An End-to-end Deep Convolutional Neural Network for a Multi-scale Image Matching and Localization Problem

Sungsoo Ha, Yuewei Lin, Xiaojing Huang, Hanfei Yan, Wei Xu
Brookhaven National Lab
Upton, NY, USA
{sungsooha,ywlin,xjhuang,hyan,xuw}@bnl.gov

Abstract

Diverse imaging techniques are utilized in many scientific domains to acquire a rich description of the subject under study and to further discover a variety of its properties. Especially, in sample systems, by probing with optical, electron or x-ray beams, the captured images describe the sample in an extremely large range of length scales. This makes the correlation from one image to another very difficult in addition to the intrinsic appearance complexity of those scientific images. In this paper, we aim to tackle this multi-scale image matching and localization problem by proposing an end-to-end deep convolutional neural network. Our proposed network is designed to first generate different filters according to the two queried images originated from different length scales. Then, to compute the correlation map, we use these filters to predict the correspondence between the two images. For the training and evaluation, we collect a number of electron microscopy experiments to form a multi-scaled image patch dataset comprised of various material structures. We observe about 90% accuracy for multi-scale image matching and localization while a triplet-based network shows about 78% accuracy.

1. Introduction

In a wide variety of sample systems, the cross-modal and multi-length scale imaging approach utilizing complementary imaging techniques is of great interest in solving scientific problems. Images obtained from different microscopy methods reveal different aspects of the sample's elemental, chemical or physical properties. These integrated data can provide a comprehensive view of the sample under study across multiple magnitude of length scales with various contrast mechanisms. This approach is expected to make significant impacts in many fields, where heterogeneity and complexity play critical roles, for instance, material science, medical imaging, biology and geographical science. Thus,

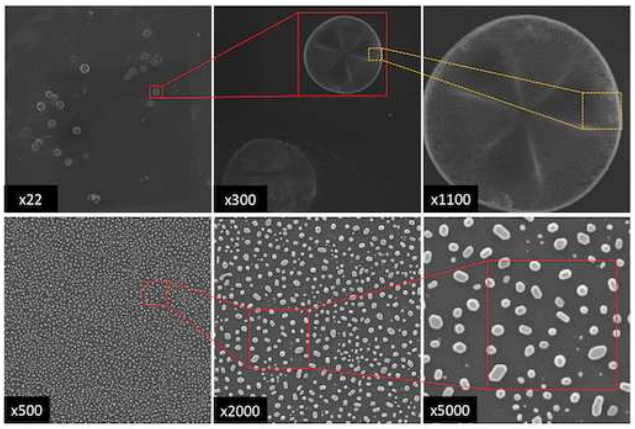

Figure 1. Example SEM images in different zoom factors.

a robust and automatic image matching and localization approach is demanded to rescue the scientists from tedious ad hoc operations.

However, there are a few critical challenges associated with this mission. First, compared to natural images, these acquired images describe the samples in extremely small length scales from nanoscale to micron and thus present significantly abstract and obscure structural appearances to identify and compare. Second, considering these images are represented with a large range of length scales, various contrast mechanisms and diverse quality levels, thus, it is extremely challenging to robustly find a correlative match and integrate data even within the single modality setting. For example, for the electron microscopy imaging as shown in Figure 1, the length scale of the field of views can vary over several orders. Exhaustive search in a huge search space (at least, four dimensions for location and size) is impossible in practice. Without fiducial references, it is extremely difficult to correlate features imaged with very different magnifications.

