USC ISI DiSPARITY Self-Evaluation Report

Contents

1 Introduction

This report summarizes the self-evaluation efforts of the University of Southern California's Information Sciences Institute (USC/ISI) DiSPARITY team. The report is organized as follows. Section 2 presents the technical approach and self-evaluation results for the provenance detection task, including provenance filtering and provenance graph construction, as well as evaluation results on the NIMBLE 2017 dataset. Section 3 presents our novel, end-to-end, deep learning approach to splicing detection and localization, and the evaluation results on a new datasets that we internally created to train and test the developed approach. Finally, Section 4 summarizes the work done.

2 Provenance Detection

2.1 Problem Definition and Challenges

Provenance detection consists of two main tasks – provenance filtering and provenance graph construction. Provenance filtering can be thought of as a generalization of content-based image retrieval (CBIR), in which a query (or probe) image is used to retrieve the top N visually similar images from a reference (or world) dataset. In CBIR, the query images is assumed not to be digitally manipulate. In provenance filtering, however, the query/probe image is likely to include one or more digital manipulations (e.g. additions or removal of objects or people). Therefore, in provenance filtering, the task is to retrieve the base image, which contributed the scene background, and all donor images, which contributed smaller parts of the query image (in case of object addition, for example). The set of base and donor images constitute the provenance of a given probe image. In the second provenance detection task, i.e. provenance graph building, a directed genealogy graph must be constructed from the provenance of the probe images to represent how the probe was created step by step using its provenance images.

Retrieving the base of a query image is a relatively standard CBIR task. However, one of the main challenges in provenance detection is that the probe image must be segmented into different regions that can serve as *query images* for retrieving donors. Furthermore, reference images must be searched for matching regions of the queries, rather than overall image-to-image matching, which significantly increases the computational complexity of provenance detection. For example, we implemented a vanilla retrieval system using Constitutional Neural Network (CNN) features extracted using Overfeat Library [20] and the Fisher vector-based feature embedding [18][19]. Although the performance of image retrieval using this method yields 77.9% mean average precision on the Holidays dataset, the donor detection performance using this method on the NIST NC2016 [1] Web data only yields 13.6% recall rate from top 10 retrieval results.

2.2 Technical Approach

Figure 1 shows the overall architecture of the developed provenance detection system. Our system consists of a feature representation stage, which samples sub-regions from reference and probe images and extracts learned representations for each sub-region using VGG19 CNN. For a given probe image, base and donor images are retrieved independently and combined to provide the provenance set of the probe image. In order to construct a genealogy graph of the probe, we use a minimum spanning tree (MST) of the top 50 matching images from the provenance set. In the following, we discuss

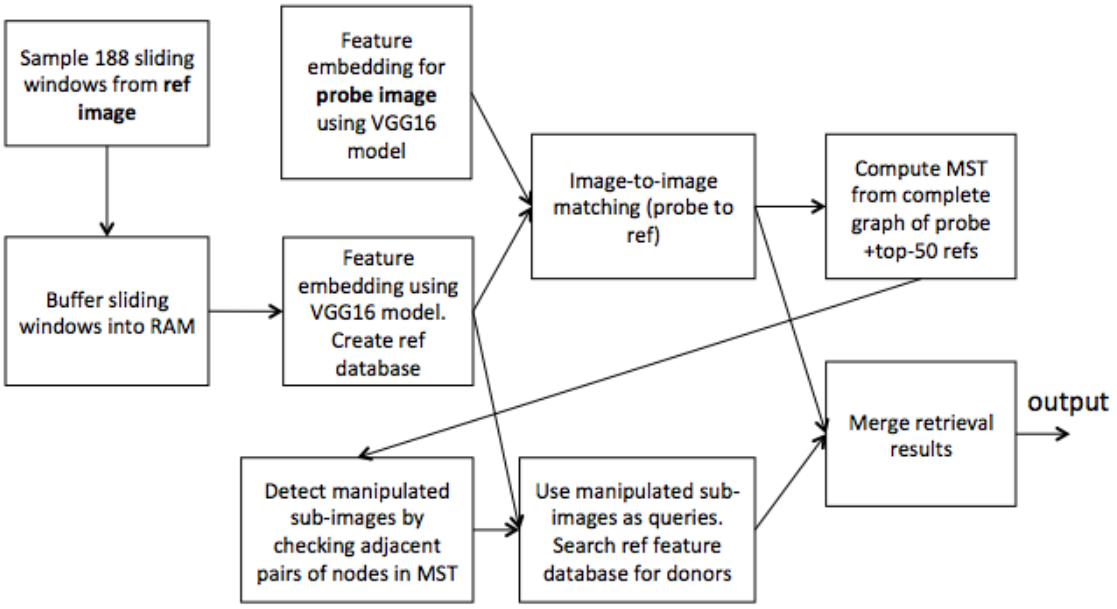


Figure 1: Provenance system diagram

each of these steps in more details.

Feature Representation

We use a *multi-scale* approach in which we exploit VGG19 CNN to extract learned representation of probe and reference images. Each image is processed at 6 different scales. At each scale, the image is divided evenly into multiple sub-images. The total number of sub-images for all 6 scales is 188. VGG is used to extract a 1000D feature vector for each scale and sub-images, creating a 188×1000 representation for each image.

Base Detection

It is important to clarify that a base image not only means the unaltered image that is used as a foundation for creating the tampered image, but also all intermediates images derived from the unaltered image. In the mean time, a donor image is not derived from the base. The donor image provides a source for an object (e.g. people, vehicles, etc.) to be spliced into the probe image. In this stage, we only detect all base images (unaltered and intermediate) that match the probe images

In order to retrieve the base image(s) of a given probe, we search the world set for reference images most similar to the probe image, by calculating the cosine similarity between each sub-image from the probe image and the corresponding sub-image in the reference image. The similarity between the probe image and the reference image is then calculated as the maximum of all these sub-image similarity scores. Only the top 3 scales are used in this step.

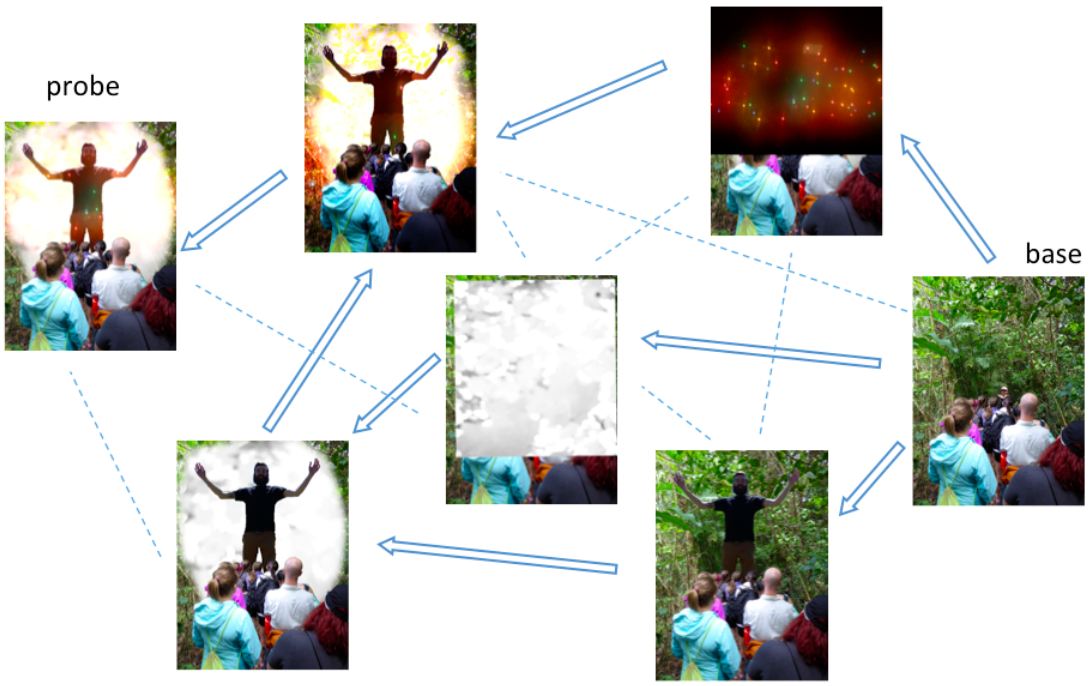


Figure 2: A sample genealogy graph of provenances

Donor detection

The major challenge in donor detection is finding objects spliced into base images during the course of manipulation and finally into the probe. Once these spliced objects are found, we can use them as queries, and search sub-regions in reference images for these queries.

