

Automating the Art of Data Augmentation

Part I Overview

Edited by Sharon Li (<http://www.yixuanli.net/~albertgu/>) and Chris Ré (<https://cs.stanford.edu/people/chismre/>)

Posted on February 26, 2020

Data augmentation is a de facto technique used in nearly every state-of-the-art model in applications such as image and text classification. Heuristic data augmentation schemes are often tuned manually by human experts with extensive domain knowledge, and may result in suboptimal augmentation policies. In this blog post, we provide a broad overview of recent efforts in this exciting research area, which resulted in new algorithms for automating the search process of transformation functions, new theoretical insights that improve the understanding of various augmentation techniques commonly used in practice, and a new framework for exploiting data augmentation to patch a flawed model and improve performance on crucial subpopulation of data.

Why Data Augmentation?

Modern machine learning models, such as deep neural networks, may have billions of parameters and accordingly require massive labeled training datasets—which are often not available. The technique of artificially expanding labeled training datasets—known as data augmentation—has quickly become critical for combatting this data scarcity problem. Today, data augmentation is used as a secret sauce in nearly every state-of-the-art model for image classification, and is becoming increasingly common in other modalities such as natural language understanding as well. The goal of this blog post is to provide an overview of recent efforts in this exciting research area.

Heuristic data augmentation schemes often rely on the composition of a set of simple transformation functions (TFs) such as rotation and flip (see Figure 1). When chosen carefully, data augmentation schemes tuned by human experts can improve model performance. However, such heuristic strategies in practice can cause large variances in end model performance, and may not produce parameterizations and compositions needed for state-of-the-art models. For example, brightness and saturation enhancements might produce unrecognizable images when applied together, but produce realistic images when paired with geometric transformations.

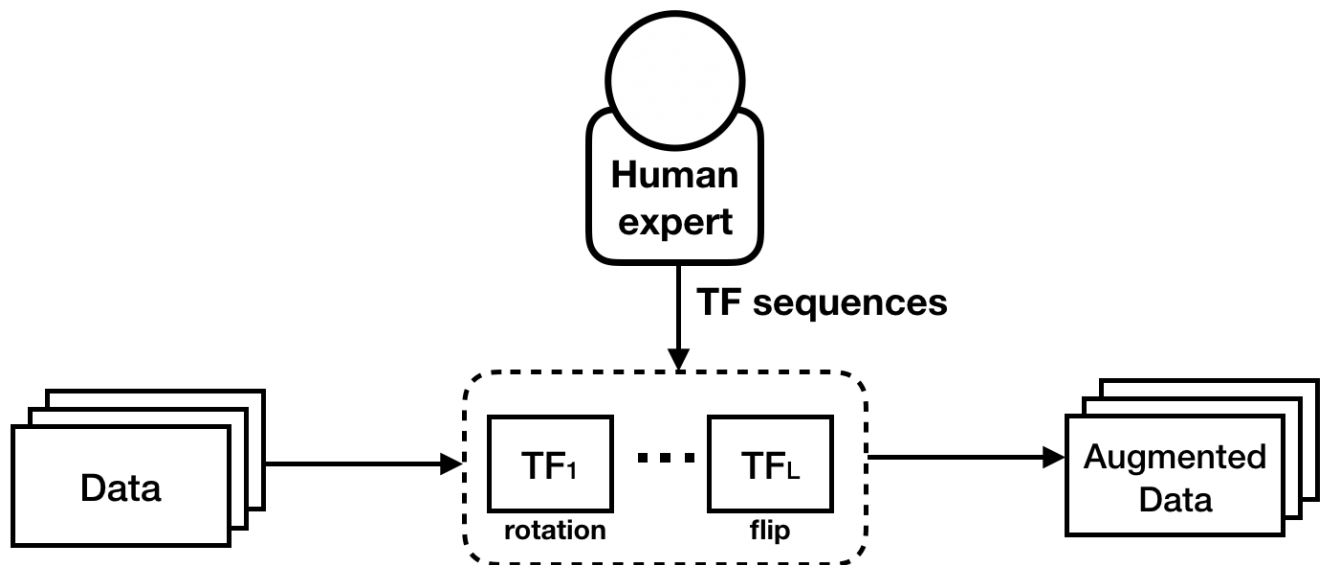


Figure 1. Heuristic data augmentations apply a deterministic sequence of transformation functions tuned by human experts.

The Need for Automated Data Augmentation

The limitations of conventional approaches that manually select which transformation function to apply reveal huge opportunities for further research advances. Below we summarize a few challenges that motivate some of the works in the area of data augmentation.

- From **manual** to **automated** search algorithms: As opposed to performing suboptimal manual search, automated data augmentation approaches hold promise to search for more powerful parameterizations and compositions of transformations. How can we design learnable algorithms that explore the space of transformation functions efficiently and effectively, and find augmentation strategies that can outperform human-designed heuristics?
- From **practical** to **theoretical** understanding: Despite the rapid progress of creating various augmentation approaches pragmatically, understanding their benefits remains a mystery because of a lack of analytic tools. How can we theoretically understand various data augmentations used in practice?
- From **coarse-grained** to **fine-grained** model quality assurance: While most existing data augmentation approaches focus on improving the overall performance of a model, it is often imperative to have a finer-grained perspective on critical subpopulations of data. When a model exhibits inconsistent predictions on important subgroups of data, how can we exploit data augmentations to mitigate the performance gap in a prescribed way?

We'll proceed in three separate blogs, and describe ideas and research works to overcome each of the challenges above. We offer code implementing some of the algorithms. You can learn and explore various data augmentation approaches!

Related Posts

- Automating the Art of Data Augmentation (Part II: Practical Methods)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part2/>)
- Automating the Art of Data Augmentation (Part III: Theory)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part3/>)
- Automating the Art of Data Augmentation (Part IV: New Direction)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part4/>)

← **PREVIOUS POST** (</2019-10-10-NONEUCLIDEAN/>)

NEXT POST → (</2020-02-26-DATA-AUGMENTATION-PART2/>)



(</feed.xml>)



(<https://github.com/HazyResearch>)

HazyResearch • 2020 • hazyresearch.cs.stanford.edu (<http://hazyresearch.cs.stanford.edu/>)

Theme by beautiful-jekyll (<https://deanattali.com/beautiful-jekyll/>)

Automating the Art of Data Augmentation

Part II Practical Methods

Posted on February 26, 2020

Instead of performing manual search, automated data augmentation approaches hold promise to search for more powerful parameterizations and compositions of transformations. Perhaps the biggest difficulty with automating data augmentation is how to search over the space of transformations. This can be prohibitively expensive due to the large number of transformation functions in the search space. How can we design learnable algorithms that explore the space of transformation functions efficiently and effectively, and find augmentation strategies that can outperform human-designed heuristics? In this blog post, we describe a few recent practical methods that address this important problem.

Learnable Data Augmentations

TANDA (Ratner et al. 2017) proposes a method to learn augmentations, which models data augmentations as sequences of *Transformation Functions (TFs)* provided by users. For example, these might include “*rotate 5 degrees*” or “*shift by 2 pixels*”. This framework consists of two core components (1) **learning a TF sequence generator** that results in useful augmented data points, and (2) **using the sequence generator** to augment training sets for a downstream model. Under this framework, the objective and design choice of the TF sequence generator can be instantiated in various ways. Using a similar framework, a subsequent line of work including AutoAugment (Cubuk et al. 2018), RandAugment (Cubuk et al. 2019) and Adversarial AutoAugment (Zhang et al. 2019) have demonstrated state-of-the-art performance using learned augmentations.

In the following, we describe these methods that attempt to learn data augmentations in more detail.

TANDA

In TANDA, a TF sequence generator is trained to produce realistic images by having to fool a discriminator network, following the GANs framework (Goodfellow et al. 2014). That is, we can reasonably assume that we won’t turn an image of a plane into one of a dog, but we might turn it into an indistinguishable garbage image! Such an assumption allows us to leverage generative adversarial networks (GANs), where we simultaneously learn a generator and a discriminator. As shown in Figure 1, the objective for the generator is to produce sequences of TFs such that the

augmented data point can fool the discriminator; whereas the objective for the discriminator is to produce values close to 1 for data points in the original training set and values close to 0 for augmented data points.

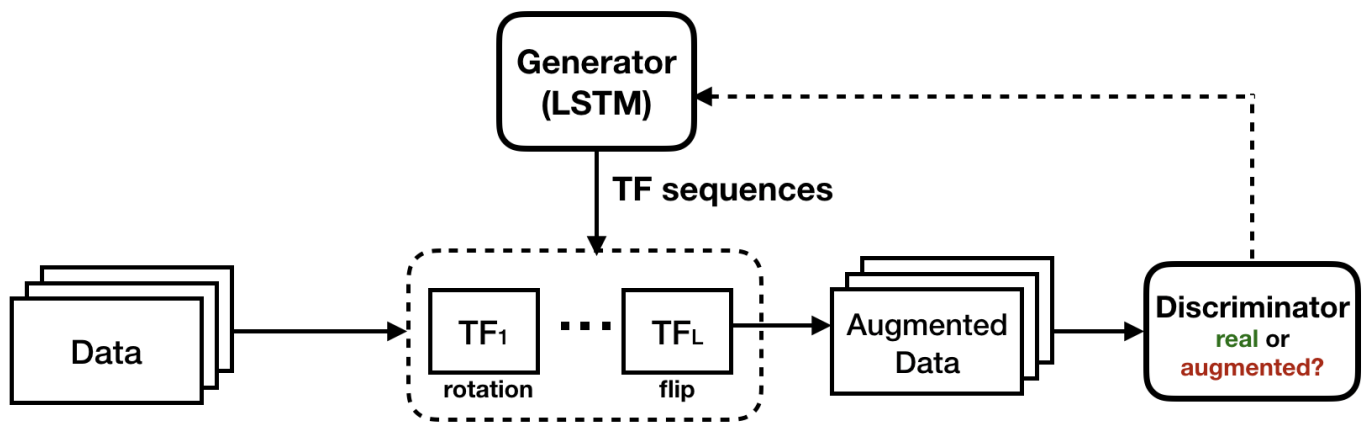


Figure 1: Automating data augmentation with TANDA (Ratner et al. 2017). A TF sequence generator is trained adversarially to produce augmented images that are realistic compared to training data.

Does learning an augmentation model produce better end classifier results than heuristic data augmentation approaches? In our experiments, we show the efficacy of our approach on both image and text datasets, achieving improvements of **4.0%** on CIFAR-10, **1.4 F1 points** on the ACE relation extraction task, and **3.4%** on a medical imaging dataset, as compared to standard heuristic augmentation approaches.

