香 港 中 文 大 學
The Chinese University of Hong Kong

A simple approach for improving Softmax CE Loss

Xufeng Yao

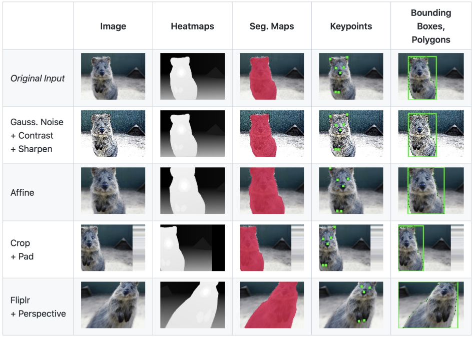
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Outline

Outline

Introduction

Data augmentation is a technique that aims to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model.

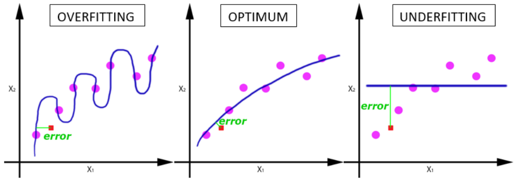


Examples of data augmentation [1]

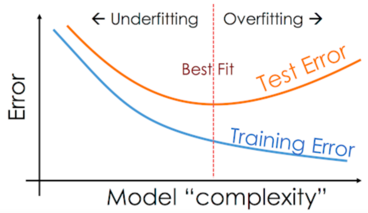
Outline

Under-fitting and Over-fitting

Underfitting refers to a model that can neither model the training dataset nor generalize to new dataset. Overfitting refers to the scenario where a machine learning model can't generalize or fit well on unseen dataset.



(a)



(b)

Bias and Variance Tradeoff

Assume a function with noise $y = f(x) + \epsilon$ where noise has zero mean and σ^2 variance.
We want to learn a function $\hat{f}(x)$ that approximates y.
We define Bias $\hat{f}(x) := \mathbb{E}[\hat{f}(x)] - f(x)$ and Var $\hat{f}(x) := \mathbb{E}[(\mathbb{E}(\hat{f}(x)) - \hat{f}(x))^2]$.
Then we can decompose its expected error:

$$
\begin{aligned}
\mathbb{E}[(\hat{f}(x) - y)^2] &= \mathbb{E}[\hat{f}(x) - f(x) - \epsilon + \mathbb{E}[\hat{f}(x)] - \mathbb{E}[\hat{f}(x)])^2] \\
&= \mathbb{E}[((\hat{f}(x) - \mathbb{E}[\hat{f}(x)]) + (\mathbb{E}[\hat{f}(x)] - f(x) - \epsilon))^2] \\
&= \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2 + 2(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])(\mathbb{E}[\hat{f}(x)] - f(x) - \epsilon) \\
&\quad +(\mathbb{E}[\hat{f}(x)] - f(x) - \epsilon)^2] \\
&= \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2 + (\mathbb{E}[\hat{f}(x)] - f(x))^2 + \epsilon^2 \\
&= Bias[\hat{f}(x)]^2 + Var[\hat{f}(x)] + \sigma.
\end{aligned} \tag{1}
$$

Note that f is the true function thus $\mathbb{E}[f] = f$.
Therefore, we find our expected error can be decided by both true error and complexity term.

L2 weight decay

Let $y = \sum wx + \epsilon$ where $\epsilon \in \mathbf{N}(0, \sigma^2)$, then we rewrite the equation:

$$\mathcal{L}_\theta = \sum (y - wx)^2 + \lambda w^2 \tag{2}$$

Suppose $P(y|x, w) = \mathbf{N}(y|wx, \sigma^2)$, then the likelihood function is:

$$\begin{aligned} \mathcal{L} &= \arg\max_\theta \ln \prod \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2}(y - wx)) \\ &= -\frac{1}{2\sigma^2} \sum (y - wx)^2 - m \ln \sigma\sqrt{2\pi} \\ &\Leftrightarrow \arg\min_\theta \sum (y - wx)^2 \end{aligned} \tag{3}$$

