# Towards Principled Test-Time Augmentation

**Anonymous Authors**[1]

## Abstract

Test-time augmentation—the pooling of predictions across transformed versions of a test input to increase accuracy—is a common practice in image classification. Despite being a widely-used technique, there is little work on how to best perform test-time augmentation.

We present a learning-based method for test-time augmentation. The key idea underpinning our method is to jointly learn across many classes which augmentations are most helpful. Our method can be easily applied to pretrained networks, requires no retraining, and significantly outperforms existing approaches. We ground our method in an empirical study of test-time augmentation methods and report our results on ImageNet across a number of popular architectures and augmentations.

## 1. Introduction

Data augmentation—expanding 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 useful to augment the test set in order to obtain estimates of uncertainty (Matsunaga et al., 2017; Smith & Gal, 2018; Ayhan & Berens, 2018; Wang et al., 2019a), greater robustness (Prakash et al., 2018; Song et al., 2017; Cohen et al., 2019), or improved accuracy (Krizhevsky et al., 2012; Szegedy et al., 2015; Simonyan & Zisserman, 2014; Jin et al., 2018; Matsunaga et al., 2017). This process is called Test-Time Augmentation (TTA) and entails pooling predictions from several transformed versions of a given input to obtain an overall prediction. For example, when attempting to classify an image with a neural network, one might average the predictions from various cropped versions of that image. In this work, we focus on TTA for the purpose of increasing classification accuracy on natural images.

Test-time augmentation provides several advantages over other methods of increasing accuracy, such as carefully tuning model hyperparameters, obtaining a larger training set, or ensembling a set of models. The first is that it is extremely simple to put into practice; it can be implemented with off-the-shelf functionality in many libraries (Paszke et al., 2017; Chollet et al., 2015) with no changes to the underlying model and no requirement to retrain or collect additional data. The second is that, unlike the use of a larger architecture, TTA does not increase the size of a model. This can be crucial on resource-constrained devices (Howard et al., 2017; Yang et al., 2017). The main downside of test-time augmentation is that it adds computational burden, since one must run inference on multiple inputs per prediction instead of just one.

The goal of this work is to enable improved test-time augmentation for modern image classification networks. We pursue this goal through two means: 1) An empirical evaluation of augmentation choices across 4 architectures trained on ILSVRC2012 (ImageNet) and 2) a learning-based method for test-time augmentation. This method consists of jointly identifying both a useful set of test-time augmentations and a function to aggregate the predictions from each. Experiments show that our method significantly outperforms existing approaches, and provides consistent accuracy gains across a variety of modern networks. Moreover, it is nearly free in terms of model size and training time as it adds a model-agnostic number of parameters and takes less than ten minutes to train.

In short, our contributions are as follows:

- Empirical evaluation of test-time augmentation functions commonly used in practice, and actionable recommendations for when to use each.
- A new test-time augmentation method that learns a set of augmentations for a given model and dataset, as well as how to aggregate the associated predictions. Our method significantly outperforms existing approaches and provides consistent accuracy gains across numerous architectures.

Code to reproduce our experiments is available at `https://smarturl.it/PrincipledTTA`.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

## 2. Related Work

There has been a great deal of work on improving augmentation at training time (Geng et al., 2018; Paschali et al., 2019; Hoffer et al., 2019; Hu et al., 2019; Ho et al., 2019b; Xie et al., 2019; Park et al., 2019; Wang et al., 2019b; Cubuk et al., 2019b) as well as explaining training-time augmentation's efficacy from a theoretical perspective (Dao et al., 2019; Chen et al., 2019). Because our focus is only on test-time augmentation for the purpose of increasing image classification accuracy, we limit our discussion to work considering this problem.

Image augmentation at test-time has been used to measure model uncertainty (Matsunaga et al., 2017; Smith & Gal, 2018; Ayhan & Berens, 2018; Wang et al., 2019a), attack models (Su et al., 2019; Moosavi-Dezfooli et al., 2016; Goodfellow et al., 2014), defend models (Prakash et al., 2018; Song et al., 2017; Cohen et al., 2019), and increase test accuracy (Howard, 2013; Sato et al., 2015; He et al., 2016; Simonyan & Zisserman, 2014; Szegedy et al., 2015; Krizhevsky et al., 2012).

Most works describing a test-time augmentation method for increasing classification accuracy present it as a supplemental detail to different methodological contribution being the focus of the paper. For example, Krizhevsky et al. (2012) note they 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. (2016) describe a similar setup, though also include an additional variation involving many more augmentations. The latter variation incorporates rescaling of the input in addition to cropping and flipping. This cropping, scaling, and flipping combination is also employed by Simonyan & Zisserman (2014) and Szegedy et al. (2015), 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.

