Transformation Pursuit for Image Classification

Mattis Paulin [1,*] Jérôme Revaud[1,*] Zaid Harchaoui[1,*] Florent Perronnin[2] Cordelia Schmid[1,*]

[1] Inria [2] Computer Vision Group, XRCE, France

Abstract

A simple approach to learning invariances in image classification consists in augmenting the training set with transformed versions of the original images. However, given a large set of possible transformations, selecting a compact subset is challenging. Indeed, all transformations are not equally informative and adding uninformative transformations increases training time with no gain in accuracy. We propose a principled algorithm – Image Transformation Pursuit (ITP) – for the automatic selection of a compact set of transformations. ITP works in a greedy fashion, by selecting at each iteration the one that yields the highest accuracy gain. ITP also allows to efficiently explore complex transformations, that combine basic transformations. We report results on two public benchmarks: the CUB dataset of bird images and the ImageNet 2010 challenge. Using Fisher Vector representations, we achieve an improvement from 28.2% to 45.2% in top-1 accuracy on CUB, and an improvement from 70.1% to 74.9% in top-5 accuracy on ImageNet. We also show significant improvements for deep convnet features: from 47.3% to 55.4% on CUB and from 77.9% to 81.4% on ImageNet.

1. Introduction

The focus of this work is image classification. This is a very challenging problem because objects of the same class may exhibit large variations in appearance. Such intra-class variations fall into two categories. The *intrinsic* variability corresponds to the fact that two instances of the same object class can be visually different, even when viewed under similar conditions (*e.g*. different zebras have different patterns). The *extrinsic* variability refers to those differences in appearance that are not specific to the object class (*e.g*. different viewpoints, lighting conditions, image compression).

To learn invariance, the classification system should be trained with as many variations of the considered objects as possible. We use here the term "invariant" in a loose manner, implying that a learning system is invariant if, for a given object, the predicted label remains unchanged for all possible variations of images of the class. Following the taxonomy of [5], there are three approaches to building invariant learning systems from finite training sets: (i) generating virtual examples by applying transformations to the original samples to account for the variations that are expected at test time [15, 9, 10, 27, 14]; (ii) designing a feature representation which is inherently invariant to the variations to be expected [29]; (iii) embedding the invariance in the structure of the learning system, a popular example being convolutional nets [3, 28]. Examples of (ii) include invariant kernels for support vector machines [26] but are limited to invariance to transformations satisfying a particular Lie group structure. On the other hand, examples of (iii) usually correspond to learning architectures intended to mimic biological visual systems [3] that exhibit attractive invariance properties; it remains unclear however how to reverse-engineer such architectures to build in relevant invariances for a particular task.

We propose here to explore virtual example generation (option (i)), by explicitly generating virtual examples at training and at test time. While it might be difficult to generate virtual samples that reflect intrinsic class variations (except for specific object classes such as pedestrians [20, 22]), it is possible to generate virtual samples that simulate extrinsic variations by applying simple geometric

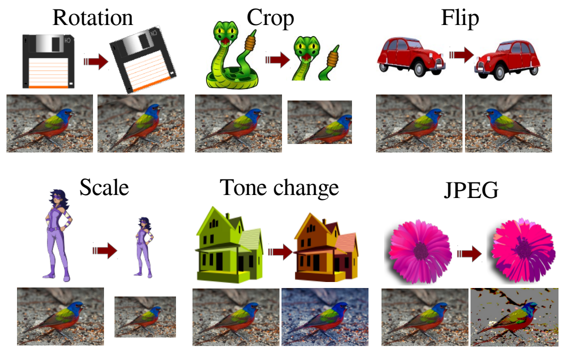

Figure 1: Illustrations of image transformations, on simple icons and on an image from the CUB dataset.

[*]LEAR team, Inria Grenoble Rhône-Alpes, Laboratoire Jean Kuntzmann, CNRS, Univ. Grenoble Alpes, France.

and colorimetric transformations. In this work, we focus on such transformations, as they have been shown to increase classification accuracy, especially on simple tasks such as digit recognition [15, 9, 10, 34], and, more recently, on ImageNet [14].

We believe that the selection of an effective set of transformations is essential for the success of the approach. Indeed, transformations that are too conservative (*e.g*. removing the first column of pixels) have little if any impact, whereas they come with significant computational overhead both at training and testing time. On the other hand, transformations that are too extreme (*e.g*. a vertical flip) will lead to unlikely images and may decrease classification accuracy. We are not aware of any approach to selecting a set of transformations except manually by trial and error. This might be acceptable when the number of possible transformations is limited, for instance when dealing with small (*e.g*. 256 pixels) black-and-white images of digits. However, such a manual selection process is not applicable when there is a large number of possible transformations, a must when dealing with realistic datasets. In this work, for instance, we consider 40 simple transformations as well as their combinations which results in $>$1,000 possible transformations.

