Learning to Compose Hypercolumns for Visual Correspondence

Juhong Min, Jongmin Lee, Jean Ponce, Minsu Cho

Learning to Compose Hypercolumns for Visual Correspondence

Juhong Min[1,2] Jongmin Lee[1,2] Jean Ponce[3,4] Minsu Cho[1,2]

[1]POSTECH* [2]NPRC† [3]Inria [4]ENS‡
http://cvlab.postech.ac.kr/research/DHPF/

Abstract. Feature representation plays a crucial role in visual correspondence, and recent methods for image matching resort to deeply stacked convolutional layers. These models, however, are both monolithic and static in the sense that they typically use a specific level of features, *e.g.*, the output of the last layer, and adhere to it regardless of the images to match. In this work, we introduce a novel approach to visual correspondence that dynamically composes effective features by leveraging relevant layers conditioned on the images to match. Inspired by both multi-layer feature composition in object detection and adaptive inference architectures in classification, the proposed method, dubbed *Dynamic Hyperpixel Flow*, learns to compose hypercolumn features on the fly by selecting a small number of relevant layers from a deep convolutional neural network. We demonstrate the effectiveness on the task of semantic correspondence, *i.e.*, establishing correspondences between images depicting different instances of the same object or scene category. Experiments on standard benchmarks show that the proposed method greatly improves matching performance over the state of the art in an adaptive and efficient manner.

Keywords: visual correspondence, multi-layer features, dynamic feature composition

1 Introduction

Visual correspondence is at the heart of image understanding with numerous applications such as object recognition, image retrieval, and 3D reconstruction [12]. With recent advances in neural networks [19, 20, 22, 32, 50], there has been a significant progress in learning robust feature representation for establishing correspondences between images under illumination and viewpoint changes. Currently, the de facto standard is to use as feature representation the output of deeply stacked convolutional layers in a trainable architecture. Unlike in object classification and detection, however, such learned features have often achieved

*Pohang University of Science and Technology, Pohang, Korea
†The Neural Processing Research Center, Seoul, Korea
‡École normale supérieure, CNRS, PSL Research University, 75005 Paris, France

only modest performance gains over hand-crafted ones [6, 40] in the task of visual correspondence [48]. In particular, correspondence between images under large intra-class variations still remains an extremely challenging problem [5, 10, 17, 25, 26, 28, 29, 30, 33, 39, 42, 44, 45, 46, 47, 49, 53, 57] while modern neural networks are known to excel at classification [19, 22]. What do we miss in using deep neural features for correspondence?

Most current approaches for correspondence build on monolithic and static feature representations in the sense that they use a specific feature layer, *e.g.*, the last convolutional layer, and adhere to it regardless of the images to match. Correspondence, however, is all about precise localization of corresponding positions, which requires visual features at different levels, from local patterns to semantics and context; in order to disambiguate a match on similar patterns, it is necessary to analyze finer details and larger context in the image. Furthermore, relevant feature levels may vary with the images to match; the more we already know about images, the better we can decide which levels to use. In this aspect, conventional feature representations have fundamental limitations.

In this work, we introduce a novel approach to visual correspondence that dynamically composes effective features by leveraging relevant layers conditioned on the images to match. Inspired by both multi-layer feature composition, *i.e.*, hypercolumn, in object detection [18, 31, 35, 38] and adaptive inference architectures in classification [11, 51, 54], we combine the best of both worlds for visual correspondence. The proposed method learns to compose hypercolumn features on the fly by selecting a small number of relevant layers in a deep convolutional neural network. At inference time, this dynamic architecture greatly improves matching performance in an adaptive and efficient manner. We demonstrate the effectiveness of the proposed method on several benchmarks for semantic correspondence, *i.e.*, establishing visual correspondences between images depicting different instances of the same object or scene categories, where due to large variations it may be crucial to use features at different levels.

2 Related work

Feature representation for semantic correspondence. Early approaches [3, 4, 15, 27, 37, 52, 55] tackle the problem of visual correspondence using hand-crafted descriptors such as HOG [6] and SIFT [40]. Since these lack high-level image semantics, the corresponding methods have difficulties with significant changes in background, view point, deformations, and instance-specific patterns. The advent of convolutional neural networks (CNN) [19, 32] has led to a paradigm shift from this hand-crafted representations to deep features and boosted performance in visual correspondence [10, 44, 57]. Most approaches [5, 17, 29, 47] learn to predict correlation scores between local regions in an input image pair, and some recent methods [25, 26, 28, 45, 46, 49] cast this task as an image alignment problem in which a model learns to regress global geometric transformation parameters. All typically adopt a CNN pretrained on image classification as their backbone, and make predictions based on features from its final convolutional layer. While some

methods [39, 56] have demonstrated the advantage of using different CNN layers in capturing low-level to high-level patterns, leveraging multiple layers of deeply stacked layers has remained largely unexplored in correspondence problems.

Multi-layer neural features. To capture different levels of information distributed over all intermediate layers, Hariharan *et al.* propose the hypercolumn [18], a vector of multiple intermediate convolutional activations lying above a pixel for fine-grained localization. Attempts at integrating multi-level neural features have addressed object detection and segmentation [31, 35, 38]. In the area of visual correspondence, only a few methods [42, 44, 53] attempt to use multi-layer features. Unlike ours, however, these models use static features extracted from CNN layers that are chosen manually [44, 53] or by greedy search [42]. While the use of hypercolumn features on the task of semantic visual correspondence has recently been explored by Min *et al.* [42], the method predefines hypercolumn layers by a greedy selection procedure, *i.e.*, beam search, using a validation dataset. In this work, we clearly demonstrate the benefit of a dynamic and learnable architecture both in strongly-supervised and weakly-supervised regimes and also outperform the work of [42] with a significant margin.

Dynamic neural architectures. Recently, dynamic neural architectures have been explored in different domains. In visual question answering, neural module networks [1, 2] compose different answering networks conditioned on an input sentence. In image classification, adaptive inference networks [11, 51, 54] learn to decide whether to execute or bypass intermediate layers given an input image. Dynamic channel pruning methods [13, 21] skip unimportant channels at run-time to accelerate inference. All these methods reveal the benefit of dynamic neural architectures in terms of either accuracy or speed, or both. To the best of our knowledge, our work is the first that explores a dynamic neural architecture for visual correspondence.

Our main contribution is threefold: (1) We introduce a novel dynamic feature composition approach to visual correspondence that composes features on the fly by selecting relevant layers conditioned on images to match. (2) We propose a trainable layer selection architecture for hypercolumn composition using Gumbel-softmax feature gating. (3) The proposed method outperforms recent state-of-the-art methods on standard benchmarks of semantic correspondence in terms of both accuracy and speed.

3 Dynamic hyperpixel flow

Given two input images to match, a pretrained convolutional network extracts a series of intermediate feature blocks for each image. The architecture we propose in this section, *dynamic hyperpixel flow*, learns to select a small number of layers (feature blocks) on the fly and composes effective features for reliable matching of the images. Figure 1 illustrates the overall architecture. In this section, we describe the proposed method in four steps: (i) multi-layer feature extraction, (ii) dynamic layer gating, (iii) correlation computation and matching, and (iv) training objective.