Assume $w \in \mathcal{N}(0, \tau^2)$, then we have the likelihood estimator:

$$\begin{aligned} \mathcal{L} &= \arg\max_w \ln \prod \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2}(y - wx)) \prod \frac{1}{\tau\sqrt{2\pi}} \exp(-\frac{1}{2}(\frac{w}{\tau})^2) \\ &= -\frac{1}{2\sigma^2} \sum (y - wx)^2 - \frac{1}{2\tau^2} \sum w^2 - m \ln \sigma\sqrt{2\pi} - n \ln \tau\sqrt{2\pi} \end{aligned} \tag{4}$$

Bayesian interpretation

From the above, we can find that the l2 regularization can be interpreted as a zero mean Guassian prior. We further derive it from a MAP perspective:

$$\underbrace{P(A|B)}_{\text{posterior}} = \frac{\overbrace{P(B|A)}^{\text{likelihood}}\overbrace{P(A)}^{\text{prior}}}{\underbrace{P(B)}_{\text{marginal}}}$$

Bayes' theorem

$$\begin{aligned}
\hat{\theta}_{MAP} &= \arg\max_{\theta} P(\theta|y) \\
&= \arg\max_{\theta} \frac{P(y|\theta)P(\theta)}{P(y)} \\
&= \arg\max_{\theta} P(y|\theta)P(\theta) \\
&= \arg\max_{\theta} \log P(y|\theta)P(\theta)
\end{aligned} \tag{5}$$

Noise Injection

Adding some noise mask in the input is a typical method of data augmentation. Suppose the noise $\epsilon \in \mathcal{N}(0, \sigma^2)$, then we have the new input: $x = x + \epsilon_i$. Then we have:

$$\begin{aligned}\mathbb{E}[(y-\hat{y})^2] &= \mathbb{E}[(y - wx - w\epsilon)^2] \\ &= \mathbb{E}[(y-wx)^2 - 2(y-wx)w\epsilon + (w\epsilon)^2] \\ &= \mathbb{E}[(y-wx)^2] - \mathbb{E}2[y-wx]w\epsilon + \mathbb{E}[(w\epsilon)^2]' \\ &= \mathbb{E}[(y-wx)^2] + \mathbb{E}[(w\epsilon)^2]\end{aligned} \tag{6}$$

since ϵ is independent of $(y-t)$, we don not need to take it into consideration.
Though noise injection is not exactly equivalent to L2 weight decay, it has similar effects compared with l2 penalty.
Suppose we add some noise on output, for some constant ϵ, the label y_t is correct with $1-\epsilon$. Then it regularizes a model based on the softmax with k output and replace hard label i.e., 0 and 1 with $\frac{\epsilon}{k}$ and $1 - \frac{k-1}{k}\epsilon$. That can be regareded as the basis of label smoothing.

Stein's paradox

Sometimes a mathematical result is strikingly contrary to generally held belief even though an obviously valid proof is given.

- Assuming you are taking part in a basketball camp, how to measure your shooting accuracy?

James-Stein estimator

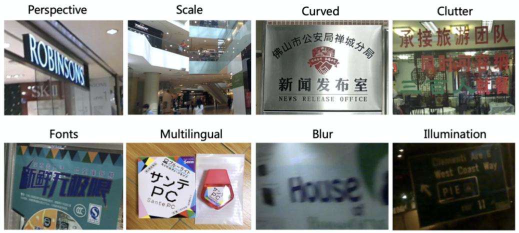
Let $Y \in \mathcal{N}(\theta, \sigma)$, Stein demonstrated that in terms of mean squared error $\mathbb{E}[(\theta - \hat{\theta})^2]$, the least squares estimator $\hat{\theta}_{LS}$ is not optimal to a shrinkage based estimators, such as James Stein estimator $\hat{\theta}_{JS}$, The paradoxical result, that there is a (possibly) better and never any worse estimate of θ in mean squared error as compared to the sample mean.

