Estimating Environmental Cost Throughout Model's Adaptive Life Cycle

Vishwesh Sangarya[1], **Richard Bradford** [2], **Jung-Eun Kim** [1*]

[1] Computer Science, North Carolina State University
[2] Collins Aerospace

Abstract

With the rapid increase in the research, development, and application of neural networks in the current era, there is a proportional increase in the energy needed to train and use models. Crucially, this is accompanied by the increase in carbon emissions into the environment. A sustainable and socially beneficial approach to reducing the carbon footprint and rising energy demands associated with the modern age of AI/deep learning is the adaptive and continuous reuse of models with regard to changes in the environment of model deployment or variations/changes in the input data. In this paper, we propose `PreIndex`, a predictive index to estimate the environmental and compute resources associated with model retraining to distributional shifts in data. `PreIndex` can be used to estimate environmental costs such as carbon emissions and energy usage when retraining from current data distribution to new data distribution. It also correlates with and can be used to estimate other resource indicators associated with deep learning, such as epochs, gradient norm, and magnitude of model parameter change. `PreIndex` requires only one forward pass of the data, following which it provides a single concise value to estimate resources associated with retraining to the new distribution shifted data. We show that `PreIndex` can be reliably used across various datasets, model architectures, different types, and intensities of distribution shifts. Thus, `PreIndex` enables users to make informed decisions for retraining to different distribution shifts and determine the most cost-effective and sustainable option, allowing for the reuse of a model with a much smaller footprint in the environment. The code for this work is available here:
https://github.com/JEKimLab/AIES2024PreIndex

Introduction

Considering the entire development life-cycle of a neural network model, the impact of the training procedure on the environment is substantial, especially with respect to carbon emissions and energy consumption. It would be preferable for a model to be used frequently for a long time "as is". However, this is not always feasible. Models must sometimes adapt to a new distribution, environment, or situation - for example, some ground truths might be changed, some data samples might become stale, or some new data samples might need to come into play. A sustainable solution to such situations, ultimately with regard to energy, carbon emissions, and resource consumption, is *reusing* an existing model to adapt to such changes. That is, models can be retrained and adapted to new distributions with minimal retraining instead of training from *scratch*. Retraining can achieve satisfactory accuracy on the new distribution while exhibiting lower computation costs, thus reducing the carbon footprint and energy consumption during the model's development and deployment. The primary objective of this paper is the introduction of a novel metric designed to predict and estimate the environmental resource costs associated with reusing a model for distributional shifts. We provide empirical evidence to show that model retraining significantly lowers compute costs compared to training a new model from scratch.

As deep learning models become more prevalent in everyday applications, the associated compute demand increases significantly, leading to substantial electricity consumption for model training and inference. This trend has significant social implications, primarily involving the increased release of carbon compounds into the atmosphere. Various works (Verdecchia, Sallou, and Cruz 2023; Schwartz et al. 2019) have highlighted the importance of considering sustainable and socially conscious practices in AI research and application. (Fu, Hosseini, and Plataniotis 2021; Bannour et al. 2021) detail the growing carbon footprint of recent deep learning models for vision and language tasks. Recent research works (Yarally et al. 2023; Moro, Ragazzi, and Valgimigli 2023) have shifted towards a sustainability-focused approach to model development by being sustainability-oriented instead of performance-oriented. These research works call attention to the ongoing need for deep learning progress to balance energy demands and carbon emissions with societal concerns. Moreover, there is a requirement for reducing carbon emissions, which in turn contributes to lower climate change, thus protecting areas and populations at risk from the impacts of climate change and scarcity of energy.

Hence, we propose a predictive index (`PreIndex`) to estimate the environmental cost of retraining a model to new changes in the data. `PreIndex` can be used to estimate the resources that would be expended if a model is retrained to a

*Correspondence.

new distribution. PreIndex quantifies the change and collapse of the class decision boundaries due to the shift in distribution, and also quantifies the shift in the representation as a result of the changing distribution. PreIndex requires only a single forward pass of the model, following which it provides a single concise quantitative estimate to compare and predict a model's retraining cost. A lower value of PreIndex indicates that fewer resources would be expended when retrained, and vice versa.

We conduct experiments to validate that PreIndex is an effective estimator of environmental costs such as carbon emissions and energy usage, as well as several other retraining indicators such as epochs, gradient norm, and change in parameter magnitudes. Through extensive experiments over convolutional architectures and also Vision Transformers (ViT), we show that PreIndex is model agnostic and can be used with different architectures without requiring any modifications to the model structure. By leveraging PreIndex, deep learning practitioners and organizations can make informed decisions on deploying models that meet sustainability and resource usage goals.

Related Work

Distribution shifts can occur due to several factors and can be of different types and intensities as seen in (Hendrycks and Dietterich 2019; Arjovsky et al. 2020; Hendrycks et al. 2021). Augmentation techniques (Hendrycks et al. 2020; Liu et al. 2022; Zhang et al. 2018; Kim, Choo, and Song 2020; Lee et al. 2020) have shown to provide robustness to certain types of distribution shifts, but are computationally heavy and require training a model from scratch. These methods produce robust models for certain distribution shifts, but have marginal improvements on other distribution shifts. Several studies (Geirhos et al. 2020; Yin et al. 2020; Ford et al. 2019) have shown that there is a non-uniform improvement in robustness to the different distribution shifts; in some cases, improvement on one type of noise or corruption results in decreased performance on other distributional shifts. Methods using test time adaptation (Lim et al. 2023; Niu et al. 2022; Goyal et al. 2022; Wang et al. 2022) exhibit only minimal improvements in model robustness, rely on batch data, and fail to provide substantial benefits in scenarios with elevated noise levels. If the test time information is insufficient for adapting the model's prediction, these methods fail to provide accurate and confident outputs during inference.

(Schwartz et al. 2019) originally coined the concept of Green AI and Red AI, emphasizing how the substantial growth in computational complexity and resource usage of models led to only marginal enhancements in accuracy. They highlight the need for sustainable practices to go hand in hand with performance improvements when developing models. (Verdecchia, Sallou, and Cruz 2023) conducted a systematic survey of recent works in sustainable deep learning and showed that energy consumption and carbon footprint are the two most predominant measures to quantify sustainability. Related works (Xu et al. 2023, 2021; García-Martín et al. 2019; McDonald et al. 2022) tackle the issue of energy consumption of neural networks. (Bannour et al. 2021; Fu, Hosseini, and Plataniotis 2021) demonstrate the significant environmental impact of modern deep learning methods due to the carbon footprint associated with training vision and language models. (Dodge et al. 2022) shows that training ViTs (transformer-based architectures)(Dosovitskiy et al. 2021) emits a considerably greater amount of carbon compared to convolutional networks. Similarly, (Strubell, Ganesh, and McCallum 2019) evaluates the high energy consumption of transformer-based models, thereby leading to higher carbon emissions. (Henderson et al. 2020; Patterson et al. 2021) draw attention to the lack of carbon emission reporting in deep learning research. They provide frameworks for carbon emission measurement and documentation with an emphasis on quantifying the sustainability aspects of training models. (Schmidt et al. 2021; Anthony, Kanding, and Selvan 2020; Budennyy et al. 2023; Lacoste et al. 2019) provide tools and frameworks to estimate carbon emissions based on energy usage while training models and emissions from energy generation. (Sangarya, Bradford, and Kim 2023) explore the impact of individual noise types on model adaptation by using original-noise image pairs.

