Automating Augmentation through Random Unidimensional Search

Xiaomeng Dong, Michael Potter, Gaurav Kumar, Yun-Chan Tsai, V. Ratna Saripalli
GE Healthcare

Abstract

It is no secret amongst deep learning researchers that finding the optimal data augmentation strategy during training can mean the difference between state-of-the-art performance and a run-of-the-mill result. To that end, the community has seen many efforts to automate the process of finding the perfect augmentation procedure for any task at hand. Unfortunately, even recent cutting-edge methods bring massive computational overhead, requiring as many as 100 full model trainings to settle on an ideal configuration. We show how to achieve equivalent performance in just 6: with Random Unidimensional Augmentation.

1 Introduction

Data augmentation is a widely used technique to improve deep-learning model performance. It is sometimes described as a “freebie” (Bochkovskiy, Wang, and Liao 2020) because it can improve model performance metrics without incurring additional computational costs at inferencing time. Unfortunately, creating a good data augmentation strategy typically requires human expertise and domain knowledge (Cubuk et al. 2020), which is inconvenient during initial development as well as when transferring existing strategies between different tasks. In an effort to overcome these drawbacks, researchers have begun looking for an automated solution to data augmentation.

AutoAugment (Cubuk et al. 2019) and its variants (FastAA (Lim et al. 2019) and PBA (Ho et al. 2019)) automated the data augmentation process by introducing augmentation parameters which are then jointly optimized alongside the neural network parameters during training. While these methods do offer an automated solution to the problem, they also introduce massive search spaces which in turn significantly increases the time required to train a model. For example, AutoAugment uses Reinforcement Learning (RL) on a search space of size $10^{32}$, which costs thousands of GPU hours to find a solution for a single task. Although later methods such as FastAA and PBA greatly improved the search and reduced computation requirements, they can still be undesirable due to the complexity of implementing joint optimization algorithms.

RandAugment (Cubuk et al. 2020) took a different approach by completely removing the policy optimization while achieving better results than prior methods. Unlike its predecessors which rely on applying RL to a search space of size $10^{32}$, RandAugment uses only two global parameters, reducing the search space from $10^{32}$ to $10^2$ so that a grid search can be a simple yet viable solution to the problem. As a result, RL is no longer needed for the policy search, making the method significantly easier to implement and more computationally feasible for practical usage.

Despite the significant complexity and efficiency enhancements made by RandAugment, there is still room for improvement. For example, the default setting of RandAugment uses a 10x10 grid search for the $10^2$ search space. While it is technically possible to run any training task 100 times, the computational cost of doing so may still be prohibitive, especially on large-scale datasets.

To address these problems, we propose Random Unidimensional Augmentation (RUA): a simpler yet more effective automated data augmentation workflow. The goal of RUA is to achieve the following two objectives:

1. Reduce the computational cost required to perform automated search, without sacrificing performance.
2. Eliminate the need for problem-specific human expertise in the process, enabling a fully automated workflow.

2 Methods

2.1 Dimensionality reduction: 2D to 1D

There are 2 global parameters defined in the search space of RandAugment: $M$ and $N$. $M$ represents the global distortion magnitude which controls the intensity of all augmentation operations. $N$ is the number of transformations to be applied in each training step. By default, $M$ and $N$ are both...
integer ranging from 1 to 10, with 10 giving the maximum augmentation effects.

Although the definitions of \( M \) and \( N \) are different, the end result of increasing their values is the same: more augmentation. If they could be merged into a single augmentation parameter, then the search space could be reduced by an order of magnitude. To check whether this might be possible, we ran RandAugment on a full 10x10 grid for two classification tasks. We used ResNet9 for Cifar10, and WRN-28-2 (Zagoruyko and Komodakis 2016) for SVHN (Netzer et al. 2011). Their test accuracies are shown in Figure 1.

The gradients in Figure 1 show a diagonal trend from the bottom left to the top right. Although the optimal accuracy regions vary between the two problems, the fact that both exhibit an approximately diagonal gradient raises the possibility of traversing the two parameters simultaneously. We proceed to do this, introducing a single parameter \( r \in [0, 1] \) such that \( r = M/M_{\text{max}} \) and \( r = N/N_{\text{max}} \). We then define our augmentation operation parameters directly in terms of \( r \), eliminating the need to pick an explicit value for \( M_{\text{max}} \).

