A Lightweight Measure of Classification Difficulty from Application Dataset Characteristics

Bryan Bo Cao[1,2], Abhinav Sharma[1,2], Lawrence O'Gorman[1], Michael Coss[1], and Shubham Jain[2]

[1] Nokia Bell Labs, Murray Hill, NJ, USA
[2] Stony Brook University, Stony Brook, NY, USA
[1]{bryan.cao,abhinav.6.sharma}@nokia.com
[1]{larry.o_gorman,mike.coss}@nokia-bell-labs.com
[2]{boccao,abhinsharma,jain}@cs.stonybrook.edu

Although accuracy and computation benchmarks are widely available to help choose among neural network models, these are usually trained on datasets with many classes, and do not give a good idea of performance for few (< 10) classes. The conventional procedure to predict performance involves repeated training and testing on the different models and dataset variations. We propose an efficient cosine similarity-based classification difficulty measure S that is calculated from the number of classes and intra- and inter-class similarity metrics of the dataset. After a single stage of training and testing per model family, relative performance for different datasets and models of the same family can be predicted by comparing difficulty measures – without further training and testing. Our proposed method is verified by extensive experiments on 8 CNN and ViT models and 7 datasets. Results show that S is highly correlated to model accuracy with correlation coefficient $|r| = 0.796$, outperforming the baseline Euclidean distance at $|r| = 0.66$. We show how a practitioner can use this measure to help select an efficient model 6 to 29× faster than through repeated training and testing. We also describe using the measure for an industrial application in which options are identified to select a model 42% smaller than the baseline YOLOv5-nano model, and if class merging from 3 to 2 classes meets requirements, 85% smaller.

Keywords: classification difficulty · class similarity · neural network selection · image classification · efficient models.

1 Introduction

Much information is available to compare neural network models. Besides inherent features such as size and speed, model performance is measured by accuracy on public datasets. This information is invaluable for comparing models, but it is often far-removed from predicting how a model will perform on a particular application. One reason for this is that most public datasets have many classes (e.g., 1000 for ImageNet [7], 80 for COCO [26]). But many applications have far fewer classes. For example, 7 for object identification while driving [5], 6 for wildlife animal detection [35], 4 for cancer cell classification [41], and 2 each for

crowd [30], cattle [20], hardhat [55] and ship, SAR detection [27], to name but a few. Another reason is that the difficulty of the instances in the datasets is often unknown in both the benchmark and application dataset. For both these reasons, it is difficult to extrapolate from performance for the large public datasets to performance on a particular application of much different number of classes and similarity between classes.

The classification difficulty of an application depends in large part upon its dataset characteristics. For instance, an application whose classes have very similar features will generally have higher classification difficulty than for one with lower inter-class similarity. Although the general relationship between data characteristics and classification difficulty is known, there are benefits to quantifying this by means of a difficulty measure. For instance, knowing the classification performance of a model on a dataset, how will it perform on a different dataset? One could train and test, or alternatively one could compare dataset difficulties. We show how the latter is a lightweight approach requiring much less computation especially for few-class (< 10 classes) applications.

In this paper, we propose a quantitative, lightweight measure of classification difficulty for an application dataset based upon the number of classes and class similarity. Although the measure can be applied to datasets of all sizes, it is most suitable for datasets of fewer classes ($<$10), which are typical of many practical applications. For these applications, the difficulty measure can help direct a practitioner to a model whose balance of accuracy and computational efficiency meet the requirements.

Contributions of this work are summarized as follows:

1. **Analytical** – Determination of a mathematical relationship between classification difficulty and the dataset characteristics.
2. **Experimental** – Experimental results showing how dataset classification difficulty relates to model performance.
3. **Practical** – Quantifying the efficiency advantage of using dataset classification difficulty for smaller size and lower power model selection and, dataset modification.
4. **Use Case** – An industrial example of how dataset classification difficulty is used to adjust application specifications toward a more efficient model choice.

This paper is organized as follows. In Section 2, we discuss related literature on model selection and difficulty measures. In Section 3, a new measure of dataset classification difficulty is presented. We present experimental evidence of the quantitative relationship between dataset characteristics and model performance in Section 4. We show an example of how the measure is used in an industrial application in Section 5. The results are discussed and summarized in Section 6.

2 Related Work

2.1 Model Selection

In recent years, a plethora of neural architectures have been designed, trained, and made easily available such that a practitioner will usually select a model

rather than designing or training one from scratch. However, with the increasing number of efficient architectures (e.g. MobileNets [18, 43, 23] , SqueezeNet [24] ShuffleNets [56, 31] , EfficientNet [51], etc.), selecting a proper model to satisfy an application's requirements becomes even more challenging. To select a model, existing approaches can be categorized into four options: 1) off-the-shelf, 2) transfer learning plus targeted training, 3) scaled model selection, and 4) selection from model repository (with caveats as described below).

For the first option, numerous off-the-shelf models [11, 19, 10] have been trained on a dataset containing the same classes as the application, for instance for pedestrian detection [8], flower classification [54], fish classification [50], and food detection [49]. Although this can save substantial time from data collection and training, it often fails in real-world applications due to a feature shift in deployed environments due to such factors as different camera capture angles, backgrounds, scales, etc.

A second option is transfer learning [58] by which a model is trained on a larger, standard dataset such as ImageNet [7] or COCO [26], stripped of its classification layer (leaving the backbone), then fine-tuned on objects of the application of interest. A drawback to this popular choice is that the backbone incorporates extraneous features than are often needed for the application classes. Although transfer learning facilitates the selection of a neural model with high accuracies, the model will inevitably be larger than one trained on only the classes of interest for the same level of accuracy.

