Efficient Semantic Matching with Hypercolumn Correlation
— *Supplementary Material* —

Seungwook Kim Juhong Min Minsu Cho
Pohang University of Science and Technology (POSTECH), South Korea
http://cvlab.postech.ac.kr/research/HCCNet

In this supplementary material, we provide additional implementation details (Section 1), quantitative results with analyses (Section 2), and additional qualitative results (Section 3) of our proposed HCCNet, an efficient and effective semantic matching learner on a hypercolumn correlation.

1. Additional implementation details.

Visualization of feature slicing. The visualization of our proposed feature slicing is provided in Figure 1. Given L intermediate feature maps whose channel dimensions sum up to C, using the slice size of N results in a total of $G = \frac{C}{N}$ feature slices *with the same channel dimensions* for hypercolumn correlation computation.

Hyperparameters for flow field formation and keypoint transfer. We provide detailed hyperparameters for our methods explained in the paper. $\mathbf{G}^{\mathbf{p}} \in \mathbb{R}^{30\times30}$, the 2-dimensional Gaussian kernel centered on $\mathbf{p} = \arg\max_{\mathbf{y}} \mathbf{C}^{\text{out}}_{\mathbf{x},\mathbf{y}}$ in Eqn. 8 of the main paper, has a standard deviation of 10. The distance threshold τ for the soft sampler in Eqn. 9 of the main paper is set to 0.1 during training, and 0.05 during inference.

Coordinate normalization of dense flow field. Adhering to the conventions used in [6–8], we normalize the coordinates of the dense regular grids - $\mathbf{P}_X, \mathbf{P}_Y \in \mathbb{R}^{\bar{H}\bar{W}\times2}$ for the source and target images, respectively - such that the dense regular grids and the dense flow field will have coordinates in the range $\left[\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right]$. This coordinate normalization aims at numerically stabilizing the loss gradients.

ResNet-101 feature extractor settings. Throughout training, the weights of the ResNet-101 [3] feature extractor network are frozen up to `conv3_x` to prevent overfitting the network to the train set, since all semantic matching benchmarks have a significantly smaller number of images ($\tilde{2}$K) compared to the ImageNet dataset [1].

Additional details for feature slicing and hypercolumn correlation construction. As mentioned in Section 3 of the main paper, we view each feature map at every layer as a composition of multiple sub-features concatenated along the channel dimension: $\hat{\mathbf{X}}^{(l)} := \text{concat}_{g\in G^{(l)}} \left[\mathbf{X}^{(g)}\right]$ for all $l \in [L]$, where $G^{(l)}$ is the number of slices used to divide feature map $\mathbf{X}^{(l)}$. Throughout the main paper, we refer to $G^{(l)}$ as (number of) groups, and the resulting channel size of each slice as the slice size. In our experimental settings, we use the intermediate features extracted from the bottleneck layers of `conv3_x`, `conv4_x` and `conv5_x`, where the number of bottleneck layers are 4, 23 and 3, respectively, and the intermediate features have channel sizes of 512, 1024 and 2048, respectively. We use a slice size of 256, resulting in $\frac{(4\times512)+(23\times1024)+(3\times2048)}{256} = 124$ number of groups. Note that the first column of Table 5 of the main table refers to the slice size, while G in Figure 4 of the main paper refers to the group number. Therefore, in Figure 4 of the main paper, $G = 30$, $G = 62$, $G = 124$ and $G = 248$ corresponds to slice sizes of None (not sliced, bottleneck layer features are used directly), 512, 256 and 128, respectively.

Experimental environment. All experiments are run on a machine with an Intel Xeon Gold 6242 CPU and an NVIDIA TITAN RTX with 24G VRAM.

2. Additional quantitative results and analysis.

Analysis on efficacy of feature slicing. To provide an in-depth analysis on feature slicing, we plot input $\mathbf{C}$ and hidden $\zeta(\mathbf{C}\mathbf{W}_{\text{hid}})$ correlation statistics from our pointwise convolution in Fig. 2 where each bar represents average magnitude (y-axis) for some group g (x-axis) of the hypercolumn correlation[1]. Both models exhibit similar trends, but the statistics of using our proposed feature slicing (blue) shows higher variance in magnitudes, indicating fine-grained differentiation of relevant correlation activations. In contrast, the statistics without feature slicing (red) is relatively uniform, implying the inability to exploit rich channel-wise information of the input feature pairs $\{(\hat{\mathbf{X}}^{(l)}, \hat{\mathbf{Y}}^{(l)})\}_{l\in[L]}$.