This parameterization can be found in Table 1. This formulation leaves \( N_{\text{max}} \) as the single open parameter in the method. While one could simply set \( N_{\text{max}} = 10 \) in the footsteps of RandAugment, it can also be set lower while still providing adequate gradient traversal. We defer further discussion of this to Section 2.4.

In situations where \( r \star N_{\text{max}} \) is not an integer, we apply \( \lfloor r \star N_{\text{max}} \rfloor \) augmentations, plus a final augmentation which executes with probability equal to the floating point remainder. For example, if \( r \star N_{\text{max}} = 3.14 \), then 3 augmentations will be guaranteed, and a fourth will execute with 14% probability.

**2.2 More search with less compute**

Another interesting observation one can make from Figure 1 is that, traversing the diagonals of both Cifar10 and SVHN, accuracy first increases to a maximum and then decreases. In other words, there appears to be unimodality with respect to \( r \). If we extract these diagonal terms and plot their relative accuracies against \( r \) (Figure 2), the unimodal trend becomes more apparent.

To further verify this phenomena, we use a different architecture and apply the RUA transformations from Table 1 (right), then plot the relative performance against \( r \) in Figure 3. As we can see, despite some randomness on the local
Table 1: Augmentations and their associated parameters. Augmentations marked with a “∗” have non-zero impact at \( r = 0 \) under RA, but are zero-aligned under RUA (Section 2.3).

| Augmentations | RandAug (RA) | RUA |
|---------------|--------------|-----|
| Identity      | -            | -   |
| AutoContrast  | -            | -   |
| Equalize      | degree = ±30\(^\circ\) | -   |
| AutoContrast  | -            | -   |
| Solarize*     | threshold = 256\(^r\) | threshold = 256 - \( U(0, 256) \) |
| Rotate*       | degree = \( U(-90, 90) \) | -   |
| Rotate*       | degree = \( U(0, 256) \) | bit shift = \( U(0, 7) \) |
| Solarize      | threshold = 256\(^r\) | -   |
| Posterize*    | bit shift = 8 - 4\(^r\) | bit shift = \( U(0, 7) \) |
| Color*        | factor = 1.8\(^r\) + 0.1 | factor = \( U(-0.9r, 0.9r) \) |
| Contrast*     | factor = 1.8\(^r\) + 0.1 | factor = \( U(-0.9r, 0.9r) \) |
| Brightness*   | factor = 1.8\(^r\) + 0.1 | factor = \( U(-0.9r, 0.9r) \) |
| Sharpness*    | factor = 1.8\(^r\) + 0.1 | factor = \( U(-0.9r, 0.9r) \) |
| Shear-X       | coef = ±0.3\(^r\) | coef = \( U(-0.5r, 0.5r) \) |
| Shear-Y       | coef = ±0.3\(^r\) | coef = \( U(-0.5r, 0.5r) \) |
| Translate-X   | coef = ±100\(^r\) | coef = \( U(-r, r) \ast \text{width}/3 \) |
| Translate-Y   | coef = ±100\(^r\) | coef = \( U(-r, r) \ast \text{height}/3 \) |

Figure 3: RUA test accuracy vs \( r (N_{max} = 5) \).

PyramidNet-Cifar10

![Graph showing accuracy relative to baseline against r](image)

Algorithm 1: Golden Section Search (Max-Finding)

Require: \( f, a, b, \text{maxIter} \)
\[
\begin{align*}
\phi_1, \phi_2 & \leftarrow \sqrt{\frac{5}{2}} - \frac{1}{2}, \frac{\sqrt{5} - 1}{2} \\
 h & \leftarrow b - a \\
c & \leftarrow a + \phi_2 \ast h \\
d & \leftarrow a + \phi_1 \ast h \\
y_c & \leftarrow f(c) \\
y_d & \leftarrow f(d)
\end{align*}
\]
for \( i \) from 1 to \( \text{maxIter} \) do
  if \( y_c > y_d \) then
    \( b \leftarrow d \)
    \( d \leftarrow c \)
    \( y_d \leftarrow y_c \)
    \( h \leftarrow \phi_1 \ast h \)
    \( c \leftarrow a + \phi_2 \ast h \)
    \( y_c \leftarrow f(c) \)
  else
    \( a \leftarrow c \)
    \( c \leftarrow d \)
    \( y_c \leftarrow y_d \)
    \( h \leftarrow \phi_1 \ast h \)
    \( d \leftarrow a + \phi_2 \ast h \)
    \( y_d \leftarrow f(d) \)
end if
end for
if \( y_c > y_d \) then
  return \( c \)
else
  return \( d \)
end if

