Journal of Imaging Science and Technology® 63(6): 060402-1–060402-16, 2019.

Real-time Small-object Change Detection from Ground Vehicles Using a Siamese Convolutional Neural Network

Sander R. Klomp▲ **and Dennis W. J. M. van de Wouw**
Eindhoven University of Technology, SPS-VCA Group of Electr. Eng., Groene Loper 3, 5612 AE Eindhoven, the Netherlands
ViNotion B.V., Daalakkersweg 2-58, 5641 JA Eindhoven, the Netherlands
E-mail: s.r.klomp@tue.nl

Peter H. N. de With
Eindhoven University of Technology, SPS-VCA Group of Electr. Eng., Groene Loper 3, 5612 AE Eindhoven, the Netherlands

Abstract. *Detecting changes in an uncontrolled environment using cameras mounted on a ground vehicle is critical for the detection of roadside Improvised Explosive Devices (IEDs). Hidden IEDs are often accompanied by visible markers, whose appearances are a priori unknown. Little work has been published on detecting unknown objects using deep learning. This article shows the feasibility of applying convolutional neural networks (CNNs) to predict the location of markers in real time, compared to an earlier reference recording. The authors investigate novel encoder–decoder Siamese CNN architectures and introduce a modified double-margin contrastive loss function, to achieve pixel-level change detection results. Their dataset consists of seven pairs of challenging real-world recordings, and they investigate augmentation with artificial object data. The proposed network architecture can compare two images of* 1920×1440 *pixels in 27 ms on an RTX Titan GPU and significantly outperforms state-of-the-art networks and algorithms on our dataset in terms of F-1 score by 0.28.* © *2019 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.ImagingSci.Technol.2019.63.6.060402]

1. INTRODUCTION

Small-object change detection in an uncontrolled environment compared to an earlier moment in time is a challenging yet important problem for security applications. Basic applications include detecting roadside markers for Improvised Explosive Devices (IEDs) [1], litter detection, or detecting suspicious abandoned objects in public spaces. Detection of IED markers is the main reason for this research, as real-time automated detection can reduce IED casualties by spotting suspicious objects at a safe distance. In addition, the detection of unknown objects becomes particularly complicated when data is captured from moving ground vehicles. The use of moving vehicles is motivated because it enlarges the field of exploration significantly compared to existing change detection and anomalous object detection systems, which mostly rely on static cameras. This motivates the desire for real-time automated detection from a ground vehicle.

Real-time mobile automated systems for the detection of small objects have been explored in the past. Specifically for IED detection various techniques may be used, such as LiDAR [2], ground-based monocular camera imagery [3–5], ground-based stereo imagery [6, 7], or imagery from unmanned aerial vehicles [8]. What these methods have in common is that they search for small environmental changes compared to imagery captured at an earlier moment in time. More general methods for change detection from ground vehicles using camera imagery focus on background subtraction techniques [9, 10] or deep learning [11–13]. The background subtraction research for non-static cameras primarily distinguishes the objects from background based on motion, while we are interested in static objects. Research on deep learning focuses more on changed areas in the image, instead of searching specifically for unknown objects. Finally, a system for abandoned object detection from a ground vehicle is proposed in [14], which works well in controlled indoor environments. However, none of these methods are sufficiently capable of detecting small-object changes in the outdoor environments of our dataset due to severe lighting differences and dynamic backgrounds, such as moving tree branches. An example of these challenges in our dataset is shown in Figure 1.

This research is an extended version of [15]. It focuses on change detection between two aligned videos from different moments in time, with robustness to illumination differences, dynamic surroundings and small alignment errors, using a novel Convolutional Neural Network (CNN) architecture. The videos are acquired from a moving ground vehicle. Image alignment is performed using the system of Van de Wouw *et al.* [6]. This system consists of a processing chain using GPS-based image retrieval to recover a past image with the best possible viewpoint overlap, depth image generation from stereo cameras, and image alignment based on both features and depth information. These aligned images are suitable for 2D change detection. Van de Wouw *et al.* [6] used heuristics and post-processing to detect the changes, which yielded false positives on both dynamic backgrounds and complex lighting changes, especially hard shadows. The purpose of this article is to replace the decision

▲ IS&T Member.

Received June 24, 2019; accepted for publication Sept. 26, 2019; published online Dec. 16, 2019. Associate Editor: Shujian Yu.
1062-3701/2019/63(6)/060402/16/$25.00

(a) Live image (b) Closest reference image based on viewpoint overlap

Figure 1. Visual example of a frame pair with significant lighting and dynamic background changes.

made by our novel CNN architecture, aiming at improved detection while maintaining real-time operation.

Our contributions are fourfold. First, we propose a novel CNN architecture that can perform change detection in real time (around 27 ms per frame) on high-resolution imagery (1920×1440 pixels), where the network is inspired by proven concepts from related domains, i.e., object detection, patch matching, and semantic segmentation networks, thus allowing us to experiment freely with architectural design decisions. Second, we propose the use of an extended double-margin contrastive loss function for the training of general Siamese networks for change detection. Third, we extensively investigate the generalization power of the network by training and testing on different object classes and artificially augmented images. Fourth, we optimize the speed of the network with a more efficient architecture choice, optimized post-processing, and the use of publicly available optimization tools. Contributions 3 and 4 are new compared to [15]. All contributions are extensively validated on our own experimental datasets of ground-based images, containing changes in the form of static objects.

2. RELATED WORK

Change detection is a broad subject that can be applied to various types of data, such as 1D time series, images, graphs, or videos [16]. We limit ourselves to change detection in images (extracted from videos), surveys of which can be found in [17, 18]. These surveys focus primarily on conventional computer vision image processing, which we found to be insufficiently robust to dynamic backgrounds and strong lighting differences. Hence the focus of this section lies on deep learning-based approaches due to their recent success and advances in multiple computer vision problems.

CNNs can be used for change detection in various ways. In the absence of training data, CNNs pretrained on the ImageNet dataset [19] may be used as a feature extractor. The Euclidean distance in feature space is then used to generate a difference image [11, 20–22]. This is shown to outperform handcrafted features, especially when employing features from multiple layers in the network [21]. However, the CNN features were never specifically trained to be well-separable by Euclidean distance, which may limit change detection accuracy. Furthermore, most existing algorithms employ either VGG-16 [23] or Alexnet [24] as their feature extractor. VGG-16 is computationally expensive, which prevents real-time performance on high-resolution images. In contrast, Alexnet has relatively poor detection performance compared to current state-of-the-art networks. Our preliminary experiments have shown that the use of pretrained networks as feature extractors without retraining is not suited for the complex outdoor scenes in our dataset due to poor accuracy, especially in the presence of hard shadows. Instead of using a pretrained feature extractor network, it is possible to learn the important features per image by using auto-encoders [25–28]. This unsupervised method learns the features at test time. However, this is too slow for real-time performance, requiring minutes of processing time per image.

If training data are available, supervised networks can be used to learn change detection features directly from the training data. For static cameras, the CDnet dataset [29] is commonly used for this purpose, though for moving cameras no commonly used benchmark dataset exists. Non-static camera change detection is studied extensively in remote sensing [30–38] and mobile ground-based imagery [12, 13]. However, none of these works use pretrained weights to initialize their networks, even though it has a proven value for classification and object detection tasks, especially when only a limited amount of annotated data are available [39]. Our dataset is relatively small, hence pretrained weights are essential. Khan *et al.* [40] and Yang *et al.* [41] indeed exploit a pretrained network, but otherwise focus on weakly supervised training with computationally expensive components which are not suited for our purpose.

