Optimized Class-specific Data Augmentation for Plant Stress Classification

Nasla Saleem, Aditya Balu, Talukder Zaki Jubery, Arti Singh, Asheesh K. Singh, Soumik Sarkar, Baskar Ganapathysubramanian*

Iowa State University
*baskarg@iastate.edu

Abstract

Data augmentation has the potential to significantly improve the performance of deep learning-based image classifiers. However, a key challenge in applying data augmentation is choosing an effective set of augmentations from a large pool of candidates. Recently, automated augmentation strategies have produced state-of-the-art results for image classification. Most results have focused on improving the *total accuracy* of the classifier, often at the cost of reduced performance of a finite number of classes. We explore a Genetic Algorithm-based optimization to identify the ideal class-specific augmentations that maximize the *mean-per-class accuracy*, starting from a well-trained classifier (which serves as our baseline). We illustrate the utility of this strategy on a well-studied problem (and associated dataset) of classifying soybean leaf stresses. Our (preliminary) work indicated improvements over our baseline model and showed improvement in the *mean-per-class accuracy* from 90.68% to 93.11% across generations. Identifying class-specific augmentations can provide contextual information to end users. This approach is computationally less expensive than traditional Network-Architecture-Search (NAS), as we only seek to fine-tune the baseline classifier.

Introduction

Deep Learning models have performed remarkably well when trained on massive amounts of data. Data augmentation techniques have been widely used for generating additional data to improve the robustness of many computer vision tasks (Shorten and Khoshgoftaar 2019). These augmentations transform images to increase the diversity of the data. However, picking a good set of augmentations that efficiently improves the model performance and capture prior knowledge requires expertise and manual work in each domain. The recent developments in automated machine learning (AutoML) have led to automated search methods for augmentation policies on datasets and therefore have the potential to address some weaknesses of traditional data augmentation methods (Cubuk et al. 2018; Ho et al. 2019; Zoph et al. 2020; Lim et al. 2019). Among these, AutoAugment (Cubuk et al. 2018, 2020) stands out with state-of-the-art results in CIFAR-10 (Krizhevsky, Hinton, and others 2009), CIFAR-100 (Krizhevsky, Hinton, and others 2009), and ImageNet (Deng et al. 2009). In AutoAugment, the search for optimal augmentation policy for datasets is employed as a discrete problem using Reinforcement Learning (RL) as the search controller. However, the search method used in this work is computationally infeasible to run for most non-CS domain users. Furthermore, while these transformations encoding suitable invariances are intuitive for some classes, it may not be the same for all. For example, in MNIST, if you apply the transformation of vertical flip for class 0, it is still class 0. But when you apply the same transformation to class 6, it becomes class 9.

Another interesting aspect of data augmentation that has gotten little attention is that these transformations often depend on the classes considered. For object detection problems, using color transformations can improve the generalizability of the model to identify airplanes and cars which are invariant to it, but it will reduce the *accuracy* of classes that are strongly defined by their colors, such as apples and oranges. The motivation for this study arose when we explored the effect of various transforms on the performance of a classifier trained on a well-studied dataset of soybean leaf stresses across nine different classes (Ghosal et al. 2018). Figure 1 shows the performance of a well-trained classifier across the classes when each transformation is applied. As shown in the figure, different augmentations show quite different performances across the classes. It can also be seen that the stresses Bacteria Blight and Bacterial Pustule show significantly lower accuracies than the rest of the classes. Moreover, some augmentations, usually color transformations, make it even harder for a classifier to distinguish between these two classes, as seen in the figure. This exploration motivated two questions: (a) Can class-specific augmentations be designed to improve *mean-per-class accuracy*? and, (b) do these class-specific augmentations provide some domain-specific insight?

We deploy a practical method for this problem using genetic algorithms (GA). In our implementation, each set of augmentations has several choices and probabilities of possible augmentations for each class. We use GA to find the best operations choices for training a neural network that yields the best *mean-per-class accuracy* on the test dataset.

