Scratching the Surface: Reflections of Training Data Properties in Early CNN Filters

Grayson Jorgenson
Pacific Northwest National Laboratory
grayson.jorgenson@pnnl.gov

Cassie Heine
Pacific Northwest National Laboratory
cassie.heine@pnnl.gov

Robin Cosbey
Pacific Northwest National Laboratory
robin.cosbey@pnnl.gov

Abby Reynolds
Pacific Northwest National Laboratory
abby.reynolds@pnnl.gov

Davis Brown
Pacific Northwest National Laboratory
davis.brown@pnnl.gov

Henry Kvinge
Pacific Northwest National Laboratory
henry.kvinge@pnnl.gov

Tim Doster
Pacific Northwest National Laboratory
timothy.doster@pnnl.gov

Tegan Emerson
Pacific Northwest National Laboratory
tegan.emerson@pnnl.gov

Abstract

The ability to understand deep learning models by analyzing their weights is key to advancing the growing field of model interpretability. In this article, we study information about the training data of convolutional neural network (CNN) models that can be gleaned from analyzing just the first of their learned filters. While gradient updates to the model weights during training become increasingly complex in the deeper layers of typical CNNs, the updates to the initial layer can be simple enough that high-level dataset properties such as image sharpness, noisiness, and color distribution are prominently featured. We give a simple mathematical justification for this and demonstrate how training dataset properties appear in this way for several standard CNNs on a number of datasets.

1 Introduction

The success of deep learning has resulted in the generation of enormous amounts of neural network weights for models trained to perform almost every conceivable task. It is becoming increasingly important to be able to understand weights as a data type in their own right (1). Enhancing our ability to analyze and understand the model weights is key to advancing interpretability research (21) (22) (23) and addressing the increasing safety and privacy considerations of frontier AI models trained on more data than can be manually vetted by humans (17).

We are interested in a scenario that we refer to as having "static" access to a model: we assume we only have read access to the model weights and do not necessarily have the ability to do forward passes or backpropagate gradients through the model. The latter abilities, which we refer to as having "dynamic" access to a model, enable techniques such as model inversion (5), but require having model implementation code and a sufficient compute environment. In this work we focus purely on the static access scenario, motivated by the goal of developing weight analysis techniques that can help (1) reduce the compute cost of model interpretability studies, and (2) limit the risk

1st Conference on Topology, Algebra, and Geometry in Data Science (TAG-DS 2025).

of inadvertently generating unsafe or illegal content when studying models that may have been trained on harmful datasets (18). This work is a first step towards developing these capabilities by understanding the mathematical fingerprints left by properties of the training data on the model weights.

The specific question we tackle in this article is that of what properties of the training data of convolutional neural network (CNN) models for image processing can be readily deduced from the weights. For this initial exploration, we consider benign, global properties of the dataset such as image quality measures including sharpness, noisiness, and the image color distribution.

Our main result is an observation that properties of the training images for CNNs are prominently featured in the first filters of the models due to the simplicity of the gradient updates for those weights when a standard learning algorithm is used. This applies to both training CNNs from scratch and from pretrained weights when no weights are frozen. We provide examples of this observation in practice using several popular CNNs such as members of the ResNet (6), MobileNet (7), and ConvNeXt (8) architecture families, and on a number of image datasets with varying properties.

The article is organized as follows: Section 2 illustrates our main observation by examining a simplified scenario. Section 3 introduces the image quality metrics and metrics on convolutional filters we employ in our analysis, and Section 4 provides details for the datasets and models we consider for our empirical results. Finally 5 demonstrates image quality properties we can observe in practice.