AutoAugment

Using a similar framework, AutoAugment (Cubuk et al. 2018) demonstrated state-of-the-art performance using learned augmentation policies. In this work, a TF sequence generator learns to directly optimize for validation accuracy on the end model (see Figure 2), instead of optimizing for realisticness of augmented images as in TANDA. However, the search process can be computationally expensive due to the need to train a classification model in every gradient step for the generator. To address this issue, AutoAugment searches for augmentation policies on a surrogate dataset that is orders of magnitude smaller than the original dataset.

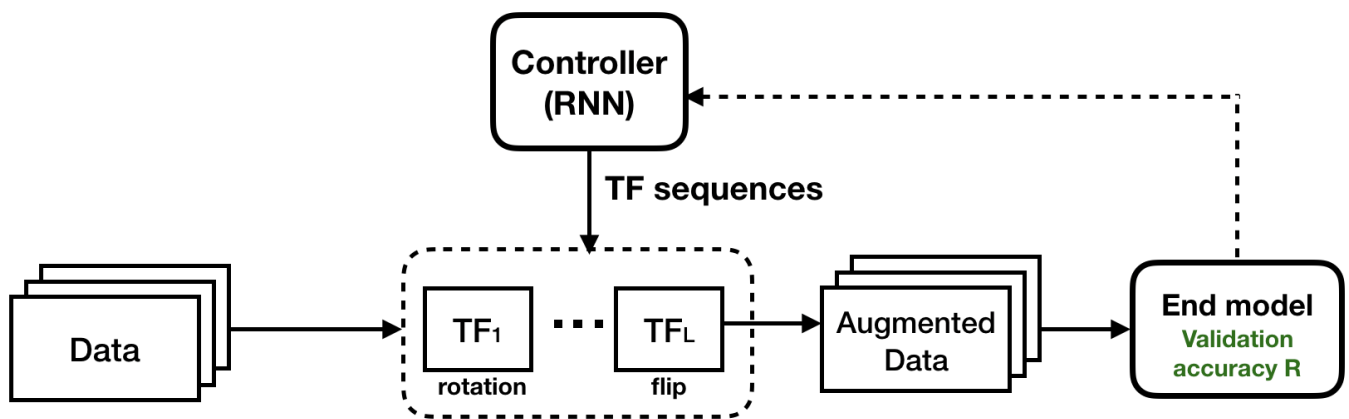


Figure 2: AutoAugment (Cubuk et al. 2018) can be seen as a variant similar to the TANDA framework. A TF sequence generator is trained to optimize for the validation accuracy on the end model.

RandAugment

More recently, RandAugment (Cubuk et al. 2019) found that simple random sampling over the transformation functions with grid search over the parameters of each transformation can outperform AutoAugment. Specifically, they replace the learned TF sequences and probabilities for applying each TF with a parameter-free procedure, which always selects a transformation with uniform probability. They reduce the search space by only learning the magnitude of each transformation function. RandAugment demonstrated **0.6%** increase over the previous state-of-the-art on ImageNet classification with less computational cost. The overview of RandAugment learning framework is illustrated in Figure 3 below.

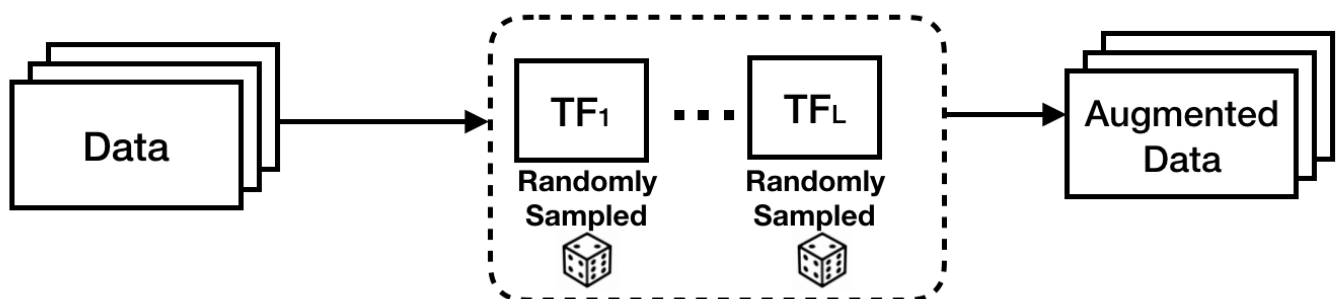


Figure 3: RandAugment (Cubuk et al. 2019) reduces the computational cost by performing random sampling over the transformation function space. Only the magnitude parameters associated with each TF are learned.

Adversarial AutoAugment

Very recently, Zhang et al. proposed a computationally-affordable data augmentation method called Adversarial AutoAugment, establishing a new state-of-the-art on several image classification tasks. They propose an adversarial framework to jointly optimize the end model training and augmentation policy search. As shown in Figure 4, the TF sequence generator attempts to increase the training loss of the end model through generating adversarial augmentation policies, while the end model is trained to be robust against those hard examples and therefore improves the generalization. Compared to AutoAugment, this leads to about **12×** reduction in computing cost and **11×** shortening in time overhead on ImageNet, although the overall search can still be expensive (1280 GPU hours estimated on 64 NVIDIA Tesla V100s!).

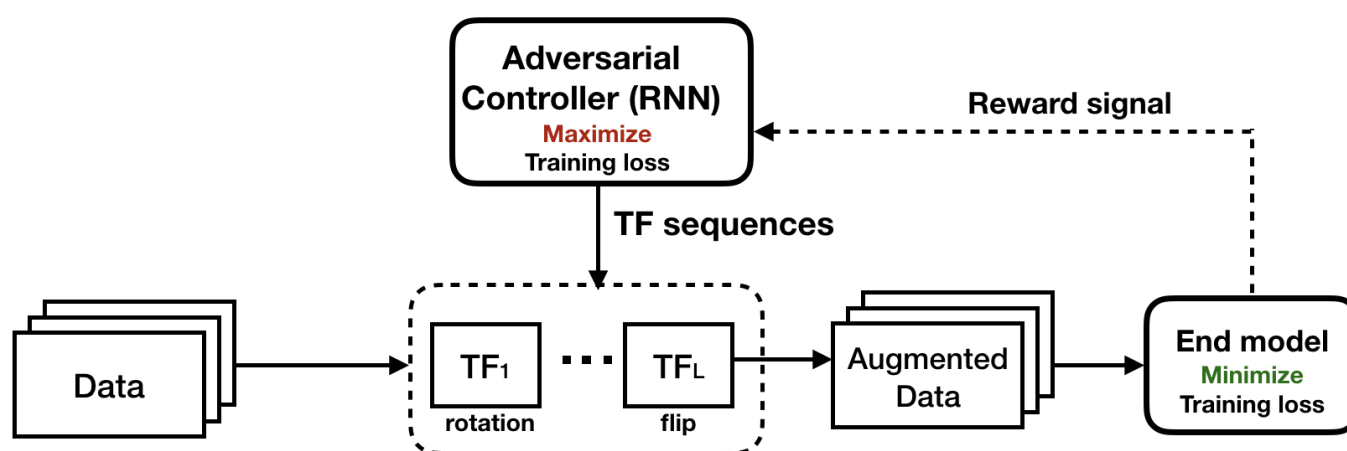


Figure 4. Adversarial AutoAugment training framework (Zhang et al. 2019) is formulated as an adversarial min-max game.

Summary

In this blog post, we have described a few recent practical methods that can automatically learn augmentation policy, showing promise to replace human-designed data augmentations. TANDA proposes a method to learn data augmentations as sequences of Transformation Functions (TFs) provided by users. Using a similar framework, AutoAugment demonstrated state-of-the-art performance using learned augmentation, although the search process can be computationally expensive. Subsequent works RandAugment and Adversarial AutoAugment have been proposed to reduce the computational cost, establishing state-of-the-art performance on image classification benchmarks.

Links:

- **TANDA**: <https://github.com/HazyResearch/tanda> (<https://github.com/HazyResearch/tanda>)
- **AutoAugment**: <https://github.com/tensorflow/models/tree/master/research/autoaugment> (<https://github.com/tensorflow/models/tree/master/research/autoaugment>)

- **RandAugment**: <https://github.com/ildoonet/pytorch-randaugment>
(<https://github.com/ildoonet/pytorch-randaugment>)
- **Adversarial AutoAugment**: <https://arxiv.org/abs/1912.11188>
(<https://arxiv.org/abs/1912.11188>)
- We will do something that's faster soon!

Related Posts

- Automating the Art of Data Augmentation (Part I: Overview)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part1/>)
- Automating the Art of Data Augmentation (Part III: Theory)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part3/>)
- Automating the Art of Data Augmentation (Part IV: New Direction)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part4/>)

← **PREVIOUS POST** (/2020-02-26-DATA-AUGMENTATION-PART1/)

NEXT POST → (/2020-02-26-DATA-AUGMENTATION-PART3/)



(/feed.xml)



(<https://github.com/HazyResearch>)

HazyResearch • 2020 • hazyresearch.cs.stanford.edu (<http://hazyresearch.cs.stanford.edu/>)

Theme by beautiful-jekyll (<https://deanattali.com/beautiful-jekyll/>)

Automating the Art of Data Augmentation

Part III Theory

Posted on February 26, 2020

Despite the rapid progress of practical data augmentation techniques, precisely understanding their benefits remains a mystery. In the field of deep learning, data augmentation is commonly understood to act as a regularizer by increasing the number of data points and constraining the model (Goodfellow et al. 2016, Zhang et al. 2017). The folklore wisdom behind data augmentation is that adding more labeled data improves generalization, i.e. the performance of the trained model on unseen test data (Shorten & Khoshgoftaar, 2019). However, even for simpler models, it is not well-understood how training on augmented data affects the learning process, the parameters, and the decision surface of the resulting model. This is exacerbated by the fact that data augmentation is performed in diverse ways in modern machine learning pipelines, for different tasks and domains, thus precluding a general model of transformation.

How can we theoretically characterize and understand the effect various data augmentations used in practice? In this blog post, we describe recent research that shed light on this question.

Understanding Data Augmentation through the Lens of Kernel Methods

Inspired by the impressive performance of learnable data augmentations, Dao et al. 2019 (<http://proceedings.mlr.press/v97/dao19b/dao19b.pdf>) developed a theoretical framework which underscores the connection between augmentation and kernel theory. Built on the connection, Dao et al. 2019 further show that a kernel classifier on augmented data has certain invariance and regularization effect, which we explain below.

Data Augmentation as a Kernel

Our work (Dao et al. 2019) models data augmentation as a Markov Chain, in which augmentation is performed via a random sequence of transformations—akin to how data augmentation is performed in practice. We define Markov Chain as follows: (1) we choose a data point (say an image) uniformly at random from the training dataset; (2) we define a transition from one data point to another by applying a transformation chosen from a predefined set randomly according to some fixed probabilities; (3) to avoid composing too many transformations, with some probability we terminate the Markov Chain by staying at the current data point.