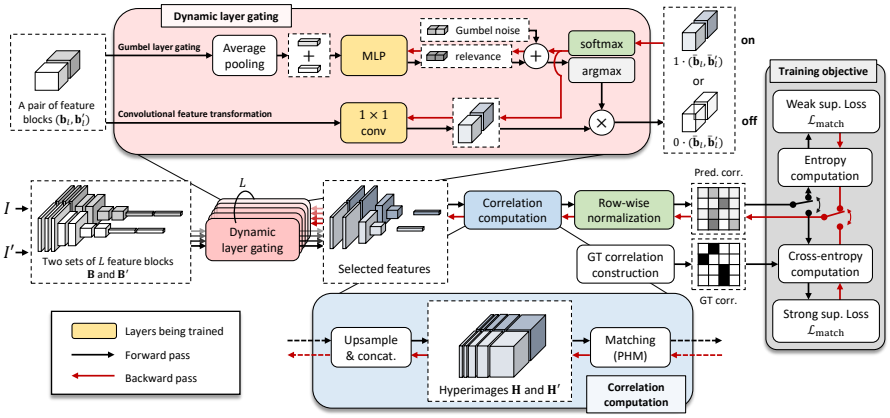


Fig. 1: The overall architecture of Dynamic Hyperpixel Flow (DHPF).

3.1 Multi-layer feature extraction

We adopt as a feature extractor a convolutional neural network pretrained on a large-scale classification dataset, *e.g.*, ImageNet [7], which is commonly used in most related methods [5, 17, 28, 30, 33, 42, 45, 46, 47, 49, 23]. Following the work on hypercolumns [18], however, we view the layers of the convolutional network as a non-linear counterpart of image pyramids and extract a series of multiple features along intermediate layers [42].

Let us assume the backbone network contains L feature extracting layers. Given two images I and I', source and target, the network generates two sets of L intermediate feature blocks. We denote the two sets of feature blocks by $\mathbf{B} = \{\mathbf{b}_l\}_{l=0}^{L-1}$ and $\mathbf{B}' = \{\mathbf{b}'_l\}_{l=0}^{L-1}$, respectively, and call the earliest blocks, $\mathbf{b}_0$ and $\mathbf{b}'_0$, *base* feature blocks. As in Fig. 1, each pair of source and target feature blocks at layer l is passed to the l-th layer gating module as explained next.

3.2 Dynamic layer gating

Given L feature block pairs $\{(\mathbf{b}_l, \mathbf{b}'_l)\}_{l=0}^{L-1}$, L layer gating modules learn to select relevant feature block pairs and transform them for establishing robust correspondences. As shown in the top of Fig. 1, the module has two branches, one for layer gating and the other for feature transformation.

Gumbel layer gating. The first branch of the l-th layer gating module takes the l-th pair of feature blocks $(\mathbf{b}_l, \mathbf{b}'_l)$ as an input and performs global average pooling on two feature blocks to capture their channel-wise statistics. Two average pooled features of size $1 \times 1 \times c_l$ from $\mathbf{b}_l$ and $\mathbf{b}'_l$ are then added together to form a vector of size c_l. A multi-layer perceptron (MLP) composed of two fully-connected layers with ReLU non-linearity takes the vector and predicts a relevance vector $\mathbf{r}_l$ of size 2 for gating, whose entries indicate the scores for selecting or skipping

('on' or 'off') the l-th layer, respectively. We can simply obtain a gating decision using argmax over the entries, but this naïve gating precludes backpropagation since argmax is not differentiable.

To make the layer gating trainable and effective, we adopt the Gumbel-max trick [14] and its continuous relaxation [24, 41]. Let $\mathbf{z}$ be a sequence of i.i.d. Gumbel random noise and let Y be a discrete random variable with K-class categorical distribution $\mathbf{u}$, *i.e.*, $p(Y = y) \propto u_y$ and $y \in \{0, ..., K-1\}$. Using the Gumbel-max trick [14], we can reparamaterize sampling Y to $y = \arg\max_{k \in \{0,...,K-1\}}(\log u_k + z_k)$. To approximate the argmax in a differentiable manner, the continuous relaxation [24, 41] of the Gumbel-max trick replaces the argmax operation with a softmax operation. By expressing a discrete random sample y as a one-hot vector $\mathbf{y}$, a sample from the Gumbel-softmax can be represented by $\hat{\mathbf{y}} = \text{softmax}((\log \mathbf{u} + \mathbf{z})/\tau)$, where τ denotes the temperature of the softmax. In our context, the discrete random variable obeys a Bernoulli distribution, *i.e.*, $y \in \{0, 1\}$, and the predicted relevance scores represent the log probability distribution for 'on' and 'off', *i.e.*, $\log \mathbf{u} = \mathbf{r}_l$. Our Gumbel-softmax gate thus has a form of

$$\hat{\mathbf{y}}_l = \text{softmax}(\mathbf{r}_l + \mathbf{z}_l), \tag{1}$$

where $\mathbf{z}_l$ is a pair of i.i.d. Gumbel random samples and the softmax temperature τ is set to 1.

Convolutional feature transformation. The second branch of the l-th layer gating module takes the l-th pair of feature blocks $(\mathbf{b}_l, \mathbf{b}'_l)$ as an input and transforms each feature vector over all spatial positions while reducing its dimension by $\frac{1}{\rho}$; we implement it using 1×1 convolutions, *i.e.*, position-wise linear transformations, followed by ReLU non-linearity. This branch is designed to transform the original feature block of size $h_l \times w_l \times c_l$ into a more compact and effective representation of size $h_l \times w_l \times \frac{c_l}{\rho}$ for our training objective. We denote the pair of transformed feature blocks by $(\bar{\mathbf{b}}_l, \bar{\mathbf{b}}'_l)$. Note that if l-th Gumbel gate chooses to skip the layer, then the feature transformation of the layer can be also ignored thus reducing the computational cost.

Forward and backward propagations. During training, we use the *straight-through* version of the Gumbel-softmax estimator [24]: forward passes proceed with discrete samples by argmax whereas backward passes compute gradients of the softmax relaxation of Eq.(1). In the forward pass, the transformed feature pair $(\bar{\mathbf{b}}_l, \bar{\mathbf{b}}'_l)$ is simply multiplied by 1 ('on') or 0 ('off') according to the gate's discrete decision $\mathbf{y}$. While the Gumbel gate always makes discrete decision $\mathbf{y}$ in the forward pass, the continuous relaxation in the backward pass allows gradients to propagate through softmax output $\hat{\mathbf{y}}$, effectively updating both branches, the feature transformation and the relevance estimation, regardless of the gate's decision. Note that this stochastic gate with random noise increases the diversity of samples and is thus crucial in preventing mode collapse in training. At test time, we simply use deterministic gating by argmax without Gumbel noise [24]. As discussed in Sec. 4.2, we found that the proposed hard gating trained with

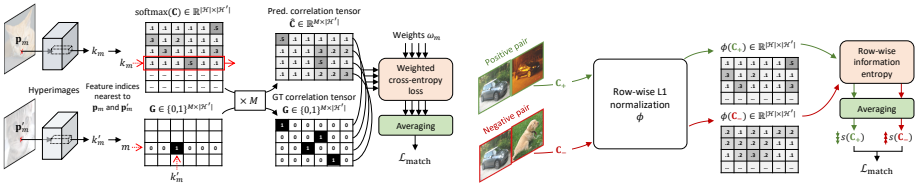
Gumbel softmax is superior to conventional soft gating with sigmoid in terms of both accuracy and speed.

