inductive bias while robust to variations and having task-agnostic utility [LeCun and Misra, 2021]. We adapt and examine different NN architectures focusing on an attention-based module to model associations between the different weights in a single NN. We apply self-supervised learning with reconstruction [Goodfellow et al., 2016b] and contrastive losses [Chen et al., 2020]. So that we can preserve the distinctive information about trained NNs and compactly relate to their common properties without relying on supervisory feedback from weak labels. To enable efficient training, we further propose novel augmentation methods. We introduce structure preserving permutations as augmentations, which take into account the structural symmetries within the NN weights that we find necessary and crucial for learning. We also adapt erasing [Zhong et al., 2020] and noise [Goodfellow et al., 2016b] as augmentations for NN weights. We give an overview for our learning approach in Figure 1.

To validate our hypothesis, we perform extensive numerical experiments over different populations of trained NN models. To do so, we use publicly available as well as generate and publish NN model zoos. In contrast to [Unterthiner et al., 2020, Eilertsen et al., 2020], our zoos contain models with different initialization points and diverse configurations, and include dense number of checkpoints, *i.e.*, model versions for its evolving points during training. Our ablation study confirms that the various factors for generating a population of trained NNs play a vital role in how and which properties are recoverable for trained NNs [Unterthiner et al., 2020]. In addition, the relation between generating factors and model zoo diversity reveals that seed variation for the trained NNs in the zoos is beneficial and adds another perspective when recovering NNs' properties.

Similarly, as in the common representation learning setups [Bengio et al., 2015], we use linear probing [Grill et al., 2020] and evaluate multiple downstream tasks to see the potential of our approach. We find that not only learned NNs but also their *neural representations*[1] contain the latent footprint of their training data [Sablayrolles et al., 2020]. We show that our task-agnostic neural representations have high utility on tasks like hyper-parameters, test accuracy, and generalization gap prediction. Furthermore, we demonstrate improved performance compared to the state-of-the-art on the previously mentioned tasks and outlay the advantages in an out-of-distribution prediction.

[1]By neural representation, we refer to a learned representation from a population of NNs given a model zoo.

Figure 2: Trained NN weights are the input sequence to a transformer. **Left**. Each element in the sequence represents the weight that connects two neurons at two different layers. **Right**. Each element in the sequence represents the set of weights related to one neuron.

Figure 3: Multi-head attention-based encoder. **Left**. Regular sequence-to-sequence translation, each element of the output sequence is used. **Right**. An additional *compression token* is added to the sequence. From the output sequence, only the compression token is taken.

2 Model Zoos and Augmentations

Model Zoo We denote by $\mathcal{D}$ a data set that contains data samples with their corresponding labels. We denote as λ the set of hyper-parameters used for training (*e.g.* loss function, optimizer, learning rate, weight initialization, batch-size, epochs). We define with A the specific NN architecture and with $l(.)$ the learning procedure. Training under different prescribed configurations $\{\mathcal{D}, \lambda, A, l(.)\}$ results in a population of NNs which we refer to as *model zoo*. We convert the weights and biases of all NNs in the model zoo into a vectorized form. In the resulting set $\mathcal{W} = \{\mathbf{w}_1, ..., \mathbf{w}_M\}$, $\mathbf{w}_i$ denotes the flattened vector of dimension N, representing the weights and biases for one trained NN model.

Augmentations. In many machine learning tasks, data augmentation helps to learn robust models, both by increasing the number of training samples and preventing overfitting of strong features [Shorten and Khoshgoftaar, 2019]. We propose three methods to augment individual instances of our model zoos to support our self-supervised representation learning approach.

Neurons in dense layers can change position without changing the overall mapping of the network, if in-going and out-going connections are changed accordingly [Bishop, 2006]. The relation between equivalent versions of the same network translates to permutations of incoming weights with matrix $\mathbf{P}$ and transposed permutation $\mathbf{P}^{\mathrm{T}}$ of the outgoing weights ($\mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{I}$). Considering the output at layer $l+1$, with weight matrices $\mathbf{W}$, biases $\mathbf{b}$, activations $\mathbf{a}$ and activation function σ, we have

$$\mathbf{z}^{l+1} = \mathbf{W}^{l+1}\sigma(\mathbf{W}^{l}\mathbf{a}^{l-1} + \mathbf{b}^{l}) + \mathbf{b}^{l+1} = \hat{\mathbf{W}}^{l+1}\sigma(\hat{\mathbf{W}}^{l}\mathbf{a}^{l-1} + \hat{\mathbf{b}}^{l}) + \mathbf{b}^{l+1},$$

where $\hat{\mathbf{W}}^{l+1} = \mathbf{W}^{l+1}(\mathbf{P}^{l})^{\mathrm{T}}$, $\hat{\mathbf{W}}^{l} = \mathbf{P}^{l}\mathbf{W}^{l}$ and $\hat{\mathbf{b}}^{l} = \mathbf{P}^{l}\mathbf{b}^{l}$ are the permuted weight matrices and bias vector, respectively. The equivalences hold not only for the forward pass, but also for the backward pass and weight update[2]. The permutation can be applied also to kernels of convolution layers. While the *permutation augmentation* differs significantly from existing augmentation techniques, flips for images are similar, but specific instances from the set of possible permutations in the image domain. Empirically, we found the permutation augmentation crucial for our learning approach.