We show that such the effect of applying the Markov Chain on the training dataset (combined with a k-nearest neighbor classifier) is akin to averaging the features of the transformed data resulting from various transformations. The results have two important implications:

- Kernels appear naturally in relation to data augmentation, even when we do not start with a kernel classifier. This underscores the connection between augmentation and kernels even with complicated compositional models.
- Data augmentation—a process that produces synthetic training examples from the raw data—can be understood more directly based on its effect on downstream components in the learning process, such as the features of the original data and the resulting learned

model. We make this link more explicit in the section below.

Effects of Augmentation: Invariance and Regularization

Built on the connection between kernel theory and data augmentation, Dao et al. 2019 show that a kernel classifier on augmented data approximately decomposes into two components: (i) an averaged version of the transformed features, and (ii) a data-dependent variance regularization term. This suggests a more nuanced explanation of data augmentation—namely, that it improves generalization both by inducing invariance and by reducing model complexity. Dao et al. 2019 validate the quality of our approximation empirically, and draw connections to other generalization-improving techniques, including recent work in invariant learning (Zhao et al., 2017; Mroueh et al., 2015; Raj et al., 2017) and robust optimization (Namkoong & Duchi, 2017).

Concurrent to the above work that shows the feature averaging effect of data augmentation on kernel methods, Chen et al. 2019 uses group theory in an empirical risk minimization setting to capture the intuition that a neural net classifier is invariant to compositions of transformations over images.

Understanding Data Augmentation through a Simplified Linear Setting

One limitation of the above works is that it is challenging to pin down the effect of applying a particular transformation on the resulting kernel. Furthermore, it is not yet clear how to apply data augmentation efficiently on kernel methods to get comparable performance to neural nets. In our recent work, we consider a simpler linear setting that is capable of modeling a wide range of linear transformations commonly used in image augmentation to gain insight.

We consider three categories of transformation functions.

- *Label-invariant base transformation* functions such as rotation and horizontal flip;
- *Label-mixing transformations* including mixup (Zhang et al., 2017; Inoue, 2018), which produce new data by randomly interpolating the features of two data points (e.g. a cat and a dog) as well as the associated labels;
- *Compositions of label-invariant transformations* such as random cropping and rotation followed by horizontal flipping (as in TANDA and AutoAugment).

What are the main theoretical takeaways?

We offer the following theoretical insights by considering a conceptually simple over-parametrized linear model, where the training data lies in a low-dimensional subspace.

- Our first insight is that *label-invariant transformations* can add new information to the training data. In our conceptualized model, we show that the estimation error of the ridge estimator can be reduced by adding new points that are outside the span of the training data.
- Our second insight is that mixup (Zhang et al. 2017) can play an effect of regularization through shrinking the weight of the training data relative to the L2 regularization term on the training data. We validate this insight on MNIST by showing that mixing same-class digits can reduce the variance (instability) of the predictions made by the trained model.

- Finally, for *compositions of label-invariant transformations*, we can show that they have an effect akin to adding new information as a corollary of combining multiple label-invariant transformations. However, the effect can be either positive or negative.

Theory on Adversarial Data Augmentation

Besides improving generalization performance, data augmentation has been used to improve the robustness of model training. Rajput et al. 2019 analyze the robustness that DA begets by quantifying the margin that DA enforces on empirical risk minimizers. Xie et al. 2019 provide precise conditions under which data augmentation hurts test accuracy for minimum norm estimators in linear regression. To mitigate the failure modes of augmentation, the authors introduce X-regularization, which uses unlabeled data to regularize the parameters towards the non-augmented estimate.

From Theory to Practice: Establishing New State-of-the-art Performance

One insight from our investigation is that different (compositions of) transformations show very different end performance. For RandAugment, since the algorithm does random sampling over the space of transformations, the final outcome is kind of like averaging the effect of each transformation. On the other hand, this does not make use of the fact that certain transformations are better performing than others.

In our new algorithm, we propose an uncertainty-based random sampling scheme which, among the transformed data points, picks those with the highest losses, i.e. those “providing the most information” (see Figure 1). We find that there is a large variance among the performance of different transformations and their compositions. Unlike RandAugment (Cubuk et al., 2019), which averages the effect of all transformations, the idea behind our sampling scheme is to better select the transformations that can improve the accuracy of the trained model more than others.

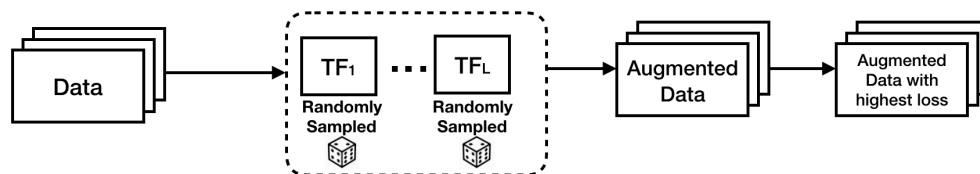


Figure 1: Uncertainty-based random sampling scheme for data augmentation. Each transformation function is randomly sampled from a set of pre-specified operations. We select among the transformed data points with highest loss for end model training.

We show that our proposed scheme improves performance on a wide range of datasets and models. Our sampling scheme achieves higher accuracy by finding more useful transformations compared to RandAugment on three different CNN architectures. For example, our method outperforms RandAugment by **0.59%** on CIFAR-10 and 1.24% on CIFAR-100 using Wide-ResNet-28-10. Our method also achieves comparable test accuracy to a state-of-the-art method, Adversarial AutoAugment (*Adv. AA*, Zhang et al. 2020). Since we only need to do inference on the augmented data without needing to train an additional adversarial network, our method reduces the cost of training the model by **5x** compared to *Adv. AA*.

Using Our Approach


For more information, please check out our Notebook (to be released soon)!

Related Posts

- Automating the Art of Data Augmentation (Part I: Overview)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part1/>)
- Automating the Art of Data Augmentation (Part II: Practical Methods)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part2/>)
- Automating the Art of Data Augmentation (Part IV: New Direction)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part4/>)

← **PREVIOUS POST** ([HTTP://HAZYRESEARCH.CS.STANFORD.EDU/2020-02-26-DATA-AUGMENTATION-PART2/](http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part2/))

NEXT POST → ([HTTP://HAZYRESEARCH.CS.STANFORD.EDU/2020-02-26-DATA-AUGMENTATION-PART4/](http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part4/))

 (<http://hazyresearch.cs.stanford.edu/feed.xml>)  (<https://github.com/HazyResearch>)

HazyResearch • 2020 • hazyresearch.cs.stanford.edu (<http://hazyresearch.cs.stanford.edu/>)

Theme by beautiful-jekyll (<https://deanattali.com/beautiful-jekyll/>)

Automating the Art of Data Augmentation

Part IV New Direction

Karan Goel (<https://krandiash.github.io/>), Albert Gu (<http://web.stanford.edu/~albertgu/>), Sharon Li (<http://www.yixuanli.net/~albertgu/>) and Chris Ré (<https://cs.stanford.edu/people/chismre/>)

Posted on February 26, 2020

Data augmentation techniques have proven powerful in building machine learning models in applications such as image and text classification. However, most of the machine learning research today is still carried out solving fixed tasks. In the real world, machine learning models in deployment can fail in various ways, due to unanticipated changes in data. This raises the concerning question of how we can move from model building to model maintenance in an adaptive manner. In this post, we'd like to shed some light on this question through the lens of *model patching*—the first framework that exploits data augmentation to mitigate the performance issue of a flawed model in deployment.

What is Model Patching?

Model patching enables automating the process of model maintenance and improvement when a deployed model exhibits flaws. Model patching is becoming a late breaking area that would alleviate the major problem in safety-critical systems, including healthcare (e.g. improving models to produce MRI scans free of artifact (<https://ai.facebook.com/blog/fastmri-leverages-adversarial-learning-to-remove-image-artifacts/>)) and autonomous driving (e.g. improving perception models that may have poor performance on irregular objects or road conditions).

To give a concrete example, in skin cancer detection, researchers have shown that standard classifiers have drastically different performance on two subgroups of the cancerous class, due to the classifier's association between colorful bandages with benign images (see Figure 1, left). This subgroup performance gap has also been studied in parallel research from our group (Oakden-Rayner et al., 2019 (<https://arxiv.org/abs/1909.12475>)), and arises due to classifier's reliance on subgroup-specific features, e.g. colorful bandages.

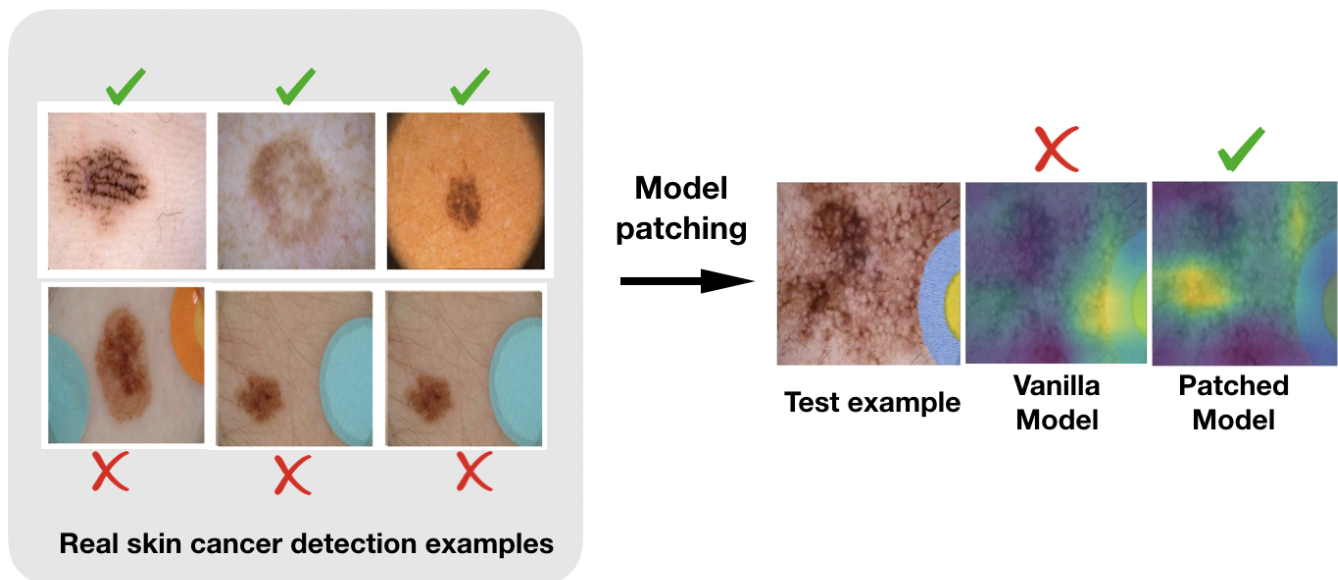


Figure 1: A standard model trained on a skin cancer dataset exhibits a subgroup performance gap between images of malignant cancers with and without colored bandages. GradCAM illustrates that the vanilla model spuriously associates the colored spot with benign skin lesions. With model patching, the malignancy is predicted correctly for both subgroups.

