EFFICIENT AUGMENTATION VIA DATA SUBSAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Data augmentation is commonly used to encode invariances in learning methods. However, this process is often performed in an inefficient manner, as artificial examples are created by applying a number of transformations to all points in the training set. The resulting explosion of the dataset size can be an issue in terms of storage and training costs, as well as in selecting and tuning the optimal set of transformations to apply. In this work, we demonstrate that it is possible to significantly reduce the number of data points included in data augmentation while realizing the same accuracy and invariance benefits of augmenting the entire dataset. We propose a novel set of subsampling policies, based on model influence and loss, that can achieve a 90% reduction in augmentation set size while maintaining the accuracy gains of standard data augmentation.

1 INTRODUCTION

Data augmentation is a process in which the training set is expanded by applying class-preserving transformations, such as rotations or crops for images, to the original data points. This process has become an instrumental tool in achieving state-of-the-art accuracy in modern machine learning pipelines. Indeed, for problems in image recognition, data augmentation is a key component in achieving nearly all state-of-the-art results (Cireşan et al., 2010; Dosovitskiy et al., 2016; Graham, 2014; Sajjadi et al., 2016). Data augmentation is also a popular technique because of its simplicity, particularly in deep learning applications, where applying a set of known invariances to the data is often more straightforward than trying to encode this knowledge directly in the model architecture.

However, data augmentation can be an expensive process, as applying a number of transformations to the entire dataset may increase the overall size of the dataset by orders of magnitude. For example, if applying just 3 sets of augmentations (e.g., translate, rotate, crop), each with 4 possible configurations, the dataset can easily grow by a factor of 12 (if applied independently), all the way to 64x (if applied in sequence). While this may have some benefits in terms of overfitting, augmenting the entire training set can also significantly increase data storage costs and training time, which can scale linearly or superlinearly with respect to the training set size. Further, selecting the optimal set of transformations to apply to a given data point is often a non-trivial task. Indeed, applying transformations not only takes processing time, but also frequently requires some amount of domain expertise. Augmentations are often applied heuristically in practice, and small perturbations are expected (but not proven) to preserve classes. If more complex augmentations are applied to a dataset, they may have to be verified on a per-sample basis.

In this work, we aim to make data augmentation more efficient and user-friendly by identifying subsamples of the full dataset that are good candidates for augmentation. In developing policies for subsampling the data, we draw inspiration from the virtual support vector (VSV) method, which has been used for this purpose in the context of SVMs (Burges & Schölkopf, 1997; Decoste & Schölkopf, 2002). The VSV method attempts to create a more robust decision surface by augmenting only the samples that are close to the margin—i.e., the support vectors. The motivation is intuitive: if a point does not affect the margin, then any small perturbation of that point in data space will likely yield a point that is again too far from the margin to affect it. The method proceeds by applying class-preserving data augmentations (e.g., small perturbations) to all support vectors in the training set. The SVM is then retrained on the support vector dataset concatenated with the augmented dataset, and the end result is a decision surface that has been encoded with transformation invariance while augmenting many fewer samples than found in the full training set.

Although proven to be an effective approach for SVMs, methods utilizing support vectors typically do not generalize well to other classifiers. Therefore, in this work, we aim to develop policies that can effectively reduce the augmentation set size while applying to a much broader class of models. A key step in developing these policies is to determine some metric by which to rank the importance of data points for augmentation. We build policies based on two key metrics. First, we make a natural generalization of the VSV method by measuring the loss induced by a training point. Second, we explore using the *influence* of a point as an indicator of augmentation potential. Influence functions, originating from robust statistics, utilize more information than loss (i.e., residuals) alone, as they take into account both leverage and residual information.

The contributions of this paper are as follows. First, we demonstrate that it is typically unnecessary to augment the entire dataset to achieve high accuracy—for example, we can maintain 99.8% or more of the full augmentation accuracy while only augmenting 10% of the dataset in the case of translation augmentations, and we observe similar behavior for other augmentations. Second, we propose several policies to select the subset of points to augment. Our results indicate that policies based off of training loss or model influence are an effective strategy over simple baselines, such as random sampling. Finally, we propose several modifications to these approaches, such as sample reweighting and online learning, that can further improve performance. Our proposed policies are simple and straightforward to implement, requiring only a few lines of code. Throughout, our experiments are performed on common benchmark datasets, such as MNIST, CIFAR10, and NORB.

2 Related Work

In the domain of image classification, most state-of-the-art pipelines use some form of data augmentation (Cireşan et al., 2010; Dosovitskiy et al., 2016; Graham, 2014; Sajjadi et al., 2016). This typically consists of applying crops, flips, or small affine transformations to all the data points in the training set, with parameters drawn randomly from hand-tuned ranges. Beyond image classification, various studies have applied data augmentation techniques to modalities such as audio (Uhlich et al., 2017) and text (Lu et al., 2006). The selection of these augmentation strategies can have large performance impacts, and thus can require extensive selection and tuning (Ratner et al., 2017).

Motivated by the ubiquity of data augmentation and the difficulty in selecting augmentations, there has been a significant amount of work in selecting and tuning the best *transformations* to use when performing augmentation. For example, Fawzi et al. (2016) use adaptive data augmentation to choose transformations that maximize loss for the classifier; Ratner et al. (2017) propose learning a sequence model of composed transformations; and Cubuk et al. (2018) suggest a reinforcement learning approach. In contrast to the above works, our aim in this work is instead to select which *data points* to augment while holding transformations fixed. We note that our subsampling policies are therefore complementary to many of the described approaches, and in fact could be quite beneficial for approaches such as reinforcement learning that can quickly become infeasible for large datasets and transformation spaces. Finally, we note that several recent works have proposed augmentation strategies based on adversarial training approaches, such as robust optimization frameworks or generative adversarial networks (GANs) (Goodfellow et al., 2014; Antoniou et al., 2017; Volpi et al., 2018). These approaches generate artificial points from some target distribution, rather than by directly transforming the original training points. We view these works as orthogonal and complementary approaches to the proposed work, which is designed in concert with more traditional data augmentation strategies.

The area of work most closely related to our own is that of the Virtual Support Vector (VSV) method (Burges & Schölkopf, 1997; Decoste & Schölkopf, 2002). This method was proposed in the context of support vector machines as a way to reduce the set of candidate points for augmentation by limiting transformations to only support vectors. In the context of SVMs, the motivation is straightforward, as points that are far from the margin are unlikely to affect future models if they are transformed via small perturbations. However, to the best of our knowledge, there has been no work extending these ideas to methods beyond SVMs, where the notion of support vectors no longer applies.

Inspired by the VSV work, we similarly seek ways to downsample the set of candidate points for augmentation, though through metrics beyond support vectors. One such metric is that of model influence, which has been rigorously studied in the field of robust statistics as a way to determine

which data points are most impactful on the model. Model influence has been studied extensively in the regression literature (Hoaglin & Welsch, 1978; Pregibon et al., 1981; Cook, 1986; Walker & Birch, 1988), and more recently, in non-differentiable (SVMs) and non-convex (deep networks) settings (Koh & Liang, 2017). We also explore policies based off of simpler notions, such as loss; indeed, one of our proposed policies (based on loss) results in the original VSV method as a special case, when the number of augmented points is fixed to be exactly the number of support vectors. We discuss additional details on these metrics and the resulting policies in Section 4.