In computer vision and natural language processing, masking parts of the input has proven to be helpful for generalization [Devlin et al., 2019]. We adapt an approach of *random erasing* of sections in the vectorized forms of trained NN weights. Similarly as in [Zhong et al., 2020], we apply erasing augmentation with a probability p to an area that is randomly chosen with a lower and upper bounds b_{low} and b_{up}. In our experiments, we set $p = 0.5$, $b_{low} = 0.03$, $b_{up} = 0.3$ and erase with zeros. Adding *noise augmentation* is another way of altering the exact values of NN weights without overly affecting their mapping, and has long been used in other domains [Goodfellow et al., 2016b]. We note that noise and erasing as weight augmentations previously were not explored in this domain.

3 Neural Representation Learning

Commonly, self-supervised representation learning employs a discriminative or generative approach Grill et al. [2020]. We follow a discriminative modeling principle such as contrastive learning and borrow ideas from generative approaches such as auto-encoding to learn representations in an self-supervised fashion. We adopt an encoder-decoder setup where we use different losses.

Architectures and Self-Supervised Losses. We denote the encoder as $g_\theta(\mathbf{w}_i)$ its parameters as θ, and the neural representation with dimension L as $\mathbf{z}_i = g_\theta(\mathbf{w}_i)$. We denote the decoder as $h_\psi(\mathbf{z}_i)$, its parameters as ψ, and the reconstructed NN weights as $\hat{\mathbf{w}}_i = h_\psi(\mathbf{z}_i) = h_\psi(g_\theta(\mathbf{w}_i))$. To learn the

[2]Details, formal statements and proofs can be found in the Appendix

encoder and decoder parameters, we use a loss $\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_c$ composed of MSE reconstruction term $\mathcal{L}_{MSE} = \frac{1}{M}\sum_{i=1}^{M} \|\mathbf{w}_i - h_\psi(g_\theta(\mathbf{w}_i))\|_2^2$ and the NT_Xent loss Chen et al. [2020] term $\mathcal{L}_c$. We denote this architecture with its loss as $\mathrm{E}_c\mathrm{D}$. In contrastive learning, many methods prevented mode collapse by using negative samples. Our loss $\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_c$, contains a reconstruction term $\mathcal{L}_{MSE}$, which can be seen as a regulizer that prevents mode collapse. Therefore, we also experiment with replacing $\mathcal{L}_c$ in our loss with a modified contrastive term

$$\mathcal{L}_{c+} = \sum_i -\log\left(\exp(\mathrm{sim}(\mathbf{z}_i^j, \mathbf{z}_i^k)/\tau\right),$$

where each $\mathbf{z}_i$ is randomly augmented twice and forms the two *views* $\mathbf{z}_i^j$ and $\mathbf{z}_i^k$, while $\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T \mathbf{z}_j / ||\mathbf{z}_i|| ||\mathbf{z}_j||$ is the cosine similarity. We denote the encoder-decoder with the loss $\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{c+}$ as $\mathrm{E}_{c+}\mathrm{D}$. We also experiment with encoder-decoder and a reconstruction loss $\mathcal{L} = \mathcal{L}_{MSE}$ and denote it as ED, and consider only an encoder with a contrastive loss $\mathcal{L} = \mathcal{L}_c$ and denote it as E_c. In all of the architectures, we embed the neural representation $\mathbf{z}_i$ in a low dimensional space, $L < N$. We would like to point out that Eilertsen et al. [2020] and Unterthiner et al. [2020] manually selected statistics and summaries. In our approach, $\mathbf{z}_i$ can also be seen as a summary, which is learned to compactly extract the relevant and meaningful information from the weights.

Attention Module. Our encoder and decoder pairs are symmetrical and of the same type. We apply fully connected feed-forward networks (FFN) as baselines. As there is no intuition on good inductive biases in the weight space, we further use multi-head self-attention modules (Att) [Vaswani et al., 2017] as an architecture with very little inductive bias. In the multi-head self-attention module, we apply learned position encodings to preserve structural information [Dosovitskiy et al., 2020]. We propose two methods to encode the weights into a sequence (Figure 2). In the first method, we encode each weight as a token in the input sequence. In the second method, we linearly transform the weights of one neuron or kernel and use it as a token. Further, we apply two variants to compress representations in the latent space (Figure 3). In the first variant, we aggregate the output sequence of the transformer and linearly compress it to a neural representation $\mathbf{z}_i$. In the second variant, similarly to Devlin et al. [2019], Zhong et al. [2020], we add a learned token to the input sequence that we dub *compression token*. After passing the input sequence trough the transformer, only the compression token from the output sequence is linearly compressed to a neural representation $\mathbf{z}_i$. Without the compression token, the information is distributed across the output sequence. The compression token aggregates information while the input sequence passes through the transformer. Its dimension is directly tied to the dimension of the value tokens and so its capacity affects the overall memory consumption.

Downstream Tasks. We use linear probing [Grill et al., 2020] as a proxy to evaluate the utility of the learned neural representations. As downstream tasks we use accuracy prediction (Acc), generalization gap (GGap), epoch prediction (Eph) as proxy to model versioning, F-Score [Goodfellow et al., 2016a] prediction (F_c), learning rate (LR), ℓ_2-regularization (ℓ_2-reg), dropout (Drop) and training data fraction (TF). Using such targets, we solve a regression problem and measure the R^2 score [Wright, 1921]. We also evaluate for hyper-parameters prediction tasks, like the activation function (Act), optimizer (Opt), initialization method (Init). Here, we train a linear perceptron by minimizing a cross entropy loss [Goodfellow et al., 2016b] and measure the prediction accuracy.

