Difficulty Estimation with Action Scores for Computer Vision Tasks

Octavio Arriaga[1], Sebastian Palacio[2], Matias Valdenegro-Toro[3]
[1]University of Bremen, [2]German Research Center for Artificial Intelligence, [3]University of Groningen
arriagac@uni-bremen.de, sebastian.palacio@dfki.de, m.a.valdenegro.toro@rug.nl

Abstract

As more machine learning models are now being applied in real world scenarios it has become crucial to evaluate their difficulties and biases. In this paper we present an unsupervised method for calculating a difficulty score based on the accumulated loss per epoch. Our proposed method does not require any modification to the model, neither any external supervision, and it can be easily applied to a wide range of machine learning tasks. We provide results for the tasks of image classification, image segmentation, and object detection. We compare our score against similar metrics and provide theoretical and empirical evidence of their difference. Furthermore, we show applications of our proposed score for detecting incorrect labels, and test for possible biases.

1. Introduction

Current state-of-the-art algorithms in machine learning rely on the use of overparametrized models such as deep neural networks (DNNs). These architectures contain sufficient structural priors to reduce the solution space to a computable and generalizable one, but not restricted enough to prevent them from learning unstructured data nuances [19, 24] and surface statistics imperceptible to humans [8, 13]. Moreover, as these models are now being deployed into real world scenarios, it has become critical to create a systematic evaluation of their performance that can discover these unwarranted properties. This is particularly complicated due to the vast amount of samples required to optimize current machine learning models.

Entropy is a useful metric that provides additional predictive information about the uncertainty of individual samples. However, we provide evidence suggesting that entropy does not provide all relevant dimensions required to asses incorrect samples or possible biases. Furthermore, current state-of-the-art DNNs have been shown to predict uncalibrated confidences [9]. This limitation has called for better uncertainty prediction techniques. However, these methods can become computationally expensive since they often require model ensembles or multiple forward passes [10]. Additional metrics have been defined in the literature, but they have been only applied either to small datasets or are restricted to a single task [1, 3]. Furthermore, they have not presented a principled definition based on desired properties, but rather have relied on hypotheses specific to gradient descent methods.

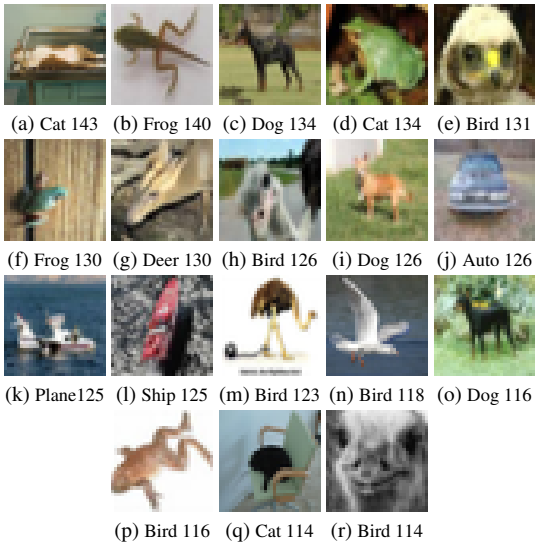


Figure 1. Hardest samples in the CIFAR10 test split using a ResNet architecture. Each subcaption shows the true label and the computed action score of that sample. This exemplifies how the action score measures learning difficulty, as these images are all in uncommon object poses, show ambiguous objects, or are mislabeled samples (e.g d and p).

In this work we provide a formal definition of difficulty based on a variational principle. In the context of machine learning this implies that samples that do not conform to the optimization dynamics must be considered unnatural or difficult. We show that our method can provide machine learning researchers and engineers with an additional tool that inspects incongruous samples and biases in datasets and models.

We summarize our contributions as follows: We present

a computational method based on the physical quantity *Action* ($\mathscr{A}$) that determines how difficult it was to learn a sample. We provide an information theoretic formulation in order to understand its differences to other known quantities such as information gain and entropy. We provide experimental results in multiple supervised and unsupervised machine learning tasks such as image classification, object detection, image segmentation and dimensionality reduction. Finally, we apply our method as a human auditing tool to inspect incorrectly labeled samples and possible biases in a model.

2. Related work

Per-sample metrics have been proposed in order to characterize the uniqueness of each sample. In [14] the input space is represented by a small set of typical and atypical per-class examples as means to explain a model's generalization capabilities. By taking into consideration different training dynamics (i.e., metrics that depend on the model as it trains), [20] can split a dataset into hard- or easy-to-learn samples, spotting OOD points, as well as identifying mislabeled samples.

In order to tune a model's uncertainty estimation [22] proposes a metric for difficulty by adding an additional output branch that increases the weight for samples with low entropy, according to the original classifier. Similar architectures have been tested under the paradigm of curriculum learning where either a teacher network adapts the importance of samples to a student model [12] or concurrently trained networks dynamically estimate how hard a given batch will be for the other members of the ensemble [11]. While the use of per-sample loss values is widespread in the literature, these metrics often rely on the modification or an addition to the original model [11, 12, 22]. Moreover they are often formulated for an individual task such as image classification [1, 3, 14]. Thus in this work we compare our metric against the predicted entropy given it's semantic similarity and generic applicability to any model or task.