Figure 2 illustrates our idea of finding spliced objects. First, we build complete graph using base images detected in the first round which is based on the base detection method mentioned above. We keep top N (we use $N = 50$ in the experimental evaluation) base images for each probe. The complete graph uses these 50 base images as nodes and pairwise similarity scores between them as edge values.

A minimum spanning tree is computed from this complete graph. Each pair of adjacent nodes from the minimum spanning tree are examined by subtracting one from another. The result image may have non-zero regions indicating manipulated regions in one of the two reference images. We take each manipulated region as a query, and search the world set for donors containing this query image. Since manipulations do not have to be splicing but can also be copy/move or some regional image enhancement, we only use a region of a base image as a donor query when the region is the only one found from that base. False alarms can be effectively filtered out using this strategy.

When we search reference images for donors, we compute the similarity between each donor query and each sub-region of a reference image. We still use VGG features and cosine similarity measurement. But all 6 scales, all 188 sub-images from each reference image, are considered in this step.

Provenance Set and Genealogy Graph

For a given probe image, we generate top-200 provenance set by top matching base and donor images based on the similarity scores. The procedure is optimized using NC2017 Dev3Beta1 dataset.

To generate the genealogy graph of a probe image, we start by creating the provenance set and the probe image, as discussed before. A minimum spanning tree is constructed using all pairs of images in the provenance set, where each image represents a node and the similarity scores represent the edge weights between the nodes. The graph is then reorganized such that the probe image is the root. Further, we prune the graph using the edge weights to produce only the top K nodes, where K is selected empirically using NC2017 Dev3Beta1 dataset.

2.3 Evaluation Data

The method was evaluated using a number of NIST NIMBLE dataset releases [1] [2]. The NC2016 Web dataset has 724 probes and a world set of 1124 reference images. Out of 724 probes, we found 128 that have spliced objects corresponding to 264 donor images in the world set of 1124 reference images. The NC2017Dev1Beta4 dataset has 65 probe images and a world set of 1631 reference images. The NC2017Dev3Beta1 dataset has 2260 probe images and a world set of 3441 reference images. The NIMBLE Evaluation world set has 1008681 reference images.

2.4 Evaluation Results

Segmentation-based donor detection:

We ran the first set of experiments on NC2016 Web to evaluate several donor detection strategies. Bases and donors were explicitly annotated in NC2016. Probes in NC2016 Web have very little modification in their background scenes. Thus, base detection was very accurate. We ran an image-to-image matching using 512d Fisher vectors embedding raw Overfeat features and got 100% recall rate from top 1 base detection results. We also used a Deep Metric Learning (DML) [27] to represent the image using a 128d feature vector. DML has several pre-trained models. First, we tried the model trained on the CARS dataset. We obtained manipulated regions for each probe by subtracting the probe from its base, and used these manipulated regions as queries to search the NC2016 world set for donor images. In contrast to the sampling approach described above, we tried the Berkeley Semantic Segmentation algorithm [24] to divide each reference image into semantically homogeneous regions, and computed a feature vector from each region (later, we will present experimental results showing the sampling method performs better than this segmentation method though.) For donor detection, we took the maximum of cosine similarities between donor queries of the probe and regions of the reference image as the similarity between the probe and the reference image. As a contrastive condition, we also measured recall rates of donors using image-to-image matching score (no selection of sub-images from the probe and the reference image). The recall rates of donors from top 10 results of these experiments are shown in Table 1. By comparing the first two rows of results, the DML features from the CARS model outperforms the Fisher vector method. By comparing the 2^{nd} and the 3^{rd} rows of results, the introduction of segmentation schemes for both probes and reference images significantly improved donor detection performance.

Table 1: Impact of the segmentation scheme on donor detection (NC2016).

Method	Top 10 recall rate
Fisher w/o segmentation	13.6%
DML (CARS) w/o segmentation	18.9%
DML (CARS) w/ semantic segmentation	31.4%

Fine-tuning Donor Detection

We made two attempts to further improve our core algorithm for donor detection. First, keeping Berkerly's semantic segmentation unchanged, we tried the pre-trained DML models using the online_products dataset and CUB dataset, respectively, and two Imagenet contest participating object detection feature extractors: Inception V3 [29] and VGG-19 [26]. The recall rates of donor detection from top 10 results are shown in Table 2. The best performance is obtained from using the VGG-19 model.

Table 2: Comparison of feature embedding models (NC2016).

Method	Top 10 recall rate
DML (CARS) w/ semantic segmentation	31.4%
DML (online_products) w/ semantic segmentation	45.1%
DML (CUB) w/ semantic segmentation	61.7%
Inception V3 w/ semantic segmentation	81.1%
VGG-19 w/ semantic segmentation	81.8%

The second attempt was to replace semantic segmentation with dense sampling of probe and reference images. As described before, we used 188 sub-images and extracted CNN learned representations from each sub-image. Table 3 indicates significant improvement of donor detection.

Table 3: Comparison of semantic segmentation vs. densely sampled sub-images (NC2016).

Method	Top 10 recall rate
DML (CUB) w/ semantic segmentation	61.7%
DML (CUB) w/ densely sampled sub-images	81.8%

Provenance Filtering Performance on NC2017 Data

We evaluated provenance filtering performance on both NC2017Dev1Beta4 and NC2017Dev3Beta1. The experiment on NC2017Dev1Beta4 used the probe set and world set from NC2017Dev1Beta4 only. We generated top 200 provenance filtering results by taking top 50 base detection results and additional top 150 donor detection results. The performance is shown in Table 4.

Table 4: Provenance filtering results on NC2017Dev1Beta4.

Result	Recall rate
Top 50 base detection results	78.1%
Top 50 base detection + top 150 donor detection results	90.9%

For NC2017Dev3Beta1, we had a chance to run our search system on a much larger world set created by merging the NC2017Dev3Beta1 world set and the NIMBLE evaluation world set. To generate top 200 results for each probe, we first filled the bucket with top 110 detected base images, and then filled the 90 remaining with top donors detected. When fusing base images and donor images, we used the cosine similarity scores for base images but did not use them for donors. Instead, we used the following score for each donor:

$$score(\text{donor}) = 0.9/\text{rank}(\text{donor}) \quad (1)$$

where rank(donor) is the donor detection rank for the donor image. This score is always smaller or equal to 0.9. The cap 0.9 imposed to donor scores prevent donors from being 1.0 when rank(donor) is 1. The performance is shown in Table 5. The gain from adding donor detection results is very small. We cannot find which provenances are bases or donors from the ground truth of that dataset but we found the dataset has very few donors after a visual examination.

Table 5: Provenance filtering results on NC2017Dev3Beta1.

Result	Recall rate (base only)	Recall rate (110 bases and then 90 donors)
Top 50 results	93.7%	94.0%
Top 100 results	93.9%	94.3%
Top 200 results	93.9%	94.3%

Graph Building Results

We evaluated our provenance graph building method with respect to different maximum number of nodes on NC2017 Dev3-Beta1 dataset. Table 6 shows these results, where best scores are indicated in bold font.

Table 6: Provenance results on NC2017Dev3Beta1.

#Max Nodes	MeanSimNLO	MeanSimNO	MeanSimLO	MeanNodeRecall
5	0.181238125315	0.323336383573	0.0294198584849	0.193914517725
10	0.314880510106	0.534688387660	0.0794494984609	0.371907844130
15	0.409656695674	0.682396122668	0.1170226257260	0.540148103242
20	0.472234556772	**0.777937157058**	0.1403277670010	0.690469666044
25	**0.477790504935**	0.765763055720	**0.1473198018190**	0.752900066207
30	0.456001256462	0.705338594064	0.1462554818990	0.759856364840
35	0.436594355859	0.654438834853	0.1454015553690	0.765888823850
40	0.418745397019	0.610120086271	0.1448606644320	0.770309485196
45	0.402475077677	0.571594074708	0.1444433649680	0.773902573332
50	0.387435517583	0.537760247947	0.1439085714310	**0.776572327323**

2.5 Discussion

Experimental evaluations show that base detection has much higher accuracy than donor detection. This is primarily because of the fact that base regions are much larger than donor regions, and therefore representation matching for base regions is relatively easier. We will conduct a detailed

analysis of detection results to determine failure cases for donor and base detection. We currently use VGG19 for extraction learned representations. In order to improve base detection, we plan to use more recent feature extractors, such as Residual Networks and Inception networks.