4 Empirical Evaluation

4.1 Model Zoos

Publicly Available Model Zoos. Unterthiner et al. [2020] introduced model zoos of CNNs with 4970 parameters trained on MNIST [LeCun and Cortes], Fashion-MNIST [Xiao et al., 2017], CIFAR10 [Krizhevsky et al.] and SVHN [Netzer et al., 2011], and made them available under CC BY 4.0. We refer to these as `MNIST-HYP`, `FASHION-HYP`, `CIFAR10-HYP` and `SVHN-HYP`. We categorize these zoos as *large* due to their number of parameters. In their model zoo creation, the CNN architecture and seed were fixed, while the activation function, initialization method, optimizer, learning rate, ℓ_2 regularization, dropout and the train data fraction were varied between the models.

Our Model Zoos. We hypothesize that using only one fixed seed may limit the variation in characteristics of a zoo. To address that, we train zoos where we also vary the seed and perform an ablation

	TETRIS-SEED						
	Rec.	Eph	Acc	F_{C0}	F_{C1}	F_{C2}	F_{C3}
E_c	-	40.3	42.4	38.2	27.3	46.5	33.4
ED	93.5	**94.4**	81.2	59.8	63.5	55.6	59.2
E_cD	76.8	92.1	**85.7**	**66.0**	**70.0**	**66.3**	**81.7**
E_{c+}D	**94.3**	91.7	71.6	52.3	58.9	42.5	54.2

Table 1: Ablation results over self-supervised learning losses. All models implemented with attention-based reference architecture. All values are given in %.

	TETRIS-SEED						
	Rec.	Eph	Acc	F_{C_0}	F_{C_1}	F_{C_2}	F_{C_3}
FF	11.7	73.8	69.6	51.5	52.2	52.4	59.4
Att_W	64.1	93.6	79.6	63.5	**70.0**	53.7	65.8
Att_{W+t}	20.8	56.9	52.8	44.5	46.9	43.8	41.8
Att_N	**82.2**	87.3	80.5	63.5	65.1	63.0	66.5
Att_{N+t}	76.8	**92.1**	**85.7**	**66.0**	**70.0**	**66.3**	**81.7**

Table 2: Ablation results in % under E_cD setup. We use feed-forward FF and attention-based variants with weight and neuron encoding Att_W and Att_N each with $+t$ and without compress. token.

study below. As a toy example, we first create a 4x4 grey-scaled image data set that we call *tetris* by using four tetris shapes.

	TETRIS-SEED							
	-	P	E	N	P,E	P,N	E,N	P,E,N
ED	79.4	**93.8**	75.1	79.7	93.5	93.8	75.3	93.5
E_cD	38.3	48.2	53.9	19.7	79.1	52.6	42.9	81.1
E_{c+}D	**81.0**	**93.8**	**83.2**	**82.0**	**94.3**	**94.0**	**83.8**	**94.3**

Table 3: R^2 reconstruction score given in % in the augmentation ablation for different attention-based architectures. We use: permutation (P), erasing (E) and noise (N) augmentation.

We introduce two zoos, which we call `TETRIS-SEED` and `TETRIS-HYP`, which we group under *small*. Both zoos contain FFN with two layers and have a total number of 100 learnable parameters. In the `TETRIS-SEED` zoo, we fix all hyper-parameters and vary only the seed to cover a broad range of the weight space. The `TETRIS-SEED` zoo contains 1000 models that are trained for 75 epochs. To enrich the diversity of the models, the `TETRIS-HYP` zoo contains FFNs, which vary in activation function [`tanh`, `relu`], the initialization method [`uniform`, `normal`, `kaiming normal`, `kaiming uniform`, `xavier normal`, `xavier uniform`] and the learning rate [$1e$-3, $1e$-4, $1e$-5]. In addition, each combination is trained with 100 different seeds. Out of the 3600 models in total, we have successfully trained 2900 for 75 epochs - the remainders crashed and are disregarded. Similarly to `TETRIS-SEED`, we further create zoos of CNN models with 2464 parameters, each using the MNIST and Fashion-MNIST data sets, called `MNIST-SEED` and `FASHION-seed` and grouped them as *medium*. To maximize the coverage of the weight space, we again initialize models with seeds 1-1000 [3].

Model Zoo Generating Factors. Prior work discusses the impact of random seeds on properties of model zoos. While Yak et al. [2019] use multiple random seeds for the same hyper-parameter configuration, Unterthiner et al. [2020] explicitly argue against that to prevent information leakage between samples. To disentangle the generating factors (seeds and hyper-parameters) and model properties, we have created five zoos with approximately the same number of CNN models trained on MNIST [3]. `MNIST-SEED` varies only the random seed (1-1000), `MNIST-HYP-1-FIX-SEED` varies the hyper-parameters with one fixed seed per configuration (similarly to [Unterthiner et al., 2020]). To decouple the hyper-parameter configuration from one specific seed, `MNIST-HYP-1-RAND-SEED` draws 1 random seeds for each hyper-parameter configuration. To investigate the influence of repeated configurations, in `MNIST-HYP-5-FIX-SEED` and `MNIST-HYP-5-RAND-SEED` for each hyper-parameter configurations we add 5 models with different fixed and random seeds, respectively.

