When and Why Test-Time Augmentation Works

Divya Shanmugam
MIT CSAIL
Cambridge, MA 02139
divyas@mit.edu

Davis Blalock
MIT CSAIL
Cambridge, MA 02139
dblalock@pm.me

Guha Balakrishnan
Amazon Research
Seattle, WA 98109
balakg@mit.edu

John Guttag
MIT CSAIL
Cambridge, MA 02139
guttag@csail.mit.edu

Abstract

Test-time augmentation (TTA)—the aggregation of predictions across transformed versions of a test input—is a common practice in image classification. In this paper, we present theoretical and experimental analyses that shed light on 1) when test time augmentation is likely to be helpful and 2) when to use various test-time augmentation policies. A key finding is that even when TTA produces a net improvement in accuracy, it can change many correct predictions into incorrect predictions. We delve into when and why test-time augmentation changes a prediction from being correct to incorrect and vice versa. Our analysis suggests that the nature and amount of training data, the model architecture, and the augmentation policy all matter. Building on these insights, we present a learning-based method for aggregating test-time augmentations. Experiments across a diverse set of models, datasets, and augmentations show that our method delivers consistent improvements over existing approaches.

1. Introduction

Data augmentation—the expansion of a dataset by adding transformed copies of each example—is a common practice in machine learning. Typically, data augmentation is performed when a model is being trained. However, it can also be used at test-time to obtain greater robustness [18, 22, 5], improved accuracy [12, 24, 20, 10, 13], or estimates of uncertainty [13, 21, 1, 25]. Test-Time Augmentation (TTA) entails pooling predictions from several transformed versions of a given test input to obtain a "smoothed" prediction. For example, one could average the predictions from various cropped versions of a test image, so that the final prediction is robust to any single unfavorable crop.

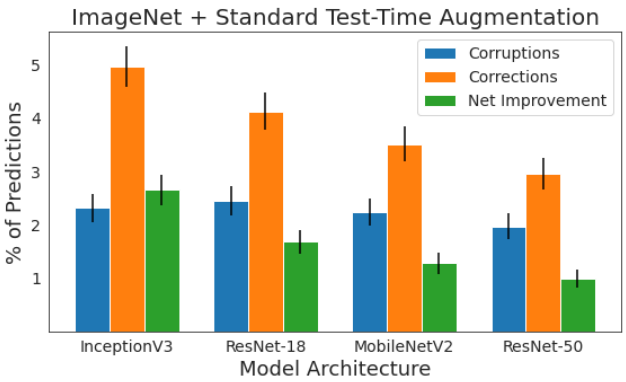


Figure 1: **Percentage of predictions corrected (orange) and corrupted (blue) by TTA**. Reported work on TTA typically examines the net improvement (green). This paper provides analysis on what factors influence TTA improvements and a method that accounts for these factors.

TTA is popular because it is easy to use. It is simple to put into practice with off-the-shelf libraries [17, 4], makes no change to the underlying model, and requires no additional data. However, despite its popularity, there is relatively little research on when and why TTA works. For example, what constitutes a good TTA policy? What kind of images benefit from TTA and what kind do not? How does model architecture affect the magnitude of improvement TTA introduces?

Consider the performance of a TTA policy that includes flips, crops, and scales in Fig. 1. While the net improve-

ment (green) is positive for each network architecture, a sizeable number of predictions are also changed to be incorrect (blue). Moreover, the improvement varies widely over architectures.

The goal of our work is twofold: (1) to understand which predictions TTA changes and why and (2) to use these insights to increase the classification accuracy of TTA methods. To do this, we first provide an empirical analysis of the factors that contribute to TTA performance and show how the dataset, model architecture, and augmentation types all matter. Following this analysis, we present a learning-based method for TTA that depends upon these factors. This method learns a function that aggregates the predictions from each augmentation of a test image.

Our contributions are as follows:

- *Insights into TTA* that reveal the dependence of TTA on characteristics of the training set, architecture, and augmentations involved. We derive these insights from extensive experiments and include practical takeaways for the use of TTA.
- *A new TTA method* that learns to aggregate predictions from different transformations for a given model and dataset. Our method significantly outperforms existing approaches, providing consistent accuracy gains across numerous architectures, datasets, and augmentation policies. We also show that the combination of TTA with smaller models can match the performance of larger models.