In terms of donor detection, we have shown that using segmented objects as queries improves the detection results, since it generates a more object-specific representation, in contrast to direct image-to-image matching. Rather than using a fixed image sub-sampling (188 windows), we will continue investigating using image segmentation and object detection techniques. Further, we will also investigate re-ranking methods to improve the accuracy of donor detection.

To improve the accuracy of our provenance graph construction algorithm, we plan to develop multi-tasking deep networks that can classify the manipulation type and estimate the manipulation parameters, if any. Even though the NC2017Dev3Beta1 dataset only has a very small fraction of donor provenances (note that links coming from donor nodes are far less accurate than links connecting base images due to the lower provenance detection performance of donors than that of base images), the 14.7% link similarity still looks a bit lower than what we expect to be useful practically. To overcome the data limitation challenge, we will create a new dataset that can be used to train the deep network. The new dataset will be similar to the one we created for splicing detection, with emphasis on probe-donor relationships and manipulation parameters.

3 Splicing Detection

3.1 Problem Description and Challenges

The recent NIMBLE 2017 Challenge from National Institute of Standards and Technology,[1] has reformulated the image splicing detection problem such that given a query image Q and a potential donor image P, the goal is to solve not only the detection problem, i.e., whether or not Q contains spliced regions from P, but also the localization problem, i.e., segmenting the spliced region(s) in both the donor and the spliced images. Since this new problem formulation constrains image splicing detection to a pair of images, we refer to it as the constrained image splicing detection (CISD) problem. Figure 3 shows three input samples along with their ground truth splicing masks and detection labels of CISD. This CISD problem can be viewed as a new formulation of the classic copy-move detection problem in the sense that it also looks for a potential region that is copy-move from image P to image Q. The CISD problem plays an important role in producing an image phylogeny graph [11, 10, 9] for a query image given a big dataset, especially in explaining how two images in neighboring nodes are associated.

It is worth noting that the two-input nature of the new CISD problem makes many existing image hypothesis used in classic copy-move and splicing detection no longer applicable. For example, [3] proposed an image splicing detection algorithm by differentiating single and double JPEG compression, but it is not useful for the CISD problem because (1) we can neither guarantee that the two inputs will be in JPEG format, nor that the inputs are compressed with the same quantization table at the same level of quality; and (2) even if both image are in JPEG format and one region in P is detected as doubly compressed, the CISD question *whether this region is originally from* Q still cannot be answered. As a result, the new CISD definition urges the use of more robust assumptions and features.

Visual clues, representing the correspondences between splicing regions [30, 34, 36, 13, 4, 15, 21], are useful for approaching the CISD problem since they are probably the least constraining assumptions that could be made. Using visual clues requires (1) representations for visual clues, and (2) mechanism to determine which two representations match. As one can see, these are exactly the first two steps ("representation" and "matching") in any general forgery detection framework (GFDF) while the last step (i.e. post-processing) in the GFDF is really to take advantage of the consistency within a set of true matchings to reduce false alarms. Though it seems that classic copy-move and splicing detection algorithms [36, 14, 13, 30, 34] relying on visual features can be easily modified for the new CISD problem, we note that the two major drawbacks of these existing algorithms are (1) these methods use handcrafted features which are less robust against image transformations (e.g., compression, noise addition and geometric transformations) which makes them inadequate for the CISD problem and (2) tuning each of three stages in GFDF on its own only optimizes performance disjointly rather than for the entire system as a whole.

We conceptually follow the GFDF and use Deep Matching and Verification Network (DMVN)—a novel deep learning-based splicing detection and localization method that is (1) end-to-end optimized, unlike previous GFDF approaches, (2) does not depend on extracting handcrafted, unrobust feature representations, (3) uses fully learnable parameters for image matching, and (4) mimics the human validation process to see whether visual evidence are enough to determine splicing relationships. Furthermore, this method is also distinct from recent deep learning based forgery detection

[1] https://www.nist.gov/itl/iad/mig/nimble-challenge-2017-evaluation

practices [33, 22, 8, 7] in the sense that our approach (1) is a full end-to-end deep learning solution instead of a deep learning module only for feature extraction [7, 8], (2) performs both localization and detection tasks instead of one or the other [33, 8, 22], and (3) introduces novel deep learning modules (*Deep Dense Matching* and *Visual Consistency Validator*) for performing visual matching and validation (see Fig.4-(a)).

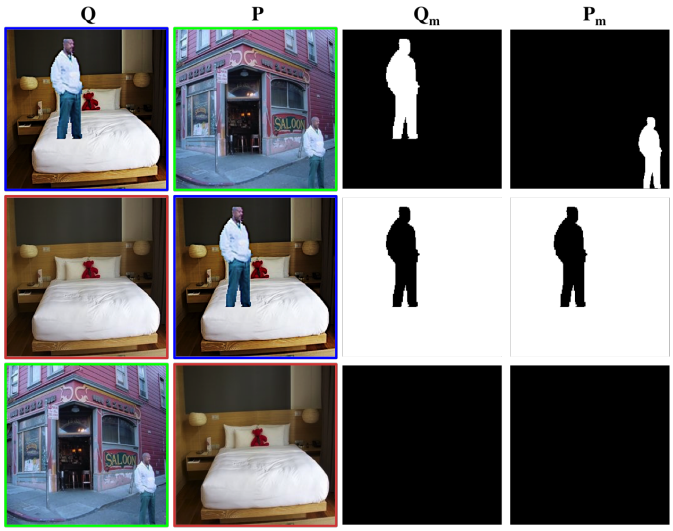


Figure 3: Constrained image splicing detection problem, where true spliced pixels are labeled as white. From top to bottom, sample detection labels are 1, 1, and 0, respectively.

3.2 Technical Approach

As shown in Figure 4, the overall network is designed such that block-wise learned representations of the input images are extracted using a convolutional network network – *CNN Feature Extractor* (e.g., AlexNet, ResNet or VGG) and fed into the proposed *Deep Dense Matching* module, which performs (as the name implies) dense matching between the two input images. In order to segment the splicing masks in the two images, we use an inception-based *Mask Deconvolution* module [28]. Further, the predicted masks are fed into a *Visual Consistency Validation* module that forces the model to focus on the segmented areas in both images. Finally, a Siamese-like module is used to extract splicing-specific dense representations of the segmented regions in the donor and query images, and it produces a probability value indicating the likelihood that the donor image was used to splice the query image. We describe the details of each of these stages in the following.

Splicing Localization Branch

Although other CNN models (e.g., ResNet [12]) can be also applied, we use the first four convolutional blocks of the VGG16 model [25] just for the sake of simplicity. Consequently, the two network inputs Q and P (of shape $3 \times 256 \times 256$) are transformed into deep tensor representations Q_f and P_f (of shape $512 \times 16 \times 16$.) It is well known that CNN features like Q_f and P_f have already exhibited certain level of invariance to luminance, scale, and rotation.