Furthermore, estimating the influence of difficult samples on a model's prediction has found numerous applications in machine learning, ranging from outlier detection, to sample selection, or curriculum learning. [16] proposes a method to select a subset of the dataset that maximizes the average performance of an ensemble on a binary task. A similar study proposes several gradient-based metrics to quantify the number of redundant samples in a given dataset [21].

3. Action Scores for Difficulty Estimation

We first motivate our proposed technique by observing how the loss decreases over different samples. Most common losses take the mean per batch or display the total mean loss of a whole data split, which can in turn overshadow the training dynamics. We hypothesize that easy samples should have a fast decreasing loss, while hard samples should exhibit a slow decreasing loss or, in some circumstances, an increasing value during the optimization.

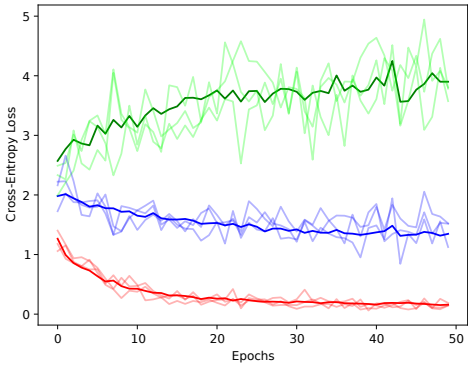


Figure 2. Clustering of per-sample loss curves on CIFAR10 with Keras CNN, and three cluster centers (red, green, and blue). Each cluster center is shown with the top three closest curves. This diagram shows the motivation for the action score, as per-sample loss behavior across epochs is different and measures how difficult is each sample to be learned by the model.

We perform a simple experiment by training a small VGG-like network with 5 layers on CIFAR10. Using this model we compute the loss for each sample at each epoch. We cluster these curves using K-Means with $k = 3$. We can observe in Figure 2 clustered loss curve patterns (increasing, slow decrease or constant, and decrease) which we interpret as samples having different difficulties (hard, medium, easy).

Given our previous observations we proceed by creating a principled definition of difficulty rather than relying on proxy metrics. We define difficulty as the total amount of effort required to solve a task. Within the context of machine learning we define the effort $\mathscr{E}$ to be proportional to the product of the accumulated loss over a period of time $\mathscr{L}_t$ and that length of time Δt:

$$\mathscr{E}_t = \mathscr{L}_t \times \Delta t. \tag{1}$$

The correlation between loss and time in equation 1 implies that if a loss was small but the time required to compute it was long, then the effort must grow proportionally to that time. Following our definition, we can formally define the difficulty of a sample as the total effort over time [1]:

$$\mathscr{A} = \sum_{t=0}^{T} \mathscr{E}_t = \sum_{t=0}^{T} \mathscr{L}_t \times \Delta t. \tag{2}$$

[1]This quantity resembles the functional definition of the action in physics. The principle of least action states that the action is stationary to a first order. However, within our framework we have a variational principle that states that this integral should be minimized. Given these similarities we name our difficulty score as *action* ($\mathscr{A}$).

In the specific case of neural networks given a loss function $\mathscr{L}$ and a model m with free parameters θ_n, we define the action $\mathscr{A}$ of a sample $x \in X$ and label $y \in Y$ within an optimization step n as

$$\mathscr{A}(x) = \sum_{n=0}^{N} \mathscr{L}(y, m(x; \theta_n))\Delta n. \tag{3}$$

We define an optimization step to be an epoch, and the value of that step to be equal to one. The granularity of this step is chosen for computational convenience; however, one could also choose a Δt that represents difficulty at every batch update, or even the actual processing (wall) time of a single epoch or batch.

An additional characteristic of equation 3 is that it can be ordered across different sets. For example, in a classification task one can compute the difficulty across classes x_c. In a segmentation task one can compute a difficulty mask by calculating a per pixel difficulty x_{ij}, or also across models m_i. Consequently, for this specific case with neural networks, the action of a sample is the accumulated loss over all epochs. Therefore, samples with a higher accumulated loss represent samples that are more difficult to learn. Within this framework we can also recover sample pairs that accumulate the least amount of loss during optimization. These samples reflect which elements are easier to learn as well as possible biases that might be present in the model or the dataset. We would like to emphasize that this method can be applied to any learning algorithm that is optimized iteratively, and is not limited to artificial neural networks nor supervised methods. Moreover, calculating $\mathscr{A}$ does not require any modification to the original model.