Given the challenges mentioned above, we aim to find out the correspondence between two images potentially

containing the same object of interest that might be originated from the same or different scales, contrasts, and noise levels. More specifically, we aim to match and localize a target image in a source image, if it exists, by adopting a deep neural network (DNN). Different from semantic retrieval tasks, our focus is to match exact instance (not category) of interests and localize it in source image. Although there are several existing works for either image matching or localization tasks by utilizing DNN, their objectives or tasks are not well aligned with ours. For example, using a DNN, multiple objects can be identified and localized with bounding boxes or segmented for more precise boundaries [16, 13, 19, 7]. However, these works detect objects based on their categories without considering which one is exact instance of interests. Image feature descriptors from object detection-oriented DNNs, Siamese-based DNNs and Triplet-based DNNs have been actively adapted not only for comparing two image patches [6, 11, 1, 21, 15, 25, 27] but also for developing instance-based image retrieval algorithms [20, 4], where the-state-of-the-art performance with natural images is observed. These works aim to match the same objects (or instances) in two images/patches, usually by forming the task as a classification problem, i.e., given two images/patches, deciding whether they belong to the same object (instance). However, there is no localization part to match the exact object bounding boxes.

To tackle the matching and localization problem jointly, in this work, we propose a new DNN named *FTFY (Find That For You)*, which matches two input images in multi-scale imaging configurations and localizes one to the other simultaneously. Our main contributions are as follows:

- An object detection/localization DNN and an image matching DNN are combined into a single network architecture that is trained in an end-to-end manner;
- A hyper-network structure [5] is employed to compute correlation between two images and achieve simultaneous image matching and localization with a multi-scale image dataset;
- We establish a multi-scale image dataset that is designed to train multi-scale image matching and localization not only for the proposed FTFY network but also for a triplet-based network with IOU-based sampling scheme.

The reminder of our paper is structured as follows: Section 2 explains how the FTFY network is differentiated from other related networks. Section 3 and Section 4 present the details of FTFY network architecture and training strategies, respectively. Section 5 presents how we build multi-scaled image patch dataset and how it is sampled to train the FTFY and a triplet-based networks and Section 6 shows the experimental results with this dataset. Finally, Section 7 concludes the paper with future works.

2. Related Work

Deep neural networks (DNNs) have been shown a big success in object classification task. Having repeated convolution operations in-between max-pooling operations allows DNNs to learn richer features at every spatial scale [10, 26, 22]. Introducing residual connections to DNNs that are getting increasingly deeper makes it possible to train them robustly and to achieve the-state-of-the-art performance [8, 24], not only to classify an image but also to figure out what and where objects are in the image [16, 3, 19, 13, 7]. Specifically, after taking an input image, these networks propose regions that might contain some objects belonging to pre-defined classes, classify the objects in each proposed regions, and finally localize the objects with bounding boxes or segment them to provide more precise boundaries. In YOLO network [16], the proposed regions are defined as grid cells of an input image feature map and a single object is detected from each grid cell; while faster R-CNN [19] uses the notion of anchor boxes to more robustly propose regions that might contain one or more objects. While these two networks are focused on single-scale object detection, SSD network [13] detects objects in multi-scale by making prediction on multi-scale feature maps. Recently, YOLO network is extended to support the advantages of these two networks to further boost its performance [17, 18]. Mask R-CNN [7] extends faster R-CNN [19] by adding a branch for predicting an object mask in parallel with the existing branch for bounding box prediction. Although there have been more advanced networks proposed by many researchers for simultaneous object detection and segmentation (SDS), there is one important thing that makes our proposed network different from these. In the proposed network, the object to find and localize in the input image (referred as *source image* in this paper) is given as a second input image (referred as *target image*), and their correlation is computed within the proposed network. Therefore, the target image is usually a focused description of the object while the source image captures the same object in a larger field of view.

To compute the correlation between two images, we adopt the Siamese neural network architecture [2]. The Siamese network extracts features separately from two compared inputs with two identical sub-networks (*i.e.* shared parameters and weights). Using the outputs of the two sub-networks, two image patches can be compared if they are matched as a binary classification problem [6, 11, 1]. Alternatively, the network can be trained with Euclidean distance-based loss function so that the output features can be used as image descriptors to retrieve similar image patches with Euclidean distance [21, 15, 25]. These networks outperform the image matching performance compared to traditional hand-crafted image feature descriptors like SIFT [14]. In the sense of computing correlation be-