4.2 Training and Testing Setup

Architectures. We evaluate our approach under different types of architectures, including E_c, ED, E_cD and E_{c+}D, see Section 3. As encoders E and decoders D, we experimented with the architectures introduced in Section 3. The encoder E and decoders D in the FFN baseline are symmetrical 10 [FC-ReLU]-layers each and linearly reduce dimensionality for the input to the latent space. Considering the attention-based encoder and decoder, on the `TETRIS-SEED` and `TETRIS-HYP` zoos, we used 2 attention blocks with 1 attention head each, token dimensions of 128 and FC layers in the attention module of dimension 512. On the larger zoos, we use up to 4 attention heads in 4 attention blocks, token dimensions of up to size 800 and FC layers in the attention module of dimension 1000.

[3] Full details on the generation of the zoos can be found in the Appendix

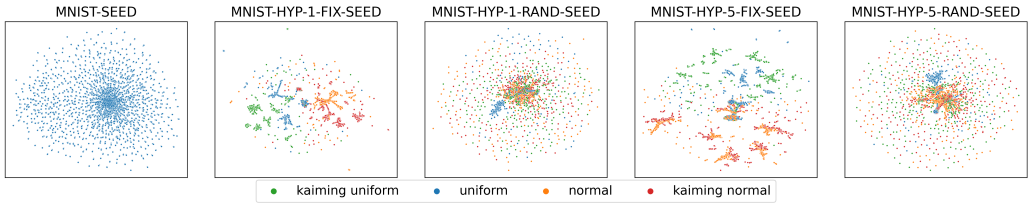

Figure 4: UMAP dimensionality reduction for NN model zoos created with different generating factors. Colors represent initialization methods. Compare numerical results in Table 4.

	MNIST-SEED			MNIST-HYP-1-FIX-SEED			MNIST-HYP-1-RAND-SEED			MNIST-HYP-5-FIX-SEED			MNIST-HYP-5-RAND-SEED		
VAR	.234			.155			.152			.091			.092		
VAR_c	.234			.101			.164			.094			.100		
R_{tr}	78.5			90.9			78.5			86.5			81.4		
R_{tes}	59.5			79.0			57.9			78.8			63.7		
	W	s(W)	E_{c+}D	W	s(W)	E_{c+}D	W	s(W)	E_{c+}D	W	s(W)	E_{c+}D	W	s(W)	E_{c+}D
EPH	54.9	**97.7**	95.9	03.1	06.2	**06.8**	-67	**04.2**	-14	01.1	**12.4**	03.4	-20	**07.8**	-03.
ACC	82.8	**98.4**	98.1	07.7	74.4	**87.0**	$<\kappa$	**56.6**	55.5	21.5	66.1	**78.7**	$<\kappa$	**69.1**	58.1
GGAP	$<\kappa$	**45.3**	40.4	-72	35.8	**49.4**	$<\kappa$	**24.7**	01.8	$<\kappa$	33.9	**35.7**	$<\kappa$	**41.5**	10.8
INIT	–	–	–	**88.5**	67.3	85.6	39.2	**58.2**	44.8	**82.5**	61.3	76.9	37.6	**55.4**	40.1

Table 4: R^2 score in %. Results about the impact of the generating factors for the model zoos. VAR is the variance of the weights. VAR_c denotes the mean of the variances for groups of samples with shared initialization method and activation function. R_{tr} and R_{tes} are the reconstruction R^2 of train and test split on a reference architecture after 1750 epochs. $\kappa = -170$.

Neural Representation Learning and Downstream Tasks. We apply the proposed data augmentation methods for representation learning (see Section 2). We run our representation learning algorithms for up to 2500 epochs, using the adam optimizer [Kingma and Ba, 2014], a learning rate of 1e-4, weight decay of 1e-9, dropout of 0.1 percent and batch-sizes of 500. In all of our experiments, we use 85% of the model zoos for training and 15% for testing. We use checkpoints of all epochs, but ensure that samples from the same models are either in the train or in the test split of the zoo. As quality metric for the self-supervised learning, we track the reconstruction R^2 on the test split of the zoo. As a proxy for how much useful information is contained in the neural representation, we evaluate on downstream tasks as described in Section 3. To ensure numerical stability of the solution to the linear probing, we apply Tikhonov regularization [Tikhonov and Arsenin, 1977] with regularization parameter α in the range $[1e\text{-}5, 1e3]$ (we choose α by cross-validating over the R^2 score for the training split of the zoo) and report the R^2 score of the test split of the zoo. To minimize the cross entropy loss for the categorical hyper-parameter prediction, we use adam optimizer [Kingma and Ba, 2014] with learning rate of $1e\text{-}4$ and weight-decay of $1e\text{-}6$. The linear probing is applied to the same train-test splits as it is in our representation learning setup.