The two works most closely related to our own are those of Sato et al. (2015) and Howard (2013). The former seek to improve classification accuracy by employing test-time augmentation, just as we do. 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 (2013) considers the same problem and uses an optimized set of augmentations just as we do. This method of choosing augmentations is described only as a "greedy algorithm" that "starts with the best prediction and at each step adds another prediction until there is no additional improvement," without offering further detail. The method is evaluated on a single network and dataset, and does not learn to aggregate predictions as we do.

## 3. Problem Statement

Our goal is to construct a classifier that operates by feeding transformed copies of a given image into a pretrained model and intelligently aggregating the resulting predictions. We assume four inputs:

1. A pretrained black-box model $f : \mathcal{X} \rightarrow \mathbb{R}^C$ that maps images to vectors of class probabilities. We use $\mathcal{X}$ to denote the space of images on which the model can operate and $C$ to denote the number of classes.
2. A set of $M$ *augmentation functions*, $\mathcal{A} = \{a_m\}_{m=1}^M$. Each function $a_m : \mathcal{X} \rightarrow \mathcal{X}$ is a transform designed to preserve class-relevant information while modifying variables presumed to be class independent such as image scale or color balance.
3. A set of $N$ images $\mathbf{X} = \{x_i\}_{i=1}^N$ and associated labels $\{y_i\}_{i=1}^N$, $y_i \in \{1, \dots, C\}$. This could be a subset of the training set.
4. A count $K$, the number of augmentation functions that can be used.

Given these inputs, our task is to learn a pair $(\mathcal{B}, g)$, where:

- $\mathcal{B} \subset \mathcal{A}$ is a set of $K$ augmentation functions, and
- $g : \mathbb{R}^{C \times K} \rightarrow \{1, \dots, C\}$ is an *aggregation function*. $g$ that takes in the vectors of predictions for all $K$ transformed versions of a given image and uses them to produce an overall prediction.

In the next section, we examine how existing approaches to test-time augmentation fit into this formulation, examine their benefits, and conclude with actionable recommendations. We then discuss a learning-based method to learn $\mathcal{B}$ and $g$ from $N$ labeled training images in Section 5.

## 4. Current Test-Time Augmentation

In this section, we examine existing approaches to test-time augmentation, in which model predictions on a set of augmentations are combined using a simple average. In this setting, the $M$ augmentations are predefined by the user, $g$ weights each augmentation equally, and $K$ is equal to $M$. We will refer to this setting of $g$ as *standard test-time augmentation*.

Through an empirical analysis of the benefits and drawbacks of this approach, we deliver practical recommendations about the use of test-time augmentation and intuition for how it transforms model predictions.

## 4.1. Networks

We consider four widely-used models for image classification: MobileNetV2, ResNet-18, ResNet-50, and ResNet-101. We choose these for their ubiquity across experiments on ImageNet. We use the pretrained models from the PyTorch model zoo with an image resolution set to be 224x224. Each of these networks is a modification of the residual network architecture, making the number of trainable parameters the major differentiator between them. Training time augmentation for each model includes horizontal flips and random crops only.

## 4.2. Augmentations

We consider various augmentations that have previously been used at test time. Each augmentation corresponds to a set of augmentation functions. Included in each set is the identity augmentation function, along with an augmentation function for each possible transform. When an augmentation is not discrete (e.g. *brightness* and *crop*), the set contains the identity augmentation function and the application of a randomly sampled augmentation function.

- *hflip* has two augmentation functions: the identity function and a function that applies a horizontal flip.
- *crop* has five augmentation functions: an identity-like function (the center crop) and four functions that produce a 224x224 crop from each corner.
- *brightness* has two augmentation functions: the identity function and a function that applies a brightness transform parameterized by $\alpha$, where $\alpha$ is drawn from a uniform random distribution over [.9,1.1].
- *rotation* has three augmentation functions: the identity function, a function that applies a rotation transform of $\theta$ degrees, where $\theta$ is uniformly sampled over [-3, 0], and a function that applies the same rotation transform sampled over [0, 3].
- *combination* composes the preceding augmentations to produce a set of $2 \cdot 5 \cdot 2 \cdot 3 = 60$ unique augmentation functions.

We limit the magnitudes of the brightness and rotation changes so that the transformed images remain close to the distribution of images the pre-trained models have seen.

There are a number of more complex augmentations that have been shown to improve model accuracy when included during training (Cubuk et al., 2018; Ho et al., 2019a; Cubuk et al., 2019a). We limit our focus to the above augmentations for two reasons: 1) their prevalence in the literature and 2) their simplicity.

## 4.3. Empirical Study

In this experiment, we evaluate the effect of standard test-time augmentations on Top-1 and Top-5 image classification accuracy.