Finally, we note that this work is closely related to work in data subsampling methods for general dataset reduction (i.e., not in the context of data augmentation). For example, work using gradients (Zhu, 2016), leverage (Drineas et al., 2011), and influence functions (Ting & Brochu, 2017; Wang et al., 2018) have shown better results than uniform sampling of data samples in the original dataset. Our scenario differs from the subsampling scenarios in these works as we anticipate ultimately increasing the size of the dataset through augmentation, rather than decreasing it as is the case with subsampling. Indeed, subsampling methods are motivated by being unable to train models on entire datasets due to the datasets being too large. Our motivation is instead that the full *augmented dataset* may be too large, but the original training set is sufficiently small to be handled without special consideration. We therefore assume it is possible to obtain exact fitting information (e.g., influence, loss, etc.) by fitting a model to the original data. Further, the interpretation of our scenario differs, as the subsampling is performed with the ultimate aim being to retain the accuracy of the some yet-to-be-determined fully augmented dataset, as opposed to the original dataset.

3 Motivation: On the Effectiveness of Subsampling

In this work, we seek to make data augmentation more efficient by providing effective policies for subsampling the original training dataset. To motivate the effect of subsampling prior to augmentation, we begin with a simple example. In Table 1, we report the effect that performing translation augmentations has on the final test accuracy for several datasets (MNIST, CIFAR10, NORB). In the second column, we provide the final test accuracy assuming *none* of the training data points are augmented, and in the last column, the final test accuracy after augmenting *all* of the training data points (i.e., our desired test accuracy). Note that the test dataset in these examples has also been augmented with translation so as to highlight the effect of augmentation; we provide full experimental details in Section 5. In columns 3-8, we report test accuracies from augmenting 5, 10, and 25 percent of the data, where these subsamples are either derived using simple random sampling, or via our proposed policies (to be discussed in Section 4).

An immediate take-away from these results is that, even in the case of simple random sampling, it is clear that it is often unnecessary to augment the entire dataset to achieve decent accuracy gains. For example, augmenting just 25% of the dataset selected at random can yield more than half of the total accuracy gain from full augmentation. However, it is also evident that subsampling can be done more effectively with the appropriate policy. Indeed, as compared to random sampling, when augmenting just 10% of the data, these optimal policies can achieve almost identical results to full augmentation (within .1-.2% for MNIST and CIFAR, with even higher accuracy for NORB). These results aim to serve as starting point for the remaining paper: We describe our proposed policies in detail in Section 4, and provide full experiments and experimental details in Section 5.

Dataset	No Aug.	Baseline Random Policy			Best Policy			Full Aug.
	0%	5%	10%	25%	5%	10%	25%	100%
MNIST	85.4%	96.8%	98.1%	98.9%	98.6%	99.1%	99.4%	99.3%
CIFAR10	96.3%	96.5%	96.7%	97.0%	97.0%	97.2%	97.3%	97.3%
NORB	87.3%	88.0%	88.1%	88.8%	90.2%	90.0%	89.8%	89.7%

Table 1: Best observed policy vs. expected baseline with translate augmentations for various percentages of the training set being augmented. The best policies are capable of reaching near full augmentation performance with a small augmentation budget.

4 AUGMENTATION SET SELECTION POLICIES

In this section, we provide details on our augmentation policies, including their general structure (described below), the metrics they utilize (Section 4.1), and improvements such as reweighting or online learning (Section 4.2).

Setup. The aim in this work is to find some subset $\mathcal{S} := \{(x_i, y_i), \ldots (x_j, y_j)\}$ of the full training set $\mathcal{D} := \{(x_1, y_1), \ldots (x_n, y_n)\}$, such that augmenting only the subset $\mathcal{S}$ results in similar performance to augmenting the entire dataset $\mathcal{D}$. More precisely, the goal is to minimize the size of $\mathcal{S}$, $|\mathcal{S}|$, subject to the constraint that perf($\mathcal{S}_{aug}$) $\approx$ perf($\mathcal{D}_{aug}$), where $\mathcal{S}_{aug}$ and $\mathcal{D}_{aug}$ represent the dataset after appending augmented examples generated from the original examples in $\mathcal{S}$ or $\mathcal{D}$, respectively. We note that while the performance measure perf($\cdot$) may be broadly construed, we focus in our experiments on measuring specifically performance based on test accuracy.

General Policies. Our proposed policies consist of two parts: (i) an *augmentation score* which maps each training point (x_i, y_i) to a value $s \in \mathbb{R}$, and (ii) a *policy* by which to sample points based on these augmentation scores. We describe two metrics by which augmentation scores are generated, including loss and model influence, in Section 4.1. In terms of policies for subset selection based on these scores, we first explore two simple policies—deterministic and random. In particular, given a set of augmentation scores $\{s_1, \ldots, s_n\}$ for the n training points, we select a subset $\mathcal{S} \subseteq \mathcal{D}$ either by ordering the points based on their scores and taking the top k values (in a deterministic fashion), or by converting each augmentation score s_i to a probability $\pi_i(z) \in [0, 1]$, and then sampling according to this distribution without replacement. As the augmentation scores (and resulting policies) may be affected by updates to the model after each augmentation, we additionally explore in Section 4.2 the effect of iteratively updating or re-weighting scores to adjust for shifts in the underlying model. A non-exhaustive overview of the various augmentation policies is provided in Table 2.

Policy Name	Selection Function	Update Scores	Downweight Points
Baseline	$P(z_i) = \frac{1}{n}$	X	X
Random	$P(z_i) = \frac{s_i}{\sum_j s_j}$	X	X
Deterministic	Rank$(z_i) = \texttt{SELECT}_S^{-1}(s_i)$	X	X
Update Random	$P(z_i) = \frac{s_i}{\sum_j s_j}$	✓	X
Downweight Random	$P(z_i) = \frac{s_i}{\sum_j s_j}$	X	✓

Table 2: Overview of the augmentation policies and their parameters, where s_i is the augmentation score given to point $z_i = (x_i, y_i)$. The $\texttt{SELECT}_S^{-1}$ function corresponds to the inverse of an order statistic function. As a baseline, we compare to sampling data points at random, ignoring the augmentation scores. Note that the notation here is simplified to allow sampling with replacement, though in practice we perform sampling without replacement.

4.1 METRICS: LOSS AND INFLUENCE

We propose two metrics to determine our augmentation scores: training loss and model influence.

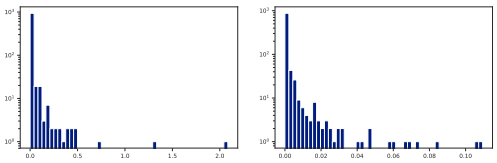
Training loss. One method to obtain augmentation scores is the loss at a point in the training set. This can be viewed as a more direct generalization of the virtual support vector (VSV) method, as support vectors are points with non-zero loss. However, studying loss directly will allow us: (i) to extend to methods beyond SVMs, and (ii) to expand the augmented set to data points beyond just the fixed set of support vectors.

Influence. We also explore policies based on Leave-One-Out (LOO) influence, which measures the influence that a training data point has against its own loss when it is removed from the training set. We follow the notation used in Koh & Liang (2017). Let $\hat{\theta}$ be the minimizer of the loss, which is assumed to be twice differentiable and strictly convex in θ. We define the influence of upweighting a point, z, on the loss at a test point, z_{test}, as $\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) := -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z_{\text{test}}, \hat{\theta})$. It follows that if the test point is z, then the LOO influence can be calculated as:

$$\mathcal{I}_{\text{LOO}}(z) := \mathcal{I}_{\text{up,loss}}(z, z) = -\nabla_\theta L(z, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \,. \tag{1}$$

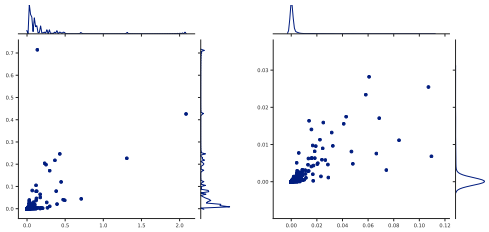
For our augmentation scores, we care only about the magnitude of the LOO influence, so it can be assumed that the sign is dropped.