In order to fix such flaws in a deployed model, domain experts have to resort to manual data cleaning to erase the differences between subgroups, e.g. removing markings on skin cancer data with Photoshop (Winkler et al. 2019), and retrain the model with the modified data. This can be extremely laborious!

How can we automate this tedious process for domain experts? Can we somehow learn transformations that allow augmenting examples to balance population among groups in a prescribed way? This is exactly what we are addressing through this new framework of model patching.

How Does Model Patching Work?

We start by presenting the conceptual framework of model patching, which consists of two stages (as shown in Figure 2).

- **Learn inter-subgroup transformations** between different subgroups. These transformations are class-preserving maps that allow semantically changing a datapoint's subgroup identity (e.g. add or remove colorful bandages).
- **Retrain to patch the model** with augmented data, encouraging the classifier to be robust to their variations.

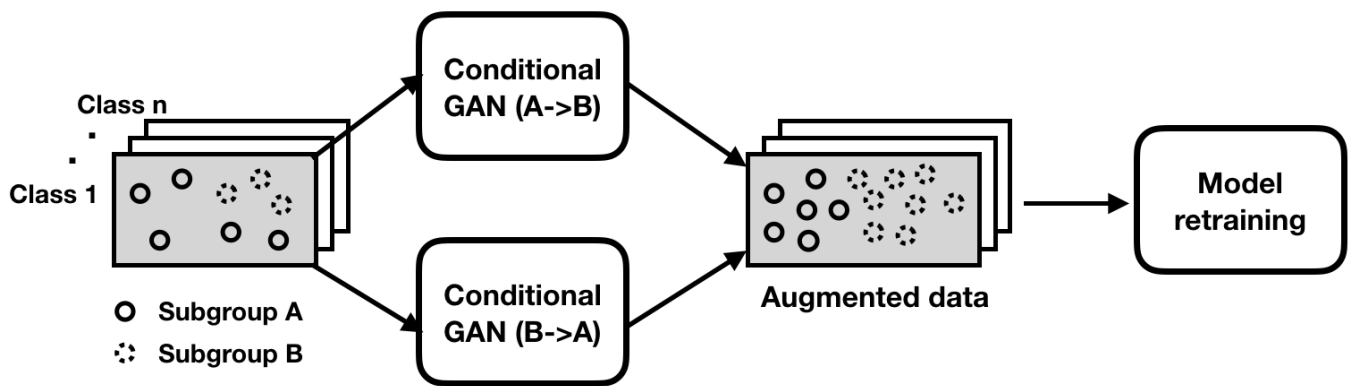


Figure 2: Model Patching framework with data augmentation. The highlighted box contains samples from a class with differing performance between subgroups A and B. Conditional generative models are trained to transform examples from one subgroup to another (A->B and B->A) respectively.

CLAMP: An Instantiation of Model Patching

Now we describe **Class-conditional Learned Augmentations for Model Patching (CLAMP)**, an instantiation of our first end-to-end model patching framework.

CLAMP instantiates the first stage of model patching with CycleGANs (Zhu et al. 2017). The generative models are trained class-conditionally to transfer examples from one subgroup to another. The transformation functions learned are semantically meaningful, intended to capture the salient features between the subgroups of interest. This contrasts with previous data augmentation approaches where the generator learns to compose augmentations via generic transformation functions.

Perhaps the major challenge of model patching lies in the second stage—model retraining. While seemingly straightforward, solutions that directly use augmented data for training can fail, since the augmentations will typically introduce artifacts that may distort class information. To mitigate this issue, we introduce a novel *subgroup consistency regularizer* which provably forces the classifier to (1) preserve the class information; and (2) become invariant to subgroup-specific features.

We use Figure 3 to illustrate how our regularizer works. For a given data point x , we apply the transformations learned through CycleGAN and obtain augmented examples in subgroup A and B respectively. Our regularizer L_t encourages the prediction on x to be similar to the average prediction of CycleGAN-translated examples, and L_s encourages the predictions on corresponding augmented examples to be consistent with each other. In our training objective, this can be formulated in terms of KL-divergence in the output space.

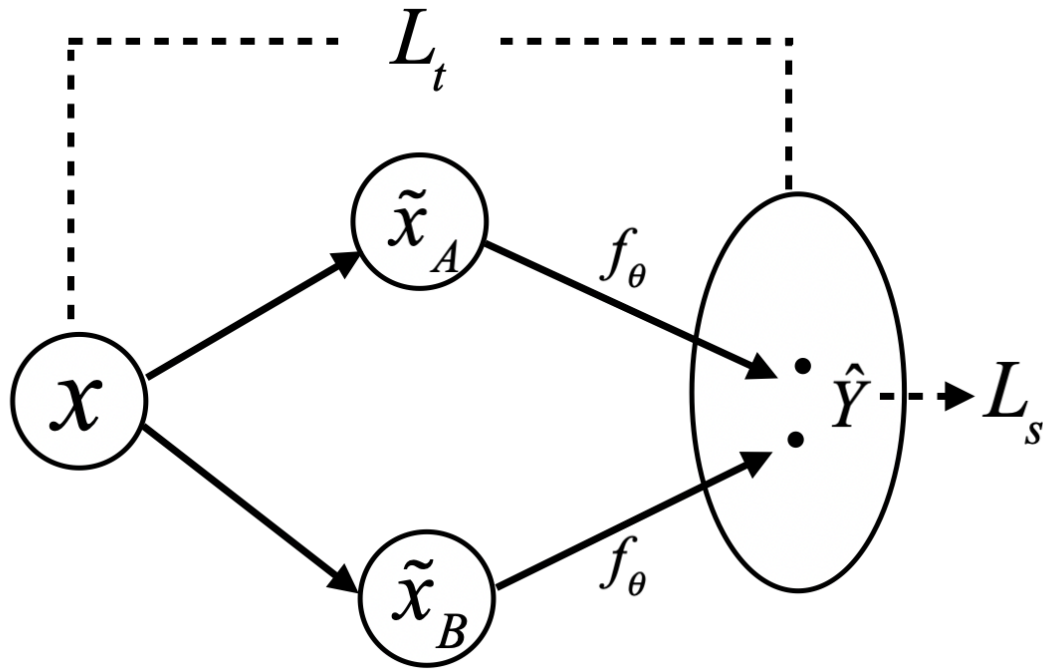


Figure 3. Illustration of subgroup consistency regularizer.

We further combine this regularizer with a robust training objective that is inspired by recent work of Group Distributionally Robust Optimization (GDRO, Sagawa et al. 2019). We extend GDRO to a class-conditional training objective that jointly optimizes for the worst-subgroup performance in each class. Taken together, these steps comprise our full method Class-conditional Learned Augmentations for Model Patching (CLAMP), the first end-to-end instantiation of our model patching framework.

CLAMP’s Performance

Does our model patching method produce better end classifier results than vanilla ERM and robust training baselines? To answer this question, we evaluate CLAMP on various benchmark datasets including MNIST (<http://yann.lecun.com/exdb/mnist/>), CelebA (<http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>), Waterbirds (<https://arxiv.org/abs/1911.08731>), and a real-world skin cancer dataset ISIC (<https://www.isic-archive.com/>). For each model, we measure overall average accuracy, as well as robust accuracy on the worst-performing subgroup.

On several benchmark datasets, Table 1 shows that CLAMP can improve both the aggregate and robust accuracy, mitigating the tradeoff between the two that ERM and GDRO experience. Furthermore, CLAMP is able to balance the performance of subgroups within each class, reducing the performance gap by up to **24x**. On real-world clinical dataset ISIC, CLAMP improves robust

accuracy by **11.7%** compared to the robust training baseline. Through visualization, we also show in Figure 1 that CLAMP successfully removes the model’s reliance on the spurious feature (colorful bandages), shifting its attention to the skin lesion—true feature of interest.

Dataset	Method	Subgroup		Y		Aggregate	Robust	Subgroup Gap (%)	
		Acc. (%)	Acc. (%)	Z	Z	Acc. (%)	Acc. (%)	Y	Y
MNIST-		even	even	odd	odd			even	odd
		clean	zigzag	clean	zigzag				
	ERM	86.96	73.51	71.47	75.21	76.75 (1.60)	71.47 (1.50)	13.45	3.73
	GDRO	98.10	93.31	96.82	97.15	96.35 (0.49)	93.31 (1.30)	4.79	0.79
	CLAMP	98.85	97.89	97.98	97.87	97.55 (0.46)	97.77 (0.42)	0.96	0.17
CelebA- Undersampled		non-blonde	non-blonde	blonde	blonde			non-blonde	blonde
		female	male	female	male				
	ERM	81.09	98.08	98.13	60.04	88.26 (1.88)	62.22 (6.83)	16.99	38.09
	GDRO	89.26	92.24	94.08	82.20	90.91 (0.78)	82.20 (3.13)	2.98	11.88
	CLAMP	92.15	93.73	91.13	83.53	92.90 (0.35)	83.90 (1.31)	1.83	8.07
Waterbirds		landbird	landbird	waterbird	waterbird			landbird	waterbird
		land	water	land	water				
	ERM	98.92	75.12	72.71	94.95	86.31 (0.39)	72.71 (2.36)	23.80	22.24
	GDRO	94.46	83.81	88.19	92.36	89.39 (0.19)	83.81 (0.39)	10.65	4.17
	CLAMP	90.84	90.40	89.69	89.58	90.89 (0.87)	89.12 (0.36)	0.43	1.04

Table 1. A comparison between CLAMP and other methods. For each dataset, Y denotes class label and Z denotes subgroup label. Evaluation metrics include robust accuracy (i.e. worst-performance among all subgroups), aggregate accuracy and the subgroup performance gap. Results are averaged over 3 trials (one standard deviation is indicated in parentheses).

Our results suggest that the model patching framework is a promising direction for automating the process of model maintenance. We envision that model patching can be widely useful for many other domain applications. If you are intrigued by the latest research on model patching, please follow our Hazy Research repository on Github where the code will be released soon.

Our work is still in progress. If you have any feedback, we’d like to hear from you!