2 Mathematical Intuition

The key observation of this paper is summarized by the following argument. Let A be an $n \times n$ convolutional filter,

$$A = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n} \end{bmatrix}$$

Consider the operation of this filter on an image B of the same $n \times n$ height and width,

$$B = \begin{bmatrix} b_{1,1} & \dots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \dots & b_{n,n} \end{bmatrix}$$

$$A \cdot B = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} b_{i,j}.$$

Suppose we have a differentiable function $g : \mathbb{R} \to \mathbb{R}$, and let

$$h(a_{1,1}, \dots, a_{n,n}, b_{1,1}, \dots, b_{n,n}) = A \cdot B$$

Then by the chain rule we have

$$\frac{d}{da_{i,j}}(g \circ h)(a_{1,1}, \dots, a_{n,n}, b_{1,1}, \dots, b_{n,n}) = b_{i,j} g'(A \cdot B)$$

If g is thought to represent a loss function such as when training a neural network, then the gradient update to A coming from this loss would be of the form

$$-\alpha g'(A \cdot B) \begin{bmatrix} b_{1,1} & \dots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \dots & b_{n,n} \end{bmatrix}$$

where here α represents the learning rate.

That is, the gradient update to A is just a scalar multiple of B. In this way, spatial relationships between the "pixels" of B are directly transferred to the weights of A.

This has several implications:

- At one extreme, if B were a scalar matrix (all values the same), then the gradient update of the weights of A would be identical.
- If one applied a Gaussian blur kernel to the image, the gradient update to A would be similarly blurred.
- Additive noise added to the original image also directly transfers.
- In the scenario where a CNN were trained on grayscale images, despite having a 3-channel input, the gradient updates to the filters would be desaturated.

See Figure 1 for a simple visualization with a toy CNN composed of a single convolutional layer with 64 $3\times7\times7$ filters (modeled after the first layer of a ResNet) showing some of these observations in action. In each row, the "model" was trained on a dataset of random images of the described form, and g was taken to be the normed difference between the output of the layer and the ones tensor of the same shape. Subtracting the starting, randomly initialized convolutional filter weights from the final "learned" weights yields the cumulative gradient update to each filter, which visually reflects properties such as smoothness of the training images.

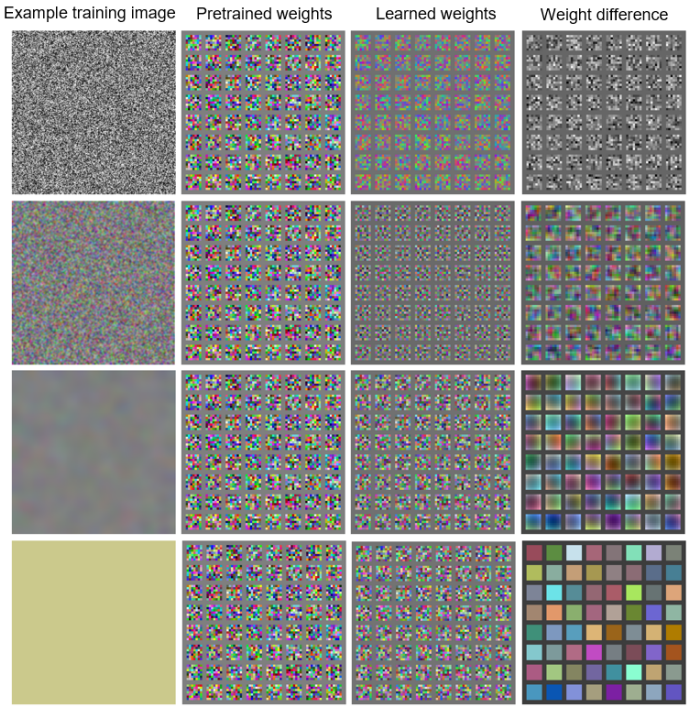

Figure 1: Row 1 is a grayscale Gaussian noise dataset, Rows 2 and 3 are RGB noise with Gaussian blur, and Row 4 was a dataset of random solid color images.

This illustration represents an oversimplified scenario to help reduce the complexity of the argument; the situation becomes more intricate when B is allowed to be much larger than the filter (as is the usual case), the gradient update is accumulated over many training samples, and a multivariate loss function is used. Despite these extra theoretical complexities, we show that in practice the same basic observations still appear to hold for standard CNN architectures and datasets in Section 5.

3 Predicting Training Image Properties from the Weights

We seek to empirically measure the effects of the observation from Section 2. To do this, we first need to train a collection of models on a diverse array of datasets encapsulating wide ranges of the properties we study. The datasets we use and their variations are defined in Section 4. We quantitatively measure properties of the images making up those datasets such as sharpness and noisiness, and then try to predict the values of those metrics from the weights of the models trained on those datasets.