We propose a principled approach for selecting a set of transformations, termed *Image Transformation Pursuit* (ITP). ITP computes a set of optimal transformations from a large "dictionary" of transformations (see Fig. 1), by iteratively and greedily selecting one transformation at a time. ITP is reminiscent of pursuit algorithms such as matching pursuit or basis pursuit [19], which compute a signal approximation from a dictionary by iteratively selecting one atomic element at a time from the dictionary. Furthermore, ITP also allows to efficiently explore the set of second order transformations (*e.g*. crop+flip), that correspond to combinations of basic first order transformations.

We report results for ITP on two public benchmarks: the CUB dataset of Birds and the ImageNet 2010 challenge. On both datasets we report significant improvements for Fisher as well as deep convnet features. One important conclusion of our work is that it is crucial to *apply transformations both at training and test time for best performance*.

2. Related Work

We focus our literature review on those works which augment the training (and possibly test) set(s) by adding "virtual examples". This is a well-known approach to enforcing robustness of a learning system to variations of the input. For image classification and related problems, there are two ways to apply this approach, either by applying a transformation directly on the descriptor, or by applying a transformation on the image and then computing a visual descriptor.

Virtual examples. The first strategy, equivalent to noise injection, is itself equivalent in many cases to enforcing a particular regularization functional. In particular, generating virtual examples by adding Gaussian noise directly on the training examples is equivalent to ℓ_2-regularization for quadratic loss functions [4]; see [1, 30, 21, 31, 7, 18] for more general equivalence results including drop-out noise [14, 33]. While the noise injection strategy is simple to implement, it can be difficult to interpret from a computer vision point of view. Indeed, the virtual image corresponding to the noise-corrupted visual descriptor is intractable for several noise distributions, and as a consequence the choice of relevant noise distributions is unclear.

Virtual images. The second strategy works directly at the image level. For instance, [15, 10, 14] generate virtual images from the original ones using plausible transformations such as crop and flip. In contrast to the virtual examples strategy, this strategy is more intuitive and easier to interpret, as one can consider realistic transformations. Yet, generating virtual images and extracting their features is significantly more expensive than adding noise directly on the features. An exception is the digit recognition task, where computing elastic deformation fields can be performed in an elegant and computationally efficient manner [16]. Our ITP approach applies the virtual images strategy by greedily selecting a limited number of transformations from a large dictionary, hence boosting performance without compromising scalability. Furthermore, it requires potentially no prior knowledge, as it can work with arbitrarily large sets of transformations.

Finally, it is worth mentioning that most previous works only perform transformations on the training set [15, 10, 14]. A few approaches also considered transforming the test images [9, 14]. Our approach can benefit from virtual examples *both at training and at testing time*.

3. Learning with transformations

We first describe the families of transformations that we apply to the training and test images (Section 3.1). We then explain the proposed Image Transformation Pursuit (ITP) algorithm (Section 3.2). We finally discuss the score aggregation stage when transformed images are generated at test time (Section 3.3).

3.1. Image transformations

We considered 7 families of transformations (for a total for 40 possible transformations) as well as their combinations.

Flip. This transformation horizontally mirrors an image. It encodes the natural symmetry of most scenes and objects (digits are an exception). Many approaches use flipping to increase their training set size without prior knowledge, see for instance [14, 12].

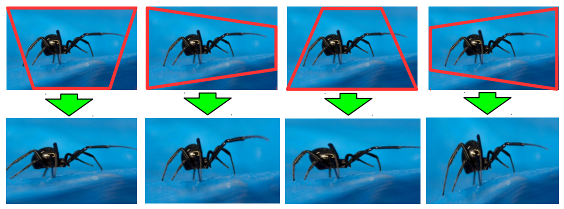
Figure 2: Four of the eight homographies we use.

Crop. We consider the restriction of an image to a specific sub-window. We use 10 different crops, defined relatively to the image size. Prior to training/testing, we randomly draw 10 sub-windows (x_0, y_0, x_1, y_1) with $(x_0, y_0) \in [0, 0.25]^2$ and $(x_1, y_1) \in [0.75, 1]^2$.
Homography. To model viewpoint changes, we consider homographic transformations of the initial images. We restrict ourselves to horizontal and vertical pannings, and select 8 homographies (see Fig. 2).
Scale. We scale down images using bilinear interpolation. We consider 5 downscaling factors, respectively $\left(\sqrt{1.5}\right)^n$ with $n \in \{1, \dots, 5\}$.
Colorimetry. We consider colorimetric transformations. Similarly to [14], we compute the covariance matrix of the RGB components on the whole dataset. Denoting $\lambda_1, \lambda_2, \lambda_3$ and p_1, p_2, p_3 respectively its eigen-values and -vectors, we add to each pixel $p \in [0, 255]^3$ a quantity $\varepsilon_1\lambda_1 p_1 + \varepsilon_2\lambda_2 p_2 + \varepsilon_3\lambda_3 p_3$. We generate three different random triplets $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ sampled from a normal distribution $\mathcal{N}(0; 0.1)$.
JPEG Compression. Despite being designed to minimize the change observable by a human, JPEG compression can have a dramatic effect on local descriptors. To account for variations in image encoding, we consider three different levels of JPEG compression: 30, 50 and 70.
Rotation. To be robust to camera orientation, we rotate images around their centers. We consider 10 rotations of $\{-15, -12, \dots, -3, +3, \dots, +12, +15\}$ degrees.
Order-K Transformations. By order-K transformations, we denote all compositions of K of the previously defined transformations (e.g. for $K = 2$, crop+flip or crop+JPEG).

