Learning to Compose Domain-Specific Transformations for Data Augmentation

Alexander J. Ratner*, Henry R. Ehrenberg*, Zeshan Hussain,
Jared Dunnmon, Christopher Ré
Stanford University
{ajratner,henryre,zeshanmh,jdunnmon,chrismre}@stanford.edu

September 7, 2017

Abstract

Data augmentation is a ubiquitous technique for increasing the size of labeled training sets by leveraging task-specific data transformations that preserve class labels. While it is often easy for domain experts to specify individual transformations, constructing and tuning the more sophisticated compositions typically needed to achieve state-of-the-art results is a time-consuming manual task in practice. We propose a method for automating this process by learning a generative sequence model over user-specified transformation functions using a generative adversarial approach. Our method can make use of arbitrary, non-deterministic transformation functions, is robust to misspecified user input, and is trained on unlabeled data. The learned transformation model can then be used to perform data augmentation for any end discriminative model. In our experiments, we show the efficacy of our approach on both image and text datasets, achieving improvements of 4.0 accuracy points on CIFAR-10, 1.4 F1 points on the ACE relation extraction task, and 3.4 accuracy points when using domain-specific transformation operations on a medical imaging dataset as compared to standard heuristic augmentation approaches.

1 Introduction

Modern machine learning models, such as deep neural networks, may have billions of free parameters and accordingly require massive labeled data sets for training. In most settings, labeled data is not available in sufficient quantities to avoid overfitting to the training set. The technique of artificially expanding labeled training sets by transforming data points in ways which preserve class labels – known as *data augmentation* – has quickly become a critical and effective tool for combatting this labeled data scarcity problem. Data augmentation can be seen as a form of *weak supervision*, providing a way for practitioners to leverage their knowledge of invariances in a task or domain. And indeed, data augmentation is cited as essential to nearly every state-of-the-art result in image classification [4, 8, 12, 26] (see Appendix A.1), and is becoming increasingly common in other modalities as well [21].

Even on well studied benchmark tasks, however, the choice of data augmentation strategy is known to cause large variances in end performance and be difficult to select [12, 8], with papers often reporting their heuristically found parameter ranges [4]. In practice, it is often simple to formulate a large set of primitive transformation operations, but time-consuming and difficult to find the parameterizations and compositions of them needed for state-of-the-art results. In particular, many transformation operations will have vastly different effects based on parameterization, the set of other transformations they are applied with, and even their particular order of composition. For example, brightness and saturation enhancements might be destructive when applied together, but produce realistic images when paired with geometric transformations.

Given the difficulty of searching over this configuration space, the de facto norm in practice consists of applying one or more transformations in random order and with random parameterizations selected from hand-tuned ranges. Recent lines of work attempt to automate data augmentation entirely, but either rely on

*Authors contributed equally

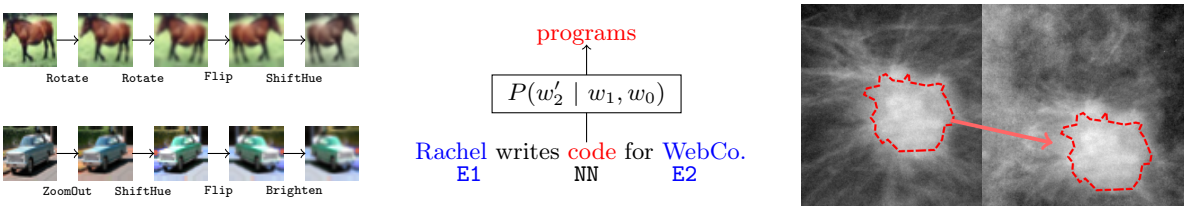


Figure 1: Three examples of transformation functions (TFs) in different domains: Two example sequences of incremental image TFs applied to CIFAR-10 images (*left*); a conditional word-swap TF using an externally trained language model and specifically targeting nouns (`NN`) between entity mentions (`E1`,`E2`) for a relation extraction task (*middle*); and an unsupervised segementation-based translation TF applied to mass-containing mammography images (*right*).

large quantities of labeled data [1, 22], restricted sets of simple transformations [9, 14], or consider only local perturbations that are not informed by domain knowledge [1, 23] (see Section 4). In contrast, our aim is to directly and flexibly leverage domain experts' knowledge of invariances as a valuable form of weak supervision in real-world settings where labeled training data is limited.

In this paper, we present a new method for data augmentation that directly leverages user domain knowledge in the form of transformation operations, and automates the difficult process of composing and parameterizing them. We formulate the problem as one of learning a generative sequence model over black-box *transformation functions (TFs)*: user-specified operators representing incremental transformations to data points that need not be differentiable nor deterministic. For example, TFs could rotate an image by a small degree, swap a word in a sentence, or translate a segmented structure in an image (Fig. 1). We then design a generative adversarial objective [10] which allows us to train the sequence model to produce transformed data points which are still within the data distribution of interest, using unlabeled data. Because the TFs can be stochastic or non-differentiable, we present a reinforcement learning-based training strategy for this model. The learned model can then be used to perform data augmentation on labeled training data for any end discriminative model.

Given the flexibility of our representation of the data augmentation process, we can apply our approach in many different domains, and on different modalities including both text and images. On a real-world mammography image task, we achieve a 3.4 accuracy point boost above randomly composed augmentation by learning to appropriately combine standard image TFs with domain-specific TFs derived in collaboration with radiology experts. Using novel language model-based TFs, we see a 1.4 F1 boost over heuristic augmentation on a text relation extraction task from the ACE corpus. And on a 10%-subsample of the CIFAR-10 dataset, we achieve a 4.0 accuracy point gain over a standard heuristic augmentation approach and are competitive with comparable semi-supervised approaches. Additionally, we show empirical results suggesting that the proposed approach is robust to misspecified TFs. Our hope is that the proposed method will be of practical value to practitioners and of interest to researchers, so we have open-sourced the code at `https://github.com/HazyResearch/tanda`.

2 Modeling Setup and Motivation

In the standard data augmentation setting, our aim is to expand a labeled training set by leveraging knowledge of class-preserving transformations. For a practitioner with domain expertise, providing individual transformations is straightforward. However, high performance augmentation techniques use *compositions* of finely tuned transformations to achieve state-of-the-art results [8, 4, 12], and heuristically searching over this space of all possible compositions and parameterizations for a new task is often infeasible. Our goal is to automate this task by learning to compose and parameterize a set of user-specified transformation operators in ways that are diverse but still preserve class labels.