A third option is to choose from a scaled model family – a collection of models that share the same general architecture, but whose size (width and depth) are adjusted with a scaling factor. Examples include the EfficientNet family [51] from B0 to B7, and YOLO family [22] from nano to extra-large scales. This paper focuses on efficiency for small, practical applications. Therefore, experiments on the smaller sizes of the EfficientNet image classifier and the YOLO object detector [12] are chosen for testing our small model regime.

The fourth option is to select a pre-trained model from a model repository [14, 57]. This is a fast way to start using a model, but it is different than the focus of this paper in two ways. Our focus is on selecting efficient models to match dataset characteristics. The most efficient models must be trained on the dataset of the application. This is unlike a model repository whose models may be efficient, but are unlikely for the application dataset specifically. The second difference is that our methods are directed to the *data side* versus the *model side* [21, 46]. This enables different application datasets to be compared by their classification difficulty rather than different models to be compared by their performance on pre-trained datasets.

2.2 Image Classification Difficulty

No matter which type of classifier is used, the empirically observed behavior of classifiers is strongly data dependent. Previous to the widespread adoption of neural networks, classification difficulty was largely measured by the ability to distinguish classes volumetrically in multi-dimensional feature space. A classical

measure is Fisher's Discriminant Ratio, by which a large difference in class means and small sum of their variances describes a less difficult classification problem [40]. In [17], complexity measures are described that include feature overlap, feature efficiency, separability of classes, and geometry of the class manifolds. Although the embedding space of a neural network is also a feature space, neural networks often have much higher dimensionality of nonintuitive (machine-learned) features, which have the ability to better distinguish highly non-linear class boundaries, thus leading to neural network classification difficulty measures different from these previous measures.

A number of image difficulty measures have been proposed. For metric learning [2, 29, 9], the loss function is set to minimize the similarity between images of the same class during training, where similarity is measured as the dot product between embeddings (usually from the last hidden layer of the model) of two images [52, 15, 37]. Another way to measure difficulty is by classification error on a difficulty-scaled range of datasets [44, 33, 39] or models [6]. Machine difficulty scores can also be used to prune filters associated with easier features during training [38], and by more highly weighting filters of difficult features during inference [32].

One difference from these previous papers is that we focus upon *classification difficulty*. Many references calculate *single-image* difficulty for purposes of curriculum, or simple-to-difficult, learning [52, 1, 15, 37] and scaled model selection [48]. In [32], intra-class difficulty is measured for the purpose of weighting classes differently during training. In contrast to single-image difficulty, we incorporate intra- and inter-class similarity in determining a difficulty measure for application datasets containing many images of multiple classes, as shown in Section 3. And is Section 4, we show by experiment how the measure varies for different numbers of classes, similarities, models, and datasets.

3 Dataset Classification Difficulty Measure

Cosine similarity is a common measure used to quantify the similarity between two feature vectors,

$$\mathrm{S}(\mathbf{x}_i, \mathbf{x}_j) = \cos(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{||\mathbf{z}_i||\,||\mathbf{z}_j||}, \tag{1}$$

where $\mathbf{z}_i$ and $\mathbf{z}_j$ are feature vectors of image i and j respectively.

Whereas equation 1 is the similarity between two vector instances, we are interested in the average similarity between pairs of instances in the same class, and pairs of instances between classes respectively,

$$\textbf{intra-class: } S_R(C) = \frac{1}{n_1} \sum_{i,j \in C, i \neq j} S(\mathbf{x}_i, \mathbf{x}_j) \tag{2}$$

$$\textbf{inter-class: } S_E(C_a, C_b) = \frac{1}{n_2} \sum_{i \in C_a, j \in C_b} S(\mathbf{x}_i, \mathbf{x}_j) \tag{3}$$

where n_1 is the number of intra-class pair combinations from a single class set of instances $\mathbf{x} \in C$, and n_2 is the number of inter-class pair combinations between instances of two classes $x_i \in C_a$ and $x_j \in C_b$.

For a classification problem with N_{CL} classes, the average intra- and inter-class similarities are respectively,

$$\bar{S}_R = \frac{1}{n_3} \sum S_R, \qquad \bar{S}_E = \frac{1}{n_4} \sum S_E, \tag{4}$$

where n_3 is the number of classes N_C and $n_4 = \binom{N_C}{2} = N_C(N_C - 1)/2$ is the number of combinations of class pairs without repetition.

Our earlier results (summarized in Table 1 in later sections) show that classifying a dataset is difficult when images in a class are similar to other classes. This observation implies that a dataset's difficulty is directly related to inter-class similarity and inversely to the intra-class similarity. Based on this, our difficulty measure design rationale is to capture both types of similarity jointly, summarized as **weighted similarity score** $\bar{S}$. To further ensure the score falls in a consistent range, we additionally design a measure, dubbed **soft similarity score** $\hat{S}$ that normalizes the weighted intra- and inter-class similarity scores by its maximum. Formally, we define $\bar{S}$ and $\hat{S}$ as,

$$\bar{S} = \frac{1 + \lambda_s \bar{S}_R - (1 - \lambda_s)\bar{S}_E}{2}, \qquad \hat{S} = \frac{\lambda_s \bar{S}_R - (1 - \lambda_s)\bar{S}_E}{max(\lambda_s \bar{S}_R, (1 - \lambda_s)\bar{S}_E)} \tag{5}$$

where λ_s is a weighting factor (default to 0.5) to balance $\bar{S}_R$ and $\bar{S}_E$.