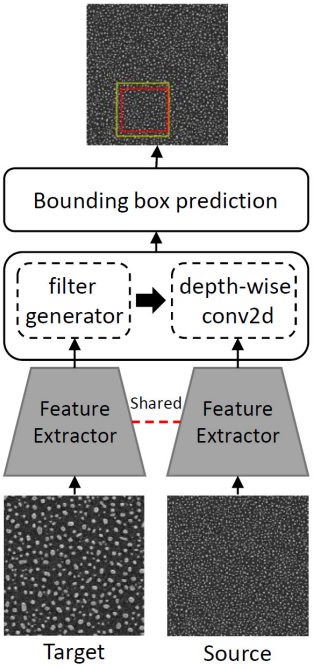

Figure 2. The overview of the FTFY network with three major components: the feature extraction, the filter generator for correlation map, and the bounding box prediction.

tween two input images, our proposed network is similar to these networks but with a major difference in how the two output features are further utilized in the following network architecture. Specifically, the feature of the source image is fed to *a convolution layer* to predict if the source image contains the target image or a similar one, and to infer its location if any, which is the same as the above discussed networks. Different than the source image, the output feature of the target image is fed to another sub-network to generate parameters of *the convolution layer* by borrowing the concept of hyper-network [5], referred as a filter generator in our proposed network.

3. Network architecture

The FTFY network consists of three major parts. First, in the feature extraction part, it extracts valuable features for computing the correlation between two images (Section 3.1). Then, features from a target image are passed to filter generator to generate filters and convolve them with the source image features to obtain a correlation map between the two images (Section 3.2). Finally, the correlation map is used to predict the matched regions as rectangle-shaped bounding boxes (Section 3.3). Figure 2 illustrates the overview of the FTFY network. In the following subsections, we will explain the detailed network design.

3.1. Feature extraction

The feature extraction is adapted from the YOLO network [16]. More specifically, the feature extraction part consists of 13 convolution layers where each convolution layer is followed a batch normalization layer [9] and a non-linear activation layer. Four average pooling layers are added after 2-nd, 5-th, 8-th and 11-th convolution layers. To recover some information losses as the network goes deeper and to have more finely grained features, features from 5-th, 8-th, and 11-th are concatenated by stacking adjacent pixels into different channels, and it is fed to the 12-th convolution layer. It is worth noting that the network for the feature extraction is shared between two input images, a source and a target, of the FTFY network.

3.2. Correlation between two images

We compute the correlation between a source and a target images by convolving the feature map, S, of the source image with the filters, F, generated from the feature map of the target image. It is assumed that we have N_f of filters generated from the target feature map and each filter has size of $f_h \times f_w \times f_d$ where f_h, f_w and f_d represent height, width, and depth of the filter, respectively. Also, we assume that S_i is a feature vector at each spatial location, i, F_j is the j-th filter ($j \in [1, \ldots, N_f]$), and both have the same length. Then, convolving S_i and F_j is equivalent to computing cosine distance between both vectors. As there are N_f different filters, it will result in N_f different distance metrics at each spatial location, i. We interpret the distance vector length of N_f as a local correlation vector and the whole set of the correlation vectors as a correlation map between the source and the target images.

We employ the idea of hyper-network[5] to generate the N_f filters from a feature map size of $H_{fm} \times W_{fm} \times D_{fm}$ where H_{fm}, W_{fm} and D_{fm} are height, width, and the number of channels of the feature map. Afterwards, for each channel image, we first transform the image feature space to another latent space that would be more suitable for the filter generation. This is completed by applying a sequence of convolution, batch normalization and non-linear activation. Here, we use D_{fm} number of filters with $H_{fm} \times W_{fm}$ filter size for the convolution operation. This is, in fact, equivalent to using fully-connected layer but we avoid it for the computational efficiency. Then, the feature map in the latent space has the size of $D_{fm} \times 1 \times 1 \times D_{fm}$ as we applied the projection for each channel. Lastly, we apply another convolution layer (1×1 filter size) to output N_f channels followed by batch normalization and non-linear activation layers. After reshaping the output, we have the generated filter tensor size of $N_f \times f_h \times f_w \times f_d$ where f_h, f_w, and f_d are 1, 1 and D_{fm}, respectively. For example, with 128×128 input image, the feature map is $8 \times 8 \times 512$, the latent variables are $512 \times 1 \times 1 \times 512$, and finally the generated filter

tensor becomes $N_f \times 1 \times 1 \times 512$ where N_f is set to 256 in our implementation.