To understand the potential of using training loss and model influence for scoring, we provide a histogram of model influence across the CIFAR and NORB datasets in Figure 1. Full results for all datasets and for training loss are provided in Appendix A. In Figure 1, we see that while most of the mass is centered around 0 (which we utilize to avoid points), there is sufficient variability to allow for ranking points by preference. Further, as seen in Figure 2, these values are correlated before and after augmentation, indicating that these metrics are a reliable measure of the future impact of a data point after augmentation. In terms of quantifying correlation, we observe spearman correlations between 0.5 and 0.97 with p-values less than 0.001 (and usually orders of magnitude less).



(a) CIFAR10 (b) NORB

Figure 1: Distribution of influence function values on initial training set for translate augmentations. Most values are not influential and can therefore be augmented with low priority. We find similar results when measuring training loss (Appendix A).



(a) CIFAR10 (b) NORB

Figure 2: Distribution of influence values on initial training set (x-axis) vs. final training set (y-axis) for translate augmentations. Points which that are un-influential typically remain un-influential.

4.2 Refinements: Sample Reweighting and Score Updating

Reweighting. To motivate reweighting individual samples, consider an augmentation which is the identity map: $f_T : z_i \to \{z_i\}$. Since we add augmentations back to the training set, our augmentation policy will for example duplicate selected samples, resulting in a net effect which reweighs samples with twice the original weight. Using transformations that result in larger augmentation sets will result in larger weights. One approach is post-processing; Fithian & Hastie (2014), for example, show that the case of class imbalanced sampling can be corrected with a shift of the logistic regressor's bias. To normalize for the effect of this implicit reweighing, we divide the weights of the original samples and its augmented samples by the size of that set, $|f_T(z_i)|$. Under this scheme, we guarantee that we conserve the weight originally assigned to a point (and conserve the ratios of labels). More sophisticated polices, such as reweighing samples by a measure of how trustworthy they are (e.g., perhaps some bounds can be derived on the label-preserving properties of an augmentation), remain to be investigated as future work.

We find that in many cases, the performance of reweighting is similar in expectation to the base case, but with lower variance. However, in some cases, reweighting can in fact have a negative impact, as we discuss in Section 5.2. We expect this policy to be more useful in the case of class imbalance, where the duplication of minority class samples may significantly alter the distribution over classes.

Updating scores. Once we decide to augment a data point, we can either continue to use the same influence information which we derived for the un-augmented dataset, or we can choose to update it. The reason for doing this is to account for the drifting model behavior as points are added to the training set and the model is retrained. However, if having a single estimate of influence for the whole lifetime of the model is sufficient, then avoiding repeated influence calculations will reduce the amount of computation required while also enabling an increased level of parallelism (e.g., minibatching, distributed computations). We find that this modification results in similar behavior to that of reweightings, where expected performance of a policy remains similar, but its variance decreases. Overall, we haven't observed a significant enough effect to suggest that this technique is justified given the extra cost it requires. The benefit of this is that it implies that many applications may need to only compute selection metadata once time throughout the augmentation process.

5 EXPERIMENTS

In this section we provide detailed results on the performance of our proposed policies for data subsampling. For all experiments, we use a Convolutional Neural Network (CNN) to create bottleneck features, which we then use as input into a linear logistic regression model. This is equivalent to freezing the weights of the CNN, or using a basis function, $\phi(\cdot)$, to transform the inputs (Nasrabadi, 2007), and allows us to quickly calculate training loss and model influence. We explore the results of our augmentation policies on three datasets: MNIST, CIFAR, and NORB. For MNIST features, we use a LeNet architecture (LeCun et al., 1998) with ReLu activations, and for CIFAR and NORB, we use ResNet50v2 (He et al., 2016). While for CIFAR and NORB we generate the bottleneck features once due to cost, for MNIST, we additionally study the effect of re-generating these features as new points are selected and augmented (i.e., training both the features and model from scratch throughout the experiments).

In terms of augmentations, we consider three examples: translation, rotation, and crop. To control for sources of variance in model performance, all augmentations under consideration are applied exhaustively in a deterministic fashion to any selected samples, and the resulting augmented points are then added back to the training set. Formally, given a data point, $z = (x, y) \in \mathcal{X} \times \mathcal{Y}$, our augmentation is a map from a data point to a finite set of data points: $f_T : z \to \{z_1, \ldots, z_n : z_i \in \mathcal{X} \times \mathcal{Y}\}$. We controlled for augmentation-induced regularization by performing a simple cross validation sweep for the regularization parameter λ each time the model was re-trained, and we found regularization to have negligible impact in the trends we observed. For all datasets and augmentations, we make the effect of augmentation more apparent by adding augmented test points to the test set. For example, in the case of translation, we test the performance of applying translation augmentations to the original training set, and then determine the accuracy using an augmented variant of the test data that has been appended with translated test examples. All augmentations are performed using Imgaug[1], and our code is written in Python using Keras CNN implementations. Our code will be made publicly available online. Full implementation details are provided in Appendix B.

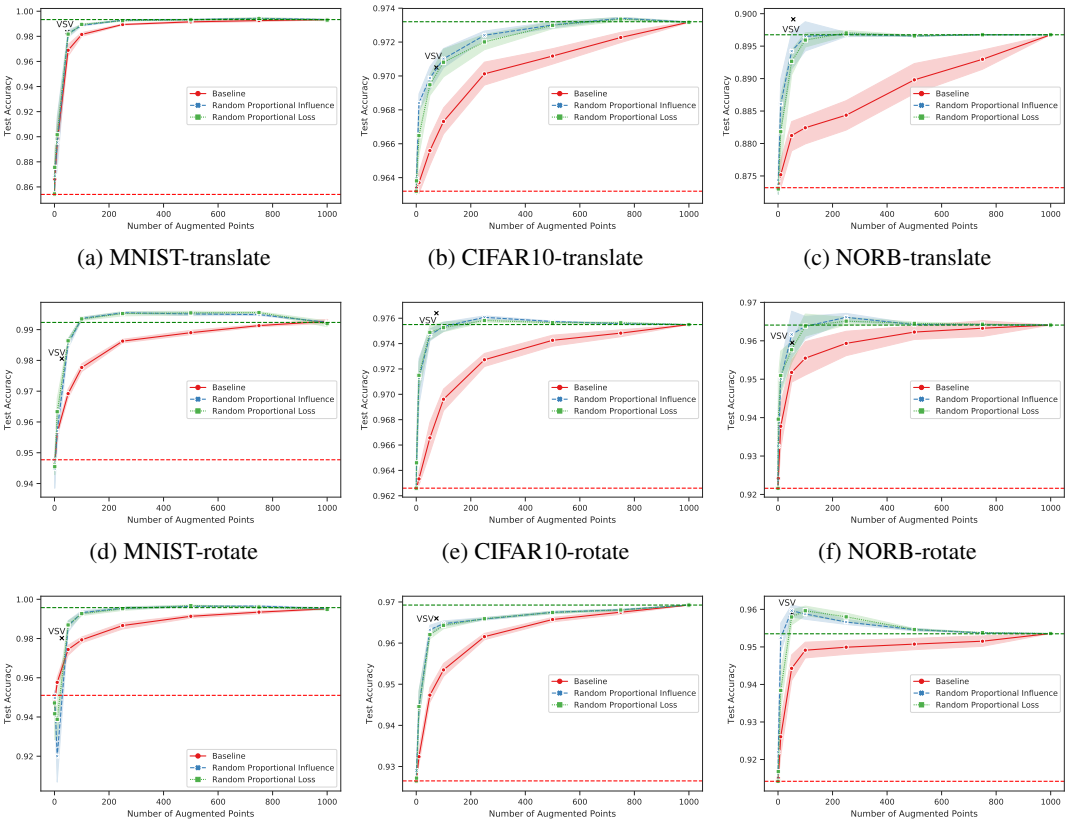
5.1 GENERAL POLICIES: INFLUENCE AND LOSS