2. Related Work

Image augmentation at test-time has been used to measure model uncertainty [13, 21, 1, 25], to attack models [23, 15, 7], to defend models [18, 22, 5], and to increase test accuracy [9, 19, 8, 20, 24, 12]. Because our focus is on test-time augmentation for the purpose of increasing image classification accuracy, we limit our discussion to work considering this problem.

Most works describing a test-time augmentation method for increasing classification accuracy present it as a supplemental detail, with a different methodological contribution being the focus of the paper. In the presentation of Alexnet, Krizhevsky *et al.* [12] make predictions by "extracting five 224 $\times$ 224 patches...as well as their horizontal reflections...and averaging the predictions made by the network's softmax layer on the ten patches." He *et al.* [8] describe a similar setup and include an additional variation involving many more augmentations. The latter variation incorporates rescaling of the input in addition to cropping and flipping. The cropping, scaling, and flipping combination is also employed by Simonyan *et al.* [20] and Szegedy *et al.* [24], though with differing details in each case. While most of these papers report results with and without test-time augmentation, none offers a systematic investigation into the merits of each augmentation function or how their benefits might generalize to other networks or datasets.

The works most closely related to our own are those of Sato *et al.* [19], Howard *et al.* [9], Molchanov *et al.* [14], and Kim *et al.* [11]. The first seeks to improve classification accuracy by employing test-time augmentation. Their method samples augmentation functions randomly for each input, and makes predictions by averaging the log class probabilities derived from each transformed image. In contrast, we optimize both the set of augmentations used and the function that aggregates the predictions from each. Howard *et al.* [9] consider the problem of selecting a set of useful augmentations and proposes a method of choosing augmentations described as a "greedy algorithm" that "starts with the best prediction and at each step adds another prediction until there is no additional improvement." The method is evaluated on a single network and dataset, and does not learn to aggregate predictions as we do. Most recently, Molchanov *et al.* [14] propose Greedy Policy Search, which constructs a test-time augmentation policy by greedily selecting augmentations to include in a fixed-length policy. The predictions generated from the policy are aggregated using a simple average. Similarly, Kim *et al.* [11] present a method to learn an instance-aware test-time augmentation policy. The method selects test-time augmentations with the lowest predicted loss for a given image, where the predicted loss is learned from the training data.

Our work differs in that we focus on the factors that influence test-time augmentation and, given those factors, how we can learn to *aggregate* augmentation predictions. The solution we propose—learning the optimal weights per augmentation—can be applied in conjunction with the aforementioned methods.

3. Understanding TTA Empirically

What affects the performance of TTA in practice? We approach this question empirically, examining the dependence of TTA on the data, architectures, and type of augmentations.

3.1. Setup

Datasets We use two datasets: ImageNet (1000 classes) and Flowers-102 (102 classes). Our preprocessing pipeline is identical for each dataset: the shortest dimension of each image is resized to 256 pixels, followed by a center crop to produce a 256x256 pixel image. We chose these datasets for their differences in difficulty and domain—the architectures we considered can achieve >90% accuracy on Flowers102 and 70-80% on ImageNet. Our choice of datasets is also motivated by the fact that they share the same preprocessing pipeline, allowing us to isolate the benefit of TTA.

Models We evaluate the performance of four architectures: ResNet-18, ResNet-50, MobileNetV2, and InceptionV3. We include MobileNetV2 to examine the performance of TTA for space-constrained applications, where repeated inference might be preferable to deploying a larger network. We downloaded pretrained models from the PyTorch model zoo, where each model is trained on normalized image crops of size 224x224 and with the same augmentation policy, which includes horizontal flips and random crops [3]. To produce pretrained models for Flowers102, we use the finetuning procedure presented by [16]. This procedure starts with a pretrained ImageNet network and freezes the weights in all but the last layer. The network is then trained on the new dataset for 100 epochs, using a batch size of 32, SGD optimizer (learning rate=.01, momentum=.9), and a dropout probability of .2.

Augmentation Policies We consider two augmentation policies. *Standard* reflects the typical augmentations used for TTA (flips, crops, and scales) and *Expanded* includes a more comprehensive set of augmentations, such as intensity transforms. Readers interested in the specific augmentations may refer to the appendix. Each policy replaces the model's original predictions with an average of predictions on transformed images.