Related Posts

- Automating the Art of Data Augmentation (Part I: Overview)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part1/>)
- Automating the Art of Data Augmentation (Part II: Practical Methods)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part2/>)
- Automating the Art of Data Augmentation (Part III: Theory)
(<http://hazyresearch.cs.stanford.edu/2020-02-26-data-augmentation-part3/>)

← PREVIOUS POST (/2020-02-26-DATA-AUGMENTATION-PART3/)

ADVERSARIAL AUTOAUGMENT

Xinyu Zhang

Huawei

zhangxinyu10@huawei.com

Qiang Wang

Huawei

wangqiang168@huawei.com

Jian Zhang

Huawei

zhangjian157@huawei.com

Zhao Zhong

Huawei

zorro.zhongzhao@huawei.com

ABSTRACT

Data augmentation (DA) has been widely utilized to improve generalization in training deep neural networks. Recently, human-designed data augmentation has been gradually replaced by automatically learned augmentation policy. Through finding the best policy in well-designed search space of data augmentation, AutoAugment (Cubuk et al., 2018) can significantly improve validation accuracy on image classification tasks. However, this approach is not computationally practical for large-scale problems. In this paper, we develop an adversarial method to arrive at a computationally-affordable solution called *Adversarial AutoAugment*, which can simultaneously optimize target related object and augmentation policy search loss. The augmentation policy network attempts to increase the training loss of a target network through generating adversarial augmentation policies, while the target network can learn more robust features from harder examples to improve the generalization. In contrast to prior work, we reuse the computation in target network training for policy evaluation, and dispense with the retraining of the target network. Compared to AutoAugment, this leads to about $12\times$ reduction in computing cost and $11\times$ shortening in time overhead on ImageNet. We show experimental results of our approach on CIFAR-10/CIFAR-100, ImageNet, and demonstrate significant performance improvements over state-of-the-art. On CIFAR-10, we achieve a top-1 test error of 1.36% , which is the currently best performing single model. On ImageNet, we achieve a leading performance of top-1 accuracy 79.40% on ResNet-50 and 80.00% on ResNet-50-D without extra data.

1 INTRODUCTION

Massive amount of data have promoted the great success of deep learning in academia and industry. The performance of deep neural networks (DNNs) would be improved substantially when more supervised data is available or better data augmentation method is adapted. Data augmentation such as rotation, flipping, cropping, *etc.*, is a powerful technique to increase the amount and diversity of data. Experiments show that the generalization of a neural network can be efficiently improved through manually designing data augmentation policies. However, this needs lots of knowledge of human expert, and sometimes shows the weak transferability across different tasks and datasets in practical applications. Inspired by neural architecture search (NAS) (Zoph & Le, 2016; Zoph et al., 2017; Zhong et al., 2018a;b; Guo et al., 2018), a reinforcement learning (RL) (Williams, 1992) method called AutoAugment is proposed by Cubuk et al. (2018), which can automatically learn the augmentation policy from data and provide an exciting performance improvement on image classification tasks. However, the computing cost is huge for training and evaluating thousands of sampled policies in the search process. Although proxy tasks, i.e., smaller models and reduced datasets, are taken to accelerate the searching process, tens of thousands of GPU-hours of consumption are still required. In addition, these data augmentation policies optimized on proxy tasks are not guaranteed to be optimal on the target task, and the fixed augmentation policy is also sub-optimal for the whole training process.

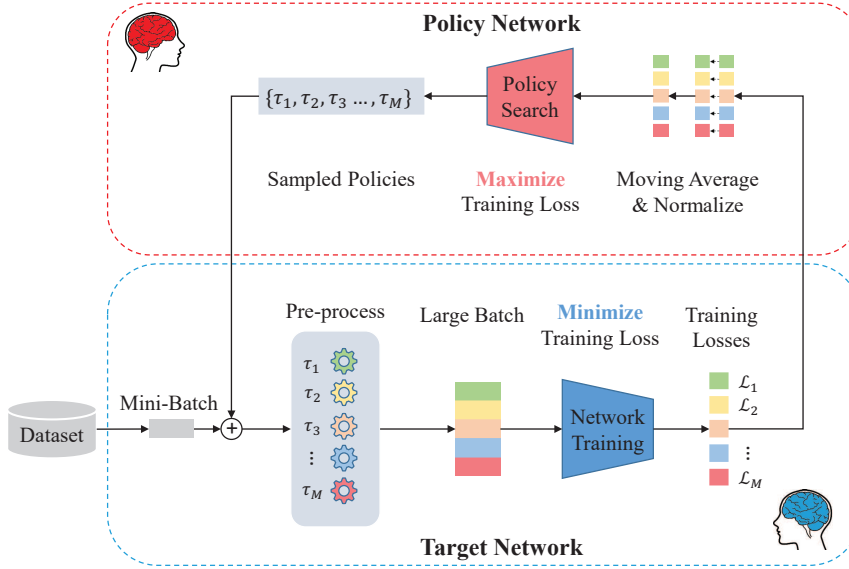


Figure 1: The overview of our proposed method. We formulate it as a Min-Max game. The data of each batch is augmented by multiple pre-processing components with sampled policies $\{\tau_1, \tau_2, \dots, \tau_M\}$, respectively. Then, a target network is trained to minimize the loss of a large batch, which is formed by multiple augmented instances of the input batch. We extract the training losses of a target network corresponding to different augmentation policies as the reward signal. Finally, the augmentation policy network is trained with the guideline of the processed reward signal, and aims to maximize the training loss of the target network through generating adversarial policies.

In this paper, we propose an efficient data augmentation method to address the problems mentioned above, which can directly search the best augmentation policy on the full dataset during training a target network, as shown in Figure 1. We first organize the network training and augmentation policy search in an adversarial and online manner. The augmentation policy is dynamically changed along with the training state of the target network, rather than fixed throughout the whole training process like normal AutoAugment (Cubuk et al., 2018). Due to reusing the computation in policy evaluation and dispensing with the retraining of the target network, the computing cost and time overhead are extremely reduced. Then, the augmentation policy network is taken as an adversary to explore the weakness of the target network. We augment the data of each mini-batch with various adversarial policies in parallel, rather than the same data augmentation taken in batch augmentation (BA) (Hoffer et al., 2019). Then, several augmented instances of each mini-batch are formed into a large batch for target network learning. As an indicator of the hardness of augmentation policies, the training losses of the target network are used to guide the policy network to generate more aggressive and efficient policies based on REINFORCE algorithm (Williams, 1992). Through adversarial learning, we can train the target network more efficiently and robustly.

The contributions can be summarized as follows:

- Our method can directly learn augmentation policies on target tasks, i.e., target networks and full datasets, with a quite low computing cost and time overhead. The direct policy search avoids the performance degradation caused by the policy transfer from proxy tasks to target tasks.
- We propose an adversarial framework to jointly optimize target network training and augmentation policy search. The harder samples augmented by adversarial policies are constantly fed into the target network to promote robust feature learning. Hence, the generalization of the target network can be significantly improved.
- The experiment results show that our proposed method outperforms previous augmentation methods. For instance, we achieve a top-1 test error of 1.36% with PyramidNet+ShakeDrop (Yamada et al., 2018) on CIFAR-10, which is the state-of-the-art performance. On ImageNet, we improve the top-1 accuracy of ResNet-50 (He et al., 2016) from 76.3% to 79.4% without extra data, which is even 1.77% better than AutoAugment (Cubuk et al., 2018).

2 RELATED WORK

Common data augmentation, which can generate extra samples by some label-preserved transformations, is usually used to increase the size of datasets and improve the generalization of networks, such as on MINST, CIFAR-10 and ImageNet (Krizhevsky et al., 2012; Wan et al., 2013; Szegedy et al., 2015). However, human-designed augmentation policies are specified for different datasets. For example, flipping, the widely used transformation on CIFAR-10/CIFAR-100 and ImageNet, is not suitable for MINST, which will destroy the property of original samples.

Hence, several works (Lemley et al., 2017; Cubuk et al., 2018; Lin et al., 2019; Ho et al., 2019) have attempted to automatically learn data augmentation policies. Lemley et al. (2017) propose a method called Smart Augmentation, which merges two or more samples of a class to improve the generalization of a target network. The result also indicates that an augmentation network can be learned when a target network is being training. Through well designing the search space of data augmentation policies, AutoAugment (Cubuk et al., 2018) takes a recurrent neural network (RNN) as a sample controller to find the best data augmentation policy for a selected dataset. To reduce the computing cost, the augmentation policy search is performed on proxy tasks. Population based augmentation (PBA) (Ho et al., 2019) replaces the fixed augmentation policy with a dynamic schedule of augmentation policy along with the training process, which is mostly related to our work. Inspired by population based training (PBT) (Jaderberg et al., 2017), the augmentation policy search problem in PBA is modeled as a process of hyperparameter schedule learning. However, the augmentation schedule learning is still performed on proxy tasks. The learned policy schedule should be manually adjusted when the training process of a target network is non-matched with proxy tasks.

Another related topic is Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), which has recently attracted lots of research attention due to its fascinating performance, and also been used to enlarge datasets through directly synthesizing new images (Tran et al., 2017; Perez & Wang, 2017; Antoniou et al., 2017; Gurumurthy et al., 2017; Frid-Adar et al., 2018). Although we formulate our proposed method as a Min-Max game, there exists an obvious difference with traditional GANs. We want to find the best augmentation policy to perform image transformation along with the training process, rather than synthesize new images. Peng et al. (2018) also take such an idea to optimize the training process of a target network in human pose estimation.

3 METHOD

In this section, we present the implementation of *Adversarial AutoAugment*. First, the motivation for the adversarial relation between network learning and augmentation policy is discussed. Then, we introduce the search space with the dynamic augmentation policy. Finally, the joint framework for network training and augmentation policy search is presented in detail.

3.1 MOTIVATIONS

Although some human-designed data augmentations have been used in the training of DNNs, such as randomly cropping and horizontally flipping on CIFAR-10/CIFAR-100 and ImageNet, limited randomness will make it very difficult to generate effective samples at the tail end of the training. To struggle with the problem, more randomness about image transformation is introduced into the search space of AutoAugment (Cubuk et al., 2018) (described in Section 3.2). However, the learned policy is fixed for the entire training process. All of possible instances of each example will be send to the target network repeatedly, which still results in an inevitable overfitting in a long-epoch training. This phenomenon indicates that the learned policy is not adaptive to the training process of a target network, especially found on proxy tasks. Hence, the dynamic and adversarial augmentation policy with the training process is considered as the crucial feature in our search space.