3.2. Image Transformation Pursuit (ITP)

In this section we assume that transformations are only generated on the set of training images $\mathcal{D} = \{x_1, \dots, x_n\}$. Let $\mathbb{T} = \{\mathcal{T}_1, \dots, \mathcal{T}_{|\mathbb{T}|}\}$ denote the set of $|\mathbb{T}|$ possible transformations. For simplicity, $\mathcal{T}_1$ denotes the identity. Let us associate with each transformation $\mathcal{T}_k$ a binary variable b_k indicating whether $\mathcal{T}_k(\mathcal{D})$ should be added to the original training set $\mathcal{T}_1(\mathcal{D})$ (*i.e.* by default $b_1 = 1$). The augmented training set is:

$$\mathcal{D}^a = \bigcup_{\substack{k=0 \\ \text{s.t. } b_k=1}}^{|\mathbb{T}|} \mathcal{T}_k(\mathcal{D}). \tag{1}$$

Our goal is to find the binary vector $b = [b_1 \dots b_{|\mathbb{T}|}]$ such that training with $\mathcal{D}^a$ yields the best possible accuracy on the test set. Because the number of possible combinations of transformations is exponential in $|\mathbb{T}|$, it is impossible to exhaustively test all combinations. We now propose a tractable alternative.
Algorithm. Our transformation selection algorithm, called ITP, relies on a greedy search strategy in the space $\mathbb{T}$. We start with the identity transformation $\mathcal{T}_1$, *i.e.* with original images. We then maintain a set of current transformations and monitor the gain on validation accuracy obtained by adding a new transformation. For each candidate transformation $\mathcal{T}$, we train a classifier on a subset $\mathcal{D}^a_{\text{train}}$ of $\mathcal{D}^a$, to which we add $\mathcal{T}(\mathcal{D}_{\text{train}})$. Each classifier is tested on the remaining part $\mathcal{D}^a_{\text{val}} \cup \mathcal{T}(\mathcal{D}_{\text{val}})$; we use average prediction at test time on the transformed data, as in section 3.3. The transformation that yields the best validation accuracy is selected. The process is then iterated with the remaining transformations. The algorithm stops after selecting a fixed number of transformations $T \leq |\mathbb{T}|$, or when the gain in validation accuracy obtained by adding a new transformation drops below a threshold; see Alg. 1.

Algorithm 1 Image Transformation Pursuit (ITP)

```
INPUTS: Dataset $\mathcal{D}$. Set $\mathbb{T}$ of base transformations. Threshold $\Delta$ or number $T$ of transformations.
INITIALIZATION: $S = \{\mathcal{T}_1\}$.
While $|S| \leq T$ and perf. gain higher than $\Delta$ do
    • split $\mathcal{D}$ into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$.
    • For $\mathcal{T} \in \mathbb{T}, \mathcal{T} \notin S$ do
        – Let $S^+ = S \cup \{\mathcal{T}\}$.
        – train on $S^+(\mathcal{D}_{\text{train}})$
        – compute perf. $\mathcal{P}_{\mathcal{T}}$ on $S^+(\mathcal{D}_{\text{val}})$.
    • $S = S \cup \{\text{argmax}_{\mathcal{T}} \mathcal{P}_{\mathcal{T}}\}$.
OUTPUT: $S$
```

Selecting Order-K Transformations. By order-K transformations, we mean those transformations that combine K base-level transformations. To select among the increasingly larger amount of order-K transformations, we propose two sub-optimal methods derived from our ITP algorithm. Both rely on the idea that if some transformations are beneficial to classification, their composition is also likely to be.

The first alternative, which we call K**-ITP**, starts by running $(K-1)$-ITP until convergence of the validation accuracy, or until a fixed number of transformations is chosen. Then, it generates all compositions of order K from the returned transformations, and continues appending them greedily in the same fashion as before.

The second alternative, which we call K-ITP-Scratch or

K-**ITP-S**, starts as before by generating the set S of order-$K-1$ transformations by running $(K-1)$-ITP-S (1-ITP-S equals ITP). The set S' of all order-K transformations is obtained by combining elements of S. A ITP procedure is then started from scratch, drawing transformations from $S \cup S'$.