4. Experimental Results

We start by showing the theoretical differences between the predicted entropy $\mathrm{H}(P(y_s|x_s, \theta^*))$ and $\mathscr{A}$. First, we formulate the loss $\mathscr{L}$ as a probabilistic loss function. This implies that $\mathscr{L}$ compares the predicted probability density function against the *true* density function using a divergence. For the specific case of the Kullback-Leibler divergence (information gain) we have:

$$\mathscr{A}(x_s) = \sum_{n=0}^{N} D_{\mathrm{KL}}(P(y_s|\theta^*) \parallel P(\hat{y_s}|x_s, \theta^n))\Delta n, \tag{4}$$

Using the relation $D_{\mathrm{KL}}(P \parallel Q) = \mathrm{H}(P, Q) - \mathrm{H}(P)$ we obtain:

$$\begin{aligned}\mathscr{A}(x_s) = \sum_{n=0}^{N} &\mathrm{H}(P(y_s|\theta^*), P(\hat{y_s}|x_s, \theta^n))\Delta n \\ &- N\mathrm{H}(P(y_s|\theta^*), P(y_s|\theta^*))\Delta n.\end{aligned} \tag{5}$$

The last term to the right is often ignored from the optimization procedure since it does not depend on any optimizable parameters. We disregard this value because it does not depend on a sample x_s. Thus, the difficulty of a sample is:

$$\mathscr{A}(x_s) = \sum_{n=0}^{N} \mathrm{H}(P(y_s|\theta^*), P(\hat{y_s}|x_s, \theta^n))\Delta n. \tag{6}$$

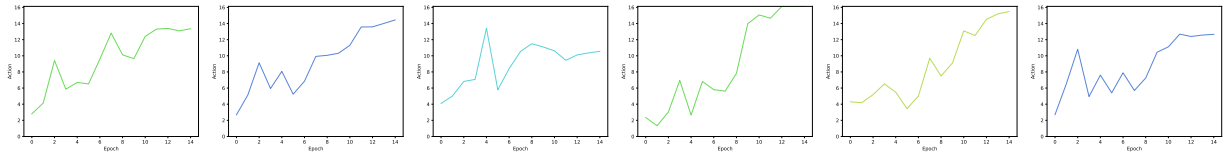
Hence, $\mathscr{A}$can be interpreted as the accumulated message length during optimization by using the running distribution $P(\hat{y_s}|x_s, \theta^n)$. This result shows the differences between $\mathscr{A}$ and entropy when using a KL divergence. From these results we can also interpret the action as the accumulated discrepancy while learning. Thus, if our hypotheses are constantly unable to explain our observations while learning, we deem that concept as hard, or ourselves as incapable of learning. In the following sections we show experimental results on both of these possibilities.

4.1. Image classification

We test our method in the following standard classification datasets: MNIST [6], FashionMNIST [23], KuzushijiMNIST [4], FERPlus [2] and CIFAR10 [15], using multiple CNN models: ResNet, Xception and a simple VGG-like network. In this section we will only show our results on the ResNet models trained on CIFAR10, but all additional results can be found in the supplementary material.

At every epoch we calculate and store the loss (cross entropy) of each sample in the test set. After the conclusion of the training phase we compute the action of each sample by summing up the stored losses. In Figure 1 we display samples with the highest action scores. We observe that our method successfully detects a mislabeled sample in the CIFAR10 test dataset (1d). In section 4.4 we conduct a more thorough test that measures the ability of our metric to successfully recover mislabeled samples. Figure 4 shows samples with the lowest action score for this experiment. We note that those samples correspond to only automobiles showing at least two wheels. Figure 3 shows the loss across epochs of the top hardest samples from the same experiment shown in Figure 1. We can observe that in all samples the loss does increase throughout the optimization process, as suggested in our original discussion shown in Figure 2.

In order to discover possible biases within our hardest samples, we compute for each sample its closest neighbors in feature space. These results are shown in Figure 33 (in the appendix). Neighbors were computed for the same test split using the L2 norm of the last feature layer before the final prediction. For our trained ResNet model in CIFAR10 this corresponds to a 256 dimensional vector. Note that the neighbors of all hard samples belong to the same class (except for the query sample itself). For example, the hardest sample on the first row corresponds to a *cat* lying horizontally, but the neighbors are all *ships*. Similarly, the hard sample on the second row is a *frog* while all neighbors are *airplanes*.



(a) Cat 143 (b) Frog 140 (c) Dog 134 (d) Cat 134 (e) Bird 131 (f) Frog 130

Figure 3. Loss function across epochs from the top hardest samples in CIFAR10 test split using a ResNet architecture, with the true class and its action score.

(a) Auto 0.00 (b) Auto 0.00 (c) Auto 0.00 (d) Auto 0.00 (e) Auto 0.01 (f) Auto 0.01 (g) Auto 0.01 (h) Auto 0.01 (i) Auto 0.01

Figure 4. Easiest samples in CIFAR10 test split using a ResNet architecture. Each subcaption shows the true label and the computed action score of that sample.