If σ^2 is known, the James-Stein estimator is given by: $\hat{\theta}_{JS} = (1 - \frac{(m-2)\sigma^2}{||y||^2})y$.

The phenomenon of over-fitting is really an unfortunate property of maximum likelihood.

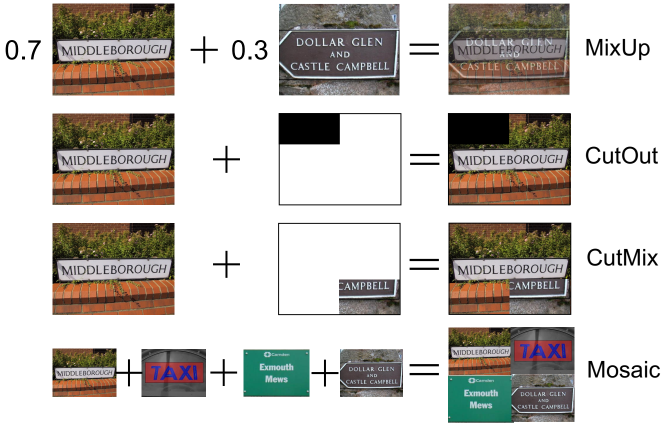
Outline

Common augmentations in OCR



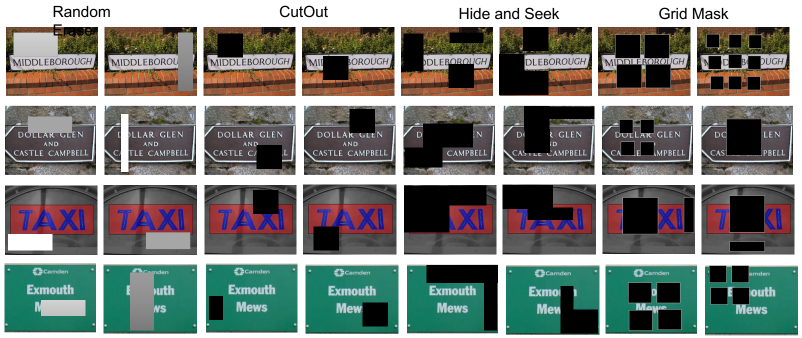
Common augmentation methods

Mix Family in OCR



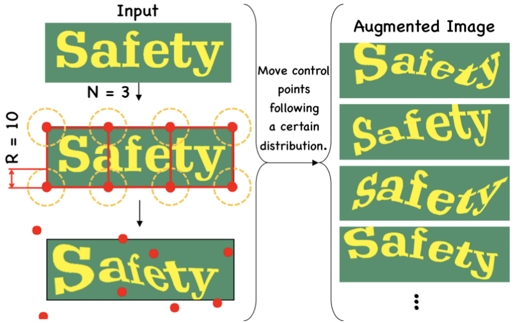
Mix family[2]–[6]

Mask family in OCR



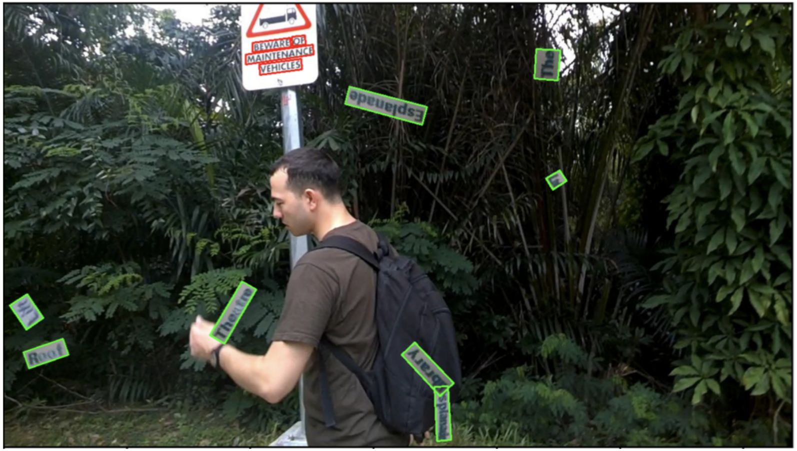
Mask family [3], [7]–[9]

