Modeling Local and Global Deformations in Deep Learning: Epitomic Convolution, Multiple Instance Learning, and Sliding Window Detection

George Papandreou
Google
gpapan@google.com

Iasonas Kokkinos and Pierre-André Savalle
CentraleSupélec and INRIA
[iasonas.kokkinos,pierre-andre.savalle]@ecp.fr

Abstract

Deep Convolutional Neural Networks (DCNNs) achieve invariance to domain transformations (deformations) by using multiple 'max-pooling' (MP) layers. In this work we show that alternative methods of modeling deformations can improve the accuracy and efficiency of DCNNs. First, we introduce epitomic convolution *as an alternative to the common convolution-MP cascade of DCNNs, that comes with the same computational cost but favorable learning properties. Second, we introduce a* Multiple Instance Learning *algorithm to accommodate global translation and scaling in image classification, yielding an efficient algorithm that trains and tests a DCNN in a consistent manner. Third we develop a DCNN sliding window detector that explicitly, but efficiently, searches over the object's position, scale, and aspect ratio.*

We provide competitive image classification and localization results on the ImageNet dataset and object detection results on Pascal VOC2007.

1. Introduction

Over the last few years Deep Learning has been the method of choice for image classification [24, 26, 40, 41], attaining even super-human performance levels [16, 20], while a host of other works have shown that the features learned by deep neural networks can be successfully employed in other tasks [4, 11, 31, 37, 39, 45]. The key building blocks of deep neural networks for images have been around for many years [25]: (1) Deep Convolutional Neural Networks (DCNNs) with small receptive fields that spatially share parameters within each layer, and (2) gradual abstraction and spatial resolution reduction along the network hierarchy, typically via max-pooling [21, 38]. The recent success of DCNNs can be mostly attributed to large datasets, GPU computing and well-engineered choices.

In this work we aim at enriching the set of tools used to model deformations in DCNNs, by exploiting established computer vision techniques, such as image epitomes, multi-scale pyramids and Procrustes analysis. We combine these techniques with ideas from machine learning (back-propagation, multiple-instance learning), object recognition (image patchworks, sliding window detection), and signal processing (the à trous algorithm) and develop algorithms of higher accuracy and/or efficiency than the ones obtained using more standard deformation modeling tools.

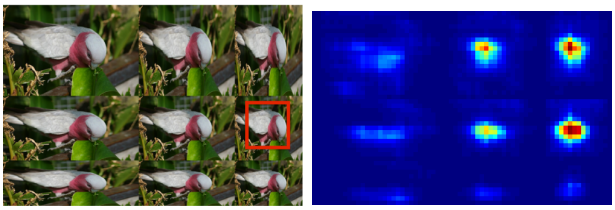


Figure 1. Image deformations can challenge high-level vision, but modeling their effects can lead to simple and accurate recognition algorithms. Here we show how our object detection system performs scale, position and aspect ratio search: scaled and squeezed versions of an image are fed to a fully convolutional DCNN, until at some point the object can be contained in a square of fixed size. At that point the detector's score (shown on the right) is maximized, providing a tight bounding box around the object. Our detector only has to consider normalized object instances.

In Section 2 we deal with the modeling of **local deformations in image classification**. For this we introduce the epitomic image representation [23] into the setting of DCNNs. While originally developed for generative image modeling, we show here that the epitome data structure can be used to train DCCNs discrimatively; we show that while coming at the exact same computation cost, epitomic convolution allows for faster convergence during training and higher classification accuracy when compared to its max-pooled counterpart.

In Section 3 we address to the modeling of **global transformations in image classification**. Our goal is to explicitly deal with object scale and position when applying DCNNs to image classification. While a standard practice is to fuse classification results extracted from multiple im-

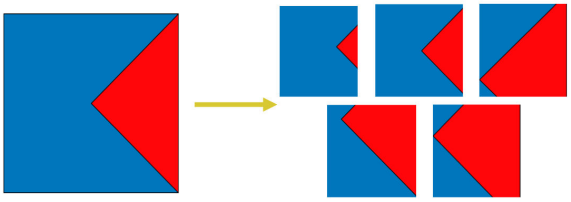


Figure 2. An epitome is a data structure to represent *a set* of small images related to each other through translation and cropping.

(a) (b)

Figure 3. (a) Max-pooled convolution: For each filter we look for its best match within a small window in the data layer. (b) Proposed epitomic convolution: For input data patches sparsely sampled on a regular grid we look for their best match in each epitome.

age windows, we show that by using a principled Multiple Instance Learning (MIL) framework we obtain substantially larger gains. An algorithmic contribution is that we show how MIL can be efficiently implemented for fully-convolutional DCNNs by compacting an image pyramid into the patchwork data structure of [9].

Finally, in Section 4 we turn to the modeling of **global transformations in object detection**. Rather than using region proposals to come up with candidate object boxes, we explicitly search over positions, scales, and aspect ratios, as illustrated in Fig. 1; this can be understood as a variant of Procrustes analysis used in AAMs [5], where global deformations are first discarded before performing a finer deformation modeling. We show that by performing this explicit search over position, scale, and aspect ratios we can obtain results that are comparable to the current-state-of-the-art while being substantially simpler and easier to train, as well as six times faster, thanks to the sharing of computation during convolutions. An algorithmic contribution that we introduce in this context is that we accelerate sliding window detection by using the à trous (with holes) algorithm to reduce the effective size and receptive field of a DCNN pre-trained on ImageNet.