Out-of-Distribution Experiments. We follow a setup for out-of-distribution experiments similar to [Unterthiner et al., 2020]. We investigate how well the linear probing estimator computed over neural representations generalize to yet unseen data. Therefore, we use the zoos `MNIST-HYP`, `FASHION-HYP`, `CIFAR10-HYP` and `SVHN-HYP`. On each zoo, we apply our self-supervised approach to learn their corresponding neural representations and fit a linear probing estimator to each of them (in-distribution). We then apply both the neural representation mapper and the linear probing estimator of one zoo on the weights of the other zoos (out-of-distribution). The target ranges and distributions vary between the zoos. Linear probe prediction may preserve the relation between predictions, but include a bias. Therefore, we use Kendall's τ coefficient as performance metric, which is a measure of rank correlation. It measures the ordinal association between two measured quantities [Kendall, 1938].

Baselines, Computing Infrastructure and Run Time. As baseline, we use the model zoos (W). In addition, we also consider linear PCA (PCA_l), cosine similarity PCA (PCA_c), radial basis kernel PCA (PCA_r), and UMAP (U_m) [McInnes et al., 2018]. We further compare to layer-wise weight-statistics (mean, var, quintiles) s(W) as in [Unterthiner et al., 2020]. Similar features are used in [Eilertsen et al., 2020]. As computing hardware, we use half of the available resources from NVIDIA DGX2 station with 3.3GHz CPU and 1.5TB RAM memory, that has a total of 16 1.75GHz GPUs, each with 32GB memory. To create one small and medium zoo, it takes 1 to 2 days and 10 to 12 days, respectively.

	TETRIS-SEED						
	W	PCA_l	PCA_c	PCA_r	U_m	s(W)	E_cD
Eph	59.8	49.3	78.5	93.7	65.7	**96.4**	95.2
Acc	53.5	28.7	76.5	75.7	76.0	86.6	**87.0**
F_{C0}	30.6	11.9	54.8	50.7	50.3	62.2	**67.3**
F_{C1}	45.3	31.2	50.7	59.9	39.2	62.1	**68.2**
F_{C2}	47.3	12.5	42.6	40.3	46.9	60.9	**66.4**
F_{C3}	57.8	47.3	**75.2**	73.5	72.9	71.8	60.3

	TETRIS-HYP						
	W	PCA_l	PCA_c	PCA_r	U_m	s(W)	E_cD
Rec	–	79.7	43.4	-30	–	–	**91.0**
Eph	02.8	02.9	01.9	13.2	0.*	16.2	**17.2**
Acc	21.5	22.1	25.6	32.4	0.*	48.5	**60.8**
LR	00.0	00.0	00.2	44.0	00.0	**53.1**	49.0
Act	75.0	73.5	79.6	80.4	46.7	73.8	**87.6**
Init	38.0	36.9	36.1	36.5	29.6	48.1	**48.6**

Table 5: R^2 given in %. **Left**. Reconstruction, epoch, accuracy and class-wise F-scores prediction. **Right**. Epoch, accuracy, learning rate, seed, activation function and initialization prediction.

For one experiment over the small zoo it takes around 3 hours to learn the neural representation on a single GPU and evaluate on the downstream tasks. It takes approximately 1 day for the medium zoos and 2 to 3 days for the large scale zoos for the same experiment.

4.3 Results

Augmentation Ablation. To evaluate the impact of the proposed augmentations (Section 2) for our representation learning method, we present an ablation analysis, in which we measure R^2 for ED, E_cD and E_{c+}D[4]. We use 120 permutations, a probability of 0.5 for erasing the weights, and zero-mean noise with standard deviation 0.05 (see Table 3). We find the permutation augmentation to be necessary for generalization - particularly under higher compression ratios. The additional samples generated with the permutation appear to effectively prevent overfitting of the training set. Without the permutation augmentation, the test performance diverges after few training epochs. Erasing further improves test performance and allows for extended training without overfitting. The addition of noise yields inconsistent results and is difficult to tune, so, we omit it in our further experiments.

Architecture Ablation. The different architectures are compared in Table 1. The results show that within the set of used NN architectures for neural representation learning, the attention-based architectures learn considerably faster, yield lower reconstruction error and have the highest performance on the downstream tasks compared to the FFN-based architectures. We attribute this to the attention modules, which are able to reliably capture long-range relations on complex data. While tokenizing each weight individually (Att_W) is able to learn, the computational load is significant, even for a small zoo, due to the large number of tokens in the sequences. The memory load prevents the application of that encoding on larger zoos. We find that embedding all weights of one neuron (or convolutional kernel) to one token combined with compression tokens (Att_{Nt}) shows the overall best performance and scales to larger architectures. Compression tokens achieve higher performance. The dedicated token gathers information from all other tokens of the sequence in several attention layers. This appears to enable the neural representation to grasp more relevant information than linearly compressing the entire sequence. On the other hand, compression tokens are only an advantage, if their capacity is high enough, in particular higher than the bottleneck. On the large zoos, compression tokens could not be applied successfully. High capacity compression tokens required overall high token dimensions, which exceeded the available memory.

