Genetic Learning for Designing Sim-to-Real Data Augmentations

Bram Vanherle BRAM.VANHERLE@UHASSELT.BE
Nick Michiels NICK.MICHIELS@UHASSELT.BE
Frank Van Reeth FRANK.VANREETH@UHASSELT.BE
Hasselt University - tUL
Flanders Make,
Expertise Centre for Digital Media

Reviewed on OpenReview:
https://openreview.net/group?id=ICLR.cc/2024/Workshop/DMLR

Abstract

Data augmentations are useful in closing the sim-to-real domain gap when training on synthetic data. This is because they widen the training data distribution, thus encouraging the model to generalize better to other domains. Many image augmentation techniques exist, parametrized by different settings, such as strength and probability. This leads to a large space of different possible augmentation policies. Some policies work better than others for overcoming the sim-to-real gap for specific datasets, and it is unclear why. This paper presents two different interpretable metrics that can be combined to predict how well a certain augmentation policy will work for a specific sim-to-real setting, focusing on object detection. We validate our metrics by training many models with different augmentation policies and showing a strong correlation with performance on real data. Additionally, we introduce GeneticAugment, a genetic programming method that can leverage these metrics to automatically design an augmentation policy for a specific dataset without needing to train a model.

Keywords: Computer Vision, Data Augmentation, Sim-to-Real, Synthetic Data

1 Introduction

Modern deep learning architectures can solve increasingly complex computer vision problems like object detection, segmentation, and pose estimation. These techniques, however, require large amounts of labeled training data to function properly. Manually capturing and annotating training images is a cumbersome task. Additionally, humans introduce biases and errors into the data, which can lead to worse downstream performance Northcutt et al. (2021). For this reason, synthetic data has become more popular for training machine learning models. Images are rendered from a scene representation, and the annotations are derived from that representation. This way, the costly annotation step is avoided. One popular approach is leveraging graphics rendering engines Borkman et al. (2021); Greff et al. (2022).

A major downside of synthetic data is that rendered images look slightly different from real images. This is mostly due to the difficulty of exactly simulating physical lighting. Although these rendered images look good to the human eye, there are still subtle differences.

©2023 Vanherle et al..

Neural networks are sensitive to this domain shift. When trained on synthetic data, they tend to perform significantly worse on real data Peng et al. (2018). The larger this domain shift is, the larger the performance gap.

Much research has been done to investigate ways to overcome this domain gap. Some have proposed widening the domain of the synthetic data by introducing extreme randomization during rendering in a technique called Domain Randomization Tobin et al. (2017); Tremblay et al. (2018). Others have tackled this problem by increasing photorealism Movshovitz-Attias et al. (2016); Roberts et al. (2021) or by modeling the real domain more accurately Wood et al. (2021). Further techniques include refining the synthetic images to match the real images better. This is done based on image statistics Abramov et al. (2020) or using learning-based approaches Zhao et al. (2023a). Domain Adaptation techniques are also used to overcome the sim-to-real domain gap Zhao et al. (2023b).

Data augmentation is a technique that alters images from the training set to create new images, thus enlarging the training set and potentially increasing the performance of the neural network Mikołajczyk and Grochowski (2018). Since it introduces randomness and widens the training data distribution, data augmentation is also very useful for training on synthetic data Carlson et al. (2019); Pashevich et al. (2019). Augmentation strategies can be manually designed, sampled from a predefined structure Cubuk et al. (2020); Müller and Hutter (2021), or generated automatically for a given problem Cubuk et al. (2019); Ho et al. (2019).

Many different individual augmentation operations exist, such as cropping, rotating, blurring, and adding noise. When faced with a specific sim-to-real scenario, it is unclear what combination of augmentations will work well for that dataset. This paper presents two metrics that explain why certain augmentations work well for specific scenarios. We argue that a good augmentation for sim-to-real decreases the distance to the real dataset while increasing the variation in the training data. We train a large amount of object detection models using different augmentation policies and show that our proposed metrics are strongly correlated with downstream performance. Finally, we show that these metrics can guide a genetic programming algorithm in designing an augmentation strategy automatically without needing to train any models. We compare our results to existing augmentation methods and to established domain adaptive object detection models. Code is available at: `https://github.com/EDM-Research/genetic-augment`.

2 Method

We consider an object detection neural network f. The network has a backbone b that computes a feature map and a detection head h that generates detections from the features computed by b. This neural network is trained on a set of synthetic images $\mathcal{X}^S = \left\{x_i^S\right\}^{N_S}$ and their accompanying labels $\mathcal{Y}^S = \left\{y_i^S\right\}^{N_S}$. Training is done by optimizing the weights of f to minimize the object detection loss $\mathcal{L}$ over the synthetic data:

$$\min_{\theta} \frac{1}{N_S} \sum_{i=0}^{N_S} \mathcal{L}(a(x_i^S), y_i^S) \tag{1}$$

with a the augmentation strategy. The goal is to select a to optimize the performance of the trained network f on the real target data $(\mathcal{X}^T, \mathcal{Y}^T)$. We measure this performance

using mean average precision (mAP) Everingham et al. (2010) averaged over the 0.5 to 0.95 threshold range. We aim to select the augmentation strategy a unsupervised; i.e., only $\mathcal{X}^S$ and $\mathcal{X}^T$ are used to guide the selection. For this work, we focus on the augmentations that operate on pixel values and do not affect the object detection annotations.