Since these contributions are to some extent orthogonal, we describe prior work and provide experimental results separately within each section. We have implemented the proposed methods using Caffe [22]; code and models are made publicly available from http://cvn.ecp.fr/iasonas/deepdet.

2. Deep Epitomic Convolutional Networks

2.1. From flat to deep: Epitomic Convolution

Image epitomes The image epitome, illustrated in Fig. 2, is a data structure introduced in [23] to learn translation-aware image representations. An epitome is a single, large image patch which can produce several small patches by first picking an epitome position and then cropping a small window; it can be understood as a 'pallette' of patches from which we can pick a patch at will.

Epitomic Convolution We propose to use epitomes as an efficient method for parameter sharing and local deformation modeling in Deep Learning. In particular, we introduce *Epitomic Convolution* as a new module in multi-layered architectures. It acts as an alternative and has the same complexity as a consecutive pair of convolution and max-pooling layers. Its main asset is that, by virtue of relying on the epitome data structure, it allows us to share parameters across the different patch models, thereby allowing for faster convergence and better generalization.

Epitomic Convolution and its relationship to max-pooled convolution are illustrated in Fig. 3. In max-pooling we search in the input image for the strongest response of a filter. Instead, in Epitomic Convolution we search across the set of smaller filters encapsuled in an epitome for the strongest response to an image patch.

In more detail, in standard max-pooled convolution we have a dictionary of K filters of spatial size $W \times W$ pixels spanning C channels, which we represent as real-valued vectors $\{\mathbf{w}_k\}_{k=1}^K$ with $W \cdot W \cdot C$ elements. We apply each of them in a convolutional fashion to every $W \times W$ input patch $\{\mathbf{x}_i\}$ densely extracted from each position in the input layer which also has C channels. A reduced resolution output map is produced by computing the maximum response within a small $D \times D$ window of displacements $p \in \mathcal{N}_{input}$ around positions i in the input map which are D pixels apart from each other. The output map $\{z_{i,k}\}$ of max-pooled convolution has spatial resolution reduced by a factor of D across each dimension and consists of K channels, one for each of the K filters. Specifically:

$$(z_{i,k}, p_{i,k}) \leftarrow \max_{p \in \mathcal{N}_{image}} \mathbf{x}_{i+p}^T \mathbf{w}_k \quad (1)$$

where $p_{i,k}$ points to the input layer position where the maximum is attained (argmax).

In the proposed epitomic convolution scheme we replace the filters with mini-epitomes $\{\mathbf{v}_k\}_{k=1}^K$ of spatial size $V \times V$ pixels, where $V = W + D - 1$. Each mini-epitome contains D^2 filters $\{\mathbf{w}_{k,p}\}_{k=1}^K$ of size $W \times W$, one for each of the $D \times D$ displacements $p \in \mathcal{N}_{epit}$ in the epitome. We *sparsely* extract patches $\{\mathbf{x}_i\}$ from the input layer on a regular grid with stride D pixels. In the proposed epitomic convolution model we reverse the role of filters and input layer patches, computing the maximum response over epitomic positions

rather than input layer positions:

$$(y_{i,k}, p_{i,k}) \leftarrow \max_{p \in \mathcal{N}_{epitome}} \mathbf{x}_i^T \mathbf{w}_{k,p} \quad (2)$$

where $p_{i,k}$ now points to the position in the epitome where the maximum is attained. Since the input position is fixed, we can think of epitomic matching as an input-centered dual alternative to the filter-centered standard max-pooling.

Similarly to max-pooled convolution, the epitomic convolution output map $\{y_{i,k}\}$ has K channels and is subsampled by a factor of D across each spatial dimension. Epitomic convolution has the same computational cost as max-pooled convolution. For each output map value, they both require computing D^2 inner products followed by finding the maximum response. Epitomic convolution requires D^2 times more work per input patch, but this is exactly compensated by the fact that we extract input patches sparsely with a stride of D pixels.

Epitomic DCNNs To build a deep epitomic model, we stack multiple epitomic convolution layers on top of each other. The output of each layer passes through a rectified linear activation unit $y_{i,k} \leftarrow \max(y_{i,k}+\beta_k, 0)$ and fed as input to the subsequent layer, where β_k is the bias. We learn the model parameters (epitomic weights and biases for each layer) in a supervised fashion by error back propagation. We detail our model architecture and training methodology in the experimental section.

Related work Our model builds on the epitomic image representation [23], which was initially geared towards image and video modeling tasks. Single-level dictionaries of image epitomes learned in an unsupervised fashion for image denoising have been explored in [1,3]. Recently, single-level mini-epitomes learned by a variant of K-means have been proposed as an alternative to SIFT for image classification [35]. To our knowledge, epitomes have not been studied before in conjunction with deep models or learned to optimize a supervised objective.

Computing the maximum response over filters rather than image positions, as we do in Eq. 2 resembles the maxout scheme of [14]. Similarly to epitomic matching, the response of a maxout layer is the maximum across filter responses. But, the epitomic layer is hard-wired to model position invariance, since filters extracted from an epitome share values in their area of overlap. This parameter sharing significantly reduces the number of free parameters that need to be learned. Maxout is typically used in conjunction with max-pooling [14], while epitomes fully substitute it.