In this Section, we define both the metrics on images that we seek to predict, see Section 3.1, as well as a set of eight metrics defined for the first layer filters of a CNN, see Section 3.2.

We use the filter metrics to encode and reduce down each model we consider into just an 8-dimension vector. These vectors are then fed into a "meta model" (a model trained on other models) such as a random forest regressor to predict the image properties. Our experimental setup and results for this are explained in detail in Section 5.

3.1 Image Properties and Metrics

We try to predict from model weights three properties of the training images: sharpness, noisiness, and relative object scale. The first two are standard image quality properties for which a multitude of measures exist.

Sharpness A number of metrics exist for no-reference sharpness evaluation (12). We choose to measure the sharpness of a color image via the Laplacian operator. We convolve an image with the kernel

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

and then take the variance of the result as a rough estimate of the image sharpness. In Section 4 we create datasets of varying sharpness by applying different levels of Gaussian and resize blur.

Noisiness We estimate image noisiness with a wavelet-based Gaussian variance estimator implemented in the scikit-image Python package (14) (13). In Section 4 we create datasets of varying noisiness by adding different intensities of Gaussian noise.

Relative object scale The right notion of object scale is subtle and difficult to define a computable measure for given an image dataset without spatial annotations such as bounding boxes. In Section 4 we generate synthetic datasets where we have full control over object scale to address this issue. We then attempt to predict the dataset scale factor.

3.2 CNN Filter Metrics

Given that the choice of initial filter kernel sizes can vary between CNN architectures, we seek to featurize the filters via a selection of metrics that encode various properties such as the smoothness, color, structures, and noisiness of the filters. This provides a universal encoding of a fixed dimension that we can then use to train a model to predict training image properties from filter weights. In this subsection, we define 8 metrics on the first layer convolutional filters of a CNN. Here we assume that this first layer has filters of shape $3 \times n \times n$, designed to process RGB input imagery.

Let F be an ordered sequence of N $3 \times n \times n$ filters, $F = (f_1, \ldots, f_N)$. We use the notation $f_i[j]$ to mean the jth color channel of the filter f_i, when thought of as an RGB image. We define the following metrics on F.

Channel saturation Let f_i' denote the min/max normalization of f_i. Let $S = \sum_{i=1}^{N} |f_i|$. Then

$$\text{channel\_saturation}(F) = \sum_{i=1}^{N} \frac{|f_i'||f_i'[0] - f_i'[1]||f_i'[1] - f_i'[2]|}{2Sn^2}$$

Convex hull volume We compute the volume of the convex hull in $\mathbb{R}^3$ spanned by the 3×1 vectors that make up the N filters $f_1, \ldots, f_n$. The filters are normalized with respect to the minimum and maximum of the filter bank F.

Distance to Gaussian blur We convolve each filter with the Gaussian blur kernel

$$C = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

and take the ℓ_2 norm difference with the original filter. That is

$$\text{dist\_to\_gaussian}(F) = \frac{1}{S}\sum_{i=1}^{N} |f_i| \left| \frac{f_i}{|f_i|} - \frac{C \star f_i}{|C \star f_i|} \right|,$$

where $\star$ denotes matrix convolution.

Filter variance This is simply the weighted average of the variance of each filter by the filter norm. That is,

$$\text{filter\_variance}(F) = \frac{1}{S}\sum_{i=1}^{N} |f_i| \text{Var}(f_i)$$

Spatial roughness This metric captures the distances between each spatial vector in the filter and its 8 spatial neighbors (when not on the edge of the filter). Let $f_i^{(a,b)}$ denote the 3×1 vector at the spatial position a, b of filter f_i. Let $g_i = \frac{f_i}{|f_i|}$. Define

$$\text{filter\_roughness}(f_i) = \frac{|f_i|}{8}\sum_{j=2}^{n-1}\sum_{k=2}^{n-1}\sum_{a=-1}^{1}\sum_{b=-1}^{1} |g_i^{(j+a,k+b)} - g_i^{(j,k)}|.$$