3.3. Bounding box prediction

By using the correlation map in Section 3.2, the matched area of the target image to the source image is predicted with rectangle-shaped bounding boxes. Note that the correlation map is a set of local correlation vectors where each vector represents correlation between a local region of the source image and the full target image. To consider a much wider region for each correlation vector, we first need to propagate the local correlation into their adjacent locations. We implement this by processing the correlation map with 6 convolution layers where each convolution layer is followed a batch normalization and non-linear activation layers.

With the processed correlation map, we predict B number of bounding boxes and its confidence score, C, at each spatial location as in [16]. The confidence score, C, implies how confident the network is with the predicted bounding box and it reflects Intersection-Over-Union (IOU) between predicted bounding boxes and ground truths. In this work, a bounding box is represented by four parameters, x_c, y_c, w, and h. Here, x_c and y_c are the central location of a bounding box with respect to each cell (*i.e*. normalized by a cell size). w and h denote width and height of the bounding box with respect to full image size (*i.e*. normalized by an image size). In our implementation, we pass the processed correlation map into a convolution layer (1×1 filter size) and then a sigmoid activation layer.

4. Training

We have trained the proposed FTFY network with the multi-part root mean square (RMS) losses from the YOLO [16] after initializing the feature extractor part with the weights from a triplet network equipped with the global orthogonal regularization (GOR) [27].

For the triplet network, we add a fully connected layer that outputs a feature vector length of 128 and l2-normalization layer at the end of the feature extractor part. To train the triplet network, we randomly sample $1M$ triplets out of multi-scaled image patch dataset (See Section 5.1). The training batch size is set to 64. We use SGD with momentum in the optimization. The learning rate starts at 0.01, with momentum 0.9. The learning rate is reduced after $10,000$ gradient updates by a factor of 0.96. We use the default setting for the parameters in the triplet loss as described in [27]. The triplet network is trained with 25 epochs and it converges before the end of training.

To train the FTFY network, we use mini-batch gradient descent algorithm to minimize the multi-part RMS losses. We have created $74,020$ target image patches where each patch is down-sampled by a factor of $k \in \{1, 2, 4, 6, 8, 10\}$ from the original image patch. Each target patch is associated with at least 2 to at most 6 multi-scaled source images where their down factors are also in $\{1, 2, 4, 6, 8, 10\}$. The details on the target-source image pairs for the FTFY network are explained in the Section 5.2. During the training, we randomly pick one of source images available for each target image. The training batch size is set to 8. The learning rate starts at 0.01 and reduced after $100,000$ gradient updates by a factor of 0.96. We use the default setting for the parameters in the multi-part RMS losses as described in [16]. The FTFY network is trained with 100 epochs and it converges around 60 epochs.

Finally, to avoid over-fitting, we add dropout layers [23] at the end of feature extractor, filter generator, and bounding box predictor, along with random source image selection in different scales. We use leaky rectified linear activation function for the all non-linear activation layers, if it is not specified, in the both networks.

5. Multi-scaled image patch dataset

In this section, we present the details of the multi-scaled image patch dataset used to train the triplet and the FTFY networks. The image dataset used to build the image patch dataset is acquired by a bench-top scanning electron microscope (SEM). There are 222 SEM images obtained by scanning 24 different kinds of samples. Each image has the dimensions of 850×1280; while the imaging pixel size is varying from 20 nm to 200 nm.

The first step in constructing multi-scale image patch dataset with the SEM images is to select key points from each SEM images. Different types of samples look very different and have different regions (or objects) of interests for the scientists. Moreover, the same objects captured in different scales have very different pixel dimensions. To consider these variations, we manually draw square-shaped bounding boxes that cover the interest regions (or objects) for each SEM image. Note that we set the minimum side length of the square as 15 pixels.