The *Standard* test-time augmentation policy produces 30 transformed versions per test image (a cross product of 2 flips, 5 crops, and 3 scales). The 5 crops correspond to the center crop and a crop from each corner. The three scale parameters are 1 (original image), 1.04 (4% zoomed in) and 1.10 (10% zoomed in), based on work that shows multi-scale evaluation improves model performance [20].

The *Expanded* test-time augmentation policy produces 128 transformations for each test image, consisting of 8 binary transforms from the PIL library [17] and 12 continuous transforms. We include 10 evenly-spaced magnitudes of each continuous transformation. We base this set of augmentations on AutoAugment [6] with two major distinctions: 1) We make each augmentation function deterministic, to allow us to understand the specific relationship between an augmentation and model predictions, and 2) we do not consider combinations of these base transformations, because enumerating trillions of perturbations would be infeasible.

Metrics We use two performance metrics: 1) the percentage of predictions *corrected*, where a correction describes an instance where TTA changes an incorrect prediction to a correct one, 2) the percentage of predictions *corrupted*, where TTA changes a correct prediction to an incorrect one.

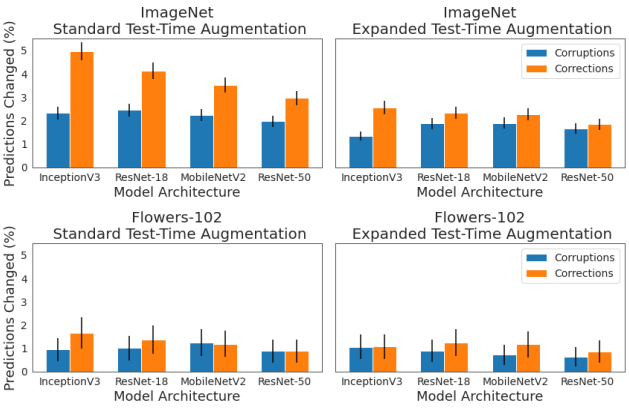


Figure 2: **Percentage of predictions corrected (orange) and corrupted (blue) by two TTA policies (Standard, Expanded).** Results for two datasets (ImageNet, Flowers-102) and four popular neural network models. Models are ordered by accuracy on classification task.

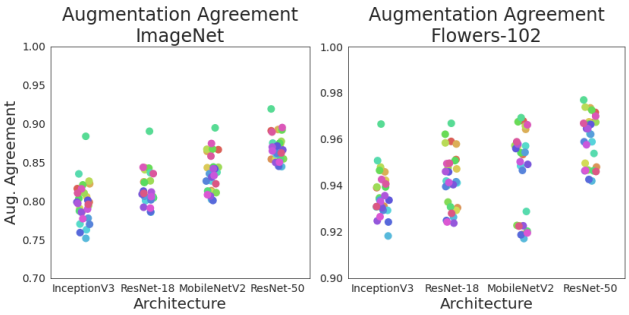


Figure 3: **Architectures that benefit least from standard TTA are also the least sensitive to the augmentations.**. We list the architectures in decreasing order of benefit from TTA. Each dot corresponds to a single augmentation in the standard augmentation policy (e.g. a horizontal flip, scale of 4%, and a center crop). Dots of same color represent the same augmentation.

3.2. Overall results

Figure 2 plots the percentages of corruptions and corrections introduced by the standard and expanded TTA policies. The net effect of TTA is nearly always positive. However, the number of incorrect predictions introduced by the method represents a significant percentage of the changes introduced. In the context of ImageNet and ResNet-18, a little over one third of the labels changed by the standard TTA policy are incorrect.

The magnitude of changes TTA introduces for Flowers-102 is much lower than for ImageNet. This is to be expected since Flowers-102 exhibits a more consistent type of scene. Futhermore, the flowers are typically centered and taken from the same perspective while ImageNet objects are

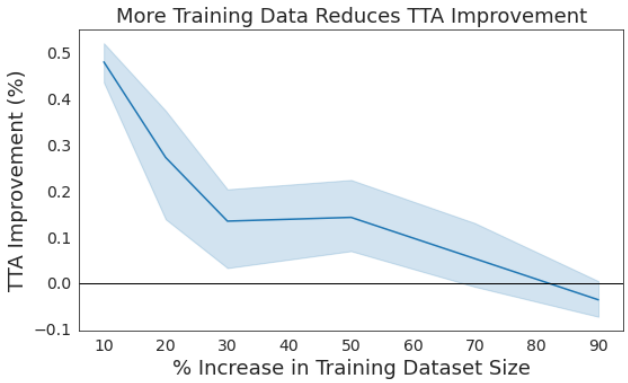


Figure 4: **Increase in amount of training data is correlated with lower TTA improvement.** Results for a ResNet-50 architecture on Flowers-102, where we plot the percentage increase in quantity of training data relative to the original training set on the x-axis. Highlighted area represents standard deviation over five trials.

not.

