MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Deformable Part Networks

Zhang, Z.; Lin, R.; Sullivan, A.

TR2018-071 July 12, 2018

Abstract

In this paper we propose novel Deformable Part Networks (DPNs) to learn poseinvariant representations for 2D object recognition. In contrast to the state-of-theart pose-aware networks such as CapsNet [30] and STN [20], DPNs can be naturally interpreted as an efficient solver for a challenging detection problem, namely Localized Deformable Part Models (LDPMs) where localization is introduced to DPMs as another latent variable for searching for the best poses of objects over all pixels and (predefined) scales. In particular we construct DPNs as sequences of such LDPM units to model the semantic and spatial relations among the deformable parts as hierarchical composition and spatial parsing trees. Empirically our 17-layer DPN can outperform both CapsNets and STNs significantly on affNIST [30], for instance, by 19.19% and 12.75%, respectively, with better generalization and better tolerance to affine transformations.

arXiv

Deformable Part Networks

Ziming Zhang[†], **Rongmei Lin**[‡], **Alan Sullivan**[†]
[†]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139-1955
[‡]Department of Computer Science, Emory University, Atlanta, GA 30322
{zzhang, sullivan}@merl.com, rongmei.lin@emory.edu

Abstract

In this paper we propose novel Deformable Part Networks (DPNs) to learn *pose-invariant* representations for 2D object recognition. In contrast to the state-of-the-art pose-aware networks such as CapsNet [30] and STN [20], DPNs can be naturally *interpreted* as an efficient solver for a challenging detection problem, namely Localized Deformable Part Models (LDPMs) where localization is introduced to DPMs as another latent variable for searching for the best poses of objects over all pixels and (predefined) scales. In particular we construct DPNs as sequences of such LDPM units to model the semantic and spatial relations among the deformable parts as hierarchical composition and spatial parsing trees. Empirically our 17-layer DPN can outperform both CapsNets and STNs significantly on affNIST [30], for instance, by 19.19% and 12.75%, respectively, with better generalization and better tolerance to affine transformations.

1 Introduction

Very recently Sabour *et al.* [30] proposed a new network architecture called CapsNet and a dynamic routing training algorithm which connects the capsules [17], a new type of neurons that output vectors rather than scalars in conventional neurons, in two adjacent layers and groups similar features in higher layers. Later on Hinton *et al.* [16] proposed another EM-based routing-by-agreement algorithm for training CapsNet. In contrast to conventional convolutional neural networks (CNNs) that totally ignore the spatial relations between the filters, the intuition behind CapsNet is to achieve "viewpoint invariance" in recognizing objects for better generalization which is inspired by inverse graphics [15]. Technically, CapsNet not only predicts classes but also encodes extra information such as geometry of objects, leading to richer representation than CNNs. For instance, in [16] 4×4 pose matrices are estimated to capture the spatial relations between the detected parts and a whole. Empirically unlike CNNs the performance of CapsNet on real and more complex data has not been verified yet, partially due to the high computation that prevents it from being applicable widely.

In fact exploring such invariant representations for object recognition has a long history in the literature of neural science as well as computer vision. For instance, in [19] Isik *et al.* observed that object recognition in the human visual system is developed in stages with invariance to smaller transformations arising before invariance to larger transformations, which supports the design of feed-forward hierarchical models of invariant object recognition. In computer vision part-based representation (*e.g.* [8]) is one of the most popular invariant object representations. In general part-based models consider an object as a graph where each node represents an object part and each edge represents the (spatial) relation between the parts. Conceptually part-based representation is view-invariant in 3D and pose-invariant (*i.e.* translation, rotation, and scale) in 2D. Although the complexity of part-based models in inference on general graphs could be very high [1], for tree structures such as star graphs this complexity can be linear to the number of parts [7].

Particularly Deformable Part Models (DPMs) [5] have achieved big success in object detection where, in general, we are interested in locating objects using bounding boxes. Based on the pictorial models

Preprint. Work in progress.

[6], *i.e.* star graphs, DPMs decompose an object into a collection of smaller parts, then detect these parts as well as modeling the geometric relations between the parts and the potential object center that are taken as latent variables. It has been demonstrated in the literature that DPMs are more robust to pose variations in objects than conventional detection methods such as template matching (*e.g.* 2D convolution). Recently in [11] DPMs is reinterpreted based on the operators in CNNs.

As discussed above, both pose matrix and part-based representation are capable of capturing spatial relations among the parts in objects, but part-based representation (*e.g.* DPMs) seems more suitable to be incorporated with conventional deep models. *So can we design a deep network based on part-based representation to learn pose-invariant object features as well as being trained efficiently?*

Contributions: In this paper we propose novel *Deformable Part Networks* (DPNs) to efficiently learn pose-invariant representation by estimating object poses in inference.