[1] E.g., $\frac{1}{HWHW}\sum_{(\mathbf{x},\mathbf{y})} |\mathbf{C}_{\mathbf{x},\mathbf{y},g}|$ given $\mathbf{C}$ as input.

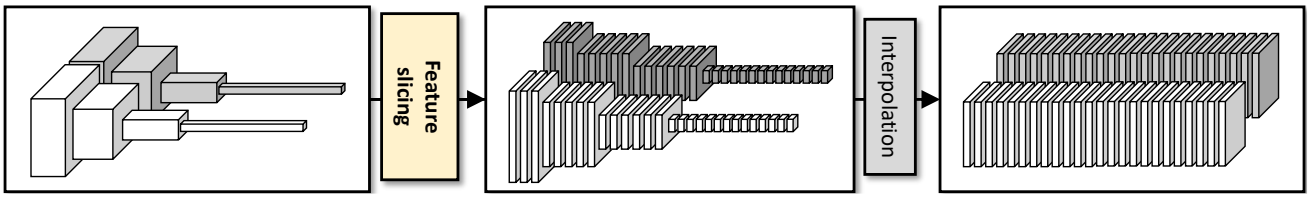

Figure 1. **Visualization of feature slicing.** Feature slicing slices the multi-level features with varying channel dimensions to an increased number of equi-channel features for the computation of rich hypercolumn correlation.

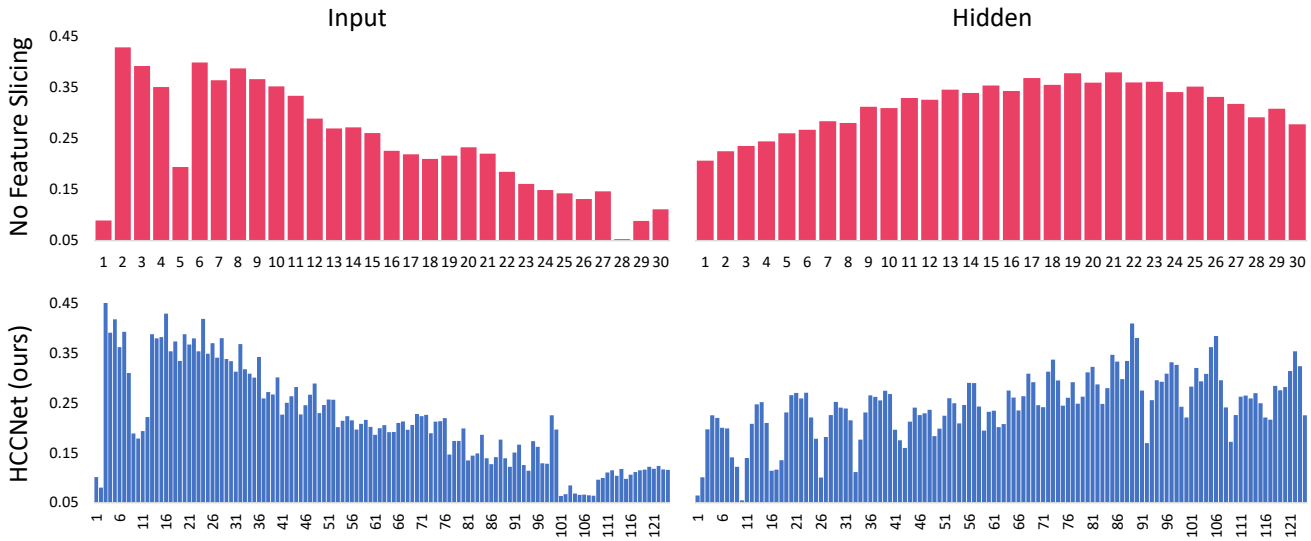

Figure 2. **Average magnitude for some group g with and without feature slicing.** The high variance of magnitudes when using our feature slicing implies that feature slicing enables fine-grained differentiation of relevant correlation activations to establish more reliable correspondences.

Influence of image size on HCCNet. In Table 1, we compare the results of HCCNet when using varying image sizes. It can be seen that our image size of 240×240 achieves the best performance-compute tradeoff. We suggest that the optimal image size depends on the method, as existing methods show optimal performances at varying image dimensions.

Img size	PF-PASCAL 0.05	PF-PASCAL 0.1	time (*ms*)	mem. (GB)	FLOPs (G)
224	60.4	88.1	29	2.0	1.5
240 (Ours)	80.2	92.4	30	2.0	1.7
256	77.9	92.0	30	2.2	2.0
320	71.2	89.7	38	3.1	4.0
384	67.1	87.3	49	4.3	7.3

Table 1. **Results of HCCNet when using varying image sizes.** The results show that our current setting of using 240 as the height and width of the images yields the best results.