In light of this unimodal property, we can leverage algorithms that are more efficient than grid search to explore a larger search space using less computation. One such algorithm is golden-section search \( (\text{Kiefer} [1953]) \). Golden-section search is a simple method that is widely used for finding the maximum or minimum of a unimodal function over a given interval. The pseudo code for golden-section search is given in Algorithm 1.

With golden-section search, every evaluation (after the first) of the search space will reduce the remaining search space by a constant factor of \( \approx 0.618 \) (inverse golden ratio). As a result, we can search over 90% of the domain of \( r \) using only 6 evaluations. Note that this search space reduction does not require any human expertise or intervention, allow-
2.3 **RUA augmentation parameters**

After our search space reduction, we are left with one parameter, \( r \), which controls the global augmentation intensity. The exact manner of this control is given in Table 1 (right). A zero value of \( r \) means no augmentation, whereas a value of 1 achieves maximum augmentation.

This is a conceptual divergence from RandAugment, as 6 of their 14 transformations are not set up to scale this way. These transformations are marked with a “*” in Table 1 (left). For example, the transformation intensity of Solarize and Posterize are inversely correlated with \( r \). Moreover, Color, Contrast, Brightness, and Sharpness are all ‘shifted’ in that they cause no augmentation when \( r = 0.5 \), whereas values closer to 0 or 1 lead to stronger alterations to the input.

In addition to aligning \( r \) with augmentation intensity, we also introduce non-deterministic parameter selection into our augmentations. For example, rather than rotating an image exactly \( \pm 30 \) degrees whenever the Rotate operation is applied, we instead draw from a random uniform distribution \((U)\) to more thoroughly cover the augmentation space. The maximum intensity of certain augmentations are also increased to keep the expected intensity consistent in spite of the switch to uniform distributions. We justify each of these decisions with an ablation study in Section 3.2.

2.4 **Selecting a maximum \( N \)**

One question which must be answered when applying RUA is what value to use as \( N_{max} \). While one may be content to use 10, since that was the extent of the RandAugment search space, other numbers may well be equally valid. We ran a second grid search (Figure 5) using our RUA augmentation parameters to verify that large values of \( N_{max} \) may not be necessary in order to achieve a good performance while traversing the diagonal gradient we observed in Section 2.1. Based on this search, we examined what outcomes a user would achieve if they ran RUA using different values of \( N_{max} \) ranging from 1 to 10. This sensitivity analysis
is shown in Figure 6. For both SVHN and Cifar10, setting $N_{\text{max}} > 5$ does not appear to provide any significant benefit.

Given that RUA is relatively insensitive to higher values of $N_{\text{max}}$, there are pragmatic reasons to choose values smaller than 10. Applying a large number of transformations during training can severely bottleneck the training speed. See Figure 7 for an example. For our hardware, with any $N \geq 3$ the cpu-based preprocessing became rate limiting, especially once $N \geq 5$. This may be one reason why RandAugment never chose $N > 3$ in their sub-grid selections. With these factors in mind, we selected $N_{\text{max}} = 5$ for our final experiments.

3 Experiments

3.1 RUA performance assessment

In order to perform a direct comparison with previous works, we deploy RUA in the same training setting used by RandAugment on Cifar10, Cifar100, SVHN, and ImageNet. Details regarding the parameters used in each experiment are given in Table 2. There are a few things worth highlighting about our experimental parameters:

1. In order to be consistent with previous works, we also applied default augmentations before and after applying RUA augmentation on different tasks. For example, pad-and-crop, horizontal flip, and cutout (Devries and Taylor 2017) are used on the Cifar 10/100 datasets.