In our method, transformations are modeled as sequences of incremental user-specified operations, called transformation functions (TFs) (Fig. 1). Rather than making the strong assumption that all the provided TFs preserve class labels, as existing approaches do, we assume a weaker form of class invariance which enables

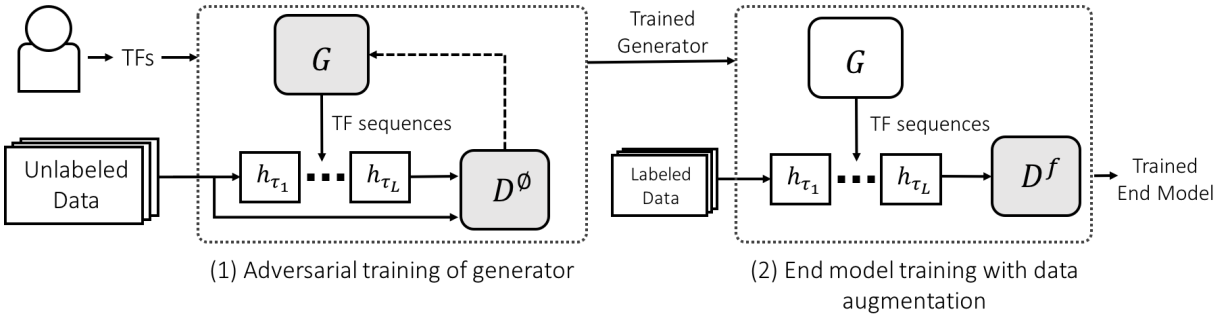


Figure 2: A high-level diagram of our method. Users input a set of transformation functions $h_1, ..., h_K$ and unlabeled data. A generative adversarial approach is then used to train a "null-class" discriminator, $D^{\emptyset}$, and a generator, G, which produces TF sequences $h_{\tau_1}, ..., h_{\tau_L}$. Finally, the trained generator is used to perform data augmentation for an end discriminative model D^f.

us to use *unlabeled* data to learn a generative model over transformation sequences. We then propose two representative model classes to handle modeling both commutative and non-commutative transformations.

2.1 Augmentation as Sequence Modeling

In our approach, we represent transformations as sequences of incremental operations. In this setting, the user provides a set of K TFs, $h_i : \mathcal{X} \mapsto \mathcal{X}$, $i \in [1, K]$. Each TF performs an incremental transformation: for example, h_i could rotate an image by five degrees, swap a word in a sentence, or move a segmented tumor mass around a background mammography image (see Fig. 1). In order to accomodate a wide range of such user-defined TFs, we treat them as black-box functions which need not be deterministic nor differentiable.

This formulation gives us a tractable way to tune both the parameterization and composition of the TFs in a discretized but fine-grained manner. Our representation can be thought of as an implicit binning strategy for tuning parameterizations – e.g. a 15 degree rotation might be represented as three applications of a five-degree rotation TF. It also provides a direct way to represent compositions of multiple transformation operations. This is critical as a multitude of state-of-the-art results in the literature show the importance of using compositions of more than one transformations per image [8, 4, 12], which we also confirm experimentally in Section 5.

2.2 Weakening the Class-Invariance Assumption

Any data augmentation techique fundamentally relies on some assumption about the transformation operations' relation to the class labels. Previous approaches make the unrealistic assumption that all provided transformation operations preserve class labels for all data points. That is,

$$y(h_{\tau_L} \circ \ldots \circ h_{\tau_1}(x)) = y(x) \tag{1}$$

for label mapping function y, any sequence of TF indices $\tau_1, ..., \tau_L$, and *all* data points x.

This assumption puts a large burden of precise specification on the user, and based on our observations, is violated by many real-world data augmentation strategies. Instead, we consider a weaker modeling assumption. We assume that transformation operations will not map between classes, but might destructively map data points out of the distribution of interest entirely:

$$y(h_{\tau_L} \circ \ldots \circ h_{\tau_1}(x)) \in \{y(x), y_{\emptyset}\} \tag{2}$$

where $y_{\emptyset}$ represents an "out-of-distribution" null class. Intuitively, this weaker assumption is motivated by the categorical image classification setting, where we observe that transformation operations provided by the user will almost never turn, for example, a plane into a car, but may often turn a plane into an indistinguishable "garbage" image (Fig. 3). We are the first to consider this weaker invariance assumption, which we believe

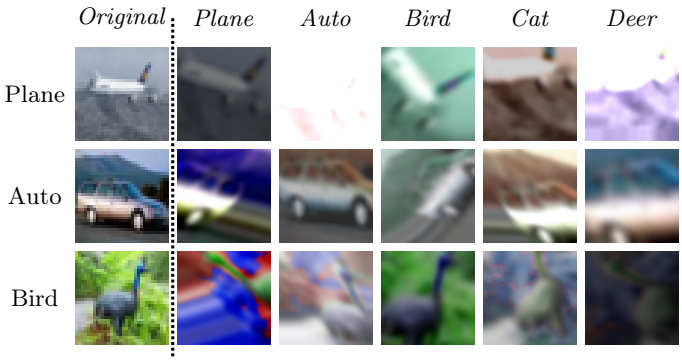


Figure 3: Our modeling assumption is that transformations may map out of the natural distribution of interest, but will rarely map *between* classes. As a demonstration, we take images from CIFAR-10 (each row) and randomly search for a transformation sequence that best maps them to a different class (each column), according to a trained discriminative model. The matches rarely resemble the target class but often no longer look like "normal" images at all. Note that we consider a fixed set of user-provided TFs, not adversarially selected ones.

Figure 4: Some example transformed images generated using an augmentation generative model trained using our approach. Note that this is not meant as a comparison to Fig. 3.

more closely matches various practical data augmentation settings of interest. In Section 5, we also provide empirical evidence that this weaker assumption is useful in binary classification settings and over modalities other than image data. Critically, it also enables us to learn a model of TF sequences using unlabeled data alone.

2.3 Minimizing Null-Class Mappings Using Unlabeled Data

Given assumption (2), our objective is to learn a model G_θ which generates sequences of TF indexes $\tau \in \{1, K\}^L$ with fixed length L, such that the resulting TF sequences $h_{\tau_1}, ..., h_{\tau_L}$ are less likely to map data points into $y_\emptyset$. Crucially, this does not involve using the class labels of any data points, and so we can use unlabeled data. Our goal is then to minimize the the probability of a generated sequence mapping unlabeled data points into the null-class, with respect to θ:

$$J_\emptyset = \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[P(y(h_{\tau_L} \circ \ldots \circ h_{\tau_1}(x)) = y_\emptyset) \right] \tag{3}$$

where $\mathcal{U}$ is some distribution of unlabeled data.

Generative Adversarial Objective In order to approximate $P(y(h_{\tau_1} \circ \ldots \circ h_{\tau_L}(x)) = y_\emptyset)$, we jointly train the generator G_θ and a discriminative model $D^\emptyset_\phi$ using a generative adversarial network (GAN) objective [10], now minimizing with respect to θ and maximizing with respect to ϕ:

$$\tilde{J}_\emptyset = \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[\log(1 - D^\emptyset_\phi(h_{\tau_L} \circ \ldots \circ h_{\tau_1}(x))) \right] + \mathbb{E}_{x' \sim \mathcal{U}} \left[\log(D^\emptyset_\phi(x')) \right] \tag{4}$$

As in the standard GAN setup, the training procedure can be viewed as a minimax game in which the discriminator's goal is to assign low values to transformed, out-of-distribution data points and high values to real in-distribution data points, while simultaneously, the generator's goal is to generate transformation sequences which produce data points that are indistinguishable from real data points according to the discriminator. For $D^\emptyset_\phi$, we use a 9-layer all-convolution CNN as in [25]. Further details are in the Appendix.

