Deep AutoAugment

Yu Zheng [1] **Zhi Zhang** [2] **Shen Yan** [1] **Mi Zhang** [1]

Abstract

While recent automatic data augmentation works lead to state-of-the-art results, their design spaces and the derived data augmentation strategies still incorporate strong human priors. In this work, instead of selecting a set of hand-picked default augmentations alongside the searched data augmentations, we propose a fully automated approach for data augmentation search called Deep AutoAugment (DeepAA). We propose a search strategy that matches the directions of the validation gradients and the training gradients averaged over all possible augmentations. Our experiments show that DeepAA achieves strong performance on CIFAR-10/100 and SVHN with much less search cost compared to state-of-the-art data augmentation search methods.

1. Introduction

Many of the state-of-the-art results rely on data augmentation (DA) since it effectively regularizes the model by increasing the number of data points (Goodfellow et al., 2016; Zhang et al., 2017). A large body of data augmentation transformations has been proposed (Inoue, 2018; Zhang et al., 2018; DeVries & Taylor, 2017; Yun et al., 2019; Hendrycks et al., 2020; Yan et al., 2020) to improve the model performance and robustness. While applying a set of well designed augmentation transformations could help yield considerable performance enhancement, manually selecting high-quality augmentation transformations and determining how they should be combined still require strong domain expertise and prior knowledge of the dataset of interest. With the recent trend of automated machine learning, data augmentation search flourished in the image domain (Cubuk et al., 2019; 2020; Ho et al., 2019; Lim et al., 2019; Hataya et al., 2020; Li et al., 2020; Liu et al., 2021), which learns augmentation policies over a set of transformations.

[1]Michigan State University [2]Amazon Web Services. Correspondence to: Yu Zheng <zhengy30@msu.edu>.

The 38^{th} International Conference on Machine Learning (ICML) Workshop on Machine Learning for Data: Automated Creation, Privacy, Bias (ML4data).

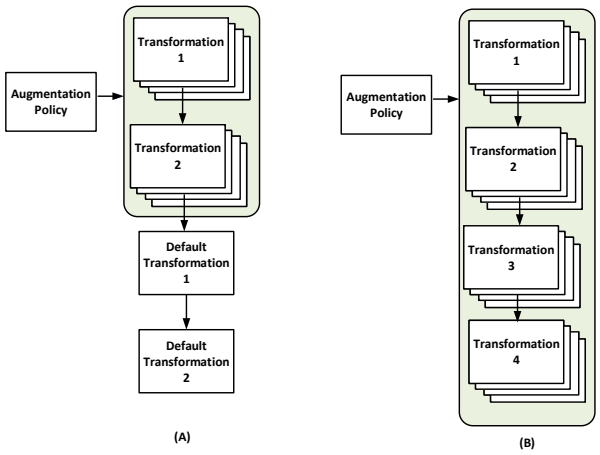

Figure 1. (A) Existing automated data augmentation methods (shallow augmentation search layers followed by hand-picked transformations). (B) Deep AutoAugment (deep augmentation search layers without hand-picked transformations).

Although previous works achieved SOTA performance on several vision tasks, we argue that data augmentation in (Cubuk et al., 2019; 2020; Ho et al., 2019; Lim et al., 2019; Hataya et al., 2020; Li et al., 2020; Liu et al., 2021) is *not fully automated*. This is because these methods only search for shallow layers of augmentation policy followed by *hand-picked default* transformations (Figure 1(A)), while a fully automated search should be ideally performed without any domain knowledge. The augmentation pipeline with default transformations are known to perform strong even with random policies (LingChen et al., 2020; Müller & Hutter, 2021). In other words, the previously designed search space still embeds strong priori.