(Agarwal, D'souza, and Hooker 2022; Lee and AlRegib 2020; Huang, Geng, and Li 2021) show how gradients are an important tool in measuring the difficulty of samples and identifying samples that belong to new distributions. They show how the gradients are steeper and have larger values for difficult data and for out-of-distribution data. Studies such as (Stacke et al. 2020) use the change in layer representation to study pathology data and focus their work to individual layers of a model to show it correlates to accuracy loss on domain shifts.

Various research studies (Raiber and Kurland 2017; Abou-Moustafa and Ferrie 2012) demonstrate that commonly used distance measures are not sufficient to be an effective distance metric. (Tolstikhin et al. 2018; Arjovsky, Chintala, and Bottou 2017; Faber et al. 2021) study the drawbacks of prevalent divergence metrics for specific use cases. (Hubert and Arabie 1985) introduced the Adjusted Rand Index for comparing clustering labels, and (Santos and Embrechts 2009) illustrate the use of Adjusted Rand Index to evaluate supervised classification and feature selection. (Deng et al. 2023) compares the performance benefits of retraining the entire pre-trained model versus retraining only the feature extractor for classification tasks.

PreIndex

In this section, we provide a detailed overview regarding PreIndex, which consists of three components – inverse adjusted rand index, averaged sample representation distance, and noise variance scaling. We introduce each component individually and provide the final formulation of PreIndex at the end of the section.

Table 1 provides all notations used in the following subsections.

Adjusted Rand Index for Distribution Shift We examine the collapse of decision boundaries between classes' data in the representation space as a result of noise. We quantify this change and shift in the decision boundaries by perform-

Symbol	Description	Symbol	Description
p	Average sample distance	X_{clean}	Clean Image
n_s	Number of samples	X_{noisy}	Noisy Image
n_l	Number of layers	$CentM$	Centroids matrix
n_c	Number of clusters	P	PDF function
WD	Wasserstein distance	Var	Variance function
h_{rep}	Height of filter representation	h_{img}	Height of image
w_{rep}	Width of filter representation	w_{img}	Width of image
Y	Class label of a sampple	λ	Constant factor
r	Cluster labels row sum	t	True labels row sum
E	Normalized Euclidean distance	W	Flattened Weight vector
M	Model	R	Representation of a sample
c	Number of classes in the data	f	Number of filters in a layer
$ActO_l$	Activation output function for layer l		
s	Standard deviation of pixel differences		
$\bar{s}$	Average deviation over all intensities of specific noise type		
d	Distance per sample at a given layer		

Table 1: Notation

ing clustering on the data representation of the distribution-shifted data. The shift and collapse of the decision boundaries is quantified by obtaining representation data of the entire distributional shift data, followed by clustering on the representation data to generate cluster labels. The cluster labels and the true labels are evaluated to quantify the change in the decision space. Adjusted Rand Index (Hubert and Arabie 1985) is useful to assess a clustering algorithm. Adjusted Rand Index (`ari`) is defined as,

$$\texttt{ari} = \frac{\sum_{i,j=0}^{n_c} \binom{n_{i,j}}{2} - \left[\sum_{i=0}^{n_c} \binom{r_i}{2} \sum_{j=0}^{n_c} \binom{t_j}{2}\right] / \binom{n_s}{2}}{(1/2)\left[\sum_{i=0}^{n_c} \binom{r_i}{2} + \sum_{j=0}^{n_c} \binom{t_j}{2}\right] - \left[\sum_{i=0}^{n_c} \binom{r_i}{2} \sum_{j=0}^{n_c} \binom{t_j}{2}\right] / \binom{n_s}{2}} \tag{1}$$

where t represents the total count of true labels for each label in the contingency table of true labels vs. representation labels via clustering. r represents the summed values of representation cluster labels in the contingency table. A contingency table in this scenario is a matrix that summarizes the number of samples belonging to the same cluster or having the same label in both clustering scenarios. Here, by 'both clustering scenarios,' we refer to the representation-based clustering labels and the true labels. n_c is the number of cluster labels, which is equal to the number of class labels. $n_{i,j}$ represents the value in each entry of the contingency table, which is common to both cluster labels for a given label i and j. n_s is the total number of samples.

`ari` takes the value 0 for purely random clustering, and 1 for identical clustering. For our estimator, it is required to have a low value for decision boundaries which are well separated and high value for boundaries which overlap and result in incorrect representation cluster labels. Hence, for our estimator, `PreIndex`, we take the complement of `ari` and define it as

Algorithm 1: Representation data's cluster initialization

Input: Number of samples n_s, where X_i is the ith sample
Number of labels c,
Model M with representation output (R, Y),
for each sample where R is the representation data and Y is the label

Output: Centroids matrix $CentM$, where $CentM_p$ is the pth centroid vector

$CentM \leftarrow$ *Empty Vector of size c*
for $i \leftarrow 1$ to k **do**
  $cur\_centroid \leftarrow \overrightarrow{0}$
  $label\_count \leftarrow 0$
  for $j \leftarrow 1$ to n_s **do**
    $(R, Y) \leftarrow M(X_j)$
    if $Y == i$ **then**
      $cur\_centroid \leftarrow cur\_centroid + R$
      $label\_count \leftarrow label\_count + 1$
    end if
  end for
  $CentM_i \leftarrow cur\_centroid / label\_count$
end for

$$inv\_ari = \frac{(1/2)\left[\sum_{i=0}^{n_c} \binom{r_i}{2} + \sum_{j=0}^{n_c} \binom{t_j}{2}\right] - \sum_{i,j=0}^{n_c} \binom{n_{i,j}}{2}}{(1/2)\left[\sum_{i=0}^{n_c} \binom{r_i}{2} + \sum_{j=0}^{n_c} \binom{t_j}{2}\right] - \left[\sum_{i=0}^{n_c} \binom{r_i}{2} \sum_{j=0}^{n_c} \binom{t_j}{2}\right] / \binom{n}{2}} \tag{2}$$

The data representation is obtained from the final convolution layer for a convolutional network' case while from the final dense layer in the last transformer encoder block for a vision transformer's case. To obtain the representation labels by clustering, we use KMeans with 3 different centroid initialization techniques as follows:

1. Using the original data to obtain centroids as detailed in

Algorithm 1,

2. Initializing by Kmeans++ (Arthur, Vassilvitskii et al. 2007), and
3. Initializing by random cluster assignment, and selecting the cluster with least entropy among 20 random initialization seeds.