2. In Cifar10, RandAugment trained for 1800 epochs whereas the official implementation of PyramidNet (Han, Kim, and Kim 2017) and ShakeDrop (Yamada et al. 2019) trained for 300 epochs. We picked 900 epochs as a compromise between different official implementation settings.

3. In every dataset we hold out 5k training samples as evaluation data for selecting the best $r$. After selecting $r$, we put the hold-out set back into the training set and train again. We then record the test performance at the end of that final training.

The final test results of RUA are shown in Table 3, where our performance scores are from an average of 10 independent runs. The results of previous methods including the baseline, AA, Fast AA, PBA, and RA are taken from previous work (Cubuk et al. 2020). The best accuracies for each column are highlighted in bold.

As demonstrated in Table 3, RUA achieved equal or better test scores than previous state-of-the-art methods on 4 out of 5 tasks, while reducing the search space by an order of magnitude. For the Cifar10 tasks, we are equivalent to the best prior methods, with one-tailed t-test p-values of 0.0017 and 0.034. For Cifar100 and ImageNet our performance exceeds that of prior methods, with one-tailed t-test p-values of 0.002 and 0.039. On SVHN, despite being outperformed by RandAugment, RUA nonetheless achieved competitive performance on par with AutoAugment.

3.2 Ablation study

We conducted an ablation study on the various design decisions outlined in Section 2.3. The results of this study are given in Table 4. There are several noteworthy takeaways from these comparisons. First, making the “*” augmentations positively correlated with $r$ is always beneficial. This can be seen through pairwise comparisons of rows 1 vs 5, 2 vs 6, 3 vs 7, and 4 vs 8. The second takeaway is that using a random distribution to draw the transformation arguments is always beneficial. This can be seen through pairwise comparisons of rows 1 vs 3, 2 vs 4, 5 vs 7, and 6 vs 8. Finally, increasing the maximum strength of augmentations (for example rotating $\pm 90$ rather than $\pm 30$) is always deleterious on its own (rows 1 vs 2 and 5 vs 6), but advantageous when paired with random sampling (rows 3 vs 4 and 7 vs 8). This is not particularly surprising since larger
| Dataset       | CIFAR10       | CIFAR10       | CIFAR100      | SVHN (Core)   | ImageNet      |
|--------------|---------------|---------------|---------------|---------------|---------------|
| Network      | PyramidNet-272-200 | Wide-ResNet-28-10 | Wide-ResNet-28-10 | Wide-ResNet-28-2 | ResNet50      |
| Epochs       | 900           | 200           | 200           | 200           | 180           |
| Batch Size   | 128           | 128           | 128           | 128           | 4096          |
| Image Preprocessing | mean-std-Normalize | mean-std-Normalize | mean-std-Normalize | Divide by 255 | None          |
| Augmentations | [pad-and-crop, horizontal flip, RUA, Cutout] | [pad-and-crop, horizontal flip, RUA, Cutout] | [pad-and-crop, horizontal flip, RUA, Cutout] | [RUA, Cutout] | [random resized crop, horizontal flip, RUA] |
| Optimizer    | SGD           | SGD           | SGD           | SGD           | SGD           |
| Weight Decay | $1e^{-4}$     | $5e^{-4}$     | $5e^{-4}$     | $5e^{-4}$     | $1e^{-4}$     |
| Initial LR   | 0.1           | 0.1           | 0.1           | 0.1           | 1.6           |
| LR Schedule  | Cosine Decay  | Cosine Decay  | Cosine Decay  | Cosine Decay  | $\times 0.1$ at epoch 60, 120, and 160 |
| Momentum     | 0.9           | 0.9           | 0.9           | 0.9           | 0.9           |
| $N_{max}$    | 5             | 5             | 5             | 5             | 5             |
| Best $r$     | 0.867         | 0.6           | 0.733         | 0.8           | 0.666         |

Table 2: Experiment parameter details. Note that PyramidNet uses ShakeDrop regularization for consistency with the RandAugment experimental setup.