The purpose of *Deep Dense Matching* is to find possible matching regions between representations Q_f

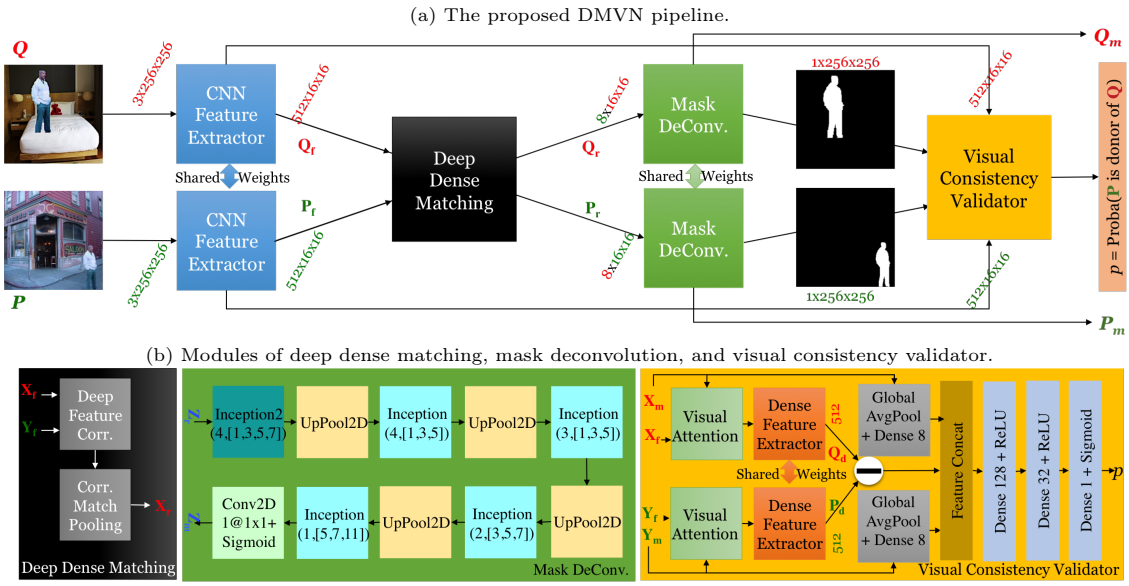


Figure 4: Deep matching and validation network for the constrained image splicing detection and localization.

and P_f. As shown in Fig. 4(b), this is achieved through two steps, namely *Deep Feature Correlation* and *Correspondence Match Pooling*. In *Deep Feature Correlation*, we exhaustively compute matching response using cross-correlation over all possible translations, as shown in Eq. (2)

$$\text{corr}(\mathsf{P}_f, \mathsf{Q}_f)[x, y, i, j] = \text{trans}(\mathsf{P}_f, x, y)[:, i, j] \cdot \mathsf{Q}_f[:, i, j] \tag{2}$$

where $\cdot$ is the dot product operator, and $\text{trans}(\mathsf{Z}_f, x, y)$ circularly translates Z_f *w.r.t.* (x, y) pixels, as defined in Eq. (3).

$$\text{trans}(\mathsf{Z}_f, x, y)[:, i, j] = \mathsf{Z}_f[:, \text{mod}(i + x, 16), \text{mod}(j + y, 16)] \tag{3}$$

In *Correspondence Match Pooling*, we extract out meaningful response maps using three types of pooling—average pooling as defined in Eq. (4)

$$\text{avgPool}\big(\text{corr}(\mathsf{P}_f, \mathsf{Q}_f)\big)[i, j] = \textstyle\sum_{x=0}^{15} \sum_{y=0}^{15} \text{corr}(\mathsf{P}_f, \mathsf{Q}_f)[x, y, i, j]/256 \tag{4}$$

max pooling as defined in Eq. (5)

$$\text{maxPool}\big(\text{corr}(\mathsf{P}_f, \mathsf{Q}_f)\big)[i, j] = \max_{x,y}\{\text{corr}(\mathsf{P}_f, \mathsf{Q}_f)[x, y, i, j]\} \tag{5}$$

and argsort pooling as defined in Eq. (6)

$$\text{argsortPool}\big(\text{corr}(\mathsf{P}_f, \mathsf{Q}_f)\big)[k] = \text{corr}(\mathsf{P}_f, \mathsf{Q}_f)[k_x, k_y, i, j] \tag{6}$$

where (k_x, k_y) in Eq. (6) is determined by the kth maximum response over all translations. Finally, we concatenate one average, one max, and the top six argsort response maps along the feature dimension and obtain the dense matching response Q_r of shape $8 \times 16 \times 16$ for Q. By interchanging the roles of P_f and Q_f in $\text{corr}(\cdot, \cdot)$, one can therefore obtain P_r.

Fig. 5 visualizes intermediate results of the proposed *Deep Dense Matching* layer for the three testing samples from Fig. 3. As one can see, the proposed deep dense matching module 1) successfully

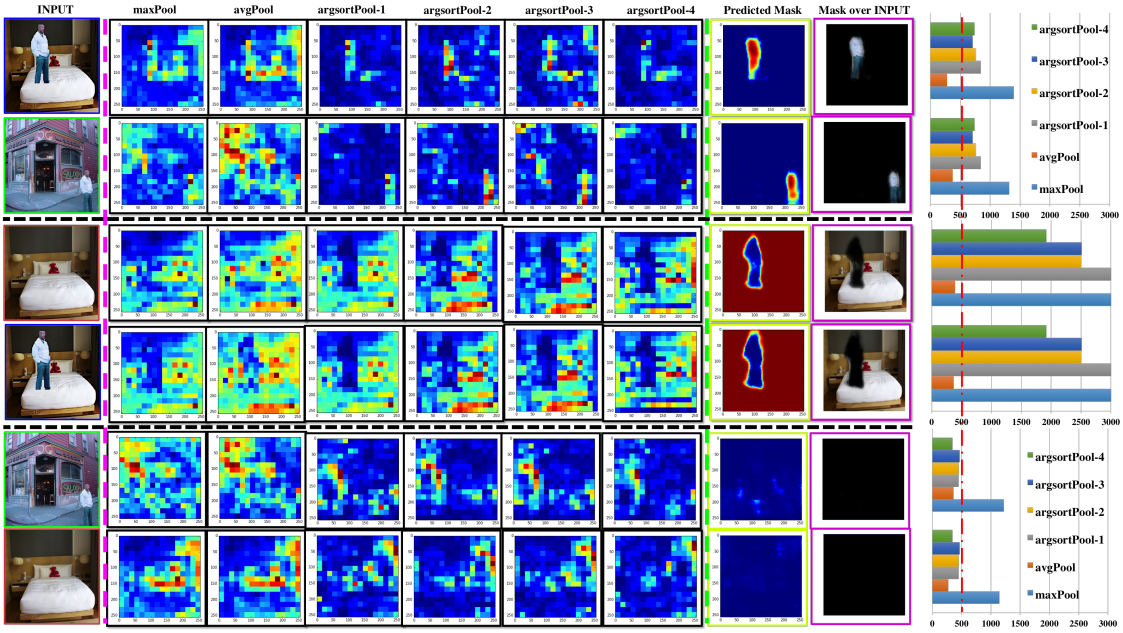


Figure 5: Visualization of selected DMVN layers and the statistics of the 5th maximum responses from correlation matching pooling layers. Note: 1) all layers are shown by linearly rescaling to [0,1], and visualized *w.r.t.* the `Jet` color map, where more red means higher response; 2) data ranges of pooled matching response maps could be very different, but predicted masks share the same range [0,1]; and 3) rough maximum values of matching responses can be seen in the statistics on the far-right.

localizes potential splicing regions, and 2) produces substantially higher responses to the two positive samples above than the negative sample (see the red dash line on the right half stats figure in Fig. 5), implying that the previous *CNN Feature Extractor* and *Deep Feature Correlation* provides meaningful representation with high discernibility.

In order to produce a splicing mask from the dense response map, we use a *Mask Deconvolution* module, as shown in Fig 4(b), where we gradually deconvolve a response map by a factor of 2 until its size reaches the size of input, i.e., 256×256. During each deconvolution stage, we apply an inception module [28] with a larger filter size and a smaller number of filters, where the two types of inception modules can be seen in Fig. 6. This enables us to obtain splicing masks for both image P and Q, i.e., outputs P_m and Q_m (see examples in the "Predicted Mask" column in Fig. 5.)