2.2. Image Classification Experiments

Image classification tasks We quantitatively evaluate the proposed deep epitomic models in image classification experiments on the Imagenet ILSVRC-2012 large-scale image classification task [7]. This dataset contains more than 1.2 million training images, 50,000 validation images, and 100,000 test images. Each image is assigned to one out of 1,000 possible object categories. Performance is evaluated using the top-5 classification error.

Layer	1	2	3	4	5	6	7	8	Out
Type	conv + lrn + max	conv + lrn + max	conv	conv	conv	conv + max	full + dropout	full + dropout	full
Output size	96	192	256	384	512	512	4096	4096	1000
Filter size	8x8	6x6	3x3	3x3	3x3	3x3	-	-	-
Input stride	2x2	1x1	1x1	1x1	1x1	1x1	-	-	-
Pooling size	3x3	2x2	-	-	-	3x3	-	-	-

Table 1. Architecture of the baseline *Max-Pool* DCNN (Class-A).

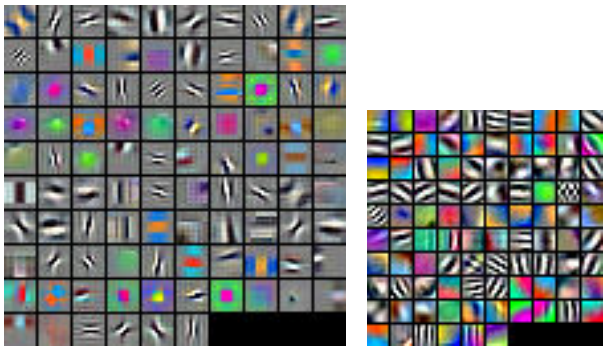
Layer	1	2	3	4	5	6	7	8	Out
Type	epit-conv + lrn	epit-conv + lrn	conv	conv	conv	epit-conv	full + dropout	full + dropout	full
Output size	96	192	256	384	512	512	4096	4096	1000
Epitome size	12x12	8x8	-	-	-	5x5	-	-	-
Filter size	8x8	6x6	3x3	3x3	3x3	3x3	-	-	-
Input stride	4x4	3x3	1x1	1x1	1x1	3x3	-	-	-
Epitome stride	2x2	1x1	-	-	-	1x1	-	-	-

Table 2. Architecture of the proposed *Epitomic* DCNN (Class-A).

Network architecture and training methodology For our Imagenet experiments, we compare the proposed deep epitomic networks with deep max-pooled convolutional networks. We use as similar architectures as possible, involving in both cases six convolutional layers, followed by two fully-connected layers and a 1000-way softmax layer. We use rectified linear activation units throughout the network. Similarly to [24], we apply local response normalization (LRN) to the output of the first two convolutional layers and dropout to the output of the two fully-connected layers. We refer to these as Class-A models.

The architecture of our baseline *Max-Pool* network is specified on Table 1. It employs max-pooling in the convolutional layers 1, 2, and 6. To accelerate computation, it uses an image stride equal to 2 pixels in the first layer. It has a similar structure with the Overfeat model [39], yet significantly fewer neurons in the convolutional layers 2 to 6. Another difference with [39] is the use of LRN, which to our experience facilitates training.

The architecture of the proposed *Epitomic* network is specified on Table 2. It has exactly the same number of neurons at each layer as the *Max-Pool* model but it uses mini-epitomes in place of convolution + max pooling at layers 1, 2, and 6. It uses the same filter sizes with the *Max-Pool* model and the mini-epitome sizes have been selected so as to allow the same extent of translation invariance as the corresponding layers in the baseline model. We use input image stride equal to 4 pixels and further perform epitomic search with stride equal to 2 pixels in the first layer to also



(a) (b)

Figure 4. Filters at the first convolutional layer of: (a) Proposed *Epitomic* model with 96 mini-epitomes, each having size 12×12. (b) Baseline *Max-Pool* model with 96 filters of size 8×8 each.

accelerate computation.

We have also tried variants of the two models above where we mean- and contrast- normalize the filters in layers 1, 2, and 6 of the networks before computing the neuron responses, similarly to [45]. We refer to the supplementary material for more details.

We closely follow the methodology of [24] in training our models, in terms of weight initialization, learning rate, weight decay, and momentum parameter selection. Training each of the three models takes two weeks using a single NVIDIA Titan GPU. Similarly to [4], we resize the training images to have small dimension equal to 256 pixels while preserving their aspect ratio. We also subtract for each image pixel the global mean RGB color values computed over the whole Imagenet training set. During training, we present the networks with 220×220 crops randomly sampled from the resized image area, flipped left-to-right with probability 0.5, also injecting global color noise exactly as in [24]. During evaluation, we present the networks with 10 regularly sampled image crops (center + 4 corners, as well as their left-to-right flipped versions).

Weight visualization We visualize in Fig. 4 the layer weights at the first layer of the networks above. The networks learn receptive fields sensitive to edge, blob, texture, and color patterns.