As a theoretical grounding of DPNs, we first propose a new challenging optimization problem in Sec. 2, namely *Localized Deformable Part Models* (LDPMs), to learn these deformable parts as well as searching for the *best* pose of an object within multiple localized windows. The intuition of introducing localization into DPMs is that the deformation penalties of the parts are essentially dependent on the sizes of windows, and so is pose estimation. Meanwhile, localization enlarges the pose-searching space, leading to better capability of DPMs in modeling.

The huge parameter space in solving LDPMs, however, brings significant computational challenges as well. Therefore, we propose DPNs as an efficient solver for LDPMs in Sec. 3. To regularize the parameter space, we propose using deformable part composition and spatial parsing trees that allow us to perform brute-force pose estimation in inference over all pixels as well as predefined windows. We also propose a new network operator, namely *Deformable Maxout* (DM), to learn the deformation penalties as well as doing inference with the same complexity as 2D convolution. With such help our DPNs can be trained efficiently using stochastic gradient descent (SGD).

To demonstrate the effectiveness of DPNs, we review some related work in Sec. 4 and conduct comprehensive experiments on MNIST [24], affNIST [30] and CIFAR-100 [22] in Sec. 5. We visualize the learned deformable part composition, spatial parsing trees, and feature distributions. Compared with some state-of-the-art networks, *i.e.* VGG16 [31], ResNet32 [14], Spatial Transformer Networks (STNs) [20], Deformable Convolutional Networks (DCNs) [2] and CapsNets, our DPNs can achieve better accuracy with better generalization to the number of training samples and better tolerance to affine transformations.

To summarize, the main contributions of the paper are:

- *C1.* We propose novel Localized Deformable Part Models (LDPMs) that aims to learn the deformable parts as well as detecting the best object poses for recognition.
- *C2.* We further propose novel Deformable Part Networks (DPNs) as an efficient solver for (regularized) LDPMs to learn pose-invariant object representations hierarchically.
- *C3.* We demonstrate the superiority of DPNs over the state-of-the-art networks.

2 Localized Deformable Part Models

Deformable Part Models (DPMs) [5]: Given a window x which is associated with the root filter, a DPM tries to learn a spatial configuration z between the window center and part filters as well as a linear classifier $\mathbf{w}$ so that the following latent support vector machine (SVM) problem is optimization:

$$\min_{\mathbf{w}\in\mathcal{W}} \left\{ \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \ell\left(y, \max_{z\in\mathcal{Z}} \mathbf{w}^T\phi(x,z)\right) \right\}, \tag{1}$$

where *w.l.o.g.* y denotes a binary class label for x, $\mathcal{W}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}$ denote the corresponding feasible spaces[1], ℓ denotes a loss function such as hinge loss, $\phi(x,z)$ denotes a structural feature vector involving appearance features of the (root and part) filters and deformation penalties of the parts (*e.g.* the distance between the detected and learned locations of a part *w.r.t.* the root filter), and $(\cdot)^T$ denotes the matrix transpose operator. Note that DPMs can be applied to image recognition as well by taking each image as a window.

[1]For simplicity of the expressions, without explicit mentioning in this paper we assume that such feasible spaces satisfy the constraints on the variables such as regularization.

Pose Awareness: Ideally a part-based representation is pose-invariant. Empirically, however, this nice property for recognition seems impossible to be achieved due to the visual ambiguity of parts. With the help of deformable parts, DPMs are more robust to such variance in appearance by taking spatial structures of the parts into account, *i.e.* estimating poses of objects.

The ability of pose estimation in DPMs, however, is limited. First of all, the predefined and fixed scales of the root filters on image pyramid imply that DPMs can work well in the scenarios where at the coarse level the object poses need to be close to those defined in the root filters. Secondly the part deformations have no dependency on the object scales, making the pose estimation sensitive to the filter responses outside the object.

In fact, pose estimation heavily depends on *object scales*. Imagining two same faces but with different scales, the part detectors for eyes, nose, mouth, *etc.* as well as their deformations in the faces may be represented differently. For instance, the eye detector for the larger face may be visually bigger (*i.e.* covering more pixels) than that for the smaller face in order to detect the same "eye" semantically. Therefore, to improve pose estimation in DPMs, we propose using *localization* to search for the best object scale in a window, *i.e.* object proposals (*e.g.* [39]), a widely used technique in object detection such as RCNN [10]. As illustrated in Fig. 1, rather than feeding the window of dog into a DPM, we extract proposals from the window and then feed them into a DPM for recognition. Then our LDPMs can capture the best pose with highest detection score among all the proposals.