Then

$$\text{filter\_roughness}(F) = \frac{1}{S}\sum_{i=1}^{N} \text{filter\_roughness}(f_i).$$

Bregman distance This is defined to be

$$\frac{1}{S}\sum_{i=1}^{N} |f_i| \left| \frac{f_i}{|f_i|} - \frac{B(f_i)}{|B(f_i)|} \right|,$$

where B represents total variation denoising using split-Bregman optimization (15), as implemented in scikit-image (14)

Variance of the Laplacian Treats the filter as a small image, using the definition in the preceding subsection used to estimate image sharpness. The final metric is the sum of these weighted by the filter norms.

Wavelet approximation of the Gaussian noise variance Uses the estimator of (13) from the previous subsection that approximates the noisiness of the filters as images. Final metric is the sum of these weighted by the filter norms.

4 Datasets and Models

Our study requires a large collection of models and associated image property metrics we can then attempt to predict from the weights. We trained several thousand models across a diverse set of datasets where we had fine-grained control over image sharpness, noisiness, and object scale. Section 4.1 describes the datasets considered while Section 4.2 provides the different model architectures we worked with.

4.1 Datasets

We use three image datasets for our empirical analysis: ImageNette (2), WHOI-Plankton (4), and a collection of variants on a 3D shape classification dataset we generated with Blender (3) that we refer to as BlenderShapes. Brief descriptions of each base dataset are as follows:

ImageNette is a subset of the ImageNet dataset using only 10 of the classes, and consists of 10k training images and 1k testing images. These are natural RGB images, primarily taken with handheld cameras.

WHOI-Plankton is a dataset of millions of grayscale images of microscopic plankton. We downsample this dataset to 20 classes and the same number of training and testing images as ImageNette.

BlenderShapes is a dataset of synthetic 400×400 images each featuring a single 3D shape from one of 6 classes: sphere, cube, triangular prism, cylinder, cone, and torus. The textures, lighting, object orientation, and background plane are all randomized. There are 6k training images, and 600 testing images, balanced evenly across the classes. Our goal in creating this dataset was to study the question of whether another dataset property, object scale, can be deduced from CNN model weights. We made five versions of this dataset, each featuring its objects at a different scale relative to the camera. Figure 2 shows some examples. By creating the BlenderShapes dataset, we were able to ensure the sharpness metrics for the different sizes remain similar, while object scale changes as desired. This was to avoid the issue of conflating sharpness with object scale when upsampling/downsampling ImageNette or WHOI-Plankton (as, for example, upsampling with bilinear interpolation reduces image sharpness).

For each dataset, we trained models on a number of image quality variations of that dataset. Specifically, we varied the sharpness by applying a Gaussian blur and resizing, and the noisiness by adding Gaussian noise. All images in the original datasets were normalized to 224×224 resolution. For each core dataset (ImageNette, WHOI-Plankton, BlenderShapes), we made a total of 20 variations. The specifics are given below:

- The original dataset, unperturbed
- 5 size dilations of the datasets, at $1.25, 1.5, 2.0, 2.5, 3.0$ times the native resolution
- 4 versions of the dataset where the images were first scaled to $0.1, 0.25, 0.5, 0.75$ resolution, then scaled back up to 224×224 resolution
- 5 Gaussian blurred versions of the datasets, with kernels $3, 7, 11, 15, 19$ and $\sigma = 1, 2, 3, 4, 5$, respectively
- 5 versions of the dataset with Gaussian noise added, with $\sigma = 0.1, 0.25, 0.5, 0.75, 1.0$

Figure 2: **Left:** Random images selected from the BlenderShapes datasets. Each selection comes from one of the different scale variations of the dataset. **Right:** Examples of the sharpness and noisiness variations of the ImageNette dataset used in our experiments.