3.3 Correlation computation and matching

The output of gating is a set of selected layer indices, $S = \{s_1, s_2, ..., s_N\}$. We construct a *hyperimage* $\mathbf{H}$ for each image by concatenating transformed feature blocks of the selected layers along channels with upsampling: $\mathbf{H} = \left[\zeta(\bar{\mathbf{b}}_{s_1}), \zeta(\bar{\mathbf{b}}_{s_2}), ..., \zeta(\bar{\mathbf{b}}_{s_N})\right]$, where ζ denotes a function that spatially upsamples the input feature block to the size of $\mathbf{b}_0$, the *base* block. Note that the number of selected layers N is fully determined by the gating modules. If all layers are off, then we use the base feature block by setting $S = \{0\}$. We associate with each spatial position p of the hyperimage the corresponding image coordinates and hyperpixel feature [42]. Let us denote by $\mathbf{x}_p$ the image coordinate of position p, and by $\mathbf{f}_p$ the corresponding feature, i.e., $\mathbf{f}_p = \mathbf{H}(\mathbf{x}_p)$. The hyperpixel at position p in the hyperimage is defined as $\mathbf{h}_p = (\mathbf{x}_p, \mathbf{f}_p)$. Given source and target images, we obtain two sets of hyperpixels, $\mathcal{H}$ and $\mathcal{H}'$. In order to reflect geometric consistency in matching, we adapt probablistic Hough matching (PHM) [4, 17] to hyperpixels, similar to [42]. The key idea of PHM is to re-weight appearance similarity by Hough space voting to enforce geometric consistency. In our context, let $\mathcal{D} = (\mathcal{H}, \mathcal{H}')$ be two sets of hyperpixels, and $m = (\mathbf{h}, \mathbf{h}')$ be a match where $\mathbf{h}$ and $\mathbf{h}'$ are respectively elements of $\mathcal{H}$ and $\mathcal{H}'$. Given a Hough space $\mathcal{X}$ of possible offsets (image transformations) between the two hyperpixels, the confidence for match m, $p(m|\mathcal{D})$, is computed as $p(m|\mathcal{D}) \propto p(m_{\mathrm{a}}) \sum_{\mathbf{x}\in\mathcal{X}} p(m_{\mathrm{g}}|\mathbf{x}) \sum_{m\in\mathcal{H}\times\mathcal{H}'} p(m_{\mathrm{a}}) p(m_{\mathrm{g}}|\mathbf{x})$ where $p(m_{\mathrm{a}})$ represents the confidence for appearance matching and $p(m_{\mathrm{g}}|\mathbf{x})$ is the confidence for geometric matching with an offset $\mathbf{x}$, measuring how close the offset induced by m is to $\mathbf{x}$. By sharing the Hough space $\mathcal{X}$ for all matches, PHM efficiently computes match confidence with good empirical performance [4, 15, 17, 42]. In this work, we compute appearance matching confidence using hyperpixel features by $p(m_{\mathrm{a}}) \propto \mathrm{ReLU}\left(\frac{\mathbf{f}_p \cdot \mathbf{f}'_p}{\|\mathbf{f}_p\| \|\mathbf{f}'_p\|}\right)^2$, where the squaring has the effect of suppressing smaller matching confidences. On the output $|\mathcal{H}| \times |\mathcal{H}'|$ correlation matrix of PHM, we perform soft mutual nearest neighbor filtering [47] to suppress noisy correlation values and denote the filtered matrix by $\mathbf{C}$.

Dense matching and keypoint transfer. From the correlation matrix $\mathbf{C}$, we establish hyperpixel correspondences by assigning to each source hyperpixel $\mathbf{h}_i$ the target hyperpixel $\hat{\mathbf{h}}'_j$ with the highest correlation. Since the spatial resolutions of the hyperimages are the same as those of base feature blocks, which are relatively high in most cases (*e.g.*, 1/4 of input image with ResNet-101 as the backbone), such hyperpixel correspondences produce quasi-dense matches.

Furthermore, given a keypoint $\mathbf{p}_m$ in the source image, we can easily predict its corresponding position $\hat{\mathbf{p}}'_m$ in the target image by transferring the keypoint using its nearest hyperpixel correspondence. In our experiments, we collect all correspondences of neighbor hyperpixels of keypoint $\mathbf{p}_m$ and use the geometric average of their individual transfers as the final prediction $\hat{\mathbf{p}}'_m$ [42]. This consensus



(a) Strongly-supervised loss. **(b)** Weakly-supervised loss.

Fig. 2: Matching loss computation using (a) keypoint annotations (strong supervision) and (b) image pairs only (weak supervision). Best viewed in electronic form.

keypoint transfer method improves accuracy by refining mis-localized predictions of individual transfers.

3.4 Training objective

We propose two objectives to train our model using different degrees of supervision: strongly-supervised and weakly-supervised regimes.

Learning with strong supervision. In this setup, we assume that keypoint match annotations are given for each training image pair, as in [5, 17, 42]; each image pair is annotated with a set of coordinate pairs $\mathcal{M} = \{(\mathbf{p}_m, \mathbf{p}'_m)\}_{m=1}^{M}$, where M is the number of match annotations.

To compare the output of our network with ground-truth annotations, we convert the annotations into a form of discrete correlation matrix. First of all, for each coordinate pair $(\mathbf{p}_m, \mathbf{p}'_m)$, we identify their nearest position indices (k_m, k'_m) in hyperimages. On the one hand, given the set of identified match index pairs $\{(k_m, k'_m)\}_{m=1}^{M}$, we construct a ground-truth matrix $\mathbf{G} \in \{0,1\}^{M \times |\mathcal{H}'|}$ by assigning one-hot vector representation of k'_m to the m-th row of $\mathbf{G}$. On the other hand, we construct $\hat{\mathbf{C}} \in \mathbb{R}^{M \times |\mathcal{H}'|}$ by assigning the k_m-th row of $\mathbf{C}$ to the m-th row of $\hat{\mathbf{C}}$. We apply softmax to each row of the matrix $\hat{\mathbf{C}}$ after normalizing it to have zero mean and unit variance. Figure 2a illustrates the construction of $\hat{\mathbf{C}}$ and $\mathbf{G}$. Corresponding rows between $\hat{\mathbf{C}}$ and $\mathbf{G}$ can now be compared as categorical probability distributions. We thus define the strongly-supervised matching loss as the sum of cross-entropy values between them:

$$\mathcal{L}_{\text{match}} = -\frac{1}{M} \sum_{m=1}^{M} \omega_m \sum_{j=1}^{|\mathcal{H}'|} \mathbf{G}_{mj} \log \hat{\mathbf{C}}_{mj}, \tag{2}$$

where ω_m is an importance weight for the m-th keypoint. The keypoint weight ω_m helps training by reducing the effect of the corresponding cross-entropy term if the Eucliean distance between predicted keypoint $\hat{\mathbf{p}}'_m$ and target keypoint $\mathbf{p}'_m$ is smaller than some threshold distance δ_{thres}:

$$\omega_m = \begin{cases} \left(\|\hat{\mathbf{p}}'_m - \mathbf{p}'_m\| / \delta_{\text{thres}}\right)^2 & \text{if} \quad \|\hat{\mathbf{p}}'_m - \mathbf{p}'_m\| < \delta_{\text{thres}}, \\ 1 & \text{otherwise.} \end{cases} \tag{3}$$

The proposed objective for strongly-supervised learning can also be used for self-supervised learning with synthetic pairs [45, 49]*, which typically results in trading off the cost of supervision against the generalization performance.

Learning with weak supervision. In this setup, we assume that only image-level labels are given for each image pair as either positive (the same class) or negative (different class), as in [23, 47]. Let us denote the correlation matrix of a positive pair by $\mathbf{C}_+$ and that of a negative pair by $\mathbf{C}_-$. For $\mathbf{C} \in \mathbb{R}^{|\mathcal{H}|\times|\mathcal{H}'|}$, we define its correlation entropy as $s(\mathbf{C}) = -\frac{1}{|\mathcal{H}|}\sum_{i=1}^{|\mathcal{H}|}\sum_{j=1}^{|\mathcal{H}'|}\phi(\mathbf{C})_{ij}\log\phi(\mathbf{C})_{ij}$ where $\phi(\cdot)$ denotes row-wise L1-normalization. Higher correlation entropy indicates less distinctive correspondences between the two images. As illustrated in Fig. 2b, assuming that the positive images are likely to contain more distinctive correspondences, we encourage low entropy for positive pairs and high entropy for negative pairs. The weakly-supervised matching loss is formulated as

$$\mathcal{L}_{\text{match}} = \frac{s(\mathbf{C}_+) + s(\mathbf{C}_+^\top)}{s(\mathbf{C}_-) + s(\mathbf{C}_-^\top)}. \tag{4}$$

Layer selection loss. Following the work of [54], we add a soft constraint in our training objective to encourage the network to select each layer at a certain rate: $\mathcal{L}_{\text{sel}} = \sum_{l=0}^{L-1}(\bar{z}_l - \mu)^2$ where $\bar{z}_l$ is a fraction of image pairs within a mini-batch for which the l-th layer is selected and μ is a hyperparameter for the selection rate. This improves training by increasing diversity in layer selection and, as will be seen in our experiments, allows us to trade off between accuracy and speed in testing.

Finally, the training objective of our model is defined as the combination of the matching loss (either strong or weak) and the layer selection loss: $\mathcal{L} = \mathcal{L}_{\text{match}} + \mathcal{L}_{\text{sel}}$.

4 Experiments

In this section we compare our method to the state of the art and discuss the results. The code and the trained model are available online at our project page.

Feature extractor networks. As the backbone networks for feature extraction, we use ResNet-50 and ResNet-101 [19], which contains 49 and 100 conv layers in total (excluding the last FC), respectively. Since features from adjacent layers are strongly correlated, we extract the base block from `conv1` maxpool and intermediate blocks from layers with residual connections (before ReLU). They amounts to 17 and 34 feature blocks (layers) in total, respectively, for ResNet-50 and ResNet-101. Following related work [5, 17, 28, 33, 42, 45, 46, 47, 49, 23], we freeze the backbone network parameters during training for fair comparison.

*For example, we can obtain keypoint annotations for free by forming a synthetic pair by applying random geometric transformation (*e.g.*, affine or TPS [8]) on an image and then sampling some corresponding points between the original image and the warped image using the transformation applied.

Table 1: Performance on standard benchmarks in accuracy and speed (avg. time per pair). The subscript of each method name denotes its feature extractor. Some results are from [25, 28, 33, 42]. Numbers in bold indicate the best performance and underlined ones are the second best. The average inference time (the last column) is measured on test split of PF-PASCAL [16] and includes all the pipelines of the models: from feature extraction to keypoint prediction.

Sup.	Sup. signal	Methods	PF-PASCAL PCK @ α_{img} 0.05	0.1	0.15	α_{bbox} 0.1	PF-WILLOW PCK @ α_{bbox} 0.05	0.1	0.15	Caltech-101 LT-ACC	IoU	time (ms)
none	-	PF_{HOG} [15]	31.4	62.5	79.5	45.0	28.4	56.8	68.2	0.78	0.50	>1000
self	synthetic pairs	$\text{CNNGeo}_{\text{res101}}$ [45]	41.0	69.5	80.4	68.0	36.9	69.2	77.8	0.79	0.56	40
		$\text{A2Net}_{\text{res101}}$ [49]	42.8	70.8	83.3	67.0	36.3	68.8	84.4	0.80	0.57	53
weak	bbox	$\text{SF-Net}_{\text{res101}}$ [33]	53.6	81.9	90.6	78.7	46.3	74.0	84.2	0.88	0.67	51
	image-level labels	$\text{Weakalign}_{\text{res101}}$ [46]	49.0	74.8	84.0	72.0	37.0	70.2	79.9	0.85	**0.63**	41
		$\text{RTNs}_{\text{res101}}$ [28]	55.2	75.9	85.2	-	41.3	71.9	86.2	-	-	376
		$\text{NC-Net}_{\text{res101}}$ [47]	54.3	78.9	86.0	70.0	33.8	67.0	83.7	0.85	0.60	261
		$\text{DCC-Net}_{\text{res101}}$ [23]	55.6	**82.3**	90.5	-	43.6	73.8	86.5	-	-	>261
		$\text{DHPF}^{\mu=0.4}_{\text{res50}}$ (ours)	54.8	79.0	89.8	74.5	48.7	75.7	87.3	0.85	0.59	**31**
		$\text{DHPF}_{\text{res50}}$ (ours)	54.7	79.0	89.7	74.5	**51.8**	78.7	89.6	0.85	0.59	33
		$\text{DHPF}_{\text{res101}}$ (ours)	**56.1**	82.1	**91.1**	**78.5**	50.2	**80.2**	**91.1**	**0.86**	0.61	56
strong	src & trg keypoint matches	$\text{SCNet}_{\text{vgg16}}$ [17]	36.2	72.2	82.0	48.2	38.6	70.4	85.3	0.79	0.51	>1000
		$\text{HPF}_{\text{res50}}$ [42]	60.5	83.4	92.1	76.5	46.5	72.4	84.7	**0.88**	**0.64**	34
		$\text{HPF}_{\text{res101}}$ [42]	60.1	84.8	92.7	78.5	45.9	74.4	85.6	0.87	0.63	63
		$\text{DHPF}^{\mu=0.4}_{\text{res50}}$ (ours)	70.2	89.1	94.0	85.0	45.8	73.3	86.6	0.86	0.60	**30**
		$\text{DHPF}_{\text{res50}}$ (ours)	72.6	88.9	94.3	85.6	47.9	74.8	86.7	0.86	0.61	34
		$\text{DHPF}_{\text{res101}}$ (ours)	**75.7**	**90.7**	**95.0**	**87.8**	**49.5**	**77.6**	**89.1**	0.87	0.62	58