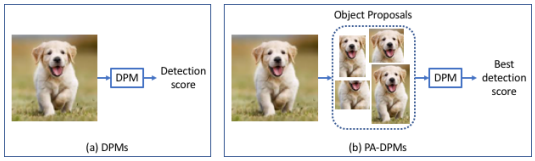

Figure 1: Illustration of comparison between DPMs and our LDPMs, *i.e.* without *vs.* with localization.

Formulation: Similar to Eq. 1, we propose another latent SVM problem for LDPMs as follows:

$$\min_{\mathbf{w}\in\mathcal{W}} \left\{ \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \ell\left(y, \max_{(h,z)\in\mathcal{H}\times\mathcal{Z}} \left\{\mathbf{w}^T\phi(x,h,z)\right\}\right)\right\}, \qquad (2)$$

where $h \in \mathcal{H}$ denotes a proposal within a window x, and $\phi(x,h,z)$ denotes the structural feature vector conditioned on latent variables h and z. Note that variable h accounts for the *multi-scale* localization.

In contrast to DPMs where the inference is conducted in the 2D image space, our LDPMs perform the inference in a 4D space consisting of different proposals as its points. The higher dimensionality brings not only new challenges into learning, but also the flexibility in modeling that may lead us to some solutions with better accuracy and convergence.

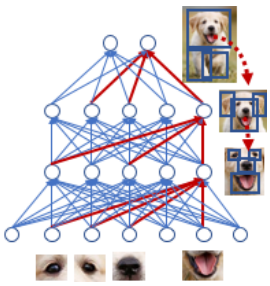
Figure 2: Illustration of deformable part composition.

3 Deformable Part Networks

3.1 Mathematical Modeling

Deformable Part Composition & Spatial Parsing Tree as Regularization: In general the search space for pose estimation can be as huge as $O(N^P)$ where N is the image size and P is the number of parts. In order to explore such huge space efficiently, we propose using tree-structural composition to model the semantic dependency between the parts explicitly. As shown in [27] deep models can approximate complex functions (*e.g.* the decision function $\mathbf{w}^T\phi(x,h,z)$ in Eq. 2) more efficiently and compactly than shallow models. Fig. 2 illustrates an example of our deformable part composition, where each node represents a detector from object classes down to the parts and the red edges denote the dependency between the fired detectors. To facilitate the learning we share all the parts among the classes as suggested in [35] that may lead to better model complexity as well. The responses of fired detectors at a lower layer are passed to the detectors at a higher layer, together with their supporting regions in the image that form spatial parsing trees for estimating poses. These parsing trees actually follow the methodology in DPMs to control the appearance variances of parts based on the semantic hierarchy. Note that each detector can have multiple fires with different windows in image domain.

We would like to learn a model that can learn the deformable part composition as well as inferring the spatial parsing tree in each image as pose estimation to generate pose-invariant representations.

Key Notations: We denote $\{(x, y)\} \subseteq \mathcal{X} \times \mathcal{Y}$ as the training data with image x and its label y, $i \in x$ as the i-th pixel in x, and $h(i) \in \mathcal{H}$ as a window centered at i that consists of a collection of pixels. We also predefine an N-layer deformable part composition where there exist $d_n (n \in [N])$ nodes in the n-th layer. Further we denote $\psi(x, h(i), n) \in \mathbb{R}^{d_n}$ as the scoring vector at pixel i within window $h(i)$ in image x using the detectors in the n-th layer, $\psi(x, i, n) \in \mathbb{R}^{d_n}$ accordingly for window $h(i) = \{i\}$, and $\alpha(h, j, n), \beta(h, j, n) \in \mathbb{R}^{d_n}$ as the deformation penalty parameters at pixel $j \in h$ within window h for the detectors in the n-th layer. We also denote matrix $\mathbf{W}_n \in \mathbb{R}^{d_{n-1} \times d_n}, \forall n$ as the semantic composition weights between the $(n-1)$-th and n-th layers, ℓ as the loss function for recognition, σ as the activation function such as ReLU [28] for firing detectors, and all the $\max$ operators in the following sections are entry-wise.

Formulation for Brute-Force Pose Estimation: Based on deformable part composition and spatial parsing trees, we manage to convert the pose estimation problem at inference time in Eq. 2 to localizing a proper window per pixel where an LDPM is conducted for pose estimation.

To do so, we first decompose the feature vector $\phi(x, h(i), z) = \{\psi(x, h(i), n)\}_{1 \leq n \leq N}$ as a collection of scoring vectors using the detectors per layer. Then we explicitly define $\psi(x, h(i), n)$ as follows:

$$\psi(x, h(i), n) = \max_{j \in h(i)} \Big\{ \alpha(h, j, n) \otimes \psi(x, j, n) \oplus \beta(h, j, n) \Big\}, \forall x, \forall i, \forall n, \tag{3}$$

where $\otimes, \oplus$ denote the entry-wise product and summation between two vectors, respectively. Different from previous works such as [5; 29] where deformation penalties are parameterized based on explicit deformation features such as distances, we directly model these penalties using latent learnable functions α, β that take window size, pixel location, and detectors as input and output vectors to penalize the deformation *w.r.t.* the window center.