None of the methods mentioned in this paragraph pay particular attention to computational efficiency.

Because computational efficiency is crucial for real-time systems, existing change detection CNNs are less suited for our use case. Hence, we prefer building a network from more basic components, by borrowing techniques that have been proven to work well in other computer vision tasks. The research fields from which we draw inspiration for our network are classification, semantic segmentation, and patch matching.

Classification: Canziani *et al.* [42] provide an overview of speed considerations for recent popular classification networks. This overview supports a base network choice that operates in real time and may also achieve good change detection performance. It shows that especially ResNet variants [43] have high performance for relatively low computational cost. Deeper networks generally do not improve the detection rate of small objects [44], which makes efficient shallow networks especially viable for our use case.

Semantic segmentation: Change detection requires creating a pixel-level prediction based on image data, which makes it closely related to the field of semantic segmentation. Recent work focuses on segmentation networks that can operate in real time, e.g., LinkNet [45] or ERFNet [46]. These efficient networks achieve surprisingly good performance compared to state-of-the-art expensive semantic segmentation networks such as RefineNet [47] and DeepLab [48], or memory-intensive networks like DenseNets [49], which are incapable of fitting all feature maps in memory on most hardware for high-resolution images.

Patch matching: Patch matching is closely related to change detection, as it compares two patches and determines their similarity score, while ignoring irrelevant differences such as illumination, translation, and rotation. In general, three main architecture options exist for patch matching: Siamese, Pseudo-Siamese, and stacked (or "2-channel") networks [50]. Both Siamese and Pseudo-Siamese networks consist of two branches that each take an image patch as input, where Siamese networks exploit the extra assumption that the order of the two input images does not influence the change detection result. Stacked networks process image patches jointly as a single 6-channel "image" and are the most common in change detection literature [12, 13].

From the above discussion, we have concluded that designing a network based on classification, semantic segmentation, and patch matching is a viable approach because these aspects are considered essential for building a change detection network. Although broadly used, straightforward use of pretrained networks without fine-tuning is not adopted due to poor performance. Designing our own network from scratch will allow us to perform pixel-level change detection in real time.

The article continues as follows. First the design of the network is described, after which the loss function is elaborated upon. Third, we introduce our new post-processing approach. Next we describe the creation of the datasets, including the insertion of artificial changes. The results cover a speed versus performance trade-off, additional speed optimization, loss-function exploration, state-of-the-art comparison, and network generalization experiments.

2.1 *ESOCNet: Proposed Architecture*

This section presents the network architecture consisting of a Siamese encoder–decoder architecture. It is structured as follows. First, it describes a method for achieving efficient pixel-level features through a smart choice of encoder and decoder. Second, we show the extension of this basic network to make it specifically suited for change detection with the definition of an appropriate loss function. Third, a post-processing operation to suppress invalid detections is proposed, which is uniquely coupled to our network architecture. Finally, all parameters are listed for reproducibility.

The architecture is dubbed "Efficient Small-Object Change detection Network" (ESOCNet), and is an updated version of our previous ECDNet [15] architecture, for real-time change detection between aligned images. The network consists of a Siamese encoder–decoder architecture, as shown in Figure 2. The diagram portrays all feature maps with their corresponding resolutions and layer depths (result of the number of used filters in that layer) of the encoder, decoder, and integrated post-processing.

2.2 *Achieving Efficient Pixel-level Features*

Finding a good trade-off between computation speed, performance, and memory requirement is crucial to allow for real-time processing of high-resolution images. The most influential component for this trade-off is the encoder, which acts as a feature extractor. The choice of the decoder network is less influential, as it only upsamples the features back to original resolution.

2.2.1 *Encoder Network*

This work employs a ResNet-18-based encoder architecture, inspired by the accuracy versus speed trade-off analysis from [42]. In the proposed architecture, ResNet-18 is cut-off after the fourth residual block. This enables the detection of small objects, which may otherwise vanish due to excessive downsampling. The encoder choice is verified in the results section, through comparisons with deeper and shallower ResNet variants.

2.2.2 *Decoder Network*

A decoder is employed to expand the downsampled encoder features back to pixel-level features. Similar to ERFNet [46], we employ a decoder with fewer filters than the encoder, which improves speed at the expense of a small loss in accuracy. We draw inspiration from the decoder from ERFNet [46] for its simplified one-dimensional (1D) residual blocks, which results in the partitioning of the residual blocks into multiple 1D convolutions for faster processing at the cost of growth in memory. A comparison of regular ResNet-18 blocks versus non-bottleneck 1D blocks is shown in Figure 3.

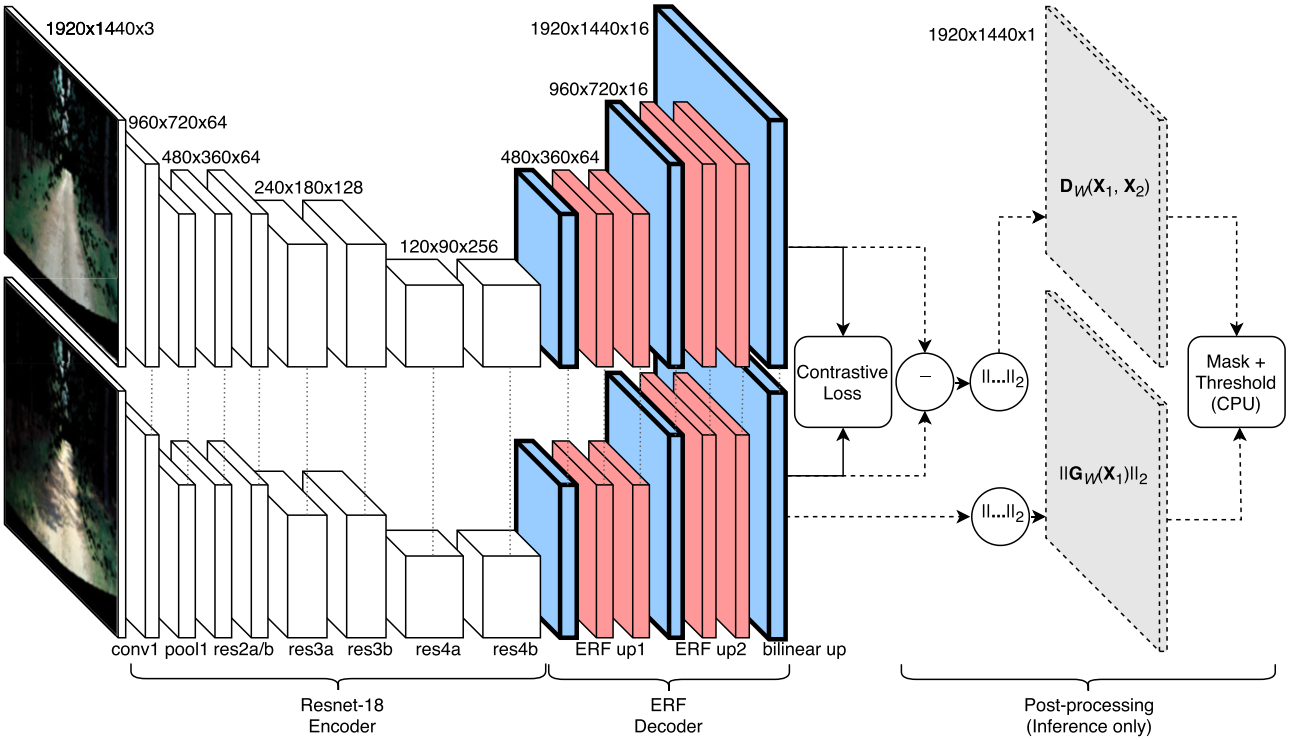


Figure 2. Proposed architecture for change detection (ESOCNet). White blocks refer to the encoder: ResNet-18, which is cut-off after the Res4b layer. Colored blocks highlight the decoder, consisting of bilinear upsampling blocks (blue, bold) followed by residual blocks (red). Top and bottom networks share parameters, indicated by vertical dotted lines. The numbers show activation map sizes after each block for default input size (1920 × 1440), but the network can operate on arbitrary image sizes. Dashed lines indicate that parts are only present during inference, not training.