In this work, we propose Deep AutoAugment (DeepAA), a multi-layer data augmentation search method which aims to remove the need of hand-crafted default transformations (Figure 1(B)). DeepAA fully automates the data augmentation by searching a deep data augmentation policy on an expanded set of transformations that includes the widely adopted search space as well as the default transformations (flips, Cutout and pad-and-crop), leading to a total of 139 different transformations. DeepAA treats the training gradient as a function of augmentation policy, and formulates data augmentation search as maximizing the cosine similarity

between training and validation gradients. The independent sampling at each augmentation layer and the estimated mean gradient of training data are implemented to speed up and stabilize the data augmentation policy optimization.

We evaluate the performance of DeepAA on three datasets CIFAR-10, CIFAR-100, and SVHN, and compare the results with SOTA data augmentation search methods including AutoAugment (AA) (Cubuk et al., 2019), PBA (Ho et al., 2019), Fast AutoAugment (FastAA) (Lim et al., 2019), DADA (Li et al., 2020), and RandAugment (RA) (Cubuk et al., 2020). Our results show that DeepAA achieves strong performance on CIFAR-10, CIFAR-100, and SVHN with much less policy search cost compared to state-of-the-art automated data augmentation methods.

2. Related Work

Automated Data Augmentation. Automating data augmentation policy design has recently emerged as a promising paradigm for data augmentation. The pioneer work on automated data augmentation was proposed in AutoAugment (Cubuk et al., 2019), where the search is performed under reinforcement learning framework. This method requires to train the neural network repeatedly, which takes thousands of GPU hours to converge. Subsequent works (Lim et al., 2019; Li et al., 2020; Liu et al., 2021) aim at reducing the computation cost. Fast AutoAugment (Lim et al., 2019) treats data augmentation as inference time density matching which can be implemented efficiently with Bayesian optimization. Differentiable Automatic Data Augmentation (DADA) (Li et al., 2020) further reduces the computation cost through a reparameterized Gumbel-softmax distribution (Jang et al., 2017). To avoid relaxing categorical distribution to Gumbel-softmax, Direct Differentiable Augmentation Search (DDAS) (Liu et al., 2021) exploits meta-learning with one step gradient update. RandAugment (Cubuk et al., 2020) introduces a simplified search space containing two interpretable hyperparameters, which can be optimized simply by grid search. Although many automated data augmentation methods have been proposed, the use of default augmentations still imposes strong domain knowledge. DeepAA is proposed to address this constraint.

Gradient Direction Matching. Our work is also related to gradient direction matching. Investigating gradient direction of an individual training sample sheds lights on the influence of such example on the final performance of models trained with stochastic gradient descent (Pruthi et al., 2020; Hara et al., 2019). To utilize the gradient direction to optimize data usage, DDS (Wang et al., 2020) uses the gradient inner product as a reward to up-weight the data that has similar gradient with the development set. A similar approach was proposed in (Müller et al., 2021), which uses the next batch as a cheap proxy of the development set and proposes an efficient method to compute the example-wise alignment. (Du et al., 2018) observes that the cosine similarity between gradients of different tasks provides a signal to detect whether an auxiliary loss is helpful to the main loss. In this work, we propose to use the cosine similarity between the augmented training data and validation data as a reward signal to optimize the sampling probability of the corresponding data augmentation transformations.

3. Deep AutoAugment

We first introduce the search space of the data augmentation policy. We then formulate a new search strategy based on gradient direction matching to search the data augmentation policy in an unconstrained and efficient way.

3.1. Search Space

Let $\mathcal{O} = \{o_1, ..., o_N\}$ denote a set of N augmentation transformations, where each transformation $o : \mathcal{X} \to \mathcal{X}$ transforms an input image in the space $\mathcal{X}$. To augment an image $x \in \mathcal{X}$, we apply K layers of transformations to the image as $\tilde{x} = o_{n_K} \circ o_{n_{K-1}} \circ \cdots \circ o_{n_1}(x)$. In the k^{th} layer, the transformation $o_{n_k} \in \mathcal{O}$ is sampled based on an N-dimension categorical distribution p_{θ_k} parameterized by θ_k, where $n_k \sim p_{\theta_k}$. The data augmentation policy hence consists of K categorical distributions $\mathcal{P} = \{p_{\theta_1}, \cdots, p_{\theta_K}\}$.