Now based on Eq. 2 and Eq. 3 we propose a new specific LDPM formulation as follows[2]:

$$\begin{aligned} &\min_{\{\mathbf{W}_n\} \subseteq \mathcal{W}, \alpha \in \mathcal{A}, \beta \in \mathcal{B}} \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \ell\Big(y, \max_i \psi(x, i, N)\Big), \\ &\text{s.t. } \psi(x, i, n) = \sigma\left(\max_{h(i) \in \mathcal{H}} \mathbf{W}_n^T \psi(x, h(i), n-1) \right), \forall x, \forall i, \forall n, \end{aligned} \tag{4}$$

where $\psi(x, i, 0) \in \mathbb{R}^{d_0}, \forall x, \forall i$ denotes the raw image feature vector at pixel i in image x. As discussed in Sec. 2, the detectors for the same semantic concept (*e.g.* eye) may be dependent on window sizes. To account for the semantic consistency within different windows, here we intentionally employ multiple instance learning (MIL), parameterized by $\mathbf{W}_n, \forall n$, to select the best window.

By simultaneously considering all the pixels and window sizes defined in $\mathcal{H}$ for solving the specific LDPM problem in Eq. 4, we indeed search for the best poses of objects in a brute-force manner at inference time in a 4D window space. Together with variables $\mathbf{W}_n, \forall n$ and latent functions α, β that conduct the part detection and deformation in multi-scale scenarios, our approach can manage to estimate the best poses and thus generate pose-invariant representations for object recognition.

3.2 DPNs: an Efficient LDPM Solver

The outputs of α, β in Eq. 3 can be considered as two sets of filters, similar to those in 2D convolution. To do the inference in Eq. 3, we propose a new network operator, namely *Deformable Maxout* (DM), by applying $\otimes, \oplus$ sequentially over different channels, which has the same computational complexity as 2D convolution.

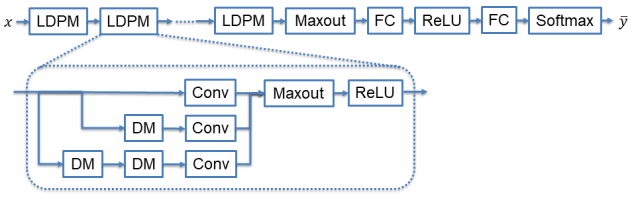

Figure 3: Illustration of DPN architecture.

Considering the computational efficiency, we refer to the inception in GoogLeNet [32] and predefine the feasible window set $\mathcal{H}$ for measuring part deformation as a collection of 1×1, 3×3 and 5×5. With the increase of network depth, larger windows (*e.g.* 7×7) will be covered as the receptive fields

[2]For simplicity in formulation we keep the image sizes unchanged during both learning and inference.

of neurons, and due to the characteristic of deformation any arbitrary window within the receptive field can be potentially localized as a part. This implicitly favors the brute-force pose estimation.

Architecture: We illustrate our DPNs in Fig. 3 as an efficient solver for LDPMs defined in Eq. 4. In feedforward procedure, each LDPM unit computes the detection scores as well as conducting pose estimation where the filters in convolution layers are *shared*, and the maxout layer [12] is over $\mathcal{H}$. The maxout layer outsize LDPM units is over the image domain. The convolution layers and DM layers are responsible for learning $\mathbf{W}_n, \forall n$ and α, β in Eq. 4, respectively, that are updated in back-propagation. Since the DM operator is differentiable with the same complexity as 2D convolution, our DPNs can be trained as efficiently as CNNs.

Implementation: Without fine-tuning we set both filter size in all the convolution layers and window size in DM layers to 3×3. Similar to [33], we use two sequential DM operations to approximate the output of DM with 5×5 windows. We initialize the network based on [13], and optimize it using ADAM [21]. Batch normalization [18] can be employed as well. Particularly as demonstration, in our experiments we set the numbers of kernels in LDPM units as well as the fully connected (FC) layer to $32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512 \rightarrow 1024 \rightarrow 10$, respectively. This is equivalent to a 17-layer CNN by taking the longest path in the network from input to output and counting DM, Conv, and FC only along the path. Considering the datasets, we apply image downsampling twice by 2 after the LDPM units with 64 and 256 filters, respectively, for better computational efficiency with little impact on pose estimation as the DM layers will compensate for the inaccuracy in localization.