2.1 Metrics

A good augmentation should increase the variation in the training data. A model trained on more diverse data is more likely to generalize to unseen data. For sim-to-real specifically, a good augmentation will also make the synthetic training images more similar to the real data. This could be done by, for example, introducing some noise or changing the image's brightness. For humans, it is difficult to estimate whether an augmentation will increase variation or decrease distance. Especially considering an entire dataset. For this reason, we introduce two metrics that can estimate these properties for a proposed augmentation policy given a synthetic and real dataset. Metrics are computed in the feature space. A backbone b is used that is pre-trained on ImageNet Deng et al. (2009) so that no models need to be trained to find an optimal augmentation setup.

To measure the variation of an augmented dataset, we compute the average of the per-feature variance over the features computed over a set of augmented synthetic images. For each synthetic image, the feature map $\mathbf{h}^S$ is computed after applying the augmentation $b(a(x_i^S)) = \mathbf{h}^S$. All feature maps are flattened to a one-dimensional vector of size N_d to form the set of feature vectors for the synthetic images H^S. The variance of a feature point is computed over the dataset $Var(H_d^S)$. This is averaged over all feature points $d \in [0, 1, \ldots, N_d]$.

To measure the distance between the augmented synthetic images and the real images, we measure the Wasserstein-1 distance in feature space on a per-feature basis. Feature maps $\mathbf{h}^T$ are computed for each real image. This is done using the same backbone as the synthetic images but without applying the augmentation $b(x_i^T) = \mathbf{h}^T$. The first Wasserstein distance is computed between the distributions of feature point d for both the synthetic and real features $W(H_d^T, H_d^S)$. This is averaged over all feature points $d \in [0, 1, \ldots, N_d]$.

Yamaguchi et al. (2019) proposed similar metrics to find an optimal additional dataset for training a multi-domain learning GAN to create additional training samples. They measure the distance between the target and additional dataset using the Frechet Inception Distance (FID) Heusel et al. (2017). Additionally, they use Multi-scale Structural Similarity Wang et al. (2003) among the images of the additional dataset as a measure of variation in the data. We find that our metrics are better suited to predicting model performance given a specific augmentation, as detailed in Appendix B. Additionally, our proposed metrics are much faster, as multi-scale SSIM and FID are computationally expensive. This is important as the Genetic Learning procedure presented in the next section requires many evaluations of these metrics.

2.2 Genetic Learning

Although these metrics offer more explainability, it is still difficult to manually design augmentations to satisfy these metrics. For this reason, we introduce GeneticAugment, an algorithm that can automatically learn an augmentation policy that does well on these met-

rics given a synthetic and real target dataset. Specifically, we aim to learn an augmentation policy a consisting of a set of individual augmentations $a = [a_1, a_2, \ldots, a_N]$ executed sequentially or at random. Each individual augmentation is defined by the selected augmentation method, a strength value s, and a probability p. The sequential nature of the augmentation strategies lends itself well to genetic algorithm learning Katoch et al. (2020); Fortin et al. (2012). This also allows for much flexibility in designing augmentation strategies, as it does not require differentiation.

To find an augmentation policy through genetic learning, we spawn a population of 100 random augmentation policies. These can all have a random or a fixed length and consist of several random individual augmentations with a random strength and probability. In each generation, 200 offspring of the previous generation are created. Offspring is produced by crossing over two individuals of the previous generation or by mutating individuals. When two individuals are crossed over, this is done using a one-point crossover. Mutating an individual is done by replacing some of its augmentations with another random augmentation. Augmentation policies can also be extended, or an augmentation can be randomly removed. When the offspring is created, the offspring's and population's fitness is computed. The fitness is defined by positive variance and negative distance, as defined in the previous section. To optimize multiple objectives, NSGA-II Deb et al. (2002) is used to select 100 non-dominated individuals for the next generation. Non-dominated individuals from all generations are kept in a Pareto front.

3 Experiments

We train many object detection models using different augmentation techniques and settings to show the large difference between the performances of different augmenters. We train each model to detect cars on the synthetic Sim10k Johnson-Roberson et al. (2017) dataset and test it on the real Cityscapes Cordts et al. (2016) dataset. We train a MaskRCNN He et al. (2017) model with a ResNet-18 He et al. (2016) backbone. For more details, refer to Appendix A.

We test 27 different augmentation techniques. These techniques fall under four categories: blurring, color augmentation, noise, and sharpening. We test each technique at three different strengths: half of the default strength ($s_{0.5}$), default strength ($s_{1.0}$), and double strength ($s_{2.0}$). We also test each technique when only applying it randomly half the time ($p_{0.5}$). This totals to $27 \times 4 = 108$ models trained and a baseline model without augmentation. A complete list of augmentations with parameters can be found in Appendix A.

An overview of the results of these experiments is shown in Fig. 1. We notice that most augmentations lead to a better generalizing model. However, a lot of them do not improve the model by much. Some augmentation policies greatly improve the model, and some make the model way worse. In general, blurring augmentations and randomly applied augmentations do slightly better than other methods. It is difficult to predict why some augmentations are better than others. This highlights the need for explainable metrics to examine augmentations. More detailed results are given in Appendix B.

Usually, multiple augmentations are combined to form an augmentation policy. To examine this scenario, we combine multiple augmentations based on their performance in the previous experiment. Specifically, we combine the n best and n worst augmentation methods

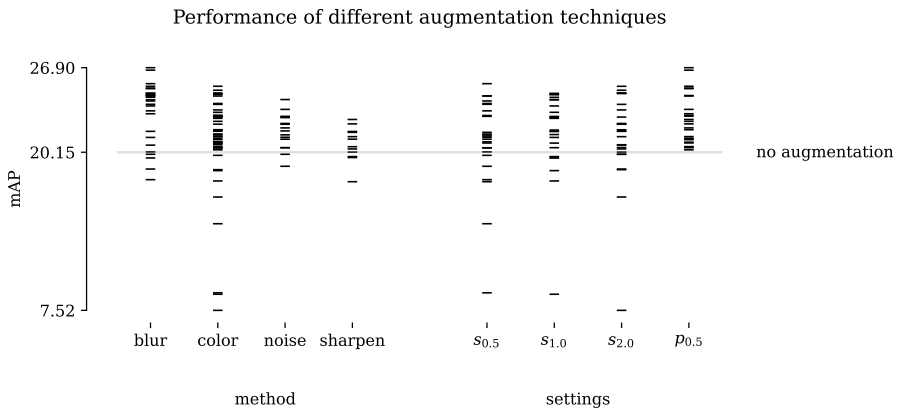

Figure 1: Performance of different augmentations on Sim10k → Cityscapes.