Diversity Objective An additional concern is that the model will learn a variety of null transformation sequences (e.g. rotating first left than right repeatedly). Given the potentially large state-space of actions,

and the black-box nature of the user-specified TFs, it seems infeasible to hard-code sets of "null ops" to avoid. To mitigate this, we instead consider a second objective term:

$$J_d = \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[d(h_{\tau_L} \circ \ldots \circ h_{\tau_1}(x), x) \right] \tag{5}$$

where $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is some distance function. For d, we evaluated using both distance in the raw input space, and in the feature space learned by the final pre-softmax layer of the discriminator $D_\phi^\emptyset$. Combining eqns. 4 and 5, our final objective is then $J = \tilde{J}_\emptyset + \alpha J_d^{-1}$ where $\alpha > 0$ is a hyperparameter. We minimize J with respect to θ and maximize with respect to ϕ.

2.4 Modeling Transformation Sequences

We now consider two model classes for G_θ:

Independent Model We first condsider a *mean field* model in which each sequential TF is chosen independently. This reduces our task to one of learning K parameters, which we can think of as representing the task-specific "accuracies" or "frequencies" of each TF. For example, we might want to learn that elastic deformations or swirls should only rarely be applied to images in CIFAR-10, but that small rotations can be applied frequently. In particular, a mean field model also provides a simple way of effectively learning stochastic, discretized parameterizations of the TFs. For example, if we have a TF representing five-degree rotations, `Rotate5Deg`, a marginal value of $P_{G_\theta}(\texttt{Rotate5Deg}) = 0.1$ could be thought of as roughly equivalent to learning to rotate $0.5L$ degrees on average.

State-Based Model There are important cases, however, where the independent representation learned by the mean field model could be overly limited. In many settings, certain TFs may have very different effects depending on which other TFs are applied with them. As an example, certain similar pairs of image transformations might be overly lossy when applied together, such as a blur and a zoom operation, or a brighten and a saturate operation. A mean field model could not represent such disjunctions as these. Another scenario where an independent model fails is where the TFs are non-commutative, such as with lossy operators (e.g. image transformations which use aliasing). In both of these cases, modeling the sequences of transformations could be important. Therefore we consider an long short-term memory (LSTM) network as as a representative sequence model. The output from each cell of the network is a distribution over the TFs. The next TF in the sequence is then sampled from this distribution, and is fed as a one-hot vector to the next cell in the network.

3 Learning a Transformation Sequence Model

The core challenge that we now face in learning G_θ is that it generates sequences over TFs which are not necessarily differentiable or deterministic. This constraint is a critical facet of our approach from the usability perspective, as it allows users to easily write TFs as black-box scripts in the language of their choosing, leveraging arbitrary subfunctions, libraries, and methods. In order to work around this constraint, we now describe our model in the syntax of reinforcement learning (RL), which provides a convenient framework and set of approaches for handling computation graphs with non-differentiable or stochastic nodes [29].

Reinforcement Learning Formulation Let τ_i be the index of the ith TF applied, and $\tilde{x}_i$ be the resulting incrementally transformed data point. Then we consider $s_t = (x, \tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_t, \tau_1, ..., \tau_t)$ as the state after having applied t of the incremental TFs. Note that we include the incrementally transformed data points $\tilde{x}_1, ..., \tilde{x}_t$ in s_t since the TFs may be stochastic. Each of the model classes considered for G_θ then uses a different *state representation* $\hat{s}$. For the mean field model, the state representation used is $\hat{s}_t^{\text{MF}} = \emptyset$. For the LSTM model, we use $\hat{s}_t^{\text{LSTM}} = \mathsf{LSTM}(\tau_t, s_{t-1})$, the state update operation performed by a standard LSTM cell parameterized by θ.

Policy Gradient with Incremental Rewards Let $\ell_t(x, \tau) = \log(1 - D_\phi^\emptyset(\tilde{x}_t))$ be the *cumulative loss* for a data point x at step t, with $\ell_0(x) = \ell_0(x, \tau) \equiv \log(1 - D_\phi^\emptyset(x))$. Let $R(s_t) = \ell_t(x, \tau) - \ell_{t-1}(x, \tau)$ be the *incremental reward*, representing the difference in discriminator loss at incremental transformation step t. We can now recast the first term of our objective $\tilde{J}_\emptyset$ as an expected sum of incremental rewards:

$$U(\theta) \equiv \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[\log(1 - D_\phi^\emptyset(h_{\tau_1} \circ \ldots \circ h_{\tau_L}(x))) \right] = \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[\ell_0(x) + \sum_{t=1}^{L} R(s_t) \right] \tag{6}$$

We omit ℓ_0 in practice, equivalent to using the loss of x as a baseline term. Next, let π_θ be the stochastic transition policy implictly defined by G_θ. We compute the recurrent policy gradient [34] of the objective $U(\theta)$ as:

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[\sum_{t=1}^{L} R(s_t) \nabla_\theta \log \pi_\theta(\tau_t \mid \hat{s}_{t-1}) \right] \tag{7}$$

Following standard practice, we approximate this quantity by sampling batches of n data points and m sampled action sequences per data point. We also use standard techniques of discounting with factor $\gamma \in [0, 1]$ and considering only future rewards [13]. See Appendix C for details.

4 Related Work

We now review related work, both to motivate comparisons in the experiments section and to present complementary lines of work.

Heuristic Data Augmentation Most state-of-the-art image classification pipelines use some limited form of data augmentation [12, 8]. This generally consists of applying crops, flips, or small affine transformations, in fixed order or at random, and with parameters drawn randomly from hand-tuned ranges. In addition, various studies have applied heuristic data augmentation techniques to modalities such as audio [33] and text [21]. As reported in the literature, the selection of these augmentation strategies can have large performance impacts, and thus can require extensive selection and tuning by hand [4, 8] (see Appendix A.1 as well).

Interpolation-Based Techniques Some techniques have explored generating augmented training sets by interpolating between labeled data points. For example, the well-known SMOTE algorithm applies this basic technique for oversampling in class-imbalanced settings [3], and recent work explores using a similar interpolation approach in a learned feature space [6]. [14] proposes learning a class-conditional model of diffeomorphisms interpolating between nearest-neighbor labeled data points as a way to perform augmentation. We view these approaches as complementary but orthogonal, as our goal is to directly exploit user domain knowledge of class-invariant transformation operations.