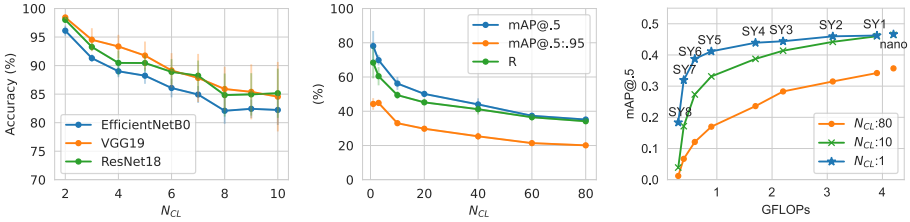
4 Experiment

In this section, we begin with experiments showing the effect of the number of classes on model accuracy. We then add similarity, and in the last subsection show how the combined difficulty measure is used for a real application.

4.1 Number of Classes

It is known that accuracy reduces when more classes are involved, or equivalently a larger model is needed to maintain the same accuracy. This is because more visual features are needed to separate the classes and the decision boundary is more complex accordingly. We perform experiments in this section, first to confirm this relationship empirically, and second to gain a more quantitative insight into how the relationship changes across the range of few to more classes.

We performed four sets of experiments. The first was for object detection using the YOLOv5-nano [22] backbone upon randomly-chosen, increasing-size class groupings of the COCO dataset [26]. Ten groups with N_{CL} of {1, 2, 3, 4, 5, 10, 20, 40, 60, 80} were prepared. For each group, we trained a separate YOLOv5-nano model from scratch. we set the initial learning rate as 0.01 with weight decay 0.0005 at image size 640 using SGD optimizer. As seen in Fig. 1 (a), accuracy decreases with number of classes as expected. But perhaps not



(a) N_{CL} on CIFAR10 (b) N_{CL} on COCO (c) sub-YOLO on COCO

Fig. 1. Overall relationship of performance versus the number of classes N_{CL}. Each dot denotes an average of 3 subsets for each N_{CL}, while error bars represent standard deviations (each multiplied by 5 in (b) for visibility). (a) Image classification accuracy decreases for the classifiers tested when the number of CIFAR-10 classes is increased from 2 to 10. (b) Object detection accuracy and recall (R) decrease when the number of COCO classes is increased from 1 to 80. (c) Accuracy plot for increasingly smaller models from YOLOv5-nano through eight sub-YOLO models (SY1-8) and class groupings of 1 (N_{CL}:1), 10 (N_{CL}:10), and 80 (N_{CL}:80).

anticipated is the fact that the accuracy decrease is steep for very few classes, say 5-10 or fewer, and flattens beyond 10. We will examine later in the paper the difference between classification with few versus more classes.

The second set of experiments is for image classification on the CIFAR-10 dataset. With many fewer classes in CIFAR-10 [25] than COCO (10 versus 80), we expect to see how the number of classes and accuracy relate for this smaller range. We extracted subsets of classes — which we call groups – from CIFAR-10 with N_{CL} ranging from 2 to 9. For example, one group with $N_{CL} = 4$ contains *airplane*, *cat*, *automobile*, and *ship* classes. We trained 3 classifiers from scratch for each group, EfficientNet-B0, VGG19 [47], and MobileNet V2 [43]. Results of the image classification experiments are shown in Fig. 1 (middle). The classifiers used for testing, showed the expected trend of accuracy reduction as N_{CL} per group increased. However, the trend was not as monotonic as might be expected. We hypothesized that this might be due to the composition of each group. Class groupings were randomly chosen, so they have different levels of inter-class similarity. We explore how inter-class similarity affects accuracy in the next section.

The third set of experiments involves reducing model size for classifying different numbers of classes and measuring accuracy versus computation effort in GFLOPS. We prepared 90 random class groups from the COCO minitrain dataset [42]. There are 80 groups with $N_{CL} = 1$, each containing a single class from 80 classes. There are 8 groups with $N_{CL} = 10$. The final dataset is the original COCO minitrain with $N_{CL} = 80$. We scale YOLOv5 layers and channels down in model size with the depth and width factors already used for scaling the family up in size from nano to x-large. Starting with depth and width multiples of 0.33 and 0.25 for YOLOv5-nano, we reduce these in step sizes of 0.04 for depth and 0.03 for width. In this way, we design a monotonically decreasing sequence

of sub-YOLO models denoted as SY1 to SY8. We train each model separately for each of the six groupings. Results of sub-YOLO detection are shown in Fig. 1 (right). There are three lines where each point of $mAP@.5$ is averaged across all models in all datasets for a specific N_{CL}. An overall trend is observed that fewer-class models (upper-left blue star) achieve higher efficiency than many-class models. Another important finding is that, whereas the accuracies for 80 classes drops steadily from the YOLOv5-nano size, accuracy for 10 classes is fairly flat down to SY2, which corresponds to a 36% computation reduction, and for 1 class down to SY4, which corresponds to a 72% computation reduction.