Analysis on feature slicing strategy. In Table 2, we compare against other potential slicing strategies. HCCNet clusters neighbouring features into a slice (*Neighbour*). *Random*$_{test}$ refers to random clustering of features at test time for the *Neighbour* model, without replacement. *Even dist.* clusters the features in an evenly distributed manner *i.e.,* features of a slice comes from across the entire list of features. Our strategy yields the best performance.

Slicing strategy	PF-PASCAL 0.05	PF-PASCAL 0.1
None	77.3	92.2
Neighbour (Ours)	80.2	92.4
Random$_{test}$	$54.9_{\pm 0.40}$	$77.7_{\pm 0.62}$
Even dist.	75.8	91.1

Table 2. **Results of HCCNet when using varying feature slicing strategies.** It can be seen that our current choice of assigning neighbouring features into a slice shows better performances compared to random or evenly distributed assignment of features.

Comparison to SCorrSAN [5]. We compare HCCNet against SoTA semantic matching methods of VAT [4] and SCorrSAN [5]. For SCorrSAN, the code was released but

Method	SPair-71k @α_{bbox} 0.05	SPair-71k @α_{bbox} 0.1	PF-PASCAL @α_{img} 0.05	PF-PASCAL @α_{img} 0.1	Image size	time (*ms*)	memory (GB)	FLOPs (G)
VAT [4]	31.7	54.2	78.3	92.3	512×512	127	3.6	68.0
SCorrSAN [5]	-	49.8	-	-	256×256	**28**	**1.5**	2.1
SCorrSAN$_{\text{MT}}$ [5]	-	**55.3**	**81.5**	**93.3**	256×256	**28**	**1.5**	2.1
HCCNet (ours)	**35.8**	54.8	80.2	92.4	240×240	30	2.0	**1.7**

Table 3. **PCK results on SPair-71k and PF-PASCAL in comparison to VAT and SCorrSAN.** We compare HCCNet with SoTA semantic matching methods of VAT [4] SCorrSAN [5], where HCCNet still exhibits competitive performance and the lowest FLOPs. While SCorrSAN$_{\text{MT}}$ shows the best results, their main contributions are distillation and label densification schemes which are both complementary to the model architecture used. SCorrSAN shows the results when the label densification and distillation schemes are unused. Therefore, HCCNet is also expected to benefit significantly when these schemes are applied.

not its pretrained weights - we therefore take the PCK results from their paper, and report the latency, FLOPs and memory using an unpretrained version of SCorrSAN. We include VAT specifically for a comparative evaluation of HCCNet on a lower PCK threshold of SPair-71k dataset (*i.e.* at $\alpha_{\text{bbox}} = 0.05$). Table 3 presents the results.

It can be seen that HCCNet outperforms VAT on all datasets at all thresholds, while incurring significantly lower latency (approx. 3 times), memory (approx. 1.5 times) and FLOPs (40 times). For SCorrSAN, we report two results - one without both knowledge distillation and label densification (SCorrSAN), and the other with both (SCorrSAN$_{\text{MT}}$). SCorrSAN exhibits notably lower FLOPs compared to HCCNet thanks to their efficient spatial context encoder module, albeit incurring higher FLOPs. While their reported performance is considerably lower that those of VAT or HCCNet, they can leverage the lightweightedness of their SCorrsSAN model to perform knowledge distillation and label densification to largely boost their performance (SCorrSAN$_{\text{MT}}$). We highlight that the knowledge distillation and label densification scheme of SCorrSAN$_{\text{MT}}$ is complementary to the model architecture, and we expect a significant performance improvement once we apply SCorrSAN's training scheme to HCCNet.

Ablation study and analyses on SPair-71k. We conduct the same set of ablative and analytical experiments in the main paper, but on the SPair-71K dataset instead of PF-PASCAL dataset.

Table 4 shows that our choice of using the intermediate features extracted from `conv3_x` to `conv5_x` yields the best results on the SPair-71k dataset as well, which is consistent with the results on the PF-PASCAL dataset (Table 4 of the main paper).

Table 5 shows that our choice of using a slice size of 256 strikes the best balance between performance and efficiency, which is also consistent with the results on PF-PASCAL from the main paper (Table 5 of the main paper).