Implementation Details. We consider linear SVM classifiers trained with Stochastic Gradient Descent (SGD) [6, 2]. At each main iteration of ITP, we determine the gain offered by a specific transformation at a low cost using the following strategy. Indeed, at iteration k (i.e. k transformations are already selected), a classifier w_k is learned using SGD on half of the training set augmented with these k transformations. For each (k+1)-th candidate transformation, an SGD is run using w_k as a warm start and few epochs. We determine the initial learning rate η_0 following the heuristic given in [6]. We also cross-validate the λ regularization parameter using values around the best parameter selected at the previous main iteration. In our experiments, however, this optimal value never changed across the main iterations.

Ranking Alternative. We also propose a simple one-step alternative to ITP which we refer to as Transformation Ranking or **TR**. As is the case for ITP, we start with the original samples and quantify the gain that we would get by adding the $|\mathbb{T}|$ possible transformations. We rank the transformations based on this gain and select the top $T \leq |\mathbb{T}|$ transformations. Note that in this setting, the redundancy between the transformations is not taken into account (only the gain with respect to the original images).

3.3. Score aggregation

Virtual samples can also be generated for test images. In this case, we need to compute for each transformed test sample a score and to aggregate these scores. We now discuss possible alternatives for aggregation.

Averaging. We use the scores of all virtual examples as independent measures and average them to get a consensus similarly to [14]. Denoting by $s_t^{(k)}$ the score given by the classifier for class k to the t-th transformation ($t \leq T$), we define: $s_{\text{avg}}^{(k)} := \sum_{t=1}^{T} s_t^{(k)}$.

Maximum. We use the transformation that yields the best score: $s_{\max}^{(k)} := \max_t s_t^{(k)}$. It returns the prediction for which the classifier is the most confident.

Softmax Aggregation. The previous maximum scheme can suffer from the presence of outliers, while the averaging takes into account transformations with low confidence. A soft-max strategy $s_{\text{smax}}^{(k)} := \log \sum_{t=1}^{T} \exp s_t^{(k)}$, provides an intermediate solution.

4. Experiments

We first describe the datasets and experimental set-up. We, then, study the impact of different design choices on the ITP algorithm. We also quantitatively compare the ITP, K-ITP, K-ITP-S and TR algorithms and several baselines. Finally, we compare our method against the state-of-the-art on large-scale and fine-grained image classification datasets.

4.1. Datasets and experimental set-up

We report results on two challenging benchmarks.

CUB. The Caltech-UCSB-Birds-200-2011 dataset [32] is a fine-grained classification benchmark consisting of 200 bird species and approximately 12,000 images. We use the provided training/test split: there are approximately 30 images per class at both training and test time. The standard metric on CUB is top-1 accuracy.

ILSVRC 2010. The 2010 ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2010) dataset consists of 1,000 classes from the ImageNet dataset[1]. It is split into 1.2M images for training, 50K for validation and 150K for test. Following common practice, we report top-5 accuracy. As it is expensive to run the transformation selection on the full training set, we define a subset comprising 30 images per class (30,000 images in total). In the following, we call this subset ILSVRC-30. Except when conducting experiments on the full ILSVRC dataset, we report results on the validation set.

The experimental set-up is as follows.

Image descriptors. We extract SIFT [17] and color [8] descriptors on a dense grid at multiple scales. Their dimensionality is reduced to 61 dimensions using PCA. Following [25], we append the patch location (x, y) and scale σ to the patch descriptors thus resulting in 64-dim descriptors. This strategy can lead to results on par with spatial pyramids at a lower cost [25].

To aggregate the per-patch descriptors, we use Fisher Vectors (FV) with 256 Gaussians, thus leading to 32K-dim representations. We then use PQ compression [13] which was shown to yield an almost-negligible loss of performance in large-scale classification [24]. We use their setting and subdivide the FV into sub-vectors of 8 dimensions and assign 8 bits per sub-vector (1 bit per dimension on average).

SGD training. The SGD learning algorithm has three hyper-parameters [2] which we cross-validate: the regularization λ, the imbalance between positives and negatives β and the number of SGD epochs. At each ITP iteration, we evaluate the gain on validation accuracy after a warm start with 1 epoch on CUB and 2 epochs on ILSVRC10-30. We found these values to give satisfying results during our preliminary experiments.

Fusion of SIFT and color. Finally, confidence scores obtained independently from SIFT FV and Color FV are simply averaged to get the final prediction (denoted as "fusion"

[1]`http://www.image-net.org/challenges/LSVRC/2010/index`

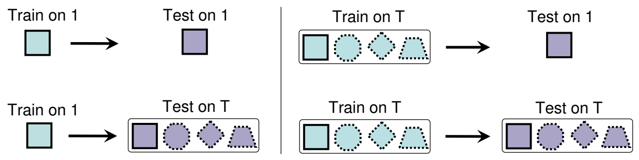

Figure 3: Illustration of the different training and test strategies. Both training and test can be done either on the original single image or on the set of all transformed images.