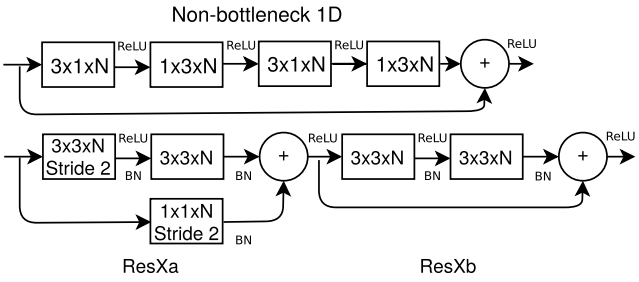


Figure 3. Non-bottleneck 1D block and regular residual block schematics. Rectangular blocks depict convolutions. In-place operations (ReLU or Batch Normalization (BN) layers) are shown on the arrows where applicable.

Here, any ResXa or ResXb block (where "X" stands for a number: 2, 3, 4, or 5) can be transformed to a non-bottleneck 1D block. Finally, we replace the deconvolution layers in the decoder by bilinear upsamplers and notice an improvement in both accuracy and speed, while simultaneously having fewer trainable parameters.

2.3 *Extending the Network for Change Detection*

A Siamese encoder–decoder architecture enables pixel-level change detection. In contrast to using a fixed pretrained network as a feature extractor, our network maximizes Euclidean distance separability between outputs, provided that it is used with an appropriate loss function.

2.3.1 *Loss Function*

The loss function that we employ to optimize Euclidean distance separability between changed and unchanged pixels, is the contrastive loss, which has been proven effective for patch matching [51]. Let $\mathbf{X}_1$ and $\mathbf{X}_2$ be two aligned input images shown to the system, and $\mathbf{G}_W(\mathbf{X}_1)$ and $\mathbf{G}_W(\mathbf{X}_2)$ the corresponding outputs of the Siamese network. The parameterized distance function $\mathbf{D}_W$ to be learned is then defined as the Euclidean distance between the outputs

$$\mathbf{D}_W(\mathbf{X}_1, \mathbf{X}_2) = \|\mathbf{G}_W(\mathbf{X}_1) - \mathbf{G}_W(\mathbf{X}_2)\|_2. \quad (1)$$

Here W refers to the learnable network parameters and $\| \bullet \|_2$ refers to the L^2 norm over channel dimension, resulting in a single-channel output image. To simplify notation $\mathbf{D}_W(\mathbf{X}_1, \mathbf{X}_2)$ is written as $\mathbf{D}$. This $\mathbf{D}$ can be viewed as a simple 2D difference image, similar to a difference image obtained by subtracting two images as commonly used in change detection. The output binary change mask $\hat{\mathbf{Y}}_{\text{thresh}}$ can now be computed by thresholding parameter $\mathbf{D}$ by some threshold T, as

$$\hat{\mathbf{Y}}_{\text{thresh}} = \mathbf{D} > T, \quad (2)$$

with $>$ the per-pixel boolean larger-than operator.

The contrastive loss from [51] can be re-defined to pixel-level operation to make it useful for change detection, similar to [34]. The pixel contrastive loss $\mathscr{L}(W)$ for an image batch of size P is then defined as a sum over pixel coordinates

(i, j) as follows:

$$\begin{aligned}\mathscr{L}(W) &= \frac{1}{P}\sum_{k=1}^{P} L(W, (\mathbf{Y}, \mathbf{X}_1, \mathbf{X}_2)^{(k)}) \\ &= \frac{1}{2P}\sum_{k=1}^{P}\sum_{i,j}(1 - Y_{i,j}^{(k)})(D_{i,j}^{(k)})^2 + Y_{i,j}^{(k)} \max \\ &\quad \times (0, m - D_{i,j}^{(k)})^2, \end{aligned} \tag{3}$$

where k is the image number in the batch, $L^{(k)}$ is the pixel-level contrastive loss of the kth image in the batch, $Y_{i,j}^{(k)}$ is the pixel label of the kth image at position (i, j), with 0 indicating no change and 1 denoting change, and $m > 0$ is a margin parameter beyond which changed pixels no longer influence the loss [51]. The margin parameter m can be chosen arbitrarily, as the network will accommodate itself to the margin, although a bad choice may impact convergence speed during training [52].

The above-defined regular contrastive loss suffers from three problems. First, as images cannot be aligned perfectly, for example, due to differing occlusion under different viewpoints, aligned images will have unmatched pixels. These pixels should receive a "don't care" label $Y_{i,j}^{(k)} = 2$. The loss function is altered such that pixels with this label do not contribute to the loss, regardless of the value of $D_{i,j}^{(k)}$, which is a simple masking operation on the per-pixel loss with valid mask $M_{i,j}^{(k)} = (Y_{i,j}^{(k)} \neq 2)$. The total loss is normalized to the number of valid pixels, to ensure that it does not depend on the number of "don't care" pixels and the input resolution.

Second, our dataset is severely imbalanced, where fewer than 1 out of 5000 pixels is a change pixel, hence class balancing is crucial. Class balancing is implemented as average frequency balancing on a per-image basis. To be partially invariant to object size, the balancing weights are computed per image instead of over the entire training set. This ensures that finding a single small object is equally valuable as finding a single large object in an image. This results in balancing weights w_C, where $C = 1$ for changed pixels and $C = 0$ for unchanged pixels. The balancing weights are defined for the individual loss terms by the inverse of the frequency of these pixels $(f_C^{(k)})$ as

$$w_C^{(k)} = \frac{K}{f_C^{(k)}} = \frac{0.5}{\epsilon + \frac{1}{N}\sum_{i,j}(Y_{i,j}^{(k)} == C)}, \tag{4}$$

where N is the number of valid pixels $(M_{i,j}^{(k)} == 1)$, K is a constant to avoid balancing for a perfectly balanced dataset (0.5 for 2 classes), and ϵ is a small constant to prevent division by zero.

Third, the margin only affects changed pixels. Unchanged pixels contribute to the loss, even if their distance is already close to zero, see Figure 4(a). This can lead to deteriorated performance due to compression of all unchanged pixels into a single point (overfitting), as has been shown for patch matching networks by Lin *et al.* [53]. Because of this behavior of the loss function, they propose a double-margin contrastive loss, which adds an additional margin to the left side of the loss equation, see Fig. 4(b). In contrast to [53], we leave the squaring operations at the same positions as in the regular contrastive loss. This slightly increases computation time, but prevents non-differentiable points (kink) and non-convexity in the loss graph. Lin *et al.* [53] also argue that both margins can be chosen equal, but if we do so with our smooth loss, the loss gradients vanish near the value of m, which makes thresholding unlikely to neatly split the two classes. This case is shown in Fig. 4(c) for our double-margin contrastive loss. Instead, we choose both margins to be unequal.