`conv` used 2_x	3_x	4_x	5_x	SPair-71k @α_{bbox} 0.05	SPair-71k @α_{bbox} 0.1	mem. (GB)	FLOPs (G)
×	×	×	✓	26.8	48.3	1.9	0.9
×	×	✓	×	32.9	51.1	1.9	1.3
×	×	✓	✓	**35.8**	54.3	1.9	1.6
×	✓	✓	✓	**35.8**	**54.8**	2.0	1.7
✓	✓	✓	✓	34.3	53.4	2.0	1.7

Table 4. **Ablation study on the backbone bottleneck features used.** The results show that our current setting of using `conv3_x` to `conv5_x` yields the best results.

Slice size	SPair-71k @α_{bbox} 0.05	SPair-71k @α_{bbox} 0.1	time (*ms*)	mem. (GB)	FLOPs (G)
-	34.4	53.9	20	2.0	0.9
512	35.3	54.5	24	2.0	1.1
256 (Ours)	**35.8**	**54.8**	30	2.0	1.7
128	34.7	52.9	43	2.2	4.0
64	35.1	52.8	70	2.2	13.3
32	33.3	52.5	127	2.7	50.7

Table 5. **Ablation study on the slice size used on the SPair-71k dataset.** The results show that our current setting of using the chunk size of 256 yields the best trade-off between performance and efficiency.

Table 6 shows that our choice of Tanh yields the best results on the SPair-71k dataset compared to using ReLU or Sigmoid, which is consistent with the results on the PF-PASCAL dataset (Table 6 of the main paper).

Results when using HPF-selected layers for hypercolumn correlation construction. HPF [9] propose to represent images using hyperpixels that leverage a small number

Activation function	SPair-71k @α_{bbox} 0.05	0.1
ReLU	**35.8**	54.4
Sigmoid	35.4	54.0
Tanh	**35.8**	**54.8**

Table 6. **Ablation study on the non-linear activation function used.** Using the Tanh activation function yields the best results, over ReLU or Sigmoid activation functions.

Method	SPair-71k @α_{bbox} 0.05	0.1	PF-PASCAL @α_{img} 0.05	0.1
HCCNet $_{\text{HPF}}$	31.7	49.7	76.3	90.8
HCCNet (ours)	**35.8**	**54.8**	**80.2**	**92.4**

Table 7. **PCK results on SPair-71k and PF-PASCAL when using bottleneck layers selected by HPF only vs. ours.** The results show that using all bottleneck layers from `conv3_x` to `conv5_x` yields significantly better results compared to using HPF [9]-selected bottleneck layers for each dataset.

of relevant features selected among early to late layers of the backbone feature extractor. Table 7 shows that using all bottleneck layers from `conv3_x` to `conv5_x` yields significantly better results compared to using HPF [9]-selected bottleneck layers for each dataset, substantiating our design choice over using HPF-selected bottleneck layers.

Additional feature slicing analysis on SPair-71k. To further investigate the impact of channel aggregation on the hypercolumn correlation, we visualize learned weight matrices $\mathbf{W}_{\text{hid}}$ and $\mathbf{W}_{\text{out}}$ with four different groups denoted by $G \in \{30, 62, 124, 248\}$ in Fig. 3 when trained on the SPair-71k dataset, as opposed to Figure 4 of the main paper which illustrates the analysis of feature slicing when trained on the PF-PASCAL dataset. We consistently observe that the weight magnitudes are significantly higher (in yellow) at deeper layers, particularly at `conv4_x` and `conv5_x`. As we increase the number of groups utilized for feature slicing (*i.e.* decrease the feature slice size), we find that the network carries out *fine-grained channel selection*, as evidenced by the weight visualization of $\mathbf{W}_{\text{hid}}$, verifying the efficacy of performing position-wise channel aggregation on hypercolumn correlation using diverse visual cues. Compared to the weight magnitudes of $\mathbf{W}_{\text{hid}}$ that are focused on specific groups, those of $\mathbf{W}_{\text{out}}$ are relatively evenly dispersed in order to effectively aggregate the information from the first channel aggregation to provide a reliable refined correlation map. The observations are overall consistent across the two datasets, demonstrating the efficacy of HCCNet regardless of the dataset it is trained on.