Following (Cubuk et al., 2019; 2020; Li et al., 2020; Liu et al., 2021; LingChen et al., 2020; Müller & Hutter, 2021), we discretize the magnitude of each transformation, and treat transformations with different discretized magnitudes as independent ones. Different from previous works, the transformation set $\mathcal{O}$ contains not only the widely adopted transformations (e.g., rotate, autoContrast) but also the ones previously used as the default augmentations (flips, Cutout, pad-and-crop). Moreover, instead of searching for a shallow augmentation policy consisting of either one or two layers of transformations, DeepAA searches for an augmentation policy with deeper layers.

3.2. Search Strategy

3.2.1. Augmentation Policy Search via Gradient Direction Matching

In DeepAA, we consider searching the data augmentation policy as a problem of gradient direction matching between the validation data and the augmented training data. The intuition behind this formulation is that an effective data augmentation should preserve data distribution (Chen et al., 2020) where the distribution of the augmented images should align with the distribution of the validation set such that the training gradient direction is close to the validation gradient direction. Therefore, we propose to maximize the

cosine similarity between the gradients of the validation data and the augmented training data.

Specifically, let x_t and D_v denote a single training sample and a batch of validation samples from the same class respectively. Since we augment the training sample x_t following the policy $\mathcal{P}$ parameterized by $\theta = \{\theta_1, \cdots, \theta_k\}$, the training gradient $g(\theta)$ should also be a function of θ. Let v denote the gradient over the validation samples. We perform the data augmentation policy search by maximizing the cosine similarity between the gradients of augmented training data and validation data as follows:

$$\begin{aligned}\theta &= \arg\max_{\theta} \text{ cosineSimilarity}(v, g(\theta)) \\ &= \arg\max_{\theta} \frac{v^T \cdot g(\theta)}{\|v\| \cdot \|g(\theta)\|}\end{aligned} \tag{1}$$

where $\|\cdot\|$ denotes the L2-norm. The distribution parameters can be optimized via gradient ascent:

$$\theta \leftarrow \theta + \eta \nabla_\theta \text{ cosineSimilarity}(v, g(\theta)) \tag{2}$$

3.2.2. Policy Optimization for One Layer

We start with the case where the data augmentation policy only contains a single augmentation layer (i.e., $\mathcal{P} = \{p_\theta\}$).

Let $L(x_t; w)$ denote the classification loss of data x_t where $w \in \mathbb{R}^p$ represents the weights of the neural network. Consider applying augmentation on the training sample x_t following the distribution p_θ, the resulting training data gradient can be calculated analytically as

$$\begin{aligned}g(\theta) &= \sum_{n=1}^{N} p_\theta(n) \nabla_w L(o_n(x_t); w) \\ &= G \cdot p_\theta\end{aligned} \tag{3}$$

where $G = [\nabla_w L(o_1(x_t); w), \cdots, \nabla_w L(o_N(x_t); w)]$ and $p_\theta = [p_\theta(1), \cdots, p_\theta(N)]^T$ is the N-dimension Categorical distribution. The gradient w.r.t. the cosine similarity in Eqn. (2) can be derived as:

$$\nabla_\theta \text{ cosineSimilarity}(v, g(\theta)) = \nabla_\theta (r^T p_\theta), \tag{4}$$

where

$$r = G^T \left(\frac{v}{\|g(\theta)\|} - \frac{v^T g(\theta)}{\|g(\theta)\|^2} \cdot \frac{g(\theta)}{\|g(\theta)\|} \right) \tag{5}$$

where the n^{th} element of vector r can be interpreted as the reward of the n^{th} augmentation transformation o_n.

Given that Eqn. (5) can be implemented efficiently via Jacobian-vector product by autodiff libraries such as TensorFlow (Abadi et al., 2016).