### 4.3.1. Experiment Set-Up

We consider each of the networks and augmentations introduced in Section 4.1 and Section 4.2. We replace each model prediction with an average over the original model prediction and model predictions on the augmented images and measure the resulting Top-1 and Top-5 classification accuracy. Results reported are based on the ILSVRC2012 validation set, containing 50000 images across 1000 classes. We report the changes in model accuracy over 5 runs (sampling 80% of each class for each run) in Figure 1. We evaluate significance using a paired t-test over 20 experiment results, representing 20 model and test-time augmentation combinations.

### 4.3.2. Results

**Certain test-time augmentations are more helpful than others.** In Figure 1 we separate results into 'beneficial' augmentations (*hflip* and *crop*) and 'harmful' augmentations (*brightness* and *rotation*), where 'beneficial' test-time augmentations consistently improve model accuracy and 'harmful' test-time augmentations consistently decrease accuracy. The improvement of the 'beneficial' standard test-time augmentations over original prediction is significant with a p-value of $3 \times 10^{-5}$.

Despite the conservative ranges employed, *brightness* and *rotation* test-time augmentations hurt model performance across all architectures. Previous work has shown the instability of model outputs to certain perturbations (Engstrom et al., 2017), and our study corroborates these results. The significant drop in accuracy caused by *brightness* suggests that these models use brightness as an important classification cue.

Relying on standard test-time augmentation places the responsibility of understanding which augmentations are appropriate at test-time on the machine learning practitioner. This is a data-dependent decision and we see that in the context of *brightness* and *rotation*, model performance suffers from the wrong choice.

Notice that the 'beneficial' test-time augmentations were also employed during training of each model. This suggests that successful test-time augmentations are related to the augmentations used during training. Or, this could also be because of the relative difficulty of learning certain invariances. We leave the study of this relationship to future work.

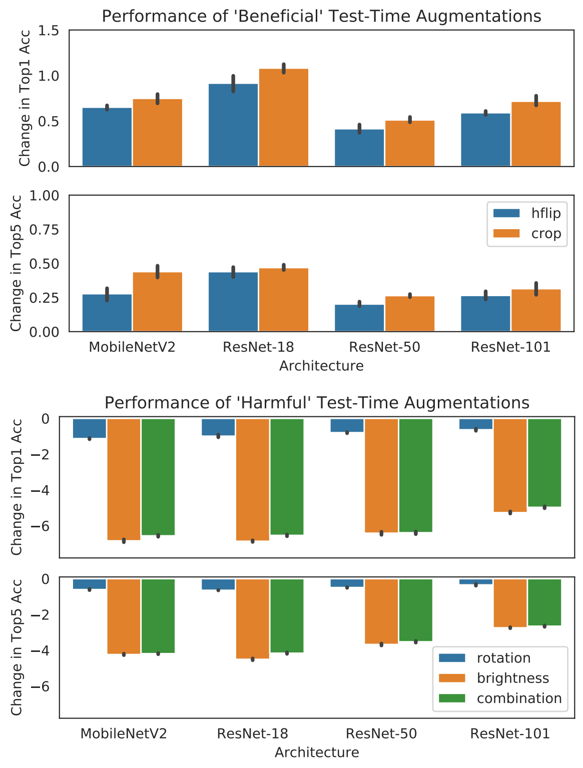


*Figure 1.* Performance of test-time augmentations across four architectures on Imagenet. The upper figure shows performance gain of the beneficial augmentations *hflip* and *crop* in terms of Top-1 and Top-5 accuracy. The bottom plot shows the decrease in the same metrics for *rotation*, *brightness*, and *combination*.

**Test-time augmentation exhibits model-specific trends.** Smaller models exhibit more variable responses to test-time augmentation procedures compared to their larger counterparts. MobileNetV2 and ResNet-18 stand to gain up to 1% in Top-1 accuracy from test-time augmentation, but also stand to lose up to 4% in Top-1 accuracy from poorly chosen test-time augmentations. We show later that our proposed method does not experience this downside.

With larger models, the magnitude of harm incurred by 'harmful' augmentations decreases. This may be because a larger model size has a higher invariance to the augmentation in question. In the following section, we present a method to identify useful test-time augmentations from a set of augmentations given a pre-trained model and a small training set.

## 5. Learned Test-Time Augmentation

We address the limitations of standard test-time augmentation by learning the set of augmentations to use, $\mathcal{B}$, and the aggregation function, $g$, in two steps. First, we learn $\mathcal{B}$ by greedily maximizing a lower bound on the training log likelihood. Given $B$, we then learn a parameter vector $\theta$ to weight the predictions from each augmentation. We elaborate on each step below.

### 5.1. Learning to Rank

We formulate the problem of selecting augmentations as that of finding a vector $\vec{w}$ such that $\frac{1}{N}\sum_{i=1}^{N} l(\vec{w}^\top \vec{v}_i, y_i)$ is minimized subject to the constraint $\|\vec{w}_0\| \leq K$. Given $\vec{w}$, the set $\mathcal{B}$ is defined as the indices of the $K$ nonzeros. We define $A_i$ to be the matrix of model predictions over the set of augmentations for image $x_i$. Thus, $A_i^m$ corresponds to the model output on test image $x_i$ transformed by augmentation function $m$.