4.2 Models

We used several standard CNN architectures: ResNet-18 and ResNet-50 (6), MobileNetv3-Large (7), and ConvNeXt-Tiny (8). We use the PyTorch (19) and timm (20) libraries for the model implementations, and source their pretrained weights from the associated repositories. We trained all models on an NVIDIA Tesla P100 GPU. With regards to CNN filters, both ResNet architectures open with a convolutional layer consisting of 64 $3 \times 7 \times 7$ filters. MobileNetv3-Large opens with 16 $3 \times 3 \times 3$ filters, and ConvNeXt-Tiny with 96 $3 \times 4 \times 4$ filters.

Table 1: Property prediction results

Experiment	Training samples	Testing samples	Test R2 score
R50 to R18, sharpness	3402	3280	0.922 ± 0.004
R50 to R18, noisiness	3402	3280	0.614 ± 0.01
R50 to R18, object scale	900	900	0.373 ± 0.004
Three to CNeXt, sharpness	10084	662	0.749 ± 0.027
Three to CNeXt, noisiness	10084	662	-0.082 ± 0.115

5 Experimental results

5.1 Experiment Setup

For each image property, we trained a random forest "meta model" on the vectors of filter metrics for the first layers of a number of trained CNNs belonging to the architecture varieties discussed in Section 4. We argue that success of the meta models in predicting image quality properties from the featurized filter weights is evidence for the prominence of this information in the first layer filters of these CNNs.

We use ImageNet-1k pretrained weights for each architecture and did not freeze any weights. Each experiment used either the Adam optimizer with a learning rate of $1e$-4 or SGD with a Cosine Annealing learning rate scheduler, starting with an initial learning rate of $1e$-1. Models were trained for 30, 20, or 25 epochs for ImageNette, WHOI-Plankton, and BlenderShapes, respectively. Checkpoints were saved every 5 epochs and we performed three trials of each training run. Dataset sharpness and noisiness were calculated as the averages of $2,000$ random samples from each dataset.

The essential signal we are trying to isolate is the gradient update that has been applied to the filters, rather than the values of the learned filters themselves. We assume in this article that we have access to the pretrained weights for a given architecture, which we take as a reasonable access assumption given the prevalence of the practice of pulling such weights from open source repositories.

To compute the filter metrics, we take the difference between the learned filters and the pretrained filters before applying the metrics. Figure 3 shows a visualization of what this looks like for a ResNet-50. Note how the observations of Section 2 appear to hold despite using a standard CNN architecture, dataset, and optimization method. Our random forest models each consisted of 30 decision tree regressors that could leverage all 8 filter metric features. We used the scikit-learn library (16) for all random forest training and evaluation.

5.2 Results

We find that the average image sharpness and image noisiness, as measured by the metrics of Section 3, are detectable from the vectors of filter metrics of the first layer weights of CNNs. Our results are summarized in Table 1.

There "R50 to R18" means that the random forest was trained on a set of ResNet-50s, and tested on a set of ResNet-18s. "Three to CNeXt" means that the random forest was trained on ResNet-18s, ResNet-50s, and MobileNetv3-Large and tested on a set of ConvNeXt-Tiny. We performed five training runs for each random forest, from which we derived the mean and error bars of our test results table. See Figure 4 for corresponding representative performance plots.

We found success in predicting sharpness from the first layer filters even when subject to a large distribution shift such as training the meta model on the ResNet and MobileNet architectures and testing on ConvNeXt. That said, predicting the noisiness of the training data appears more challenging, potentially indicating that our choice of filter metrics could be improved. We had partial success predicting object scale from the first filter weights; there is a correlation, but it appears to be a weaker signal than sharpness or noisiness under the same conditions.