The final loss function now becomes as follows:

$$\begin{aligned}\mathscr{L}(W) = \frac{1}{2PN}\sum_{k=1}^{P}\sum_{i,j} M_{i,j}^{(k)} \\ \{(1-Y_{i,j}^{(k)})w_0^{(k)}\max(0, (D_{i,j}^{(k)})-m_1)^2 \\ + Y_{i,j}^{(k)} w_1^{(k)}\max(0, m_2 - D_{i,j}^{(k)})^2\}. \end{aligned} \tag{5}$$

Equation (5) shows two separate parts with their own balancing weights, one part for the changed pixels and one part for the unchanged pixels. As shown in Fig. 4(d), the unchanged pixels are no longer forced to zero distance, and the function is smooth and convex.

2.4 *Post-processing*

The network architecture is able to distinguish between objects added or removed from a scene, even though this information has not been annotated in the training data. The assumption is that, for a dataset with mostly uninteresting background and a few small changes in the form of objects, a single Siamese network branch learns a low-magnitude response for uninteresting areas and a high-magnitude response for potential suspicious areas (a sort of "objectness" score). Then, if $\|\mathbf{G}_W(\mathbf{X}_1)\|_2 > \|\mathbf{G}_W(\mathbf{X}_2)\|_2$, at some position (i, j), the change pixel will have been caused by strong features in the live frame that were not present in the reference. Network responses for which this does not hold can be suppressed, as we are generally not interested in objects removed from the scene. In case an object has been replaced by a different object, the relative feature-map magnitudes are no longer informative. To prevent filtering out an arbitrary half of these cases, all pixels of the live frame that have sufficient response by themselves are kept in the mask, regardless of the relative magnitude. This results in a post-processing mask $\mathbf{M}_{\text{post}}$ defined by

$$\mathbf{M}_{\text{post}} = (\|\mathbf{G}_W(\mathbf{X}_1)\|_2 > \|\mathbf{G}_W(\mathbf{X}_2)\|_2) \vee (\|\mathbf{G}_W(\mathbf{X}_1)\|_2 > T), \tag{6}$$

where symbol $>$ is again the per-pixel operator and T is the same threshold as applied on $\mathbf{D}$ in Eq. (2). The final binary change map is then computed as $(\mathbf{D} \odot \mathbf{M}_{\text{post}}) > T$. We specifically choose to use the same threshold as for the creation of the binary change mask, because in the most extreme case a value in $D_{i,j} = T$ is caused by

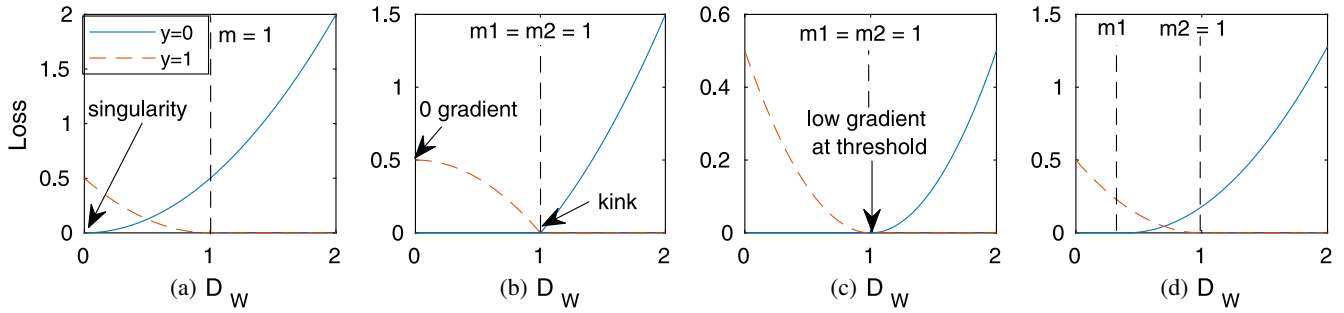


Figure 4. Loss functions and their imperfections. (a) Contrastive loss [51]: Unchanged pixels are compressed into a singularity. (b) Double-margin contrastive loss [53]: Gradient problems for low-distance change pixels, non-convexity and kink. (c) Proposed double-margin contrastive loss with equal margin. (d) Proposed double-margin contrastive loss with two different margins.

$\|\mathbf{G}_W(\mathbf{X}_1)\|_2 = T$ and $\|\mathbf{G}_W(\mathbf{X}_2)\|_2 = 0$ at that position (i, j). Hence identical thresholds guarantee that all changes caused by the live frame are preserved.

Considering that the final result will be thresholded at T after post-processing to achieve binary detection values, Eq. (6) can actually be implemented more efficiently by a simplified alternative, specified by

$$\mathbf{M}_{\text{post-alt}} = \|\mathbf{G}_W(\mathbf{X}_1)\|_2 > \mathbf{D}. \tag{7}$$

Although this does not result in the same $\mathbf{M}_{\text{post}}$, in practice it results in approximately the same binary change map after thresholding. The reason for this is that thresholding ensures, by definition, that only pixels for which $\mathbf{D} > T$ holds, remain in the final binary change map. Hence the $\|\mathbf{G}_W(\mathbf{X}_1)\|_2 > T$–part of Eq. (6) is automatically satisfied by Eq. (7). The rare exception in which Eq. (7) is slightly stricter, is that pixels for which $\mathbf{D} > \|\mathbf{G}_W(\mathbf{X}_1)\|_2 > T$ holds, are now also filtered out. In practice, the values in the 16-channel output are simultaneously high in all channels where an object is present, and low in all channels where there is no object, allowing $\mathbf{D}$ to be approximated by $\|\mathbf{G}_W(\mathbf{X}_1)\|_2 - \|\mathbf{G}_W(\mathbf{X}_2)\|_2|$. This means that in practice this rare exception only occurs when $\|\mathbf{G}_W(\mathbf{X}_2)\|_2 > \|\mathbf{G}_W(\mathbf{X}_1)\|_2 + T$ (i.e., a replaced object, for which the original object had a much stronger activation in the historic frame than the new object in the live frame). This case occurs so rarely that it has not appeared in any frame of our dataset. As for the left side of Eq. (6), $\|\mathbf{G}_W(\mathbf{X}_1)\|_2 > \|\mathbf{G}_W(\mathbf{X}_2)\|_2$, this adds nothing new to the mask. The only theoretical case where the addition would be useful, is when the following three cases hold: $\|\mathbf{G}_W(\mathbf{X}_1)\|_2 > \|\mathbf{G}_W(\mathbf{X}_2)\|_2$ and $\|\mathbf{G}_W(\mathbf{X}_1)\|_2 < \mathbf{D}$ and $\mathbf{D} > T$, which, due to the channel property described earlier in this paragraph, does not occur in practice either. Now the expensive $1920 \times 1440 \times 16$-sized L^2 norm computation $\|\mathbf{G}_W(\mathbf{X}_2)\|_2$ is avoided.

2.5 *Network Parameters*

All networks are trained with the dual-margin contrastive loss as defined earlier. The Adam optimizer [54] with learning rate 10^{-4} is used. We initialize the encoder with ResNet-18 weights pretrained on ImageNet and the decoder with Xavier initialization [55], since it empirically outperforms the more recent initialization of He [56] in our setup. Networks are trained for 10 epochs, since this appears to be sufficient for our relatively small dataset, when using pretrained weights.