The above three initialization schemes result in similar clustering labels. We use Algorithm 1 due to its computational efficiency as it does not requiring random re-initialization or iterative assignment of centroids, unlike methods such as repeated random cluster initialization and KMeans++. Algorithm 1 has quadratic runtime with respect to dataset size and number of class labels. However, it is computed only once to initialize the cluster centroids. This is an efficient approach compared to KMeans++, which has been shown to have a super-polynomial run-time starting from initialization to converge in the worst case. In Algorithm 1, we begin by creating an empty vector that has a size equal to the number of class labels, as depicted in line 1. $CentM$ is a vector of size c when initialized, but as the classes' centroids are obtained, each entry in $CentM$ is a vector itself. In the end, $CentM$ is a matrix of size c x Size of flattened representation.

Average Sample Representation Distance In this subsection, we introduce the average sample distance between representations obtained from the original and distribution-shifted sample. The average sample distance is calculated per layer of the model for each data point and then aggregated to provide a single concise scalar value. Wasserstein distance is used to find the distance between probability distributions obtained from the representation of the original sample and the distribution-shifted sample. The two data distributions are used to perform a forward pass and obtain the probability distribution from the activations of each layer.

Algorithm 2 provides the detailed procedure to obtain the layer distance per sample for a given layer l of a model. In Algorithm 2, functions P and WD represent the functions to compute probability density and the Wasserstein distance function, respectively. l_clean and l_noisy are the activation output of all filters in layer l for clean and noisy images, respectively. l_{clean} and l_{noisy} are vectors of size f – number of filters in the layer l. The activation output of each filter is averaged as depicted in lines 5–7 in the algorithm, using h_{rep} and w_{rep}, which are the height and width of each filter representation output, respectively. P_{clean} and P_{noisy} are the probability distributions for the clean and noisy samples, respectively.

As a reference, Wasserstein distance is preferred over KL-Divergence, Bhattacharya distance, Jensen-Shannon divergence, and Hellinger distance. Wassertein distance, unlike KL-divergence, is a true distance metric that exhibits symmetry (Raiber and Kurland 2017), and in contrast to Bhattacharya distance, Wasserstein distance satisfies the triangle inequality (Abou-Moustafa and Ferrie 2012). Studies such as (Faber et al. 2021; Tolstikhin et al. 2018; Abou-Moustafa and Ferrie 2012) highlight why Wasserstein distance is preferred over KL-divergence and its variants, Jensen-Shannon

Algorithm 2: Calculate layer distance per sample

Input: $ActO_l$: Layer l's output function with f filters,
X_{clean}: clean sample ; X_{noisy}: noisy sample,
$ActO_{l-1}$: output function for layer 1 through $l-1$
Output: d_l: layer distance

1: $(Rep_{clean})_{l-1} \leftarrow ActO_{l-1}(X_{clean})$
2: $(Rep_{noisy})_{l-1} \leftarrow ActO_{l-1}(X_{noisy})$
3: $l\_clean \leftarrow ActO_l((Rep_{clean})_{l-1})$
4: $l\_noisy \leftarrow ActO_l((Rep_{noisy})_{l-1})$
5: **for** $k \leftarrow 1$ to f **do**
6: $\quad l\_clean_k \leftarrow \frac{1}{h_{rep} \cdot w_{rep}} \sum_{i=1,j=1}^{h_{rep},w_{rep}} (l\_clean_k)_{i,j}$
7: $\quad l\_noisy_k \leftarrow \frac{1}{h_{rep} \cdot w_{rep}} \sum_{i=1,j=1}^{h_{rep},w_{rep}} (l\_noisy_k)_{i,j}$
8: **end for**
9: $P_{clean}, P_{noisy} \leftarrow P(l\_clean), P(l\_noisy)$
10: $d_L \leftarrow WD(P_{clean}, P_{noisy})$

distance, for scenarios where quantifying the exact difference between the distributions has greater importance than measuring the likelihood between distributions. Additionally, (Öcal, Grima, and Sanguinetti 2019) demonstrates that Wasserstein distance is effective in capturing the horizontal distances between distributions within the metric space, unlike Hellinger distance.

The process of obtaining the representation distance per sample and per layer is repeated for all samples, and for all layers of the model. The distances for all samples and across all layers are then averaged to obtain the final average sample distance p as follows:

$$p = \frac{1}{n_s} \frac{1}{n_l} \sum_{i=1}^{n_s} \sum_{k=1}^{n_l} (d_k)_i \tag{3}$$

where $(d_k)_i$ is the distance between distributions at a given layer k for the ith sample, n_s is the total number of samples used, and n_l is the number of layers in the given model.

Noise Variance Scaling (Li et al. 2020; Vargas and Su 2020) show that neural networks make incorrect predictions even with small levels of noise in an input image. In particular, (Vargas and Su 2020) illustrates the cascading impact of a single pixel change with large magnitude and its effect on neighboring values of the image representation in the deeper layers of a model.

Different noise types have different traits. In salt-pepper and impulse noises, certain individual pixels (either few or many) are associated with the noise. Hence, the noises affect a specific subset of pixels in an image with a larger magnitude of change per pixel value (that is noised). We refer to this type of noise as pixel-specific noise. Conversely, Gaussian, Blur, Frost, and Poisson noises affect (almost) all pixels in an image with a smaller change per pixel value. We refer to this as global image noise.

Pixel-specific noises, are easier to adapt to since they only affect a subset of pixels as compared to the global image noises. However, the impact of a large magnitude change of a subset of pixels can propagate to the surrounding values in the deeper layer representations. As a result

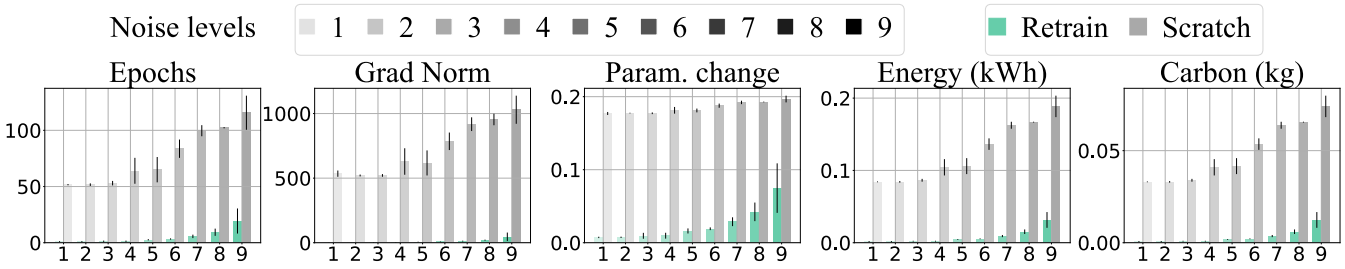

Figure 1: Training from scratch vs. retraining of ResNet18 on CIFAR10 with Poisson noise. Retraining consumes clearly less resources than training from scratch.