Classification results We report at Table 3 our results on the Imagenet ILSVRC-2012 benchmark, also including results previously reported in the literature [24, 39, 45]. These all refer to the top-5 error on the validation set and are obtained with a single network. Our best result at 13.6% with the proposed *Epitomic-Norm* network is 0.6% better than the baseline *Max-Pool* result at 14.2% error. The improved performance that we got with the *Max-Pool* baseline network compared to Overfeat [39] is most likely due to our use of LRN and aspect ratio preserving image resizing.

	Previous literature			Class-A				Class-B	Class-C
Model	Krizhevsky [24]	Zeiler-Fergus [45]	Overfeat [39]	Max-Pool	Max-Pool + norm	Epitomic	Epitomic + norm	Epitomic +norm	Epitomic +norm
Top-5 Error	18.2%	16.0%	14.7%	14.2%	14.4%	**13.7%**	**13.6%**	11.9%	10.0%

Table 3. Imagenet ILSVRC-2012 top-5 error on validation set. All performance figures are obtained with a single network, averaging classification probabilities over 10 image crops (center + 4 corners, as well as their left-to-right flipped versions). Classes B and C refer to respectively larger and deeper models.

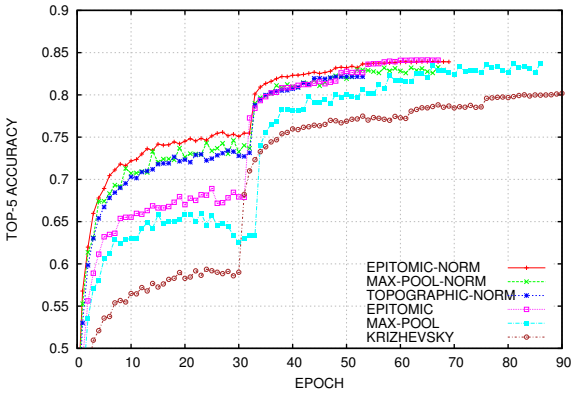


Figure 5. Top-5 validation set accuracy (center non-flipped crop only) for different models and normalization.

We show in Fig. 5 how the top-5 validation error improves as learning progresses for the different models we tested, with or without mean+contrast normalization. For reference, we also include a corresponding plot we reproduced for the original model of Krizhevsky et al. [24]. We observe that mean+contrast normalization significantly accelerates convergence of both epitomic and max-pooled models, without however significantly influencing the final model quality. The epitomic models converge faster and are stabler during learning compared to the max-pooled baselines, whose performance fluctuates more.

In [34] we report further successful experiments with deep epitomic networks on the Caltech-101, MNIST, and CIFAR-10 datasets. Finally, we discuss another deep epitomic network variant built on top of large epitomes which learns topographically organized features.

Experiments with larger and deeper epitomic networks We have also experimented with larger (Class-B) and very deep (Class-C) versions of the proposed deep epitomic networks. The large Class-B network has the same number of levels but more neurons per layer than the networks in Class-A. It achieves an error rate of 11.9%.

Inspired by the success of the top-performing methods in this year's Imagenet competition, we have also very recently experimented with a very deep network having 13 convolutional and 3 fully connected layers, which roughly follows the architecture of the 16 layer net in [40]. Our

Class-C deep epitomic network achieves 10.0% error rate in the Imagenet task. The state-of-art 16 layer net in [40] (without multi-scale training/testing) achieves an even lower 9.0% error rate, but using a more sophisticated procedure for aggregating the results of multiple image crops (in place of our simple 10-view testing procedure). Using our current methodology, we were able to train a model similar to that of [40] which could only achieve a 10.8% error rate. As extra evidence to the improved robustness of training our deep epitomic networks, we mention that we managed to train our very deep epitomic net starting from a random initialization, while [40] had to bootstrap their very deep networks from shallower ones.

3. Scale and Position Search in Classification

When used in alternation with feature downsampling ('striding', or 'decimation') a cascade of Max-Pooling or Epitomic Convolution layers can result in invariance to increasingly large-scale signal transformations, eventually dealing with global position and scale changes - achieving 'invariance by a thousand cuts'.

We argue that a better treatment of deformations can be achieved by factoring deformations into local (non-rigid) and global (translation/scale) changes. We can then explicitly simulate the effect of the global changes during training and testing, by transforming the input images, while casting model training from weak annotations as a Multiple Instance Learning (MIL) problem.

MIL-based training of DCNNs Considering a binary classification problem with N image-label pairs $\mathcal{S} = \{(X_i, y_i)\}, i = 1, \ldots, N$, training aims at minimizing:

$$C(f, \mathcal{S}) = \sum_{i=1}^{N} l(y_i, f(X_i)) + R(f), \tag{3}$$

where f is the classifier, $l(y, f(X))$ is the loss function and R is a regularizer. Our goal is to deal with the effects of deformations of the inputs X_i during training and testing.