Another consideration is how to improve the efficiency of the policy search. In AutoAugment (Cubuk et al., 2018), to evaluate the performance of augmentation policies, a lot of child models should be trained from scratch nearly to convergence. The computation in training and evaluating the performance of different sampled policies can not be reused, which leads to huge waste of computation resources. In this paper, we propose a computing-efficient policy search framework through reusing prior computation in policy evaluation. Only one target network is used to evaluate the performance of different policies with the help of the training losses of corresponding augmented

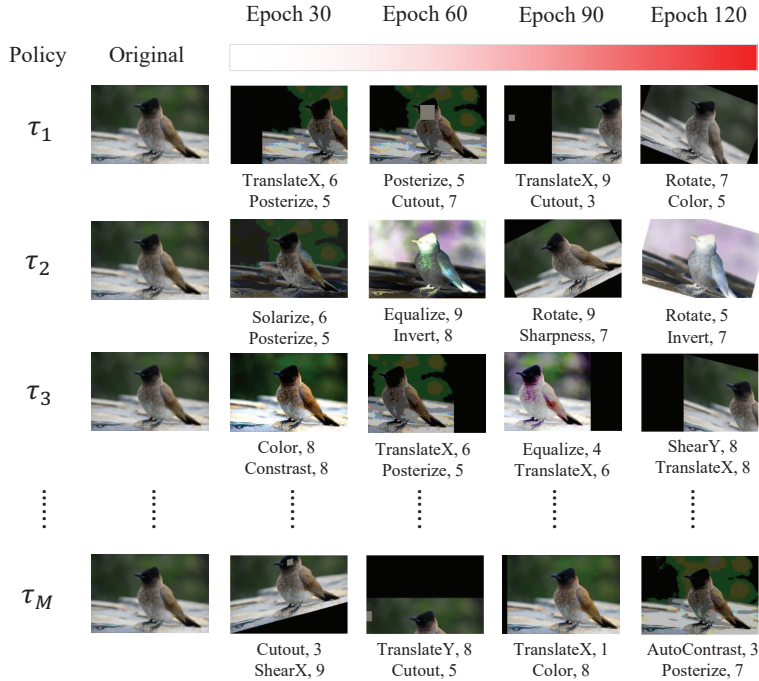


Figure 2: An example of dynamic augmentation policies learned with ResNet-50 on ImageNet. With the training process of the target network, harder augmentation policies are sampled to combat overfitting. Intuitively, more geometric transformations, such as TranslateX, ShearY and Rotate, are picked in our sampled policies, which is obviously different from AutoAugment (Cubuk et al., 2018) concentrating on color-based transformations.

instances. The augmentation policy network is learned from the intermediate state of the target network, which makes generated augmentation policies more aggressive and adaptive. On the contrary, to combat harder examples augmented by adversarial policies, the target network has to learn more robust features, which makes the training more efficiently.

3.2 SEARCH SPACE

In this paper, the basic structure of the search space of AutoAugment (Cubuk et al., 2018) is reserved. An augmentation policy is defined as that it is composed by 5 sub-policies, each sub-policy contains two image operations to be applied orderly, each operation has two corresponding parameters, i.e., the probability and magnitude of the operation. Finally, the 5 best policies are concatenated to form a single policy with 25 sub-policies. For each image in a mini-batch, only one sub-policy will be randomly selected to be applied. To compare with AutoAugment (Cubuk et al., 2018) conveniently, we just slightly modify the search space with removing the probability of each operation. This is because that we think the stochasticity of an operation with a probability requires a certain epochs to take effect, which will detain the feedback of the intermediate state of the target network. There are totally 16 image operations in our search space, including ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, Cutout (Devries & Taylor, 2017) and Sample Pairing (Inoue, 2018). The range of the magnitude is also discretized uniformly into 10 values. *To guarantee the convergence during adversarial learning, the magnitude of all the operations are set in a moderate range.*¹ Besides, the randomness during the training process is introduced into our search space. Hence, the search space of the policy in each epoch has $|S| = (16 \times 10)^{10} \approx 1.1 \times 10^{22}$ possibilities. Considering the dynamic policy, the number of possible policies with the whole training process can be expressed as $|S|^{\#epochs}$. An example of dynamically learning the augmentation policy along with the training process is shown in Figure 2. We observe that the magnitude (an indication of difficulty) gradually increases with the training process.

¹The more details about the parameter setting please refer to AutoAugment (Cubuk et al., 2018).

3.3 ADVERSARIAL LEARNING

In this section, the adversarial framework of jointly optimizing network training and augmentation policy search is presented in detail. We use the augmentation policy network $\mathcal{A}(\cdot, \theta)$ as an adversary, which attempts to increase the training loss of the target network $\mathcal{F}(\cdot, \mathbf{w})$ through adversarial learning. The target network is trained by a large batch formed by multiple augmented instances of each batch to promote invariant learning (Salazar et al., 2018), and the losses of different augmentation policies applied on the same data are used to train the augmentation policy network by RL algorithm.

Considering the target network $\mathcal{F}(\cdot, \mathbf{w})$ with a loss function $\mathcal{L}[\mathcal{F}(\mathbf{x}, \mathbf{w}), \mathbf{y}]$, where each example is transformed by some random data augmentation $o(\cdot)$, the learning process of the target network can be defined as the following minimization problem

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \mathbb{E}_{\mathbf{x} \sim \Omega} \mathcal{L}[\mathcal{F}(o(\mathbf{x}), \mathbf{w}), \mathbf{y}], \quad (1)$$

where Ω is the training set, \mathbf{x} and \mathbf{y} are the input image and the corresponding label, respectively. The problem is usually solved by vanilla SGD with a learning rate η and batch size N , and the training procedure for each batch can be expressed as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{1}{N} \sum_{n=1}^N \nabla_{\mathbf{w}} \mathcal{L}[\mathcal{F}(o(x_n), \mathbf{w}), y_n]. \quad (2)$$

To improve the convergence performance of DNNs, more random and efficient data augmentation is performed under the help of the augmentation policy network. Hence, the minimization problem should be slightly modified as

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \mathbb{E}_{\mathbf{x} \sim \Omega} \mathbb{E}_{\tau \sim \mathcal{A}(\cdot, \theta)} \mathcal{L}[\mathcal{F}(\tau(\mathbf{x}), \mathbf{w}), \mathbf{y}], \quad (3)$$

where $\tau(\cdot)$ represents the augmentation policy generated by the network $\mathcal{A}(\cdot, \theta)$. Accordingly, the training rule can be rewritten as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{1}{M \cdot N} \sum_{m=1}^M \sum_{n=1}^N \nabla_{\mathbf{w}} \mathcal{L}[\mathcal{F}(\tau_m(x_n), \mathbf{w}), y_n], \quad (4)$$

where we introduce M different instances of each input example augmented by adversarial policies $\{\tau_1, \tau_2, \dots, \tau_M\}$. For convenience, we denote the training loss of a mini-batch corresponding to the augmentation policy τ_m as

$$\mathcal{L}_m = \frac{1}{N} \sum_{n=1}^N \mathcal{L}[\mathcal{F}(\tau_m(x_n), \mathbf{w}), y_n]. \quad (5)$$

Hence, we have an equivalent form of Equation 4

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{1}{M} \sum_{m=1}^M \nabla_{\mathbf{w}} \mathcal{L}_m. \quad (6)$$

Note that the training procedure can be regarded as a larger $N \cdot M$ batch training or an average over M instances of gradient computation without changing the learning rate, which will lead to a reduction of gradient variance and a faster convergence of the target network Hoffer et al. (2019). However, overfitting will also come. To overcome the problem, the augmentation policy network is designed to increase the training loss of the target network with harder augmentation policies. Therefore, we can mathematically express the object as the following maximization problem

$$\begin{aligned} \theta^* &= \arg \max_{\theta} J(\theta), \\ \text{where } J(\theta) &= \mathbb{E}_{\mathbf{x} \sim \Omega} \mathbb{E}_{\tau \sim \mathcal{A}(\cdot, \theta)} \mathcal{L}[\mathcal{F}(\tau(\mathbf{x}), \mathbf{w}), \mathbf{y}]. \end{aligned} \quad (7)$$

Similar to AutoAugment (Cubuk et al., 2018), the augmentation policy network is also implemented as a RNN shown in Figure 3. At each time step of the RNN controller, the softmax layer will predict

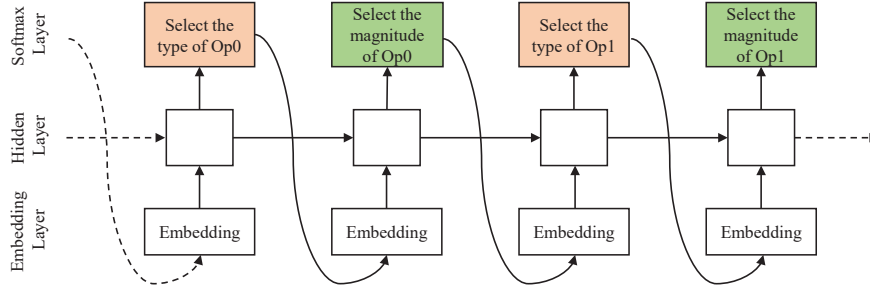


Figure 3: The basic architecture of the controller for generating a sub-policy, which consists of two operations with corresponding parameters, the type and magnitude of each operation. When a policy contains Q sub-policies, the basic architecture will be repeated Q times. Following the setting of AutoAugment (Cubuk et al., 2018), the number of sub-policies Q is set to 5 in this paper.

an action corresponding to a discrete parameter of a sub-policy, and then an embedding of the predicted action will be fed into the next time step. In our experiments, the RNN controller will predict 20 discrete parameters to form a whole policy.

However, there has a severe problem in jointly optimizing target network training and augmentation policy search. This is because that non-differentiable augmentation operations break gradient flow from the target network \mathcal{F} to the augmentation policy network \mathcal{A} (Wang et al., 2017; Peng et al., 2018). As an alternative approach, REINFORCE algorithm (Williams, 1992) is applied to optimize the augmentation policy network as

$$\begin{aligned}
 \nabla_{\theta} J(\theta) &= \nabla_{\theta} \mathbb{E}_{\mathbf{x} \sim \Omega} \mathbb{E}_{\tau \sim \mathcal{A}(\cdot, \theta)} \mathcal{L}[\mathcal{F}(\tau(\mathbf{x}), \mathbf{w}), \mathbf{y}] \\
 &\approx \sum_m \mathcal{L}_m \nabla_{\theta} p_m = \sum_m \mathcal{L}_m p_m \nabla_{\theta} \log p_m \\
 &= \mathbb{E}_{\tau \sim \mathcal{A}(\cdot, \theta)} \mathcal{L}_m \nabla_{\theta} \log p_m \\
 &\approx \frac{1}{M} \sum_{m=1}^M \mathcal{L}_m \nabla_{\theta} \log p_m,
 \end{aligned} \tag{8}$$

where p_m represents the probability of the policy τ_m . To reduce the variance of gradient $\nabla_{\theta} J(\theta)$, we replace the training loss of a mini-batch \mathcal{L}_m with $\hat{\mathcal{L}}_m$ a moving average over a certain mini-batches², and then normalize it among M instances as $\tilde{\mathcal{L}}_m$. Hence, the training procedure of the augmentation policy network can be expressed as

$$\begin{aligned}
 \nabla_{\theta} J(\theta) &\approx \frac{1}{M} \sum_{m=1}^M \tilde{\mathcal{L}}_m \nabla_{\theta} \log p_m, \\
 \theta_{e+1} &= \theta_e + \beta \frac{1}{M} \sum_{m=1}^M \tilde{\mathcal{L}}_m \nabla_{\theta} \log p_m,
 \end{aligned} \tag{9}$$

The adversarial learning of target network training and augmentation policy search is summarized as Algorithm 1.