Empirically we find that both the depth of the networks and the width of LDPM units have impact on the performance. In general the deeper and wider the network, the better the accuracy with slower running speed. For instance, on affNIST with the increase of the width from 3 (as shown currently in Fig. 3) to 7 that capture larger windows (*i.e.* from 7×7 to 13×13, step by 2), the accuracy is improved by about 2%. If the width is reduced to 1 (*i.e.* only the first branch in LDPM units left), the accuracy drops by about 8%. Relatively the depth of the networks are more important than the width of LDPM units in order to achieve good performance. This is reasonable as larger windows can be captured by the neurons at the higher layers in the networks.

4 Related Work

Besides DPMs and CapsNets, we summarize some other related work as follows.

Hierarchical Deformable Part Models (HDPMs): Felzenszwalb *et al.* [4] proposed a cascade detection algorithm based on partial hypothesis pruning for accelerating DPMs that can be defined by a grammar formalism. Zhu *et al.* [40] proposed a mixture of three-layer latent hierarchical tree models for object detection and learned it using an incremental concave-convex procedure (iCCCP). Ghiasi and Fowlkes [9] proposed a two-layer HDPM for modeling occlusion in faces that achieved state-of-the-art performance on benchmarks for occluded face localization. Tian *et al.* [34] proposed a three-layer hierarchical spatial model that can capture an exponential number of poses with a compact mixture representation on each part using exact inference. Wu *et al.* [37] proposed a And-Or car detection model to embed a grammar for representing large structural and appearance variations in a reconfigurable hierarchy that is trained using Weak-Label Structural SVM.

In contrast to these works, our LDPMs can involve much deeper deformable part hierarchy and learn these parts automatically rather than manual design based on certain prior knowledge. Further we propose using DPNs as our efficient solver for LDPMs that can naturally learn the semantic and spatial relations among the parts in the hierarchy.

Pose-Aware Networks: In terms of applications, poses are usually considered and encoded into networks for the recognition of specific object classes such as faces [26] or human [3; 36; 23]. For instance, Wei *et al.* [36] proposed Convolutional Pose Machines (CPMs) that provide a sequential prediction framework for learning rich implicit spatial models for human pose estimation. The body parts are well defined visually with clear spatial supports but without semantically composite relations. Differently our DPNs are developed for estimating the poses of general objects based on deformable part composition and spatial parsing trees.

In terms of functionality, some pose-aware network operations or modules as plug-in are proposed for existing networks. For instance, dilated convolution [38] supports exponential expansion of the receptive field (*i.e.* window) without loss of resolution or coverage and thus can help networks capture

multi-scale information. Deformable Convolutional Networks (DCNs) [2] proposed a more flexible convolutional operator that introduces pixel-level deformation, estimated by another network, into 2D convolution. Spatial Transformer Networks (STNs) [20] learn pose-invariant representations by sequential applications of a localization network, a parameterized grid generator and a sampler.

In contrast, we propose a new DM operator that can be used to learn deformation penalties for parts as well as conducting the inference for DPMs within fixed windows. Based on DM we propose LDPM units as network modules in DPNs to solve the optimization problem in LDPMs by estimating object poses as well as predicting class labels, leading to pose-invariant representations for recognition.

DPN *vs.* GoogLeNet & ResNet [14]: In particular, in terms of architecture our LDPM units in DPN are related to the inception module in GoogLeNet and ResNet. Both LDPM and inception are able to capture multi-scale information. Differently the receptive fields in inception are fixed by the sizes of convolutional filters, while LDPM manages to locate arbitrary windows with best part detection scores within each receptive field defined by DM. Compared with ResNet, LDPM can have skip connections as well by removing the convolutional layers. Differently LDPM takes entry-wise maximum over different receptive fields rather than summation, leading to feature selection in LDPM.

5 Experiments

We test and compare our DPN with some state-of-the-art networks with similar model complexity to ours, *i.e.* VGG16[3], ResNet32[4], STN[5], DCN[6], and CapsNet[7]. We use three benchmarks, namely MNIST, affNIST, and CIFAR-100. Particularly affNIST is created for testing the tolerance of an algorithm to affine transformation (*i.e.* translation, rotation, shearing and scaling). On MNIST we follow standard procedure to train the networks using the 60K training/validation samples and test them using the 10K test samples. On affNIST we follow [30] to create a new training set of 60K samples using original MNIST training/validation samples with *random translation only*, and train all the networks using it, then test them using the 10K test samples in affNIST which involve *random affine transformations*. To facilitate the training, we resize the images to 28×28, same as MNIST. On CIFAR-100 we utilize the pre-processing code[8] for network training using the 50K training samples, and test the networks using the 10K test samples.