We choose an $l_0$-constrained regression formulation for several reasons. First, it requires no knowledge or modification of the model—only precomputed outputs $A_i$. Second, this is a well-studied problem with simple and theoretically-grounded solutions. Third, the $l_0$ constraint makes it easy to choose an exact number of nonzeros; this is in contrast to an $l_1$-constrained minimization, which requires sweeping a regularization parameter to obtain a desired level of sparsity.

We consider two strategies for learning the $K$-sparse vector $\vec{w}$. The first strategy is to simply fit $w$ to minimize $\frac{1}{N}\sum_{i=1}^{N} l(\vec{w}^\top \vec{v}_i, y_i)$ with no sparsity constraint, and then zero out all but the $K$ indices with the largest magnitudes. We refer to this approach as *Logistic Regression with Hard Thresholding* (LRHT). This approach lacks guarantees, but the practice of retaining only weights with largest magnitudes is common in the neural network pruning literature (Hagiwara, 1994; Han et al., 2015b;a; Frankle & Carbin, 2018). It also has the advantage of training extremely quickly, since it only entails solving a convex minimization problem.

The second strategy we consider is Orthogonal Matching Pursuit (OMP) (Pati et al., 1993). This algorithm iteratively selects the augmentation $m$ such that column vector $\mathbb{E}_i[A_i^m]$ has highest cosine similarity with the current gradient of the loss. After selecting each augmentation, it refits the nonzero indices of $\vec{w}$ to minimize the loss. This approach has strong sparse-recovery and approximation guarantees, and is simple to implement. It does require fitting $K$ separate logistic regression models, but these convex optimization problems are still orders of magnitude faster to solve than training the original network. We can fit each model in minutes on ImageNet with a single CPU. Note that this is done once per model, not once per inference.

### 5.2. Learning to Aggregate

After learning the set of augmentations , we learn the aggregation function $g : R^{C \times K} \rightarrow R^C$. Many families of functions are possible; in principle, $g$ could even be a neural network. In order to avoid adding significant size or latency to the model, however, we only consider functions of the form

$$g(A_i) \triangleq \theta \cdot A_i \qquad (1)$$

where $\odot$ denotes element-wise product and $\Theta \in \mathbb{R}^K$ is a matrix of trainable parameters. In words, $g$ learns a weight for each augmentation, and sums the weighted predictions to produce a final prediction. In scenarios where abundant labeled training data is available, one may opt for $\Theta \in \mathbb{R}^{C \times K}$, where $\theta$ is solely tasked with learning which augmentations are helpful on a class-specific basis. We refer to the $\mathbb{R}^K$ case as "Partial-LR" and the second, which has a parameter for each class/augmentation pair, as "Full-LR". We learn the parameters $\Theta$ by minimizing the cross-entropy loss between the true labels $y_i$ and the output of $g(B)$ using gradient descent.

## 6. Experiments

There are two parts to our method: ranking and aggregation. We evaluate each in isolation below. Results show that ranking test-time augmentations with LRHT and aggregating the resulting predictions using Partial-LR is the most successful approach. Unless otherwise stated, we use "our method" as shorthand for this combination.

### 6.1. Ranking

#### 6.1.1. Experimental Set-Up

We focus on the same architectures used in Section 4. We aim to generate a sorted list of the top ten test-time augmentations from a set of sixty test-time augmentations. We generate the sixty test-time augmentations from all combinations of four transforms: *hflip* (2 transformations), *crop* (5 transformations), *brightness* (2 transformations) and *rotation* (3 transformations).

The performance of each ranking is dependent upon the aggregation function used, so we fix the aggregation function to be Partial-LR for each run. We choose Partial-LR over Full-LR since its performance is more stable (see the following section). We train our aggregation layer on each set of $k$ augmentations identified by the ranking algorithm.

We evaluate three approaches to ranking:

- Logistic Regression (LRHT): We train our aggregation model over all augmentations and order the augmentations by their learned coefficients.
- Orthogonal Matching Pursuit (OMP): We use orthogonal matching pursuit to iteratively identify $k$ components and then train our aggregation layer on this fixed set of augmentations. We use this ordering to train an aggregation function for each value of k.
- Baseline (APAC): For each test image, APAC randomly selects $k$ test-time augmentations from a set of potential augmentations.

#### 6.1.2. Results

Among all of the ranking methods, LRHT exceeds the original model accuracy for up to 10 ranked augmentations out of a set of 60. A closer examination of the relationship between LRHT and OMP can be found in the bottom plot of Figure 2 — we see that LRHT never reduces Top-1 accuracy, whereas OMP occasionally does.