| Methods | Search Space Order | CIFAR10 | CIFAR100 | SVHN (Core) | ImageNet |
|---------|--------------------|---------|----------|-------------|----------|
|         | 10^{32}            | 98.5    | 97.4     | 83.6        | 98.0     |
| Baseline| -                  | 97.3    | 96.1     | 81.2        | 96.7     | 76.3     |
| AA      | 10^{32}            | 98.5    | 97.4     | 83.3        | 98.3     | 77.6     |
| Fast AA | 10^{12}            | 98.5    | 97.3     | -           | 77.6     |
| PBA     | 10^{11}            | 98.5    | 97.4     | -           | -        |
| RA      | 10^{32}            | 98.5    | 97.3     | 83.3        | 98.3     | 77.6     |
| RUA     | 10                 | 98.5    | 97.4     | 83.6        | 98.0     | 77.7     |

Table 3: Experimental results for RUA compared with previous works. We report our average test accuracy over 10 independent runs (as in prior works). Best values in bold.
|   | Aligned | Random | Expanded | Accuracy |
|---|--------|--------|----------|----------|
| 1 | 0      | 0      | 0        | 0.916    |
| 2 | 0      | 0      | 1        | 0.912    |
| 3 | 0      | 1      | 0        | 0.917    |
| 4 | 0      | 1      | 1        | 0.920    |
| 5 | 1      | 0      | 0        | 0.917    |
| 6 | 1      | 0      | 1        | 0.915    |
| 7 | 1      | 1      | 0        | 0.920    |
| 8 | 1      | 1      | 1        | 0.922    |

Table 4: An ablation study of the RUA design decisions from Section 2.3. A ResNet9 architecture was trained on Cifar10, with accuracies averaged over 10 independent runs. ‘Aligned’ indicates our modifications to the “≈” transforms in Table 1, ‘Random’ indicates our use of a random uniform distribution, and ‘Expanded’ indicates the use of expanded augmentation parameters. Row 1 is thus analogous to running RandAugment using our dimensionality reduction and golden section search routine, and row 8 is the full RUA method.

4 Conclusion
In this work, we proposed Random Unidimensional Augmentation (RUA), an automated augmentation method providing several benefits relative to previous state-of-the-art algorithms. Our search space is one order of magnitude smaller than prior works, our transformations are more effective, and we leverage more efficient search algorithms. As a result of these improvements, RUA achieves equivalent results while requiring significantly less computation. We experimentally demonstrated RUA’s strength on the same tasks used by previous works across various network architectures and datasets. Not only that, unlike previous methods, RUA does not rely on any problem-specific human expertise, making the method truly automated and thus fit for use in conjunction with larger autoML pipelines.

References
Bochkovskiy, A.; Wang, C.; and Liao, H. M. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. CoRR, abs/2004.10934.
Cubuk, E. D.; Zoph, B.; Mané, D.; Vasudevan, V.; and Le, Q. V. 2019. AutoAugment: Learning Augmentation Strategies From Data. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, 113–123. Computer Vision Foundation / IEEE.
Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. 2020. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.
Deng, J.; Dong, W.; Socher, R.; Li, Li.; Li, K.; and Li, F. 2009. ImageNet: A large-scale hierarchical image database. In 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, 248–255. IEEE Computer Society.
Devries, T.; and Taylor, G. W. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. CoRR, abs/1708.04552.
Han, D.; Kim, J.; and Kim, J. 2017. Deep Pyramidal Residual Networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, 6307–6315. IEEE Computer Society.
He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 770–778. IEEE Computer Society.
Ho, D.; Liang, E.; Chen, X.; Stoica, I.; and Abbeel, P. 2019. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. In Chaudhuri, K.; and Salakhutdinov, R., eds., Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, 2731–2741. PMLR.
Kiefer, J. 1953. Sequential Minimax Search for a Maximum. Proceedings of the American Mathematical Society, 4(3): 502–506.
Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Master’s thesis, Department of Computer Science, University of Toronto.
Lim, S.; Kim, I.; Kim, T.; Kim, C.; and Kim, S. 2019. Fast AutoAugment. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 6662–6672.
Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. NIPS Deep Learning and Unsupervised Feature Learning Workshop.
Tan, M.; and Le, Q. V. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, 6105–6114. PMLR.
Yamada, Y.; Iwamura, M.; Akiba, T.; and Kise, K. 2019. Shakedrop Regularization for Deep Residual Learning. *IEEE Access*, 7: 186126–186136.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In Wilson, R. C.; Hancock, E. R.; and Smith, W. A. P., eds., *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press.