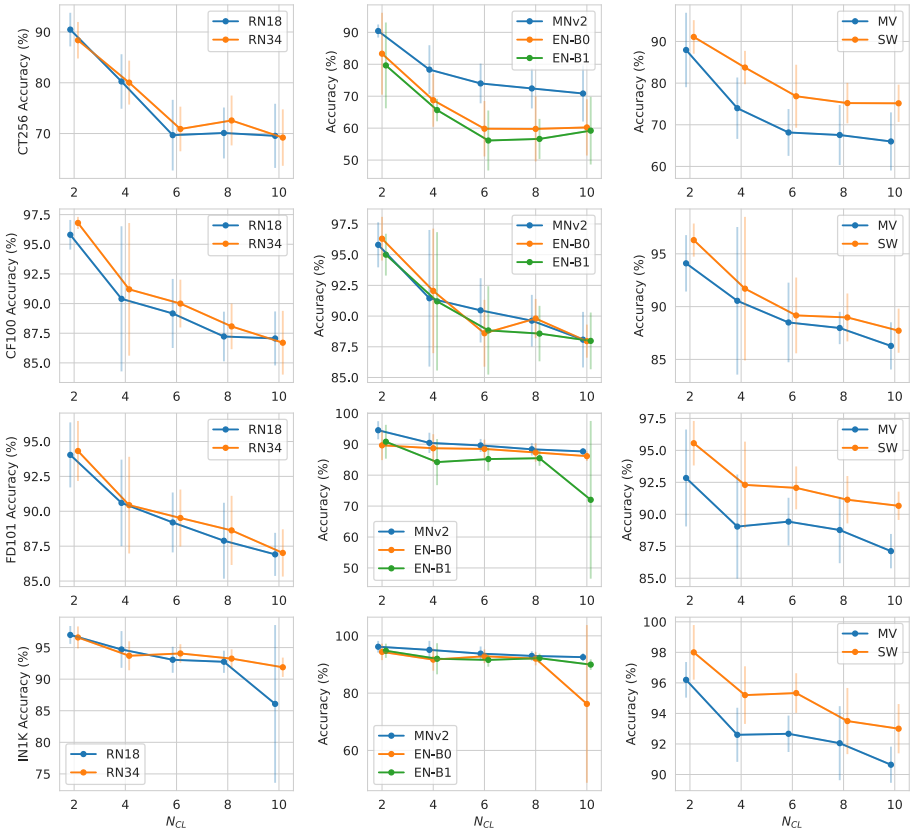


Fig. 2. Overall trend of 5 efficient CNNs and 2 ViTs on 4 datasets: model performance tends to decrease while N_{CL} increases. Each dot denotes the average accuracy of 5 subsets for N_{CL}. The error bars represent standard deviations of accuracy in 5 subsets. Accuracy: Top-1 Accuracy. RN: ResNet. MNv2: Mobilenet V2. EN: EfficientNet. MV: MobileViT. SW: Swin Transformer. CT256: CalTech256, CF100: CIFAR100, FD101: Food101, IN1K: ImageNet1K.

In the fourth set of experiments, we extend to 7 models on 4 datasets, covering 5 Convolutional Neural Networks (CNNs) and 2 Vision Transformers (ViTs) shown in Fig. 2, totaling 1225 training/testing runs. Models include ResNet18, ResNet34 [16], MobileNet V2 [43], EfficientNet-B0, EfficientNet-B1 [51], MobileViT [34] and Swin Transformer [28]. Transformers are included for completeness. However, they are initially designed for scaling up, which is not efficient as the focus of this paper. Therefore a lightweight ViT MobileViT is chosen. We select datasets consisting of natural images, including CalTech256 [13], CIFAR100 [25], Food101 [4] and ImageNet1K [7]. We vary the number of classes N_{CL} from 2 to 10 with step size 2. For each N_{CL}, 5 subsets are randomly selected (seed number 0-4) from the full dataset. A model is trained and converged in each subset, then the validation Top-1 accuracy is reported. By default, we use SGD optimizer with learning rate 0.1, momentum 0.9 and weight decay 0.0001 to train our models.

The importance of the findings in this section should be emphasized for few-class, practical applications where computational efficiency and low energy use is required. Not only is accuracy higher for fewer-class application datasets, but the practitioner can choose smaller and smaller models with small accuracy penalty, but much smaller energy use (as shown in the right plot of Fig. 1).

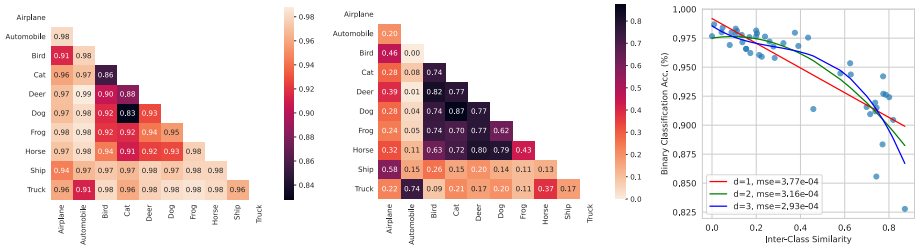
4.2 Intra- and Inter-Class Similarity

We start with an experiment for didactic purposes to indicate how our subjective notion of similarity relates to accuracy. We test with 3 models, EfficientNet-B0 (EB0), VGG-19 (V19), and MobileNet V2 (MV2). Table 1 shows accuracy results for groupings of 2 and 4 classes from the CIFAR-10 dataset, which we have subjectively attributed similarity values of "yes" and "no". For each model, the accuracies in bold are the highest for their groupings, corresponding to the class groupings of lower subjective similarity. In the final column (I-CS), the objective inter-class similarity scores also correspond to our subjective designations.

nCl	Classes	Similarity	EB0	V19	MV2	I-CS
4	cat, deer, dog, horse	yes	0.84	0.86	0.76	0.57
4	airplane, cat, auto, ship	no	**0.91**	**0.94**	**0.93**	0.12
2	deer, horse	yes	0.92	0.94	0.89	0.61
2	auto, truck	yes	0.91	0.95	0.93	0.56
2	airplane, frog	no	**0.98**	**0.98**	**0.96**	0.11
2	deer, ship	no	**0.98**	**0.98**	**0.96**	0.08

Table 1. Accuracies for three image classifiers (EB0, V19, MV2) for class groupings of $nCL = 2$ and 4 whose similarities are initially subjectively assigned, but are supported by Inter-Class Similarity (I-CS), an objective measurement in the final column.