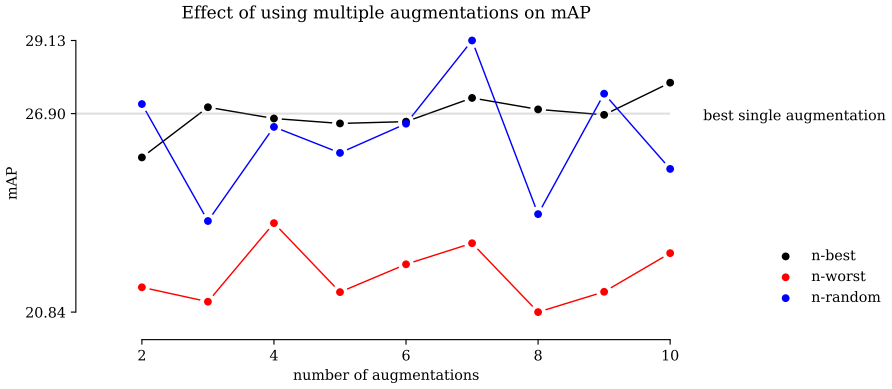

Figure 2: Performance combinations of augmentations on Sim10k → Cityscapes.

based on their performance in the $p_{0.5}$ category. As a baseline, we also combine n random augmentations. We test for $n \in [2, 10]$. Each augmentation is executed at default strength and $1/n$ probability. This way, we examine if combining good single augmentations leads to an even better augmentation strategy or whether multiple bad augmentations can form a good augmentation strategy together. The results are shown in Fig. 2. Augmentations are combined by executing them sequentially.

These results show that combining multiple good single augmentations leads to augmentation policies that struggle to outperform the best single augmentation. Combining multiple random augmentation methods results in widely varying results. The best augmentation policy we have found so far results from combining seven random augmenters. Finally, combining multiple weak augmentations slightly improves performance but comes nowhere near the best single augmentation. The performance of single augmentations cannot be used as a guide to create augmentation strategies. For this reason, we need some reliable metrics that can be used to guide the design of augmentation strategies.

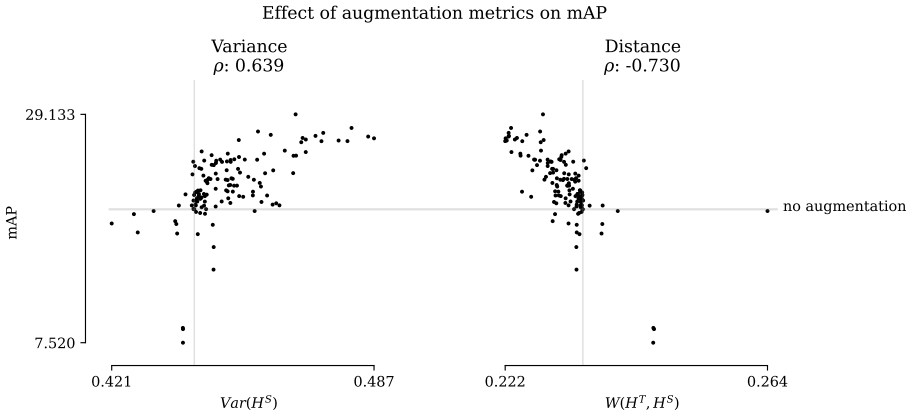

Figure 3: Relationship between the proposed metrics measured for augmentations and performance of models trained with those augmentations on Sim10k → Cityscapes.

3.1 Validation of metrics

To show the predictive power of the proposed metrics, we measure them for each model trained in the previous section. Measurement is done with a ResNet-18 pre-trained on ImageNet. This is the same backbone that the object detection models use. Metrics are measured using 2048 samples from the synthetic and real training data. We measure the correlation between the metric and the model performance using Spearman's rank correlation coefficient ρ Spearman (1904).

Fig. 3 shows the measured metrics for each augmentation plotted versus the performance of the model trained with that augmentation. There is a strong correlation between the variance and distance metrics and the model's performance on the real data. An increase in augmentation variance leads to a better generalizing model. Eventually, the increase in mAP tapers off when variation increases. Generally, when an augmentation policy decreases the distance between the real and synthetic data in feature space, it improves the model's performance on real data.

4 Genetic Learning Results

Having shown a strong correlation between the proposed metrics and the performance of augmentation strategies, we now use them to automatically design augmentation strategies for Sim10k → Cityscapes using the genetic learning procedure described in Section 2.2. For efficiency, we now use only 128 samples to evaluate an individual. Five generations are created using the genetic learning approach to find an augmentation strategy. Finally, the augmentation strategy with the lowest distance value is selected. An object detection model is trained following that policy. To ensure fair evaluation, only images of the training sets of both datasets are used to find augmentations.

Table 1: Performance of GeneticAugment policies on Sim10k → Cityscapes for different strategies compared to other methods using those strategies.