4 EXPERIMENTS AND ANALYSIS

In this section, we first reveal the details of experiment settings. Then, we evaluate our proposed method on CIFAR-10/CIFAR-100, ImageNet, and compare it with previous methods. Results in Figure 4 show our method achieves the state-of-the-art performance with higher computing and time efficiency³.

²The length of the moving average is fixed to an epoch in our experiments.

³To clearly present the advantage of our proposed method, we normalize the performance of our method in the Figure 4, and the performance of AutoAugment is plotted accordingly.

Algorithm 1 Joint Training of Target Network and Augmentation Policy Network**Initialization:** target network $\mathcal{F}(\cdot, \mathbf{w})$, augmentation policy network $\mathcal{A}(\cdot, \boldsymbol{\theta})$ **Input:** input examples \mathbf{x} , corresponding labels \mathbf{y}

```

1: for  $1 \leq e \leq epochs$  do
2:   Initialize  $\hat{\mathcal{L}}_m = 0, \forall m \in \{1, 2, \dots, M\}$ ;
3:   Generate  $M$  policies with the probabilities  $\{p_1, p_2, \dots, p_M\}$ ;
4:   for  $1 \leq t \leq T$  do
5:     Augment each batch data with  $M$  generated policies, respectively;
6:     Update  $\mathbf{w}_{e,t+1}$  according to Equation 4;
7:     Update  $\hat{\mathcal{L}}_m$  through moving average,  $\forall m \in \{1, 2, \dots, M\}$ ;
8:     Collect  $\{\hat{\mathcal{L}}_1, \hat{\mathcal{L}}_2, \dots, \hat{\mathcal{L}}_M\}$ ;
9:     Normalize  $\hat{\mathcal{L}}_m$  among  $M$  instances as  $\tilde{\mathcal{L}}_m, \forall m \in \{1, 2, \dots, M\}$ ;
10:    Update  $\boldsymbol{\theta}_{e+1}$  via Equation 9;
11: Output  $\mathbf{w}^*, \boldsymbol{\theta}^*$ 

```

4.1 EXPERIMENT SETTINGS

The RNN controller is implemented as a one-layer LSTM (Hochreiter & Schmidhuber, 1997). We set the hidden size to 100, and the embedding size to 32. We use Adam optimizer (Kingma & Ba, 2015) with a initial learning rate 0.00035 to train the controller. To avoid unexpected rapid convergence, an entropy penalty of a weight of 0.00001 is applied. All the reported results are the mean of five runs with different initializations.

4.2 EXPERIMENTS ON CIFAR-10 AND CIFAR-100

CIFAR-10 dataset (Krizhevsky & Hinton, 2009) has totally 60000 images. The training and test sets have 50000 and 10000 images, respectively. Each image in size of 32×32 belongs to one of 10 classes. We evaluate our proposed method with the following models: Wide-ResNet-28-10 (Zagoruyko & Komodakis, 2016), Shake-Shake (26 2x32d) (Gastaldi, 2017), Shake-Shake (26 2x96d) (Gastaldi, 2017), Shake-Shake (26 2x112d) (Gastaldi, 2017), PyramidNet+ShakeDrop (Han et al., 2017; Yamada et al., 2018). All the models are trained on the full training set.

Training details: The Baseline is trained with the standard data augmentation, namely, randomly cropping a part of 32×32 from the padded image and horizontally flipping it with a probability of 0.5. The Cutout (Devries & Taylor, 2017) randomly select a 16×16 patch of each image, and then set the pixels of the selected patch to zeros. For our method, the searched policy is applied in addition to standard data augmentation and Cutout. For each image in the training process, standard data augmentation, the searched policy and Cutout are applied in sequence. For Wide-ResNet-28-10, the step learning rate (LR) schedule is adopted. The cosine LR schedule is adopted for the other models. More details about model hyperparameters are supplied in A.1.

Choice of M : To choose the optimal M , we select Wide-ResNet-28-10 as a target network, and evaluate the performance of our proposed method verse different M , where $M \in \{2, 4, 8, 16, 32\}$. From Figure 5, we can observe that the test accuracy of the model improves rapidly with the increase of M up to 8. The further increase of M does not bring a significant improvement. Therefore, to balance the performance and the computing cost, M is set to 8 in all the following experiments.

CIFAR-10 results: In Table 1, we report the test error of these models on CIFAR-10. For all of these models, our proposed method can achieve better performance compared to previous methods. We achieve 0.78% and 0.68% improvement on Wide-ResNet-28-10 compared to AutoAugment and PBA, respectively. We achieve a top-1 test error of 1.36% with PyramidNet+ShakeDrop, which is 0.1% better than the current state-of-the-art reported in Ho et al. (2019). As shown in Figure 6(a) and 6(b), we further visualize the probability distribution of the parameters of the augmentation policies learned with PyramidNet+ShakeDrop on CIFAR-10 over time. From Figure 6(a), we can find that the percentages of some operations, such as TranslateY, Rotate, Posterize, and SampleParing, gradually increase along with the training process. Meanwhile, more geometric transformations, such as TranslateX, TranslateY, and Rotate, are picked in the sampled augmentation policies, which

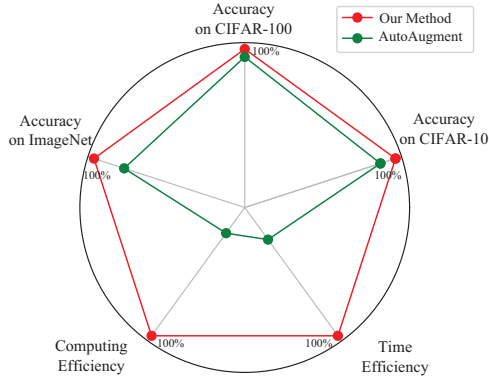


Figure 4: The Comparison of normalized performance between AutoAugment and our method. Please refer to the following tables for more details.

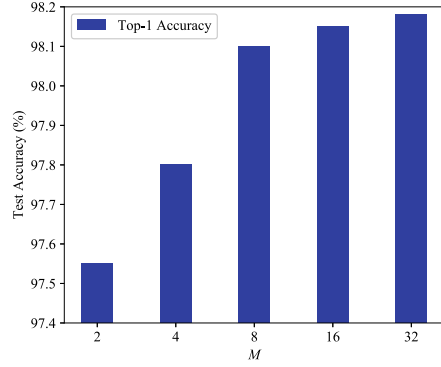


Figure 5: The Top-1 test accuracy of Wide-ResNet-28-10 on CIFAR-10 verse different M , where $M \in \{2, 4, 8, 16, 32\}$.

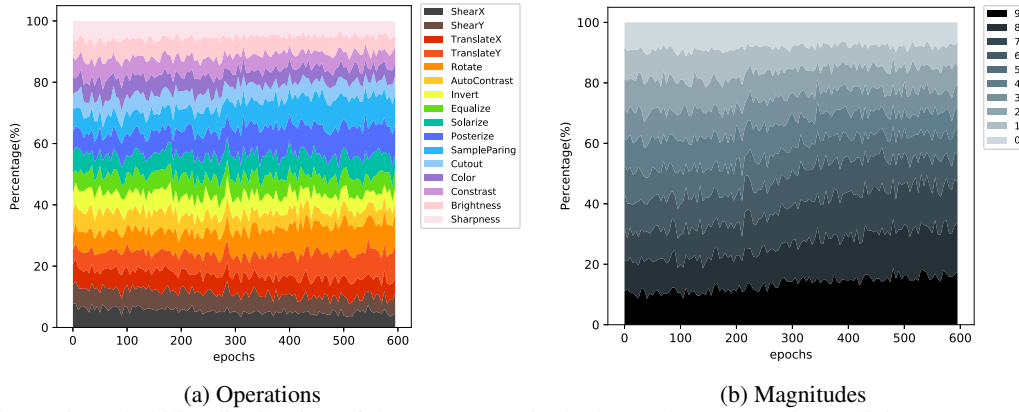


Figure 6: Probability distribution of the parameters in the learned augmentation policies on CIFAR-10 over time. The number in (b) represents the magnitude of one operation. Larger number stands for more dramatic image transformations. The probability distribution of each parameter is the mean of each five epochs.

is different from color-focused AutoAugment (Cubuk et al., 2018) on CIFAR-10. Figure 6(b) shows that large magnitudes gain higher percentages during training. However, at the tail of training, low magnitudes remain considerable percentages. This indicates that our method does not simply learn the transformations with the extremes of the allowed magnitudes to spoil the target network.

CIFAR-100 results: We also evaluate our proposed method on CIFAR-100, as shown in Table 2. As we can observe from the table, we also achieve the state-of-the-art performance on this dataset.

4.3 EXPERIMENTS ON IMAGENET

As a great challenge in image recognition, ImageNet dataset (Deng et al., 2009) has about 1.2 million training images and 50000 validation images with 1000 classes. In this section, we directly search the augmentation policy on the full training set and train ResNet-50 (He et al., 2016), ResNet-50-D (He et al., 2018) and ResNet-200 (He et al., 2016) from scratch.

Training details: For the baseline augmentation, we randomly resize and crop each input image to a size of 224×224 , and then horizontally flip it with a probability of 0.5. For AutoAugment (Cubuk et al., 2018) and our method, the baseline augmentation and the augmentation policy are both used for each image. The cosine LR schedule is adopted in the training process. The model hyperparameters on ImageNet is also detailed in A.1.

ImageNet results: The performance of our proposed method on ImageNet is presented in Table 3. It can be observed that we achieve a top-1 accuracy 79.40% on ResNet-50 without extra data. To

Table 1: Top-1 test error (%) on CIFAR-10. We replicate the results of Baseline, Cutout and AutoAugment methods from Cubuk et al. (2018), and the results of PBA from Ho et al. (2019) in all of our experiments.