the model may overestimate the raw perturbations caused by pixel-specific noises. To mitigate the overestimation of pixel-specific noises, we introduce an inverse scaling factor. The scaling factor helps reduce the value of `PreIndex` by utilizing the standard deviation of raw pixel intensities between a clean and a noisy image. We obtain the standard deviation for a specific noise type and intensity as follows:

$$s = \sqrt{Var\left([X_{clean(i,j)} - X_{noisy(i,j)}]_{(i,j)\in(h_{img},w_{img})}\right)}/\lambda \tag{4}$$

Here, clean image and noisy image are denoted as X_{clean} and X_{noisy}, respectively. h_{img} and w_{img} are the height and width of the image, respectively. The resultant standard deviation is then scaled down by a fixed constant factor λ. Var represents the variance of a given vector.

Finally, `PreIndex` for quantifying distribution shifts is formulated as,

$$PreIndex = (p + inv\_ari)\underbrace{(1/(1 + (p + (1 - ari)) * s))}_{\text{scaling factor}} - \bar{s} \tag{5}$$

For pixel-specific noises, `PreIndex` is scaled down using the scaling factor and average deviation, $\bar{s}$. Average deviation $\bar{s}$ is the average of standard deviations across all intensities for the specific noise type. It is used as an offset when scaling down `PreIndex` for pixel-specific noise. For global image noises, `PreIndex` is utilized without the scaling factor or average deviation. In `PreIndex`, p, ari, and s are obtained for each noise type with a specific intensity. Hence, the noise intensity index is omitted for the sake of simplicity. The values for average sample distance p and ari for each noise type and intensity in Eq. (5) are from Eq. (1) and Eq. (3), respectively.

Resource Indicators

Resource indicators represent the resources/cost that would have been required if a model were trained or retrained to a new target task or to a new distribution. We show that `PreIndex` has a strong correlation with, and is an effective estimator of, the various indicators listed in this section. Using `PreIndex` and based on an indicator of interest, a user can gain knowledge regarding the resource expenditure they are likely to expend if they retrain the model to a new task or distribution. We evaluate several indicators - *epochs, gradient norm, change in parameter magnitude, energy, and carbon emissions*. In particular, energy and carbon emissions represent immediate sustainability costs that are likely to embody an ultimate goal to potential users.

Epochs

When a model is retrained and adapted to a new target task or distribution, one might count the training cost/effort by looking at how many epochs are expended. For empirical evidence, we show how consistently `PreIndex` aligns with the number of epochs, which validates that `PreIndex` is an effective retraining predictive quantifier. The epochs that are reported in this paper are obtained when a model reaches a certain cutoff test accuracy for each dataset. Utilizing additional cutoff conditions, training is terminated after either 25 or 50 epochs if the accuracy gap from the designated cutoff accuracy is within 0.5% or 1%, respectively.

Gradient Norm

Gradient norm represents the difficulty of learning a new distribution and provides information regarding the likely steepness to reach convergence. For instance, several studies (Agarwal, D'souza, and Hooker 2022; Huang, Geng, and Li 2021; Lee and AlRegib 2020) use gradient norm as a proxy for sample difficulty. They compute gradients using a uniform distribution and do not make use of ground truth label information. For our objective, we report the gradient norm by utilizing the ground truth label. During retraining, we obtain the overall gradient norm for each instance by aggregating the gradient norm at each time step. The final gradient norm value represents the total magnitude of gradients the model encountered throughout the retraining process.

Parameter Change

Change in parameter value for the entire retraining process represents the magnitude of updates a parameter undergoes. That is, during the retraining process, if the model goes through consistently high parameter changes, it indicates that the model requires further updates and to learn the new distribution.

Hence, it is useful to use `PreIndex` to estimate a model's parameter change when a model is reused for a new distribution.

We obtain the change in parameter magnitudes similar to what was done in (Zhang, Bengio, and Singer 2022). However, unlike (Zhang, Bengio, and Singer 2022) that calculate

Model	Correlation	Epochs	GradNorm	Param. change
ResNet18	Pearson	0.72 (4.2e-10)	0.66 (4.2e-08)	0.63 (4.7e-07)
	Spearman	0.93 (3.1e-25)	0.93 (1.1e-24)	0.93 (1.3e-25)
GoogleNet	Pearson	0.77 (8.0e-12)	0.73 (1.7e-10)	0.67 (2.2e-08)
	Spearman	0.92 (2.8e-23)	0.91 (5.1e-22)	0.92 (5.1e-24)
VGG16	Pearson	0.61 (8.5e-07)	0.57 (4.9e-06)	0.55 (1.3e-05)
	Spearman	0.68 (1.4e-08)	0.67 (2.4e-08)	0.67 (2.0e-08)
MobileNetv2	Pearson	0.76 (1.9e-11)	0.73 (2.5e-10)	0.72 (7.4e-10)
	Spearman	0.81 (4.4e-14)	0.81 (4.9e-14)	0.83 (7.7e-15)
ViT	Pearson	0.75 (5.0e-11)	0.74 (1.5e-10)	0.74 (1.4e-10)
	Spearman	0.81 (5.9e-14)	0.82 (1.3e-14)	0.81 (4.3e-14)

Table 2: Correlation coefficients (and associated p-value) for multiple models retrained to CIFAR10 against distribution shifts. Results for more datasets and architectures are presented in the Appendix

changes between parameters' current and initial values (before model updates), we aggregate the change in parameter values between two consecutive time steps. For layer l's parameters, at time step t, Normalized Euclidean distance $E_{l,t}$ between present parameter values and parameter values in the previous time step is represented as

$$E_{l,t} = \|W_{l,t} - W_{l,t-1}\|_2 \,/\, \sqrt{\|W_{l,t}\|} \tag{6}$$

where t (≥ 1) is the current retraining time step and $t-1$ is the previous retraining time step. Distance between current parameter values $W_{l,t}$ and previous parameter values $W_{l,t-1}$ is calculated for each layer l, which are then summed and averaged by the number of layers. The normalized Euclidean distance of parameter changes is aggregated throughout the entire retraining process to obtain the final cumulative parameter changes the model undergoes.

Energy and Carbon Emission

Energy and carbon emissions provide direct real-world sustainability costs associated with training and/or retraining a model. In accordance with (Henderson et al. 2020), reporting carbon emissions is an important sustainability factor when developing models. We use CodeCarbon (Schmidt et al. 2021), a Python package, to track the energy consumption and estimate the carbon emissions of a model during retraining. The library uses geographic location information of the energy generated to calculate the estimated weight of carbon emissions. The package not only tracks energy consumption from the GPU while training models, but also tracks the energy consumption of the CPU and RAM that is expended by neural network training. The carbon emissions are calculated based on the energy generation technique of the region, such as coal, petroleum, solar, wind etc.