Batch size is set to unity to allow a high input image resolution. To cope with limited memory, 512×512-pixel crops are used during training for all network comparisons unless otherwise specified. The crops are chosen such that at least one change pixel occurs in each training crop to speed up training. In addition, on-the-fly data augmentation in the form of random horizontal mirroring is applied. Batch normalization layers in the encoder (with pretrained weights) are allowed to be updated in a moving-average sense. At test time, the final network can execute on full resolution of 1920×1440 pixels, because networks require less memory when backpropagation is not required. A summary of our parameters is shown in Table I. These parameter settings are used for all experiments unless explicitly noted otherwise.

3. EXPERIMENTAL RESULTS

This section describes the experimental validation of both our network and several state-of-the-art alternatives. In this work, all networks are implemented in the Caffe framework [57] and the reported test times are achieved on a GTX 1080Ti. For inference, we also port our network to the TensorRt framework for improved performance and experiment with an RTX Titan to obtain a 16-bit floating-point computation speed enhancement (unsupported by GTX 1080Ti). The first three subsections provide information on the dataset and evaluation metrics. Afterwards, the following experiments are described in separate subsections: (1) investigation of the impact of architectural choices on the speed and accuracy of the network; (2) measuring the effect of several additional speed optimizations; (3) impact assessment of our changes to the contrastive loss function and comparison to three other commonly used losses; (4) comparison to state-of-the-art methods; and (5) experiments to improve understanding network behavior, including (a) generalization to different environments, (b) generalization to different object types, and (c) the effect of training and testing with artificially inserted object changes.

Table I. Default network parameters.

Parameter	Value
Training parameters	
Batch size	1
Crop size	512 × 512
Epochs	10
Encoder initialization	ImageNet Pretrained
Decoder initialization	Xavier
Adam optimizer	
Learning rate	10^{-4}
β_1	0.9 (Caffe default)
β_2	0.999 (Caffe default)
ϵ	10^{-8} (Caffe default)
Contrastive loss	
m_1	0
m_2	1.0
Test-time parameters	
Crop size	1920 × 1440
Change threshold T	1.3

3.1 *Data Acquisition*

The training and validation dataset consists of four pairs of videos (1882 image pairs), with a resolution of 1920 × 1440 pixels, captured in forest-like environments. A single video pair features a recording prior to the placement of test objects (the reference (historic) images) and one afterward (the live images). The 57 unique test objects consist of, among others, colored blocks, bottles, rope, and a large tree trunk, which serve as the changes to be detected. Reference images are aligned to their corresponding live images, as in [6], but without optical flow refinement. This alignment cannot match all pixels, hence the unmatched pixels are masked out (as will be shown in Figures 7(a) and 7(b)). Colored boxes depict interesting areas of the image, namely objects added to the scene (light blue), objects removed from the scene (green), object-like image regions present in both images (yellow), and red boxes which are explained later.

A separate dataset consisting of three videos (1,884 image pairs) is employed to test the system and highlight the strengths and weaknesses of the network. This involves video pairs with the following properties: (1) video pair obtained by driving twice the exact same trajectory, resulting in a pair of videos with practically no alignment errors ("Test Easy Objects Small Misalignment"); (2) video pair with a deliberate 5-meter offset in driver path between live and reference videos, causing larger alignment artifacts ("Test Easy Objects Large Misalignment"); (3) video pair in a dunes environment instead of a forest environment, which features different objects compared to the training set ("Test Hard Objects"). The three test videos contain 50 new unique objects.

3.2 *Dataset Creation*

The datasets consist of the videos described in the previous section, with ground truth annotated at pixel level. Furthermore, for some experiments, the datasets are augmented with artificial objects as changes. Both aspects are now discussed.

3.2.1 *Semi-automatic Ground-truth Annotation*

Ground-truth changes are manually labeled by specifying bounding boxes around such changes. The bounding boxes are then automatically converted to pixel-precise annotations with a simple algorithm based on SNIC superpixels [58], where incorrect pixel-level annotations are manually corrected. Our segmentation algorithm extracts a fixed number of SNIC superpixels in a local neighborhood around each bounding box, where the neighborhood size is a multiple of the bounding-box size. Pixels inside the bounding box are set to "change" if they are part of a superpixel that sufficiently overlaps with the original bounding box. All other pixels in the bounding box are set to "don't care," to prevent small label errors from impacting the training. Furthermore, all pixels that are not matched by the alignment are set to "don't care." The assumption is that the object in the bounding box is approximately the same size or slightly smaller than the bounding box itself, and distinct from its immediate background. Objects for which this assumption does not hold may result in errors, which are manually corrected.

The pixel-level annotated train/validation dataset is split randomly into a training set (80%) and validation set (20%). After this split, training-set images without any change are discarded, which leaves the final numbers at 981 training image pairs (all with at least some change) and 376 validation image pairs (most with change, some without any change).

3.2.2 *Introducing Artificial Objects*

Convolutional neural networks (CNNs) benefit from having more training data available, but manually recording videos with and without changes and annotating them is immensely time-consuming. Additional training data can be obtained by inserting artificial objects into real scenes, using either rendered or real objects. Blending real objects into real scenes has been proven effective for instance detection [59], hence a similar technique is applied to augment our dataset. For this purpose, object cut-outs are obtained from the COCO2017 dataset [60], using the provided polygon annotations as cut-out masks. Objects that are truncated at the edge of an image, or disjoint, are discarded. Each resulting object cut-out (blob) is then blended into the scenes using one of the multiple blending methods (as in [59]): direct pixel pasting, Gaussian blurring, or Poisson blending. We also test modified Poisson blending [61], since it better preserves original object colors than regular Poisson blending. Example images of all blending methods are shown in Figure 5. Relevant parameters for reproducibility are shown in Table II.

Employing a different blending method for each object cut-out ensures that it is harder for the network to overtrain on recognizing the blending method itself, instead of the

(a) Pixel paste (b) Gaussian blur (c) Poisson blending (d) Modified Poisson blending

Figure 5. Visual example of the 4 employed blending methods.

(a) Live image with real and artificial changes, and distractors.

(b) Reference image with only distractors.

(c) Ground truth with only real and artificial changes. Black is 'no change', grey 'don't care' and white 'change'

Figure 6. Visual example of an image augmented with artificial objects taken from the COCO dataset. Green boxes show the position of the five artificially inserted changes, blue boxes show two real changes, red boxes show the five distractors. Best viewed in color and online.

Table II. Non-default blending parameters.

Blending method	Parameter	Value
Gaussian blur	Kernel size	5 × 5
Poisson blending [62]	Method	non-mixed
Modified Poisson blending [63]	Color_param	10^{-3}

object that is blended into the scene. Besides employing multiple blending methods, we also add "distractor objects" to the scene. This further discourages the network to learn to recognize the blending method. Distractor objects are objects added to both the live and reference image of a scene, but using a different blending technique. Such objects are marked as "no change" in the ground truth, while regular artificial objects are only added to one of the two images and marked as "change" in the ground truth. An example augmented image pair is shown in Figure 6.

The artificially augmented dataset consists of the same four videos used for training, but augmented with five artificial changes and five distractors per frame. Not all COCO classes are deemed representative of the objects that we search for, hence only the following classes are used: bottle, vase, cup, backpack, handbag, suitcase, sports ball, and baseball bat.