Strategy	**Method**	**mAP**
None	No augmentation	20.15
OneOf (single)	TrivialAugment Müller and Hutter (2021)	25.00
	GeneticAugment (Ours)	**26.97**
OneOf (double)	AutoAugment Cubuk et al. (2019)	24.90
	GeneticAugment (Ours)	**25.15**
TwoOf	RandAugment Cubuk et al. (2020)	23.97
	GeneticAugment (Ours)	**29.47**
Sequential	GeneticAugment (Ours)	27.47

4.1 Comparison to other augmentation methods

As a benchmark, we compare our model against other augmentation methods. These augmentation methods use different strategies. TrivialAugment selects one random augmentation from a set each time. AutoAugment learns a set of augmentation pairs, of which one pair is randomly selected for each image. For AutoAugment we use the strategy learned for ImageNet and do not retrain it for our dataset. RandAugment picks two random augmentations from a set. We train a variant of GeneticAugment that follows each of these strategies for a better comparison. We also learn a sequential strategy that executes each augmentation according to its probability. To maintain the focus on pixel domain augmentations, we remove augmentations that do not preserve the annotations, such as rotation and translation.

Table 1 shows that policies learned by GeneticAugment outperform random policies proposed by other methods for all different strategies. Also, the sequential strategy shows strong performance. This indicates that for sim-to-real, there is a clear benefit to using a learned policy over random policies and that the proposed metrics can guide the discovery of a policy. GeneticAugment offers an efficient method for finding such policies, as it does not require training models. These results also highlight the versatility of the genetic algorithm approach combined with the variance and distance metrics, as it can be applied to multiple strategies. It should be noted that GeneticAugment can choose from a wider set of augmentation techniques than the other methods. The evolution of the metrics during the training of the TwoOf strategy is shown in Fig. 4. As expected, we observe the variance in the population increasing while the distance decreases. The average length of the augmentation sequences increases during the learning. Specifics of the learned augmentation strategies are shown in Appendix B.

4.2 Comparison to Domain Adaptive Object Detection techniques

Our method leverages unlabeled images from the target domain to improve the performance of an object detection model trained on a source domain. For this reason, we also compare

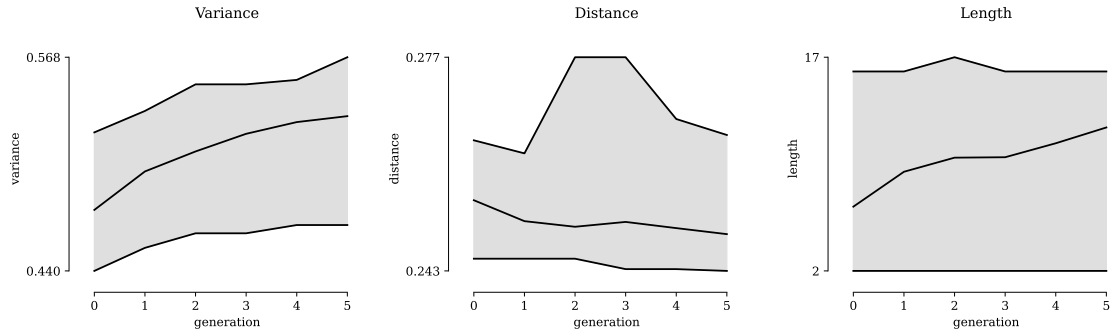

Figure 4: Evolution of the range of the metrics measured on the population during the generations of the genetic algorithm.

Table 2: Performance of different Domain Adaptive Object Detection techniques on Sim10k → Cityscapes.

Method	**Detector**	**AP**$_{50}$
Source	FasterRCNN-VGG16	42.8
ViSGA Rezaeianaran et al. (2021)	FasterRCNN-ResNet50	49.3
KTNet Tian et al. (2021)	FasterRCNN-VGG16	50.7
GeneticAugment (Ours)	FasterRCNN-VGG16	54.8
PT Chen et al. (2022)	FasterRCNN-VGG16	**55.1**

our technique to domain adaptive object detection techniques. Whereas these methods often introduce new training techniques and novel architectures, our approach is model-agnostic and works by only changing the input data through augmentations.

Following other works in the field, we train a FasterRCNN Ren et al. (2015) with a VGG16 Simonyan and Zisserman (2015) backbone. During training, we freeze the first three blocks of the backbone, as is commonly done for synthetic data Hinterstoisser et al. (2019). The TwoOf policy of the previous experiment is used during training, as it is the best-performing strategy. We also train an object detection in the same setup without augmentation as a benchmark. We report AP with a threshold of 0.5 instead of the mAP of previous experiments since this is the standard for this benchmark.

Table 2 shows the performance of GeneticAugment compared to recent works using the same architecture. We observe that the model trained with GeneticAugment outperforms most other approaches. This shows that data-driven augmentation design can obtain good sim-to-real performance without changing model architectures. Additionally, we can conclude that the designed augmentations are not limited to the backbone used to find them, as a ResNet-18 was used to find augmentations and VGG16 was used in the detection model. Note that recent works have increased the state of the art on this benchmark to 62.0 AP Zhao et al. (2023b) using the more powerful Deformable DETR architecture Zhu

et al. (2021). Since our approach is model-agnostic, it can also be applied to more modern architectures.

5 Conclusion

We showed through extensive experiments that there is a wide variety in the performance of augmentation policies for sim-to-real training, with most leading to only small benefits. To explain this difference in performance and to guide the design of augmentation policies, we presented two augmentation metrics that show a high correlation with the performance of a model trained using that augmentation. These metrics are computed using the synthetic training data and real target data. It is difficult to tune a data augmentation policy to satisfy these metrics manually. For this reason, we introduced a genetic learning algorithm that can automatically design an augmentation policy that does well on these metrics. The algorithm finds policies in a data-driven manner, so no models need to be trained. This approach leads to good sim-to-real performance, outperforming non-data-driven augmentation techniques. Additionally, it performs well compared to domain adaptive object detection methods.