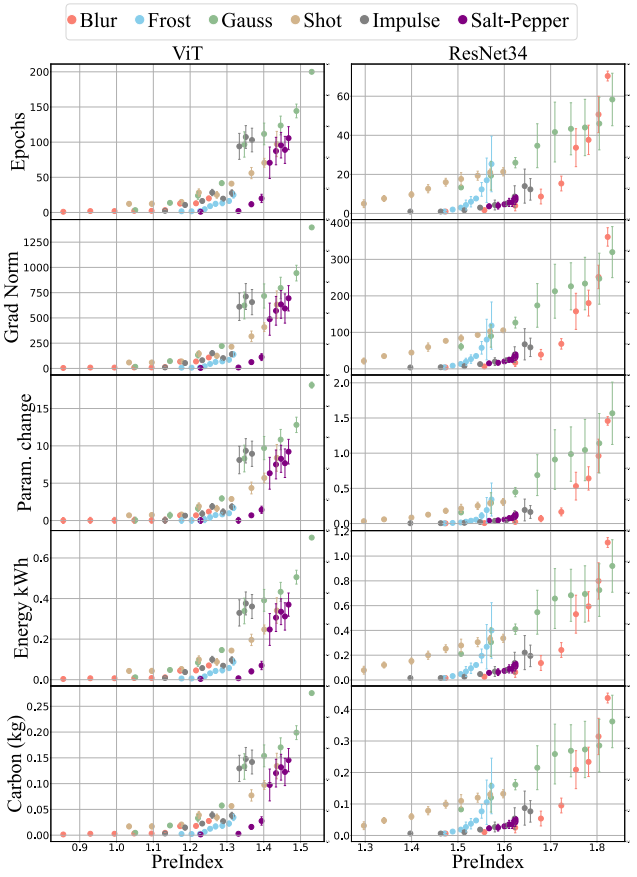

Figure 2: `PreIndex` vs. resource indicators on ResNet and ViT. Results of more architectures are presented in the Appendix.

Experiments

In this section, we evaluate `PreIndex` on three datasets – CIFAR10 (Krizhevsky, Hinton et al. 2009), CIFAR100 (Krizhevsky, Hinton et al. 2009), and TinyImageNet (Le and Yang 2015). We employ 6 noise types - Gaussian, Poisson/Shot, Blur, Frost, Salt-Pepper, and Impulse. Gaussian and Poisson/Shot are statistical noises that may arise due to errors during data capture. Blur and Frost are real-world noises due to environmental factors. Salt-Pepper and Impulse noise might occur due to artifacts or hardware issues while capturing an image. For each noise type, 9 noise intensities are generated, where the lowest level 1 is comparable to the least amount of noise and the highest level 9 is comparable to severity 4 in (Hendrycks and Dietterich 2019).

To reuse a pre-existing model, we train a randomly initial-

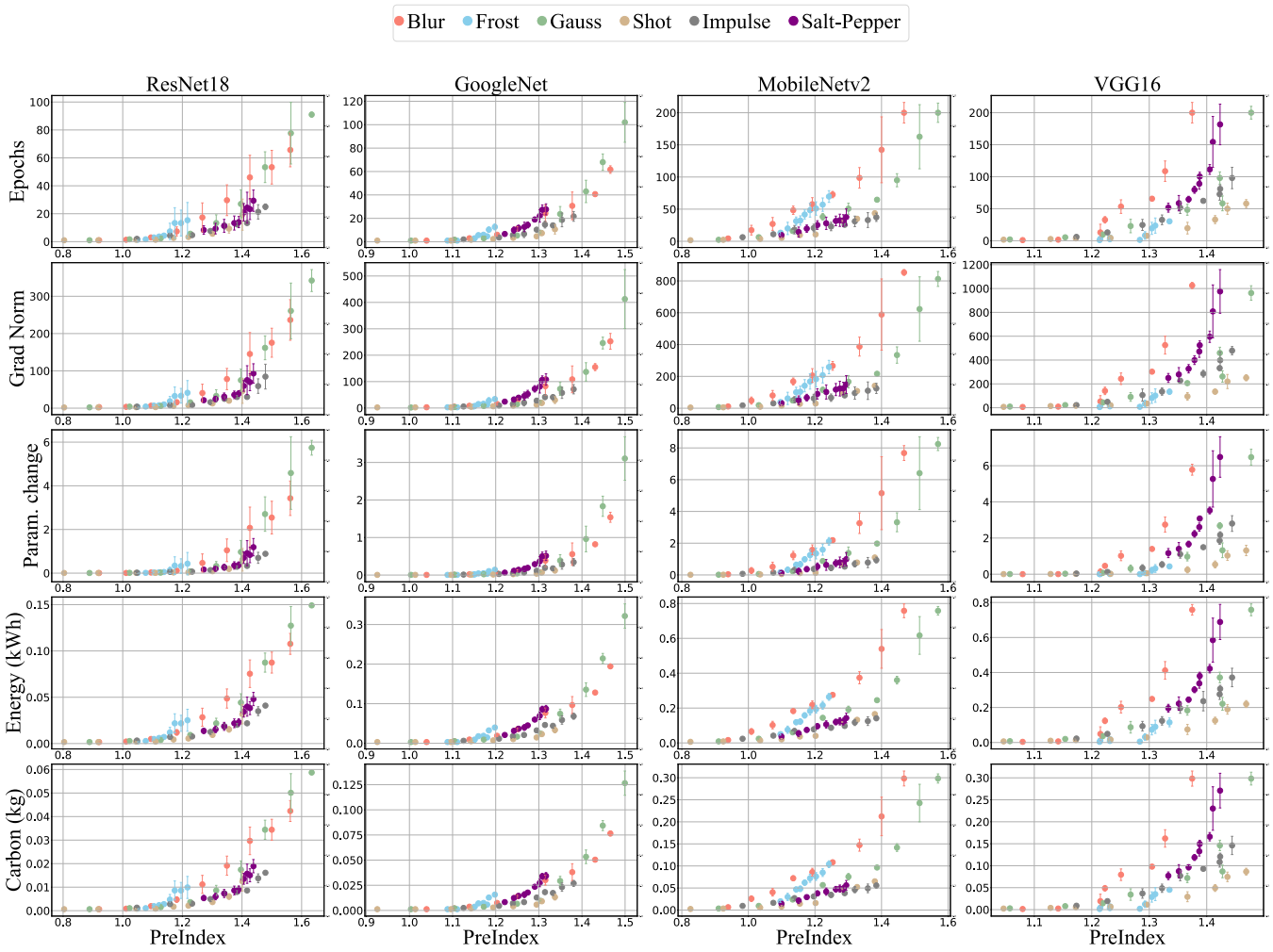

Figure 3: PreIndex vs resource indicators for various models retrained to CIFAR10 with distribution shift