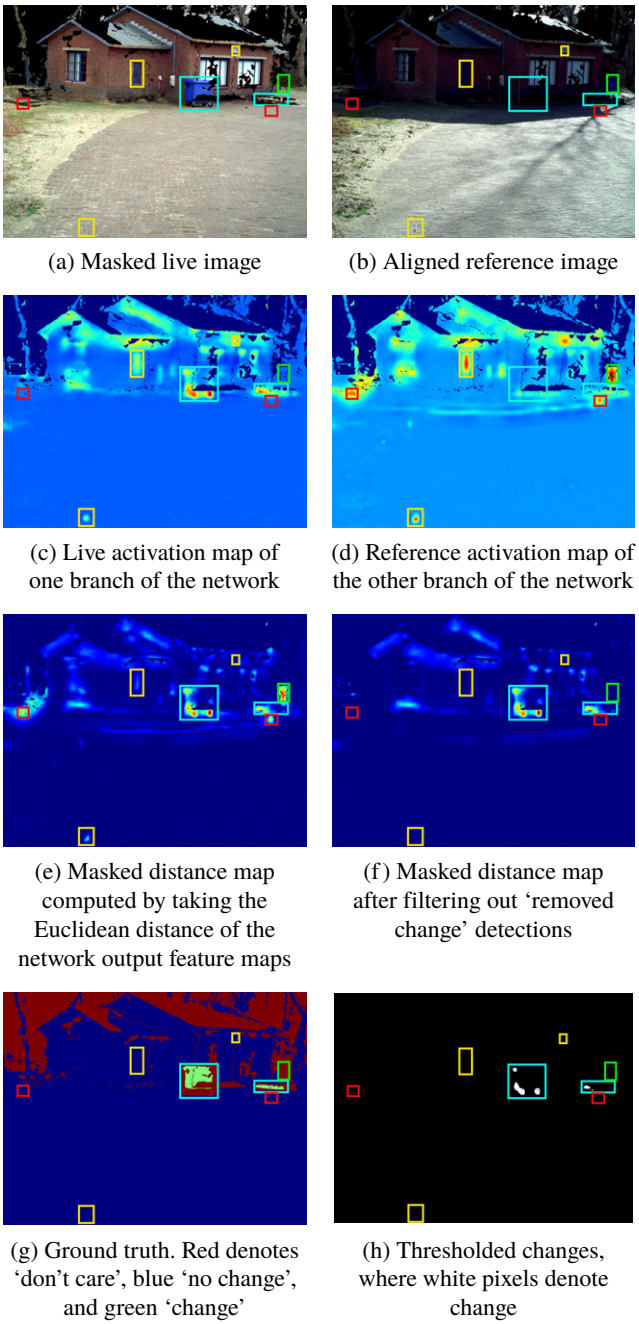
3.3 *Evaluation Metrics*

The effectiveness of change detection networks is generally reported via the F-1 score or Intersection over Union, both computed at pixel level. For fairest comparison to other networks, we also report the pixel F-1 score. In addition, we also report object F-1 scores, based on the number of changed objects detected. In the context of IED detection, the number of correct objects is a far more relevant metric, since it is important to find all objects without being biased to larger objects. The F-1 score is computed by

$$F_1 = \frac{2 \cdot \mathrm{Pre} \cdot \mathrm{Rec}}{\mathrm{Pre} + \mathrm{Rec}}, \tag{8}$$

where Rec denotes the recall which is the fraction of real pixel/object changes that is correctly detected and Pre represents the precision, which is the fraction of the total pixel/object detections that is correct.

For qualitative evaluation of generalization capability, several steps in the algorithm are visualized for an exceptionally challenging test image in Fig. 7. The network has been trained exclusively on forest environments and dirt roads, hence it has never seen paved roads, buildings of any kind, or anything resembling the changes in this image (a blue container and a wooden pellet). The L^2 norm of the activation maps of the live and reference images, and the resulting distance maps are shown in Figs. 7(c), 7(d) and 7(e),



(a) Masked live image

(b) Aligned reference image

(c) Live activation map of one branch of the network

(d) Reference activation map of the other branch of the network

(e) Masked distance map computed by taking the Euclidean distance of the network output feature maps

(f) Masked distance map after filtering out 'removed change' detections

(g) Ground truth. Red denotes 'don't care', blue 'no change', and green 'change'

(h) Thresholded changes, where white pixels denote change

Figure 7. Visual example of a challenging test frame pair and associated processing results. Red boxes show false-positive detections, light blue boxes indicate true-positive changes appearing in the live frame, green boxes show true-positive changes due to the reference. Yellow boxes show object-like "things" present in both frames. Figure is best viewed online in color.

respectively. An example of the effect of the post-processing is shown in Fig. 7(f), which can be compared to Fig. 7(e) to see that several false-positive detections are removed from the change map. An example of the final detection output after thresholding is shown in Fig. 7(h), where white blobs indicate the final detections.

3.4 *Speed versus Performance Trade-off*

There are three main choices that affect the speed and performance of the network, in descending order of importance: the choice of encoder network, the cut-off point of the encoder, and the choice of the decoder network. To achieve an extensive trade-off analysis, a parameter space exploration over all combinations of these three architectural choices is performed.

Encoder: As mentioned in [42], ResNet-18 obtains a good speed-performance trade-off for classification. However, this does not ensure good performance for change detection, hence both deeper and shallower encoder networks are considered in the exploration: ResNet-10, ResNet-18, ResNet-50, ResNet-101, ResNet-152. For the deeper networks, inference (not just training) has to be performed on 512×512 blocks to fit in memory.

Network cut-off point in the encoder: The network cut-off point affects the total number of parameters, the amount of downsampling prior to the decoder, response smoothness and localization, receptive field, and computation time. For the exploration, each point immediately after a residual block of the ResNet architecture is considered as a potential cut-off point. Skip connections supposedly allow for deeper architectures that maintain detailed pixel-precise outputs. However, we find that skip connections affect performance similarly to cutting-off the network earlier, without the advantage of lower computational cost.

Decoder: Only two decoder options are investigated, the entire ERFNet decoder and a simple bilinear upsampler, because it has a smaller impact than the other hyperparameters.

The results of the parameter space exploration are shown in Figure 8. Several interesting interpretations can be made from Fig. 8(a). First, deeper networks do not necessarily perform better on our dataset, likely because of both the small size of the objects and the small number of image pairs in the dataset. Interestingly, the deeper networks do not even necessarily achieve *lower training loss* after convergence, hence overtraining is not the issue. The deepest networks (those whose timings do not fit on the axes in Fig. 8) also require smaller training crops to fit in memory, which cause them to exploit less context during training, so that they perform worse. Second, at first sight, it may appear that blockwise processing is the cause of reduced F-1 scores. However, this is unlikely, since applying it to the ResNet-10 or ResNet-18 networks reduces F-1 scores by only 0.1%. Blockwise processing primarily affects execution time negatively. Third, the inclusion of the ERFNet decoder blocks improves performance in all cases over a single bilinear upsampler at the end of the encoder (for the validation video). However, the results for the "Large Misalignment" test video, see Fig. 8(b), do show several cases of reduced performance. This may result from weaker generalization of the decoder because it is trained from scratch. Fourth, the red-filled square (our final choice) is the best in terms of F-1 score only on two of the three test videos, where the exception is shown in Fig. 8(b). This possibly indicates overtuning of hyperparameters to the validation set, although the effect