Figure 2 demonstrates that while one can expect a consistent improvement in accuracy from TTA, the magnitude of this improvement varies. We explore the dependence of this improvement on model accuracy, dataset size, and augmentations in the following sections.

3.3. How does the model matter?

Figure 2 illustrates a downward trend: the more accurate the model, the lower the TTA gain. We hypothesize that this is because more accurate models learn the invariances that TTA typically exploits. We test this hypothesis here.

Setup We measure the *agreement* of an augmentation as the fraction of predictions for which the model's prediction on the original image and the augmented image match. We compare the agreements of the augmentations to the resulting TTA improvement of a model.

Results We order the models on the x-axis of Figure 3 by their improvement from TTA, where InceptionV3 benefits the most and ResNet-50 benefits the least. Models further right on the x-axis are less sensitive to the augmentations of the standard TTA policy.

The distribution of augmentation agreements demonstrates how invariant a given model is to a specific augmentation. For example, Figure 3 shows a single green point has the highest agreement with the original model across models and datasets. This green point corresponds to the horizontal flip augmentation, which means that each model is more invariant to flips compared to the remaining test-time augmentations.

Another observation from Fig. 3 is that MobileNetV2 has two distinctly separated sets of points for Flowers-102. Each augmentation in the lower cluster includes a 10% scale of the image. While this effect is most pronounced for MobileNetV2, it exists for ResNet-18 and ResNet-50 as well. This suggests that the fine-tuned version of MobileNetV2 is less scale invariant than ResNet-18 and InceptionV3, despite achieving a higher accuracy.

Therefore, our main takeaway in this analysis is that the benefit of TTA depends upon the model's *lack* of invariance to the given test-time augmentations.

3.4. How does training dataset size matter?

While model accuracy depends on the architecture chosen, it also depends upon the amount of available training data. Intuitively, a model trained on more data should be more invariant to augmentations, provided the additional data is not redundant. We aim to test the hypothesis that an increase in a model's training data will result in decreased TTA benefit.

Setup We consider Flowers-102 and the *Standard* TTA policy. We split the test set in half to produce the pool of extra training data (3000 images) and a test set (3000 images). We finetune a model on the original Flowers-102 training set (1020 images) and 10% increments of the additional training data, to produce 11 pretrained models. We then evaluate the benefit of TTA for each of these pretrained models to understand the relationship between dataset size and TTA performance.

Figure 4 shows that as the training dataset size increases, the benefit of TTA decreases. In particular, with a 60-70% increase in training data, TTA improvement is nearly 0. This finding is in agreement with our hypothesis and suggests that TTA is best applied with limited training data.

3.5. How does the choice of augmentations matter?

The augmentations included in a TTA policy influence which predictions are corrected and which are corrupted. We show this through a qualitative analysis of corrected and corrupted predictions on ImageNet and Flowers-102. In particular, the use of crops has dataset-specific effects that produce different types of errors.

3.5.1 ImageNet: Changes due to Label Space

The corruptions and corrections introduced by TTA on ImageNet can be classified into three cases: hierarchical labels, multiple classes, and similar labels (Figure 5).

Hierarchical labels include examples like ("plate", "guacamole") and ("table lamp", "lamp shade"). TTA often biases a prediction in favor of the smaller or uncentered component due to the crops included in the policy. Whether

TTA produces a corruption or a correction depends on the assigned label. For example, Figure 5 depicts an image where when the true label is "palace" and TTA predicts "dome."

Other changed predictions correspond to images that contain objects from *multiple classes* such as ("hook","cleaver") and ("piano", "trombone") (Figure 5). Recent work has noted this trait in ImageNet labels [2]. TTA produces incorrect labels by focusing on a different part of the image. Again, TTA predictions favor smaller objects due to crops.

