Supplementary Material

1. Ablation Studies On Data Selection

In this section, we analyze the performance of our MSCN under different data selection strategy, *i.e.*, GMM and BMM. The experiments are conducted on the Flickr30K with 20% noisy correspondence. The results in Tab.1 demonstrate that our MSCN is effective to the choice of data selection strategy. And our proposed meta-data guided method brings the best results.

Table 1. Ablation studies on Flickr30K with 20% noise rate.

	Image to Text			
Methods	R@1	R@5	R@10	SUM
GMM	76.3	94.5	97.1	267.9
BMM	76.8	94.7	96.8	268.3
Ours	**77.4**	**94.9**	**97.6**	**269.9**
	Text to Image			
GMM	58.7	82.8	89.0	230.5
BMM	59.3	83.0	89.2	231.5
Ours	**59.6**	**83.2**	**89.2**	**232.0**

2. Experiments on MS-COCO 5K

In this section, we evaluate our trained MSCN on the full 5K test set with different noise ratio, *i.e.*, 20%, 50% and 70%. The results are shown in Tab.2

Table 2. Performance of MSCN on MS-COCO full 5K test set with 20%, 50% and 70% noise ratio.

	Image to Text		
Noisy Ratio	R@1	R@5	R@10
20%	57.1	84.0	90.8
50%	54.3	82.2	88.7
70%	51.4	80.3	87.6
	Text to Image		
20%	41.0	69.8	80.3
50%	38.4	67.4	78.5
70%	36.5	65.6	77.0

3. Generating of Synthetic Noise

To evaluate our method on a range of noise ratios, we generate the synthetic noisy correspondence data from Flickr30K and MS-COCO. Specifically, we randomly generate a mismatched index list to construct noisy pairs. Here, we provide a pseudo-code to describe the generating of synthetic noisy data:

```python
def generate_noisy_correspondence(images, captions, data_length, noise_ratio):
    '''
    images: original image data
    captions: original text data corresponding to images
    data_length: the length of data
    noise_ratio: the ratio of produced synthetic noise
    '''
    t2i_index = np.arange(0, data_length)
    #random produce mismatched idxs
    idx = np.arange(data_length)
    np.random.shuffle(idx)
    noise_length = int(noise_ratio * data_length)
    shuffle_index = t2i_index[idx[:noise_length]]
    np.random.shuffle(shuffle_index)
    t2i_index[idx[:noise_length]] = shuffle_index
    #fixing captions, and using the mismatched idxs to get images
    images = images[t2i_index]
    captions = captions
    return images, captions
```

4. Training Algorithm of MSCN

Algorithm1 summarizes our proposed MSCN.

Algorithm 1: The MSCN Training Algorithm

Input: Training set $\mathcal{D}_{train}$, meta-data set $\mathcal{D}_{meta}$, models $\mathcal{M}^{(1)} = \{\mathcal{F}_W^{(1)}, \mathcal{V}_\Theta^{(1)}\}$ and $\mathcal{M}^{(2)} = \{\mathcal{F}_W^{(2)}, \mathcal{V}_\Theta^{(2)}\}$, batch size n and m, learning rate α and β.

$\mathcal{M}^{(1)}, \mathcal{M}^{(2)} \leftarrow$ WarmUp($\mathcal{D}_{train}, \mathcal{M}^{(1)}, \mathcal{M}^{(2)}$)
while $e <$ *MaxEpoch* **do**
    Construct negative meta-data and extend the meta-data set as $\mathcal{D}'_{meta}$.
    $\{\mathcal{S}_p^{(1)}, \mathcal{S}_N^{(1)}\} \leftarrow$ GetSimilarityScore($\mathcal{D}'_{meta}, \mathcal{M}^{(1)}$).
    $\{\mathcal{S}_p^{(2)}, \mathcal{S}_N^{(2)}\} \leftarrow$ GetSimilarityScore($\mathcal{D}'_{meta}, \mathcal{M}^{(2)}$).
    Initialize $BMM^{(1)}$ using $\{\mathcal{S}_p^{(1)}, \mathcal{S}_N^{(1)}\}$.
    Initialize $BMM^{(2)}$ using $\{\mathcal{S}_p^{(2)}, \mathcal{S}_N^{(2)}\}$.
    $P^{(2)} = \{p_i\}_{i=1}^N \leftarrow BMM^{(1)}(\mathcal{D}_{train}, \mathcal{M}^{(1)})$.
    $P^{(1)} = \{p_i\}_{i=1}^N \leftarrow BMM^{(2)}(\mathcal{D}_{train}, \mathcal{M}^{(2)})$.
    for $K = 1, 2$ **do**
        $\mathcal{D}'^{(K)}_{train} = \{(I_i, T_i) | \, p_i > 0.5, \forall (I_i, T_i, p_i) \in (\mathcal{D}_{train}, P^{(K)})\}$.
        while $t <$ *MaxIteration* **do**
            From $\mathcal{D}'^{(K)}_{train}$ sample a training mini-batch $\{(I_i, T_i)\}_{i=1}^n$.
            From $\mathcal{D}'_{meta}$ sample a meta mini-batch $\{(I_i, T_i, y_i)\}_{i=1}^m$.
            Compute the updated parameters for $\mathcal{M}^{(K)}$ with training batch:
              $\hat{W}^{(t)}(\Theta) = W^{(t)} - \alpha \sum_{i=1}^n \nabla_W l^{train}(I_i, T_i)\big|_{W^{(t)}}$.
            Update the meta-net $\mathcal{V}_\Theta^{(K)}$ with meta batch:
              $\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{m} \sum_{i=1}^m \nabla_\Theta l^{meta}(I_i, T_i, y_i)\big|_{\Theta^{(t)}}$.
            Update the main net $\mathcal{F}_W^{(K)}$ parameters with training batch:
              $W^{(t+1)} = W^{(t)} - \alpha \sum_{i=1}^n \nabla_W l^{train}(I_i, T_i)\big|_{W^{(t)}}$.
        end
    end
end