In Figure 3, we explore a first set of policies in which we randomly sample points for augmentation proportional either to their loss (green) or influence value (blue). To calculate the loss and influence, we incur a one-time cost of training the model on the original dataset. As a baseline (red), we compare these methods to a simple strategy in which data points for augmentation are drawn entirely at random (irrespective of loss or influence). The red, dotted horizontal line indicates the test accuracy with no augmentation, and the green, dotted line indicates the test accuracy after augmenting the entire training set. Note that all policies have the same accuracy when the number of points is 0 or k, where k is the number of points in the original training set, which correspond to the un-augmented training set and the fully augmented training set, respectively[2]. We observe similar behavior in terms of the deterministic policy, which is provided in Appendix C.

Across the datasets and transformation types, we notice several trends. First, the policies based on loss and influence consistently outperform the random baseline. This is particularly true for the rotation augmentation for all three datasets, where the random-influence and random-loss policies achieve the full augmentation accuracy with only 5-10% of the data, compared to 90-100% of the data for random sampling. Second, we note that the policies based on influence vs. loss behave very similarly. While influence has slightly better performance (particularly on the NORB dataset), the policies are for the most part equivalent. A benefit of this is that the loss calculation is slightly simpler than influence to calculate, as it does not require calculating the inverse Hessian component, $H_{\hat{\theta}}^{-1}$, as described in 1. Third, we note that it is possible to achieve *higher* accuracy than full augmentation using only a reduced set of points for augmentation, as observed in several of the plots (NORB, CIFAR-rotate, MNIST-rotate). We believe that this higher performance may be due to reduced noise in the dataset as compared to full augmentation.

[1] `https://github.com/aleju/imgaug`

[2] In practice, issues with convergence, particularly with the non-convexity of re-trained CNNs, results in solutions which are only approximately the same.



(a) MNIST-translate (b) CIFAR10-translate (c) NORB-translate

(d) MNIST-rotate (e) CIFAR10-rotate (f) NORB-rotate

(g) MNIST-crop (h) CIFAR10-crop (i) NORB-crop

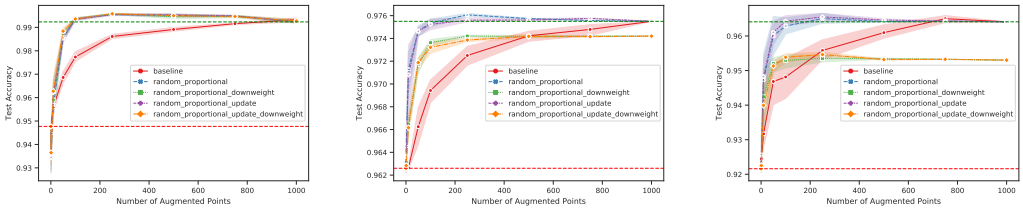
Figure 3: The performance of random policies using influence and loss vs. the baseline (simple random sampling). Random sampling based on loss/influence consistently outperforms the baseline.

Finally, we additionally explore the effect of using support vectors for augmentation, which was proposed in the Virtual Support Vector literature (Burges & Schölkopf, 1997; Decoste & Schölkopf, 2002). In particular, we find VSV points by tuning a linear SVM on the bottleneck features of the original training set, and then using these points as the set of augmentation points for the logistic regression model with bottleneck features. We use search over $C \in \{0.01, 0.1, 1, 10, 100\}$ via cross-validation, and the best resulting model is used to obtain support vectors. Interestingly, we note that, though this transfer approach was not originally proposed in the VSV literature, it results in strong performance on a few of our tests (e.g., NORB-translate, NORB-crop, CIFAR-rotate). However, the approach is not as reliable as the proposed policies in terms of finding the optimal subset of points for transformation (performing significantly below optimal, e.g., for MNIST-crop, MNIST-rotate, and CIFAR-translate), and the major limitation is that the augmentation set size is fixed to exactly equal the number of support vectors, which is more brittle than the proposed policies, which can vary depending on the desired data budget.

5.2 Refinements: Sample Reweighting and Score Updating

We additionally explore the effect of two refinements on the initial policies: reweighting the samples as they are added back to the training set, and updating the scores as the augmentation proceeds, as described in Section 4.2. This latter policy assumes that the method is run in an online fashion, in contrast to the policies described thus far. This add extra expense to the total run time, as the model must be continually updated as new points are augmented. In Figure 4, we observe the effect of these modifications for all datasets using the rotation augmentation, and using model influence as

the score. Full results are provided in Appendix C. Interestingly, while reweighting points seems to have a positive (if negligible) effect for MNIST, we see that it can actually hurt performance in CIFAR and NORB. This may indicate that the amplifying effect of augmentation may in fact be beneficial when training the model. In terms of the score updating, we see that although updating the score can have a slight positive impact (e.g., for NORB-rotate), the performance appears to roughly match that of the original policy. Given the extra expense required in model updating, we therefore conclude that the simpler policies are preferable.



(a) MNIST-rotate (b) CIFAR10-rotate (c) NORB-rotate

Figure 4: The performance of policies when point downweighting is used or augmentation scores are updated.

5.3 Understanding Policies

Finally, to give insight into the behavior of the proposed polices, we examine the 10 points with highest influence/loss vs. least influence/loss, for MNIST. We observe similar results for the other datasets (CIFAR, NORB); additional results are provided in Appendix E. These examples help to visualize the benefits of downsampling, as it is clear that the bottom set of points are all quite similar. The top points, in contrast, appear more diverse—both in terms of class label as well as features (thin lines, thick lines, slanted, straight, etc). We postulate that promoting this diversity and removing redundancy is key in learning invariances through augmentation more efficiently.

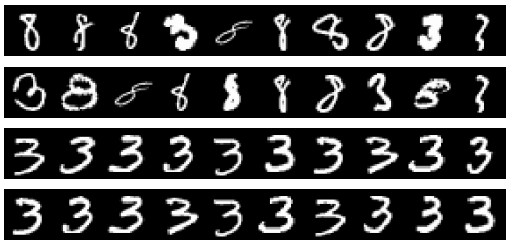


Figure 5: Points with highest influence / loss (top) and lowest influence / loss (bottom).

6 Discussion

In this paper, we have demonstrated that not all training points are equally useful for augmentation, and we have proposed simple policies that can select the most viable subset of points. Our policies, based off of notions of training loss and model influence, are widely applicable to general machine learning models. Obtaining access to an augmentation score vector can be obtained in only one training cycle on the original data (e.g., a fixed cost), yet the potential improvements in augmented training can scale superlinearly with respect to the original dataset size. With many fewer data points to augment, the augmentations themselves can be applied in a more efficient manner in terms of compute and expert oversight. At an extreme, they can be specialized on a per-example basis.

A natural area of future work is to explore subset selection policies that take the entire subset into account, rather than the greedy policies described. For example, even if two samples may independently have large leave-one-out influence, it may be the case that these points influence each other and leave-one-out influence may be an overestimate (e.g., consider the case of two identical samples). Including second-order information or encouraging subset diversity may therefore help to improve performance even further.