Datasets. Experiments are done on four benchmarks for semantic correspondence: PF-PASCAL [16], PF-WILLOW [15], Caltech-101 [34], and SPair-71k [43]. PF-PASCAL and PF-WILLOW consist of keypoint-annotated image pairs, 1,351 pairs from 20 categories, and 900 pairs from 4 categories, respectively. Caltech-101 [34] contains segmentation-annotated 1,515 pairs from 101 categories. SPair-71k [43] is a more challenging large-scale dataset recently introduced in [42], consisting of keypoint-annotated 70,958 image pairs from 18 categories with diverse view-point and scale variations.

Evaluation metrics. As an evaluation metric for PF-PASCAL, PF-WILLOW, and SPair-71k, the probability of correct keypoints (PCK) is used. The PCK value given a set of predicted and ground-truth keypoint pairs $\mathcal{P} = \{(\hat{\mathbf{p}}'_m,\ \mathbf{p}'_m)\}_{m=1}^{M}$ is measured by $\text{PCK}(\mathcal{P}) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}[\|\hat{\mathbf{p}}'_m - \mathbf{p}'_m\| \leq \alpha_\tau \max(w_\tau, h_\tau)]$. As an evaluation metric for the Caltech-101 benchmark, the label transfer accuracy (LT-ACC) [36] and the intersection-over-union (IoU) [9] are used. Running time (average time per pair) for each method is measured using its authors' code on a machine with an Intel i7-7820X CPU and an NVIDIA Titan-XP GPU.

Hyperparameters. The layer selection rate μ and the channel reduction factor ρ are determined by grid search using the validation split of PF-PASCAL. As a

Table 2: Performance on SPair-71k dataset in accuracy (per-class PCK with $\alpha_{\text{bbox}} = 0.1$). TR represents transferred models trained on PF-PASCAL while FT denotes fine-tuned (trained) models on SPair-71k.

Sup.	Methods	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	dog	horse	mbike	person	plant	sheep	train	tv	all
self	TR CNNGeo$_{\text{res101}}$ [45]	21.3	15.1	34.6	12.8	31.2	26.3	24.0	30.6	11.6	24.3	20.4	12.2	19.7	15.6	14.3	9.6	28.5	28.8	18.1
	FT CNNGeo$_{\text{res101}}$ [45]	23.4	16.7	40.2	14.3	36.4	27.7	26.0	32.7	12.7	27.4	22.8	13.7	20.9	21.0	17.5	10.2	30.8	34.1	20.6
	TR A2Net$_{\text{res101}}$ [49]	20.8	17.1	37.4	13.9	33.6	29.4	26.5	34.9	12.0	26.5	22.5	13.3	21.3	20.0	16.9	11.5	28.9	31.6	20.1
	FT A2Net$_{\text{res101}}$ [49]	22.6	18.5	42.0	16.4	37.9	30.8	26.5	35.6	13.3	29.6	24.3	16.0	21.6	22.8	20.5	13.5	31.4	36.5	22.3
weak	TR WeakAlign$_{\text{res101}}$ [46]	23.4	17.0	41.6	14.6	37.6	**28.1**	26.6	32.6	12.6	27.9	23.0	13.6	21.3	22.2	17.9	10.9	31.5	34.8	21.1
	FT WeakAlign$_{\text{res101}}$ [46]	22.2	17.6	41.9	15.1	**38.1**	27.4	**27.2**	31.8	12.8	26.8	22.6	14.2	20.0	22.2	17.9	10.4	32.2	35.1	20.9
	TR NC-Net$_{\text{res101}}$ [47]	**24.0**	16.0	45.0	13.7	35.7	25.9	19.0	50.4	14.3	32.6	27.4	19.2	21.7	20.3	20.4	13.6	**33.6**	40.4	26.4
	FT NC-Net$_{\text{res101}}$ [47]	17.9	12.2	32.1	11.7	29.0	19.9	16.1	39.2	9.9	23.9	18.8	15.7	17.4	15.9	14.8	9.6	24.2	31.1	20.1
	TR DHPF$_{\text{res101}}$ (ours)	21.5	**21.8**	**57.2**	13.9	34.3	23.1	17.3	50.4	**17.4**	34.8	**36.2**	**19.7**	24.3	**32.5**	**22.2**	17.6	30.9	36.5	**28.5**
	FT DHPF$_{\text{res101}}$ (ours)	17.5	19.0	52.5	**15.4**	35.0	19.4	15.7	**51.9**	17.3	**37.3**	35.7	**19.7**	**25.5**	31.6	20.9	**18.5**	24.2	**41.1**	27.7
strong	FT HPF$_{\text{res101}}$ [42]	25.2	18.9	52.1	15.7	38.0	22.8	19.1	52.9	17.9	33.0	32.8	20.6	24.4	27.9	21.1	15.9	31.5	35.6	28.2
	TR DHPF$_{\text{res101}}$ (ours)	22.6	23.0	57.7	15.1	34.1	20.5	14.7	48.6	19.5	31.9	34.5	19.6	23.0	30.0	22.9	15.5	28.2	30.2	27.4
	FT DHPF$_{\text{res101}}$ (ours)	**38.4**	**23.8**	**68.3**	**18.9**	**42.6**	**27.9**	**20.1**	**61.6**	**22.0**	**46.9**	**46.1**	**33.5**	**27.6**	**40.1**	**27.6**	**28.1**	**49.5**	**46.5**	**37.3**