The last subset of major changes corresponds to confusing images, a product of *similar labels* in the dataset (e.g., dog breeds). This subset is largely comprised of animals that are easily mistaken for one another. The inclusion of crops and scales often serve to increase confusion between classes when the resulting image emphasizes a non-distinguishing feature. For example, consider the "Leatherback Turtle" image in Figure 5. One way in which Leatherback Turtles differ from errapins is scale. As a result, the inclusion of scales naturally confuses the two. This type of change suggests that in the absence of labeled data, TTA could produce a useful measure of similarity between images. These similarity estimates can be used to guide contrastive learning and build embedding spaces invariant to specific augmentations [26].

Those designing TTA policies should ensure that the augmentations used have minimal correlation with the label space to avoid errors on images containing hierarchical or multiple labels. When designing TTA policies in the presence of similar labels, consider limiting the magnitude of augmentations included and choose augmentations that further distinguish confusing classes. For example, a zoomed-in version of an "Egyptian Cat" is only easier to mistake for a "Tabby" due to focus on fur (Figure 5. TTAs that benefit well-separated classes are likely different from those that benefit often confused classes.

Figure 5: **TTA changes can be grouped into three types: hierarchical labels, multiple labels, and similar labels.** We include three examples from each type. TTA favors smaller and uncentered labels.

3.5.2 Flowers-102: Changes due to Input Variation

Flowers-102 differs from ImageNet in many respects, such as dataset size, task difficulty, and class imbalance. Most importantly, it does not exhibit hierarchical labels or multiple labels. Here, we show that crops have a similarly intuitive effect on images from Flowers-102. In particular, we show that crops can hurt flowers with smaller distinguishing features (see Figure 6).

Consider images from the class most corrected by TTA ("Rose") and images from the class most corrupted ("Bougainvillea") in Figure 6. The original predictions often mistake a rose for another flower with a similar color ("Globe Flower", "Cyclamen") or shape ("Sword Lily", "Canna Lily"). TTA may correct to roses because crops maintain the petal texture, which differentiates roses from other classes. By including crops and zoomed-in portions

Figure 6: **Roses (top row) are most helped by TTA in Flowers-102, while Bougainvilleas (bottom row) are most harmed.** We show four cases of rose predictions being improved by TTA, and four cases where bougainvillea predictions are harmed. The white stamen of Bougainvilleas is both a distinguishing characteristic and prone to exclusion from certain crops, resulting in corruptions.

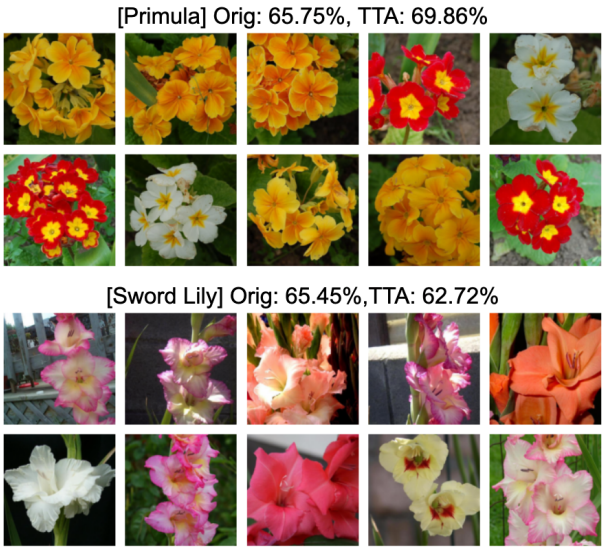


Figure 7: **Equally difficult classes produce different TTA behavior**. The training data for a class that TTA benefits ("Primula", top) look qualitatively different from a class TTA does not benefit ("Sword Lily", bottom).

of the image in the models' prediction, the model is better able to identify these textural differences.

The incorrect predictions introduced by TTA for "Bougainvillea" are likely due to crops missing the cue of the white stamen, a distinguishing characteristic for the class. Moreover, crops may focus on a portion of the background (as with "Mallow") and classify the image incorrectly. These instances shed light on the distinguishing features a model focuses on and how the inclusion of crops favors classes whose features exist in all crops.

In Figure 7, we compare images from two classes on which ResNet-50 performs equally well, "Primula" and "Sword Lily." Interestingly, TTA improves performance on only one, "Primula" and not the other. "Primula" exhibits more consistency in terms of texture, scale, and color, than images of the "Sword Lily," which is likely the reason that TTA helps the former and not the latter. This result suggests that the disparate effects of TTA could be due to differences in input variation between classes. In particular, this could be because horizontal flips and random crops are not sufficient to account for the natural variation in "Sword Lily" images.