To illustrate the relationship between class similarity scores and accuracy, we calculate these using EfficientNet-B0 for all pairwise classifications in CIFAR-10



(a) Inter-Class **Sim.** (b) Bi-Classification **Acc.** (c) Rel. of **Sim.** & **Acc.**

Fig. 3. Matrices showing relationships among pairs of classes: (a) binary classification accuracy matrix using EfficientNet-B0, (b) binary-class similarity matrix with S_E metric, (c) nonlinear relationship between binary classification accuracy (a) and similarity scores (b). The polynomial function with a degree of $d = 3$ (blue) has least mse compared with $d = 2$ (green) and $d = 1$ (red). Sim.: Similarity. Acc.: Accuracy. Rel.: Relationship.

and show results in Fig. 3. The left matrix shows accuracy results between pairs of classes. The middle matrix shows average intra-class similarity scores on the diagonal and inter-class scores in the off-diagonal boxes. The absolute value of the Pearson correlation coefficient [3] |r| between the binary classification accuracy (left matrix) and the similarity scores (middle matrix) is **0.77**, showing **strong correlation** between these two measures. All (similarity, accuracy) pairs are shown in the plot (right) clearly indicating the strong inverse correlation. Of note, the lowest similarity data point in the top left of the plot is for the (automobile, deer) pair, and the highest similarity data point in the bottom right is for the (cat, dog) pair.

We further explore the type of relationship between similarity and accuracy in terms of nonlinearity. Specifically, in the data points in Fig. 3 (c), we fit three polynomial functions $f(s) = p_0 x^d + ... + p_{(d-1)}x + p_d$ with degree $d \in \{1, 2, 3\}$. To measure these functions accuracy, we compute the Mean Squared Error $mse = \frac{1}{N}\sum_i^N (acc'_i - acc_i)^2$ between the prediction of accuracy (acc') and ground truth of accuracy (acc) with inter-class similarity as input. Results show that degree of 3 has the least $mse = 2.93 \times 10^{-4}$ compared to degrees of 2 ($mse = 3.16 \times 10^{-4}$) and 1 ($mse = 3.77 \times 10^{-4}$), respectively. The nonlinearity (e.g. $d = 3$) between similarity and accuracy can suggest that when images in a dataset become similar and similar in a certain scale, the expected gain of a model's accuracy is diminished. This insight can assist the estimation of an efficient neural network's performance in a real-world scenario under some resource-constraints.

4.3 Similarity Metric Efficiency

To select a neural network model for an application, a common first step is to identify models that attain the requirements. In this paper, we focus on efficient applications, so appropriate selections include small model families such as YOLO, MobileNet, and EfficientNet-B0 (classification or object detection variants of

these depending upon the goal of the application). A model would be chosen, trained and tested. Another model, larger or smaller depending upon results of the first choice, would be trained and tested, and compared. This training and testing cycle would continue until the model closest the the application requirements is found.

Use of the difficulty measure offers a much faster way to find an appropriate model. If a model of interest has been previously trained and tested on an application, and the difficulty measure of that application has been determined, then the practitioner can do the following. Calculate the difficulty measure of the current application dataset. If the measure is higher than previously, then the application will likely require a larger model, and if not a smaller model. This procedure guides the selection process up or down the model family levels, where the advantage is that training and testing are not required, just similarity score measurement.

Note that this is a relative measurement procedure, which needs to start with a model already trained-and-tested, but for a developer who has done many applications, this is already available. Furthermore, one could project that practitioners would publish similarity scores and associated model accuracies for applications they have done for the benefit of other practitioners. To be clear, the application classes do not have to be the same, just the relative difficulty measures.

The traditional alternative is to perform training and inference testing all combinations of attribute values. For N_{CL} classes, binary classification of pairs requires $\binom{N_{CL}}{2}$ training and inference operations. In comparison, for the difficulty measure, we need to train once for all combinations of binary classifications (the same training that would happen traditionally). Then, instead of testing, only the pairwise similarities between pairs needs by performed, an expense of one vector multiply each, rather than a full neural network test requiring the number of multiplies of the model. For subsequent tests of different datasets, training does not have to be repeated; instead the cached latent space is used for pairwise similarity tests.

An example of runtime comparison for the CIFAR10 dataset is shown in Table 4.3. We compare against three convolutional neural networks (CNNs) designed for small or embedded applications. Both CNN and similarity metric approaches require initial training. For the similarity metric, there is an additional one-time task of feeding all test images into the similarity metric, which takes 0.72 seconds, and then caching, which takes 0.63 seconds. Applying the similarity metric takes 0.76 seconds to calculate a similarity score for each pair of images. In comparison for conventional testing, each image must undergo full CNN testing to obtain its predicted accuracy.