Figure 6a shows the distribution of the action scores per class. We can observe that the classes in CIFAR10 that accumulate more loss correspond to those associated with animals rather than objects. Figure 6b shows the joint distribution of the normalized action score and the entropy of the whole test set. Entropy was calculated using an ensemble of 20 ResNets, and the normalized action score was computed by first taking the mean of the action scores across all 20 models and then performing a min-max normalization. We observe that the distribution of the action score (y-axis) exhibits a heavy tail. Furthermore, this figure shows that the single dimension of entropy cannot cover the space of difficult samples as defined by the proposed action score. Figure 6c shows the accuracy of the test set with respect to thresholded values of the normalized action score, the normalized entropy and a modified version of the action score that uses the variance across the optimization step instead of the sum. We can observe that in all circumstances the classification accuracy decreases as lower values of our metrics get masked out. We display both the mean and the 95% confidence interval across all 20 models. We can observe that the confidence interval of the entropy increases in the last thresholds, while the interval of action scores remains smaller, with a faster decreasing mean. In order to further investigate the visual differences between action scores and entropy we show in Figure 7 the samples indicated by the green and red squares of Figure 6b. These squares represent regions of interest: high-action low-entropy and low-action high-entropy respectively.

4.2. Object detection

In this experiment we calculate the action score of a multi-objective loss function used for training the single-shot object detector SSD300 [17] on PASCAL VOC 2007 [7]. The total loss of this model consists of the combination of three different losses: positive classification, negative classification and bounding box regression. The samples with the most and least localization loss are shown in Figure 8. We can observe that the most difficult samples for box regression correspond to images that contain indistinguishable small objects. Moreover, easier samples for the same loss are determined by single centered objects. Figure 9 shows the hardest positive classification loss. Within this figures we can observe that 9a and 9b display object-classes that are presented in non-conventional situations. In Figure 10 we observe the easiest positive samples. We see that single centered persons and cats are easier for the model to classify. The images for the negative classification loss are shown in the supplementary.

4.3. Segmentation

In this section we provide results using our action score in a new different setting. Additionally to ordering the per-sample action scores, we can also compute a per-pixel accumulated loss:

$$\mathscr{A}(x_{ij}) = \sum_{n=0}^{N} \mathscr{L}(y_{ij}, m(x_{ij}, \theta_n)). \tag{7}$$

This change allows us to evaluate the difficulty not only as global property of a sample but also the local elements within it. For this purpose we trained a U-NET architecture with a VGG backbone on the CityScapes segmentation dataset [5] using a linear combination of a dice, jaccard and focal loss. We compute our action scores as well as our per-pixel action scores in the CityScapes validation set. These results can be seen in 11.

We can see in the first two rows of Figure 11 that the most difficult samples contain large red *void* masks which

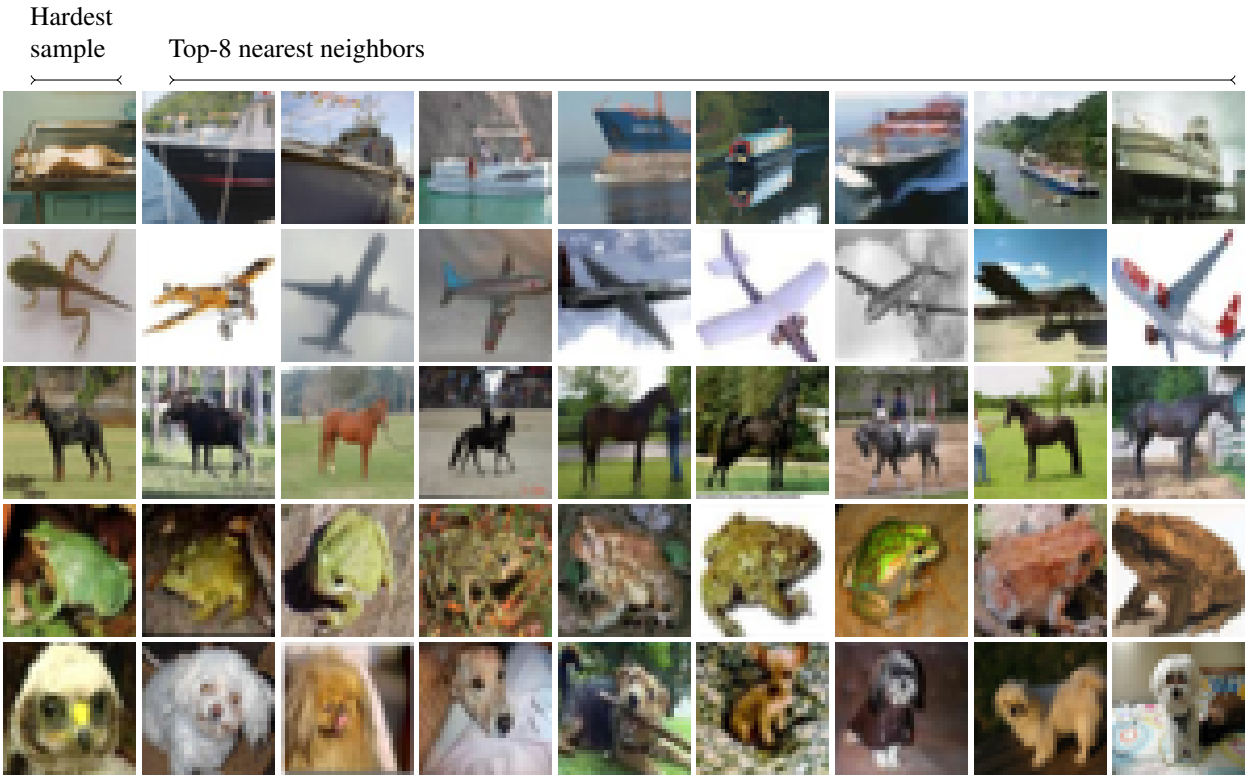
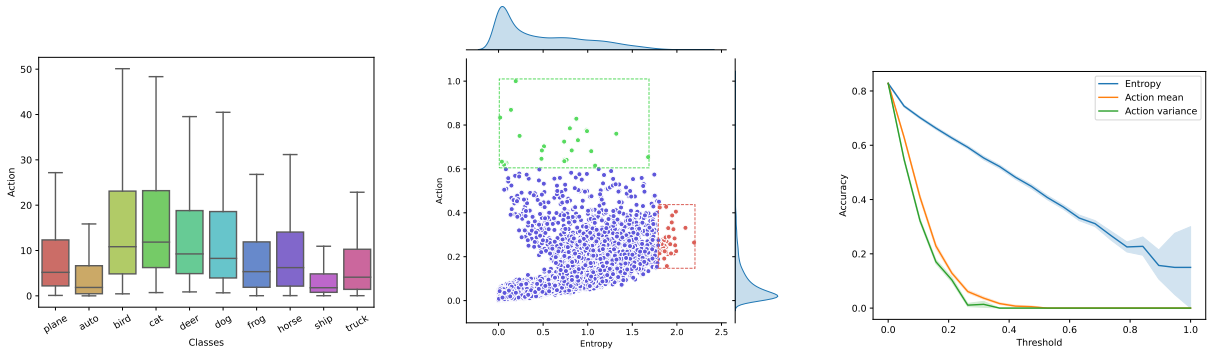