3.2.3. Policy Optimization for Multiple Layers

For augmentation layer k ($k \geq 2$), it becomes impractical to compute the training gradient $g(\theta_k)$ given that the number of augmentation combinations N^k grows exponentially w.r.t. k. To efficiently search the data augmentation policy, we propose to independently sample a batch of L ($L \ll N^{k-1}$) augmentation sequences following the distributions of the previous $k-1$ augmentation layers $\{p_{\theta_1}, \cdots, p_{\theta_{k-1}}\}$. We obtain a batch of L augmented samples as

$$\begin{aligned}&\{\tilde{x}_l | \tilde{x}_l = o_{n^l_{k-1}} \circ \cdots \circ o_{n^l_1}(x_t), \\ &n^l_{k-1} \sim p_{\theta_{k-1}}, \cdots, n^l_1 \sim p_{\theta_1},\ l = 1, 2, \cdots, L\}\end{aligned} \tag{6}$$

Based on the augmented samples, we can estimate the training data gradient at layer k as

$$\tilde{g}(\theta_k) = G_k \cdot p_{\theta_k} \tag{7}$$

where G_k is estimated via Monte Carlo average as

$$G_k = \frac{1}{L} \sum_{l=1}^{L} [\nabla_w L(o_1(\tilde{x}_l); w), \cdots, \nabla_w L(o_N(\tilde{x}_l); w)] \tag{8}$$

Substituting the analytical solution $g(\theta)$ and G in Eqn. (5) with the Monte Carlo average $\tilde{g}(\theta_k)$ and G_k, we obtain $\tilde{r}_k$ as an estimate of r. We found that it helps to stabilize the training by replacing r in Eqn. (4) with $\tilde{r}_k$ minus one standard deviation of $\tilde{r}_k$ calculated from previous iterations, as it reduces the reward that has large uncertainty.

4. Experiments and Results

In this section, we evaluate the performance of Deep AutoAugment (DeepAA) on CIFAR-10, CIFAR-100, and SVHN datasets and compare the results with a baseline augmentation method (flip left-right and random translations (Cubuk et al., 2019; 2020)) as well as SOTA automated data augmentation methods including AutoAugment (AA) (Cubuk et al., 2019), PBA (Ho et al., 2019), Fast AutoAugment (Fast AA) (Lim et al., 2019), DADA (Li et al., 2020), and RandAugment (RA) (Cubuk et al., 2020).

We set up the search space to include $K = 4$ augmentation layers, where each layer contains 15 transformations (identity, shear-x, shear-y, translate-x, translate-y, rotate, solarize, equalize, color, posterize, contrast, brightness, sharpness, autoContrast, invert) used in (Cubuk et al., 2019) as well as 3 additional transformations (flips, Cutout and pad-and-crop) which were used as the default augmentations in previous works (Cubuk et al., 2019; 2020; Lim et al., 2019; Li et al., 2020; Liu et al., 2021; LingChen et al., 2020; Müller & Hutter, 2021). Among the 15 transformations, there are 11 transformations (shear-x, shear-y, translate-x, translate-y, rotate,

Dataset	Baseline	AA	PBA	Fast AA	DADA	RA	DeepAA
CIFAR-10	3.9	2.6	2.6	2.7	2.7	2.7	**2.57 ± 0.15**
CIFAR-100	18.8	17.1	16.7	17.3	17.5	16.7	**16.30 ± 0.17**
SVHN	1.5	1.1	1.1	1.2	1.2	**1.0**	1.13 ± 0.04

Table 1. Test error rates (%) on CIFAR-10, CIFAR-100, and SVHN for Wide-ResNet-28-10 model. The reported result is the average of 4 independent runs with 95% confidence interval.

Dataset	Baseline	AA	PBA	Fast AA	DADA	RA	DeepAA
CIFAR-10	2.9	2.0	2.0	2.0	2.0	2.0	**1.89 ± 0.12**
CIFAR-100	17.1	**14.3**	15.3	14.9	15.3	-	14.81 ± 0.28

Table 2. Test error rates (%) on CIFAR-10, CIFAR-100 for Shake-Shake-2x96d model. The reported result is the average of 4 independent runs with 95% confidence interval.