Model	Baseline	Cutout	AutoAugment	PBA	Our Method
Wide-ResNet-28-10	3.87	3.08	2.68	2.58	1.90±0.15
Shake-Shake (26 2x32d)	3.55	3.02	2.47	2.54	2.36±0.10
Shake-Shake (26 2x96d)	2.86	2.56	1.99	2.03	1.85±0.12
Shake-Shake (26 2x112d)	2.82	2.57	1.89	2.03	1.78±0.05
PyramidNet+ShakeDrop	2.67	2.31	1.48	1.46	1.36±0.06

Table 2: Top-1 test error (%) on CIFAR-100.

Model	Baseline	Cutout	AutoAugment	PBA	Our Method
Wide-ResNet-28-10	18.80	18.41	17.09	16.73	15.49±0.18
Shake-Shake (26 2x96d)	17.05	16.00	14.28	15.31	14.10±0.15
PyramidNet+ShakeDrop	13.99	12.19	10.67	10.94	10.42±0.20

the best of our knowledge, this is the highest top-1 accuracy for ResNet-50 learned on ImageNet. Besides, we only replace the ResNet-50 architecture with ResNet-50-D, and achieve a consistent improvement with a top-1 accuracy of 80.00%.

Table 3: Top-1 / Top-5 test error (%) on ImageNet. Note that the result of ResNet-50-D is achieved only through substituting the architecture.

Model	Baseline	AutoAugment	PBA	Our Method
ResNet-50	23.69 / 6.92	22.37 / 6.18	-	20.60±0.15 / 5.53±0.05
ResNet-50-D	22.84 / 6.48	-	-	20.00±0.12 / 5.25±0.03
ResNet-200	21.52 / 5.85	20.00 / 4.90	-	18.68±0.18 / 4.70±0.05

4.4 ABLATION STUDY

To check the effect of each component in our proposed method, we report the test error of ResNet-50 on ImageNet the following augmentation methods in Table 4.

- **Baseline:** Training regularly with the standard data augmentation and step LR schedule.
- **Fixed:** Augmenting all the instances of each batch with the standard data augmentation fixed throughout the entire training process.
- **Random:** Augmenting all the instances of each batch with randomly and dynamically generated policies.
- **Ours:** Augmenting all the instances of each batch with adversarial policies sampled by the policy network along with the training process.

From the table, we can find that Fixed can achieve 0.99% error reduction compared to Baseline. This shows that a large-batch training with multiple augmented instances of each mini-batch can indeed improve the generalization of the model, which is consistent with the conclusion presented in Hoffer et al. (2019). In addition, the test error of Random is 1.02% better than Fixed. This indicates that augmenting batch with randomly generated policies can reduce overfitting in a certain extent. Furthermore, our method achieves the best test error of 20.60% through augmenting samples with adversarial policies. From the result, we can conclude that these policies generated by the policy network are more adaptive to the training process, and make the target network have to learn more robust features.

4.5 COMPUTING COST AND TIME OVERHEAD

Computing Cost: The computation in target network training is reused for policy evaluation. This makes the computing cost in policy search become negligible. Although there exists an increase of

Table 4: Top-1 test error (%) of ResNet-50 with different augmentation methods on ImageNet.

Method	Aug. Policy	Enlarge Batch	LR Schedule	Test Error
Baseline	standard	$M = 1$	step	23.69
Fixed	standard	$M = 8$	cosine	22.70
Random	random	$M = 8$	cosine	21.68
Ours	adversarial	$M = 8$	cosine	20.60

computing cost in target network training, the total computing cost in training one target network with augmentation policies is quite small compared to prior work.

Time Overhead: Since we just train one target network with a large batch distributedly and simultaneously, the time overhead of the large-batch training is equal to the regular training. Meanwhile, the joint optimization of target network training and augmentation policy search dispenses with the process of offline policy search and the retraining of a target network, which leads to a extreme time overhead reduction.

In Table 5, we take the training of ResNet-50 on ImageNet as an example to compare the computing cost and time overhead of our method and AutoAugment. From the table, we can find that our method is $12\times$ less computing cost and $11\times$ shorter time overhead than AutoAugment.

Table 5: The comparison of computing cost (GPU hours) and time overhead (days) in training ResNet-50 on ImageNet between AutoAugment and our method. The computing cost and time overhead are estimated on 64 NVIDIA Tesla V100s.

Method	Computing Cost			Time Overhead		
	Searching	Training	Total	Searching	Training	Total
AutoAugment	15000	160	15160	10	1	11
Our Method	~ 0	1280	1280	~ 0	1	1

4.6 TRANSFERABILITY ACROSS DATASETS AND ARCHITECTURES

To further show the higher efficiency of our method, the transferability of the learned augmentation policies is evaluated in this section. We first take a snapshot of the adversarial training process of ResNet-50 on ImageNet, and then directly use the learned dynamic augmentation policies to regularly train the following models: Wide-ResNet-28-10 on CIFAR-10/100, ResNet-50-D on ImageNet and ResNet200 on ImageNet. Table 6 presents the experimental results of the transferability. From the table, we can find that a competitive performance can be still achieved through direct **policy transfer**. This indicates that the learned augmentation policies transfer well across datasets and architectures. However, compared to the proposed method, the policy transfer results in an obvious performance degradation, especially the transfer across datasets.

Table 6: Top-1 test error (%) of the transfer of the augmentation policies learned with ResNet-50 on ImageNet.

Method	Dataset	AutoAugment	Our Method	Policy Transfer
Wide-ResNet-28-10	CIFAR-10	2.68	1.90	2.45 ± 0.13
Wide-ResNet-28-10	CIFAR-100	17.09	15.49	16.48 ± 0.15
ResNet-50-D	ImageNet	-	20.00	20.20 ± 0.05
ResNet-200	ImageNet	20.00	18.68	19.05 ± 0.10

5 CONCLUSION

In this paper, we introduce the idea of adversarial learning into automatic data augmentation. The policy network tries to combat the overfitting of the target network through generating adversarial policies with the training process. To oppose this, robust features are learned in the target network, which leads to a significant performance improvement. Meanwhile, the augmentation policy search is performed along with the training of a target network, and the computation in network training is reused for policy evaluation, which can extremely reduce the search cost and make our method more computing-efficient.

REFERENCES

- Antreas Antoniou, Amos J. Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *ICLR*, 2017.
- Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CVPR*, 2018.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.
- Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.
- Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using GAN for improved liver lesion classification. *IEEE International Symposium on Biomedical Imaging (ISBI)*, 2018.
- Xavier Gastaldi. Shake-shake regularization. *CoRR*, abs/1705.07485, 2017.
- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *NIPS*, 2014.
- Minghao Guo, Zhao Zhong, Wei Wu, Dahua Lin, and Junjie Yan. IRLAS: inverse reinforcement learning for architecture search. *CoRR*, abs/1812.05285, 2018.
- Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and Venkatesh Babu Radhakrishnan. Deligan : Generative adversarial networks for diverse and limited data. *CVPR*, 2017.
- Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *CVPR*, 2017.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.
- Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. *CoRR*, abs/1812.01187, 2018.
- Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. Population based augmentation: Efficient learning of augmentation policy schedules. *ICML*, 2019.
- Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
- Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefer, and Daniel Soudry. Augment your batch: better training with larger batches. *CoRR*, abs/1901.09335, 2019.
- Hiroshi Inoue. Data augmentation by pairing samples for images classification. *CoRR*, abs/1801.02929, 2018.
- Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.
- Alex Krizhevsky and Geoffrey E. Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.
- Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation - learning an optimal data augmentation strategy. *CoRR*, abs/1703.08383, 2017.
- Chen Lin, Minghao Guo, Chuming Li, Wei Wu, Dahua Lin, Wanli Ouyang, and Junjie Yan. Online hyper-parameter learning for auto-augmentation strategy. *CoRR*, abs/1905.07373, 2019.

- Xi Peng, Zhiqiang Tang, Fei Yang, Rogério Schmidt Feris, and Dimitris N. Metaxas. Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. *CVPR*, 2018.
- Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017.
- Julian Salazar, Davis Liang, Zhiheng Huang, and Zachary C. Lipton. Invariant representation learning for robust deep networks. *NeurIPS Workshop*, 2018.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CVPR*, 2015.
- Toan Tran, Trung Pham, Gustavo Carneiro, Lyle J. Palmer, and Ian D. Reid. A bayesian data augmentation approach for learning deep models. *NIPS*, 2017.
- Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. *ICML*, 2013.
- Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. *CVPR*, 2017.
- Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.
- Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. *CoRR*, abs/1802.02375, 2018.
- Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *British Machine Vision Conference*, 2016.
- Zhao Zhong, Junjie Yan, and Cheng-Lin Liu. Practical network blocks design with Q-learning. *CVPR*, 2018a.
- Zhao Zhong, Zichen Yang, Boyang Deng, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. BlockQNN: Efficient block-wise neural network architecture generation. *CoRR*, abs/1808.05584, 2018b.
- Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *ICLR*, 2016.
- Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CVPR*, 2017.

A APPENDIX

A.1 HYPERPARAMETERS

We detail the model hyperparameters on CIFAR-10/CIFAR-100 and ImageNet in Table 7.

Table 7: Model hyperparameters on CIFAR-10/CIFAR-100 and ImageNet. LR represents learning rate, and WD represents weight decay. We do not specifically tune these hyperparameters, and all of these are consistent with previous works, except for the number of epochs.

Dataset	Model	Batch Size ($N \cdot M$)	LR	WD	Epoch
CIFAR-10	Wide-ResNet-28-10	$128 \cdot 8$	0.1	$5e-4$	200
CIFAR-10	Shake-Shake (26 2x32d)	$128 \cdot 8$	0.2	$1e-4$	600
CIFAR-10	Shake-Shake (26 2x96d)	$128 \cdot 8$	0.2	$1e-4$	600
CIFAR-10	Shake-Shake (26 2x112d)	$128 \cdot 8$	0.2	$1e-4$	600
CIFAR-10	PyramidNet+ShakeDrop	$128 \cdot 8$	0.1	$1e-4$	600
CIFAR-100	Wide-ResNet-28-10	$128 \cdot 8$	0.1	$5e-4$	200
CIFAR-100	Shake-Shake (26 2x96d)	$128 \cdot 8$	0.1	$5e-4$	1200
CIFAR-100	PyramidNet+ShakeDrop	$128 \cdot 8$	0.5	$1e-4$	1200
ImageNet	ResNet-50	$2048 \cdot 8$	0.8	$1e-4$	120
ImageNet	ResNet-50-D	$2048 \cdot 8$	0.8	$1e-4$	120
ImageNet	ResNet-200	$2048 \cdot 8$	0.8	$1e-4$	120