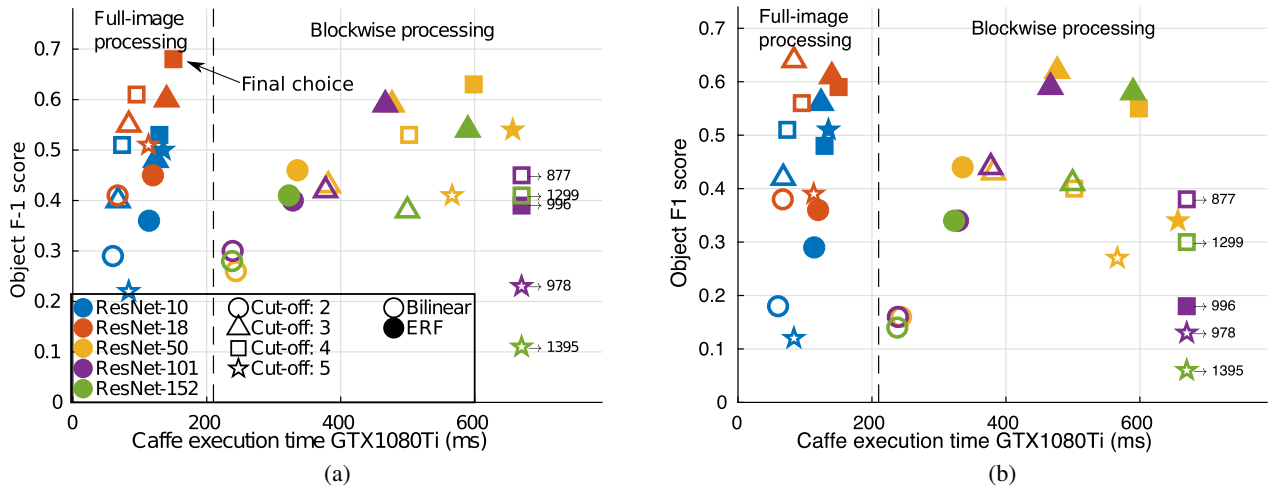


Figure 8. Performance trade-off and comparison of various architectural choices of the encoder (marker color), encoder cut-off point (marker shape), and decoder (marker fill). The final network is denoted by a red-filled square. (a) Results evaluated on the validation set. (b) Results evaluated on the large misalignment test video. Best viewed in color.

is much less severe for the other two test videos (not shown due to space considerations), where our architecture choice is still optimal, but with a smaller margin than in the validation videos. Summarizing, the red-filled square (i.e., ResNet-18 encoder cut-off after the 4th residual block with the ERF decoder) results in the highest F-1 score, while simultaneously achieving a fast execution speed, thereby offering the best trade-off.

3.5 *Optimizing Speed Further*

Because low latency is crucial for real-time detection of IEDs, several improvements have been implemented with respect to [15] to reduce the 150-ms latency of the optimal network and its CPU overhead. First of all, the expensive computations for post-processing (computing $\mathbf{D}$ and $\|\mathbf{G}_W(\mathbf{X}_1)\|$ of Eq. (7)) can easily be performed on a GPU by integrating them into the inference network, instead of performing them separately on CPU. Second, a significant execution time reduction can be achieved by optimizing the network architecture specifically for the hardware on which it will execute. This can be done using the TensorRt tools (TensorRt is commercially available from NVIDIA: https://developer.nvidia.com/tensorrt).

Since TensorRt does not support all Caffe layers for conversion, specifically the slice layer and the element-wise subtraction layer for our network, some changes to the network have to be made to circumvent those original layers without changing the resulting output. The original Caffe network takes a single 6-channel image as input, for easier lmdb-based training, followed by a slice layer to obtain two 3-channel images. In the TensorRt inference network, the slice layer is circumvented by switching to two 3-channel inputs. Next, the summation over channel dimension required for the integrated post-processing was originally implemented as a slice layer followed by an element-wise addition layer. This is now replaced by a 1×1 convolution with fixed weights of unity. For the creation of the difference image, we use an element-wise summation layer in Caffe. Whereas Caffe's element-wise layer also supports subtraction, TensorRt does not. Therefore, this layer has been replaced by two different layers: a scale layer which multiplies one of the two Siamese branches by -1, followed by a regular element-wise sum layer which sums the output activation maps of both branches.

Applying TensorRt optimizations to our network reduces the computation time to 60 ms on the GTX 1080Ti. Another common speedup method is doing computations in 16-bit floating-point precision on graphics cards that support this. The GTX 1080Ti does not support such computations, hence for this purpose an RTX Titan card is used. Its inherent higher speed already results in an execution time of 41 ms on TensorRt-optimized 32-bit floating-point computation, and using 16-bit precision reduces that further to 27 ms. Using 16-bit precision may impact the accuracy of the predictions, but we have found the difference to be negligible on our dataset. The impact of each speed optimization is shown in Table III. After all optimizations, the remaining CPU overhead of 26 ms for reading the images becomes now the main bottleneck and is difficult to optimize further.

3.6 *Impact of the Chosen Loss Function*

The effectiveness of our alterations to the contrastive loss is evaluated in an ablation study. The results are portrayed in Table IV, which shows that class balancing is crucial for good performance. At first glance, it appears that disabling "don't care" masking slightly improves performance, albeit by less than one standard deviation. However, disabling "don't care" masking slows training convergence by almost a factor two, hence enabling it is still a valid improvement. The double-margin addition to the loss function provides no clear improvement. Thus, in a pixel-level change detection setting, the singularity problem as mentioned in [53] is apparently much less influential, probably due to the large number of different background pixels in the dataset. Hence, by Occam's

Table III. Impact of speed optimization. GPU: RTX Titan. CPU: Gold 6146 Xeon(R) @ 3.20 GHz.

Optimization step	GPU time (ms)	CPU overhead (ms)	Total time (ms)
Original network [15] GTX 1080Ti (Caffe)	150	132	282
Original network [15] RTX Titan (Caffe)	110	132	242
Difference image integrated (Caffe)	112	106	218
Post-processing integrated (Caffe)	121	26	147
Optimized post-processing (Caffe)	119	26	145
Optimized with TensorRt	41	26	67
16-bit floating-point precision (TensorRt)	27	26	53

razor, maintaining a single margin is the best solution with fewest parameters.

Next, the proposed contrastive loss is compared with commonly used losses, i.e., the softmax cross-entropy loss, L^1-norm loss, and L^2-norm loss. To be able to apply these losses, both branch outputs are concatenated, resulting in a single 32-channel tensor, followed by a 1×1 convolution to reduce the number of output channels to unity. This single-channel output is then compared to the ground-truth binary change map with one of these three loss functions. The L^1- and L^2-norm loss are thus not to be confused with L^2 metric defined in Eq. (1), as they are computed between different tensors. For the loss-function comparison, the loss functions are used as a complete replacement of our proposed contrastive loss. Table IV shows the results, where all scores are computed with "don't care" masking. As we expected, the contrastive loss is more powerful for the current architecture. For reproducibility of our results, the equations of the applied softmax cross-entropy loss, L^1-norm loss and L^2-norm loss with our "don't care" masking are as follows:

$$\mathscr{L}_{Cross-Entropy}(W) = \frac{1}{PN}\sum_{k=1}^{P}\sum_{i,j} M_{i,j}^{(k)} \times \left(-Y_{i,j}^{(k)} \log \sigma(Y_{predicted(i,j)}^{(k)})\right) \quad (9)$$

where σ is the softmax function, which normalizes the network output to the unity interval,

$$\mathscr{L}_{L1}(W) = \frac{1}{PN}\sum_{k=1}^{P}\sum_{i,j} M_{i,j}^{(k)} |Y_{i,j}^{(k)} - Y_{predicted(i,j)}^{(k)}|, \quad (10)$$

$$\mathscr{L}_{L2}(W) = \frac{1}{PN}\sum_{k=1}^{P}\sum_{i,j} M_{i,j}^{(k)} \left(Y_{i,j}^{(k)} - Y_{predicted(i,j)}^{(k)}\right)^2. \quad (11)$$

3.7 *Comparison to the State of the art*

The proposed network is compared to the state of the art in both conventional methods and CNNs in Figure 9. The figure depicts (1) a baseline difference image-based change detection method of Van de Wouw *et al.* [6]; (2) CDNet by Sakurada *et al.* [12], which is a stacked Architecture; and (3) a Siamese network by Zhan *et al.* [34], who also presented a Siamese change detection architecture. The implementations are performed as follows.