Model	Correlation	Epochs	GradNorm	Param. change
ResNet18	Pearson	0.64 (1.5e-07)	0.62 (6.6e-07)	0.57 (6.0e-06)
	Spearman	0.93 (2.4e-24)	0.93 (7.6e-24)	0.93 (3.5e-24)
GoogleNet	Pearson	0.66 (6.8e-08)	0.63 (3.5e-07)	0.57 (7.0e-06)
	Spearman	0.92 (2.1e-23)	0.92 (3.1e-23)	0.92 (2.1e-23)
MobileNetv2	Pearson	0.75 (8.3e-11)	0.73 (5.1e-10)	0.70 (5.0e-09)
	Spearman	0.93 (1.3e-24)	0.93 (3.0e-24)	0.93 (5.9e-25)
VGG16	Pearson	0.67 (3.0e-08)	0.68 (1.1e-08)	0.55 (1.7e-05)
	Spearman	0.81 (1.2e-13)	0.82 (3.1e-14)	0.76 (1.6e-11)

Table 3: Correlation coefficients (and p-values) for multiple models retrained to CIFAR100 with distribution shift

ized model on the original data distribution until it reaches the minimum required accuracy for each dataset. All experiment results are an average of three runs. When training (or retraining) to higher noise levels, specific hyper-parameter tuning for each individual experiment may result in relatively fewer epochs to converge. However, this approach can impede comparison with other noise types and noise levels. For uniformity and fair comparison among all noise types and intensities, identical hyper-parameters are used.

Fig. 1 displays the various resource indicators for ResNet18 trained from scratch and retrained on CIFAR10 with different intensities of Poisson noise. It is evident that

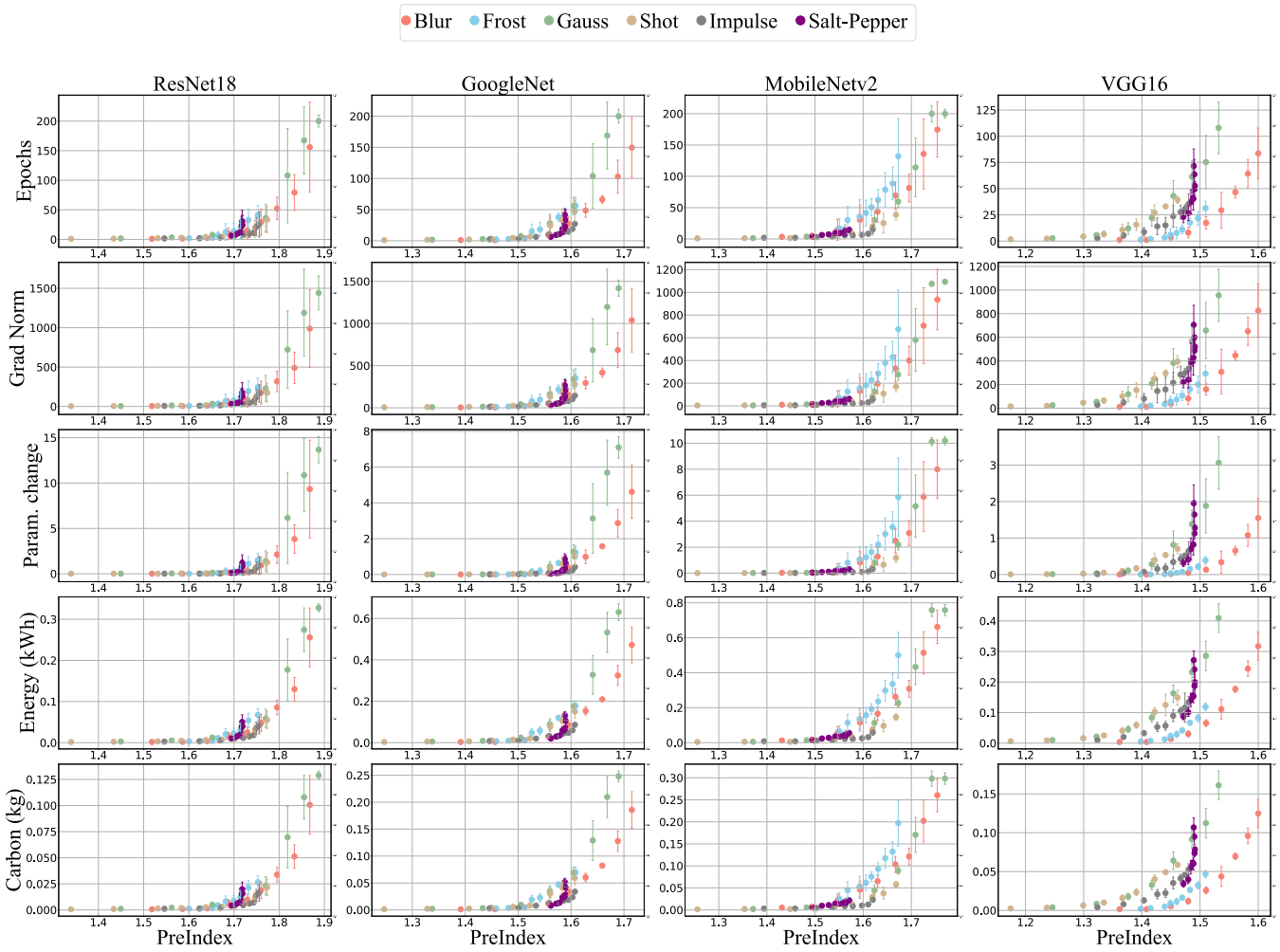

Figure 4: PreIndex vs resource indicators for various models retrained to CIFAR100 with distribution shift

retraining a model requires considerably fewer resources compared to training a new model from scratch. All the retraining indicator values for retraining are significantly lower than training from scratch.

To evaluate `PreIndex` for distribution shifts, we use Convolutional neural networks and Vision Transformers. For CNNs, we explore different model architectures - ResNets, VGG, GoogLeNet, and MobileNetv2. For Vision Transformer, we utilize a ViT model with a patch size of 4, 8 transformer blocks, a latent vector size of 512, 8 attention heads, and MLP with a hidden layer size of 1024.

Fig. 2 illustrates the correlation between `PreIndex` and all the resource indicators. In the figure, ViT and ResNet34 are retrained with distribution shifts to CIFAR10 and Tiny-ImageNet, respectively. It is evident that across the heterogeneous architectures, on various datasets, `PreIndex` has a strong correlation and aligns with all resource indicators for various types of distribution shifts.