Reproducibility Statement

In Appendix A, we provide a detailed description of how the object detection models throughout the paper are trained. We also describe exactly how the augmentations are done and with what parameters. This appendix also details how the genetic programming algorithm is implemented. For full reproducibility, code can be found at `https://github.com/EDM-Research/genetic-augment`. The synthetic Sim10k Johnson-Roberson et al. (2017) dataset and the real Cityscapes Cordts et al. (2016) dataset are publicly available for downloading. This should further encourage reproducibility.

Broader Impact Statement

The advancement of synthetic data can make computer vision AI more easily accessible to those unable to fund large-scale labeling operations. This can have positive societal effects, as smaller companies and regular people can benefit from the progress in AI. As it lowers the bar for entry for good actors, it might also do the same for bad actors. We should always be conscious of this when communicating our research.

References

A. Abramov, C. Bayer, and C. Heller. Keep it simple: Image statistics matching for domain adaptation, 2020.

S. Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi, C. Romero, W. Smith, A. Thaman, S. Warren, and N. Yadav. Unity perception: Generate synthetic data for computer vision. *CoRR*, abs/2107.04259, 2021. URL `https://arxiv.org/abs/2107.04259`.

A. Buslaev, A. Parinov, E. Khvedchenya, V. Iglovikov, and A. Kalinin. Albumentations: fast and flexible image augmentations. *ArXiv e-prints*, 2018.

A. Carlson, K. A. Skinner, R. Vasudevan, and M. Johnson-Roberson. Sensor transfer: Learning optimal sensor effect image augmentation for sim-to-real domain adaptation. *IEEE Robotics and Automation Letters*, 4(3):2431–2438, 2019.

M. Chen, W. Chen, S. Yang, J. Song, X. Wang, L. Zhang, Y. Yan, D. Qi, Y. Zhuang, D. Xie, et al. Learning domain adaptive object detection with probabilistic teacher. In *International Conference on Machine Learning*, pages 3040–3055. PMLR, 2022.

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. 2019. URL `https://arxiv.org/pdf/1805.09501.pdf`.

E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf`.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.

F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, T. Kipf, A. Kundu, D. Lagun, I. Laradji, H.-T. D. Liu, H. Meyer, Y. Miao, D. Nowrouzezahrai, C. Oztireli, E. Pot, N. Radwan, D. Rebain, S. Sabour, M. S. M. Sajjadi, M. Sela, V. Sitzmann, A. Stone, D. Sun, S. Vora, Z. Wang, T. Wu, K. M. Yi, F. Zhong, and A. Tagliasacchi. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3749–3761, June 2022.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On pre-trained image features and synthetic images for deep learning. In *Computer Vision – ECCV 2018 Workshops: Munich, Germany, September 8-14, 2018, Proceedings, Part I*, page 682–697, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-11008-6. doi: 10.1007/978-3-030-11009-3_42. URL https://doi.org/10.1007/978-3-030-11009-3_42.

D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2731–2741. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ho19b.html.

M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 746–753. IEEE, 2017.

S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80:8091 – 8126, 2020.

T. maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016.

A. Mikołajczyk and M. Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122, 2018. doi: 10.1109/IIPHDW.2018.8388338.

Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh. How useful is photo-realistic rendering for visual learning? In G. Hua and H. Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 202–217, Cham, 2016. Springer International Publishing. ISBN 978-3-319-49409-8.

S. G. Müller and F. Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 774–782, October 2021.

C. G. Northcutt, A. Athalye, and J. Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL `https://openreview.net/forum?id=XccDXrDNLek`.

A. Pashevich, R. Strudel, I. Kalevatykh, I. Laptev, and C. Schmid. Learning to augment synthetic images for sim2real policy transfer. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2651–2657. IEEE, 2019.

X. Peng, B. Usman, K. Saito, N. Kaushik, J. Hoffman, and K. Saenko. Syn2real: A new benchmark for synthetic-to-real visual domain adaptation, 2018.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf`.

F. Rezaeianaran, R. Shetty, R. Aljundi, D. O. Reino, S. Zhang, and B. Schiele. Seeking similarities over differences: Similarity-based domain alignment for adaptive object detection. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9184–9193, 2021.

M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. URL `https://arxiv.org/pdf/2011.02523.pdf`.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.1556`.

C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. ISSN 00029556. URL `http://www.jstor.org/stable/1412159`.

K. Tian, C. Zhang, Y. Wang, S. Xiang, and C. Pan. Knowledge mining and transferring for domain adaptive object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9113–9122, 2021. doi: 10.1109/ICCV48922.2021.00900.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 23–30. IEEE Press, 2017. doi: 10.1109/IROS.2017.8202133. URL `https://doi.org/10.1109/IROS.2017.8202133`.

J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977, 2018.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Z. Wang, E. Simoncelli, and A. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003. doi: 10.1109/ACSSC.2003.1292216.

E. Wood, T. Baltrušaitis, C. Hewitt, S. Dziadzio, T. J. Cashman, and J. Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3681–3691, October 2021.

S. Yamaguchi, S. Kanai, and T. Eda. Effective data augmentation with multi-domain learning gans. In *AAAI Conference on Artificial Intelligence*, 2019.

G. Zhao, T.-L. Shen, S. You, and C.-C. J. Kuo. Unsupervised synthetic image refinement via contrastive learning and consistent semantic-structural constraints. In *Defense + Commercial Sensing*, 2023a. URL `https://api.semanticscholar.org/CorpusID:258309171`.