Dataset augmentation amounts to turning an image X_i into a set of images $\mathbf{X}_i = \{X_i^1, \ldots, X_i^K\}$ by transforming $\mathbf{X}_i$ synthetically; e.g. considering T translations and S scalings yields a set with $K = TS$ elements. The most common approach to using dataset augmentation consists in treating each element of $\mathbf{X}_i$ as a new training sample, i.e. substituting the loss $l(y, f(X_i))$ in Eq. 3 by the sum of the classifier's loss on all images:

$$L(y_i, \mathbf{X}_i) \doteq \sum_{k=1}^{K} l(y_i, f(X_i^k)). \tag{4}$$

This corresponds to the dataset augmentation technique used e.g. in [18]. A recently introduced alternative is the 'sum-pooling' technique used in [40], which can be understood as using the following loss:

$$L(y_i, \mathbf{X}_i) = l(y_i, \frac{1}{K} \sum_{k=1}^{K} f(X_i^k)), \tag{5}$$

which averages the classifier's score over translated versions of the input image. The summation used in both of these approaches favors classifiers that consistently score highly on positive samples, irrespective of the object's position and scale - because this is when the loss is minimized. As such, these classifiers pursue the invariance of f.

There is however a tradeoff between invariance and classification accuracy [43]. Even though pursuing invariance accounts for the effects of transformations, it does not make the classification task any easier: the training objective aims at a classifier that would allow all transformed images to make it through its 'sieve'. By contrast, classifying objects at only a fixed scale can result in higher accuracy, since we have lower intra-class variability and can devote all modelling resources to the treatment of local deformations.

To achieve this, we let our classifier 'choose' a preferred transformation, and define the loss function to be:

$$L(y_i, \mathbf{X}_i) = l(y_i, \max_k f(X_i^k)), \tag{6}$$

which amounts to letting the classifier choose the transformation that maximizes its response on a per-sample basis, and then accordingly penalizing that response. In particular the loss function requires the classifier's response to be large on at least one position for a positive sample - and small everywhere for a negative. This criterion allows us to train a more 'picky' classifier, with a response that decays as the object deviates from a desired scale.

This idea amounts to the simplest case of Multiple Instance Learning [8]: $\mathbf{X}_i$ can be seen as a *bag of features* and the individual elements of $\mathbf{X}_i$ can be seen as *instances*. For the particular case of the hinge loss function, this would lead us to latent-SVM training [2, 10]. There exist a broad variety of MIL alternatives, that can potentially achieve higher robustness by using more than the max-scoring variables [17, 19, 36] - but the one we propose here is very easy to implement with DCNNs, as we will describe below.

Using this loss function during training amounts to treating the object's position and scale as a latent variable, and performing alternating optimization over the classifier's score function. During testing we perform a search over transformations and keep the best classifier score, which can be understood as maximizing over the latent transformation variables. The resulting score $F(\mathbf{X}_i) = \max_k f(X_i^k)$ is transformation-invariant, but is built on top of a classifier tuned for a single scale- and translation- combination.

Apart from allowing us to train a more 'picky' classifier, it is equally important that the MIL setting allows us

Model	Epitomic (Class-B)	Epitomic (patchwork)
Top-5 Error	11.9%	10.0%

Table 4. Imagenet ILSVRC-2012 top-5 error on validation set. We compare the Class-B mean and contrast normalized deep epitomic network of Table 3 with its Patchwork fine-tuned version that also includes scale and position search.

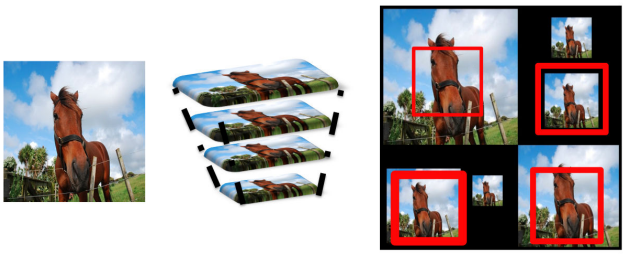


Figure 6. We use image patchworks to efficiently implement scale and position search in DCNN training: an image pyramid is unfolded into a image 'patchwork', where sliding a fixed-size window amounts to search over multiple positions and scales. The maximum classifier score on all such windows is efficiently gathered by max-pooling the DCNN's top-layer responses, accommodating scale and position changes during both training and testing.

to train and test our classifiers consistently, using the same set of image translations and scalings during both phases. This is in contrast to the use of ad-hoc image scaling during training and multiple views during testing which is commonplace in current classification practice.

Efficient Implementation with Fully Convolutional DCNNs and Patchworks We now turn to practical aspects of integrating MIL into DCNN training. DCNNs are commonly trained with input images of a fixed size, $q \times q$. For an arbitrarily-sized input image I, if we denote by $\mathcal{I}(x, y, s)$ its image pyramid, naively computing the maximization in Eq. 6 would require cropping many $q \times q$ boxes from $\mathcal{I}(x, y, s)$, evaluating f on them, yielding $f(x, y, s)$, and then penalizing $l(y, \max_{x,y,s} f(x, y, s))$ during training (we ignore downsampling and boundary effects for simplicity). Doing this would require a large amount of GPU memory, communication and computation time. Instead, by properly modifying the input and architecture of our network we can share computation to efficently implement exhaustive search during training and testing.

For this, we first draw inspiration from the 'image patchwork' technique introduced in [9] and exploited in DCNNs by [12]. The technique consists in embedding a whole image pyramid $\mathcal{I}(x, y, s)$, into a single, larger, patchwork image $P(x', y')$; any position (x', y') in P corresponds to a (x, y, s) combination in I. This was originally conceived as a means of accelerating multi-scale FFT-based convolutions in [9] and convolutional feature extraction in [12]. Instead we view it a stepping stone to implementing scale and position search during DCNN training.