4. Method

In the previous sections, we established relationships between TTA improvement and the architectures, dataset, and augmentation types. Our goal is to construct a simple learnable model that can more intelligently aggregate TTAs for a classifier by accounting for these factors. We assume three inputs to our method:

1. A pretrained black-box classifier $f : \mathcal{X} \to \mathbb{R}^C$ that maps images to vectors of class probabilities. We use $\mathcal{X}$ to denote the space of images on which the classifier can operate and C to denote the number of classes. We assume that f is not fully invariant with respect to the augmentations.
2. A set of M *augmentation functions*, $\mathcal{A} = \{a_m\}_{m=1}^M$. Each function $a_m : \mathcal{X} \to \mathcal{X}$ is a deterministic transform designed to preserve class-relevant information while modifying variables presumed to be class independent such as image scale or color balance.
3. A validation set of N images $\mathbf{X} = \{x_i\}_{i=1}^N$ and associated labels $\{y_i\}_{i=1}^N$, $y_i \in \{1, \ldots, C\}$. Images need not relate to the source domain of the model. We assume this set is representative of the test domain.

Given these inputs, our task is to learn an *aggregation function* $g : \mathbb{R}^{C \times M} \to \mathbb{R}^C$. Function g takes in the vectors of predictions for all M augmented versions of a given image and uses them to produce one prediction.

Though g can be arbitrarily complex, such as a multilayer neural network, we aim to avoid adding significant size or latency. Therefore, we only consider functions of the form:

$$g(A(x_i)) \triangleq \sum_{m=1}^{M} (\Theta \odot A(x_i))_{m,*}, \qquad (1)$$

where $\odot$ denotes an element-wise product, $A(x_i) \in \mathbb{R}^{M \times C}$ is the matrix of M augmentation predictions for input x_i,

and $\Theta \in \mathbb{R}^{M \times C}$ is a matrix of trainable parameters. In words, g learns a weight for each augmentation-class pair, and sums the weighted predictions over the augmentations to produce a final prediction. In scenarios where limited labeled training data is available, one may opt for $\Theta \in \mathbb{R}^{M}$, where Θ has one weight for each augmentation:

$$g(A(x_i)) \quad \triangleq \quad \Theta^T A(x_i). \tag{2}$$

We refer to (1) as *Class-Weighted TTA*, or *ClassTTA* and (2) as *Augmentation-Weighted TTA*, or *AugTTA*. We intend for Θ to represent an augmentation's importance to a final prediction and thus impose a constraint that its elements must be nonnegative. We learn Θ by minimizing the cross-entropy loss between the true labels y_i and the output of $g(A(x_i))$ using gradient descent. We choose between *ClassTTA* and *AugTTA* using a small held-out validation set and evaluate the performance of this method, in addition to the individual parameterizations.

5. Experimental Evaluation

We evaluate the performance of our method across the datasets and architectures laid out in Section 3.1.

We implemented our method in PyTorch [17] and employ an SGD optimizer with a learning rate of .01, momentum of .9, and weight decay of 1e-4. We apply projected gradient descent by clipping the weights to zero after each update to ensure the learned parameters are non-negative. In the following experiments, we train *ClassTTA* and *AugTTA* for 30 epochs, choose which to employ using a held-out validation set, and report our results on a held-out test set.

Dataset Splits We divide the released test sets for ImageNet and Flowers-102 into training (40%), validation (10%) and test (50%) sets. We make training and validation sets available to methods that make use of labeled data. We make both the training and validation set available for methods that operate greedily, so that each method makes use of the same amount of data.

Baselines We compare our method to three baselines:

- *Raw*: The original model's predictions, with no TTA.
- *Mean*: Average logits across augmentations. [12].
- *GPS*: Greedy Policy Search [14]. GPS uses a parameter N, for the number of augmentations greedily included in a policy. We set this parameter to 3, in line with experiments reported in the original paper. GPS makes use of all labeled data (both the training and validation set).

Statistical Significance We use a pairwise t-test to measure the statistical significance of our results and produce error bars via 5 random subsamples of the test set.

5.1. Standard TTA Policy

As shown in Figure 8a, our method significantly outperforms all baselines (p-value=7e-6). Moreover, our method significantly outperforms the original model in all 8 comparisons (p-value=4e-8). Our method outperforms other baselines in 33 of the 40 trials summarized by Figure 8.