Given a square-shaped bounding box and an associated SEM image, the region is down-sampled by factors of $\{1, 2, 4, 6, 8, 10\}$ and resized to have 128×128 dimensions. We randomly shift the region to have Intersection-Over-Union (IOU) in ranges of $[0.7, 1.0)$, $[0.5, 0.7)$, and $[0.3, 0.5)$, and collect multi-scaled image patches from each IOU ranges. This random IOU region selection is repeated four times. At the last step, we manually inspect each down-sampled image patches and discard some of them that is visually unrecognizable. We call this collected multi-scaled patches from a bounding box a *patch group* and Figure 3 shows one example of it.

Finally, the multi-scale image patch dataset contains $3,906$ patch groups ($106,010$ patches) out of 222 SEM images in total. The $2,734$ patch groups are used to generate

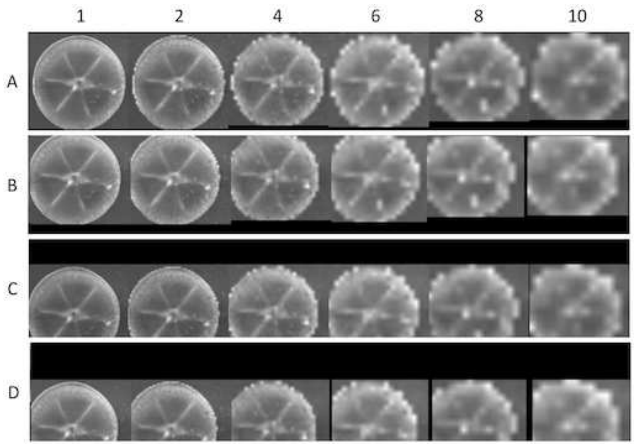

Figure 3. An example of a patch group. The numbers in x-direction represents the down sampling factor from a patch image resized to 128×128. The letters in y-direction represents (A) key position and randomly shifted position from the key position based on IOU range: (B) $[0.7, 1.0)$, (C) $[0.5, 0.7)$ and (D) $[0.3, 0.5)$.

triplet examples (See Section 5.1) and target-source image pairs (See Section 5.2), and the others are reserved to evaluate the trained network performance.

5.1. Triplet sampling

For the brevity, given a patch group, let's define set A, B, C and D as multi-scaled patches extracted from a selected bounding box and multi-scaled patches in the IOU ranges of $[0.7, 1.0)$, $[0.5, 0.7)$, and $[0.3, 0.5)$, respectively, as shown in Figure 3. We first randomly choose an anchor from set A. There are three different ways according to how positive and negative examples are chosen. First, we seek for them within a patch group such that positive is randomly chosen from A, B or C; negative is randomly chosen from a set that is originated from lower IOU range than the positive one. Second, the positive is randomly chosen from set A or B; the negative is randomly chosen from set A' or B' of different patch group but in the same SEM image. Lastly, while choosing the positive as the same manner in the second method, the negative is randomly chosen from set A" or B" of different patch group in the different kinds of SEM images. In a triplet example, there are no duplicated patches.

Note that the second method will always generate valid triplet examples because we intentionally select patch groups not to overlaps to each other within a SEM image. Also, in the third method, as a patch might match with another sampled from the same type of SEM images but in different scale, we choose the negative from different types of SEM images. We sample $1M$ triplet examples over $2,734$ training patch groups uniformly using the three sampling methods.

5.2. Target-source image pair sampling

To generate a source image for a patch group, we first compute the rectangle window, R, that encompasses all patches in the SEM image where the patch group is originated. Then, the source image is randomly cropped from the SEM image to contain the R while its size is three times larger than the R. Note that we adjust the cropped region to be in the SEM image. Multi-scaled source images are generated by down-sampling the cropped region with the factors used for the patch group and by resizing them to 256×256 dimensions. Finally, we calculate true bounding box coordinates for each target patch in the patch group with respect to each down-sampled source image. For this, we have recorded the square-shaped bounding box coordinates and selected down-sampling factors for all patches when we have built the multi-scaled image patch dataset. Note that although we have collected patches with square-shaped bounding boxes for the multi-scaled image patch dataset, the true bounding boxes in target-source image pairs are mostly rectangle-shaped because of the two facts: (1) R is likely rectangle-shaped due to the random selection scheme in different IOU ranges and (2) the cropped region is shrunk/expanded with different amount for x- and y-direction to have 256×256 dimensions.