Figure 5. The first element of each row corresponds to one of the top five hardest samples on CIFAR10 test split with a ResNet model. The subsequent elements in each row show the top eight neighbors of that first row element.



(a) Action per class (b) Action vs Entropy (c) Accuracy as function of metrics

Figure 6. Data exploration of data scores on CIFAR10. (a) shows the relative difficulty of each class, with birds and cats being the most difficult to classify. (b) Shows the joint distribution of entropy vs action score, displaying how action and entropy measure different properties of a prediction. (c) Shows relationship of accuracy as function of action, accuracy drops with higher action as those examples are more difficult to learn.

the model correctly classifies as a *road*. The other two rows show samples which contain multiple pedestrians. Furthermore, our per-pixel masks allows us to indicate which parts are effectively more difficult. We can now see that in the last two rows the regions with the most accumulated loss are those in which the ground include vegetation and a baby wagon is labeled as *void*. Additional results using our per-pixel score for dimensionality reduction can be found in the supplementary material.

4.4. Training on Noisy Labels

A critical problem for models trained on large supervised datasets is the quality of the annotations. Badly annotated samples may introduced unintended biases, and hinder the

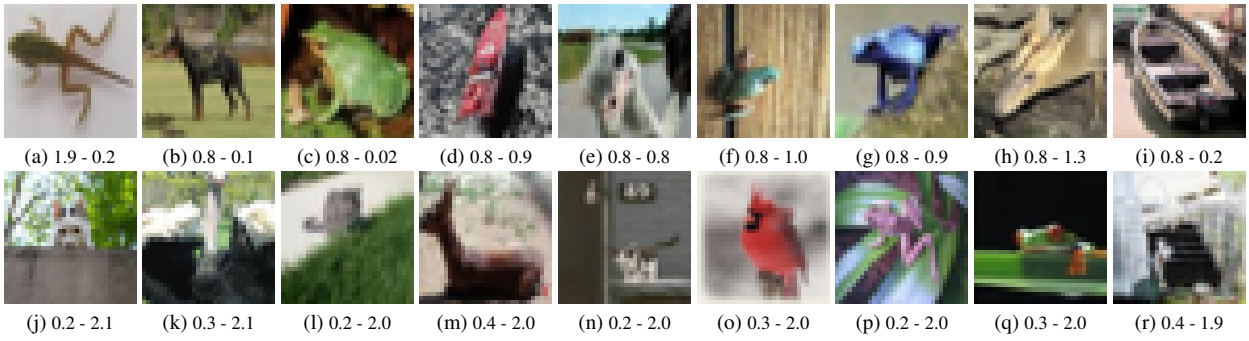


Figure 7. Comparison of action score vs entropy. Each subcaption shows the normalized action score and the entropy of that sample in the following format: {*action-entropy*}. The top row shows the mean of the normalized action samples in CIFAR10 test split using 20 ResNet architectures. The bottom row shows the samples with the highest entropy using the same 20 models as an ensemble.