REFERENCES

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

Christopher JC Burges and Bernhard Schölkopf. Improving the accuracy and speed of support vector machines. In *Advances in neural information processing systems*, pp. 375–381, 1997.

François Chollet et al. Keras. `https://keras.io`, 2015.

Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, 2010.

R Dennis Cook. Assessment of local influence. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 133–169, 1986.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Dennis Decoste and Bernhard Schölkopf. Training invariant support vector machines. *Machine learning*, 46(1-3):161–190, 2002.

Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1734–1747, 2016.

Petros Drineas, Michael W Mahoney, S Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische mathematik*, 117(2):219–249, 2011.

Alhussein Fawzi, Horst Samulowitz, Deepak Turaga, and Pascal Frossard. Adaptive data augmentation for image classification. In *2016 Ieee International Conference On Image Processing (Icip)*, number EPFL-CONF-218496, pp. 3688–3692. Ieee, 2016.

William Fithian and Trevor Hastie. Local case-control sampling: Efficient subsampling in imbalanced data sets. *Annals of statistics*, 42(5):1693, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Benjamin Graham. Fractional max-pooling. *arXiv:1412.6071*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

David C Hoaglin and Roy E Welsch. The hat matrix in regression and anova. *The American Statistician*, 32(1):17–22, 1978.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Xinghua Lu, Bin Zheng, Atulya Velivelli, and ChengXiang Zhai. Enhancing text categorization with semantic-enriched representation and training data augmentation. *Journal of the American Medical Informatics Association*, 13(5):526–535, 2006.

Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Autograd: Effortless gradients in numpy.

Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16 (4):049901, 2007.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Daryl Pregibon et al. Logistic regression diagnostics. *The Annals of Statistics*, 9(4):705–724, 1981.

Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Neural Information Processing Systems*, 2017.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Neural Information Processing Systems*, 2016.

Daniel Ting and Eric Brochu. Optimal sub-sampling with influence functions. *arXiv preprint arXiv:1709.01716*, 2017.

S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *International Conference on Acoustics, Speech and Signal Processing*, 2017.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *arXiv preprint arXiv:1805.12018*, 2018.

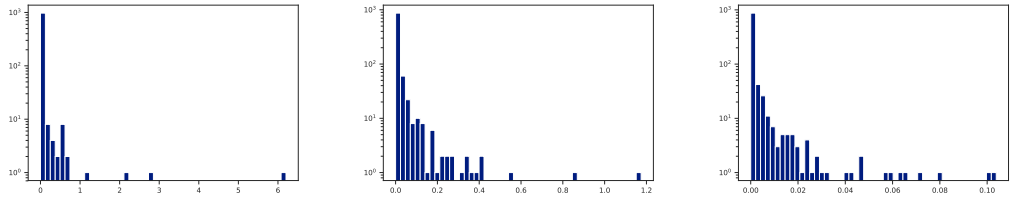
Esteban Walker and Jeffrey B Birch. Influence measures in ridge regression. *Technometrics*, 30(2): 221–227, 1988.

HaiYing Wang, Rong Zhu, and Ping Ma. Optimal subsampling for large sample logistic regression. *Journal of the American Statistical Association*, 113(522):829–844, 2018.

Rong Zhu. Gradient-based sampling: An adaptive importance sampling for least-squares. In *Advances in Neural Information Processing Systems*, pp. 406–414, 2016.

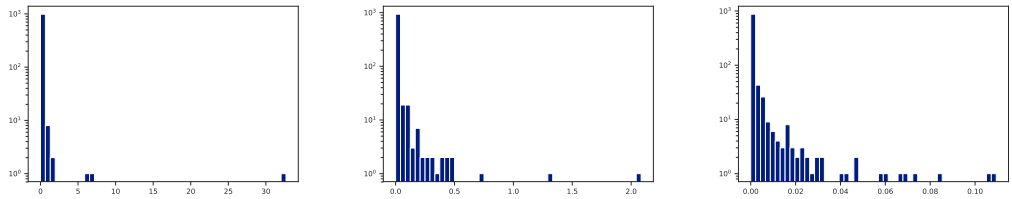
A ADDITIONAL PLOTS: METRICS

Here we provide histogram plots for loss and influence for all datasets. The key take-away from these results is that the distribution of these metrics indicate that most points have low loss and influence, and thus (according to our policies) can be augmented with low probability.



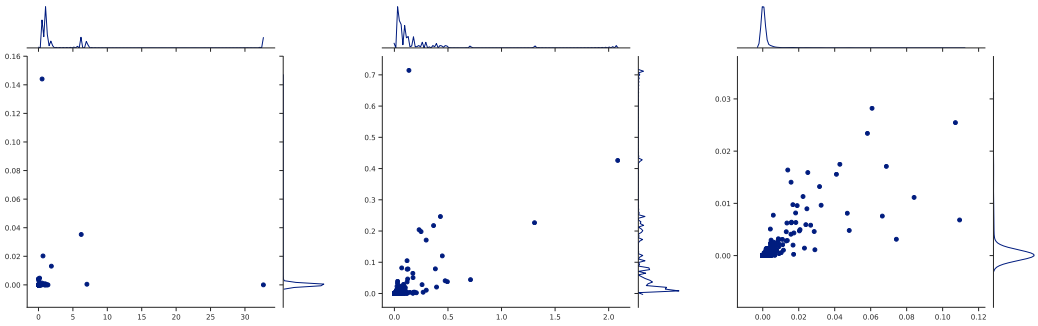
(a) MNIST (b) CIFAR10 (c) NORB

Figure 6: Distribution of Log Loss Values on initial training set for translate augmentations. The distributions seem to have similar shape, but with different scales. Most values aren't influencial and can be augmented with low priority.



(a) MNIST (b) CIFAR10 (c) NORB

Figure 7: Distribution of Influence Values on initial training set for translate augmentations. The distributions seem to have similar shape, but with different scales. Most values aren't influencial and can be augmented with low priority.



(a) MNIST (b) CIFAR10 (c) NORB

Figure 8: Distribution of Influence Values on initial training set (x-axis) vs. final training set (y-axis) for translate augmentations. The distributions seem to have similar shape, but with different scales. Most values aren't influencial and can be augmented with low priority. Points which aren't influencial usually stay uninfluencial.

B EXPERIMENT DETAILS

Here we provide full implementation details on our experiments throughout the paper.

Setup. There are a few key architectural ideas in our tests: the data, the augmentations, the selection policy, a featurization preprocessing component, and a logistic regression model. Our implementation is in Python. The dataset is loaded (possibly via third party libraries) into a numpy array. We can then run this dataset through a trained CNN model, such as ResNet50, to obtain a feature vector. The logistic regression model is then trained on this resulting "featurized" dataset and tested on a "featurized" test set. Once training is complete, both loss and influence can then be measured for each training point, and can therefore be used as scores. Augmentations are then applied exhaustively to the test set. We refer to this test set as "poisoned". The test distribution has changed, and therefore a gap has formed between the original test performance and the "poisoned" test performance. We attempt to close this gap by applying augmentations to the training set. We proceed by initializing a set with the unaugmented training set. We augment points in rounds, and the unaugmented training set corresponds to round 0. Every round, our policy is given a vector of scores and it selects a point to augment. This point is featurized and added to the set. The CNN can be optionally retrained, but the logistic regression model must be retrained to obtain the current test accuracy.