Adversarial Data Augmentation Several lines of recent work have explored techniques which can be viewed as forms of data augmentation that are adversarial with respect to the end classification model. In one set of approaches, transformation operations are selected adaptively from a given set in order to maximize the loss of the end classification model being trained [32, 9]. These procedures make the strong assumption that all of the provided transformations will preserve class labels, or use bespoke models over restricted sets of operations [30]. Another line of recent work has showed that augmentation via small adversarial linear perturbations can act as a regularizer [11, 23]. While complimentary, this work does not consider taking advantage of non-local transformations derived from user knowledge of task or domain invariances.

Finally, generative adversarial networks (GANs) [10] have recently made great progress in learning complete data generation models from unlabeled data. These can be used to augment labeled training sets as well. Class-conditional GANs [1, 22] generate artificial data points but require large sets of labeled training data to learn from. Standard unsupervised GANs can be used to generate additional out-of-class data points that can then augment labeled training sets [27, 31]. We compare our proposed approach with these methods empirically in Section 5.

5 Experiments

We experimentally validate the proposed framework by learning augmentation models for several benchmark and real-world data sets, exploring both image recognition and natural language understanding tasks. Our focus is on the performance of end classification models trained on labeled datasets augmented with our approach and others used in practice. We also examine robustness to user misspecification of TFs, and sensitivity to core hyperparameters.

5.1 Datasets and Transformation Functions

Benchmark Image Datasets We ran experiments on the MNIST [19] and CIFAR-10 [18] datasets, using only a subset of the class labels to train the end classification models and treating the rest as unlabeled data. We used a generic set of TFs for both MNIST and CIFAR-10: small rotations, shears, central swirls, and elastic deformations. We also used morphologic operations for MNIST, and adjustments to hue, saturation, contrast, and brightness for CIFAR-10.

Benchmark Text Dataset We applied our approach to the *Employment* relation extraction subtask from the NIST Automatic Content Extraction (ACE) corpus [7], where the goal is to identify mentions of employer-employee relations in news articles. Given the standard class imbalance in information extraction tasks like this, we used data augmentation to oversample the minority positive class. The flexibility of our TF representation allowed us to take a straightforward but novel approach to data augmentation in this setting. We constructed a trigram language model using the ACE corpus and Reuters Corpus Volume I [20] from which we can sample a word conditioned on the preceding words. We then used this model as the basis for a set of TFs that select words to swap based on part-of-speech tag and location relative to entities of interest (see Appendix D.2 for details).

Mammography Tumor-Classification Dataset To demonstrate the effectiveness of our approach on real-world applications, we also considered the task of classifying benign versus malignant tumors from images in the Digital Database for Screening Mammography (DDSM) dataset [16, 5, 28], which is a class-balanced dataset consisting of 1506 labeled mammograms. In collaboration with domain experts in radiology, we constructed two basic TF sets. The first set consisted of standard image transformation operations subselected so as not to break class-invariance in the mammography setting. For example, brightness operations were excluded for this reason. The second set consisted of both the first set as well as several novel segmentation-based transplantation TFs. Each of these TFs utilized the output of an unsupervised segmentation algorithm to isolate the tumor mass, perform a transformation operation such as rotation or shifting, and then stitch it into a randomly-sampled benign tissue image. See Fig. 1 (right panel) for an illustrative example, and Appendix D.3 for further details.

5.2 End Classifier Performance

We evaluated our approach by using it to augment labeled training sets for the tasks mentioned above, and show that we achieve strong gains over heuristic baselines. In particular, for a given set of TFs, we evaluate the performance of mean field (*MF*) and LSTM generators trained using our approach against two standard data augmentation techniques used in practice. The first (*Basic*) consists of applying random crops to images, or performing simple minority class duplication for the ACE relation extraction task. The second (*Heur.*) is the standard heuristic approach of applying random compositions of the given set of transformation operations, the most common technique used in practice [4, 12, 15]. For both our approaches (*MF* and *LSTM*) and *Heur.*, we additionally use the same random cropping technique as in the *Basic* approach. We present these results in Table 1, where we report test set accuracy (or F1 score for ACE), and use a random subsample of the available labeled training data. Additionally, we include an extra row for the DDSM task highlighting the impact of adding domain-specific (*DS*) TFs – the segmentation-based operations described above – on performance.

In Table 2 we additionally compare to two related generative-adversarial methods, the Categorical GAN (CatGAN) [31], and the semi-supervised GAN (SS-GAN) from [27]. Both of these methods use GAN-based

architectures trained on unlabeled data to generate new out-of-class data points with which to augment a labeled training set. Following their protocol for CIFAR-10, we train our generator on the full set of unlabeled data, and our end discriminator on ten disjoint random folds of the labeled training set not including the validation set (i.e. $n = 4000$ each), averaging the results.

Task	**%**	**None**	**Basic**	**Heur.**	**MF**	**LSTM**
MNIST	1	90.2	95.3	95.9	96.5	**96.7**
	10	97.3	98.7	99.0	**99.2**	99.1
CIFAR-10	10	66.0	73.1	77.5	79.8	**81.5**
	100	87.8	91.9	92.3	**94.4**	94.0
ACE (F1)	100	62.7	59.9	62.8	62.9	**64.2**
DDSM	10	57.6	58.8	59.3	58.2	61.0
DDSM + DS				53.7	59.9	**62.7**

Table 1: Test set performance of end models trained on subsamples of the labeled training data (%), not including validation splits, using various data augmentation approaches. *None* indicates performance with no augmentation. All tasks are measured in accuracy, except ACE which is measured by F1 score.

Model	**Acc. (%)**
CatGAN	80.42 ± 0.58
SS-GAN	81.37 ± 2.32
LSTM	81.47 ± 0.46

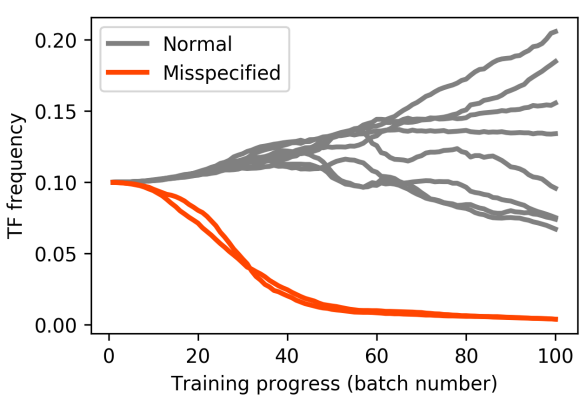
Table 2: Reported end model accuracies, averaged across 10% subsample folds, on CIFAR-10 for comparable GAN methods.

In all settings, we train our TF sequence generator on the full set of unlabeled data. We select a fixed sequence length for each task via an initial calibration experiment (Fig. 5b). We use $L = 5$ for ACE, $L = 7$ for DDSM + DS, and $L = 10$ for all other tasks. We note that our findings here mirrored those in the literature, namely that compositions of multiple TFs lead to higher end model accuracies. We selected hyperparameters of the generator via performance on a validation set. We then used the trained generator to transform the entire training set at each epoch of end classification model training. For MNIST and DDSM we use a 9-layer CNN, for CIFAR10 we use a 56-layer ResNet [15], and for ACE we use a bi-directional LSTM. Additionally, we incorporate a basic transformation regularization term as in [26] (see Appendix D.6), and train for the last ten epochs without applying any transformations as in [12]. In all cases, we use hyperparameters as reported in the literature. For further details of generator and end model training see the Appendix.