Z. Zhao, S. Wei, Q. Chen, D. Li, Y. Yang, Y. Peng, and Y. Liu. Masked retraining teacher-student framework for domain adaptive object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19039–19049, October 2023b.

X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Appendix A. Implementation Details

A.1 Object Detection Details

The object detection experiments use a MaskRCNN He et al. (2017) architecture with a ResNet-18 He et al. (2016) backbone. The backbone is initialized with weights trained on ImageNet Deng et al. (2009). All weights are retrained during training. We use a batch size of four and a learning rate of 0.00005. Following the convention in unsupervised domain adaptation, we resize each image so that the shortest size is 600 pixels. A lot of models needed to be trained during this research. To keep the total cost of training reasonable, we have chosen a short training program of 50 epochs, showing the model 1 000 images each epoch. This was enough for all models to nearly reach convergence and to compare the different augmentations.

A.2 Augmentation Details

For all augmentations, we used the Albumentations library Buslaev et al. (2018) implementation. For each augmentation, a default value was chosen. This was usually the library's default value. What default value and which function was used is detailed in Table 3. Throughout the paper, augmentation strengths were defined by a strength value. Practically, all parameter values for an augmentation were scaled by this value. In the case of a tuple, only the second value was scaled by the strength value, unless this would make that value smaller than the first value of the tuple. In the case of methods employing a kernel, it is enforced that the value remains odd. The parameter was scaled inversely for the posterize augmentation, as decreasing the number of bits strengthens the augmenter.

For the baselines TrivialAugment, AutoAugment, and RandomAugment, we used the implementations available in Torchvision maintainers and contributors (2016) at their default settings. To compare with our pixel-based method, we removed the non-pixel-based augmentations: `ShearX`, `ShearY`, `TranslateX`, `TranslateY` and `Rotate`.

A.3 Metric Computation Details

The variance is computed using the Numpy Harris et al. (2020) `var` function. With `features` the list of all features computed over the chosen sample of the synthetic data:

```
def variance(features: np.array):
    return np.mean(np.var(features, axis=0))
```

The Wasserstein distance is computed using the `wasserstein_distance` function of the Scipy library Virtanen et al. (2020). With `features` the list of all features computed over the chosen sample of the synthetic data and `reference_features` the list of all features computed over the chosen sample of the real data:

```
def wasserstein_distance(features: np.array, reference_features: np.array):
    d = []
    for i in range(features.shape[1]):
        d.append(stats.wasserstein_distance(reference_features[:, i], features[:, i]))

    return np.mean(d)
```

A.4 Genetic Learning Details

Genetic learning is done using the DEAP library Fortin et al. (2012).

Augmentations are spawned with a random length unless a fixed size is provided. This random length is sampled from a normal distribution truncated between 2 and 16. The distribution has a mean of 6 and a standard deviation of 5. Each individual is a random augmentation sampled from the pool of all augmentations parameterized by a strength value and probability. Probability is sampled between 0 and 1, and strength between 0 and 2. For the OneOf double strategy, each element in an individual consists of two random augmentations.

A mutation is done with a 30% probability. When an individual is mutated, each of its elements has a 10% chance to be swapped out by a different random augmentation. If a fixed length is not specified, there is also a 10% chance a random augmentation is added at the back or that one random augmentation from the set is removed.

Crossover happens with a 60% probability. Two individuals are crossed using a one-point method. The two individuals are cut in half randomly, and the other halves are swapped between individuals.

Appendix B. Detailed Results

Table 4 reports the individual performance of models trained on Sim10k using the different augmentation settings. Results are reported both on the Cityscapes test set, and on the Cityscapes test set processed by the same augmentation used during training. As noted in the main paper, augmentations with no randomness built in, such as invert, do badly on the test set. However, we observe this can be overcome by applying that augmentation to the test set.

Table 3: Details of what Albumentations function was used during this research at which default value.

Augmenter	**Paramter**	**Value**
GaussianBlur	blur_limit	$(3, 15)$
Blur	blur_limit	7
Defocus	radius	$(3, 10)$
GlassBlur	sigma	0.7
MedianBlur	blur_limit	7
MotionBlur	blur_limit	7
ZoomBlur	max_factor	1.09
ChannelShuffle		
CLAHE	clip_limit	4
ColorJitter (brightness)	brightness	0.3
	others	0.0
ColorJitter (contrast)	contrast	0.3
	others	0.0
ColorJitter (saturation)	saturation	0.3
	others	0.0
ColorJitter (hue)	hue	0.3
	others	0.0
Equalize		
FancyPCA	alpha	0.3
InvertImg		
Posterize	num_bits	4
RandomGamma	gamma_limit	$(80, 120)$
RandomToneCurve	scale	0.1
Emboss	strength	$(0.2, 0.5)$
Sharpen	alpha	$(0.2, 0.5)$
UnsharpMask	blur_limit	$(3, 15)$
	alpha	0.5
GaussNoise	var_limit	75.0
ISONoise	color_shift	$(0.01, 0.05)$
	intensity	$(0.1, 0.5)$
MultiplicativeNoise	multiplier	0.9
PixelDropout	dropout_prob	0.01
UniformNoise	strength	0.2

Table 4: Individual performance of each augmentation on Sim10k → Cityscapes under different settings. Performance is reported in mAP of the test set and on the test set with that augmentation applied.