Augmentations in OCR



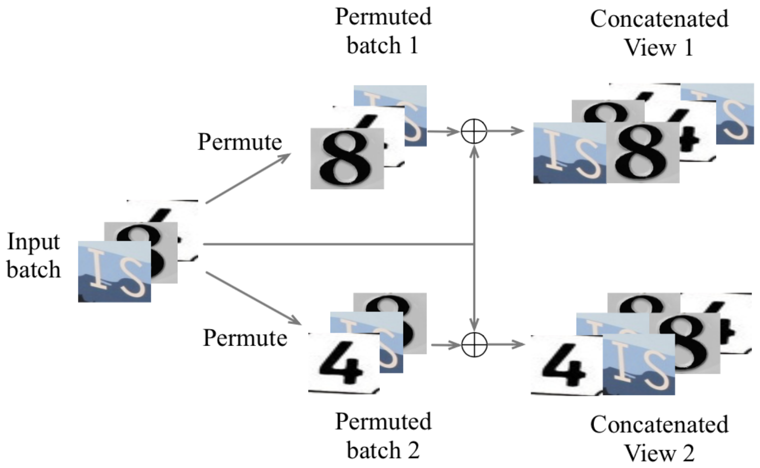
Geometric Augmentation [10]

Augmentations in OCR



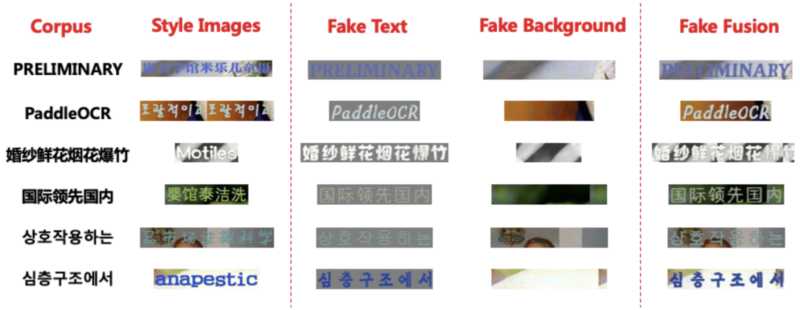
Copy-Paste [11]

Augmentations in OCR



Context-based augmentation

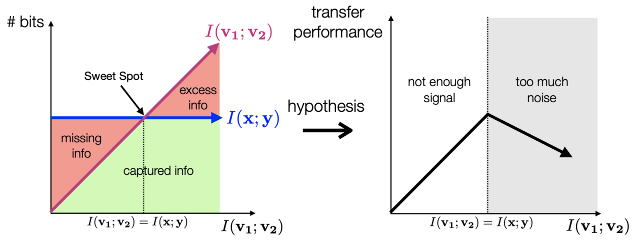
Augmentations in OCR



Style transfer [12]

Outline

What makes a good view for data augmentation

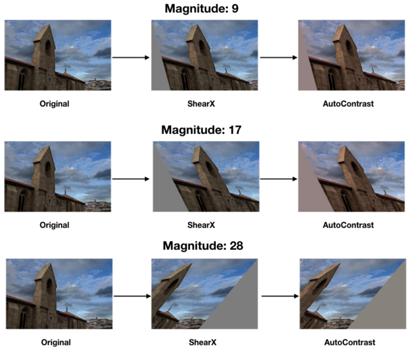


# bits
$I(\mathbf{v_1};\mathbf{v_2})$
Sweet Spot
excess info
$I(\mathbf{x};\mathbf{y})$
missing info
captured info
$I(\mathbf{v_1};\mathbf{v_2}) = I(\mathbf{x};\mathbf{y})$
$I(\mathbf{v_1};\mathbf{v_2})$
hypothesis
transfer performance
not enough signal
too much noise
$I(\mathbf{v_1};\mathbf{v_2}) = I(\mathbf{x};\mathbf{y})$
$I(\mathbf{v_1};\mathbf{v_2})$

Too weak or too strong data augmentation strategy is not we want [13]. Typically, we will miss some important information if we apply weak data augmentations. In contrast, excess information via strong data augmentations may ruin the performance of the network because it bring too much noise. We hypothesis there's a sweet spot for data augmentations for each task.