Implementation. We perform experiments in Python, using Keras (Chollet et al., 2015), Scikit-Learn (Pedregosa et al., 2011; Buitinck et al., 2013), AutoGrad (Maclaurin et al.), and Imgaug[3]. We wrap Keras implementations of the CNNs in Scikit-Learn transformers, and we create new classes utilizing Scikit-Learn classifiers and their corresponding influence functions calculated with the autograd system. This allows us to decouple input data, bottleneck features, and the final classifier that calculates influence. It also allows us to perform any additional (i.e., cross validation) tuning rather easily. Augmentations are performed by Imgaug. Our code will be made publicly available online.

Models. For all experiments, we use a CNN to create bottleneck features, which we then use as input into a linear logistic regression model. This is equivalent to freezing the weights of the CNN, or using a basis function, $\phi(\cdot)$, to transform the inputs (Nasrabadi, 2007). A LeNet archicture (LeCun et al., 1998) with ReLu activations was used for MNIST; however, this model had issues performing well on the augmented sets for CIFAR10 and NORB. We had also tried a larger model from the Keras examples on MNIST, which resulted in similar performance to using LeNet[4]. Both LeNet and the Keras net were fast to train, so we retrained the models for $40 - 50$ epochs with ADAM (Kingma & Ba, 2014) and a minibatch size of 512, which was enough to obtain convergence. We used a ResNet50v2 model (He et al., 2016) model trained on the CIFAR10 dataset for the CIFAR10 tests, and we obtained good performance without using augmentations in the training process. Using a pretrained ImageNet ResNet50 model resulted in poor performance (both computationally and in accuracy). For NORB, we were able to get good performance on the translate task without any training augmentations being applied on the NORB dataset. However, the other augmentations resulted in high prediction degradation, so the ResNet model was retrained with random rotations, shifts, and flips applied to images. All ResNet models were frozen after the initial training.

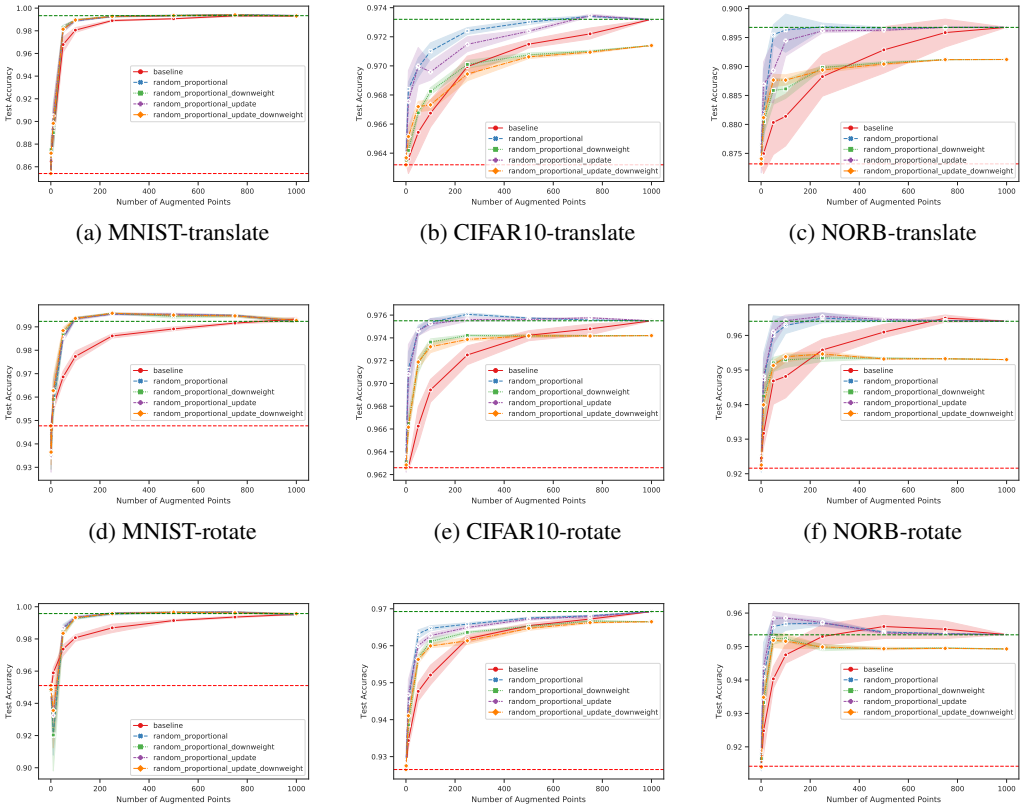
Augmentations. Our tests use translate, rotate, and crop. Each of these augmentations is applied over a range of parameters, which results in multiple augmented images. Translate is applied for 2 pixels in all cardinal directions (e.g., up, down, left, and right) on MNIST and CIFAR10, and 6 pixels for NORB (note: this pixel difference is to account for NORB images being 3 times larger than CIFAR10). Rotate is applied between for the 15 (14 after removing identity transform) rotations evenly spaced between $\pm 30°$. Crop is applied excluding the outer $[1, 2, \ldots, 6]$ pixels on all 4 image sides, and zoom is applied to rescale the resulting image back to its original dimensions. Usually, augmentations are constructed to preserve labels, but it is possible in principle to construct augmentations that utilize label information for the augmentation itself or perhaps induce a change in label (e.g., an image dataset with segmentation information can segment out all non-background classes to change the label of an image to background). Such augmentations are expensive, require domain expertise, and are hard to validate, but they may be viable if the number of total augmentations can be controlled.

[3] https://github.com/aleju/imgaug

[4] https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

C ADDITIONAL PLOTS: POLICIES

Below we provide full experiments for the randomized (Figure 9) and deterministic (Figure 10) policies using model influence as the scoring metric, across all datasets and transformations.

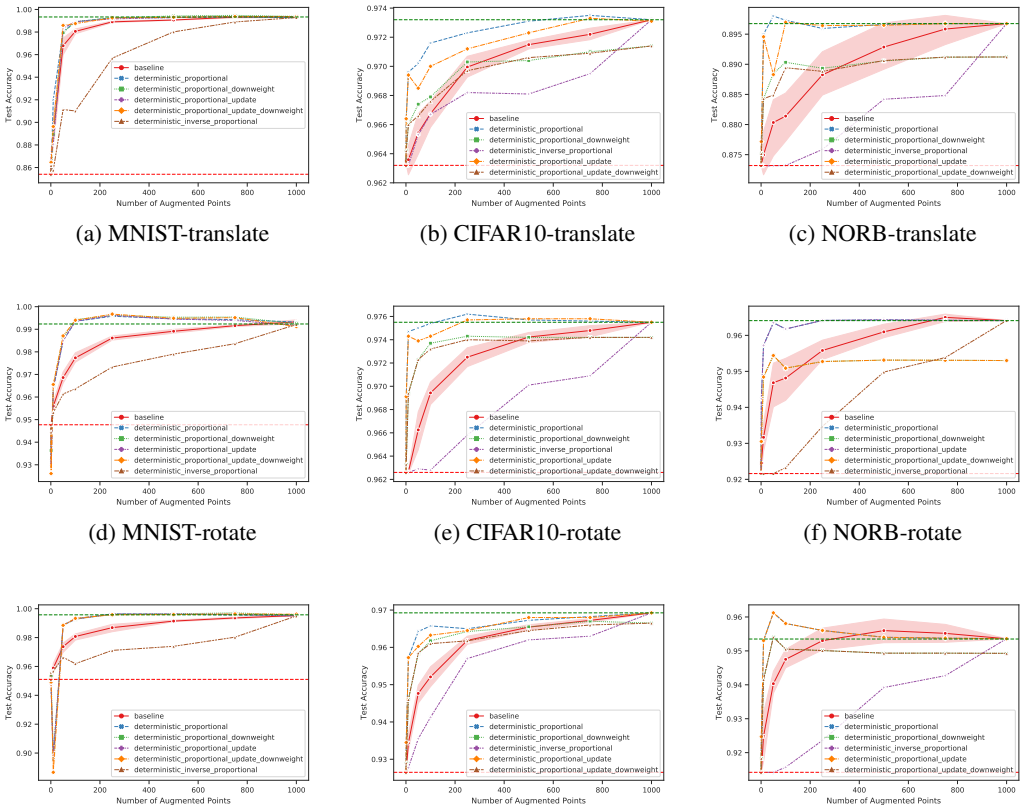


(a) MNIST-translate (b) CIFAR10-translate (c) NORB-translate

(d) MNIST-rotate (e) CIFAR10-rotate (f) NORB-rotate

(g) MNIST-crop (h) CIFAR10-crop (i) NORB-crop

Figure 9: The performance of randomized policies using influence.