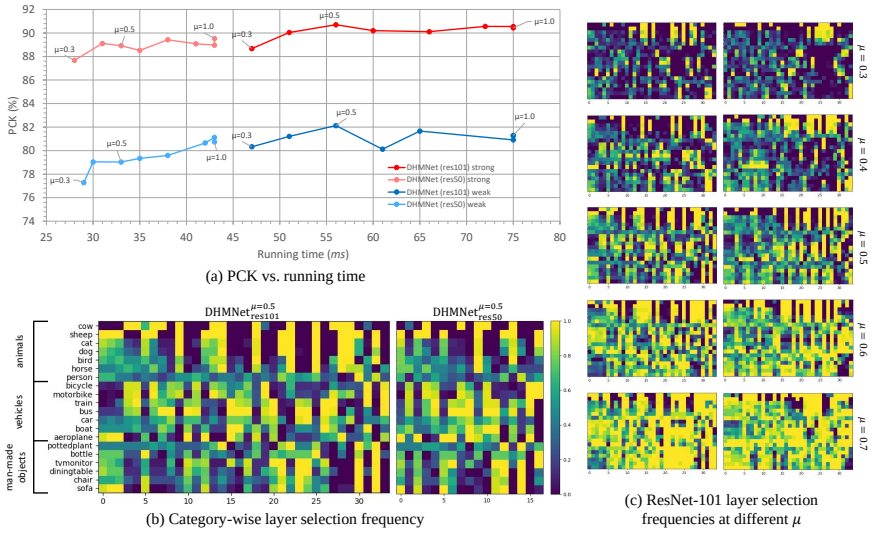
result, we set $\mu = 0.5$ and $\rho = 8$ in our experiments if not specified otherwise. The threshold δ_{thres} in Eq.(3) is set to be $\max(w_\tau, h_\tau)/10$.

4.1 Results and comparisons

First, we train both of our strongly and weakly-supervised models on the PF-PASCAL [16] dataset and test on three standard benchmarks of PF-PASCAL (test split), PF-WILLOW and Caltech-101. The evaluations on PF-WILLOW and Caltech-101 are to verify transferability. In training, we use the same splits of PF-PASCAL proposed in [17] where training, validation, and test sets respectively contain 700, 300, and 300 image pairs. Following [46, 47], we augment the training pairs by horizontal flipping and swapping. Table 1 summarizes our result and those of recent methods [15, 17, 28, 30, 42, 45, 46, 47, 49]. Second, we train our model on the SPair-71k dataset [43] and compare it to other recent methods [42, 45, 46, 47, 49]. Table 2 summarizes the results.

Strongly-supervised regime. As shown in the bottom sections of Table 1 and 2, our strongly-supervised model clearly outperforms the previous state of the art by a significant margin. It achieves 5.9%, 3.2%, and 9.1% points of PCK ($\alpha_{\text{img}} = 0.1$) improvement over the current state of the art [42] on PF-PASCAL, PF-WILLOW, and SPair-71k, respectively, and the improvement increases further with a more strict evaluation threshold, *e.g.*, more than 15% points of PCK with $\alpha_{\text{img}} = 0.05$ on PF-PASCAL. Even with a smaller backbone network (ResNet-50) and smaller selection rate ($\mu = 0.4$), our method achieves competitive performance with the smallest running time on the standard benchmarks of PF-PASCAL, PF-WILLOW, and Caltech-101.

Weakly-supervised regime. As shown in the middle sections of Table 1 and 2, our weakly-supervised model also achieves the state of the art in the weakly-supervised regime. In particular, our model shows more reliable transferablility compared to strongly-supervised models, outperforming both weakly [23] and



(a) PCK vs. running time

(b) Category-wise layer selection frequency

(c) ResNet-101 layer selection frequencies at different μ

Fig. 3: Analysis of layer selection on PF-PASCAL dataset (a) PCK vs. running time with varying selection rate μ (b) Category-wise layer selection frequencies (x-axis: candidate layer index, y-axis: category) of the strongly-supervised model with different backbones: ResNet-101 (left) and ResNet-50 (right) (c) ResNet-101 layer selection frequencies of strongly (left) and weakly (right) supervised models at different layer selection rates μ. Best viewed in electronic form.

strongly-supervised [42] state of the arts by 6.4% and 5.8% points of PCK respectively on PF-WILLOW. On the Caltech-101 benchmark, our method is comparable to the best among the recent methods. Note that unlike other benchmarks, the evaluation metric of Caltech-101 is indirect (*i.e.*, accuracy of mask transfer). On the SPair-71k dataset, where image pairs have large view point and scale differences, the methods of [46, 47] as well as ours do not successfully learn in the weakly-supervised regime; they (FT) all underperform transferred models (TR) trained on PF-PASCAL. This result reveals current weakly-supervised objectives are all prone to large variations, which requires further research in the future.

Effect of layer selection rate μ [54]. The plot in Fig. 3a shows PCK and running time of our models trained with different layer selection rates μ. It shows that smaller selection rates in training lead to faster running time in testing, at the cost of some accuracy, by encouraging the model to select a smaller number of layers. The selection rate μ can thus be used for speed-accuracy trade-off.

Analysis of layer selection patterns. Category-wise layer selection patterns in Fig. 3b show that each group of animal, vehicle, and man-made object categories shares its own distinct selection patterns. The model with a small rate ($\mu = 0.3$) tends to select the most relevant layers only while the model with larger rates

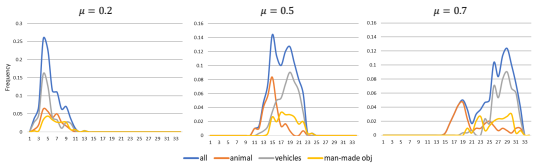


Fig. 4: Frequencies over the numbers of selected layers with different selection rates μ (x-axis: the number of selected layers, y-axis: frequency). Best viewed in electronics.

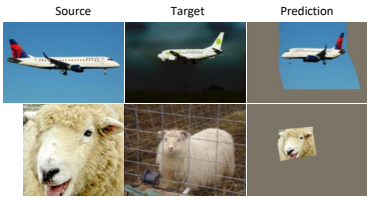


Fig. 5: Example results on SPair-71k dataset. The source images are warped to the target ones using resultant correspondences.

(a) Source (b) Target (c) DHPF (ours) (d) WeakAlign (e) A2Net (f) NC-Net (g) HPF

Fig. 6: Example results on PF-PASCAL [16]: (a) source image, (b) target image and (c) DHPF (ours), (d) WeakAlign [46], (e) A2Net [49], (f) NC-Net [47], and (g) HPF [42].

($\mu > 0.3$) tends to select more complementary layers as seen in Fig.3c. For each $\mu \in \{0.3, 0.4, 0.5\}$ in Fig.3c, the network tends to select low-level features for vehicle and man-made object categories while it selects mostly high-level features for animal category. We conjecture that it is because low-level (geometric) features such as lines, corners and circles appear more often in the vehicle and man-made classes compared to the animal classes. Figure 4 plots the frequencies over the numbers of selected layers with different selection rate μ, where vehicles tend to require more layers than animals and man-made objects.

Qualitative results. Some challenging examples on SPair-71k [43] and PF-PASCAL [16] are shown in Fig.5 and 6 respectively: Using the keypoint correspondences, TPS transformation [8] is applied to source image to align target image. The object categories of the pairs in Fig.6 are in order of table, potted plant, and tv. Alignment results of each pair demonstrate the robustness of our model against major challenges in semantic correspondences such as large changes in view-point and scale, occlusion, background clutters, and intra-class variation.