	Test				**Augmented Test**			
Augmentation	$s_{0.5}$	$s_{1.0}$	$s_{2.0}$	$p_{0.5}$	$s_{0.5}$	$s_{1.0}$	$s_{2.0}$	$p_{0.5}$
None	20.15				20.15			
gaussian blur	23.98	24.33	24.85	26.70	23.30	23.11	21.51	25.97
blur	23.45	24.53	24.84	24.64	21.66	23.22	22.38	23.84
defocus	24.24	21.33	20.71	26.90	23.10	19.86	16.73	24.48
glass blur	17.96	19.69	18.80	25.23	16.41	16.77	18.09	22.96
median blur	24.65	24.75	25.41	24.68	23.13	23.59	22.55	24.09
motion blur	25.62	24.85	21.81	25.27	24.80	23.62	19.31	24.65
zoom blur	20.17	23.84	19.98	23.22	20.17	20.04	9.56	22.16
channel shuffle	23.08	21.82	24.88	23.03	22.81	21.74	24.81	22.74
clahe	20.50	17.85	18.75	20.89	22.21	19.71	21.47	21.26
brightness	21.44	22.85	21.94	20.34	21.40	22.68	21.77	20.23
contrast	20.88	23.33	20.75	21.94	20.83	23.26	20.51	21.87
saturation	19.90	20.88	22.39	21.27	19.82	20.93	22.48	21.26
hue	24.63	24.81	23.97	25.42	24.51	24.55	23.72	25.27
equalize	14.44	18.68	16.57	21.40	22.82	24.35	22.48	22.58
fancy pca	21.60	23.01	25.10	23.09	21.55	22.97	25.12	23.07
invert	8.92	8.81	7.52	20.59	21.68	22.16	21.21	20.14
posterize	20.98	21.56	20.52	22.61	20.98	20.14	15.60	22.25
gamma	20.49	21.92	22.95	20.95	20.73	21.99	21.80	20.86
tone curve	24.04	24.76	23.53	21.15	24.00	24.69	23.15	21.10
gauss noise	21.53	24.36	21.85	21.19	20.85	23.81	20.99	20.87
iso noise	21.34	22.95	22.44	22.09	20.85	22.07	21.93	21.62
multiplicative noise	19.02	20.51	19.98	20.51	21.37	20.30	20.81	20.35
dropout	23.01	22.91	22.38	23.57	21.87	21.41	19.73	23.35
uniform noise	-	-	-	-	-	-	-	-
emboss	21.19	19.80	20.16	22.76	21.88	20.64	21.66	22.85
sharpen	17.79	19.72	20.41	22.41	21.04	23.06	22.48	22.56
unsharp mask	21.73	21.80	21.39	20.59	21.87	22.05	21.80	20.68
Average	21.02	21.76	21.30	22.79	21.76	22.03	20.91	22.43

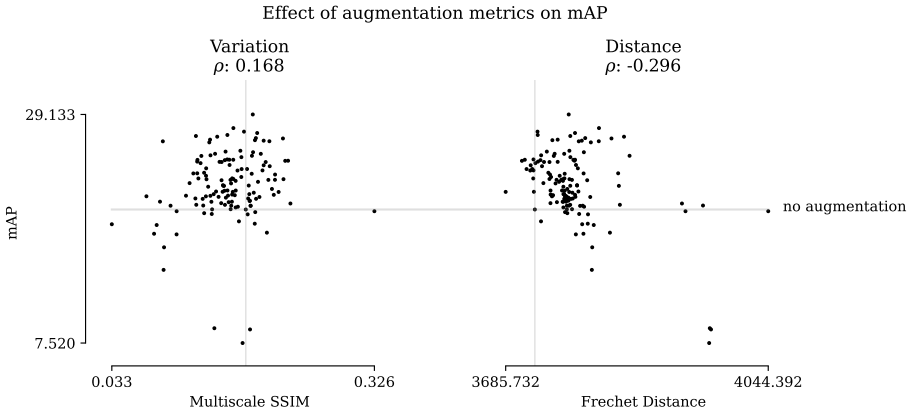

Figure 5: Relationship between the metrics proposed by Yamaguchi et al. (2019) and performance of models trained with different augmentations on Sim10k → Cityscapes.

B.1 Performance of other metrics

To validate the utility of our proposed metrics, we also use the metrics proposed by Yamaguchi et al. (2019) to predict the performance of a model trained on a specific augmentation. Yamaguchi et al. (2019) proposed measuring the variation by computing the multiscale structural similarity between all augmented images. They used the Frechet Inception Distance between the augmented and real images to measure distance. Instead of using the Inception network as a feature detector for the distance, we use ResNet, which is used for the other metrics and for training the object detection network.

The results in Fig. 5 show that there is a significantly lower correlation between these metrics and the downstream performance of the object detection model compared to our metrics. These metrics were originally designed to select a good additional dataset to train a GAN on and show good performance for this task. However, these metrics do not seem ideal for finding good image data augmentations. Potentially, because their variation metric operates in the pixel domain, whereas our metric measures variation in the feature space, which might be better to measure the impact on a downstream object detection model. Additionally, the Frechet Distance assumes the data is normally distributed, which is not necessarily true for the backbone features. Finally, our metrics are also much faster, which is necessary as the metrics need to be computed often during the genetic learning procedure.

B.2 Training sequential augmentations

We test the capabilities of our genetic learning approach by learning sequential augmentation strategies of different lengths. In augmentations of this strategy, each augmentation is executed sequentially by its given probability. We learn several augmentation policies of a fixed length $l \in [2, 4, 6, 8]$. We also learned an augmentation strategy that does not have a fixed length.

The results in Table 5 show that all augmentation strategies lead to a good increase in performance compared to using no augmentation. Even the sequence of only two augmenta-

Table 5: Performance of GeneticAugment learned sequential augmentation policies on Sim10k → Cityscapes of different lengths.