#transformations		CUB			ILSVRC-30		
#train	#test	SIFT	color	fusion	SIFT	color	fusion
1	1	16.5	23.5	28.2	34.1	28.9	40.4
1	T	17.8	24.9	28.8	36.2	30.6	42.5
T	1	20.0	26.4	31.9	38.7	31.5	44.4
T	T	**27.4**	**35.6**	**42.0**	**42.3**	**37.1**	**48.7**

Table 1: Comparison of different training and test scenarios for $T = 6$, i.e. $T - 1$ corresponds to the number of transformations used for training and test. Score aggregation is performed by averaging.

in the following).

4.2. Impact of score aggregation

We assume that, after the selection pass, we have $T - 1$ transformations per image, in addition to the original one, so that each image has T representatives. We investigate the following scenarios (Fig 3).

- Train 1/ Test 1: training and test are done only on the original images (baseline).
- Train T/ Test 1: training is performed using the original as well as the transformed images, and test is done on original test images.
- Train 1/ Test T: test is done on the transformed images, and their scores are averaged, while training is done on the original images.
- Train T/ Test T: training and test are done on the transformed images.

Table 1 shows that the "Train T/ Test T" scheme outperforms by far the other ones. This demonstrates the *importance of applying transformations at both training and test time*. In what follows, we use this scheme for all experiments unless stated otherwise.

Score aggregation scheme. We compare the different score aggregation schemes (Section 3.3) and provide results in Table 2. We observe that all schemes yield similar results. We therefore decide to use the simple average in the rest of the experiments.

4.3. Transformation selection experiments

We now evaluate ITP. At each iteration, the algorithm evaluates the gain in accuracy obtained for each candidate

Aggregation scheme	CUB (top-1)			ILSVRC-30 (top-5)		
	SIFT	color	fusion	SIFT	color	fusion
average	27.4	35.6	42.0	42.3	37.1	48.7
max	26.2	34.3	41.0	42.1	36.8	48.9
soft-max	27.4	35.1	42.1	42.6	37.3	49.2

Table 2: Comparison of score aggregation schemes.

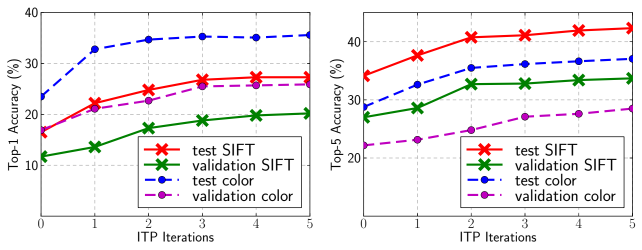

Figure 4: Evolution of the validation and test accuracy with respect to ITP iterations on CUB (left) and ILSVRC-30 (right).

transformation independently on a held-out validation set. We plot the evolution of this validation score across iterations for the selected transformations, along with the accuracy computed on the full test set, in Fig. 4. The validation accuracy, although computed on a smaller set and without waiting for the SGD to converge, is varying consistently with the full test accuracy in our experiments. This validates empirically the core of our algorithm.

Next, we compare ITP to several other selection strategies. We first consider the choice of just adding the 'flip' transformation, a rather common practice [14, 12]. The second choice to which we compare is a random selection of transformations. This is to ensure that we do not get a similar increase in accuracy with any subset of transformations. Finally, we compare with the TR variant of our algorithm, which is a cheaper version of ITP (Section 3.2). Results are presented in Fig. 5 and 6. As can be observed, the selection obtained by ITP significantly outperforms the first two baselines (flip and random). ITP itself performs at least on par with the TR variant, and sometimes significantly better (see SIFT FVs on CUB or color FVs on ILSVRC-30). Overall, both algorithms yield improvements and ITP outperforms TR: for the fusion results, ITP and TR yield a relative improvement of +49% and +41% respectively on CUB, and +21% and +19% on ILSVRC-30.

Fig. 7 (left) shows the first five transformations selected by ITP. We can see that crops are the most frequently selected transformations. We stack in Fig. 7 (right) the selected crops on the CUB dataset, thus leading to a saliency map. They clearly focus on the center of the image. One might argue from this observation that, by applying crops at training and test time, we are just learning a prior on the object location (*i.e.* a saliency map). To test this hypothesis, we perform the following experiment: for each train-

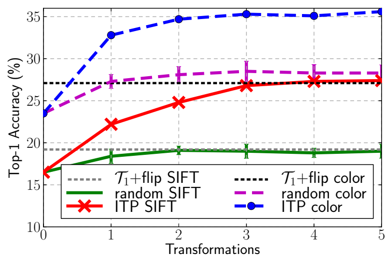

Figure 5: Accuracy on CUB after adding up to five transformations with ITP, with random selection (5 trials averaged), or with just flip in addition to the original images ($\mathcal{T}_1$).