4.4 Difficulty Measure Evaluation

The purpose of a difficulty measure is to provide a means to estimate a model's performance. To that end, we employ the Pearson correlation coefficient [3] r to measure the correlation between difficulty and accuracy. A higher value

Model	t_{train} (s/epoch)	t_{test} (s/pair)
VGG19	0.69	4.49
EfficientNet-B0	3.13	21.82
MobileNet V2	2.19	15.05
Similarity Metric	3.31	**0.76**

Table 2. Results of runtime comparison of Similarity Metrics and some popular CNN models. s: second, pair: all pair of instances in two classes.

of the absolute value $|r|$ indicates a stronger correlation of two vectors, while $|r| > 7$ is commonly considered as **strongly correlated** [53] [45]. Since we target efficiency, we use the smallest models in ResNet18 and EfficientNet-B0 in this experiments. CalTech256 is selected due to its focused object categories, balanced class distribution and consistent labels, which are suitable for resource-constrained research.

In the following experiments, similarity is computed in the latent space of image encoder in DINOv2 [36] – a large-scale model that is commonly used to extract robust image features in recent research.

Difficulty Method Comparison: We compare our proposed similarity-based measures $\bar{S}$ and $\hat{S}$ against Euclidean distance baselines $\bar{D}$ and $\hat{D}$ in DINOv2 latent space, where,

$$\bar{D} = \frac{1 + \lambda_d \bar{D}_E - (1 - \lambda_d)\bar{D}_R}{2}, \qquad \hat{D} = \frac{\lambda_d \bar{D}_E - (1 - \lambda_d)\bar{D}_R}{max(\lambda_d \bar{D}_E, (1 - \lambda_d)\bar{D}_R)} \tag{6}$$

Results in Table 3 demonstrate that our proposal S ($|r| > 0.7$) outperforms the baseline D ($|r| < 0.7$).

	|r|			
λ	$\bar{D}$	$\bar{S}$	$\hat{D}$	$\hat{S}$
0.25	0.546	0.757 (+0.211)	0.600	0.788 (+0.188)
0.50	0.696	0.789 (+0.093)	0.660	0.796 (+0.136)
0.75	0.691	0.762 (+0.071)	0.660	0.796 (+0.136)

Table 3. Comparison of similarity-based S and Euclidean distance-based D difficulty measure with various λ by the absolute value of Pearson correlation coefficient $|r|$.

Effect of λ: We vary the weight λ that balances S_R and S_E (D_R and D_E for D. Results in Table 3) show that both reach the highest scores when $\lambda = 0.5$, specifically $\bar{S} = 0.789$, $\hat{S} = 0.796$, $\bar{D} = 0.696$, $\hat{D} = 0.660$, respectively.

Difficulty Measure Ablation Study: We ablate components in $\bar{S}$ defined in Equation 5. In particular, ablating $\bar{S}_R$ reduces the absolute Pearson correlation coefficient $|r|$ from 0.789 to 0.692, while $|r|$ decreases to 0.702 by removing $\bar{S}_E$.

$w/o\ \bar{S}_R\ (\lambda_s = 0)$	$w/o\ \bar{S}_E\ (\lambda_s = 1)$	$\lvert\bar{S}$
0.692	0.702	$\lvert$0.789

Table 4. Ablation Study of $\bar{S}$ by Pearson correlation coefficient |r|.

Results in both Table 3 and 4 verify our design intuition that jointly considering intra- and inter-class difficulty measures is beneficial.

Besides helping select an appropriate model there is another practical use for difficulty measurement of an application dataset. Sometimes the requirements for an application offer options to enable trading off one requirement for another. For example, application requirements might allow two options, one for classification of 5 classes at some accuracy level and cost ceiling, and another for classification of 4 classes at the same accuracy level but at a lower cost. An industry example of this requirement tradeoff is given in Section 5.

5 Classication Difficulty for an Industry Application

We have applied the classification difficulty measure to model selection for an industry application involving video analytics for human-robot interaction.

N_{CL}	S	Class Label	Ym (21.1M)	Ys (7.2M)	Yn (1.9M)	Y-1 (1.1M)	Y-2 (0.16M)	Y-3 (0.07M)
3	0.18	p-walk	0.748	0.728	0.727	0.725	0.665	0.654
		p-cart	0.720	0.690	0.674	0.681	0.576	0.500
		robot	0.865	0.872	0.827	0.82	0.747	0.635
		mAP	0.778	0.764	0.743	0.742	0.663	0.596
2	0.15	person	0.753	0.752	0.732	0.719	0.691	0.657
		robot	0.872	0.875	0.827	0.814	0.755	0.650
		mAP	**0.812**	**0.813**	**0.779**	**0.766**	**0.723**	**0.654**
			4.4%	6.4%	4.9%	3.2%	9.1%	9.7%

Table 5. Results of detection of YOLOv5 medium (Ym), small (Ys), nano (Yn) and the sub-YOLO models (Y-1, Y-2, Y-3) on 3-class (person-walk, person-cart, robot) and 2-class (person, robot) cases respectively. The bracketed numbers below the model names are their model sizes. The accuracy is mAP@0.5. The numbers in the second from bottom row are in bold to show that 2-class accuracy is higher than for 3-class, and the numbers in the bottom row show the percentage improvement. The red numbers show that the sub-YOLO1 model can achieve similar accuracy to the YOLOs model, but with 6.5× smaller size when class grouping is reduced from 3 to 2. The bottom row shows the accuracy improvement from 3 to 2 classes for each model.