Splicing Detection Branch

Intuitively, given a query image Q and a potential donor P along with predicted splicing masks Q_m and P_m, one can easily validate whether two masks match each other by their image contents. We therefore follow this intuition to design a *Visual Consistency Validation* module to fulfill the splicing detection task as shown in Fig. 4(b). Specifically, we first use the *Visual Attention* module to zero-out all non-spliced regions in the CNN feature, as shown in Eq. (7),

$$\text{visAtt}(\mathsf{Z}_f, \mathsf{Z}'_m)[c,i,j] = \begin{cases} \mathsf{Z}_f[c,i,j], & \text{if } \mathsf{Z}'_m[i,j] > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

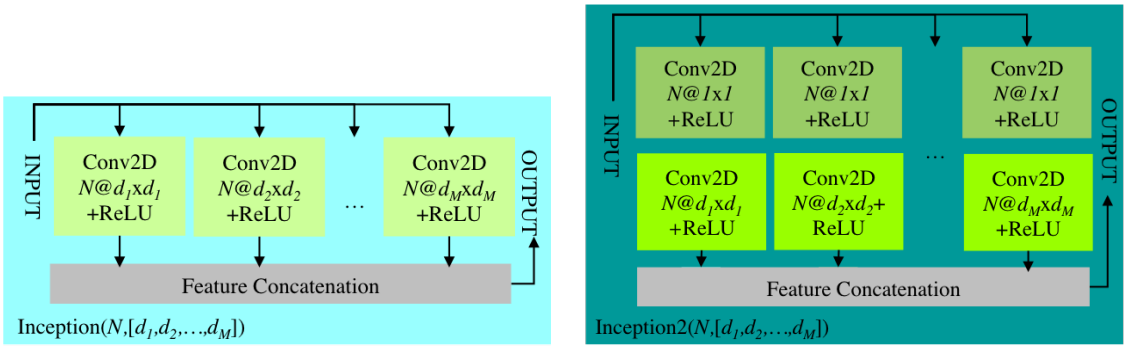


Figure 6: The internal architecture of the two types of used inception modules.

where Z'_m is the result of Z_m after classic *MaxPooling2D* for a size (16,16). This process is analogous to forcing the network to pay attention only to splicing regions. Furthermore, we follow a Siamese-like network to compare these two attention features—namely, extract a new round of features from the two attention features using the *Deep Feature Extractor* (see Fig. 7 for detailed architecture), and then compute the difference between the two resulting features. We then concatenate this feature with the feature obtained from the average mask responses. Finally, we use three stacked dense layers to infer the probability that $p = \text{Proba}(\mathsf{P} \text{ is a donor of } \mathsf{Q})$, and this fulfills the detection task as shown in Fig. 4(b).

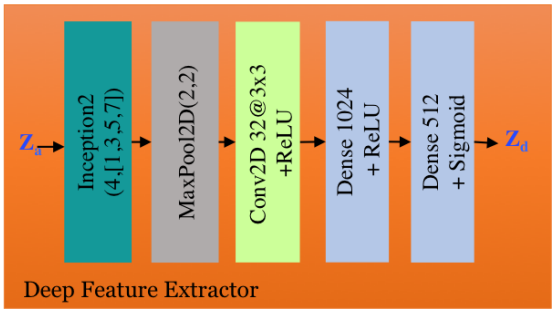


Figure 7: Internal architecture of the deep feature extractor.

3.3 Training Data and End-to-End Training Strategy

To the best of our knowledge, no dataset exists that is large enough to be directly used for training the proposed DMVN model. To overcome this limitation, we use the SUN2012 object detection dataset [32] and the MS COCO dataset [16] to create training samples according to the unsupervised generation process described in [35]. Briefly, we begin with a random image X with polygon-based object annotations, randomly select an object in X, then randomly transform this object and paste it to another randomly selected image Y to obtain a resulting composite image Z. We could harvest at most three (two positive and one negative) training $\{inputs, outputs\}$ samples for each unsupervised data generation. For instance, Fig. 3 gives a set of three training samples of this type.

In terms of the parameters controlling the data generation process, we equally likely pick an image and an object, random affine transformation involving a scale change in $\mathbb{U}(.5, 4)$, rotation in $\mathbb{U}(-10, +10)$, shift in $\mathbb{U}(-127, +127)$, translation $\mathbb{U}(-127, +127)$ and random luminance change in

$\mathbb{U}(-32, +32)$. This enables us to create as many samples as needed to train the network end-to-end. Effectively, we create 1.5 M(illion), 0.3M, and 0.3M synthesized samples for training, validation, and testing, respectively.

The proposed DMVN was implemented using the *Keras* deep learning framework with the *Theano* backend, including all custom correlation and pooling layers. Our model was trained with the *Adadelta* optimizer *w.r.t.* the *log loss* for both localization and detection branches. Since we design the splicing detection branch as a *Visual Consistency Validator* of image contents on predicted splicing masks, this branch output may not produce meaningful gradients unless the localization branch produced meaningful splicing masks. Thus, we first focus on the localization branch of DMVN model only. Once this localization branch converges, we freeze its weights, add on the detection branch, and train the detection branch until it converges. We finally unfreeze all DMVN weights and train the entire model end-to-end using the stochastic gradient optimizer with a learning rate $1e-5$ and momentum of 0.9. In summary, we achieve 98.52%, 98.67%, and 97.88% prediction accuracy on the localization branch, and 97.75%, 97.53% and 97.69% prediction accuracy on the detection branch on our synthesized training, validation, and testing datasets, respectively.

3.4 Baseline Method and Evaluation Data and Metrics

Baseline Methods and Test Settings

Since the CISD formulation is completely new, we compare against baseline algorithms from the state-of-the-art copy-move detection algorithms. We rely on visual clues, because when we concatenate the two inputs from a CISD sample into a single combined image, the resulting image contains copy-move forgery if the CISD sample is positive. Specifically, we choose the classic block matching-based approach [17], the classic Zernike moments-based block matching [23] with nearest-neighbor search, the SURF feature-based keypoint matching [5] and the dense field matching [6]. All used baselines are implemented by either a third-party or by the authors of the corresponding papers.[2]

With regard to preprocessing, we resize an image to 256×256 to fit the input size of the proposed DMVN, and thus a 256×512 image for those baseline algorithms. With regard to postprocessing, we do not apply any to DMVN, i.e., using the outputs from DMVN localization and detection branches directly, while keeping default postprocessing settings of baselines unchanged. Since some baseline methods only output a splicing mask but not a binary decision on detection, we follow the tradition in the classic ISD and copy-move community to determine that a sample is positive if no pixel is labeled as spliced in a mask. Finally, all baseline methods are run on `Intel Xeon` CPU E5-2695@2.40GHz, and the proposed DMVN is run on `Nividia TitanX` GPU.

Evaluation Data

We conduct evaluation experiments on two large datasets: 1) the paired CASIA dataset, and 2) the NIST-provided Nimble 2017 image splicing detection dataset Dev1-Beta4. The paired CASIA dataset is a modified version of the original CASIA TIDEv2.0 dataset [31][3] which contains 7200 authentic color images and 5123 color images tampered with by realistic manual manipulations

[2] Available at https://github.com/rahmatnazali/image-copy-move-detection.git, https://www5.cs.fau.de/research/software/copy-move-forgery-detection/, http://www.grip.unina.it/research/83-image-forensics/90-copy-move-forgery.html as the date of April 10, 2017.

[3] http://forensics.idealtest.org/casiav2/

(e.g., resize, deform, and blurring) through *Adobe Photoshop CS3*. It was originally collected for both the image copy-move problem and the classic image splicing detection problem. Since the CISD problem requires a pair of inputs, we select pairs of images from the original CASIA dataset to create the new paired CASIA dataset. Among the 5123 CASIA tampered images, we that found 3302 are of the copy-move problem and 1821 are of the classic ISD task. We therefore generate 3642 positive samples by pairing these 1821 spliced images with their true donor images, and collect 5000 negative samples by randomly pairing 7491 color images from the same CASIA-defined content category.

With regard to the Nimble 2017 dataset, it is provided by NIST with 98 positive samples and 529,836 negative samples. This challenging dataset is particularly designed for the CISD task with considerations to 1) a very large scale (more than a half million samples), 2) more realistic and artistic manipulations like image impainting, seam-carving etc., 3) difficult negative samples with visually similar foreground and background, and 4) mimicking the real application scenario where the ratio of negative samples to positive samples is extremely huge.

It is worth emphasizing that 1) we directly test the DMVN models trained by our synthetic data without any finetuning, and 2) ground truth splicing masks are not available for both dataset.