For the multi-scaled SEM image patch dataset, we have generated $10,601$ source image patches out of $3,906$ patch groups. During the FTFY network training, we only used source image patches associated with patch groups reserved for the training.

6. Results

To evaluate the proposed FTFY network and compare with the triplet network trained for multi-scaled image patch matching, we define an image retrieval task as follows: given an image patch as a query, whether a matched (rectangle-shaped) region for the FTFY network or the most similar patch for the triplet network can be retrieved from an entire image where the query is originated when the image is in lower resolution than the query.

More specifically, for the FTFY network, we said it is successful if it can retrieve a bounding box that has larger than 0.7 IOU with respect to the true bounding box among the top-K bounding boxes based on the predicted confidence scores. On the other hand, for the triplet network, we first sample image patches with overlaps from the entire image and collect feature vectors of the patches and the query using the triplet network. We also compute the top-K expected patches to be retrieved for the query based on the IOU. Then, we retrieve top-K patches based on the euclidean distance metric over the feature vectors and we said it is success if one of the top-K retrieved patches is in the top-K expected patches.

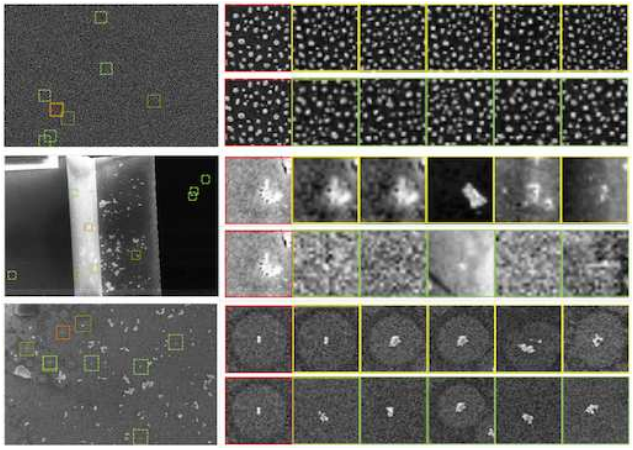

Figure 4. Examples of success on the FTFY (failure on the triplet).

For the image retrieval test, we use the patches in the reserved $1,172$ patch groups in the multi-scaled image patch dataset in Section 5. Note that we have recorded bounding box coordinates and down sampling factors for all patches. Using those information, we crop query images from the corresponding original SEM images without down sampling. Then, the original SEM images are down-sampled by the recorded factors and use them as the source images for the FTFY network and to build a feature vector database for the triplet network.

The FTFY network shows better retrieval accuracy than the triplet network as 0.897 versus 0.780 when $K = 1$ and 0.941 versus 0.902 when $K = 5$, as summarized in Table 1. It is important to note that the queried image patch sizes are all different (*i.e.* no square shape). For the triplet network, it is necessary to know the image patch size in advance to build a proper feature vector database given an entire image to be searched while it is not necessary for the FTFY.

Model	@$K = 1$	@$K = 5$	per query (sec)
Triplet	0.780	0.902	6.0
FTFY	0.897	0.941	2.6

Table 1. Image retrieval accuracy.

When it comes to the time performance, the triplet network is required to densely extract image patches from the entire image to properly search the image and to find the most similar patch to the query image. For example, given 1280×850 image to be searched and 60×60 query image, it needs to extract about $4,346$ image patches (60×60 patch size with 15×15 strides) and process over those patches. On the other hand, this is relaxed for the FTFY network such that, for example, the entire image is divided into about 40 small images (270×270 block size with 135×135 strides, about 4.5 timer larger than query image size) and process on them as source input images. With NVIDA GTX 1080, the FTFY network shows about 2.3 times faster than the triplet network (2.6 vs 6.0 seconds per query on average) to process 1280×850 image to be searched and 60×60 query image on average in the test dataset.