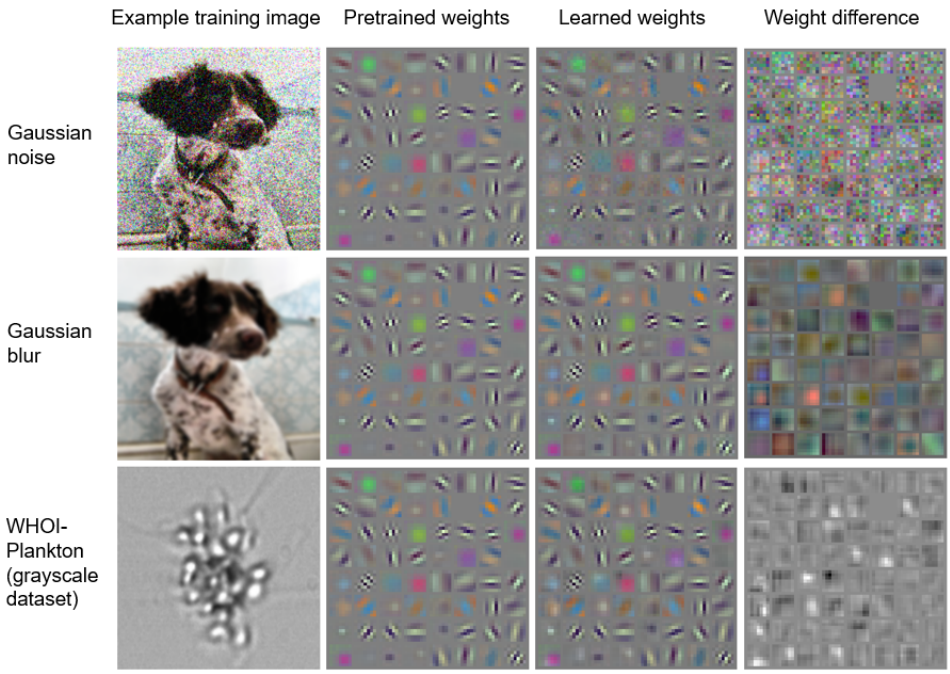

Figure 3: Shows the 64 filters of an ImageNet-1k pretrained ResNet-50 before and after training, and the resulting weight difference.

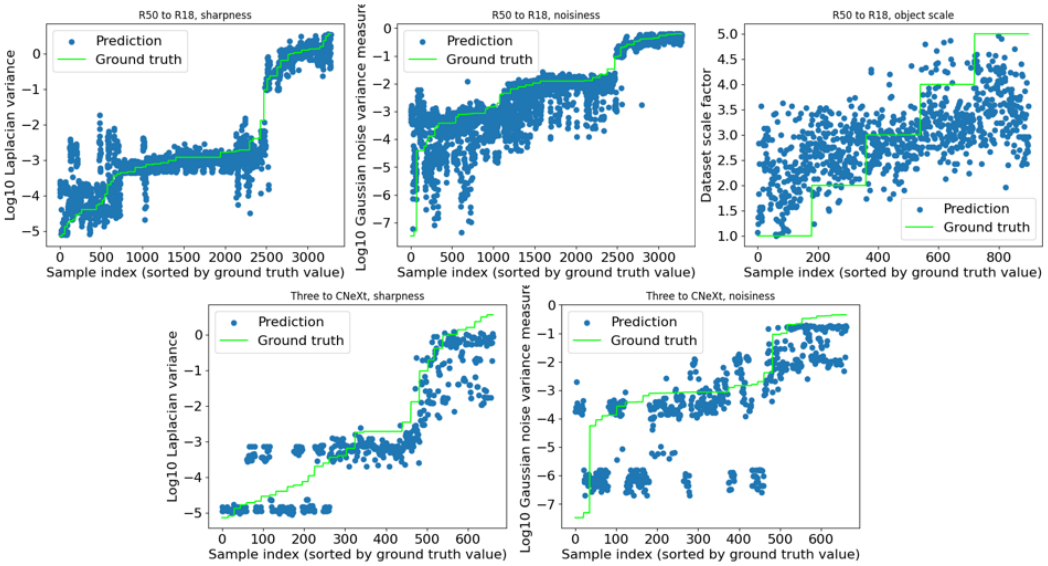

Figure 4: Random forest meta model test set performance for the different first layer filter datasets. The ground truth values for sharpness, noisiness, and object scale across all relevant test datasets are plotted in ascending order (green lines). The meta model predictions given the model weights for those datasets are plotted in blue.