Our method consistently employs *ClassTTA* on Flowers-102 and *AugTTA* on ImageNet. This is likely due to the large number of classes in ImageNet (1000) and the relatively few examples per class (25) to learn from.

Given enough data, *ClassTTA* should provide a strict improvement over *AugTTA*. Therefore, these results imply that *ClassTTA* is best applied to datasets with few classes and sufficient labeled data. We include results for each parameterization in the supplement. In some cases, our method does worse than either individual parameterization – this is because it makes use of a small hold-out validation set to decide between the two. This suggests that in some cases, it is more useful to forego selecting the parameterization and instead learn a more performant set of weights.

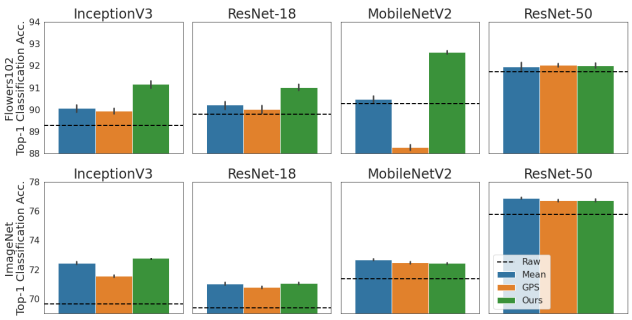
Our experiments also suggest that the combination of TTA with smaller networks can outperform larger networks without TTA and may be of use when deploying machine learning in space-constrained settings. This can be seen in the higher performance of *ClassTTA* applied to MobileNetV2 ($\sim$3.4 million parameters) compared to the original ResNet-50 model ($\sim$23 million parameters) on Flowers-102.

5.2. Expanded TTA Policy

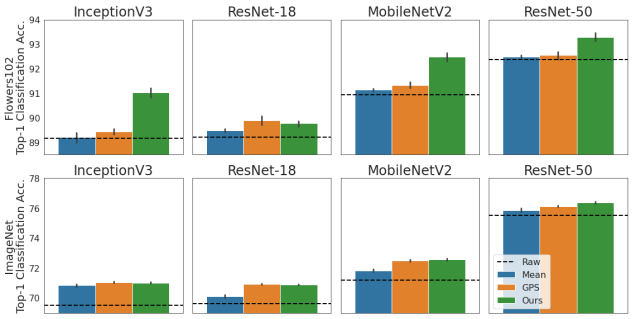
Figure 8b presents our results. Our method significantly outperforms competing baselines (p-value=8e-5). Once more, our method favors *ClassTTA* for Flowers-102 and *AugTTA* for ImageNet. Results in the supplement show that *ClassTTA* yields larger improvement for Flowers-102 and moderate improvements on ImageNet. *ClassTTA* significantly outperforms the original model on all datasets (p-value=1e-6).

Interestingly, many of the TTAs considered in this policy were not included in any model's train-time augmentation policy. As stated earlier, each model was trained with only two train-time augmentations: flips and crops. This suggests that useful test-time augmentations need not be included during training and may reflect dataset-specific invariances.

The tradeoff in using an expanded set of TTAs is the increased cost at inference time. Each additional augmentation increases the batch size that must be passed through the network. This cost may not be justified according to Figure



(a) Standard Augmentation Policy



(b) Expanded Augmentation Policy

Figure 8: Given a standard set of test-time augmentations, our method (green) outperforms competing methods (top). When this set of augmentations is expanded to include 128 distinct transformations, our method continues to outperform competing baselines (bottom). Across every experiment, our method outperforms the original model (dotted black line). A paired t-test shows that our method outperforms competing baselines significantly (p-value=1e-11).

8: the accuracy of *ClassTTA* using a standard set of TTAs is comparable to accuracy of *ClassTTA* using an expanded set of TTAs. This may be because the standard set of TTAs overlaps with the augmentations used during training. Further investigation is necessary to determine the relationship between train-time and test-time augmentation policies.

5.3. Analysis of Learned Weights

The performance of *AugTTA* and *ClassTTA* demonstrate that there are cases where the simple average over augmentation predictions is not optimal. Here, we find that the weights they learn for each augmentation make intuitive sense and confirm our qualitative observations in earlier sections.

Across all architectures on Flowers-102, our method learns to exclude the augmentations that include a 10% scale from the final image (Figure 9). This reflects our earlier observation that augmentations including a 10% scale exhibited the lowest augmentation agreement in Figure 3. Thus, it makes sense that predictions are improved by ignoring test-time augmentations with particularly low agreement.