(a) MNIST-translate (b) CIFAR10-translate (c) NORB-translate

(d) MNIST-rotate (e) CIFAR10-rotate (f) NORB-rotate

(g) MNIST-crop (h) CIFAR10-crop (i) NORB-crop

Figure 10: The performance of deterministic policies using influence.

D FULL RESULTS: POLICIES

We provide below AUC results from the plots provided in Appendix C. The AUC provides a more clear metric by which to determine the best performing policy. We separate tables across augmentation (translate/rotate/crop) and score (loss/influence).

D.1 AUC RESULTS USING LOSS

Policy	AUC Mean	AUC Std.
deterministic_proportional_update_downweight	990.688	—
deterministic_proportional_update	990.242	—
deterministic_proportional	990.100	—
random_proportional_update_downweight	989.555	0.403
random_proportional	989.489	0.504
random_proportional_downweight	989.401	0.481
random_proportional_update	989.146	0.357
deterministic_proportional_downweight	989.132	—
baseline	986.959	0.734
random_inverse_proportional	971.230	1.014
deterministic_inverse_proportional	967.365	—

Table 3: The statistics for Area Under the Curve (AUC) for MNIST Translate Loss

Policy	AUC Mean	AUC Std.
deterministic_proportional	972.568	—
random_proportional	972.320	0.231
deterministic_proportional_update	971.979	—
random_proportional_update	971.897	0.207
baseline	970.662	0.967
random_proportional_downweight	970.209	0.107
deterministic_proportional_downweight	970.163	—
deterministic_proportional_update_downweight	969.932	—
random_proportional_update_downweight	969.596	0.220
random_inverse_proportional	969.328	0.353
deterministic_inverse_proportional	968.790	—

Table 4: The statistics for Area Under the Curve (AUC) for CIFAR Translate Loss

Policy	AUC Mean	AUC Std.
deterministic_proportional	896.521	—
deterministic_proportional_update	896.521	—
random_proportional	896.099	0.283
random_proportional_update	895.909	0.647
deterministic_proportional_downweight	890.211	—
deterministic_proportional_update_downweight	890.211	—
baseline	889.826	2.438
random_proportional_update_downweight	889.739	0.225
random_proportional_downweight	889.620	0.268
deterministic_inverse_proportional	882.325	—
random_inverse_proportional	882.051	1.205

Table 5: The statistics for Area Under the Curve (AUC) for NORB Translate Loss

Policy	AUC Mean	AUC Std.
deterministic_proportional_update	993.330	—
deterministic_proportional_downweight	993.150	—
random_proportional_downweight	993.141	0.129
deterministic_proportional_update_downweight	993.118	—
random_proportional_update_downweight	993.078	0.186
random_proportional_update	993.005	0.321
random_proportional	992.991	0.473
deterministic_proportional	992.493	—
baseline	987.700	0.777
random_inverse_proportional	978.072	0.273
deterministic_inverse_proportional	977.860	—

Table 6: The statistics for Area Under the Curve (AUC) for MNIST Rotate Loss

Policy	AUC Mean	AUC Std.
deterministic_proportional	975.654	—
random_proportional	975.426	0.111
deterministic_proportional_update	975.368	—
random_proportional_update	975.280	0.158
deterministic_proportional_downweight	973.817	—
deterministic_proportional_update_downweight	973.707	—
random_proportional_downweight	973.652	0.138
random_proportional_update_downweight	973.651	0.054
baseline	973.216	0.630
random_inverse_proportional	969.506	0.308
deterministic_inverse_proportional	969.041	—

Table 7: The statistics for Area Under the Curve (AUC) for CIFAR Rotate Loss

Policy	AUC Mean	AUC Std.
deterministic_proportional	963.622	—
deterministic_proportional_update	963.622	—
random_proportional_update	963.401	0.886
random_proportional	963.366	0.317
baseline	959.825	3.569
random_proportional_update_downweight	952.807	0.433
random_proportional_downweight	952.697	0.566
deterministic_proportional_downweight	952.623	—
deterministic_proportional_update_downweight	952.623	—
random_inverse_proportional	947.599	1.885
deterministic_inverse_proportional	944.814	—

Table 8: The statistics for Area Under the Curve (AUC) for NORB Rotate Loss

Policy	AUC Mean	AUC Std.
random_proportional_update_downweight	993.684	0.794
random_proportional_update	993.641	0.641
deterministic_proportional	993.600	—
deterministic_proportional_downweight	993.557	—
random_proportional	993.478	0.388
random_proportional_downweight	992.949	0.982
deterministic_proportional_update_downweight	992.932	—
deterministic_proportional_update	992.588	—
baseline	987.750	1.523
random_inverse_proportional	976.081	0.947
deterministic_inverse_proportional	974.450	—

Table 9: The statistics for Area Under the Curve (AUC) for MNIST Crop Loss

Policy	AUC Mean	AUC Std.
deterministic_proportional	966.573	—
random_proportional	966.182	0.161
deterministic_proportional_update	966.083	—
random_proportional_update	965.162	0.467
deterministic_proportional_downweight	964.257	—
random_proportional_downweight	963.524	0.432
deterministic_proportional_update_downweight	963.057	—
baseline	962.867	0.763
random_proportional_update_downweight	962.507	0.872
random_inverse_proportional	958.632	0.474
deterministic_inverse_proportional	958.132	—

Table 10: The statistics for Area Under the Curve (AUC) for CIFAR Crop Loss

Policy	AUC Mean	AUC Std.
deterministic_proportional	954.812	—
deterministic_proportional_update	954.812	—
random_proportional	954.701	0.252
random_proportional_update	954.564	0.472
baseline	950.662	2.089
random_proportional_downweight	949.632	0.669
random_proportional_update_downweight	949.505	0.459
deterministic_proportional_downweight	949.446	—
deterministic_proportional_update_downweight	949.446	—
random_inverse_proportional	937.300	1.994
deterministic_inverse_proportional	934.525	—

Table 11: The statistics for Area Under the Curve (AUC) for NORB Crop Loss

D.2 AUC RESULTS USING INFLUENCE

Policy	AUC Mean	AUC Std.
random_proportional_update_downweight	995.372	0.218
deterministic_proportional_update_downweight	995.284	—
random_proportional_update	995.240	0.145
deterministic_proportional_update	995.063	—
deterministic_proportional	989.474	—
deterministic_proportional_downweight	989.467	—
random_proportional	989.447	0.739
random_proportional_downweight	989.391	0.772
baseline	986.093	0.806
random_inverse_proportional	969.321	1.457
deterministic_inverse_proportional	965.509	—