Self-Supervised Learning Ablation. In Table 2, we evaluate the usefulness of the self-supervised learning tasks (Section 3). The application of purely contrastive loss in E_c, appears not to provide useful representations for the downstream tasks. Among our losses, the combination of reconstruction with contrastive loss as in E_cD, provides neural representations $\mathbf{z}$ that have the best performance on the downstream tasks. The variation E_{c+}D, has the highest reconstruction accuracy. The addition of a contrastive loss helps to pronounce distinctiveness, so that the neural representations are both better at reconstructing the NN weights, as well as revealing properties about the NNs. When analyzing the impact of the compression ratio N/L in E_cD, we found that very high compression ratios hurt performance. E_cD could not be trained to satisfactory results on the medium and large model zoos. We therefore apply E_cD on the small, and E_{c+}D on the medium and large model zoos, which is less demanding and shows superior performance to ED on those zoos.

[4]We leave out E_c as it does not use a reconstruction loss

	MNIST-HYP			FASHION-HYP			CIFAR10-HYP			SVHN-HYP		
	W	S(W)	$E_{c+}D$	W	S(W)	$E_{c+}D$	W	S(W)	$E_{c+}D$	W	S(W)	$E_{c+}D$
EPH	21.6	32.1	**33.4**	21.0	32.8	**40.7**	18.7	29.4	**35.6**	14.6	37.2	**39.7**
ACC	72.7	81.9	**90.0**	66.4	79.4	**90.2**	73.5	83.2	**90.9**	79.5	82.1	**89.9**
GGAP	15.3	24.8	**32.9**	41.0	42.1	**53.7**	32.7	41.1	**55.4**	30.2	38.0	**48.9**
LR	08.4	32.5	**36.4**	29.4	35.6	**43.0**	20.8	30.5	**40.2**	20.3	33.0	**40.0**
ℓ_2-REG	08.4	15.6	**18.1**	06.9	16.4	**24.0**	02.0	13.6	**24.6**	04.8	12.4	**18.2**
DROP	23.3	18.8	**34.2**	21.9	21.9	**39.1**	10.2	15.7	**28.2**	04.4	14.0	**20.8**
TF	-1.2	07.4	**11.1**	-0.3	07.5	**13.0**	02.2	06.6	**17.0**	-3.9	08.3	**13.0**
ACT	**89.0**	83.7	86.9	**90.3**	83.4	87.8	**88.8**	80.3	86.1	**87.7**	79.7	84.0
INIT	**94.5**	72.8	93.1	**95.7**	77.5	94.9	**93.4**	75.3	92.8	**91.0**	72.8	88.4
OPT	**76.8**	67.0	72.1	**79.8**	68.8	76.0	**74.0**	67.6	72.1	**72.0**	69.4	69.9

Table 6: **Top 7 Rows**. R^2 score in % for Eph, Acc, GGap, LR ℓ_2-reg, Drop and TF prediction. **Bottom 3 Rows**. Accuracy score for Act, Init and Opt prediction.

Zoo Generating Factors Ablation. Figure 4 visualizes the weight of the zoos, which contain models of all epochs[5]. In table 4, we report numerical properties. Only changing the seeds appears to result in homogeneous development with very high correlation between $s(W)$ and the properties of the samples in the zoo. Varying the hyper-parameters reduces the correlation. With fixed seeds, visually, we observe clusters of models with shared initialization method and activation function. Quantitatively we obtain lower VAR_c and high predictive value of W for the initialization method. That seems a plausible outcome, given that the architecture and activation function determines the shape of the loss surface, while the seed and initialization method decide the starting point. Such clustering appear to facilitate representation learning, with high R^2 for reconstruction and high performance on downstream tasks. We observe similar properties in the zoos of [Unterthiner et al., 2020], see Appendix. Initializing models with random seeds visually and numerically disperses the clusters. While VAR between 1 fixed and 1 random seed is comparable, VAR_c is considerably smaller with fixed seed, the predictive value of W for the initialization methods drops significantly. Random seeds also appear to make both the reconstruction as well as NN property prediction more difficult. On the other hand, the results show that sharing the same hyper-parameter configuration across five seeds helps the representation learning and NN property prediction. Thus, we conclude that changing only the seeds results in models with very similar evolution during learning, while using one seed shared across models might create shortcuts in the weight space. Zoos that vary both appear to be most diverse and hardest for learning and NN property prediction.

Downstream Tasks. We learn and evaluate our neural representations on 11 different zoos: `TETRIS-SEED`, `TETRIS-HYP`, 5 variants of `MNIST`, `MNIST-HYP`, `FASHION-HYP`, `CIFAR10-HYP` and `SVHN-HYP`. We compare with multiple baselines and s(W). The results are shown in Tables 4, 5 and and 6. On all model zoos, neural representations learn useful features for the downstream tasks, which outperform the actual NN weights and biases, all of the baseline dimensionality reduction methods as well as s(W) on the small and large zoos. On the medium sized zoos, neural representations are on par with s(W). Our approach demonstrates that we can faithfully recover useful information. On the `TETRIS-SEED`, and `MNIST-SEED` model zoos, s(W) achieves high R^2 scores on all downstream tasks (see Table 5 left). As discussed above, these zoos contain a strong correlation between $s(W)$ and sample properties. Nonetheless, learned neural representations achieve higher R^2 scores on Acc, F_{C0-C3} and are competitive on Eph. On the `TETRIS-SEED` model zoo, we outperform the baseline methods. On `TETRIS-HYP`, the overall performance of all methods is the lowest across the downstream tasks compared to the other zoos (see Table 5 right). Here, too, neural representations have the highest R^2 score on all downstream tasks, except for the LR prediction. On `MNIST-HYP`, `FASHION-HYP`, `CIFAR10-HYP` and `SVHN-HYP`, neural representations outperform s(W) on all downstream tasks. They achieve higher R^2 score compared to W on the prediction of continuous hyper-parameters. On the categorical hyper-parameters, like activation, initialization method and optimizer, the weight space achieves the highest R^2 scores, see Table 6. We explain this with the small amount of variation in the model zoo, see Section 4.1, which allows to separate these properties in weight space.