Ablation study. We also conduct an ablation study to see the impacts of major components: Gumbel layer gating (GLG), conv feature transformation (CFT), probabilistic Hough matching (PHM), keypoint importance weight ω_m, and layer selection loss $\mathcal{L}_{\text{sel}}$. All the models are trained with strong supervision

Table 3: Ablation study on PF-PASCAL. (GLG: Gumbel layer gating with selection rates μ, CFT: conv feature transformation)

Module			PCK (α_{img})			time
GLG	CFT	PHM	0.05	0.1	0.15	(ms)
0.5	✓	✓	75.7	90.7	95.0	58
0.4	✓	✓	73.6	90.4	95.3	51
0.3	✓	✓	73.1	88.7	94.4	47
	✓	✓	70.4	88.1	94.1	64
0.5		✓	43.6	74.7	87.5	176
0.5	✓		68.3	86.9	91.6	57
		✓	37.6	68.7	84.6	124
	✓		68.1	85.5	91.6	61
0.5			35.0	54.8	63.4	173
w/o ω_m			69.8	86.1	91.9	57
w/o $\mathcal{L}_{\text{sel}}$			68.1	89.2	93.5	56

Table 4: Comparison to soft layer gating on PF-PASCAL.

Gating function	PCK (α_{img})			time
	0.05	0.1	0.15	(ms)
Gumbel$_{\mu=0.5}$	75.7	90.7	95.0	58
sigmoid	71.1	88.2	92.8	74
sigmoid$_{\mu=0.5}$	72.1	87.8	93.3	75
sigmoid + $\ell 1$	65.9	87.2	91.0	60

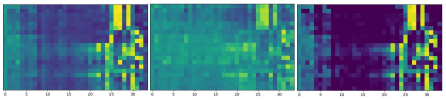


Fig. 7: ResNet-101 layer selection frequencies for 'sigmoid' (left), 'sigmoid$_{\mu=0.5}$' (middle), and 'sigmoid + $\ell 1$' (right) gating.

and evaluated on PF-PASCAL. Since the models with a PHM component have no training parameters, they are directly evaluated on the test split. Table 3 summarizes the results. It reveals that among others CFT in the dynamic gating module is the most significant component in boosting performance and speed; without the feature transformation along with channel reduction, our models do not successfully learn in our experiments and even fail to achieve faster per-pair inference time. The result of 'w/o ω_m' reveals the effect of the keypoint weight ω_m in Eq.(2) by replacing it with uniform weights for all m, *i.e.*, $\omega_m = 1$; putting less weights on easy examples helps in training the model by focusing on hard examples. The result of 'w/o $\mathcal{L}_{\text{sel}}$' shows the performance of the model using $\mathcal{L}_{\text{match}}$ only in training; performance drops with slower running time, demonstrating the effectiveness of the layer selection constraint in terms of both speed and accuracy. With all the components jointly used, our model achieves the highest PCK measure of 90.7%. Even with the smaller backbone network, ResNet-50, the model still outperforms previous state of the art and achieves real-time matching as well as described in Fig.3 and Table 1.

Computational complexity. The average feature dimensions of our model before correlation computation are 2089, 3080, and 3962 for each $\mu \in \{0.3, 0.4, 0.5\}$ while those of recent methods [42, 33, 47, 23] are respectively 6400, 3072, 1024, 1024. The dimension of hyperimage is relatively small as GLG efficiently prunes irrelevant features and CFT effectively maps features onto smaller subspace, thus being more practical in terms of speed and accuracy as demonstrated in Table 1 and 3. Although [47, 23] use lighter feature maps compared to ours, a series of 4D convolutions heavily increases time and memory complexity of the network, making them expensive for practical use (31ms (ours) vs. 261ms [47, 23]).

4.2 Comparison to soft layer gating

The Gumbel gating function in our dynamic layer gating can be replaced with conventional soft gating using sigmoid. We have investigated different types of soft gating as follows: (1) 'sigmoid': The MLP of dynamic gating at each layer predicts a scalar input for sigmoid and the transformed feature block pairs are weighted by the sigmoid output. (2) 'sigmoid$_{\mu=0.5}$': In training the 'sigmoid' gating, the layer selection loss $\mathcal{L}_{\text{sel}}$ with $\mu = 0.5$ is used to encourage the model to increase diversity in layer selection. (3) 'sigmoid + $\ell 1$': In training the 'sigmoid' gating, the $\ell 1$ regularization on the sigmoid output is used to encourage the soft selection result to be sparse. Table 4 summarizes the results and Fig. 7 compares their layer selection frequencies.

While the soft gating modules provide decent results, all of them perform worse than the proposed Gumbel layer gating in both accuracy and speed. The slower per-pair inference time of 'sigmoid' and 'sigmoid$_{\mu=0.5}$' indicates that *soft* gating is not effective in skipping layers due to its non-zero gating values. We find that the sparse regularization of 'sigmoid + $\ell 1$' recovers the speed but only at the cost of significant accuracy points. Performance drop of soft gating in accuracy may result from the *deterministic* behavior of the soft gating during training that prohibits exploring diverse combinations of features at different levels. In contrast, the Gumbel gating during training enables the network to perform more comprehensive trials of a large number of different combinations of multi-level features, which help to learn better gating. Our experiments also show that *discrete* layer selection along with *stochastic* learning in searching the best combination is highly effective for learning to establish robust correspondences in terms of both accuracy and speed.

5 Conclusion

We have presented a dynamic matching network that predicts dense correspondences by composing hypercolumn features using a small set of relevant layers from a CNN. The state-of-the-art performance of the proposed method indicates that the use of dynamic multi-layer features in a trainable architecture is crucial for robust visual correspondence. We believe that our approach may prove useful for other domains involving correspondence such as image retrieval, object tracking, and action recognition. We leave this to future work.

Acknowledgements. This work is supported by Samsung Advanced Institute of Technology (SAIT) and also by Basic Science Research Program (NRF-2017R1E1A1A01077999) and Next-Generation Information Computing Development Program (NRF-2017M3C4A7069369) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT, Korea. Jean Ponce was supported in part by the Louis Vuitton/ENS chair in artificial intelligence and the Inria/NYU collaboration and also by the French government under management of Agence Nationale de la Recherche as part of the "Investissements dâavenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