Ranking methods that produce a good ordering would demonstrate a monotonically increasing curve in which additional augmentations produce diminishing returns. However, Figure 2 shows that LRHT performance is the only one that is monotonic. OMP is not monotonic on ResNet-18 and APAC is not monotonic on MobileNetV2, ResNet-18 and ResNet-101. Moreover, LRHT outperforms original model accuracy (plotted in green) for all $k \in [2, 10]$. For both MobileNetV2 and ResNet-50, OMP selects an augmentation that does not exceed the model's original accuracy. The ordering of ranking methods in terms of Top-1 accuracy is consistent across all $k$ and all architectures, where LRHT outperforms OMP, and OMP outperforms APAC.

We also see that the application of Partial-LR does not improve performance on its own; the selection of augmentations plays a crucial role in exceeding original model accuracy.

### 6.2. Aggregation

#### 6.2.1. Experimental Set-Up

We provide each model 50 images from each ImageNet class, amounting to $\sim$5% of the training dataset. We trained each model using SGD with momentum on an NVIDIA Titan Xp GPU to convergence or 10 epochs, whichever comes first. Training parameters are the same across runs and include a learning rate of .01, momentum of .9 and weight decay of $10^{-4}$. We perform 5 runs of each experiment, sampling 40 images from each class per run.

We consider four approaches to aggregation:

- Partial Logistic Regression (Partial-LR): Partial-LR has $K$ parameters, one for each augmentation.
- Full Logistic Regression (Full-LR): Full-LR has $KC$ parameters, one for each augmentation-class pair. This enables learning which augmentations are beneficial for each class.
- Maximum Predicted Probability (Max): We choose the augmented prediction with the highest predicted probability in any class. Predicted probability offers a noisy proxy for confidence as shown by Guo et al. (2017). This baseline equates to taking the 'most confident'

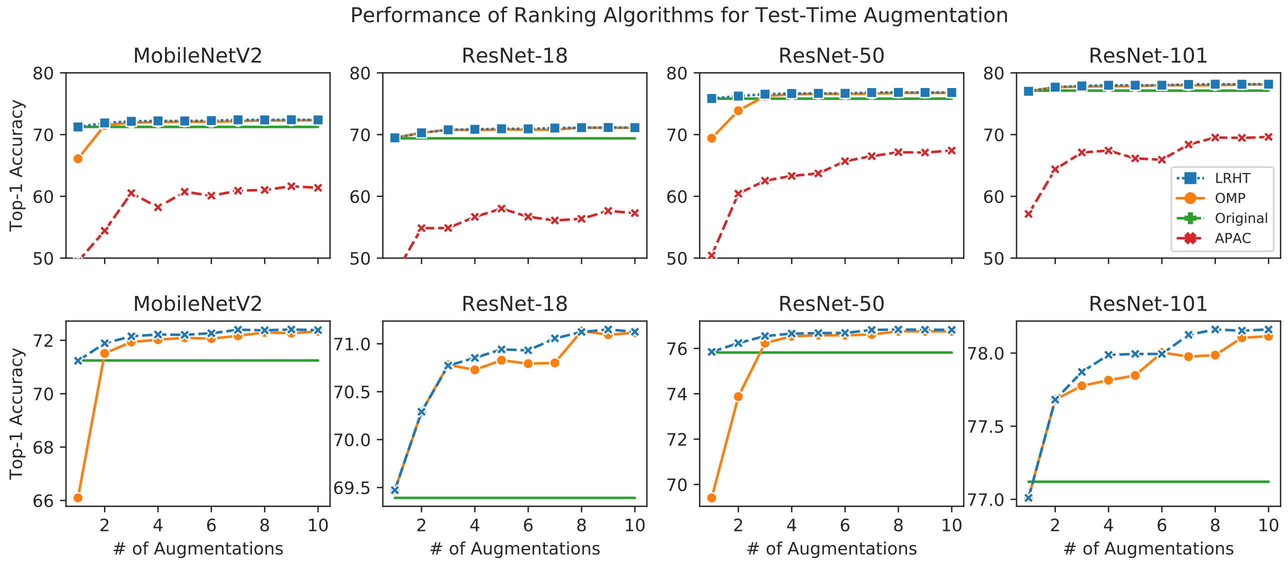


*Figure 2.* The per-augmentation gain of a ranked list of test-time augmentations, as ranked by our method (LRHT) and others. The bottom row omits APAC to enable closer inspection of LRHT, OMP and original model performance. We see that LRHT exhibits 1) monotonic behavior, signifying a useful ranking of test-time augmentations and 2) consistently the largest Top-1 accuracy gains. The bottom figure highlights the comparison between LRHT, OMP, and the original model accuracy.

prediction out of a set of predictions.

- Average (Mean): We also consider taking the simple average over the output logits for each of the augmented images, instead of learning another aggregation function. This is standard practice in test-time augmentation.