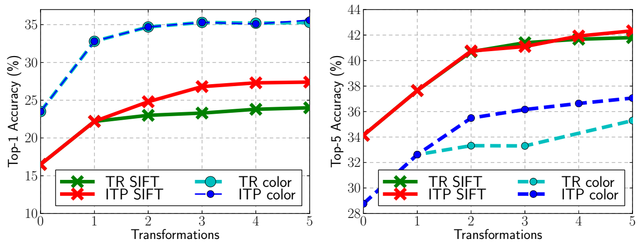

Figure 6: Evolution of the test accuracy on CUB (left) and ILSVRC-30 (right) as a function of the number of transformations selected by ITP, or its cheaper TR variant.

	CUB SIFT	CUB Color	ILSVRC-30 SIFT	ILSVRC-30 Color
1	crop5	crop1	flip	crop2
2	flip	crop5	crop0	color1
3	crop1	crop6	homo2	flip
4	crop6	crop8	crop6	crop1
5	crop0	scale0	crop1	color0

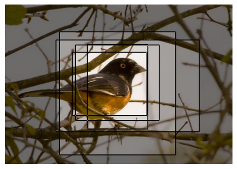

Figure 7: First five selected transformations by ITP (left); overlaying the different crops selected by 2-ITP-S for the SIFT channel (right).

ing and test image, we extract local descriptors from all its transformed versions and aggregate them in a single FV. The patches which are present in multiple croppings are, therefore, weighted more than patches which occur in few or no crops. This is equivalent to weighting patches with the saliency map [25]. On CUB (resp. ILSVRC-30), for SIFT+color we get 35.9% (resp. 44.6%) which is a significant improvement over the 28.2% (resp. 40.4%) baseline – see Table 1. Yet, this is only half of the improvement that we get with ITP (resp. 42.0% and 48.7%). This result confirms that our approach is not merely learning a prior on the object location, but also invariances.

4.4. Transformations of order 2

We refine our selection strategy by incorporating transformations of order 2, i.e., transformations of transformations (e.g. combining flip and crop). Because the number of such combinations is huge (> 1,000), we adopt the strategy described in Section 3.2. For both variants K-ITP and

Method	SIFT	color	fusion
ITP ($T = 5$)	27.4	35.6	42.0
2-ITP ($T = 10$)	32.2	37.7	45.8
2-ITP-S ($T = 5$)	31.8	37.7	45.2

Table 3: Comparison of ITP and 2-ITP on CUB.

K-ITP-S, we select the $T-1 = 5$ transformations during a first pass of order-1 ITP. We, then, generate all possible combinations of order-2 transformations from these. For 2-ITP, we continue the greedy selection for 5 additional iterations (i.e. in total, we will select $T-1 = 10$ transformations) while for 2-ITP-S we select in total 5 transformations, restarting from scratch with a dictionary of transformations combining the 5 previously selected order-1 transformations with their combinations.

Results for 2-ITP and 2-ITP-S are given in Table 3 for CUB and in Table 6 for ILSVRC. Interestingly, we observe an improvement of the performance, compared to using only order-1 transformations. For instance, +3.5% on CUB and +1.1% on ILSVRC-30 for 2-ITP-S. Note that the difference in performance is marginal between the two variants, while 2-ITP is significantly more computationally costly than 2-ITP-S—the cost of adding a new transformation grows quadratically with the number of selections. Table 4 shows that 2-ITP-S selects more order-2 than order-1 transformations, showing the interest of using higher order transformations. See Fig. 9 for examples.

	CUB SIFT	CUB Color	ILSVRC-30 SIFT	ILSVRC-30 Color
1	crop5×flip	crop5× crop8	flip×crop0	color1
2	crop1×crop0	crop1×crop8	crop6	color1× crop1
3	crop5× crop1	crop1× scale0	flip	flip× color0
4	flip× crop6	crop1×crop5	homo2	crop2×color0
5	crop1× crop6	crop5	flip×crop1	crop2×flip

Table 4: Transformations selected by 2-ITP-S.

4.5. Total training time vs. target accuracy

We plot the test accuracy as a function of the number of iterations performed by the SGD in Fig. 8 (one SGD iteration corresponds to processing a single training image). Interestingly, we observe that the learning process is faster for higher numbers of transformations T. To better exhibit this phenomenon, we show in Table 5 the number of training samples that the SGD algorithm has to process to reach a certain accuracy on the test set. It can be observed that it takes *less time* to reach a certain accuracy when T is larger (i.e. in spite of a larger training set). For instance, on CUB, reaching a 35% accuracy is achieved for $T = 5$ transformations after examining about 2 times less training samples than for $T = 2$. Similarly, on ILSVRC-30, reaching an accuracy of 45% is achieved about 3 times faster for $T = 5$ than for $T = 2$—it corresponds to 35 epochs for $T = 2$ and only 5 epochs for $T = 5$. This is an interesting find-