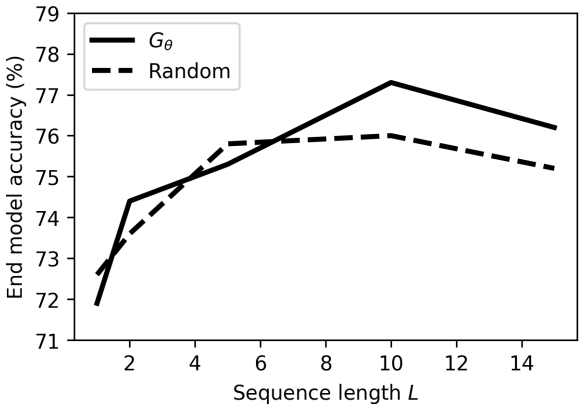
We see that across the applications studied, our approach outperforms the heuristic data augmentation approach most commonly used in practice. Furthermore, the LSTM generator outperforms the simple mean field one in most settings, indicating the value of modeling sequential structure in data augmentation. In particular, we realize significant gains over standard heuristic data augmentation on CIFAR-10, where we are competitive with comparable semi-supervised GAN approaches, but with significantly smaller variance. We also train the same CIFAR-10 end model using the full labeled training dataset, and again see strong relative gains (2.1 pts. in accuracy over heuristic), coming within 2.1 points of the current state-of-the-art [17] using our much simpler end model.

On the ACE and DDSM tasks, we also achieve strong performance gains, showing the ability of our method to productively incorporate more complex transformation operations from domain expert users. In particular, in DDSM we observe that the addition of the segmentation-based TFs causes the heuristic augmentation approach to perform significantly worse, due to a large number of new failure modes resulting from combinations of the segmentation-based TFs – which use gradient-based blending – and the standard TFs such as zoom and rotate. In contrast, our LSTM model learns to avoid these destructive subsequences and achieves the highest score, resulting in a 9.0 point boost over the comparable heuristic approach.

Robustness to TF Misspecification One of the high-level goals of our approach is to enable an easier interface for users by not requiring that the TFs they specify be completely class-preserving. The lack of any assumption of well-specified transformation operations in our approach, and the strong empirical performance realized, is evidence of this robustness. To additionally illustrate the robustness of our approach to misspecified TFs, we train a mean field generator on MNIST using the standard TF set, but with two TFs (shear operations) parameterized so as to map almost all images to the null class. We see in Fig. 5a that the



(a)



(b)

Figure 5: (a) Learned TF frequency parameters for misspecified and normal TFs on MNIST. The mean field model correctly learns to avoid the misspecified TFs. (b) Larger sequence lengths lead to higher end model accuracy on CIFAR-10, while random performs best with shorter sequences, according to a sequence length calibration experiment.

generator learns to avoid applying the misspecified TFs (red lines) almost entirely.

6 Conclusion and Future Work

We presented a method for learning how to parameterize and compose user-provided black-box transformation operations used for data augmentation. Our approach is able to model arbitrary TFs, allowing practitioners to leverage domain knowledge in a flexible and simple manner. By training a generative sequence model over the specified transformation functions using reinforcment learning in a GAN-like framework, we are able to generate realistic transformed data points which are useful for data augmentation. We demonstrated that our method yields strong gains over standard heursitic approaches to data augmentation for a range of applications, modalities, and complex domain-specific transformation functions. There are many possible future directions of research for learning data augmentation strategies in the proposed model, such as conditioning the generator's stochastic policy on a featurized version of the data point being transformed, and generating TF sequences of dynamic length. More broadly, we are excited about further formalizing data augmentation as a novel form of weak supervision, allowing users to directly encode domain knowledge about invariants into machine learning models.

Acknowledgements We would like to thank Daniel Selsam, Ioannis Mitliagkas, Christopher De Sa, William Hamilton, and Daniel Rubin for valuable feedback and conversations. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) SIMPLEX program under No. N66001-15-C-4043, the DARPA D3M program under No. FA8750-17-2-0095, DARPA programs No. FA8750-12-2-0335 and FA8750-13-2-0039, DOE 108845, National Institute of Health (NIH) U54EB020405, the Office of Naval Research (ONR) under awards No. N000141210041 and No. N000141310129, the Moore Foundation, the Okawa Research Grant, American Family Insurance, Accenture, Toshiba, and Intel. This research was also supported in part by affiliate members and other supporters of the Stanford DAWN project: Intel, Microsoft, Teradata, and VMware. This material is based on research sponsored by DARPA under agreement number FA8750-17-2-0095. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, AFRL, NSF, NIH, ONR, or the U.S. Government.

References

[1] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[2] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[4] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep big simple neural nets excel on handwritten digit recognition, 2010. *Cited on*, 80.

[5] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle, L. Tarbox, and F. Prior. The cancer imaging archive (TCIA): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057, 2013.

[6] T. DeVries and G. W. Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017.

[7] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1, 2004.

[8] A. Dosovitskiy, P. Fischer, J. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks, arxiv preprint. *arXiv preprint arXiv:1506.02753*, 2015.

[9] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard. Adaptive data augmentation for image classification. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 3688–3692. IEEE, 2016.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[12] B. Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[13] E. Greensmith, P. L. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[14] S. Hauberg, O. Freifeld, A. B. L. Larsen, J. Fisher, and L. Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, pages 342–350, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[16] M. Heath, K. Bowyer, D. Kopans, R. Moore, and W. P. Kegelmeyer. The digital database for screening mammography. In *Proceedings of the 5th international workshop on digital mammography*, pages 212–218. Medical Physics Publishing, 2000.

[17] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[21] X. Lu, B. Zheng, A. Velivelli, and C. Zhai. Enhancing text categorization with semantic-enriched representation and training data augmentation. *Journal of the American Medical Informatics Association*, 13(5):526–535, 2006.

[22] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[23] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

[24] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 313–318. ACM, 2003.

[25] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[26] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *CoRR*, abs/1606.04586, 2016.

[27] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[28] R. Sawyer Lee, F. Gimenez, A. Hoogi, and D. Rubin. Curated breast imaging subset of DDSM. In *The Cancer Imaging Archive*, 2016.

[29] J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pages 3528–3536, 2015.

[30] L. Sixt, B. Wild, and T. Landgraf. Rendergan: Generating realistic labeled data. *arXiv preprint arXiv:1611.01331*, 2016.

[31] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[32] C. H. Teo, A. Globerson, S. T. Roweis, and A. J. Smola. Convex learning with invariances. In *Advances in neural information processing systems*, pages 1489–1496, 2008.

[33] 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. *Submitted to ICASSP*, 2017.

[34] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of IGPL*, 18(5):620–634, 2010.