Fig. 3 displays the correlation between `PreIndex` and all resource indicators when retraining convolutional networks to various distribution shifts on CIFAR10 dataset. Table 2 provides the Pearson correlation coefficients and Spearman correlation coefficients, with the associated p-values, between `PreIndex` and the resource indicators when retraining to CIFAR10 datasets with distribution shift. The figures and the correlation table validate that `PreIndex` has a strong correlation with the resource indicators, and that it is an effective estimator of retraining resources across various model architectures and types of distribution shifts with multiple intensities.

Fig. 4 and Table 3 provides the trend and correlation metrics between `PreIndex` and the resource indicators, respectively. With regard to all resource indicators, `PreIndex` has an increasing monotonic relation noticeable from the graphs and displays a strong positive correlation evident from the correlation coefficients. Additionally, with

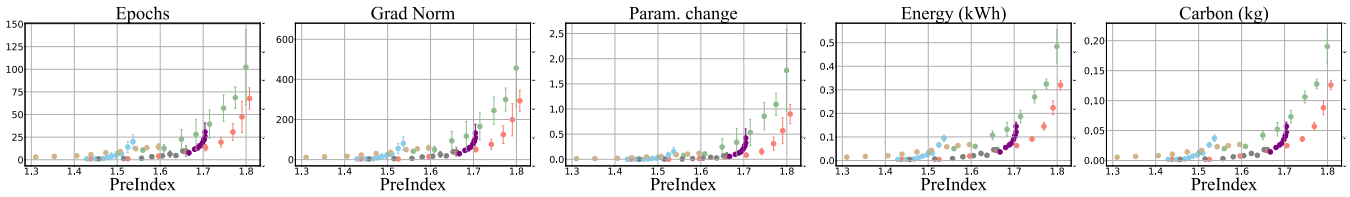

Figure 5: PreIndex vs resource indicators for GoogleNet retrained to TinyImageNet with distribution shift

Model	Correlation	Epochs	GradNorm	Param. change
ResNet34	Pearson	0.73 (1.7e-10)	0.73 (1.8e-10)	0.74 (7.7e-11)
	Spearman	0.62 (4.5e-07)	0.62 (3.8e-07)	0.68 (1.2e-08)
GoogleNet	Pearson	0.70 (2.3e-09)	0.69 (5.5e-09)	0.63 (1.9e-07)
	Spearman	0.83 (3.8e-15)	0.83 (5.4e-15)	0.83 (5.1e-15)

Table 4: Correlation coefficients (and p-values) for ResNet34 and GoogleNet retrained to TinyImageNet with distribution shift

ResNet18, GoogleNet is evaluated on the TinyImageNet. Fig. 5 and Table 4 illustrate the relation between `PreIndex` and the resource indicators for GoogleNet retrained on distribution shifted TinyImageNet.

Conclusion

We introduced a novel metric to estimate the various resources that would be expended when reusing a model by adapting to distributional shifts. We validate the effectiveness of `PreIndex` on Convolutional and Transformer based networks. We show that reusing a model by retraining requires significantly fewer resources than training a new model. The effectiveness of `PreIndex` for estimating environmental costs such as energy consumption and carbon emissions, as well as other resource indicators such as epochs, gradient norm, and model parameter change is empirically validated. All the results consistently verify that `PreIndex` is an effective estimator and has strong correlation metrics with all resource indicators. `PreIndex` is shown to be model agnostic, applicable to various datasets and effective for various types and levels of distribution shift, thus enabling sustainable decision making with regard to model reusability.

References

Abou-Moustafa, K. T.; and Ferrie, F. P. 2012. A Note on Metric Properties for Some Divergence Measures: The Gaussian Case. In Hoi, S. C. H.; and Buntine, W., eds., *Proceedings of the Asian Conference on Machine Learning*, volume 25 of *Proceedings of Machine Learning Research*, 1–15. Singapore Management University, Singapore: PMLR.

Agarwal, C.; D'souza, D.; and Hooker, S. 2022. Estimating Example Difficulty Using Variance of Gradients. arXiv:2008.11600.

Anthony, L. F. W.; Kanding, B.; and Selvan, R. 2020. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. arXiv:2007.03051.

Arjovsky, M.; Bottou, L.; Gulrajani, I.; and Lopez-Paz, D. 2020. Invariant Risk Minimization. arXiv:1907.02893.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein Generative Adversarial Networks. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 214–223. PMLR.

Arthur, D.; Vassilvitskii, S.; et al. 2007. k-means++: The advantages of careful seeding. In *Soda*, volume 7, 1027–1035.

Bannour, N.; Ghannay, S.; Névéol, A.; and Ligozat, A.-L. 2021. Evaluating the carbon footprint of NLP methods: a survey and analysis of existing tools. In Moosavi, N. S.; Gurevych, I.; Fan, A.; Wolf, T.; Hou, Y.; Marasović, A.; and Ravi, S., eds., *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, 11–21. Virtual: Association for Computational Linguistics.

Budennyy, S.; Lazarev, V.; Zakharenko, N.; Korovin, A.; Plosskaya, O.; Dimitrov, D.; Akhripkin, V.; Pavlov, I.; Oseledets, I.; Barsola, I.; et al. 2023. Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai. In *Doklady Mathematics*, 1–11. Springer.

Deng, A.; Li, X.; Hu, D.; Wang, T.; Xiong, H.; and Xu, C.-Z. 2023. Towards Inadequately Pre-trained Models in Transfer Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19397–19408.

Dodge, J.; Prewitt, T.; Tachet des Combes, R.; Odmark, E.; Schwartz, R.; Strubell, E.; Luccioni, A. S.; Smith, N. A.; DeCario, N.; and Buchanan, W. 2022. Measuring the Carbon Intensity of AI in Cloud Instances. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, 1877–1894. New York, NY, USA: Association for Computing Machinery. ISBN 9781450393522.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Faber, K.; Corizzo, R.; Sniezynski, B.; Baron, M.; and Japkowicz, N. 2021. WATCH: Wasserstein Change Point Detection for High-Dimensional Time Series Data. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE.

Ford, N.; Gilmer, J.; Carlini, N.; and Cubuk, D. 2019. Adversarial Examples Are a Natural Consequence of Test Error in Noise. arXiv:1901.10513.

Fu, A.; Hosseini, M. S.; and Plataniotis, K. N. 2021. Reconsidering co2 emissions from computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2311–2317.

García-Martín, E.; Rodrigues, C. F.; Riley, G.; and Grahn, H. 2019. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134: 75–88.

Geirhos, R.; Temme, C. R. M.; Rauber, J.; Schütt, H. H.; Bethge, M.; and Wichmann, F. A. 2020. Generalisation in humans and deep neural networks. arXiv:1808.08750.

Goyal, S.; Sun, M.; Raghunathan, A.; and Kolter, Z. 2022. Test-Time Adaptation via Conjugate Pseudo-labels. arXiv:2207.09640.

Henderson, P.; Hu, J.; Romoff, J.; Brunskill, E.; Jurafsky, D.; and Pineau, J. 2020. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *Journal of Machine Learning Research*, 21(248): 1–43.