References

1. Andreas, J., Rohrbach, M., Darrell, T., Klein, D.: Learning to compose neural networks for question answering. In: Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL) (2016) 3
2. Andreas, J., Rohrbach, M., Darrell, T., Klein, D.: Neural module networks. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 3
3. Bristow, H., Valmadre, J., Lucey, S.: Dense semantic correspondence where every pixel is a classifier. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2015) 2
4. Cho, M., Kwak, S., Schmid, C., Ponce, J.: Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 2, 6
5. Choy, C.B., Gwak, J., Savarese, S., Chandraker, M.: Universal correspondence network. In: Proc. Neural Information Processing Systems (NeurIPS) (2016) 2, 4, 7, 8
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005) 2
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009) 4
8. Donato, G., Belongie, S.: Approximate thin plate spline mappings. In: Proc. European Conference on Computer Vision (ECCV) (2002) 8, 12
9. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision (IJCV) **88**, 303–338 (2010) 9
10. Fathy, M.E., Tran, Q.H., Zeeshan Zia, M., Vernaza, P., Chandraker, M.: Hierarchical metric learning and matching for 2d and 3d geometric correspondences. In: Proc. European Conference on Computer Vision (ECCV) (2018) 2
11. Figurnov, M., Collins, M.D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhutdinov, R.: Spatially adaptive computation time for residual networks. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2, 3
12. Forsyth, D., Ponce, J.: Computer Vision: A Modern Approach. (Second edition). Prentice Hall (2011) 1
13. Gao, X., Zhao, Y., Dudziak, L., Mullins, R., Xu, C.z.: Dynamic channel pruning: Feature boosting and suppression. In: Proc. International Conference on Learning Representations (ICLR) (2019) 3
14. Gumbel, E.: Statistical theory of extreme values and some practical applications: a series of lectures. Applied mathematics series, U. S. Govt. Print. Office (1954) 5
15. Ham, B., Cho, M., Schmid, C., Ponce, J.: Proposal flow. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 2, 6, 9, 10
16. Ham, B., Cho, M., Schmid, C., Ponce, J.: Proposal flow: Semantic correspondences from object proposals. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **40**, 1711–1725 (2018) 9, 10, 12
17. Han, K., Rezende, R.S., Ham, B., Wong, K.Y.K., Cho, M., Schmid, C., Ponce, J.: Scnet: Learning semantic correspondence. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2017) 2, 4, 6, 7, 8, 9, 10

18. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 2, 3, 4
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 1, 2, 8
20. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 1
21. Hua, W., De Sa, C., Zhang, Z., Suh, G.E.: Channel gating neural networks. arXiv preprint arXiv:1805.12549 (2018) 3
22. Huang*, G., Liu*, Z., van der Maaten, L., Weinberger, K.: Densely connected convolutional networks. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 1, 2
23. Huang, S., Wang, Q., Zhang, S., Yan, S., He, X.: Dynamic context correspondence network for semantic alignment. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 4, 8, 9, 10, 13
24. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. In: Proc. International Conference on Learning Representations (ICLR) (2017) 5
25. Jeon, S., Kim, S., Min, D., Sohn, K.: Parn: Pyramidal affine regression networks for dense semantic correspondence. In: Proc. European Conference on Computer Vision (ECCV) (2018) 2, 9
26. Kanazawa, A., Jacobs, D.W., Chandraker, M.: Warpnet: Weakly supervised matching for single-view reconstruction. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 2
27. Kim, J., Liu, C., Sha, F., Grauman, K.: Deformable spatial pyramid matching for fast dense correspondences. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2013) 2
28. Kim, S., Lin, S., Jeon, S.R., Min, D., Sohn, K.: Recurrent transformer networks for semantic correspondence. In: Proc. Neural Information Processing Systems (NeurIPS) (2018) 2, 4, 8, 9, 10
29. Kim, S., Min, D., Ham, B., Jeon, S., Lin, S., Sohn, K.: Fcss: Fully convolutional self-similarity for dense semantic correspondence. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2
30. Kim, S., Min, D., Lin, S., Sohn, K.: Dctm: Discrete-continuous transformation matching for semantic flow. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2017) 2, 4, 10
31. Kong, T., Yao, A., Chen, Y., Sun, F.: Hypernet: Towards accurate region proposal generation and joint object detection. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 2, 3
32. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. Neural Information Processing Systems (NeurIPS) (2012) 1, 2
33. Lee, J., Kim, D., Ponce, J., Ham, B.: Sfnet: Learning object-aware semantic correspondence. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 2, 4, 8, 9, 13
34. Li, F.F., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **28**, 594–611 (2006) 9
35. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2, 3

36. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing: Label transfer via dense scene alignment. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009) 9
37. Liu, C., Yuen, J., Torralba, A.: Sift flow: Dense correspondence across scenes and its applications. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **33**, 978–994 (2011) 2
38. Liu, S., Huang, D., et al.: Receptive field block net for accurate and fast object detection. In: Proc. European Conference on Computer Vision (ECCV) (2018) 2, 3
39. Long, J.L., Zhang, N., Darrell, T.: Do convnets learn correspondence? In: Proc. Neural Information Processing Systems (NeurIPS) (2014) 2, 3
40. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision (IJCV) **60**, 91–110 (2004) 2
41. Maddison, C., Mnih, A., Whye Teh, Y.: The concrete distribution: A continuous relaxation of discrete random variables. In: Proc. International Conference on Learning Representations (ICLR) (2017) 5
42. Min, J., Lee, J., Ponce, J., Cho, M.: Hyperpixel flow: Semantic correspondence with multi-layer neural features. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13
43. Min, J., Lee, J., Ponce, J., Cho, M.: SPair-71k: A large-scale benchmark for semantic correspondence. arXiv preprint arXiv:1908.10543 (2019) 9, 10, 12
44. Novotny, D., Larlus, D., Vedaldi, A.: Anchornet: A weakly supervised network to learn geometry-sensitive features for semantic matching. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2, 3
45. Rocco, I., Arandjelovic, R., Sivic, J.: Convolutional neural network architecture for geometric matching. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2, 4, 8, 9, 10
46. Rocco, I., Arandjelović, R., Sivic, J.: End-to-end weakly-supervised semantic alignment. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 2, 4, 8, 9, 10, 11, 12
47. Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., Sivic, J.: Neighbourhood consensus networks. In: Proc. Neural Information Processing Systems (NeurIPS) (2018) 2, 4, 6, 8, 9, 10, 11, 12, 13
48. Schonberger, J.L., Hardmeier, H., Sattler, T., Pollefeys, M.: Comparative evaluation of hand-crafted and learned local features. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2
49. Seo, P.H., Lee, J., Jung, D., Han, B., Cho, M.: Attentive semantic alignment with offset-aware correlation kernels. In: Proc. European Conference on Computer Vision (ECCV) (2018) 2, 4, 8, 9, 10, 12
50. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proc. International Conference on Learning Representations (ICLR) (2015) 1
51. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. In: Proc. International Conference on Machine Learning (ICML) (2015) 2, 3
52. Taniai, T., Sinha, S.N., Sato, Y.: Joint recovery of dense correspondence and cosegmentation in two images. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 2
53. Ufer, N., Ommer, B.: Deep semantic feature matching. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2, 3
54. Veit, A., Belongie, S.: Convolutional networks with adaptive inference graphs. In: Proc. European Conference on Computer Vision (ECCV) (2018) 2, 3, 8, 11

55. Yang, F., Li, X., Cheng, H., Li, J., Chen, L.: Object-aware dense semantic correspondence. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2
56. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Proc. European Conference on Computer Vision (ECCV) (2014) 3
57. Zhou, T., Krahenbuhl, P., Aubry, M., Huang, Q., Efros, A.A.: Learning dense correspondence via 3d-guided cycle consistency. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 2