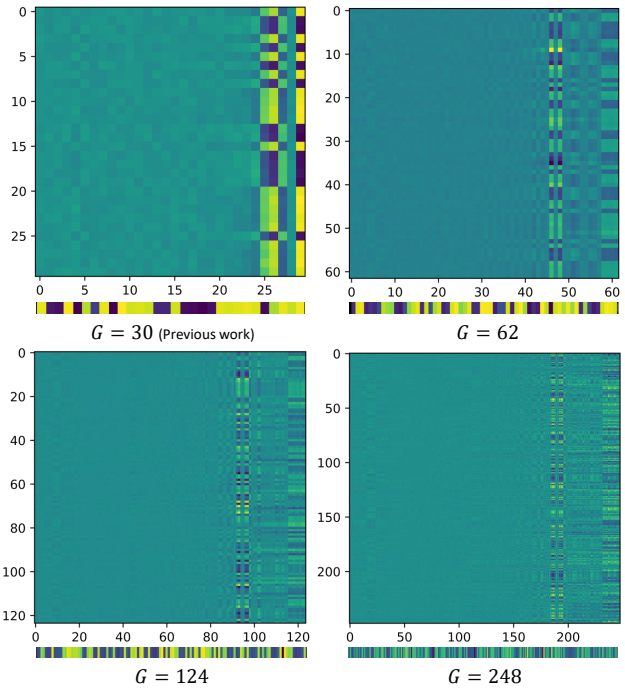

Figure 3. Visualization of learned weight matrices of $\mathbf{W}_{\text{hid}} \in \mathbb{R}^{G \times D_{\text{hid}}}$ (top) and $\mathbf{W}_{\text{out}} \in \mathbb{R}^{D_{\text{hid}} \times 1}$ (bottom) under varying $G = D_{\text{hid}} \in \{30, 62, 124, 248\}$. Trained on SPair-71k.

Variance of multiple runs.

Slice size	PF-PASCAL @ α_{img} 0.05	0.1
512	$77.8_{\pm 1.16}$	$91.9_{\pm 0.18}$
256 (Ours)	$80.0_{\pm 0.49}$	$92.2_{\pm 0.27}$
128	$79.6_{\pm 0.64}$	$92.0_{\pm 0.21}$

Table 8. **Results of HCCNet with varying slice size over multiple runs.** Using the slice size of 256 yields consistently better results over other sizes.

In Table 8, we report the PCK variance of our model over three runs with different random seed, while varying the slice size for our feature slicing scheme. The slice size of 256 still shows to be optimal, further verifying our design choice of HCCNet.

3. Additional qualitative results and analysis.

In Figure 4 we qualitatively compare HCCNet with TransforMatcher [6] on the SPair-71k dataset, where it can be seen that HCCNet established more reliable and accurate correspondences in comparison. Also, we further compare HCCNet with TransforMatcher on image pairs with larger variations in viewpoint/occlusion/truncation from the SPair-71k dataset, where HCCNet shows stronger robustness and

reliability. Note that we reproduced TransforMatcher using their released code to obtain these qualitative results. To enhance the visibility of the qualitative results for better comparison, the source images are TPS-transformed [2] to the target images using the predicted correspondences to align the common instances in each image pair.

Figure 4. **Qualitative comparison of HCCNet against TransforMatcher** [6]. Green lines represent ground truth correspondences, and blue lines represent predicted correspondences. The source images are TPS warped to the target image using the predicted correspondences for better comparison and visibility. Best viewed in electronics.

Figure 5. **Qualitative comparison of HCCNet against TransforMatcher [6] under larger viewpoint/occlusion/truncation variations.** Green lines represent ground truth correspondences, and blue lines represent predicted correspondences. The source images are TPS warped to the target image using the predicted correspondences for better comparison and visibility. Best viewed in electronics.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[2] Gianluca Donato and Serge Belongie. Approximate thin plate spline mappings. In *ECCV*, 2002. 5

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[4] Sunghwan Hong, Seokju Cho, Jisu Nam, Stephen Lin, and Seungryong Kim. Cost aggregation with 4d convolutional swin transformer for few-shot segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2022. 2, 3

[5] Shuaiyi Huang, Luyu Yang, Bo He, Songyang Zhang, Xuming He, and Abhinav Shrivastava. Learning semantic correspondence with sparse annotations. In *Proceedings of the European Conference on Computer Vision(ECCV)*, 2022. 2, 3

[6] Seungwook Kim, Juhong Min, and Minsu Cho. Transformatcher: Match-to-match attention for semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8707, 2022. 1, 4, 6, 7

[7] Junghyup Lee, Dohyung Kim, Jean Ponce, and Bumsub Ham. Sfnet: Learning object-aware semantic correspondence. In *CVPR*, 2019. 1

[8] Juhong Min and Minsu Cho. Convolutional hough matching networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2940–2950, June 2021. 1

[9] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Hyperpixel flow: Semantic correspondence with multi-layer neural features. In *ICCV*, 2019. 3, 4