Augmentation	**mAP**
None	20.15
GeneticAugment-2	26.66
GeneticAugment-4	27.07
GeneticAugment-6	**27.48**
GeneticAugment-8	24.04
GeneticAugment-n	27.47

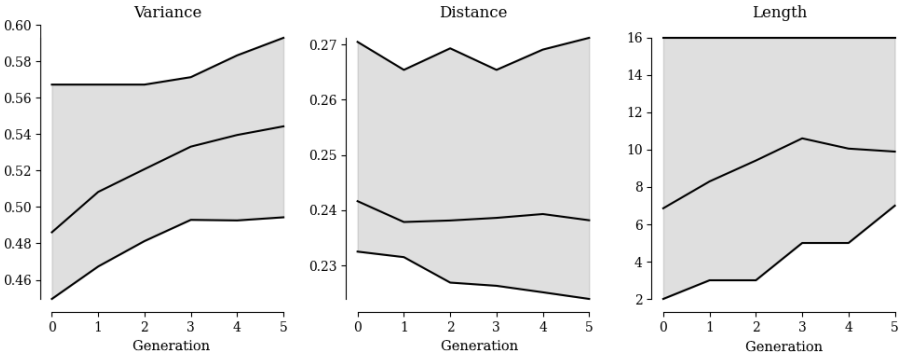

Figure 6: Evolution of the metrics measured from the individuals during multiple generations of the genetic algorithm. The range indicates the minimal and maximal values. The middle line is the average.

tions improves the mAP by more than six points. Most augmentation strategies outperform the best single augmentation found in Section 3, whereas the best multiple augmentation policy from that section is not matched. These are good results as that policy was found by combining seven random augmentations. Finding augmentations this way is unfeasible as it could require training many models before finding a good policy. Our approach, on the other hand, requires only inference of a pre-trained network.

Fig. 6 shows the progress of the metrics during the genetic learning process. The variance of the population clearly increases. The distance appears more difficult to optimize, but we observe the average and minimal individual decrease. The algorithm also prefers longer sequences over time. Fig. 7 shows the Pareto frontier after the last generation.

B.3 Learned Augmentations

In the main paper, several augmentation policies are learned following different strategies. In this section, we detail the exact policies. Table 6 shows the policy learned for the OneOf (single) strategy. Table 7 shows the policy learned for the OneOf (double) strategy. The

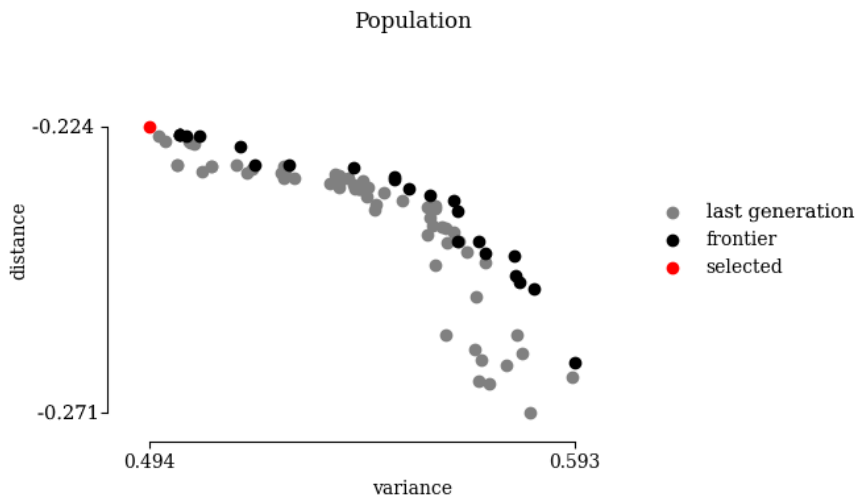

Figure 7: Pareto frontier after the last generation. Each dot represents an individual augmentation policy.

Table 6: Augmentation learned by GeneticAugment for Sim10k → Cityscapes using the OneOf (single) strategy.

Augmentation	s	p
glass blur	1.49	0.63
motion blur	1.10	0.85
uniform noise	1.41	0.96
defocus	1.80	0.79
hue	1.16	0.28
gaussian blur	0.41	0.30
defocus	0.87	0.60
uniform noise	0.17	0.96
brightness	1.77	0.05
median blur	0.29	0.45
hue	1.24	0.02

policy learned for TwoOf is shown in Table 8. Finally, the policy learned for the sequential policy is shown in Table 9.

Table 7: Augmentation learned by GeneticAugment for Sim10k → Cityscapes using the OneOf (double) strategy.

Augmentation	s	p
defocus	1.83	0.63
contrast	0.70	0.25
equalize	0.18	0.46
uniform noise	0.90	0.62

Table 8: Augmentation learned by GeneticAugment for Sim10k → Cityscapes using the TwoOf strategy.

Augmentation	s	p
equalize	0.91	0.58
uniform noise	1.59	0.11
multiplicative noise	0.17	0.55
channel shuffle	1.47	1.00
blur	1.12	0.37
gaussian blur	1.20	0.76

Table 9: Augmentation learned by GeneticAugment for Sim10k → Cityscapes using the Sequential strategy.

Augmentation	s	p
multiplicative noise	0.40	0.58
brightness	0.79	0.07
dropout	0.42	0.93
median blur	1.43	0.55
brightness	1.50	0.48
channel shuffle	0.35	0.38
channel shuffle	0.41	0.12
brightness	1.53	0.22
unsharp mask	1.16	0.64
iso noise	0.78	0.31
sharpen	1.22	0.73
contrast	1.83	0.96
posterize	0.48	0.35
median blur	0.88	0.61
fancy pca	1.14	0.66
median blur	1.00	0.05