6 Conclusion

In this work, we showed that the gradient updates for the first layer of a CNN are directly tied to the input image pixels. By featurizing the difference between the learned and initial filters of these layers using a number of metrics, we were able to successfully train random forest meta models capable of estimating image quality properties of the CNN training data. We claim that this supports the hypothesis that high-level information about the training dataset such as these image quality properties is prominent in the first layer weights. We hypothesize that, with improvements to the metrics used, this ability to predict training data properties could be further enhanced.

For the initial explorations, we made the assumption that the pretrained weights are known allowing us to compute filter differences. Alternatively, one could also take the difference between consecutive saved checkpoints of a model as a way of getting at these gradient update differences. Finally, while CNNs were the focus of this work, vision transformers are becoming increasingly prevalent in the vision space, sometimes attaining equal or better performance than their CNN counterparts (10). We also hypothesize that the methods of this article could apply to vision transformers with initial convolutions such as ViT (11) and SWIN (9).

Acknowledgments and Disclosure of Funding

This work was conducted under the Laboratory Directed Research and Development Program at PNNL, a multi-program national laboratory operated by Battelle for the U.S. Department of Energy under contract DE-AC05-76RL01830.

References

[1] *Workshop on Weight Space Learning* ICLR 2025.

[2] *Imagenette*, `https://github.com/fastai/imagenette`

[3] *Blender* `https://www.blender.org/`

[4] Orenstein, Eric C., et al. *Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification*. arXiv preprint arXiv:1510.00745 (2015).

[5] Yin, Hongxu, et al. *Dreaming to distill: Data-free knowledge transfer via deepinversion*. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[6] He, Kaiming, et al. *Deep residual learning for image recognition.* Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[7] Howard, Andrew, et al. *Searching for mobilenetv3.* Proceedings of the IEEE/CVF international conference on computer vision. 2019.

[8] Liu, Zhuang, et al. *A convnet for the 2020s.* Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.

[9] Liu, Ze, et al. *Swin transformer: Hierarchical vision transformer using shifted windows.* Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[10] Tian, Yunjie, Qixiang Ye, and David Doermann. *Yolov12: Attention-centric real-time object detectors.* arXiv preprint arXiv:2502.12524 (2025).

[11] Dosovitskiy, Alexey, et al. *An image is worth 16x16 words: Transformers for image recognition at scale.* arXiv preprint arXiv:2010.11929 (2020).

[12] Maheshwary, Priti, Mukul Shirvaikar, and Christos Grecos. *Blind image sharpness metric based on edge and texture features.* Real-Time Image and Video Processing 2018. Vol. 10670. SPIE, 2018.

[13] D. L. Donoho and I. M. Johnstone. *Ideal spatial adaptation by wavelet shrinkage.* Biometrika 81.3 (1994): 425-455. :DOI:10.1093/biomet/81.3.425

[14] Van der Walt, S., Schonberger, Johannes L, Nunez-Iglesias, J., Boulogne, Franccois, Warner, J. D., Yager, N., ... Yu, T. (2014). *scikit-image: image processing in Python.* PeerJ, 2, e453.

[15] Goldstein, Tom, and Stanley Osher. *The split Bregman method for L1-regularized problems.* SIAM journal on imaging sciences 2.2 (2009): 323-343.

[16] Pedregosa, Fabian, et al. *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research 12 (2011): 2825-2830.

[17] Birhane, Abeba, et al. *Into the laion's den: Investigating hate in multimodal datasets.* Advances in Neural Information Processing Systems 36 (2023): 21268-21284.

[18] Schramowski, Patrick, et al. *Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

[19] Paszke, Adam, et al. *Pytorch: An imperative style, high-performance deep learning library.* Advances in neural information processing systems 32 (2019).

[20] Wightman, Ross. *PyTorch image models.* `https://github.com/rwightman/pytorch-image-models`

[21] Hong, Yihuai, et al. Intrinsic evaluation of unlearning using parametric knowledge traces. *arXiv preprint arXiv:2406.11614 (2024).*

[22] Alsallakh, Bilal, et al. On Symmetries in Convolutional Weights. *arXiv preprint arXiv:2503.19215 (2025).*

[23] Horwitz, Eliahu, et al. Learning on model weights using tree experts. *Proceedings of the Computer Vision and Pattern Recognition Conference. 2025.*