A Additional Background

A.1 Review of Data Augmentation Use in State-of-the-Art

To underscore both the omnipresence and diversity of heuristic data augmentation in practice, we compiled a list of the top ten models for the well documented CIFAR-10 and CIFAR-100 tasks (Table 3). We see that in 10 out of 10 of the top CIFAR-10 results and 9 out of 10 of the top CIFAR-100 results use data augmentation, for average boosts (when reported) of 3.71 and 13.39 points in accuracy, respectively. Moreover, we see that while some sets of papers inherit a simple data augmentation strategy from prior work (in particular, all the

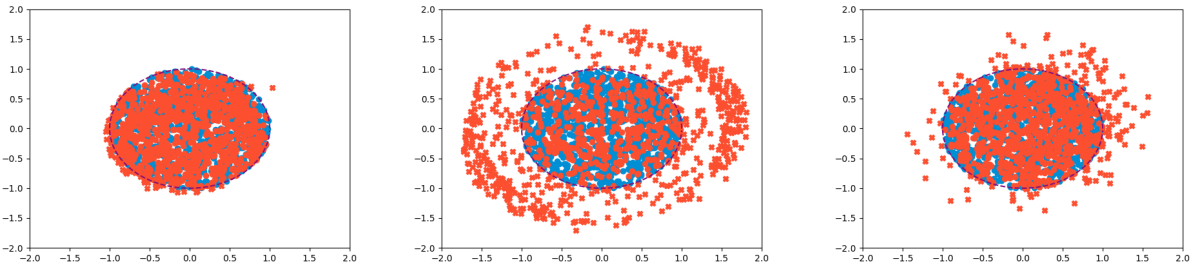
recent ResNet variants), there are still a large variety of approaches. And in general, the particular choice of data augmentation strategy is widely reported to have large effects on performance.

Note that the below table is compiled from a well-known online compendium [1] and from the latest CVPR best paper [17] (indicated by a *) which achieves new state-of-the-art results. We compile it for illustrative purposes and it is not necessarily comprehensive. Also note that we selected CIFAR-10/100 both as a representative and well-studied task, but also due to the availability of published results. For competitions such as ImageNet, although data augmentation is widely reported to be critical, many top results are reported very opaquely, with little described about implementation details such as data augmentation.

B Additional Experiments

B.1 Synthetic Data Examples

Simple Synthetic Setup As both a diagnostic tool and as a simple way to probe the properties of our approach, we construct a simple synthetic dataset consisting of points in two dimensions, uniformly selected in a ball of radius $r = 1$ around the origin. We then consider various displacement vectors as our TFs. We consider the same two generative models as in our main experiments – mean field and LSTM – and use either a basic fully-connected two-layer neural network or an oracle discriminator $f(x) = 1\{||x|| < 1\}$.



(a) Mean field model on TF set 1 (b) Mean field model on TF set 2 (c) LSTM model on TF set 2

Figure 6: Original data points (blue) are transformed using sequences of vector displacement TFs ($L = 10$) drawn from G_θ, producing augmented data points (red). G_θ is either a mean field model or an LSTM, trained with an orcale discriminator $D^\emptyset$ for 15 epochs.

Synthetic Experiments In this setting, we define $y_\emptyset(x) = 1\{||x|| \geq 1\}$, and consider two different TF sets:

1. *Good vs. Bad TFs:* In a first toy scenario we consider TFs which are vector displacements of random direction, with magnitude drawn from one of two distributions, $\mathcal{N}(\mu_1, \sigma_1)$ or $\mathcal{N}(\mu_2, \sigma_2)$, where $\mu_1 > 1 > \mu_2$. In other words, the model should learn not to select certain individual TFs.
2. *Lossy TFs:* We consider a second toy setting where random-direction displacement TFs have magnitude drawn uniformly from $\mathcal{N}(\mu, \sigma)$, $\mu < 1$; however the magnitude of each TF decays exponentially with the distance a point is outside of the unit ball. This simulates the setting where TFs are irrecoverably lossy when applied in certain sequences.

As expected, we see that while the mean field model is able to model the first setting (Figure 6a), it fails to adequately represent the second one (Figure 6b), whereas the RNN model is able to (Figure 6c).

B.2 Robustness to Transformed Test Data

We run a simple experiment to test the robustness of the trained end classification models to the individual TFs in the TF sets used. Specifically, on CIFAR-10 we create ten transformed copies a 10% subsample of the test data by transforming with a single TF, and then test the end model on this set. We compare our

[1]

Dataset	Pos.	Name	Err. w/DA	Err. w/o DA	Notes
CIFAR-10	1	DenseNet	3.46	-	Random shifts, flips
	2	Fractional Max-Pooling	3.47	-	Randomized mix of translations, rotations, reflections, stretching, shearing, and random RGB color shift operations
	3*	Wide ResNet	4.17	-	Random shifts, flips
	4	Striving for Simplicity: The All Convolutional Net	4.41	9.08	"Heavy" augmentation: images expanded, then scaled, rotated, color shifted randomly
	5*	FractalNet	4.60	7.33	Random shifts, flips
	6*	ResNet (1001-Layer)	4.62	10.56	Random shifts, flips
	7*	ResNet with Stochastic Depth (1202-Layer)	4.91	-	Random shifts, flips
	8	All You Need is a Good Init	5.84	-	Random shifts, flips
	9	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	6.05	7.62	Flips, random shifts, other simple ones
	10	Spatially-Sparse Convolutional Neural Networks	6.28	-	Affine transformations
CIFAR-100	1*	DenseNet	17.18	-	Random shifts, flips
	2*	Wide ResNets	20.50	-	Random shifts, flips
	3*	ResNet (1001-Layer)	22.71	33.47	Random shifts, flips
	4*	FractalNet	23.30	35.34	Random shifts, flips
	5	Fast and Accurate Deep Network Learning by Exponential Linear Units	-	24.28	
	6	Spatially-Sparse Convolutional Neural Networks	24.3	-	Affine transformations
	7*	ResNet with Stochastic Depth (1202-Layer)	24.58	37.80	Random shifts, flips
	8	Fractional Max-Pooling	26.39	-	Randomized mix of translations, rotations, reflections, stretching, and shearing operations, and random RGB color shifts
	9*	ResNet (110-Layer)	27.22	44.74	Random shifts, flips
	10	Scalable Bayesian Optimization Using Deep Neural Networks	27.4	-	Hue, saturation, scalings, horizontal flips

Table 3: Current state-of-the-art image classification models as ranked by reported performance on the CIFAR-10 and CIFAR-100 tasks, and their error with (*Err. w/ DA*) and without (*Err. w/o DA*) data augmentation. We include both scores and particular data augmentation techniques when reported, although the latter is rarely reported with great precision.

approach with heuristic random augmentation and no data augmentation of the model during training, and consider rotations, zooms, shears, and hue shifts. Results are presented in Figure 7.