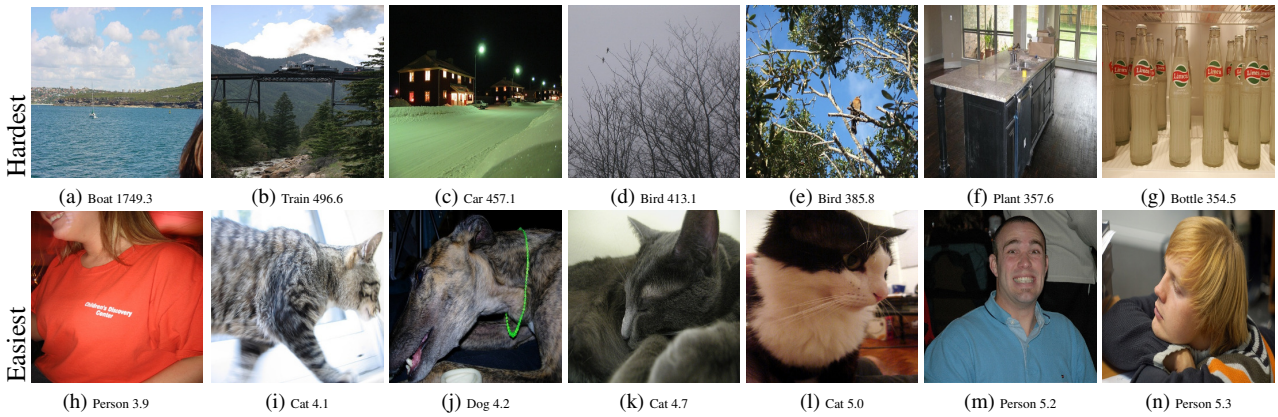


Figure 8. Most difficult (top-row) and easiest examples (bottom-row) in the VOC 2007-VAL with the SSD localization loss. The *action scores* are displayed below each image as well as the true label.

performance of a model. One of the first lines of defenses against this issue is to detect precisely the samples that have been mislabeled. We follow the experimental setup from [24], assigning random labels to 10% of CIFAR10's training dataset. Next, a small 3-layer CNN and a ResNet are trained for 85 epochs yielding a high training accuracy. By plotting the distribution of the action score on the training subset that has randomly assigned labels (Figure 12), we show that those samples are clearly differentiable (larger action score) from correctly labeled samples used for training (low action score), as well as samples used for validation (i.e., not used for training).

4.5. Curriculum by Action Score

We explore the potential for using the action score as a metric for doing dataset selection when training is constrained by a computational budget. Similar to [16], we are interested in measuring the accuracy of a neural network when training on a subset of the available data. We test whether a subset made out of samples sorted by action score yields a higher accuracy (on the entire test set) compared to the best of three randomly sampled subset of the same size. Once the training set is sorted according to the action score, the data is split into equally sized chunks with a uniform distribution across classes. A 3-layer CNN is then trained (from scratch using the same initial weights) on CIFAR-10 over 15 epochs using one chunk. The chunk with the highest test accuracy is kept, and subsequent runs will append one of the remaining chunk to the best performing one on the previous iteration.

Results in Figure 13 show that using data sorted by action score yield consistently higher accuracy on the entire test set than a randomly selected subset. Moreover, we observe that there is a tendency to select chunks that steadily increase the mean action score, peaking shortly before the entire dataset is exhausted (right after chunk 19 is added to the set). This coincides with a slower increase over the subsequent improvements that additional data bring afterwards.

4.6. Comparing Model Biases

One question of broad interest in the context of difficulty is, do different models have the same biases on the same

Figure 9. Hardest Examples on PASCAL VOC 2007 with SSD validation positive loss.

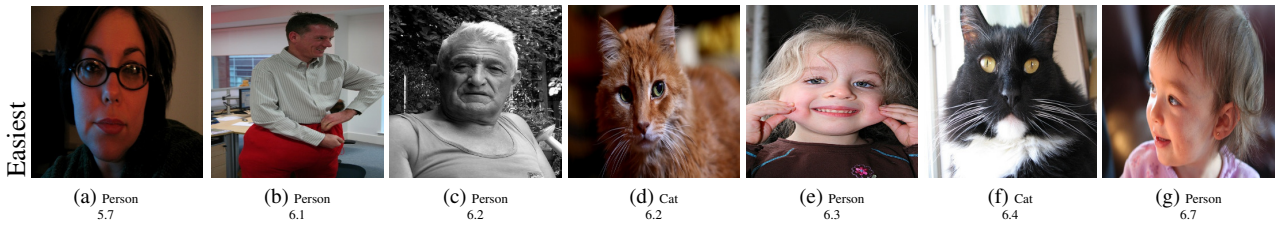


Figure 10. Easiest Examples on PASCAL VOC 2007 with SSD validation positive loss. Action score is included in each caption.