In particular, as in [27, 29, 39], we transform our DCNN into a fully-convolutional network and treat the last fully-connected layers as 1×1 convolution kernels. We can thus obtain the $f(x, y, s)$ score described above by providing P as input to our network, since the output of our network's final layer at any position (x', y') will correspond to the output corresponding to a $q \times q$ square cropped around (x, y, s). This allows us to incorporate the $\max$ operation used in MIL's training criterion, Eq. 6 as an additional max-pooling layer situated on top of the network's original score function, conveniently incorporating global scale and position search in DCNN training. Namely, we can now back-propagate the error message from the classification loss to the units that resulted in the winning scale and position combination.

Image Classification Results We have experimented with the scheme outlined above in combination with our Deep Epitomic Network (Class-B variant) presented in the previous Section. We use a 720×720 patchwork formed from 6 different image scales (square boxes with size 400, 300, 220, 160, 120, and 90 pixels). We have resized all train/test images to square size, changing their aspect ratio if needed. We initialized this scale/position search net with the parameters of our standard Class-B epitomic net and fine-tuned the network parameters for about an epoch on the Imagenet train set.

We have obtained a substantial decrease in the testing error rates, cutting the top-5 error rate from 11.9% down to 10.0%, as shown in Table 4. This reduction in error rate is competitive with the best reduction obtained by more complicated techniques involving many views for evaluation [40, 40], while also allowing for consistent end-to-end training and testing.

Object Localization Results The network outlined above also provides cues for the scale and position of the dominant object in the image. A simple fixed mapping of the "argmax" patchwork position in the last max-pooling layer (computed by averaging the bounding box positions in the training set) yields a 48.3% error rate in the Imagenet 2012 localization task without incurring any extra computation. We note that a similar object localization idea is also proposed in [29] for unsupervised object discovery, but without the maximization over scales.

4. Scale, Position and Ratio Search in Detection

Sliding windows vs. region proposals Sliding window detectors are known to deliver excellent results for ob-

jects such as pedestrians, e.g. [32], however, recent works on combining convolutional networks [39], or sliding window detectors with CNN features [13, 33, 44] still lag substantially behind the current state-of-the-art techniques [11, 15, 30] that use region proposals delivered, e.g. by selective search [42].

Such approaches deliver compelling detection results: in [11] combining RCNNs with the VGG network of [40] pushed the mean AP (mAP) performance on Pascal VOC 2007 to 62.2% (66% with bounding box regression). However, such techniques come with a high cost, requiring separate feature extraction per region - e.g the system of [11] requires 60-70 seconds per image, making it impractical for real-time applications.

Sliding window classifiers seem can turn out to be more efficient in the setting of DCNNs, since computation is naturally shared across different positions through convolutions. Apart from speed, another advantage of sliding window detection is simplicity, as it does not involve a segmentation front-end. Still the performance gap between sliding window classifiers and region proposal algorithm is present, so one would need to sacrifice accuracy for speed; we now turn to how one can avoid this.

Aspect ratio variability and search Sliding window detectors have typically substantially lower performance than region-based systems, even when trained with CNN features [13,33,44]; e.g. in [33], when training a DPM with the same features as an RCNN system we obtained lower mAP scores, while using apparently more powerful classifiers.

Our starting point for this work is our understanding from our results in [33] that this may be due to the DPM using a mixure of ratio-tuned classifiers, which practically splits the training set size by three per component - thereby making overfitting easier. Instead, as illustrated in Fig. 1, we propose to have a single classifier tuned to a fixed aspect ratio (a square) and then let the image fit to the classifier. This is similar to the approach we developed in Sec. 3 for scale-invariance, but now applied to 'squeeze'-invariance - and is commonly used in AAM learning, under the name of 'Procrustes analysis' [6].

In particular, for any given image we consider 5 aspect ratio transformations, spanning $[1/3, 3]$ with a geometric progression. For every transformed image we compute an image pyramid, and apply our fully convolutional DCNN to each level - which amounts to a joint search of scale, aspect ratio and position. This search over aspect ratios is five times more demanding than a single pyramid evaluation - but ends up requiring about 10 seconds on a Tesla K40 GPU for an average Pascal VOC 2007 image, thanks to acceleration techniques that we describe below. This is already six times faster than the RCNN counterpart - while our method can clearly work in 2 seconds if we work with objects of a fixed aspect ratio, such as faces.

Re-purposing DCNN for efficient detection We have repurposed the publicly available state-of-art 16 layer classification network of [40] (VGG-16) into an efficient and accurate sliding window detector. Our changes consist in (a) reducing the number of network parameters (b) introducing the à trous algorithm to compute the DCNN scores more densely and (c) fine-tuning the network with bounding boxes that reflect its operation test-time.

Network simplification We have accelerated our network by spatially subsampling the first fully-connected layer to a 4×4 spatial size, keeping only the interior part of the fully connected nodes. This reduces the receptive field of the network down to 128×128 pixels (rather than 224×224 required by the VGG network) and accelerates the computation time for the first FC layer by approximately 3 times. We experimented with using such a smaller network within the system of [11] and observed that performance falls only moderately, from 62.2mAP down to 59.6mAP.