Problem Setting

Assume we have K data augmentation methods and M augmentation hyper-parameters, the combination of these data augmentation methods is super large (roughly K^M). Therefore, searching a optimal set of data augmentation methods is an important problem.

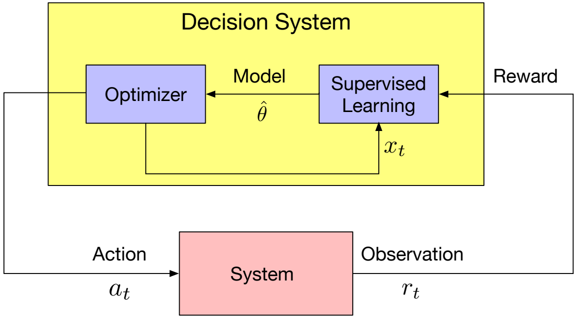


How to choose the optimal policy of data augmentation

The searching space is super large!

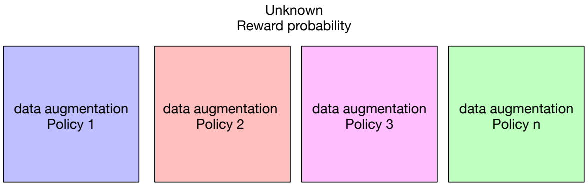
Problem Setting

Assume we have a decision system and want to learn a data augmentation policy. Each time we choose one action to get a policy p_t. We define one metric i.e., reward for evaluating the fitness of this data augmentation policy.



Overview of Decision System

Problem Definition


Unknown
Reward probability
data augmentation Policy 1
data augmentation Policy 2
data augmentation Policy 3
data augmentation Policy n

Which data augmentation policy to pick next?

Definition:

- We have K data augmentation policies with reward probabilities $\{\theta_1, \theta_2, ..., \theta_K\}$.
- At each time step t, we take an action on one data augmentation policy and receive a reward r.
- A is a set of actions, each referring to the one data augmentation policy. The value of action a is the expected reward $Q(a) = \mathbb{E}[r|a] = \theta$. In each time step t, $Q(a_t) = \theta_t$.
- R is a reward function. In our problem , R can be the entropy, rank or loss of the network.

The goal is to maximize the cumulative reward $\sum_{t=n}^{T} r_t$.

Strategy1: greedy algorithm

The first natural candidate algorithm is one which use the best set of data augmentation at every time step (after some fixed amount of exploration). Algorithm 1 details this algorithm.

Algorithm 1 greedy algo

```
1: procedure ALGO($t, cN$)
2:     while $t \neq 0$ do
3:         $t \leq cN$, select a random policy $P$ with probability $\frac{1}{N}$ and use it.
4:         $t > cN$, use policy $P$ with highest estimate
5:     end while
6:     return reward value.
7: end procedure
```

Strategy2: ϵ greedy algorithm

One way to overcome this fixed period of exploration is to force our algorithm to always explore. More precisely, at every time step t, Algorithm 2 explores a random arm with some probability ϵ.