### 6.2.2. Results

As shown in Figure 3, both Partial-LR and Full-LR can offer significant improvements over existing approaches. The success of Full-LR depends on the amount of training data available because of the number of parameters the model must learn. In the absence of a large amount of labelled data, we recommend the use of Partial-LR.

Partial-LR significantly outperforms existing approaches under a paired t-test across architectures ($p = .0008$). Improvements from Partial-LR are more consistent across architectures and augmentations when compared to Full-LR. Moreover, Partial-LR protects model performance from “harmful” augmentations by mitigating the effect of poorly chosen test-time augmentations. In the case of *combination*, which includes both beneficial and harmful augmentations, Partial-LR turns a 6% decrease in Top-1 accuracy into a 1.2% increase in Top-1 accuracy for MobileNetV2.

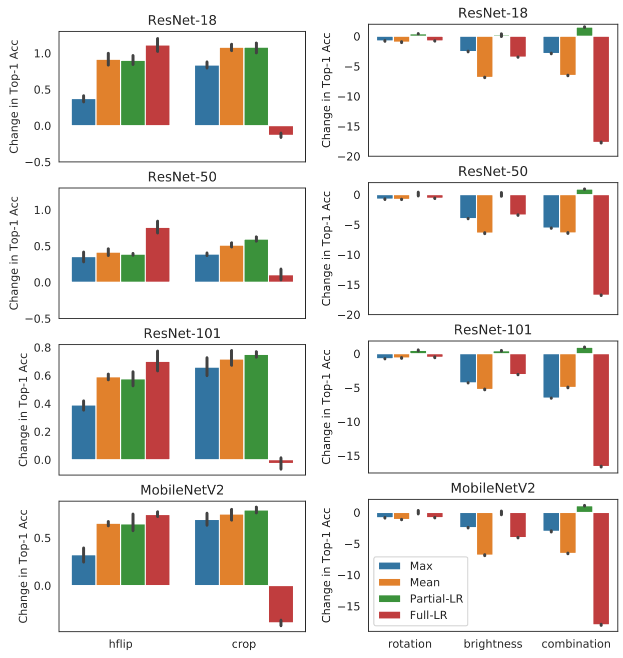
Full-LR fails to perform well when given this amount of labelled data. As described in Section 5.2, the number of parameters in Full-LR is a factor of both $K$ (the number of augmentations) and $C$ (the number of classes). This means that Full-LR for *hflip* learns 2000 parameters (2 augmentations, 1000 classes) and Full-LR using *crop* learns 5000 parameters (5 augmentations, 1000 classes). Thus, learning Full-LR for *hflip* requires less data than Full-LR for *crop*. The results in Figure 3 show that augmentation schemes that produce fewer augmentations benefit more from Full-LR than larger augmentation schemes when given the same amount of training data. For example, where Partial-LR offers no improvement over standard test-time augmentation using *hflip*, Full-LR adds up to $0.6\%$ to a model’s Top-1 accuracy.

## 7. Additional Experiments

In this section, we describe our method’s application to anytime classification, projected computational costs, and effect on class accuracies.

### 7.1. Anytime Classification

The result in Section 6.1 may be applied to the task of *anytime classification*. In this setting, a machine learning practitioner is afforded a fixed budget of time. Extra time affords extra improvements to model accuracy. Test-time augmentation falls neatly into this setting in that a higher budget for inference time allows for aggregation over a larger number of augmentations. Figure 2 demonstrates that a larger number of augmentations translates to higher model accuracy, at up to a $1\%$ improvement for popular pre-trained



(a) “Beneficial” augmentations (b) “Harmful” augmentations

*Figure 3.* Performance of four aggregation functions—Mean, Partial-LR, Full-LR, and Max—across augmentations and architectures. We see that Partial LR offers the most consistent improvements, while Full LR achieves the best accuracy when given enough training data.

models. Settings appropriate for machine learning vary widely in terms of computational resources, and test-time augmentation offers a flexible solution.

### 7.2. Speed

Since we intend this tool to be a simple addition to image classification networks, we offer a summary of the time it takes—in both training and inference—and the corresponding accuracy gain. Importantly, this computational cost is independent of model size and confers improvements in accuracy across the board. We report the training time for Partial-LR and each of the four considered architectures in Table 1. We calculate the change in inference time by averaging the inference time on 100 images, ignoring inference time on the first 50 images as a warmup period.

### 7.3. Class-Specific Changes In Accuracy

While standard test-time augmentation improves many model predictions, it also changes some predictions to be incorrect. In this experiment, we assess which classes benefit most from our proposed method for test-time augmentation.