Dense feature extraction with the à trous algorithm Direct evaluation of our network in a convolutional fashion yields very sparsely computed detection scores, having a stride of 32 pixels. In order to compute scores more densely at a target stride of 8 pixels, we use the 'hole algorithm' (à trous algorithm), first developed for efficient computation of the undecimated wavelet transform, as described in [28].

The output $y[i]$ of the à trous convolution of a 1-D input signal $x[i]$ with a filter $w[i]$ of length K is defined as

$$y[i] = \sum_{k=1}^{K} x[i + s \cdot k] \cdot w[k] \tag{7}$$

Atrous convolution thus generalizes standard convolution by introducing the input stride parameter s. This allows us to compute DCNN network responses during testing at arbitrarily high resolution. For example, in order to double the output density, it suffices to skip decimation after the last max-pooling layer and use à trous convolution with input stride $s = 2$ pixels in all subsequent convolutional layers.

DCNN finetuning for sliding window detection As in [11] we adapt a network trained for Imagenet classification to perform object detection on Pascal VOC. In [11], the regions proposed by selective search generate positive and negative training samples as follows: if a region has an Intersection-over-Union (IoU) above .5 for any bounding box of one of the 20 Pascal classes it is declared as being a positive example, or else it is a negative example. These examples are used in a network 'fine-tuning' stage, which

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Our work (VGG)	64.1	72.3	**62.8**	**44.0**	**44.2**	66.4	72.5	67.7	35.2	68.9	35.9	62.7	**69.0**	65.7	**65.8**	**36.2**	60.1	50.3	63.2	66.0	58.6
RCNN7 [11] (VGG)	**71.6**	**73.5**	58.1	42.2	39.4	**70.7**	**76.0**	**74.5**	**38.7**	**71.0**	**56.9**	**74.5**	67.9	**69.6**	59.3	35.7	**62.1**	**64.0**	**66.5**	**71.2**	**62.2**
RCNN7 [11] (UoT)	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
E2E-DPM [44] (NYU)	49.3	69.5	31.9	28.7	40.4	61.5	61.5	41.5	25.5	44.5	47.8	32.0	67.5	61.8	46.7	25.9	40.5	46.0	57.1	58.2	46.9
CNN-DPM [33] (UoT)	39.7	59.5	35.8	24.8	35.5	53.7	48.6	46.0	29.2	36.8	45.5	42.0	57.7	56.0	37.4	30.1	31.1	50.4	56.1	51.6	43.4
MP-DPM [13] (UoT)	44.6	65.3	32.7	24.7	35.1	54.3	56.5	40.4	26.3	49.4	43.2	41.0	61.0	55.7	53.7	25.5	47.0	39.8	47.9	59.2	45.2

Table 5. Detection mean Average Precision (%) on the PASCAL VOC 2007 test set, using the proposed CNN sliding window detector that performs explicit position, scale, and aspect ratio search. We compare to the RCNN architecture of [11] End-to-End (E2E) trained DPMs of [44], the DPM with CNN features (CNN-DPM) of [33] and the Max-Pooled (MP) DPM of [13]. In parenthesis we indicate the DCNN used for detection: UoT is the University of Toronto DCNN [24], VGG is the DCNN of Oxford's Visual Geometry Group [40], NYU is the custom network of [44].

amounts to running back-propagation to correctly classify these training samples. Once finetuning converges, the last layer of the network is retrained separately with a different SVM objective, and with a new definition of positives and negatives, using different overlap thresholds - since a common set of IoU thresholds would deteriorate performance.

We deviate from this training procedure in that we do not rely on the Selective Search [42] region proposals to gather training samples and do not use SVM training posterior to finetuning. In particular when gathering training samples for finetuning we keep track of all windows that would be visited by our sliding window detector; given a ground-truth bounding box, we randomly pick among them at most 50 windows that have an IoU score above 0.7 with it; we typically find so many windows, while gathering as many positive windows would not be possible for selective search windows, as they more sparsely cover the bounding boxes of the objects. Furthermore, inspired from [13], for every positive bounding box we sample 200 negative boxes that have an IoU score between 0.2 and 0.5 - which seems to be the hardest regime for sliding window detectors.

This denser sampling of candidate positions gave us substantially better localized sliding window scores than the ones obtained with the -sparser- windows used for finetuning in [11]. The score of our network trained with logistic regression was sufficiently well localized, and we observed that SVM retraining of the last layer using the system of [11] did not improve performance.

Detection Results Detection results on the test set of Pascal VOC 2007 are shown in Table 5. Aspect ratio search yields a mean Average Precision score of 58.6%, and has the highest mAP for 6 out of 20 categories.

Compared to the best sliding window detectors that employ CNN features (Rows 4-6), we have a substantially better performance: we score 11.7 points higher than the best mAP result of 46.9% reported in [44] - which can be attributed to our explicit search over aspect ratios, and the use of a more powerful DCNN classifier. The system of [44] also integrates non-maximum suppression into training, which is an interesting direction to explore in order to improve our detection.

Comparing to the RCNN system [11] with the VGG network [40] we are only doing better on 6 out of 20 categories, and our mAP is 3.6% below theirs. This is to some extent anticipated, given the smaller network and image sizes used by our system, in order to achieve our sixfold acceleration. But when compared to RCNN using the network of [24] - which takes the same amount of time as our system does- our performance is 4.2mAP points higher.