Hendrycks, D.; Basart, S.; Mu, N.; Kadavath, S.; Wang, F.; Dorundo, E.; Desai, R.; Zhu, T.; Parajuli, S.; Guo, M.; Song, D.; Steinhardt, J.; and Gilmer, J. 2021. The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 8340–8349.

Hendrycks, D.; and Dietterich, T. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. arXiv:1903.12261.

Hendrycks, D.; Mu, N.; Cubuk, E. D.; Zoph, B.; Gilmer, J.; and Lakshminarayanan, B. 2020. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. arXiv:1912.02781.

Huang, R.; Geng, A.; and Li, Y. 2021. On the Importance of Gradients for Detecting Distributional Shifts in the Wild. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Hubert, L.; and Arabie, P. 1985. Comparing partitions. *Journal of classification*, 2: 193–218.

Kim, J.-H.; Choo, W.; and Song, H. O. 2020. Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup. In *International Conference on Machine Learning (ICML)*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Lacoste, A.; Luccioni, A.; Schmidt, V.; and Dandres, T. 2019. Quantifying the Carbon Emissions of Machine Learning. arXiv:1910.09700.

Le, Y.; and Yang, X. 2015. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7): 3.

Lee, J.; and AlRegib, G. 2020. Gradients as a Measure of Uncertainty in Neural Networks. In *IEEE International Conference on Image Processing (ICIP)*.

Lee, J.-H.; Zaheer, M. Z.; Astrid, M.; and Lee, S.-I. 2020. Smoothmix: a simple yet effective data augmentation to train robust classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 756–757.

Li, Q.; Shen, L.; Guo, S.; and Lai, Z. 2020. Wavelet integrated CNNs for noise-robust image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7245–7254.

Lim, H.; Kim, B.; Choo, J.; and Choi, S. 2023. TTN: A Domain-Shift Aware Batch Normalization in Test-Time Adaptation. arXiv:2302.05155.

Liu, X.; Shen, F.; Zhao, J.; and Nie, C. 2022. RandomMix: A mixed sample data augmentation method with multiple mixed modes. arXiv:2205.08728.

McDonald, J.; Li, B.; Frey, N.; Tiwari, D.; Gadepally, V.; and Samsi, S. 2022. Great Power, Great Responsibility: Recommendations for Reducing Energy for Training Language Models. In Carpuat, M.; de Marneffe, M.-C.; and Meza Ruiz, I. V., eds., *Findings of the Association for Computational Linguistics: NAACL 2022*, 1962–1970. Seattle, United States: Association for Computational Linguistics.

Moro, G.; Ragazzi, L.; and Valgimigli, L. 2023. Carburacy: summarization models tuning and comparison in eco-sustainable regimes with a novel carbon-aware accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 14417–14425.

Niu, S.; Wu, J.; Zhang, Y.; Chen, Y.; Zheng, S.; Zhao, P.; and Tan, M. 2022. Efficient Test-Time Model Adaptation without Forgetting. arXiv:2204.02610.

Patterson, D.; Gonzalez, J.; Le, Q.; Liang, C.; Munguia, L.-M.; Rothchild, D.; So, D.; Texier, M.; and Dean, J. 2021. Carbon Emissions and Large Neural Network Training. arXiv:2104.10350.

Raiber, F.; and Kurland, O. 2017. Kullback-Leibler Divergence Revisited. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, 117–124. New York, NY, USA: Association for Computing Machinery. ISBN 9781450344906.

Sangarya, V.; Bradford, R. M.; and Kim, J.-E. 2023. Aggregate Representation Measure for Predictive Model Reusability. In *NeurIPS 2023 Computational Sustainability: Promises and Pitfalls from Theory to Deployment*.

Santos, J. M.; and Embrechts, M. 2009. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, 175–184. Springer.

Schmidt, V.; Goyal, K.; Joshi, A.; Feld, B.; Conell, L.; Laskaris, N.; Blank, D.; Wilson, J.; Friedler, S.; and Luccioni, S. 2021. CodeCarbon: estimate and track carbon emissions from machine learning computing (2021). *DOI: https://doi. org/10.5281/zenodo*, 4658424.

Schwartz, R.; Dodge, J.; Smith, N. A.; and Etzioni, O. 2019. Green AI. arXiv:1907.10597.

Stacke, K.; Eilertsen, G.; Unger, J.; and Lundström, C. 2020. Measuring domain shift for deep learning in histopathology. *IEEE journal of biomedical and health informatics*, 25(2): 325–336.

Strubell, E.; Ganesh, A.; and McCallum, A. 2019. Energy and Policy Considerations for Deep Learning in NLP. arXiv:1906.02243.

Tolstikhin, I.; Bousquet, O.; Gelly, S.; and Schoelkopf, B. 2018. Wasserstein Auto-Encoders. In *International Conference on Learning Representations*.

Vargas, D. V.; and Su, J. 2020. Understanding the One Pixel Attack: Propagation Maps and Locality Analysis. In Espinoza, H.; McDermid, J. A.; Huang, X.; Castillo-Effen, M.; Chen, X. C.; Hernández-Orallo, J.; hÉigeartaigh, S. Ó.; and Mallah, R., eds., *Proceedings of the Workshop on Artificial Intelligence Safety 2020 co-located with the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI 2020), Yokohama, Japan, January, 2021*, volume 2640 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Verdecchia, R.; Sallou, J.; and Cruz, L. 2023. A Systematic Review of Green AI. arXiv:2301.11047.

Wang, Q.; Fink, O.; Gool, L. V.; and Dai, D. 2022. Continual Test-Time Domain Adaptation. arXiv:2203.13591.

Xu, J.; Zhou, W.; Fu, Z.; Zhou, H.; and Li, L. 2021. A Survey on Green Deep Learning. arXiv:2111.05193.

Xu, Y.; Martínez-Fernández, S.; Martinez, M.; and Franch, X. 2023. Energy Efficiency of Training Neural Network Architectures: An Empirical Study. arXiv:2302.00967.

Yarally, T.; Cruz, L.; Feitosa, D.; Sallou, J.; and van Deursen, A. 2023. Uncovering Energy-Efficient Practices in Deep Learning Training: Preliminary Steps Towards Green AI. In *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, 25–36.

Yin, D.; Lopes, R. G.; Shlens, J.; Cubuk, E. D.; and Gilmer, J. 2020. A Fourier Perspective on Model Robustness in Computer Vision. arXiv:1906.08988.

Zhang, C.; Bengio, S.; and Singer, Y. 2022. Are All Layers Created Equal? arXiv:1902.01996.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. arXiv:1710.09412.

Öcal, K.; Grima, R.; and Sanguinetti, G. 2019. Parameter estimation for biochemical reaction networks using Wasserstein distances. *Journal of Physics A: Mathematical and Theoretical*, 53(3): 034002.