Better Performance: We summarize the accuracy comparison in Table 1. As we see on all the datasets our DPN significantly outperforms the three pose-aware networks, *i.e.* STN, DCN, and CapsNet. Compared with well designed CNN based networks, *i.e.* VGG16 and ResNet32, DPN is always comparable. We believe that such observations are mainly the outcomes of learning deformation in DPN that helps capture the part configurations of objects efficiently rather than "memorizing" the training instances.

	Ours (DPN17)	VGG16	ResNet32	STN	DCN	CapsNet
MNIST	99.48	**99.71**	99.23	99.26	99.45	99.66
affNIST	**97.26**	93.12	92.89	84.51	90.32	78.07
CIFAR-100	**70.96**	70.48	68.10	66.18	67.39	-

Table 1: Best accuracy (%) comparison on different datasets, where "-" indicates that we cannot achieve reasonable performance.

Better Generalization: To verify our hypothesis, we conduct a performance analysis over the number of training images per class on affNIST as shown in Fig. 4. Overall the accuracy of all the methods becomes worse when the number decreases. DPN, however, behaves much more robustly. In the extreme case where there are only 3 images per class for training, DPN can achieve **36.35%** that is **17.31%** improvement over the second best. Considering our model complexity that is much higher than ResNet32 and DCN, in such extreme cases DPN should perform worse due to the higher risk of overfitting. Surprisingly,

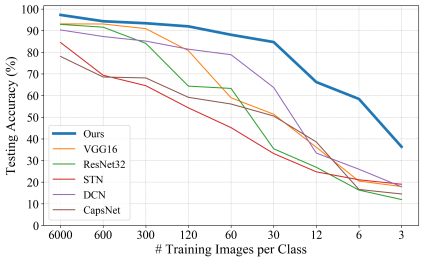

Figure 4: Performance analysis on affNIST.

[3]https://github.com/geifmany/cifar-vgg

[4]https://github.com/tensorflow/models/tree/master/research/resnet

[5]https://github.com/kevinzakka/spatial-transformer-network

[6]https://github.com/felixlaumon/deform-conv

[7]https://github.com/naturomics/CapsNet-Tensorflow

[8]https://github.com/tensorflow/models/blob/master/tutorials/image/cifar10

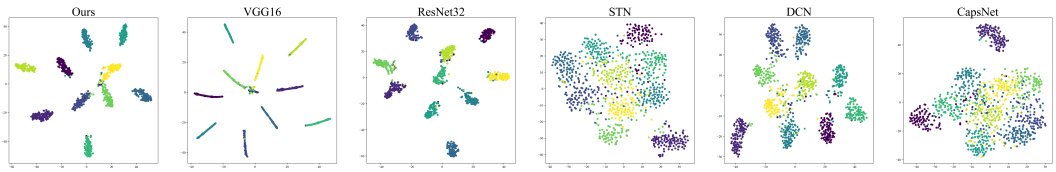

Figure 5: Feature distribution comparison on affNIST test data using t-SNE [25], one color per class.

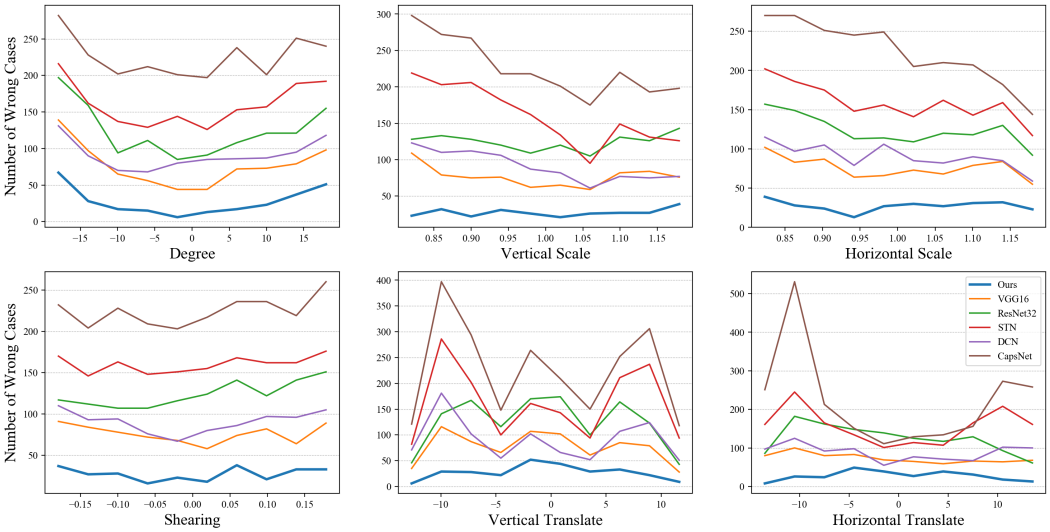

Figure 6: Analysis of failure cases on affNIST test data over different transformations.

however, this is not true on affNIST. Since the key difference between DPN and other networks is that we utilize DPMs to estimate poses, we then conclude that the learned deformable part configurations do help us recognize new digits that are never seen before, leading to better generalization.