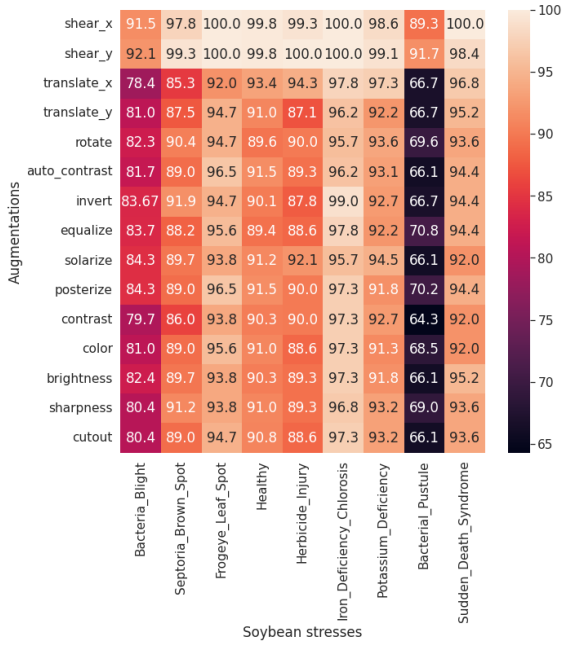

Figure 1: Effect of Augmentations on various classes

Related Work

Data augmentation has played an important role in improving the performance of classifiers, especially in image recognition tasks. Mostly, data augmentation for image classification has been designed manually. On datasets like ImageNet and CIFAR10, transformations like horizontal flip, random crop, rotation, and color transformations have been applied (Krizhevsky, Sutskever, and Hinton 2017; Sato, Nishimura, and Yokoi 2015). On datasets like MNIST, most applied transformations are position and orientation invariances. In addition, augmentations like Cutout (DeVries and Taylor 2017), Mixup (Zhang et al. 2017), and Cutmix (Yun et al. 2019), which either mask out or replace random patches, have also improved the performance of modern image classifiers. However, these methods are manually implemented and require expert knowledge or require significant manual exploration.

Several papers have attempted a more automated approach for finding data augmentation methods from data and have evolved to overcome the cumbersome task of finding augmentations manually. Smart Augmentation (Lemley, Bazrafkan, and Corcoran 2017) introduced a network that learns to generate augmented data by merging two or more samples in the same class. Another approach by (Shrivastava et al. 2017) employed a generative adversarial network (GAN) (Goodfellow et al. 2020) that generates new images that augment datasets.

The rise of AutoML in recent years (Zoph et al. 2018; Real et al. 2019) has shifted the focus to using augmentations to determine the data augmentation policy rather than generating more data with GANs. AutoAugment (Cubuk et al. 2018), uses reinforcement learning to optimize the *accuracy* of an image classifier in a discrete search space of augmentation policies. The augmentation policies in AutoAugment showed state-of-the-art performances on various benchmark datasets with different models. Population based augmentation, PBA (Ho et al. 2019) proposed a new algorithm that uses population based training to generate augmentation policies. However, the computational complexity of training a classifier from scratch for each augmentation policy makes it a less feasible option for ordinary image classification problems. Our method aims to alleviate this problem by only fine-tuning the well-trained classifier to find the best augmentation policies for a target dataset. Additionally, our focus is on improving the *mean-per-class accuracy* (rather than *total accuracy*).

Materials and Methods

We develop a computational workflow for the automated identification of augmentation policies that yields the highest *mean-per-class accuracy* on a target dataset. The workflow uses GA based optimization where an augmentation policy consists of many sub-policies for each class in a target dataset. The augmentations are randomly chosen for each image in each mini-batch based on the populations generated by the GA.

Dataset The dataset consists of $16,573$ RGB images of soybean leaflets across 9 different classes (8 different soybean stresses and healthy soybean leaflets). These classes cover a diverse spectrum of biotic and abiotic foliar stresses. Fig. 2 illustrates the imaging setup and the 9 different soybean stress classes used in our study. The training and test data consisted of $14,915$ (90 %) and 1658 (10 %) images, respectively.

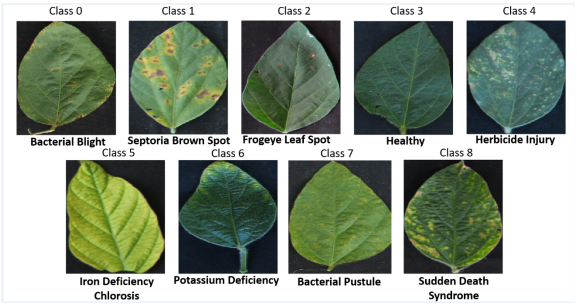

Figure 2: Image examples of healthy leaflet and eight different soybean stresses

Search Space We consider the augmentation policy search as a discrete combinatorial optimization problem. The 15 augmentations used are the same as the ones used in AutoAugment. The augmentations we searched over are ShearX, ShearY, TranslateX, TranslateY, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, and Cutout. The magnitude of augmentations is kept constant as the mean of possible values throughout the training. For each augmentation, we consider discrete probabilities 0 to 1 with an increment of 0.1 to be applied to each class.