[5]Further visualizations can be found in the Appendix

	MNIST-HYP			FASHION-HYP			SVHN-HYP			CIFAR10-HYP		
	W	S(W)	E_{c+}D	W	S(W)	E_{c+}D	W	S(W)	E_{c+}D	W	S(W)	E_{c+}D
MNIST-HYP	**.36**	.29	.35	.20	.10	**.32**	.13	**.24**	.19	-.02	-.05	**.05**
FASHION-HYP	-.03	**.07**	.04	.54	.48	**.55**	.06	**.14**	-.03	.08	.09	**.20**
SVHN-HYP	-.03	**.16**	**.16**	-.03	**.21**	.05	.44	.35	**.46**	-.02	.07	**.08**
CIFAR10-HYP	.11	.10	**.12**	.35	.35	**.45**	.14	.17	**.19**	**.41**	.30	**.41**

Table 7: Kendall's τ score for the generalization gap (GGap) prediction. We train estimators for each zoo (rows) and evaluate on all zoos (columns). The block diagonal elements contain the in-distribution prediction values. The remaining values are for out-of-distribution prediction.

Out-of-Distribution Prediction. Table 7 shows the out-of-distribution results for generalization gap prediction[6], which is a very challenging task. Neural representations outperform the baselines for Kendall's τ measure in the majority of the results. Surprisingly, W and $s(W)$ in this setup have similar performance across the used model zoos. Our approach has better scores in the 7 out of 12 results (the off diagonal elements in Table 7) and has 1 tie in the comparisons with W and $s(W)$. This verifies that our approach indeed preserves the distinctive information about trained NNs while compactly relating to their common properties, including the characteristics of the training data.

5 Related Work

There is ample research evaluating the structures of NNs by visualizing activations, *e.g.* [Zeiler and Fergus, 2014], which allow some insights in the patterns of, *e.g.* the kernels of CNNs. Other research evaluated networks by computing a degree of similarity between networks. Laakso and Cottrell [2000] compared the activations of NNs by a measure of "sameness". Li et al. [2015] computed correlations between the activations of different nets. Wang et al. [2018] tried to match the subspaces of the activation spaces in different networks [Johnson, 2019], which showed to be unreliable. Raghu et al. [2017], Morcos et al. [2018], Kornblith et al. [2019] applied correlation metrics to NN activations in order to study the learning dynamics. Jia et al. [2019] approximated the space of DNN activations with a convex hull. Jiang et al. [2019] also used activations to approximate the margin distribution and predict the generalization gap. Corneanu et al. [2020] proposed persistent homology by using a connectivity patterns in the NN activation, and compute topological summaries.

While previously mentioned related work studied measures defined on the activations for insights about the NN characteristics, the methods applied on populations of NN weights have not received much attention for the same purpose. Only recently, two publications attempted to exploit populations of NNs. Instead of classifying a classifier for hyper-parameters prediction [Eilertsen et al., 2020], we learn a general purpose neural representations in self-supervised fashion. Unterthiner et al. [2020] proposed layer-wise statistics derived from the NN models to predict the test accuracy of NN model. In our work, we model associations between the different weights in NN using an attention-based module. This helps us to learn representations that compactly extract the relevant and meaningful information while considering the correlations between the weights in an NN.

6 Conclusions

In this work, we present a novel approach to learn neural representations from the weights of neural networks. To that end, we proposed new augmentations, self-supervised learning losses and adapted multi-head attention-based architectures with suitable weight encoding for this domain. Further, we introduced several new model zoos and investigated their properties. We showed that not only learned neural networks but also their neural representations contain the latent footprint of their training data. We demonstrated high performance on downstream tasks, exceeding existing methods in hyper-parameters, test accuracy, and generalization gap prediction and showing the potential in an out-of-distribution setting.

[6]In Appendix, we also give results for other tasks, including epoch id and test accuracy prediction

References

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

Yoshua Bengio, Dong-Hyun Lee, Jörg Bornschein, and Zhouhan Lin. Towards biologically plausible deep learning. *CoRR*, abs/1502.04156, 2015.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/chen20j.html`.

C. Corneanu, M. Madadi, S. Escalera, and A. Martínez. Computing the testing error without a testing set. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2674–2682, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp Minima Can Generalize For Deep Nets. *arXiv:1703.04933 [cs]*, May 2017. URL `http://arxiv.org/abs/1703.04933`. arXiv: 1703.04933.

A. Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, M. Dehghani, Matthias Minderer, Georg Heigold, S. Gelly, Jakob Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020.

Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: dissecting the weight space of neural networks. *arXiv:2002.05688 [cs]*, February 2020. URL `http://arxiv.org/abs/2002.05688`. arXiv: 2002.05688.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016a.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016b. `http://www.deeplearningbook.org`.

Jean-Bastien Grill, Florian Strub, Florent Altché, C. Tallec, Pierre H. Richemond, Elena Buchatskaya, C. Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *ArXiv*, abs/2006.07733, 2020.

Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Z. Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10948–10957, 2020.

Yuting Jia, Haiwen Wang, Shuo Shao, Huan Long, Yunsong Zhou, and Xinbing Wang. On Geometric Structure of Activation Spaces in Neural Networks. *arXiv:1904.01399 [cs, stat]*, April 2019. URL `http://arxiv.org/abs/1904.01399`. arXiv: 1904.01399.

Yiding Jiang, Dilip Krishnan, H. Mobahi, and S. Bengio. Predicting the generalization gap in deep networks with margin distributions. *ArXiv*, abs/1810.00113, 2019.

Jeremiah Johnson. Subspace Match Probably Does Not Accurately Assess the Similarity of Learned Representations. *arXiv:1901.00884 [cs, math, stat]*, January 2019. URL `http://arxiv.org/abs/1901.00884`. arXiv: 1901.00884.

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of Neural Network Representations Revisited. *arXiv:1905.00414 [cs, q-bio, stat]*, May 2019. URL `http://arxiv.org/abs/1905.00414`. arXiv: 1905.00414.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Aarre Laakso and Garrison Cottrell. Content and cluster analysis: Assessing representational similarity in neural systems. *Philosophical Psychology*, 13(1):47–76, March 2000. ISSN 0951-5089, 1465-394X. doi: 10.1080/09515080050002726. URL `http://www.tandfonline.com/doi/abs/10.1080/09515080050002726`.

Yann LeCun and Corinna Cortes. The MNIST database of handwritten digits. URL `http://yann.lecun.com/exdb/mnist/`.

Yann LeCun and Ishan Misra. Self-supervised learning: The dark matter of intelligence, April 2021. URL `https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/`.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *ECML/PKDD (1)*, volume 9284 of *Lecture Notes in Computer Science*, pages 498–515. Springer, 2015.

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent Learning: Do different neural networks learn the same representations? *arXiv:1511.07543 [cs]*, November 2015. URL `http://arxiv.org/abs/1511.07543`. arXiv: 1511.07543.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, December 2018. URL `http://arxiv.org/abs/1802.03426`. arXiv: 1802.03426.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *arXiv:1806.05759 [cs, stat]*, June 2018. URL `http://arxiv.org/abs/1806.05759`. arXiv: 1806.05759.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. *arXiv:1706.05806 [cs, stat]*, June 2017. URL `http://arxiv.org/abs/1706.05806`. arXiv: 1706.05806.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *International Conference on Machine Learning*, pages 8326–8335. PMLR, 2020.

Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60, December 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0197-0. URL `https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0`.

Andrey N. Tikhonov and Vasiliy Y. Arsenin. *Solutions of ill-posed problems*. V. H. Winston & Sons, 1977. Translated from the Russian, Preface by translation editor Fritz John, Scripta Series in Mathematics.

Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting Neural Network Accuracy from Weights. *arXiv:2002.11448 [cs, stat]*, February 2020. URL `http://arxiv.org/abs/2002.11448`. arXiv: 2002.11448.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010. ISSN 1532-4435.

Liwei Wang, Lunjia Hu, Jiayuan Gu, Yue Wu, Zhiqiang Hu, Kun He, and John Hopcroft. Towards Understanding Learning Representations: To What Extent Do Different Neural Networks Learn the Same Representation. *arXiv:1810.11750 [cs, stat]*, October 2018. URL `http://arxiv.org/abs/1810.11750`. arXiv: 1810.11750.

Sewall Wright. Correlation and causation. *J. agric. Res.*, 20:557–580, 1921.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

Scott Yak, J. Gonzalvo, and Hanna Mazzawi. Towards task and architecture-independent generalization gap predictors. *ArXiv*, abs/1906.01550, 2019.

Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8689, pages 818–833. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10589-5 978-3-319-10590-1. doi: 10.1007/978-3-319-10590-1_53. URL `http://link.springer.com/10.1007/978-3-319-10590-1_53`.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Results of Architecture Ablation, Downstream Tasks, Table 4.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We propose a method to investigate the weights of neural network. Potential societal impacts, both positive and negative, are generic machine learning societal impacts and not specific to the proposed methods.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Permutation Equivalence: see Section 2 and Appendix
   (b) Did you include complete proofs of all theoretical results? [Yes] Permutation Equivalence: proof in the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Supplementary Material contains a deatailed walk through the pipeline.
   (b) Did you specify all the training details (e.g., data splits, hyper-parameters, how they were chosen)? [Yes] See Section 4.2 Training and Testing Setup and Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We found the representation learning experiments very stable w.r.t. seeds, but did not have the computational capacity to repeat all of them for multiple seeds. The baselines and downstream task evaluations are deterministic and therefore don't require repetition.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.2: Baslines, Computating Infrastructure and Run Time.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See section 4.1
   (b) Did you mention the license of the assets? [Yes] See section 4.1
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Datasets will be made available upon publication.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Datasets by [Unterthiner et al., 2020] are published under CC BY 4.0 and thus only require accreditation. Our datasets are generated by ourselves and require no consent.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] They do not.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Neither crowdsourcing nor human subjects were used for our research.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Neither crowdsourcing nor human subjects were used for our research.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Neither crowdsourcing nor human subjects were used for our research.