Dataset	AA	PBA	Fast AA	DADA	RA	DeepAA
CIFAR-10/100	5000	5	3.5	0.1	25	3
SVHN	1000	1	1.5	0.1	160	1.2

Table 3. Policy search time on CIFAR-10/100, and SVHN in GPU hours[2].

solarize, color, posterize, contrast, brightness, sharpness) associated with magnitude parameters. We then discretize the range of magnitudes to 12 discrete levels and treat each augmentation transform with these discrete magnitudes as 12 independent operators. For 7 transformations that do not have maginitude (i.e., identity, equalize, autoContrast, invert, flips, Cutout, and pad-and-crop), we treat each of them as a single transformation. Therefore, we have a total of $N = 139$ transformations in each augmentation layer.

4.1. CIFAR-10 and CIFAR-100

Policy Search. We followed (Cubuk et al., 2019) to search for the best policy on the reduced CIFAR-10 dataset which contains 4,000 randomly chosen examples. We use Wide-ResNet-40-2 (Zagoruyko & Komodakis, 2016) as a proxy model for computational efficiency, and train it on the reduced CIFAR-10 dataset for 50 epochs with the same parameters as in (Lim et al., 2019). We conduct the policy search for 128 epochs at each augmentation layer.

Policy Evaluation. The augmentation policy found during search is used to train Wide-ResNet-28-10 model (Zagoruyko & Komodakis, 2016) on CIFAR-10 and CIFAR-100 respectively. We use the SGD optimizer with Nesterov momentum of 0.9. We set weight decay to $5e-4$ and batch size to 128. We train Wide-ResNet-28-10 for 200 steps with a learning rate of 0.1 and a cosine learning rate decay.

Results. In Table 1, we show the test error rates on Wide-ResNet-28-10. As shown, DeepAA significantly improves the performance of baseline and outperforms SOTA automated data augmentation methods on both CIFAR-10 and CIFAR-100. In particular, we achieve a test error rate of 2.4% on CIFAR-10 and 16.3% on CIFAR-100, which is 0.2% and 0.4% better than the state-of-the-art (AA and RA) with much less policy search cost (5,000 and 25 GPU hours vs. 3 GPU hours).

4.2. SVHN

Policy Search. We search for the best policy on a subset of 2,000 images of SVHN core dataset. We use Wide-ResNet-40-2 as a proxy model for computational efficiency, and train it for 50 epochs with SGD optimizer, learning rate of 0.005, $1e-3$ weight decay and batch size of 128. We search for the augmentation parameters using the same configuration as CIFAR-10.

Policy Evaluation. The augmentation policy found during the search is used to train Wide-ResNet-28-10 model on SVHN. We train the model for 160 epochs with Nesterove momentum 0.9, learning rate of $5e-3$, cosine learning rate decay, weight decay of $1e-3$, and batch size of 128.

Results. In Table 1, we show the test error rates on Wide-ResNet-28-10. As shown, DeepAA outperforms the baseline and obtains competitive performance to other automated data augmentation methods.

[2]The time for AA is measured using P100 GPU, PBA and DADA are measured using Titan XP GPU, and FastAA and DeepAA are measured using V100 GPU. Baseline is not included, since it is manually designed. The time for RA is estimated on V100 GPU for CIFAR and SVHN datasets.