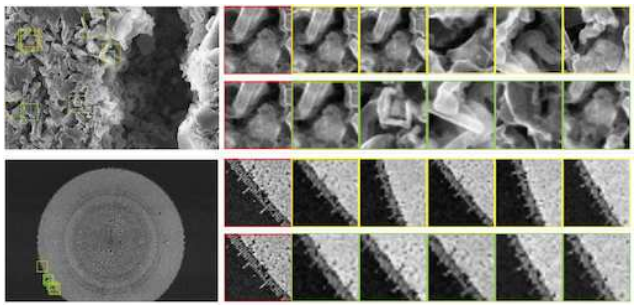

Figure 5. Examples of failure on the FTFY (success on the triplet).

Lastly, Figure 4 and Figure 5 show examples of success and failure cases for the FTFY network (reverse order for the triplet network). In those figures, top-5 retrieved results are drawn with bounding boxes over the entire searched images on the left (yellow for the FTFY and green for the triplet); and on the right it shows zoomed-in views for the retrieved results followed by the queried image (surrounded by red-colored boundary). Note that the retrieved results are in the order of the confidence score for the FTFY and of the euclidean distance for the triplet. Figure 4 shows that the FTFY network works well even over the images in complicated structures or extremely low contrast. Even for the failure cases shown in Figure 5, the FTFY network are able to localize the queried image with IOU close to 0.7.

7. Conclusion

In this paper, we presented a deep neural network to solve the multi-scale image matching and localization challenge originated in material science. More specifically, given two input images from different length scales, the FTFY network was able to match and localize one image to the other more accurately and efficiently than the state-of-the-art triplet-based network. Even for the failure cases, the FTFY network was able to localize the queried image with the IOU close to 0.7 while the triplet network missed the matched regions in its failure cases.

In the future, we would like to further extend the FTFY network to support (1) rotational invariance by allowing to predict rotated bounding boxes as in [12] and, more importantly, (2) cross image modality that is aiming to match and localize two images where both are not only from different scales but also from different image modalities.

Acknowledgment

This research was supported by two Lab Directed Research and Development projects 18-009 and 17-029 of Brookhaven National Laboratory. This research used resources in Hard X-ray Nanoprobe beamline of the National Synchrotron Light Source II, a U.S. Department of Energy (DOE) Office of Science User Facility operated for the DOE Office of Science by Brookhaven National Laboratory under Contract No. DE-SC0012704.

References

[1] H. Altwaijry, E. Trulls, J. Hays, P. Fua, and S. Belongie. Learning to match aerial images with deep attentive architectures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3547, 2016. 2

[2] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994. 2

[3] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[4] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *European Conference on Computer Vision*, pages 241–257. Springer, 2016. 2

[5] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2, 3

[6] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015. 2

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 2

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[11] B. Kumar, G. Carneiro, I. Reid, et al. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5385–5394, 2016. 2

[12] L. Liu, Z. Pan, and B. Lei. Learning a rotation invariant detector with rotatable bounding box. *arXiv preprint arXiv:1711.09405*, 2017. 6

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 2

[15] I. Melekhov, J. Kannala, and E. Rahtu. Image patch matching using convolutional descriptors with euclidean distance. In *Asian Conference on Computer Vision*, pages 638–653. Springer, 2016. 2

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2, 3, 4

[17] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017. 2

[18] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[19] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[20] A. Salvador, X. Giró-i Nieto, F. Marqués, and S. Satoh. Faster r-cnn features for instance search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–16, 2016. 2

[21] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126, 2015. 2

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 4

[24] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017. 2

[25] Y. Tian, B. Fan, F. Wu, et al. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Cvpr*, volume 1, page 6, 2017. 2

[26] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2

[27] X. Zhang, F. X. Yu, S. Kumar, and S.-F. Chang. Learning spread-out local feature descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4595–4603, 2017. 2, 4