Better Tolerance to Affine Transformation: To illustrate the differences of learned features among the networks, we show the feature distribution comparison in Fig. 5 using t-SNE. For each network we extract the features that are fed into the classifier module directly. For instance, for DPN we extract the 1024D features. As we see the our DPN features can form more compact and separable clusters than the other pose-aware networks. To quantify the distributions, we measure the intra-class (*w.r.t.* compactness) and inter-class (*w.r.t.* separability) distances in the 2D space in Fig. 5 and list the comparison in Table 2. Clearly our intra-class distance is much smaller than the others while inter-class distance is larger, leading to more discriminative (and probably pose-invariant) distributions for recognition.

	Ours	VGG16	ResNet32	STN	DCN	CapsNet
Intra-cls. dis.	**4.79**	6.20	6.57	7.31	6.19	8.20
Inter-cls. dis.	**24.78**	21.62	23.23	17.25	19.09	17.39
Intra/Inter	**0.19**	0.28	0.28	0.42	0.32	0.47

Table 2: Comparison on intra-class and inter-class distances in Fig. 5.

As shown in Fig. 6 we analyze the failure cases on affNIST as well. We quantize each parameter for affine transformation into 1 of 10 bins accordingly, and the total numbers of failure cases per network in the 6 subfigures are the same. As we see our DPN is the most robust to all of the transformations, leading to fewest failure cases among all the networks. Among different transformations DPN is more sensitive to rotation (see the top-left subfigure), as more degree for rotation is, more failure cases occur. For the others the distributions appear more or left flat. This is mainly because rotation has larger impact on the learning of parts as well as their spatial configurations, making the pose estimation less reliable.

Deformable Part Visualization: To better understand our DPN, we visualize some learned deformable parts in Fig. 7. On the left, as an example we show the hierarchical decomposition of a digit "8" using learned parts, where the red lines denote the edges with positive weights in the hierarchy. Here all the parts are rescaled to a same size per layer, and we can only show the top two layers because it becomes difficult to clearly visualize smaller parts in the lower layers with few pixels. On the right, we visualize the spatial configures of some parts in the top two layers of the spatial parsing

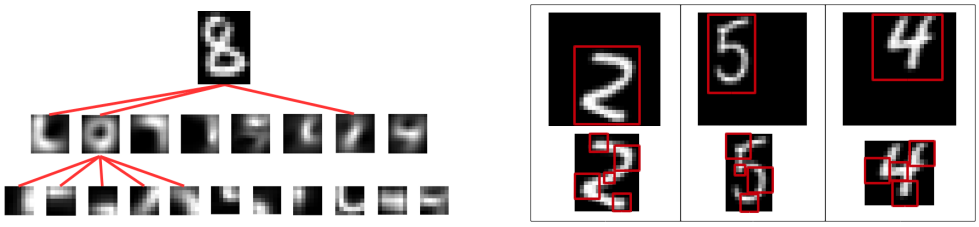

Figure 7: Deformable part visualization on affNIST (**left**) semantically and (**right**) spatially.

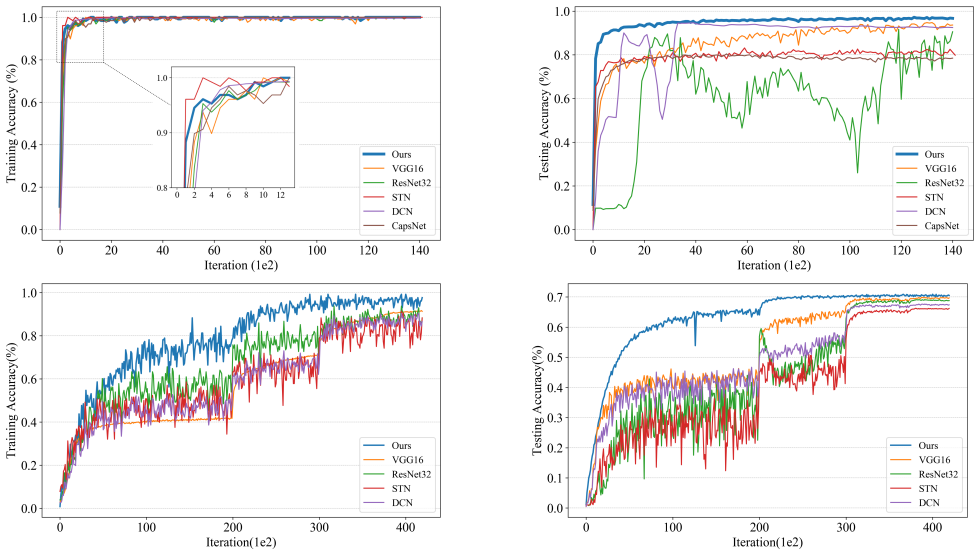

Figure 8: Illustration of training and testing accuracy on (**top**) affNIST and (**bottom**) CIFAR-100.

trees for digit "2", "5", and "4" respectively. As we see the spatial configurations of deformable parts can indeed locate the digits nicely with different windows, within which the scales of the detected parts may vary significantly. These observations come from the characteristics of LDPM units that can search for the best object poses over all possible pixels and predefined windows.