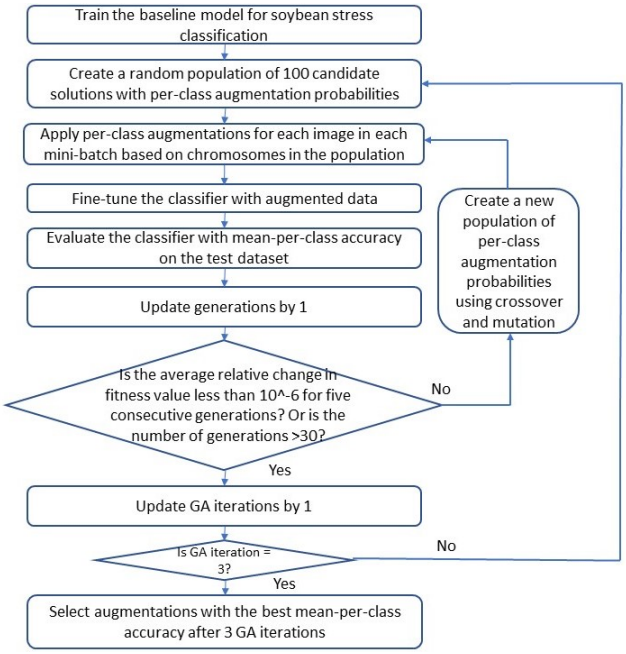

Figure 3: workflow

Search Algorithm: The search algorithm we used in our experiment has two components: An optimizer that uses GA and child networks which are used to fine-tune the classifier. First, the baseline model, a convolution neural network (CNN) ResNet50, is trained without using any augmentations. This model is then used as child networks and fine-tuned for each population in the GA over many generations. The GA starts with a population of policies that consists of the probabilities of augmentations that must be applied for each class. These per-class augmentation probabilities are randomly chosen for each image in each mini-batch. Then the child networks are finetuned on these augmentations, and the *mean-per-class accuracy* from each network is given as the fitness function for the GA. Figure. **??** shows the flowchart of the workflow.

Optimization using GA: GAs are population based heuristic solution-search methods. The population of GA is encoded as numeric string called chromosomes. Here, each chromosome (i.e. member of the population) represents the set of probabilities of augmentations to be applied for each class for training the classifier. The trained classifier then returns the *mean-per-class accuracy* of the test dataset. Each chromosome is assigned a fitness value which is the *mean-per-class accuracy* of the fine-tuned classifier. Based on the fitness scores, chromosomes are evolved over generations using selection. Chromosomes also undergo crossover and mutation in each generation. The chromosomes will evolve until the best solution is identified or a termination criterion is met.

The input chromosome comprises of strings, each representing a set of probabilities of augmentations to be applied for each class in the target dataset. We have nine

	Generation - 10	Generation - 20	Generation - 30	Baseline Model
Mean-per-class accuracy	89.1	90.5	93.11	90.68
Overall accuracy	91.3	92.6	94.9	91.20

Figure 4: Accuracies over 30 generations compared to baseline model

classes and fifteen augmentations. Therefore, each chromosome is a 9x16 long string with probabilities ranging from 0 to 1 with an increment of 0.1. The choice of GA hyper-parameters (population size, selection strategy, mutation/crossover probabilities) were chosen by first performing an analogous optimization on a canonical pathological function – the Rastrigin function (Sesha Sarath Pokuri et al. 2018) – with the same dimensionality. Results on the Rastigin function suggested a population size of 100. Steady-state selection, random mutation, and single-point crossover were chosen for selection, mutation, and crossover, respectively.

The training of CNN: The baseline model is trained on a ResNet50 backbone. The CNN is then finetuned for each population in GA with the set of augmentations defined in each chromosome. The*mean-per-class accuracy* from each model is returned as the fitness score for the corresponding chromosome.

Experimental Results and Analysis

We deployed the computational workflow on a GPU cluster available at Iowa State University. The GA was executed on 4 A100 NVIDIA GPUs (80 GB memory). Each GPU was able to simultaneously fine-tune 3 models. One generation took around 8 hours to complete. We note that this time can be significantly reduced by farming out to multiple GPUs.

We compare the *overall accuracy* and *mean-per-class accuracy* of the augmented model with that of the baseline classifier. Figure 4 shows the overall and *mean-per-class accuracy* of the classifier on the test dataset for each generation in GA and for the baseline model. It can be seen that the overall *test accuracy* of the baseline model increased from 91.20% to 94.9%.

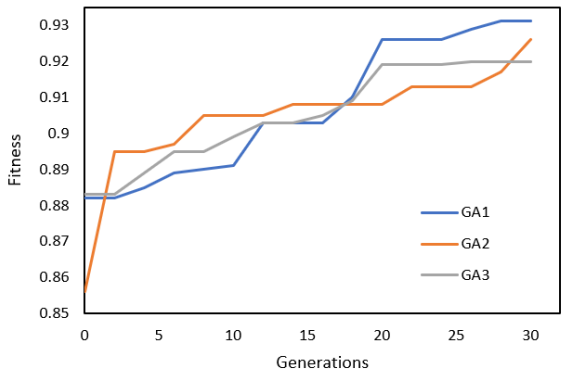

Figure 5: mean-per-class accuracy improvements across GA generations, for three different GA runs

Figure. 5 shows the fitness values of GA across 30 generations over 3 iterations. We expect (in time for the actual conference) the results to continue to improve with additional generations of the GA.