Table 12: The statistics for Area Under the Curve (AUC) for MNIST Translate

Policy	AUC Mean	AUC Std.
deterministic_proportional	972.648	—
random_proportional	972.503	0.233
random_proportional_update	972.016	0.195
deterministic_proportional_update	971.924	—
baseline	970.589	0.398
random_proportional_downweight	970.139	0.092
deterministic_proportional_downweight	970.128	—
deterministic_proportional_update_downweight	969.967	—
random_proportional_update_downweight	969.905	0.084
random_inverse_proportional	968.924	0.171
deterministic_inverse_proportional	968.695	—

Table 13: The statistics for Area Under the Curve (AUC) for CIFAR Translate

Policy	AUC Mean	AUC Std.
deterministic_proportional	896.517	—
random_proportional	896.160	0.602
deterministic_proportional_update	896.095	—
random_proportional_update	895.573	0.702
baseline	890.917	2.922
deterministic_proportional_downweight	890.194	—
deterministic_proportional_update_downweight	889.860	—
random_proportional_update_downweight	889.783	0.196
random_proportional_downweight	889.678	0.500
deterministic_inverse_proportional	882.327	—
random_inverse_proportional	881.782	0.672

Table 14: The statistics for Area Under the Curve (AUC) for NORB Translate

Policy	AUC Mean	AUC Std.
deterministic_proportional_downweight	993.580	—
deterministic_proportional_update_downweight	993.314	—
random_proportional_update_downweight	993.251	0.360
random_proportional_update	993.124	0.317
random_proportional	992.989	0.101
random_proportional_downweight	992.959	0.682
deterministic_proportional	992.943	—
deterministic_proportional_update	992.708	—
baseline	986.517	0.472
random_inverse_proportional	977.582	0.321
deterministic_inverse_proportional	977.491	—

Table 15: The statistics for Area Under the Curve (AUC) for MNIST Rotate

Policy	AUC Mean	AUC Std.
deterministic_proportional	975.624	—
random_proportional	975.474	0.151
deterministic_proportional_update	975.430	—
random_proportional_update	975.369	0.131
deterministic_proportional_downweight	973.907	—
random_proportional_downweight	973.775	0.078
deterministic_proportional_update_downweight	973.722	—
random_proportional_update_downweight	973.655	0.043
baseline	972.996	0.812
random_inverse_proportional	969.198	0.124
deterministic_inverse_proportional	968.836	—

Table 16: The statistics for Area Under the Curve (AUC) for CIFAR Rotate

Policy	AUC Mean	AUC Std.
random_proportional_update	963.841	0.327
deterministic_proportional	963.632	—
deterministic_proportional_update	963.632	—
random_proportional	963.402	0.703
baseline	958.495	1.508
random_proportional_update_downweight	952.957	0.485
random_proportional_downweight	952.807	0.473
deterministic_proportional_downweight	952.613	—
deterministic_proportional_update_downweight	952.613	—
random_inverse_proportional	946.769	0.855
deterministic_inverse_proportional	944.818	—

Table 17: The statistics for Area Under the Curve (AUC) for NORB Rotate

Policy	AUC Mean	AUC Std.
random_proportional_update	993.712	0.320
random_proportional_update_downweight	993.512	0.710
random_proportional_downweight	993.298	0.809
random_proportional	993.213	0.559
deterministic_proportional_downweight	993.098	—
deterministic_proportional_update	992.709	—
deterministic_proportional	992.574	—
deterministic_proportional_update_downweight	992.519	—
baseline	988.638	0.479
random_inverse_proportional	976.856	0.819
deterministic_inverse_proportional	975.518	—

Table 18: The statistics for Area Under the Curve (AUC) for MNIST Crop

Policy	AUC Mean	AUC Std.
deterministic_proportional	966.586	—
random_proportional	966.295	0.299
deterministic_proportional_update	966.181	—
random_proportional_update	965.699	0.249
deterministic_proportional_downweight	964.369	—
random_proportional_downweight	963.787	0.307
deterministic_proportional_update_downweight	963.305	—
random_proportional_update_downweight	963.005	0.226
baseline	962.535	1.096
random_inverse_proportional	958.520	0.344
deterministic_inverse_proportional	957.854	—

Table 19: The statistics for Area Under the Curve (AUC) for CIFAR Crop

Policy	AUC Mean	AUC Std.
deterministic_proportional	954.838	—
deterministic_proportional_update	954.838	—
random_proportional_update	954.733	0.381
random_proportional	954.434	0.250
baseline	952.340	2.624
deterministic_proportional_downweight	949.446	—
deterministic_proportional_update_downweight	949.446	—
random_proportional_downweight	949.305	0.416
random_proportional_update_downweight	949.245	0.522
random_inverse_proportional	937.868	1.603
deterministic_inverse_proportional	934.529	—

Table 20: The statistics for Area Under the Curve (AUC) for NORB Crop

E DIAGNOSING INFLUENCE AND LOSS

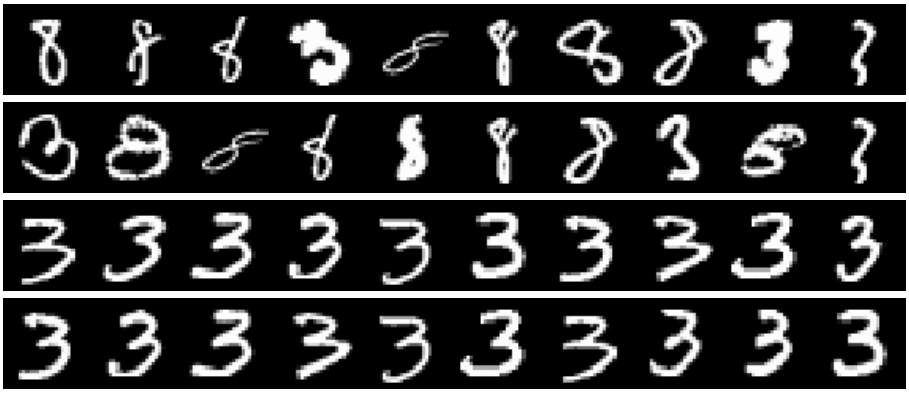


Figure 11: From top to bottom: good influence, good loss, bad influence, bad loss for MNIST.

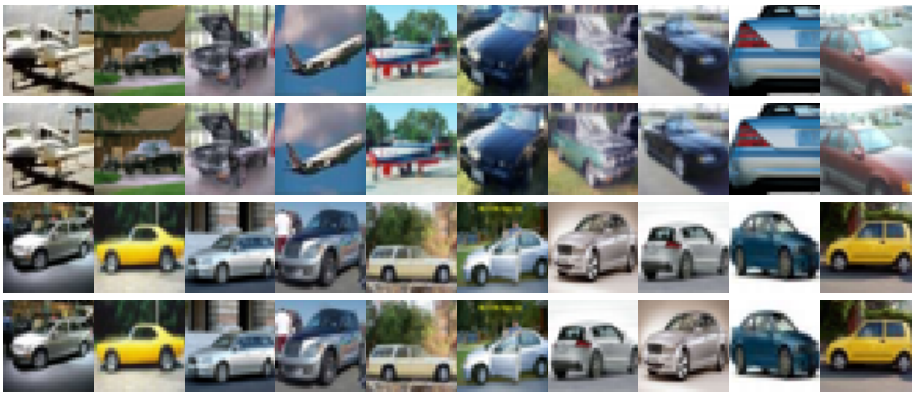


Figure 12: From top to bottom: good influence, good loss, bad influence, bad loss for CIFAR10.

Figure 13: From top to bottom: good influence, good loss, bad influence, bad loss for NORB.