Apart from the efficiency/accuracy gains, we consider simplicity as a key advantage of our system, as (i) it does not require a segmentation front-end (ii) only has a single finetuning stage (iii) deploying the detector is as simple as computing an image pyramid and providing the images as inputs to a fully-convolutional DCNN with a few lines of MATLAB code.

5. Conclusions

This paper examines multiple facets of invariance in the context of deep convolutional networks for visual recognition. First, we have proposed a new epitomic convolutional layer which acts as a substitute to a pair of consecutive convolution and max-pooling layers, and shown that it brings performance improvements and exhibits better behavior during training. Second, we have demonstrated that treating scale and position as latent variables and optimizing over them during both training and testing yields significant image classification performance gains. Pushing scale and position search further, we have shown that DCNNs can be efficient and effective for dense sliding window detection.

Acknowledgments We gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs used for this research. GP was with the Toyota Technological Institute at Chicago when this work was initiated. This work has been supported by EU Projects RECONFIG FP7-ICT-600825 and MOBOT FP7-ICT-2011-600796.

References

[1] M. Aharon and M. Elad. Sparse and redundant modeling of image content using an image-signature-dictionary. *SIAM J. Imaging Sci.*, 1(3):228–247, 2008. 3

[2] S. Andrews, T. Hofmann, and I. Tsochantaridis. Support vector machines for multiple instance learning. In *NIPS*, 2002. 5

[3] L. Benoît, J. Mairal, F. Bach, and J. Ponce. Sparse image representation with epitomes. In *Proc. CVPR*, pages 2913–2920, 2011. 3

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. arXiv, 2014. 1, 4

[5] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Trans. PAMI*, 23(6):681–685, 2001. 2

[6] T. Cootes and C. Taylor. Constrained active appearance models. In *Proc. ICCV*, volume 1, pages 748–754, 2001. 7

[7] J. Deng, W. Dong, R. Socher, L. Li-Jia, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 3

[8] T. G. Dietterich, R. H. Lathrop, and T. Lozano-perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997. 5

[9] C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. In *Computer Vision–ECCV 2012*, pages 301–311. Springer Berlin Heidelberg, 2012. 2, 6

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 32(9):1627–1645, 2010. 5

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014. 1, 7, 8

[12] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. *arXiv:1409.5403*, 2014. 6

[13] R. B. Girshick, F. N. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. *CoRR*, abs/1409.5403, 2014. 7, 8

[14] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proc. ICML*, 2013. 3

[15] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 7

[16] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. 1

[17] M. Hoai and A. Zisserman. Improving human action recognition using score distribution and ranking. In *Asian Conference on Computer Vision*, 2014. 5

[18] A. G. Howard. Some improvements on deep convolutional neural network based image classification, 2013. arXiv:1409.0575. 5

[19] Y. Hu, M. Li, and N. Yu. Multiple-instance ranking: Learning to rank images for image retrieval. In *Proc. CVPR*, 2008. 5

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 1

[21] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proc. ICCV*, pages 2146–2153, 2009. 1

[22] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding, 2013. 2

[23] N. Jojic, B. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *Proc. ICCV*, pages 34–41, 2003. 1, 2, 3

[24] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. NIPS*, 2013. 1, 3, 4, 8

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. 1

[26] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014. 1

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. 6

[28] S. Mallat. *A Wavelet Tour of Signal Processing*. Acad. Press, 2 edition, 1999. 7

[29] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Weakly supervised object recognition with convolutional neural networks. Technical Report HAL-01015140, INRIA, 2014. 6

[30] W. Ouyang, P. Luo, X. Zeng, S. Qiu, Y. Tian, H. Li, S. Yang, Z. Wang, Y. Xiong, C. Qian, Z. Zhu, R. Wang, C. C. Loy, X. Wang, and X. Tang. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *CoRR*, abs/1409.3505, 2014. 7

[31] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In *Proc. ICCV*, 2013. 1

[32] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In *ICCV*, 2013. 7

[33] P.-A. Savalle and S. Tsogkas and G. Papandreou and I. Kokkinos. Deformable part models with cnn features. In *3rd Parts and Attributes Workshop, ECCV.*, 2014. 7, 8

[34] G. Papandreou. Deep epitomic convolutional neural networks. arXiv, 2014. 4

[35] G. Papandreou, L.-C. Chen, and A. Yuille. Modeling image patches with a generic dictionary of mini-epitomes. In *Proc. CVPR*, 2014. 3

[36] S. Ray and M. Craven. Supervised versus multiple instance learning: an empirical comparison. In *Proc. ICML*, 2005. 5

[37] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *Proc. CVPR Workshop*, 2014. 1

[38] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999. 1

[39] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2014. 1, 3, 4, 6, 7

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. http://arxiv.org/abs/1409.1556/. 1, 4, 5, 6, 7, 8

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 1

[42] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. 2013. 7, 8

[43] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, 2007. 5

[44] L. Wan, D. Eigen, and R. Fergus. End-to-end integration of a convolutional network, deformable parts model and non-maximum suppression. arXiv, 2014. 7, 8

[45] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. arXiv, 2013. 1, 4