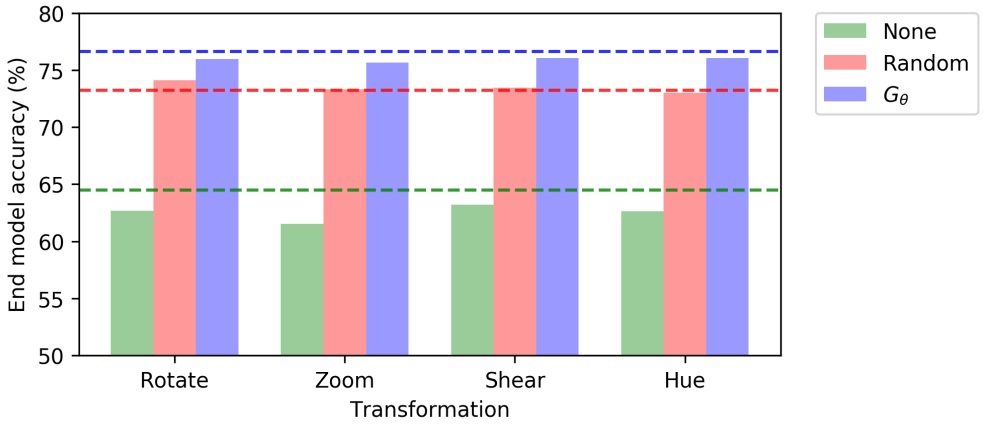


Figure 7: Accuracy scores on random 10% subsamples of test data (dotted lines) and on versions augmented with a single transformation (vertical bars) with parameters drawn uniformly at random.

We can consider evaluating the results in terms of absolute robustness – i.e. model accuracy – and relative robustness, i.e. the change in model score when applied to the transformed test set. Roughly we see that our approach is most absolutely robust. Random appears to be most relatively robust, in particular on larger transformations, which we hypothesize our approach mostly learned to avoid applying during training.

C Reinforcement Learning Formulation Details

C.1 Variance reduction methods

In Section 3, the vanilla policy gradient of our objective was given as

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[\sum_{t=1}^{L} R(s_t) \nabla_\theta \log \pi_\theta(\tau_t \mid \hat{s}_{t-1}) \right]$$

Noting that actions τ_t only impact future outcomes, following standard practice, we apply only future rewards as another variance reduction technique:

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[\sum_{t=1}^{L} \nabla_\theta \log \pi_\theta(\tau_t \mid \hat{s}_{t-1}) \sum_{t'=t}^{L} \gamma^{t'-t} R(s_t) \right]$$

where $\gamma \in [0, 1]$ is a discounting factor. Additionally, we use a baseline term b_t to reduce variance when estimating $\nabla_\theta U(\theta)$:

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau \sim G_\theta} \mathbb{E}_{x \sim \mathcal{U}} \left[\sum_{t=1}^{L} \nabla_\theta \log \pi_\theta(\tau_t \mid \hat{s}_{t-1}) \left(\left(\sum_{t'=t}^{L} \gamma^{t'-t} R(\hat{s}_{t'}) \right) - b_t \right) \right]$$

C.2 Policy gradient estimation

Using a batch of n data points and m sampled action sequences per data point – given by the state representations $\{\hat{s}^{(i,j)}\}$ and action sequences $\{\tau^{(i,j)}\}$ – the gradient estimate is computed as:

$$\nabla_\theta \hat{U}(\theta) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \left[\sum_{t=1}^{L} \nabla_\theta \log \pi_\theta(\tau_t^{(i,j)} \mid \hat{s}_{t-1}^{(i,j)}) \left(\left(\sum_{t'=t}^{L} \gamma^{t'-t} R(\hat{s}_{t'}^{(i,j)}) \right) - \hat{b}_t \right) \right]$$

where the baseline term $\hat{b}_t$ is also computed using the batch:

$$\hat{b}_t = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t'=t}^{L} \gamma^{t'-t} R\left(\hat{s}_{t'}^{(i,j)}\right)$$

In our experiments, we fixed $n = 32$ and $m = 5$.

D Experimental Details

D.1 Benchmark Image Datasets

We use the MNIST dataset with 5000 training data points used as a validation set. We use the following TFs:

- Rotation (2.5, -2.5, 5, -5, 10, and -10 degrees)
- Zoom (0.9x, 1.1x)
- Shear (0.1, -0.1, 0.2, -0.2, 0.4, and -0.4 degrees)
- Swirl (0.1, -0.1, 0.2, -0.2, 0.4, and -0.4 degrees)
- Random elastic deformations (α = 1.0, 1.25, and 1.5)
- Erosion
- Dilation

For the CIFAR-10 dataset, we use the following TFs:

- Rotation (2.5, -2.5, 5, -5 degrees)
- Zoom (0.9x, 1.1x, 0.75x, 1.25x)
- Shear (0.1, -0.1, 0.25, and -0.25 degrees)
- Swirl (0.1, -0.1, 0.25, -0.25 degrees)
- Hue Shift (by 0.1, -0.1, 0.25, and -0.25)
- Enhance contrast (by 0.75, 1.25, 0.5, and 1.5)
- Enhance brightness (by 0.75, 1.25, 0.5, and 1.5)
- Enhance color (by 0.75, 1.25, 0.5, and 1.5)
- Horizontal flip

For both datasets, we also applied random padding (by 4 pixels on each side) followed by random crops back to the original dimensions during training. We note that the choice of certain TFs to use in certain datasets was deliberate – for example, we would not expect horizontal flips or hue shifts to be appropriate in MNIST, or erosion and dilation to be useful in CIFAR-10. However, the particular choice of parameterizations was mainly due to disjoint implementations of the two experiments. For further details of the TF implementations used, see our code, which will be open-sourced after the review process.

D.2 Benchmark Text Dataset

The ACE corpus consists of news articles and broadcast transcripts, all of which are pretagged with entity mentions. The objective of the Employment relation subtask is to extract *Person-Organization* entity pairs which are implied to have an affiliation in the text. We pose this as a binary classification problem by first identifying relation *candidates*: any pair of *Person-Organization* entities which occur in the same sentence. As noted in Section 5, there are far more true negative candidates than true positive candidates. The end model is trained to classify relation candidates as either true or false relations based on the raw text of the sentence in which they occur.

The language model described in Section 5 was constructed by recording counts of unigrams following each unique trigram, bigram, and unigram in the corpus. Laplace smoothing was applied to the counts, and basic filtering was applied to the n-grams. The sampler falls back to using bigrams then unigrams if the trigram preceding the word we want to swap was filtered out of the corpus. We used the following TFs in all experiments:

- Replace a noun to the left of both entities
- Replace a noun between the two entities
- Replace a noun to the right of both entities
- Replace a verb to the left of both entities
- Replace a verb between the two entities
- Replace a verb to the right of both entities
- Replace an adjective to the left of both entities
- Replace an adjective between the two entities
- Replace an adjective to the right of both entities

D.3 DDSM Mammography Task

We use the following transformation functions:

1. Rotate Image: Rotate the entire mammogram by a deterministic angle θ. Tumor geometry is fundamentally invariant to 2-D orientation. TF run with $\theta \in [-5^o, -2.5^o, 2.5^o, 5^o]$.