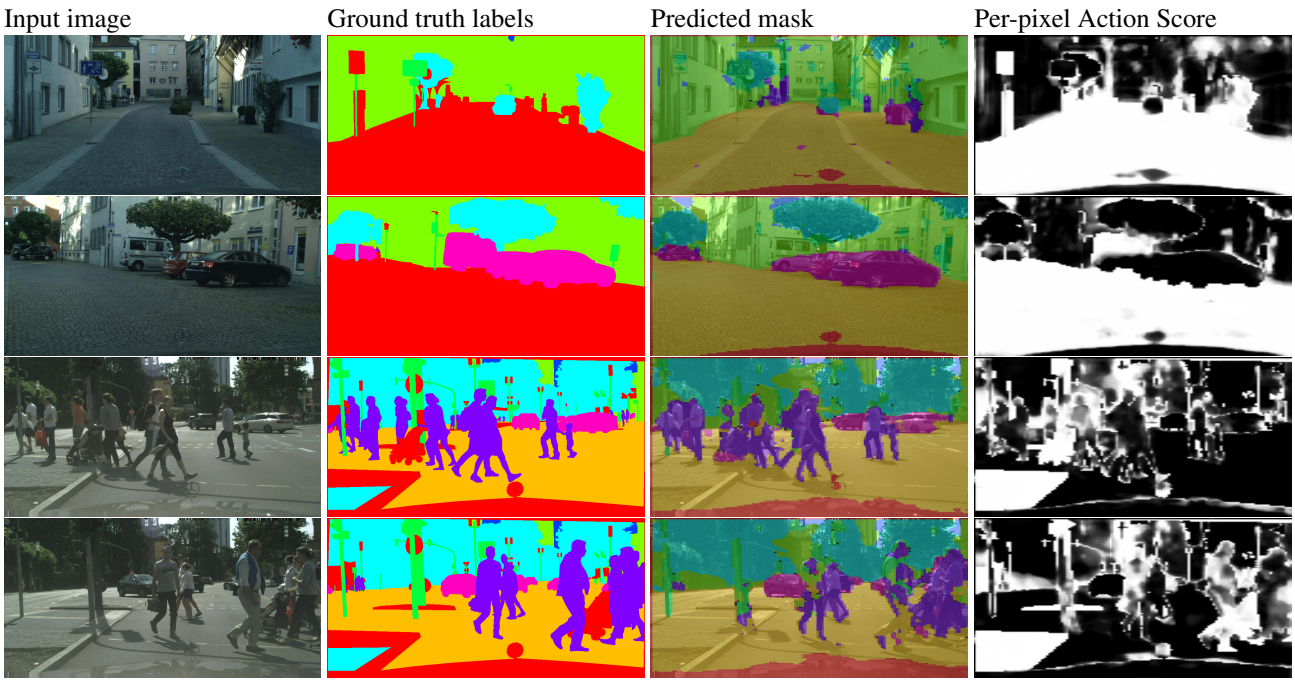


Figure 11. Comparison of action score in segmentation on the Cityscapes dataset, showing the most difficult images. The first column shows the input images, the second column the image labels, the third the predicted masks, and the last column shows our per-pixel action score. Note similar looking regions are the most difficult to classify (road, sidewalk, and person).

dataset? Intuitively one might think that most sources of biases come from the data (since data is selected and captured by a human), but we can use the action scores of different models obtained from the same dataset to approach this question. For this experiment, we compare the correlation of action scores on the CIFAR10 test set, across multiple models. As baseline we compare against correlation of action scores on multiple training runs of the same model. Results are presented in Table 1 and Figure 14.

These results indicate that correlation of action scores across models is lower than the same model over multiple runs, which we believe shows evidence that each model has a uniquely different inductive bias, which leads to producing different action scores and a variety of difficulty rankings between easiest and most difficult inputs. The supplementary material contains additional results on Fashion

Models vs Class	Overall	Airplane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
CNN-Keras vs ResnetV2	0.645	0.581	0.542	0.555	0.634	0.635	0.650	0.730	0.670	0.629	0.689
CNN-Keras vs Xception-Mini	0.607	0.604	0.542	0.565	0.547	0.538	0.622	0.668	0.643	0.595	0.630
Xception-Mini vs ResNetV2	0.711	0.689	0.702	0.686	0.671	0.646	0.722	0.742	0.656	0.764	0.775
CNN-Keras vs CNN-Keras	0.870	0.886	0.819	0.844	0.850	0.874	0.885	0.873	0.881	0.866	0.851
ResnetV2 vs ResNetV2	0.830	0.823	0.838	0.806	0.800	0.794	0.824	0.840	0.768	0.878	0.870
Xception-Mini vs Xception-Mini	0.796	0.775	0.798	0.769	0.754	0.787	0.759	0.825	0.806	0.842	0.855

Table 1. Comparison of correlation between action scores for pairs of models, separated by class, on CIFAR10. The bottom rows of this table show baseline correlations between two runs of the same model.

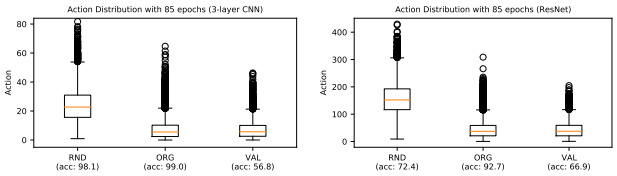


Figure 12. Distribution of action score for training samples with random labels, ground-truth and for validation samples. Distribution for a 3-layer CNN (left) and ResNet (right).