Evaluation Metrics

Since both datasets do not provide ground truth splicing masks, we focus on assessing the splicing detection performance. We follow the tradition of the classic ISD and copy-move community, namely, using precision, recall, and f-score: TP stands for *true positive*, i.e., correctly detected as spliced; FN stands for *false negative*, i.e., incorrectly detected as not-spliced; FP stands for *false positive*, i.e., incorrectly detected as spliced; and TN stands for *true negative*, i.e., correctly detected as not-spliced.

$$\text{precision} = TP/(TP + FP) \tag{8}$$

$$\text{recall} = TP/(TP + FN) \tag{9}$$

$$\text{f-score} = 2TP/(2TP + FN + FP) \tag{10}$$

Furthermore, we also use *area under the ROC curve (AUC)* to evaluate overall system performance at different operation points, where the receiver operating characteristic (ROC) curve is determined as the function of *true positive rate (TPR)* in terms of *false positive rate (FPR)*. TPR and FPR are defined as shown in Eqs. (11) and (12). The area under an ROC curve then quantifies the overability of the system to discriminate between two classes. It is worth noting that a system which is no better at identifying true positives than random guessing has an area of 0.5, and a perfect system without false positives and false negatives has an area of 1.0; and that AUC is the only official metric used by the Nimble 2017 challenge .

$$TPR = TP/(TP + FN) \tag{11}$$

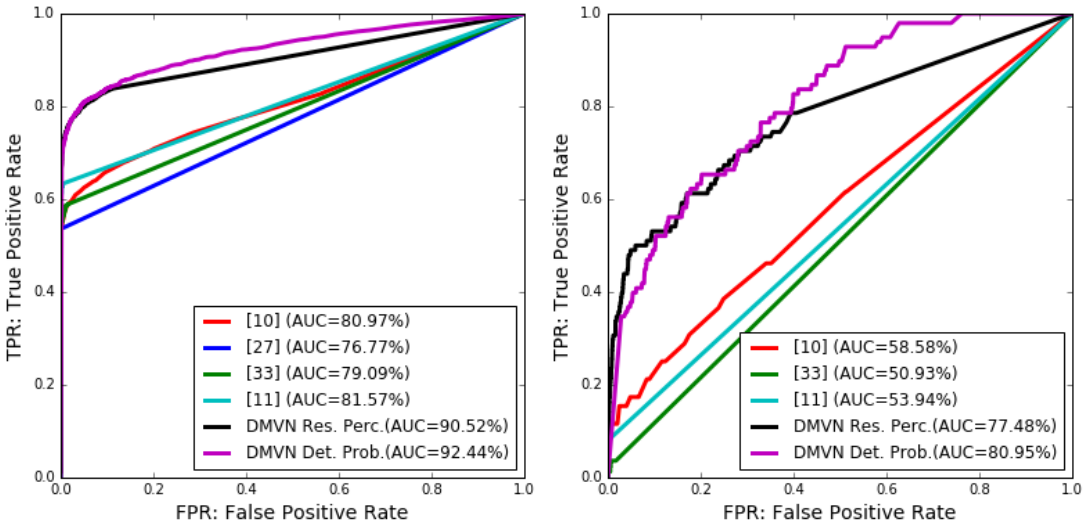
$$FPR = FP/(TN + FP) \tag{12}$$

3.5 Evaluation Results

The most challenging task in the CISD is to 1) find spliced regions under various transformations like translation, rotation, scale, crop, etc., while dealing with complicated cases like multiple instances (a donor image contains multiple instances that are similar to a true spliced region), and multiple

Table 7: CISD performance comparison on CASIA

Method	Precision	Recall	F-score	Time (sec/sample)
[5]	51.64%	82.92%	63.64%	1.85
[17]	99.69%	53.53%	69.66%	$6.27\times10^{+2}$
[23]	96.14%	58.95%	73.09%	8.61
[6]	98.97%	63.34%	77.25%	3.23
DMVN loc.	91.52%	79.18%	84.91%	7.16×10^{-2}
DMVN det	94.15%	79.08%	85.96%	8.29×10^{-2}



(a) ROC on the paired CASIA dataset

(b) ROC on the Nimble 2017 dataset

Figure 8: CISD performance comparison using ROC.

spliced regions (a donor image contributes more than one region); and 2) reduce false alarms on those visually similar but non-spliced regions.

Table 7 shows the splicing detection performance of baseline approaches and the proposed DMVN methods on the paired CASIA dataset, where *DMVN loc.* means that we determine whether a sample (a pair of images) is positive or not by checking whether any pixel in a predicted splicing mask (from the DMVN localization branch) is positive, and *DMVN det.* means that we directly use the relation probability (from the DMVN detection branch) to determine a sample's label. As one can see, the proposed DMVN methods outperform peer algorithms by a large margin in terms of f-score (∼7% higher) on the paired CASIA dataset. Note also that the proposed DMVN is significantly faster than baseline approaches by 20+ times, and that the DMVN detection branch improves our precision score from 91.52% to 94.15% while only reducing recall score for 0.1%, thus indicating the effectiveness of the proposed validation idea which relies on visual attention and Siamese architecture. Fig. 8 compares ROC and AUC scores for different methods, where *DMVN Res. Perc.* and *DMVN Det. Prob.* means that the threshold used to obtain TPR and FPR is based on the positive pixel percentage in a resulting mask, and the detection branch's output probability, respectively. Again, the proposed DMVN methods are noticeably better than others on AUC scores (10%+ on CASIA, and 20%+ on Nimble 2017).

With regard to splicing localization performance, Fig. 9 shows how the proposed DMVN method conquered this challenge on the paired CASIA dataset, where X_m indicates a splicing mask binarized with threshold 0.5, and $\mathsf{X}_m * \mathsf{X}$ indicates an overlaid image by using the splicing mask as the alpha

channel with 40% transparency. To see this, one shall go to each row in Fig. 9, where the left and right sides show true positive and negative samples, respectively. Note that two samples on each row are intentionally picked from a similar CASIA category and/or a similar object class. As one can see, the proposed DMVN method not only predicts meaningful splicing masks on those positive samples, but also correctly suppresses splicing masks on those negative samples. Fig. 10 shows the manually annotated ground truth masks along with our predicted masks for the Nimble 2017 Dev1-Beta4 dataset.

3.5.1 Discussion

Fig. 11 qualitatively compares the splicing localization performance for all supported baselines. As one can see: 1) classic exhaustive block matching method [17] is sensitive to transformation, but good at capturing nearly duplicate regions; 2) block matching algorithm relying on Zernike moments [23] handles a certain level of transformations, but fails to maintain the completeness of a splicing region (see those holes in "[23]'s Masks" in row 5 of Fig. 11); 3) a keypoint-based detector [5] may fail due to no effective keypoints or noisy keypoints, which can commonly be seen in a homogeneous region or regions with similar texture, and one has to further convert potential matching points to a mask (see the last row; finding correspondence does not mean find a mask); and 4) the proposed DMVN method does not suffer the drawbacks of the previous three methods and gives satisfactory localization results on homogeneous and non-homogeneous regions even under severe transformations.

With regard to drawbacks, the proposed DMVN has some difficulty detecting splicing objects smaller than 8×8; this is due to the down-sampling in *CNN Feature Extractor*. As one may notice, our AUC scores on the Nimble 2017 dataset are much lower than those on the paired CASIA dataset. Besides the fact of extremely unbalanced positive (98) and negative samples (529836), we notice that the Nimble 2017 Dev1-Beta4 dataset contains much more challenging samples. For example, the proposed DMVN approach produces false alarms when two images P and Q are from consecutive video frames, because it mistakenly predicts those similar but genuine objects in both P and Q as spliced objects.

Finally, we recently noticed that the proposed DMVN performance is not stable across all Nimble 2017 development sets. In particular, we observe AUC scores 0.72 for Dev1-Beta3, 0.81 for Dev1-Beta4, 0.79 for Dev2-Beta1, and 0.60 for Dev3-Beta1. Early investigation indicates that this fluctuation is mainly due to the instable performance of detecting true positive samples, especially for those with severe transformations.