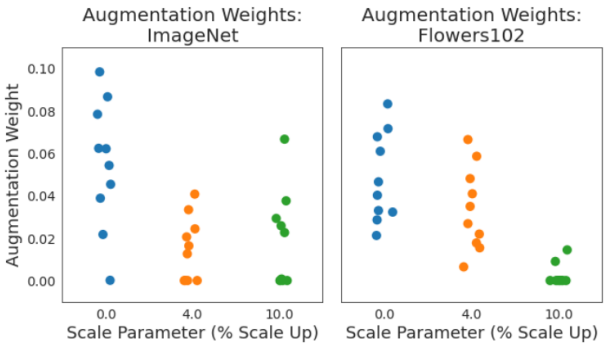


Figure 9: **Augmentations with higher scale parameters are weighted lower by our method.** Learned augmentation weights for each of the 30 augmentations included in the standard policy. Higher scales are weighted lower for both datasets.

Supporting plots for additional architectures and augmentation comparisons and the expanded test-time augmentation policy are included in the supplement. In each case, augmentations with higher scale parameters (corresponding to more zoomed-in images) are weighted lower.

6. Discussion

In this paper, we investigate when test-time augmentation (TTA) works, and when it does not. Through an analysis of two widely-used datasets—ImageNet and Flowers-102—we show that the performance of TTA depends upon the nature of the training data, models, and augmentation policies employed. We build on these insights to construct a simple method that accounts for these factors and show that it outperforms existing TTA approaches.

The insights shared in this study can improve the field's understanding of how TTA changes model decisions. This work opens promising areas for future work:

- *TTA for contrastive learning*: Contrastive learning can be described as "learning by comparing," and TTA can inform how these comparisons can and should be made. For example, crops of an image that contain different classes should not be categorized as similar. TTA could offer an interesting way to define the comparisons a network learns from.

- *Targeted train-time augmentation policies*: TTA exploits a model's lack of invariance to certain transforms. Ideally, the model would instead learn this invariance. The success of TTA signals the need for greater train-time augmentation and can inform a set of class-specific transforms to include during training.

- *Learned augmentations*: Learning the weights for each augmentation is a naive way to build on the insights presented here. One could instead learn a set of augmentations. Past work on TTA considers common augmentations but it would be interesting to consider a broader class of augmentations.

References

[1] Murat Seckin Ayhan and Philipp Berens. Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. 2018. 1, 2

[2] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020. 5

[3] Remi Cadene. Pretrained models for pytorch. https://github.com/Cadene/pretrained-models.pytorch, 4 2017. Accessed: 2019-07-22. 3

[4] Francois Chollet et al. Keras. https://keras.io, 2015. 1

[5] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019. 1, 2

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 3

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[9] Andrew G Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013. 2

[10] Hongsheng Jin, Zongyao Li, Ruofeng Tong, and Lanfen Lin. A deep 3d residual cnn for false-positive reduction in pulmonary nodule detection. *Medical physics*, 45(5):2097–2107, 2018. 1

[11] Ildoo Kim, Younghoon Kim, and Sungwoong Kim. Learning loss for test-time augmentation. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 2, 7

[13] Kazuhisa Matsunaga, Akira Hamada, Akane Minagawa, and Hiroshi Koga. Image classification of melanoma, nevus and seborrheic keratosis by deep neural network ensemble. *arXiv preprint arXiv:1703.03108*, 2017. 1, 2

[14] Dmitry Molchanov, Alexander Lyzhov, Yuliya Molchanova, Arsenii Ashukha, and Dmitry Vetrov. Greedy policy search: A simple baseline for learnable test-time augmentation. *arXiv preprint arXiv:2002.09103*, 2020. 2, 7

[15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 2

[16] Alex Parinov. cnn-finetune. https://pypi.org/project/cnn-finetune/, 7 2019. 3

[17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 1, 3, 7

[18] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8571–8580, 2018. 1, 2

[19] Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. Apac: Augmented pattern classification with neural networks. *arXiv preprint arXiv:1505.03229*, 2015. 2

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2, 3

[21] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018. 1, 2

[22] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017. 1, 2

[23] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019. 2

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1, 2

[25] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019. 1, 2

[26] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. *arXiv preprint arXiv:2008.05659*, 2020. 5