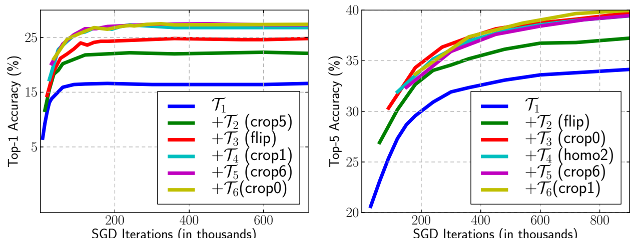

Figure 8: Test accuracy as a function of the number of SGD iterations on CUB (left) and ILSVRC-30 (right), with SIFT.

	CUB - Top-1 acc target					ILSVRC-30 - Top-5 acc target			
T	25%	30%	35%	40%	T	35%	40%	45%	48%
1	**42**	X	X	X	1	129	**683**	X	X
2	29	**48**	**121**	X	2	165	344	**2104**	X
3	31	41	79	**250**	3	**215**	307	886	**7200**
5	33	44	66	141	5	213	298	782	3508

Table 5: Number of training samples (in thousands) that the SGD has to process to reach a given accuracy on the test set, for a varying number of transformations T (here, after fusion). The 'X' sign means that the target accuracy cannot be reached at all.

ing towards reducing the training time on large-scale image classification tasks.

4.6. Comparison to state of the art

CUB. In [23], the authors report results with a similar pipeline. Using Fisher Vectors with 256 Gaussians, their baseline yields 28.2% accuracy on the full images, and 31.0% after cropping the images to 90% (value optimized on the validation set). Note that despite using slightly different descriptors, we obtain the same baseline as they do (28.2%). Our best results on CUB (using 5 order-2 transformations) are significantly higher (45.2%). Our method even outperforms their scheme when they use bounding box annotations for training (42.2% with DPM and 41.9% with their data-driven approach).

ILSVRC 2010. As it is computationally expensive to apply ITP on the full ILSVRC dataset, we instead use the same transformations as those selected on ILSVRC-30. With this set, we learn a classifier using all images of ILSVRC 2010. We experiment only with ITP and 2-ITP-S, as 2-ITP requires to replicate the dataset two times more than 2-ITP-S, which takes too much memory. We report the top-5 accuracy on the test set in Table 6 (right). We improve over our baseline from 70.1% to 74.9% in top-5 accuracy. On the color channel, the improvement is even larger (+7%). In comparison, the winning team of the challenge, NEC-UIUC-Rutgers, achieved 71.8%.

We slightly outperform results in [24], i.e. 74.3% accuracy. Note, that we only use 32K-dim descriptors, while they used 524K-dim descriptors. Recently, [14] reports an 83.0% accuracy, but with a different classification framework (deep learning); see next paragraph for a comparison with their features.

Deep convolutional features. We investigate the effect of ITP on the DeCAF features [11], to show (i) that our approach can significantly improve over features with built-in invariance properties and (ii) that the choice of transformations heavily depends on the features. These features consist of the output of the seventh layer of a deep convolutional network, trained on the images of the ImageNet 2012 challenge. We use the publicly available implementation[2]. We run an ITP pipeline to select five order-one transformations. On CUB flip, crop5, crop8, rot-12° and rot9° were selected. On ILSVRC 2010 flip, crop7, rot3°, jpg70% and scale50% were selected. On CUB, accuracy increases from 47.3% to 55.4% (+8.1), and from 77.9% to 81.4% (+3.5) on ILSVRC 2010. Note that DeCAF features are learned on the images from the 2012 challenge, which only presents a partial overlap with the 2010 one. On CUB, for fair comparison with the Fisher experiments, our results were conducted on the full images, with no prior knowledge on the bounding box, and therefore are lower than [11].

	ILSVRC-30			ILSVRC Full		
	SIFT	Color	Fusion	SIFT	Color	Fusion
Original	34.1	28.9	40.4	64.6	57.5	70.1
ITP	41.8	37.1	48.4	68.7	64.5	74.8
2-ITP-S	42.6	38.3	**49.5**	69.2	64.5	**74.9**

Table 6: Top-5 accuracy for ILSVRC10. (Left) Training with 30 images per class and testing on the validation set. (Right) Training with the full training set and results on the test set.

5. Conclusion

The proposed Image Transformation Pursuit (ITP) algorithm allows to efficiently select a set of informative transformations while at the same time keeping a moderate training cost. Furthermore, it gracefully handles complex 2nd order transformations that combine basic transformations. ITP achieves significant improvements in terms of classification accuracy on both the CUB and the ImageNet 2010 challenge dataset, with only a handful of transformations, both for Fisher-vector and convnet DeCAF features. Another conclusion of our work is that transforming test images and aggregating the corresponding scores is complementary to training set augmentation and leads to significant performance gains.