*Table 1.* Test-time augmentation training cost, inference cost, and Top-1 accuracy gain.

| | | | |
|---|---|---|---|
| | MOBILENETV2 | | |
| TTA | TRAINING (S) | △ INFERENCE (S) | △ ACCURACY |
| *hflip* | 258.0 | $0.0021 \pm 0.0065$ | $0.65 \pm 0.10$ |
| *crop* | 540.1 | $0.0012 \pm 0.0042$ | $0.79 \pm 0.04$ |
| *comb.* | 396.6 | $0.2648 \pm 0.0318$ | $1.08 \pm 0.06$ |
| | RESNET-18 | | |
| TTA | TRAINING (S) | △ INFERENCE (S) | △ ACCURACY |
| *hflip* | 136.5 | $0.0004 \pm 0.0038$ | $0.91 \pm 0.06$ |
| *crop* | 314.5 | $0.1054 \pm 0.0084$ | $1.08 \pm 0.08$ |
| *comb.* | 511.9 | $0.1054 \pm 0.0084$ | $1.52 \pm 0.06$ |
| | RESNET-50 | | |
| TTA | TRAINING (S) | △ INFERENCE (S) | △ ACCURACY |
| *hflip* | 163.6 | $0.0003 \pm 0.0020$ | $0.39 \pm 0.01$ |
| *crop* | 425.5 | $0.0020 \pm 0.0072$ | $0.59 \pm 0.03$ |
| *comb.* | 511.7 | $0.1532 \pm 0.0079$ | $0.92 \pm 0.03$ |
| | RESNET-101 | | |
| TTA | TRAINING (S) | △ INFERENCE (S) | △ ACCURACY |
| *hflip* | 197.8 | $0.0007 \pm 0.0099$ | $0.58 \pm 0.06$ |
| *crop* | 440.3 | $0.0011 \pm 0.0089$ | $0.75 \pm 0.02$ |
| *comb.* | 542.2 | $0.5162 \pm 0.0407$ | $0.94 \pm 0.05$ |

#### 7.3.1. EXPERIMENT SET-UP

We produce Figure 4 by identifying classes for which the application of standard test-time augmentation improves that class prediction accuracy by more than 5%. Since the ImageNet validation set contains 50 images per class, this is equivalent to looking at classes in which test-time augmentation corrects 3 predictions. This leaves us with approximately 200 improved classes, out of the original 1000. We bin these classes by their original Top-1 Accuracy, producing ten groups of classes for different brackets of model performance. Within each group, we plot the number of classes whose accuracy improves with the application of our test-time augmentation in blue and standard test-time augmentation in orange. It is important to note that these graphs do not display the *magnitude* of change, but only the sign. Focusing on the sign prevents the measurement from being dominated by large accuracy changes in a few classes.

#### 7.3.2. RESULTS

Two trends are of note in Figure 4. Using the combination of LRHT with Partial-LR, we increase the frequency of TTA improvement across every bin of class accuracy. In particular, we see dramatic improvements in the frequency of TTA success for classes in the range of 40-70% Top-1 prediction accuracy.

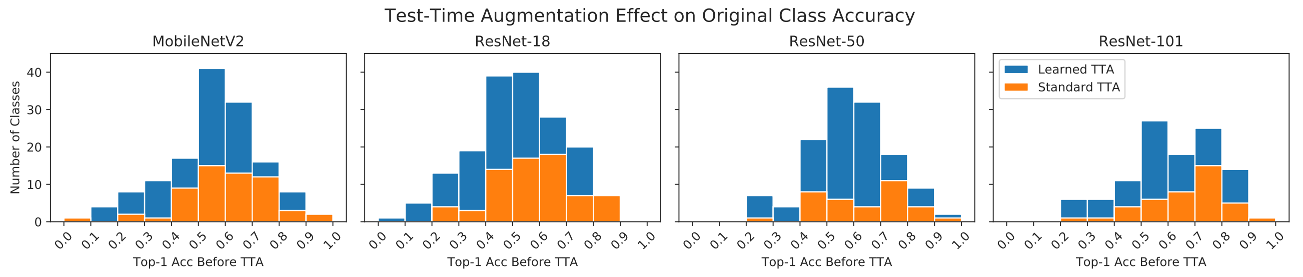


*Figure 4.* Class-specific improvements in accuracy from the application of our method vs. standard test-time augmentation. The blue histogram shows the number of classes improved by our method in a particular range of original Top-1 accuracies. The orange histogram plots the same for standard test-time augmetnation. Three trends are noteworthy: 1) Our method improves performance for more classes than standard test-time augmentation; 2) across architectures, our method improves TTA performance on classes whose original prediction accuracy falls between .4 and .7; and 3) larger models see fewer classes improved compared to their smaller counterparts.

Moreover, our method for test-time augmentation improves more classes than standard test-time augmentation. Standard test-time augmentation improves Top-1 classification accuracy for 51 $\pm$ 17 classes, while our method improves Top-1 accuracy for 137 $\pm$ 23 classes.