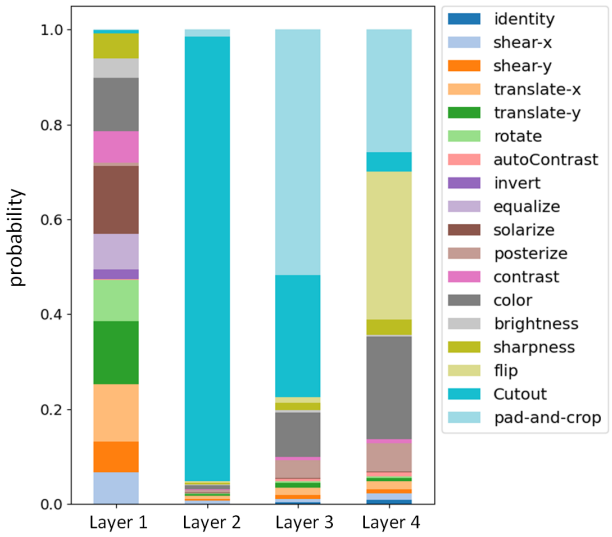

Figure 2. The distribution of augmentation transformations at each layer of the policy for CIFAR-10/100. The probability of each transformation (e.g., rotate) is summed up over all 12 discrete intensity levels of the corresponding transformation.

4.3. Policy Search Efficiency

Table 3 compares the policy search time on CIFAR-10, CIFAR-100, and SVHN in GPU hours. As shown, it only takes DeepAA 3 GPU hours for the policy search on CIFAR-10/100 and 1.2 GPU hours on SVHN, which are less than both Fast AA and RA that are claimed to be two practical automated data augmentation methods (Lim et al., 2019; Cubuk et al., 2020).

4.4. Policy Visualization

In Figure 2 and Figure 3, we show one searched augmentation policy for CIFAR-10/100 dataset. Figure 2 shows the distribution of transformations at each layer of the augmentation pipeline. As shown, the first layer contains a diverse set of transformations; the second layer contains primarily Cutout; the third layer is dominated by pad-and-crop, Cutout and color; and the fourth layer is dominated by flips, pad-and-crop, posterize and color. Figure 3 shows the magnitudes of 11 transformations that have magnitude parameters (the transformations without magnitude parameters (i.e., identity, autoContrast, invert, equalize, flips, Cutout and pad-and-crop) are not shown). We observe that the magnitudes of these transformations are high in the first layer, while in the second to fourth layers, the magnitudes of these transformations excluding solerize and posterize are centered around zero. This indicates that the corresponding transformations are close to the identity function. It should be noted that the last three layers are dominated by flips, pad-and-crop, and Cutout as observed in Figure 2, and hence the augmentation pipeline does not collapse to the identity function.