The objective is to recognize human activity from fixed hallway cameras of an assembly factory so as to reduce human-robot interaction (HRI). Three classes were identified and trained for this application, *person-walk*, *person-cart*, and *robot*. The *person* class was initially separated into two, *person-walk* and *person-cart* (person walking and person pushing a cart). This was because this distinction was deemed useful – and we didn't want to retrain if we just trained on two classes at the outset.

Results in Table 5 show in general that the 3-class group with similarity value 0.18 has lower accuracy across models than the 2-class group with similarity value 0.15. For the 3-class option, one good choice that balances accuracy and size would be the sub-YOLO1 model, whose accuracy is just $0.743 - 0.742 = 0.001$ less than the YOLO-nano model, but whose size is $1.1/1.9 = 0.579$ (or 42%) of the YOLO-nano. When the *person-walk* and *person-cart* classes are merged into a single *person* class, then sub-YOLO1 could be chosen with essentially the same accuracy as YOLOs, but with 85% smaller size.

6 Conclusion

The difficulty measure proposed here provides a relative measure that, knowing the performance of a model for one dataset, one can predict the model performance for the same dataset on different models of a model family or on other datasets on the same model.

In this paper, we have proposed a measure of dataset classification difficulty based upon three characteristics of a dataset, number of classes, intra-class similarity, and inter-class similarity. We have experimented with 9 neural network models on 7 datasets to demonstrate the relationship between model accuracy and dataset difficulty. Our proposed similarity-based method outperforms the baseline using Euclidean distance in terms of correlation with accuracy by Pearson correlation coefficient. We have shown the utility of the difficulty measure in guiding a practitioner to an efficient model architecture without repeated training and testing for different datasets.

7 Acknowledgement

This research has been supported in part by the National Science Foundation (NSF) under Grant No. CNS-2055520.

References

1. Appalaraju, S., Chaoji, V.: Image similarity using deep cnn and curriculum learning. arXiv preprint arXiv:1709.08761 (2017)
2. Bellet, A., Habrard, A., Sebban, M.: A survey on metric learning for feature vectors and structured data. arXiv preprint arXiv:1306.6709 (2013)
3. Benesty, J., Chen, J., Huang, Y., Cohen, I.: Pearson correlation coefficient. In: Noise reduction in speech processing, pp. 1–4. Springer (2009)

4. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – mining discriminative components with random forests. In: European Conference on Computer Vision (2014)
5. Cai, Y., Luan, T., Gao, H., Wang, H., Chen, L., Li, Y., Sotelo, M.A., Li, Z.: Yolov4-5d: An effective and efficient object detector for autonomous driving. IEEE Trans. on Instrumentation and Measurement **70**, 1–13 (2021)
6. Chang, Y.J., Hong, D.Y., Liu, P., Wu, J.J.: Efficient inference on convolutional neural networks by image difficulty prediction. In: 2022 IEEE Int. Conf. on Big Data (Big Data). pp. 5672–5681 (2022)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
8. Du, X., El-Khamy, M., Lee, J., Davis, L.: Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection. In: 2017 IEEE Winter Conf. on Applications of Computer Vision (WACV). pp. 953–961 (2017)
9. Duffner, S., Garcia, C., Idrissi, K., Baskurt, A.: Similarity Metric Learning, pp. 103–125. Springer Int. Publishing, Cham (2021)
10. Face, H.: Hugging face model hub (August 22 2023), https://huggingface.co/models, accessed on 2023-08-22
11. Foundation, T.L.: Pytorch model hub (August 22 2023), https://pytorch.org/hub/, accessed on 2023-08-22
12. Ganesh, P., Chen, Y., Yang, Y., Chen, D., Winslett, M.: Yolo-ret: Towards high accuracy real-time object detection on edge gpus. In: Proc. IEEE/CVF Winter Conf. on Applications of Computer Vision. pp. 3267–3277 (2022)
13. Griffin, G., Holub, A., Perona, P.: Caltech 256 (Apr 2022). https://doi.org/10.22002/D1.20087
14. Guo, P., Hu, B., Hu, W.: Sommelier: Curating DNN models for the masses. In: Proc. 2022 Int. Conf. on Management of Data. p. 1876–1890. SIGMOD '22, Association for Computing Machinery (2022)
15. Hannemose, M.R., Sundgaard, J.V., et al.: Was that so hard? estimating human classification difficulty. In: Wu, S., et al. (eds.) Appl.s of Medical Artificial Intelligence. pp. 88–97. Springer Nature Switzerland (2022)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
17. Ho, T.K., Basu, M.: Complexity measures of supervised classification problems. IEEE Trans. on Pattern Analysis and Machine Intelligence **24**(3), 289–300 (2002)
18. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
19. Inc., G.: Tensorflow model hub (August 22 2023), https://www.tensorflow.org/hub, accessed on 2023-08-22
20. J.G.A. Barbedo, L.V. Koenigkan, T.S.P.S.: A study on the detection of cattle in uav images using deep learning. Sensors **19**(24) (2019)
21. Jia, H., Chen, H., Guan, J., Papernot, N.: A zest for LIME: Toward architecture-independent model distances. In: ICLR 2022 - 10th Int. Conf. on Learning Representations. p. 1876–1890. Virtual, France (Apr 2022)
22. Jocher, G., et. al.: ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support (Oct 2021)
23. Koonce, B., Koonce, B.: Mobilenetv3. Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization pp. 125–144 (2021)