Training & Testing Behavior: To validate our results, we show the training and testing accuracy behavior of each network on affNIST (full training dataset) and CIFAR-100, where each iteration contains 128 mini-batches. On affNIST we set learning rate to 0.001. On CIFAR-100, we start with learning rate 0.1, divide it by 10 after 20K and 30K iterations, and terminate training after 42K iterations. As we see all the networks are well trained with convergence. In the testing stage our DPN converges much faster than the others, leading to big gaps in the first few iterations. Similar observations can be made in training as well. From this perspective, we can also demonstrate that DPN has better generalization than the other networks.

We record the training time of each network from tensorboard and list them in Table 3. Since the complexity of DM is the same as 2D convolution, our DPNs should be able to be trained as efficiently as other CNN based networks. Indeed we can verify this by comparing DPN17 with VGG16 and ResNet32. In addition DPN17 can be trained faster than DCN and CapsNet.

	Ours	VGG16	ResNet32	STN	DCN	CapsNet
affNIST	37.8	14.7	47.3	12.7	83.0	65.0
CIFAR-100	102	52.0	138.6	63.0	152.0	-

Table 3: Comparison on training time (s) per epoch.

6 Conclusion

In this paper we propose novel Deformable Part Networks (DPNs) for 2D object recognition that can be interpreted as detecting objects within the networks for learning pose-invariant features. By comparing with some state-of-the-art networks, we demonstrate that empirically our DPNs can achieve better performance with better generalization and better tolerance to affine transformations, thanks to the learning of spatial configurations of deformable parts for pose estimation.

References

[1] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR*, volume 1, pages 10–17, 2005.

[2] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *CVPR*, pages 764–773, 2017.

[3] K. Duan, D. Batra, and D. Crandall. Human pose estimation through composite multi-layer models. *Signal Processing*, 110:15–26, May 2015.

[4] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, pages 2241–2248, 2010.

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

[6] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. *Theory Of Computing*, 8:415–428, 2012.

[8] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973.

[9] G. Ghiasi and C. C. Fowlkes. Occlusion coherence: Localizing occluded faces with a hierarchical deformable part model. In *CVPR*, pages 2385–2392, 2014.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[11] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *CVPR*, pages 437–446, 2015.

[12] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, pages III–1319, 2013.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[15] G. Hinton. Taking inverse graphics seriously. https://www.cs.toronto.edu/ hinton/csc2535/notes/lec6b.pdf.

[16] G. Hinton, N. Frosst, and S. Sabour. Matrix capsules with em routing. In *ICLR*, 2018.

[17] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[19] L. Isik, E. M. Meyers, J. Z. Leibo, and T. Poggio. The dynamics of invariant object recognition in the human visual system. *Journal of neurophysiology*, 111(1):91–102, 2013.

[20] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[23] V. Kumar, A. Namboodiri, M. Paluri, and C. Jawahar. Pose-aware person recognition. In *CVPR*, pages 6223–6232, 2017.

[24] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[25] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.

[26] I. Masi, S. Rawls, G. Medioni, and P. Natarajan. Pose-aware face recognition in the wild. In *CVPR*, pages 4838–4846, 2016.

[27] H. N. Mhaskar and T. Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

[28] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[29] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In *CVPR*, pages 2403–2412, 2015.

[30] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *NIPS*, pages 3859–3869, 2017.

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

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, J.-H. Rick Chang, et al. Going deeper with convolutions. In *CVPR*, 2015.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.

[34] Y. Tian, C. L. Zitnick, and S. G. Narasimhan. Exploring the spatial hierarchy of mixture models for human pose estimation. In *ECCV*, pages 256–269. Springer, 2012.

[35] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, 2004.

[36] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, pages 4724–4732, 2016.
[37] T. Wu, B. Li, and S.-C. Zhu. Learning and-or model to represent context and occlusion for car detection and viewpoint estimation. *PAMI*, 38(9):1829–1843, 2016.
[38] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
[39] Z. Zhang and P. H. Torr. Object proposal generation using two-stage cascade svms. *TPAMI*, 38(1):102–115, 2016.
[40] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, pages 1062–1069. IEEE, 2010.