2. Zoom Image: Zoom in on the entire mammogram by a deterministic factor γ. Tumor classification is insensitive to γ for γ close to one. TF run with $\gamma \in \{0.98, 1.02\}$

3. Enhance Image Contrast: Enhance contrast values of grayscale image by a deterministic factor γ. Tumor classification is insensitive to γ for γ close to one. TF run with $\gamma \in \{0.95, 1.05\}$

4. Translate and Transplant Image: Extract pixels within mass segmentation. Perform translation of a bounding box of side length N pixels about mass center by a deterministic vector $\hat{g}$. Transplant the translated bounding box containing the mass onto a randomly sampled normal tissue image using Poisson blending. [24]. Retains information about the mass itself within the context of a different set of normal tissue background. The bounding box also retains information about the tissue in the tumor near field. TF run with $N = 10$, $\hat{g} \in \{(-3, 0), (3, 0), (0, -3), (0, 3), (0, 0)\}$.

5. Rotate and Transplant Image: Extract pixels within mass segmentation. Perform rotation of a bounding box of side length N pixels about mass center by a deterministic angle θ. Transplant the rotated bounding box containing the mass onto a randomly sampled normal tissue image using Poisson blending. [24]. Retains information about the mass itself within the context of a different set of normal tissue background. The bounding box also retains information about the tissue in the tumor near field. TF run with $N = 10$, $\theta \in \{-5^o, -2.5^o, 2.5^o, 5^o\}$.

Note that for the Poisson blending TFs, it is important that the translation and rotation domains be specified such that excessive proximity to the boundary of the destination image does not introduce spurious gradient information into the blended image.

D.4 Details of Generative Adversarial Network Models

All models, both for the generator training as described in this section, and the end classification model training described next, were implemented in Tensorflow [2].

Discriminator For image tasks, the discriminator used in the training of the generator in our approach was the same model as in [25], a 9-layer all-convolution CNN with leaky ReLUs and batch norm. For the text task, we used a unidirectional RNN with basic LSTM cells.

Mean Field Model The mean field model is represented simply as a length K vector of unbounded variables, where K is the number of TFs. Applying the softmax function to this vector yields the TF sampling distribution.

LSTM In the LSTM model, we create a length-L RNN with basic LSTM cells. The input and output size for each cell is K. We feed an indicator vector of the last TF used as the input to each cell, except for the first cell, which recieves a randomly initialized variable vector as its input. The output of each cell is shifted and scaled to range from $-r$ to r, where r is a hyperparameter. Applying the softmax function to the shifted and scaled output yields the stochastic policy: a sampling distribution over the K TFs. In our experiments, we fix $r = 2$ to avoid overfitting.

Training and Model Selection Procedure We trained the TF sequence generators jointly with the discriminator using SGD with momentum (fixed at 0.9), in an adversarial manner as described in [10]. We performed an initial search over the TF sequence length L as described in Section 5, and then held it fixed at $L = 10$ for all subsequent experiments. We searched over a range of values for learning rates for the generator and discriminator, as well as for hyperparameters specific to our formulation, such as the diversity objective term coefficient α, the diversity objective term distance metric d (choosing between distance in the raw input space or in the feature space learned by the final pre-softmax layer of the discriminator), and whether or not to split the data used for the discriminator and generator training steps.

We selected final generators to use for test set evaluation by using them to augment training data for end classification models then evaluated on the validation set. In addition, we filtered some generators out based on their loss (according to the discriminator $D_\phi^\emptyset$) as compared to that of random TF sequences.

Diversity Objective For the diversity objective term, we tried both distance in the raw pixel-level input space and distance in the feature space learned by the final pre-softmax layer of the discriminator as choices for distance metric d. During training of the generators, we measured the average pairwise generalized Jaccard distance. For CIFAR-10, as an example, the final batches had an average distance of 0.52 compared to 0.86 for randomly generated sequences, which implied diversity in the learned sequences. We also measured the ratio of unique TF n-grams, and measured 0.37 compared to 0.98 for random sequences as expected.

D.5 End Classification Models

MNIST and DDSM For MNIST and DDSM we use a similar architecture to the discriminator in the previous section, adapted for the multinomial classification setting: a 9-layer all-convolution CNN with leaky ReLUs and batch norm.

CIFAR-10 Given the flexibility of end classifier choice with our approach, for CIFAR-10 we used a more computationally expensive but standard model: a 56-layer ResNet as described in [15]. We used batch norm, regularization, learning rate schedule, and all other hyperparameters as reported in [15].

[2] https://www.tensorflow.org

ACE The end model used for the ACE task was a bidirectional recurrent neural network using LSTM cells with attention mechanisms. The maximum sentence length and attention window length were both 50. Word embeddings were initialized from pretrained vectors via [2], and updated during training. Hyperparameters were selected via a cursory grid search, and fixed for experiments.

D.6 End Model Training

Basic Training Procedure with Data Augmentation We trained all end models using minibatch stochastic gradient descent with momentum (fixed at 0.9), using a fixed learning rate schedule set once for each model and then fixed for all experiments. To perform data augmentation, during end classifier training we transformed some portion of each minibatch, $p_{\text{transform}}$. For all experiments we used $p_{\text{transform}} = 1.0$. Additionally, for the last ten epochs of training, we switched to $p_{\text{transform}} = 0.0$ following reported practice in the literature [12]. For all other hyperparameters we used default values as reported in the respective literature, held fixed at these values for all experiments.

Transformation Regularization Term We additionally apply a *transformation regularization (TR)* term to the transformed data points for all image experiments by adding a term to the loss function which is the distance between the pre-softmax layer logits for each data point and its transformed copy, similar to the term in [26]. Given the fact that we are producing these transformed data points anyway, incorporating this term introduces little additional overhead.

Task	%	Augmentation Model	TR Term Coefficient	Accuracy (Dev.)
CIFAR-10	10	Heuristic	0	77.4
		Heuristic	0.1	77.5
		LSTM	0	80.4
		LSTM	0.1	81.6

Table 4: A simple study of the effect of adding a transformation regularization (TR) term to the objective function, evaluated on a labeled validation set. We see that adding the term improves performance for both heuristic (random) TF sequences and for TF sequences generated by the trained LSTM model, and that there is a larger positive effect for the latter.

In an early calibration experiment (Table 4), we found that introducing this regularization term (using a coefficient of 0.1 and unlabeled data batch size of 20% that of the labeled data batch size) yielded improvements in performance to the end model with both learned transformation sequences and random sequences. However, we see that the positive effect is much larger for the trained LSTM sequences (1.2 points versus 0.1 points in accuracy). We chose to subsequently keep this term fixed, viewing further calibration and exploration of this term as largely orthogonal to our central experimental questions. However, we believe that this is an extremely interesting and empirically proimising area for future study, especially given the indication that this term may be more effective when used in conjunction with a trained augmentation model such as ours.