24. Koonce, B., Koonce, B.: SqueezeNet. Springer (2021)
25. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
26. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conf. on computer vision. pp. 740–755. Springer (2014)
27. Liu, S., Kong, W., Chen, X., Xu, M., Yasir, M., Zhao, L., Li, J.: Multi-scale ship detection algorithm based on a lightweight neural network for spaceborne sar images. Remote Sensing **14**(5) (2022)
28. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)
29. Lu, J., Hu, J., Zhou, J.: Deep metric learning for visual understanding: An overview of recent advances. IEEE Signal Processing Magazine **34**(6), 76–84 (2017)
30. M. Sabokrou, M. Fayyaz, M.F.Z.M.R.K.: Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. Computer Vision and Image Understanding **172**, 88–97 (2018)
31. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proc. European Conf. on computer vision (ECCV). pp. 116–131 (2018)
32. Marsden, M., McGuinness, K., et al.: Investigating class-level difficulty factors in multi-label classification problems. In: 2020 IEEE Int. Conf. on Multimedia and Expo (ICME). pp. 1–6 (2020)
33. Meding, K., Buschoff, L.M.S., Geirhos, R., Wichmann, F.A.: Trivial or impossible — dichotomous data difficulty masks model differences (on imagenet and beyond). In: Int. Conf. on Learning Representations (2022)
34. Mehta, S., Rastegari, M.: Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. arXiv preprint arXiv:2110.02178 (2021)
35. Nguyen, H., Maclagan, S.J., Nguyen, et al.: Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring. In: 2017 IEEE Int. Conf. on Data Science and Advanced Analytics. pp. 40–49 (2017)
36. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
37. Peng, B., Islam, M., Tu, M.: Angular gap: Reducing the uncertainty of image difficulty through model calibration. MM '22, Association for Computing Machinery, New York, NY, USA (2022)
38. Pentsos, V., Spantidi, O., Anagnostopoulos, I.: Dynamic image difficulty-aware DNN pruning. Micromachines **14**(5) (2023)
39. Pliushch, I., Mundt, M., Lupp, N., Ramesh, V.: When deep classifiers agree: Analyzing correlations between learning order and image statistics. In: Avidan, S., et al. (eds.) ECCV. pp. 397–413. Springer Nature Switzerland (2022)
40. Richard O. Duda, P.E.H.: Pattern Classification and Scene Analysis. Wiley-Interscience (1973)
41. Salman, M., Çakar, G., Azimjonov, J., Kösem, M., Cedimoğlu, I.: Automated prostate cancer grading and diagnosis system using deep learning-based yolo object detection algorithm. Expert Systems with Applications **201**, 117148 (2022)
42. Samet, N., Hicsonmez, S., Akbas, E.: Houghnet: Integrating near and long-range evidence for bottom-up object detection. In: Eur. Conf. Comp. Vis. (ECCV) (2020)

43. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
44. Scheidegger, F., Istrate, R., Mariani, G., Benini, L., Bekas, C., Malossi, C.: Efficient image dataset classification difficulty estimation for predicting deep-learning accuracy. In: The Visual Computer. vol. 37, pp. 1593–1610 (2021)
45. Schober, P., Boer, C., Schwarte, L.A.: Correlation coefficients: appropriate use and interpretation. Anesthesia & analgesia **126**(5), 1763–1768 (2018)
46. Shah, H., Park, S.M., Ilyas, A., Madry, A.: ModelDiff: A framework for comparing learning algorithms. In: Proc. 40th Int. Conf. on Machine Learning. Proc.of Machine Learning Research, vol. 202, pp. 30646–30688 (23–29 Jul 2023)
47. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
48. Soviany, P., Ionescu, R.T.: Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction. In: 2018 20th Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). pp. 209–214 (2018)
49. Subhi, M.A., Md. Ali, S.: A deep convolutional neural network for food detection and recognition. In: 2018 IEEE-EMBS Conf. on Biomedical Engineering and Sciences (IECBES). pp. 284–287 (2018)
50. Tamou, A., Benzinou, A., Nasreddine, K., Ballihi, L.: Transfer learning with deep convolutional neural network for underwater live fish recognition. In: 2018 IEEE Int. Conf. on Image Processing, Appl.s and Systems (IPAS). pp. 204–209 (2018)
51. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: Int. Conf. on machine learning. pp. 6105–6114 (2019)
52. Tudor Ionescu, R., Alexe, B., Leordeanu, M., Popescu, M., Papadopoulos, D.P., Ferrari, V.: How hard can it be? estimating the difficulty of visual search in an image. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (June 2016)
53. Wicklin, R.: Weak or strong? how to interpret a spearman or kendall correlation. https://blogs.sas.com/content/iml/2023/04/05/interpret-spearman-kendall-corr.html (2024), https://blogs.sas.com/content/iml/2023/04/05/interpret-spearman-kendall-corr.html, accessed on 2024-06-04
54. Wu, Y., Qin, X., Pan, Y., Yuan, C.: Convolution neural network based transfer learning for classification of flowers. In: 2018 IEEE 3rd Int. Conf. on Signal and Image Processing (ICSIP). pp. 562–566 (2018)
55. Y. Li, H. Wei, Z.H.J.H.W.W.: Deep learning-based safety helmet detection in engineering management based on convolutional neural networks. Advances in Civil Engineering **2020**, 88–97 (2020)
56. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proc. IEEE Conf. on computer vision and pattern recognition. pp. 6848–6856 (2018)
57. Zhou, Z.H., Tan, Z.H.: Learnware: small models do big. Science China Information Sciences **67**, 1869–1919 (2023)
58. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proc. IEEE **109**(1), 43–76 (2021)