Acknowledgements This research was partially supported by the grants "Axes" (EU), "Allegro" (ERC), "Gargantua" (CNRS-Mastodons) and "FIRE-ID" (ANR).

[2]`https://github.com/UCB-ICSI-Vision-Group/decaf-release/`

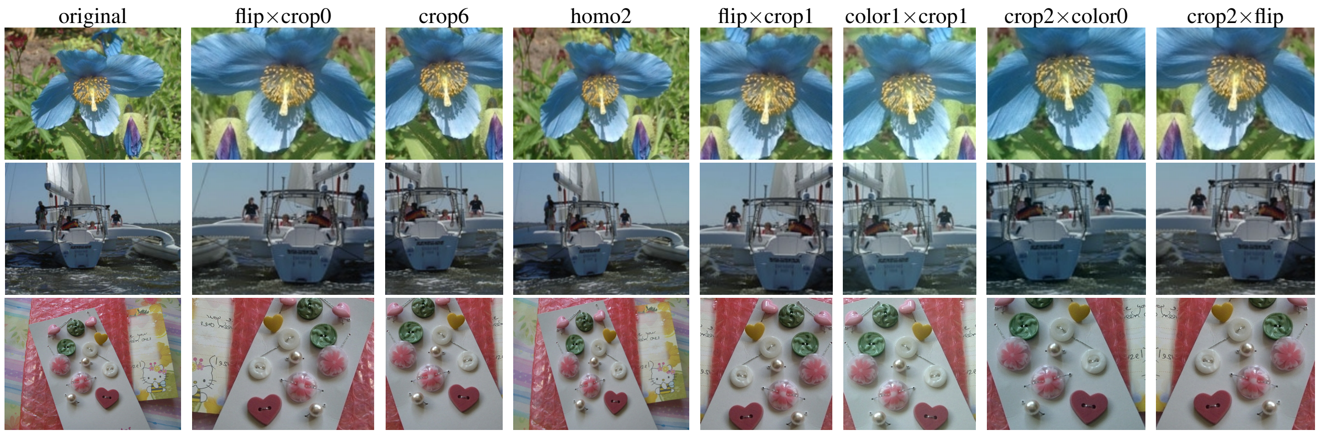

Figure 9: Examples of transformed images on ILSVRC10 (transformations selected by 2-ITP-S, see Table 4).

References

[1] Y. S. Abu-Mostafa. Hints. *Neural Computation*, 7(4), 1995.

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. *IEEE TPAMI*, 2013.

[3] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2009.

[4] C. Bishop. Training with noise is equivalent to Tikhonov regularization. In *Neural computation*, 1995.

[5] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford UP, 1995.

[6] L. Bottou. Stochastic gradient tricks. In *Neural Networks, Tricks of the Trade, Reloaded*. Springer, 2012.

[7] M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*, 2012.

[8] S. Clinchant, G. Csurka, F. Perronnin, and J.-M. Renders. XRCE's participation to Imageval. *ImageEval workshop at CVIR*, 2007.

[9] D. DeCoste and M. Burl. Distortion-invariant recognition via jittered queries. In *CVPR*, 2000.

[10] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 2002.

[11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *ICML*, 2014.

[12] E. Gavves, B. Fernando, C. G. M. Snoek, A. W. M. Smeulders, and T. Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, 2013.

[13] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 2011.

[14] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proc. of the IEEE*, 1998.

[16] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. In *Large Scale Kernel Machines*. MIT Press, 2007.

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[18] L. Maaten, M. Chen, S. Tyree, and K. Q. Weinberger. Learning with marginalized corrupted features. In *ICML*, 2013.

[19] S. Mallat. *A wavelet tour of signal processing (3rd ed.)*. Academic Press, 2008.

[20] J. Marín, D. Vázquez, D. Gerónimo, and A. López. Learning appearance in virtual scenarios for pedestrian detection. In *CVPR*, 2010.

[21] P. Niyogi, F. Girosi, and T. Poggio. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE*, 1998.

[22] L. Pishchulin, A. Jain, C. Wojek, M. Andriluka, T. Thormählen, and B. Schiele. Learning people detection models from few training samples. In *CVPR*, 2011.

[23] J. A. Rodriguez and D. Larlus. Predicting an object location using a global image representation. In *ICCV*, 2013.

[24] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

[25] J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 2012.

[26] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[27] P. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, 2003.

[28] K. Sohn and H. Lee. Learning invariant representations with local transformations. *ICML*, 2012.

[29] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3), 2007.

[30] V. N. Vapnik. *Statistical learning theory*. Wiley, 1998.

[31] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[32] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The CUB-200-2011 Dataset. Technical report, CalTech, 2011.

[33] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[34] L. Yaeger, R. Lyon, and B. Webb. Effective training of a neural network character classifier for word recognition. In *NIPS*, 1996.