4 Summary

In this report, we have presented the DiSPARITY team's self-evaluation efforts in provenance filtering and graph construction, and splicing detection. For provenance filtering, we used a multi-scale approach for image representation, in which we adopted VGG19 as a feature extractor. Each image is represented using 188 regions at multiple scale and a 1000D feature vector is extracted using VGG19. Cosine similarity is calculated between corresponding regions in ear pair of images and the top N images are retrieved as base images of the given probe image. In order to retrieve the donor images, we construct a minimum spanning tree from the top N images and use image differencing to create a set of probe regions. VGG19 is used again to extract learned representations from the probe regions and these representations are used as new queries to search the references set, at a

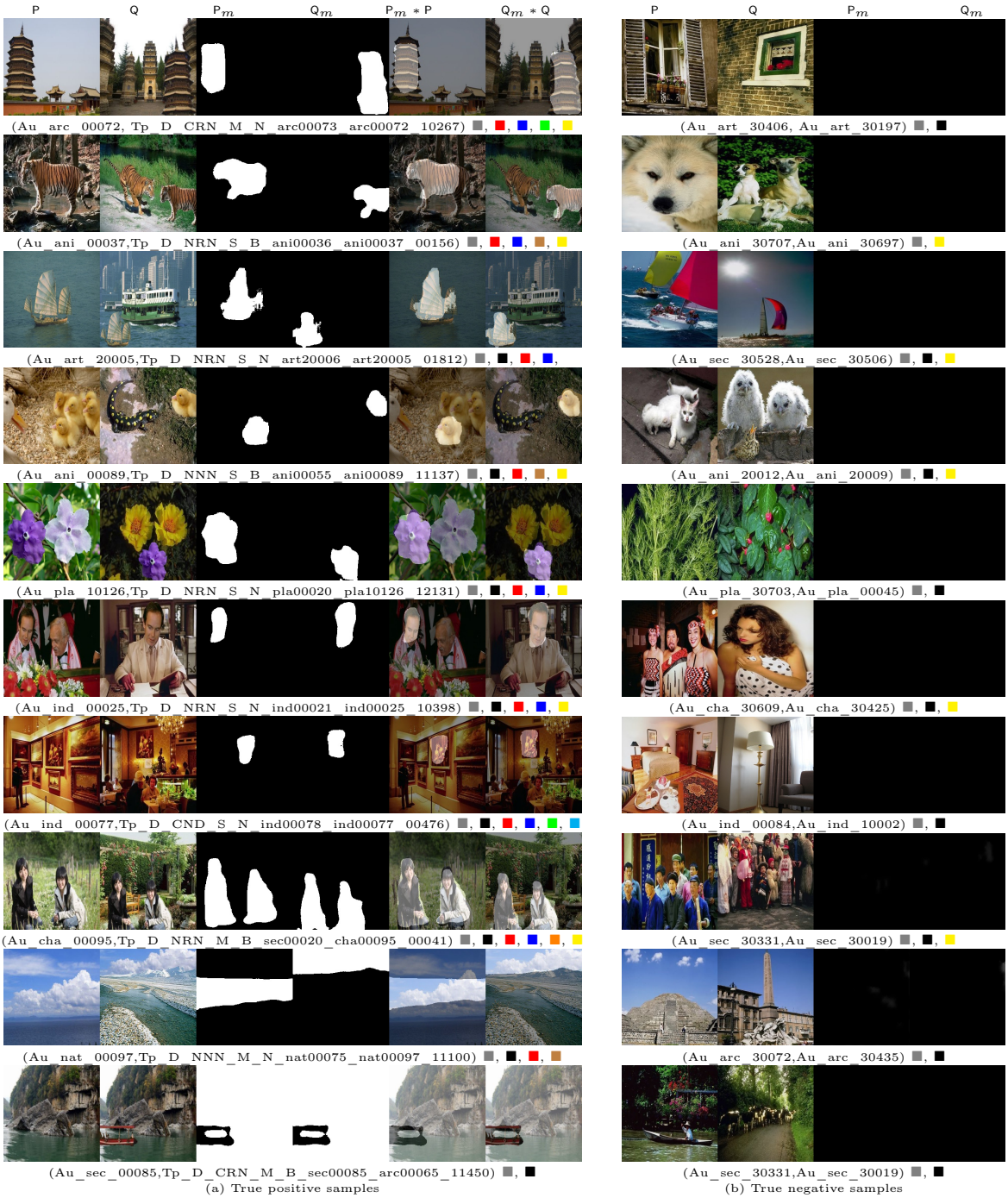
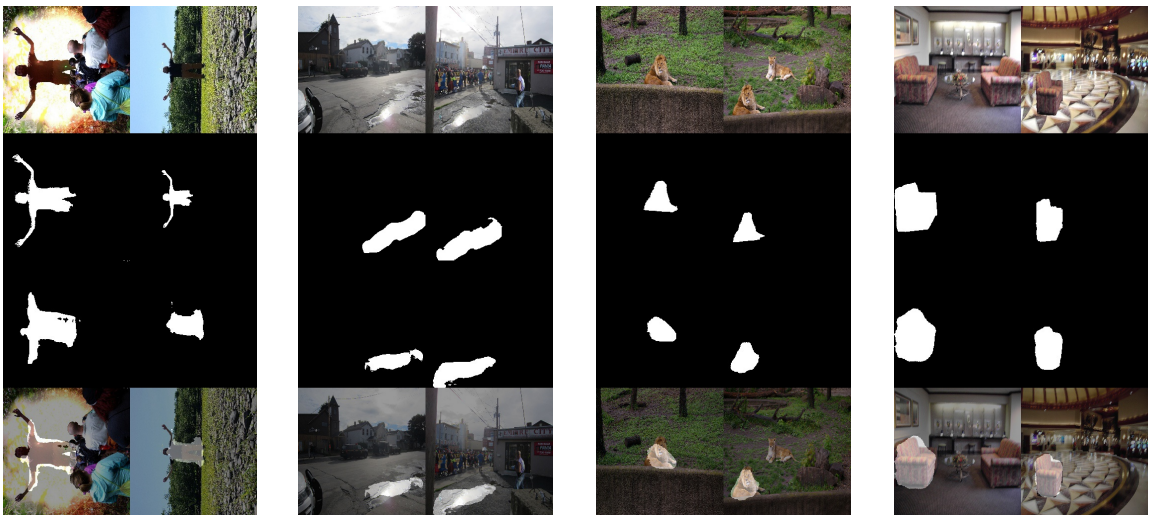


Figure 9: DMVN localized splicing masks on the paired CASIA TIDEv2.0 dataset. (PID, QID) indicate the original CASIA filename of P and Q. Color blocks indicate different factors which splicing localization need to be robust against, namely ■: translation, ■: scale, ■: rotation, ■: perspective, ■: crop, ■: multiple instances, ■: multiple splicing objects, ■:similar foreground, ■: similar background.



(a) (PID,QID)=(19e32f2b29a2b345dfd28a8ad857b3fc,797a0b5dd9c47a4b391f4cee60ba9354) (b) (PID,QID)=(c0cfa4a6a95b3762f9d966e09a0340b9,030d521fca99981c442e60e4b2155699) (c) (PID,QID)=(9883d12156ed82344d65fe19575e1d84,8b570c4bd216f5144774b9c0bc61bbd4) (d) (PID,QID)=(4bf6a60d74ecd725e3e4b0eedc7d7cc0,96492c250780ebababfcebc585ef1db6)

Figure 10: Predicted splicing masks the NC2017 dataset. From top to bottom, input pair, manually annotated ground truth masks, DMVN predicted splicing masks, and overlaid masks.

finer scale, to retrieve donor images.

For the splicing detection task, we presented a novel, end-to-end deep neural architecture that concurrently solves both splicing detection and localization. The new architecture – deep matching and verification network – takes a pair of query and probe images as inputs and consists of two multi-tasking branches. The first branch estimates the probability that the probe image has been used to splice query image. The second branch produces image masks localizing the spliced region in the query image and splicing region in the probe image. The network uses classic CNN feature extraction modules (e.g. VGG or AlexNet) to extract learned representations of the input images. These representations are then fed into a novel Deep Dense Matching module that finds correspondences between the inputs images by correlating the extracted representations using a visual attention mechanism. A deconvolutional layer is used to construct binary masks representing splicing and spliced regions.