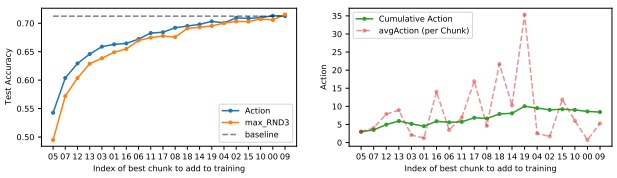
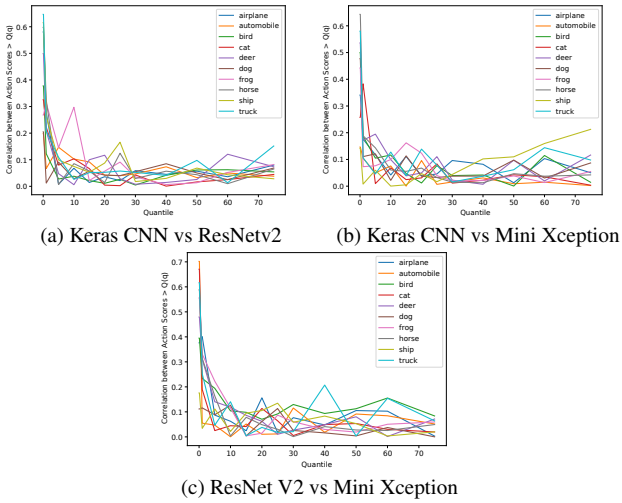


Figure 13. Training with samples that maximize accuracy. Left: sorting the dataset by action score consistently yields better curricula than an equally sized random sample. Right: mean action score as the size of the dataset increases. The average action score increases as more data becomes available.



(a) Keras CNN vs ResNetv2 (b) Keras CNN vs Mini Xception

(c) ResNet V2 vs Mini Xception

Figure 14. Correlations as a function of the top action scores, where the threshold is a particular quantile of the data. Our results show that correlation is only relatively high for lower action scores, and higher action scores having low correlation. Results on CIFAR10.

MNIST and FERPlus, which strengthen our conclusions.

5. Conclusions and Future Work

In this work we presented a method for calculating the difficultly and possible biases of a model. Our method doesn't requires external supervision nor a modification of the original model, and it can be easily integrated in any learning framework. We tested our method in multiple different settings that included a wide range of models and tasks. Our results indicate that the maximum and minimum *action scores* do qualitatively correspond to difficult or biased samples. Moreover, we explored applications of our metric as a data auditing tool that can systematically identify mislabeled samples and regions of interest.

We believe that our action score can be used by the community to diagnose datasets and models, and to preemptively evaluate their biases. The limitations of our method include the need to train a model to estimate the difficulty on a given set, and that the score could be sensitive to high-level training hyper-parameters such as the optimizer.

Acknowledgements

This work was supported through several grants: German Federal Ministry of Economics and Energy during the Projects KiMMI-SF and PhysWM (Grant 50RA2022, and 50RA2126B), German Federal Ministry of Education and Research through Project SustainML (Grant 101070408), and Carl Zeiss Foundation through Project Sustainable Embedded AI (P2021-02-009).

References

[1] Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10368–10378, 2022. 1, 2

[2] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *ACM International Conference on Multimodal Interaction (ICMI)*, 2016. 3

[3] Nicholas Carlini, Ulfar Erlingsson, and Nicolas Papernot. Distribution density, tails, and outliers in machine learning: Metrics and applications. *arXiv preprint arXiv:1910.13427*, 2019. 1, 2

[4] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018. 3

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 4

[6] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 3

[7] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 4

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1

[9] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. 1

[10] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–319, 2020. 1

[11] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, volume 31, 2018. 2

[12] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018. 2

[13] Jason Jo and Yoshua Bengio. Measuring the tendency of cnns to learn surface statistical... *arXiv preprint arXiv:1711.11561*, 2017. 1

[14] Been Kim, Oluwasanmi Koyejo, Rajiv Khanna, et al. Examples are not enough, learn to criticize! criticism for interpretability. In *NeurIPS*, pages 2280–2288, 2016. 2

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3

[16] Agata Lapedriza, Hamed Pirsiavash, Zoya Bylinskii, and Antonio Torralba. Are all training examples equally valuable? *arXiv preprint arXiv:1311.6510*, 2013. 2, 6

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 4

[18] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019. 10

[19] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled. pages 427–436, 2015. 1

[20] Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, 2020. 2

[21] Kailas Vodrahalli, Ke Li, and Jitendra Malik. Are all training examples created equal? an empirical study. *arXiv preprint arXiv:1811.12569*, 2018. 2

[22] Pei Wang and Nuno Vasconcelos. Towards realistic predictors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 36–51, 2018. 2

[23] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 3

[24] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. 1, 6