Our method also decreases the accuracy in fewer classes than standard test-time augmentation. We include the histogram relating this relationship in the supplemental material. In conjunction, these results show that our method decreases the accuracy for fewer classes than standard test-time augmentation and increases the accuracy of more classes.

## 8. Conclusion

The goal of this work is to enable better test-time augmentation for image classification networks. To do this, we introduce a method of intelligently selecting augmentations and aggregating the predictions derived from them. Experiments on ILSVRC2012 using several networks show that our method consistently outperforms existing approaches. While the performance gain is small in absolute terms, it is large compared to the current gains from test-time augmentation. Our method offers a statistically significant and practically free improvement to existing pre-trained models.

## References

Ayhan, M. S. and Berens, P. Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. 2018.

Chen, S., Dobriban, E., and Lee, J. H. Invariance reduces Variance: Understanding Data Augmentation in Deep Learning and Beyond. *arXiv:1907.10905 [cs, math, stat]*, July 2019. URL http://arxiv.org/abs/1907.10905. arXiv: 1907.10905.

Chollet, F. et al. Keras. https://keras.io, 2015.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

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

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019a.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. RandAugment: Practical automated data augmentation with a reduced search space. *arXiv:1909.13719 [cs]*, November 2019b. URL http://arxiv.org/abs/1909.13719. arXiv: 1909.13719.

Dao, T., Gu, A., Ratner, A. J., Smith, V., De Sa, C., and Ré, C. A Kernel Theory of Modern Data Augmentation. *arXiv:1803.06084 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1803.06084. arXiv: 1803.06084.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. Exploring the landscape of spatial robustness. *arXiv preprint arXiv:1712.02779*, 2017.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Geng, M., Xu, K., Ding, B., Wang, H., and Zhang, L. Learning data augmentation policies using augmented random search. *arXiv:1811.04768 [cs]*, November 2018. URL http://arxiv.org/abs/1811.04768. arXiv: 1811.04768.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1321–1330. JMLR. org, 2017.

Hagiwara, M. A simple and effective method for removal of hidden units and weights. *Neurocomputing*, 6(2):207–218, 1994.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ho, D., Liang, E., Stoica, I., Abbeel, P., and Chen, X. Population based augmentation: Efficient learning of augmentation policy schedules. *arXiv preprint arXiv:1905.05393*, 2019a.

Ho, D., Liang, E., Stoica, I., Abbeel, P., and Chen, X. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. *arXiv:1905.05393 [cs, stat]*, May 2019b. URL http://arxiv.org/abs/1905.05393. arXiv: 1905.05393.

Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., and Soudry, D. Augment your batch: better training with larger batches. *arXiv:1901.09335 [cs, stat]*, January 2019. URL http://arxiv.org/abs/1901.09335. arXiv: 1901.09335.

Howard, A. G. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Hu, T., Qi, H., Huang, Q., and Lu, Y. See Better Before Looking Closer: Weakly Supervised Data Augmentation Network for Fine-Grained Visual Classification. *arXiv:1901.09891 [cs]*, March 2019. URL http://arxiv.org/abs/1901.09891. arXiv: 1901.09891.

Jin, H., Li, Z., Tong, R., and Lin, L. A deep 3d residual cnn for false-positive reduction in pulmonary nodule detection. *Medical physics*, 45(5):2097–2107, 2018.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Matsunaga, K., Hamada, A., Minagawa, A., and Koga, H. Image classification of melanoma, nevus and seborrheic keratosis by deep neural network ensemble. *arXiv preprint arXiv:1703.03108*, 2017.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *arXiv:1904.08779 [cs, eess, stat]*, July 2019. URL http://arxiv.org/abs/1904.08779. arXiv: 1904.08779.

Paschali, M., Simson, W., Roy, A. G., Naeem, M. F., Göbl, R., Wachinger, C., and Navab, N. Data Augmentation with Manifold Exploring Geometric Transformations for Increased Performance and Robustness. *arXiv:1901.04420 [cs, eess, stat]*, January 2019. URL http://arxiv.org/abs/1901.04420. arXiv: 1901.04420.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pp. 40–44. IEEE, 1993.

Prakash, A., Moran, N., Garber, S., DiLillo, A., and Storer, J. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8571–8580, 2018.

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

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Smith, L. and Gal, Y. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

Su, J., Vargas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., and Vercauteren, T. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019a.

Wang, X., Wang, K., and Lian, S. A Survey on Face Data Augmentation. *arXiv:1904.11685 [cs]*, April 2019b. URL http://arxiv.org/abs/1904.11685. arXiv: 1904.11685.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised Data Augmentation for Consistency Training. *arXiv:1904.12848 [cs, stat]*, September 2019. URL http://arxiv.org/abs/1904.12848. arXiv: 1904.12848.

Yang, T.-J., Chen, Y.-H., and Sze, V. Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6071–6079, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.643. URL http://ieeexplore.ieee.org/document/8100126/.