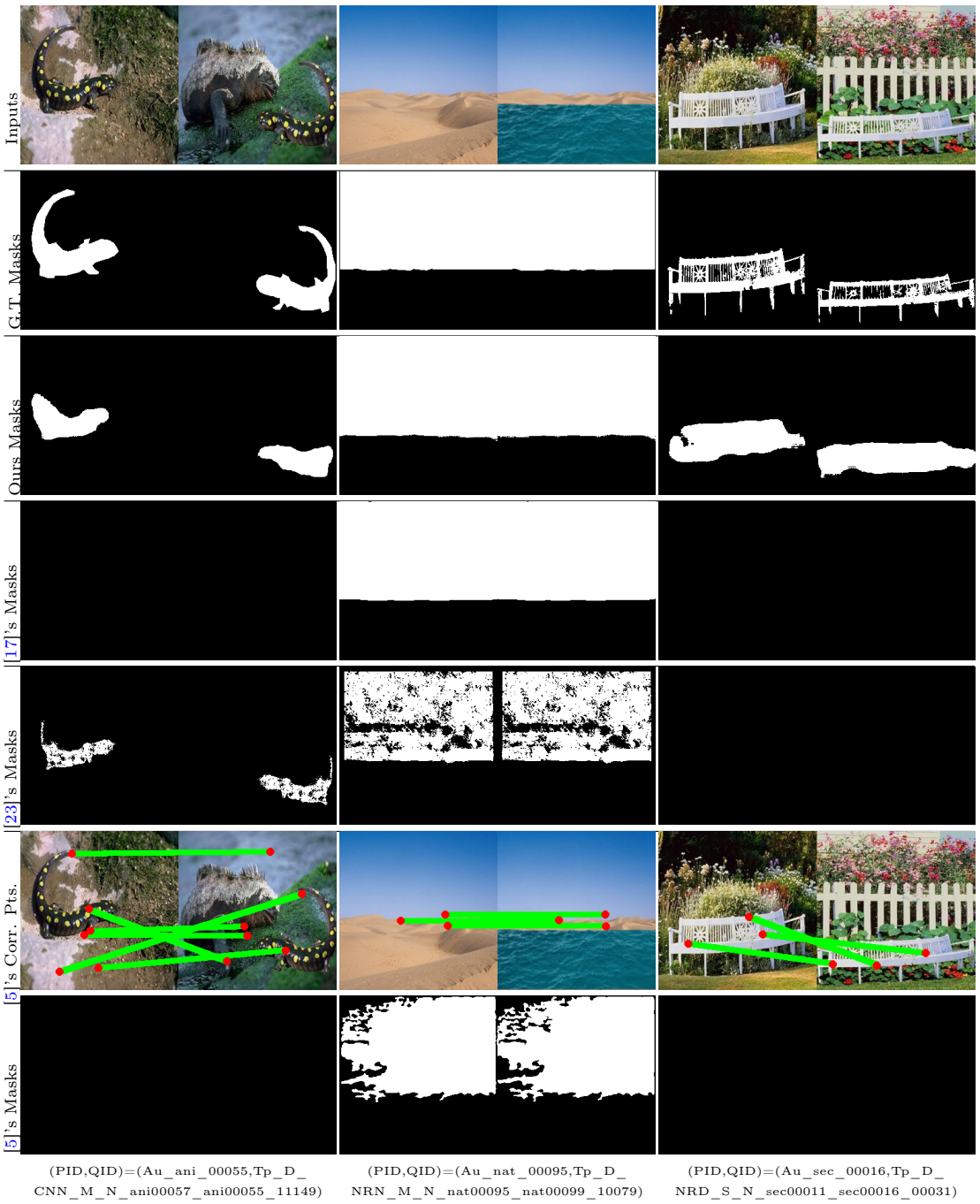


Figure 11: Visual comparison of detected masks. From top to bottom, input pair, manually annotated ground truth masks, predicted masks of using the proposed DMVN method, predicted masks of Alg. [17], [23]; the last two rows are [5]'s matched keypoints and predicted masks.

References

[1] https://www.nist.gov/itl/iad/mig/nimble-challenge.

[2] https://www.nist.gov/itl/iad/mig/nimble-challenge-2017-evaluation.

[3] Irene Amerini, Rudy Becarelli, Roberto Caldelli, and Andrea Del Mastio. Splicing forgeries localization through the use of first digit features. In *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*, pages 143–148. IEEE, 2014.

[4] Edoardo Ardizzone, Alessandro Bruno, and Giuseppe Mazzola. Copy–move forgery detection by matching triangles of keypoints. *IEEE Transactions on Information Forensics and Security*, 10(10):2084–2094, 2015.

[5] Vincent Christlein, Christian Riess, Johannes Jordan, Corinna Riess, and Elli Angelopoulou. An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on information forensics and security*, 7(6):1841–1854, 2012.

[6] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Efficient dense-field copy–move forgery detection. *IEEE Transactions on Information Forensics and Security*, 10(11):2284–2297, 2015.

[7] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. *arXiv preprint arXiv:1703.04615*, 2017.

[8] Davide Cozzolino and Luisa Verdoliva. Single-image splicing localization through autoencoder-based anomaly detection. In *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pages 1–6. IEEE, 2016.

[9] Alberto A de Oliveira, Pasquale Ferrara, Alessia De Rosa, Alessandro Piva, Mauro Barni, Siome Goldenstein, Zanoni Dias, and Anderson Rocha. Multiple parenting phylogeny relationships in digital images. *IEEE Transactions on Information Forensics and Security*, 11(2):328–343, 2016.

[10] Zanoni Dias, Siome Goldenstein, and Anderson Rocha. Large-scale image phylogeny: Tracing image ancestral relationships. *Ieee Multimedia*, 20(3):58–70, 2013.

[11] Zanoni Dias, Anderson Rocha, and Siome Goldenstein. Image phylogeny by minimal spanning trees. *IEEE Transactions on Information Forensics and Security*, 7(2):774–788, 2012.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[13] Ce Li, Qiang Ma, Limei Xiao, Ming Li, and Aihua Zhang. Image splicing detection based on markov features in qdct domain. *Neurocomputing*, 228:29–36, 2017.

[14] Haodong Li, Weiqi Luo, Xiaoqing Qiu, and Jiwu Huang. Image forgery localization via integrating tampering possibility maps. *IEEE Transactions on Information Forensics and Security*, 12(5):1240–1252, 2017.

[15] Jian Li, Xiaolong Li, Bin Yang, and Xingming Sun. Segmentation-based image copy-move forgery detection scheme. *IEEE Transactions on Information Forensics and Security*, 10(3):507–518, 2015.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[17] Weiqi Luo, Jiwu Huang, and Guoping Qiu. Robust detection of region-duplication forgery in digital image. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 746–749. IEEE, 2006.

[18] Florent Perronnin and Christopher R. Dance. Fisher kenrels on visual vocabularies for image categorizaton. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2007.

[19] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 143–156, 2010.

[20] Xiang Zhang Michaël Mathieu Rob Fergus Yann LeCun Pierre Sermanet, David Eigen. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, page abs/1312.6229, 2013.

[21] Chi-Man Pun, Xiao-Chen Yuan, and Xiu-Li Bi. Image forgery detection using adaptive over-segmentation and feature point matching. *IEEE Transactions on Information Forensics and Security*, 10(8):1705–1716, 2015.

[22] Yuan Rao and Jiangqun Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pages 1–6. IEEE, 2016.

[23] Seung-Jin Ryu, Min-Jeong Lee, and Heung-Kyu Lee. Detection of copy-rotate-move forgery using zernike moments. In *International Workshop on Information Hiding*, pages 51–65. Springer, 2010.

[24] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[27] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. *CoRR*, abs/1511.06452, 2015.

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826, 2016.

[30] Wei Wang, Jing Dong, and Tieniu Tan. Effective image splicing detection based on image chroma. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1257–1260. IEEE, 2009.

[31] Wei Wang, Jing Dong, and Tieniu Tan. Image tampering detection based on stationary distribution of markov chain. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2101–2104. IEEE, 2010.

[32] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

[33] Ying Zhang, Lei Lei Win, Jonathan Goh, and Vrizlynn LL Thing. Image region forgery detection: A deep learning approach. In *Proceedings of the Singapore Cyber-Security Conference (SG-CRC) 2016: Cyber-Security by Design*, volume 14, page 1. IOS Press, 2016.

[34] Xudong Zhao, Shenghong Li, Shilin Wang, Jianhua Li, and Kongjin Yang. Optimal chroma-like channel design for passive color image splicing detection. *EURASIP Journal on Advances in Signal Processing*, 2012(1):240, 2012.

[35] Jun-Yan Zhu, Philipp Krahenbuhl, Eli Shechtman, and Alexei A Efros. Learning a discriminative model for the perception of realism in composite images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3943–3951, 2015.

[36] Ye Zhu, Xuanjing Shen, and Haipeng Chen. Copy-move forgery detection based on scaled orb. *Multimedia Tools and Applications*, 75(6):3221–3233, 2016.