Algorithm 2 ϵ greedy algo

```
1: procedure $\epsilon$ GREEDY ALGORITHM($t, cN$)
2:     while $t \neq 0$ do
3:         With probability $1 - \epsilon$, select the policy $P$ with the highest estimate;
4:         with probability $\epsilon$, select a random policy $P$ with probability $\frac{1}{N}$ and use it.
5:     end while
6:     return reward value.
7: end procedure
```

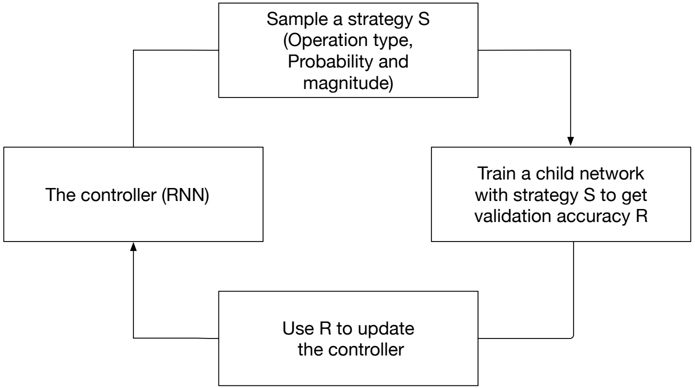
Strategy3: UCB [14] algorithm

The mechanics of the upper confidence bound (UCB) algorithm is simple. At each round, we simply choose the augmentation policy that has the highest empirical reward estimate up to that point plus some term that's inversely proportional to the number of times the arm has been played

Algorithm 3 UCB algo

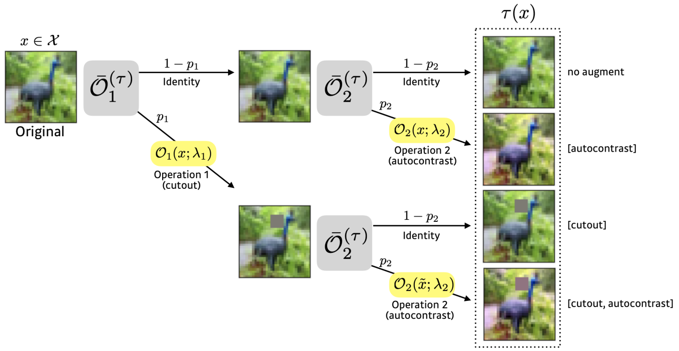
```
1: procedure UCB($t, cN$)
2:     while $t \neq 0$ do
3:         For $t = 1, .., K$, where $K$ is the number of the policies, choose policy $P_t$
4:         For $t = K + 1, ..., T$, choose the policy with the criterion:
5:         $P_t = \arg\max_{i \in \{1,...,K\}} UCB_{i,t-1}$
6:     end while
7:     return reward value.
8: end procedure
```

AutoAugment[15]: Learning Augmentation Policies from Data


Sample a strategy S (Operation type, Probability and magnitude)
Train a child network with strategy S to get validation accuracy R
Use R to update the controller
The controller (RNN)

Overview of our framework of using a search method (e.g., Reinforcement Learning) to search for better data augmentation policies. A controller RNN predicts an augmentation policy from the search space. A child network with a fixed architecture is trained to convergence achieving accuracy R. The reward R will be used with the policy gradient method to update the controller so that it can generate better policies over time.

Fast AutoAugment[16]



$\tau(x)$
$x \in \mathcal{X}$
Original
$\bar{\mathcal{O}}_1^{(\tau)}$
$1-p_1$
Identity
p_1
$\mathcal{O}_1(x;\lambda_1)$
Operation 1 (cutout)
$\bar{\mathcal{O}}_2^{(\tau)}$
$1-p_2$
Identity
p_2
$\mathcal{O}_2(x;\lambda_2)$
Operation 2 (autocontrast)
$\mathcal{O}_2(\tilde{x};\lambda_2)$
no augment
[autocontrast]
[cutout]
[cutout, autocontrast]