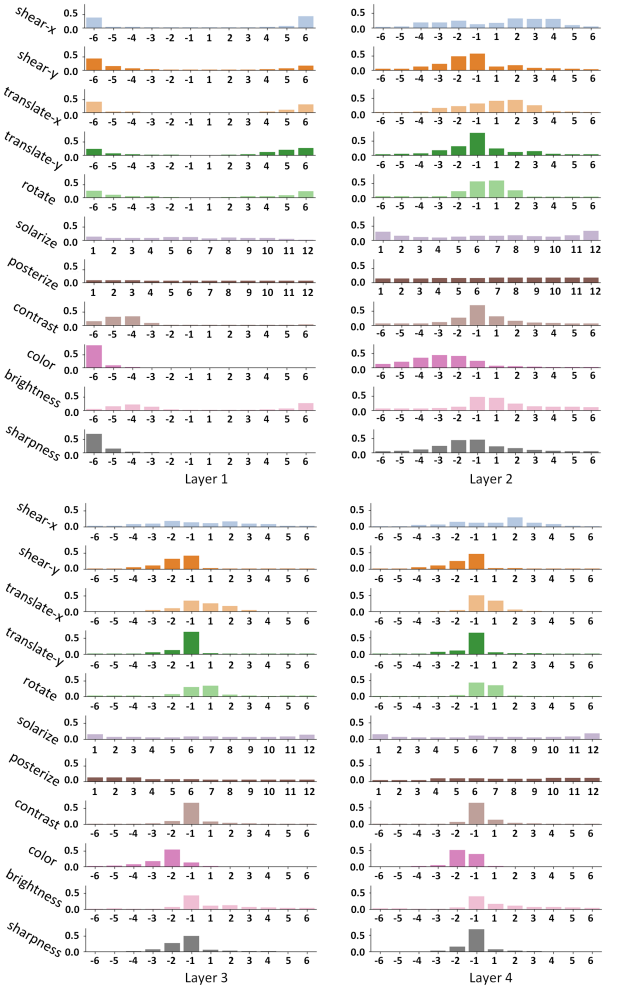

Figure 3. The distribution of discrete magnitudes of each augmentation transformation in each layer of the policy for CIFAR-10/100. The x-axis represents the discrete magnitudes and the y-axis represents the probability. The magnitude is discretized to 12 levels with each transformation having its own range. A large absolute value of the magnitude corresponds to high transformation intensity. Note that we do not show identity, autoContrast, invert, equalize, flips, Cutout and pad-and-crop because they do not have intensity parameters.

5. Conclusion

In this work, we propose Deep AutoAugment (DeepAA), a multi-layer data augmentation search method without using hand-picked transformations. DeepAA formulates data augmentation search as a gradient direction matching objective between the validation data and the augmented training data. The independent sampling at each layer and the estimated mean gradient of training data are proposed to speed up and stabilize the policy optimization. Our experiment results show that DeepAA achieves state-of-the-art results compared to other automated data augmentation methods.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. This work was partially supported by Amazon AWS Machine Learning Research Award.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

Chen, S., Dobriban, E., and Lee, J. H. A group-theoretic framework for data augmentation. *Journal of Machine Learning Research*, 21(245):1–71, 2020.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, pp. 702–703, 2020.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Du, Y., Czarnecki, W. M., Jayakumar, S. M., Farajtabar, M., Pascanu, R., and Lakshminarayanan, B. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*. MIT press Cambridge, 2016.

Hara, S., Nitanda, A., and Maehara, T. Data cleansing for models trained with sgd. *arXiv preprint arXiv:1906.08473*, 2019.

Hataya, R., Zdenek, J., Yoshizoe, K., and Nakayama, H. Faster autoaugment: Learning augmentation strategies using backpropagation. In *European Conference on Computer Vision*, pp. 1–16. Springer, 2020.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. Augmix: A simple data processing method to improve robustness and uncertainty. *ICLR*, 2020.

Ho, D., Liang, E., Chen, X., Stoica, I., and Abbeel, P. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pp. 2731–2741. PMLR, 2019.

Inoue, H. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017.

Li, Y., Hu, G., Wang, Y., Hospedales, T., Robertson, N. M., and Yang, Y. Differentiable automatic data augmentation. In *European Conference on Computer Vision*, pp. 580–595. Springer, 2020.

Lim, S., Kim, I., Kim, T., Kim, C., and Kim, S. Fast autoaugment. *NeurIPS*, 2019.

LingChen, T. C., Khonsari, A., Lashkari, A., Nazari, M. R., Sambee, J. S., and Nascimento, M. A. Uniformaugment: A search-free probabilistic data augmentation approach. *arXiv preprint arXiv:2003.14348*, 2020.

Liu, A., Huang, Z., Huang, Z., and Wang, N. Direct differentiable augmentation search. *arXiv preprint arXiv:2104.04282*, 2021.

Müller, S., Biedenkapp, A., and Hutter, F. In-loop meta-learning with gradient-alignment reward. *arXiv preprint arXiv:2102.03275*, 2021.

Müller, S. G. and Hutter, F. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. *arXiv preprint arXiv:2103.10158*, 2021.

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33, 2020.

Wang, X., Pham, H., Michel, P., Anastasopoulos, A., Carbonell, J., and Neubig, G. Optimizing data usage via differentiable rewards. In *International Conference on Machine Learning*, pp. 9983–9995. PMLR, 2020.

Yan, S., Song, H., Li, N., Zou, L., and Ren, L. Improve unsupervised domain adaptation with mixup training. In *arXiv preprint arXiv: 2001.00677*, 2020.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *ICLR*, 2018.