For the model's performance across all the classes, we plot the confusion matrix of the baseline model (Figure 6) and that of the augmented model (Figure 7). As mentioned earlier, the classifier performs poorly in predicting Bacteria Blight (class 0) and Bacterial Pustule (class 7) and misclassifies them interchangeably. When we observe the classifier's performance on these classes, the confusion matrices show that the miss-classification on these classes has significantly reduced, thereby increasing the *per-class accuracies*. As the GA has not reached its termination criteria, we expect the performances in these classes to improve in the next generations.

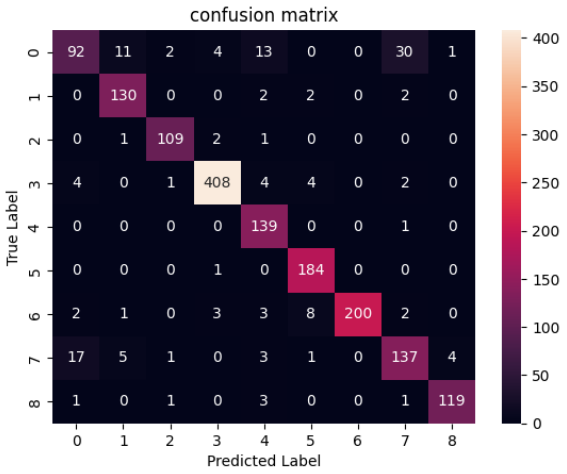

Figure 6: Classification accuracy confusion matrix of the baseline model

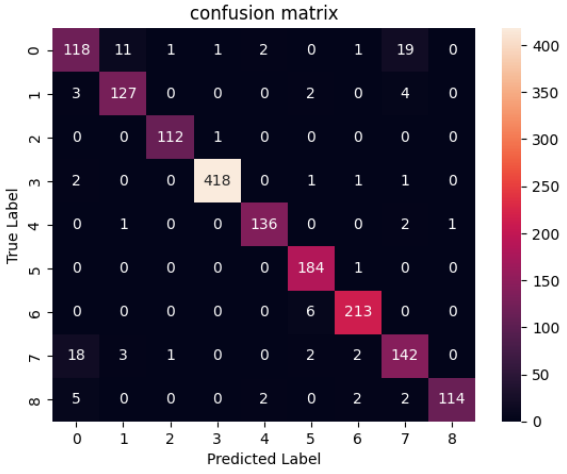

Figure 7: Classification accuracy confusion matrix of the optimized model

Our workflow improves the performance of the classifier, especially among confusing and difficult classes.

Conclusion

The main objective of this study is to explore an automated data augmentation workflow for training deep learning models that are optimized by GA. GA-based optimization protocol using CNN as a classifier was used to find the set of augmentations that delivers the maximum *mean-per-class accuracy* on the test dataset. This goal has been successfully achieved as the algorithm can improve the per-class accuracies of the difficult classes in the dataset.

References

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

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. V. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 702–703.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

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

Ghosal, S.; Blystone, D.; Singh, A. K.; Ganapathysubramanian, B.; Singh, A.; and Sarkar, S. 2018. An explainable deep machine vision framework for plant stress phenotyping. *Proceedings of the National Academy of Sciences* 115(18):4613–4618.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM* 63(11):139–144.

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

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM* 60(6):84–90.

Lemley, J.; Bazrafkan, S.; and Corcoran, P. 2017. Smart augmentation learning an optimal data augmentation strategy. *Ieee Access* 5:5858–5869.

Lim, S.; Kim, I.; Kim, T.; Kim, C.; and Kim, S. 2019. Fast autoaugment. *Advances in Neural Information Processing Systems* 32.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, 4780–4789.

Sato, I.; Nishimura, H.; and Yokoi, K. 2015. Apac: Augmented pattern classification with neural networks. *arXiv preprint arXiv:1505.03229*.

Sesha Sarath Pokuri, B.; Lofquist, A.; Risko, C. M.; and Ganapathysubramanian, B. 2018. Paryopt: A software for parallel asynchronous remote bayesian optimization. *arXiv e-prints* arXiv–1809.

Shorten, C., and Khoshgoftaar, T. M. 2019. A survey on image data augmentation for deep learning. *Journal of big data* 6(1):1–48.

Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; and Webb, R. 2017. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2107–2116.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6023–6032.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.

Zoph, B.; Cubuk, E. D.; Ghiasi, G.; Lin, T.-Y.; Shlens, J.; and Le, Q. V. 2020. Learning data augmentation strategies for object detection. In *European conference on computer vision*, 566–583. Springer.