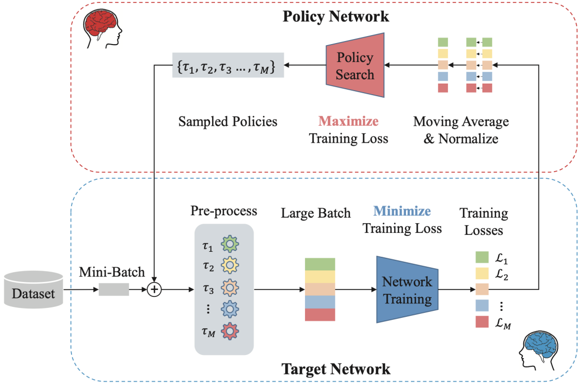
Each sub-policy τ consists of 2 operations; for instance, τ =[cutout, autocontrast] is used in this figure. Each operation $\bar{\mathcal{O}}_i^{(\tau)}$ has two parameters: the probability p_i of calling the operation and the magnitude λ_i of the operation. These operations are applied with the corresponding probabilities. As a result, a sub-policy randomly maps an input data to the one of 4 images. Note that the identity map (no augmentation) is also possible with probability $(1-p_1)(1-p_2)$.

RandAugment[17]

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugment(N, M):
"""Generate a set of distortions.

   Args:
     N: Number of augmentation transformations to
          apply sequentially.
     M: Magnitude for all the transformations.
"""

  sampled_ops = np.random.choice(transforms, N)
  return [(op, M) for op in sampled_ops]
```

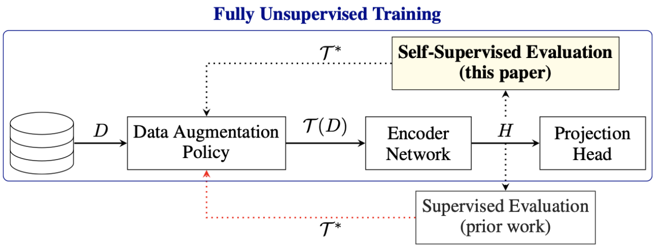
RandomAugment always select a transformation with uniform probability $\frac{1}{K}$. Given N transformations of the image, RandomAugment than express K^N potential policies.

Adversarial AutoAugment[18]



The overview of our proposed method. 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.

SelfAugment [19]



The blue box highlights our fully unsupervised training pipeline for instance contrastive representation learning: data D are augmented with policy T , then encoded into representations, H, which are fed into a projection head yielding features that determine the InfoNCE loss. As shown by the red arrow, prior work uses supervised evaluations of the representations, F to inform the training process,

References I

[1] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte, *et al.*, *imgaug*, `https://github.com/aleju/imgaug`, Online; accessed 01-Feb-2020, 2020.

[2] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

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

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

References II

[5] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[6] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2918–2928.

[7] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 13 001–13 008.

[8] K. K. Singh, H. Yu, A. Sarmasi, G. Pradeep, and Y. J. Lee, "Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond," *arXiv preprint arXiv:1811.02545*, 2018.

References III

[9] P. Chen, S. Liu, H. Zhao, and J. Jia, “Gridmask data augmentation,” *arXiv preprint arXiv:2001.04086*, 2020.

[10] C. Luo, Y. Zhu, L. Jin, and Y. Wang, “Learn to augment: Joint data augmentation and network optimization for text recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 746–13 755.

[11] Y. Du, C. Li, R. Guo, C. Cui, W. Liu, J. Zhou, B. Lu, Y. Yang, Q. Liu, X. Hu, *et al.*, “Pp-ocrv2: Bag of tricks for ultra lightweight ocr system,” *arXiv preprint arXiv:2109.03144*, 2021.

[12] L. Wu, C. Zhang, J. Liu, J. Han, J. Liu, E. Ding, and X. Bai, “Editing text in the wild,” in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 1500–1508.

[13] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, “What makes for good views for contrastive learning?” *arXiv preprint arXiv:2005.10243*, 2020.

References IV

[14] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 397–422, 2002.

[15] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[16] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," *Advances in Neural Information Processing Systems*, vol. 32, pp. 6665–6675, 2019.

[17] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[18] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong, "Adversarial autoaugment," *arXiv preprint arXiv:1912.11188*, 2019.

References V

[19] C. J. Reed, S. Metzger, A. Srinivas, T. Darrell, and K. Keutzer, “Selfaugment: Automatic augmentation policies for self-supervised learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2674–2683.