Table IV. Ablation study for the proposed contrastive loss function.

Loss-Function Comparisons	Object F-1 Validation	Object F-1 Test
Baseline: Our contrastive loss	0.67	0.60
no class balancing	0.22	0.17
no "don't care" masking	0.69	0.60
no double margin	0.67	0.60
Softmax cross-entropy loss	0.49	0.29
L^1-norm loss	0.10	0.17
Euclidean (L^2-norm) loss	0.01	0.06

Van de Wouw's Adaptive Threshold YUV: Van de Wouw's method consists of adaptive thresholding on a difference image in YUV color space [6] after refinement of the alignment through optical flow. The results for this method reported in the current work are poorer than the original reported results, because our dataset is considerably more challenging than those used in their experiments. Note that the test sets in Fig. 9 have no error bars for this method because the method is not trained, hence cross-validation has no effect on test accuracy.

Zhan's Siamese Network: The network from [34] is implemented and altered to improve its performance on our dataset. We have used our improved contrastive loss with class balancing and "don't care" handling and have tuned their parameters to our dataset. Their network also results in noise pixels in the change map, which we remove using an additional morphological filtering step.

Sakurada's CDNet: CDNet [12] is implemented without the additional optical flow input, which according to their article should only change performance by a few percent. The following additional design assumptions are made for parameters not explicitly mentioned in [12]: concatenate skip connections, ReLU in decoder, convolution padding to maintain feature-map size, default dropout parameters, and the Adam optimizer [54] with default parameters. Furthermore, to achieve good results on our dataset, the

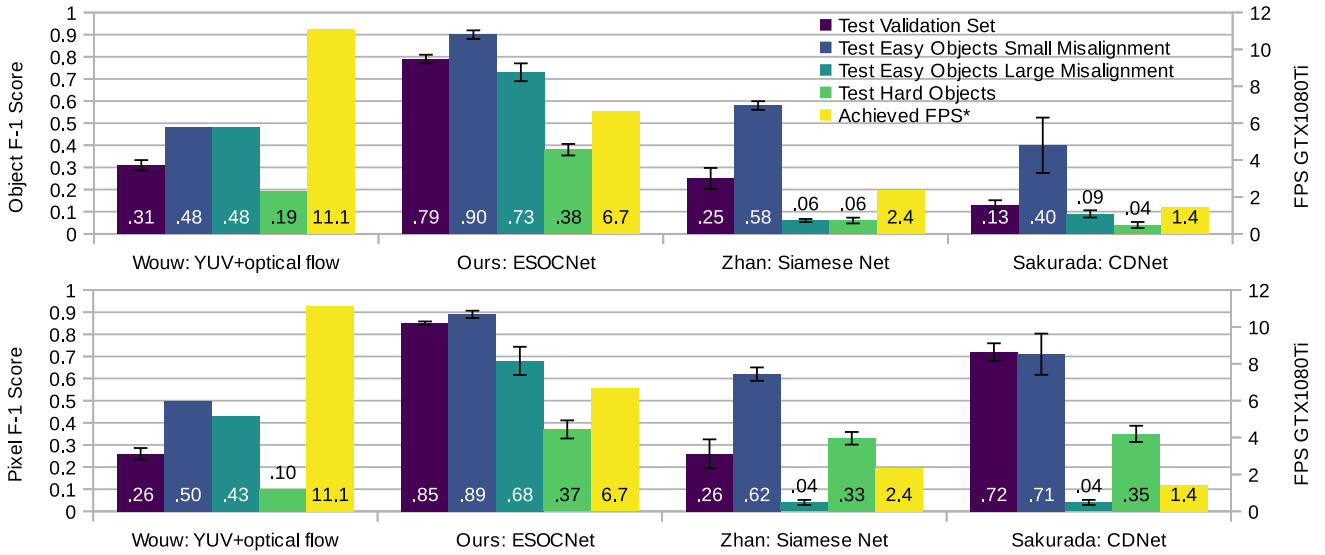


Figure 9. F-1 scores and execution speeds on a GTX 1080Ti of our network versus the performances of state-of-the-art methods when applied to our validation dataset, sorted on speed. *Execution times for Zhan and Sakurada is computed using overlapping 1024 × 1024 blocks due to memory constraints. With sufficient memory, the fps values would increase to 3.7 and 2.2, respectively.

following additions are necessary: morphological filtering, replacement of the L^1-norm loss by the softmax cross-entropy loss, removal of the deepest three layers, and training for four times as many epochs compared to our network.

The resulting F-1 scores and execution speeds in frames per second (fps) on a GTX 1080Ti are shown in Fig. 9. The error bars indicate standard deviations for five-fold cross-validation. The proposed approach outperforms all earlier work in terms of F-1 score, while having a smaller computational cost than other CNN approaches. For this test, 1024 × 1024 crops have been used to train our network to highlight its memory efficiency with respect to the other networks. If the same 512 × 512 crops are used in all networks, the object F-1 score of our network drops by 0.03–0.11 depending on the video, but it is still the highest score for all videos.

Both Zhan's and Sakurada's networks perform poorly under large misalignment errors, due to the vast numbers of false positives. In contrast, our network performance is not strongly affected by alignment errors, since a single network branch approximately falls back to being an "objectness" detector provided that the misalignment is large. For example, neither grass nor road result in "objectness," hence poor alignment that causes these two to overlap, does not result in false positives in our network.

The F-1 score of the proposed network on the "Hard Objects" video is similar to Zhan's and Sakurada's networks, because the network has a difficulty in finding all pixels of a large change. This means that our network likely contains less scale invariance than Sakurada's and Zhan's networks. The pixel F-1 scores of their networks on the "Hard Objects" video are high due to a few well-detected large objects. Generally, Sakurada's network is better at finding an accurate object outline, while our network is better at finding objects as blobs, not necessarily with an accurate outline.

Interestingly, the image differencing of Wouw still performs reasonably well on the "Large Misalignment" videos, despite even optical flow not being able to repair these alignments. This we attribute to the objects in those videos having bright colors that are clearly different from the entire environment. Finally, while the method of Wouw is significantly faster than our base network, it cannot be optimized further using TensorRt. Hence, after speed optimizations, our network is both the fastest and best-performing method compared to all three state-of-the-art alternatives.

3.8 *Understanding the Network Behavior*

Up to this point, all experiments have focused on network performance for the entire validation and test sets. This section describes various experiments that provide insight into specific situations, such as when and why does the network perform suboptimally and the possibility of alleviating these situations with artificial data.

3.8.1 *Difference with Object Detection*

For the proposed network, it is possible to use a single branch of the network and ignore the output of the other, resulting in a kind of object detector, or more precisely, semantic segmentation with only 2 classes: "object" and "not object." Comparing the use of the network as an object detector versus the change detection mode shows that change detection achieves higher object F-1 scores than pure object detection by as little as 5% for the "Large Misalignment" video, up to as much as 21% for the "Hard Objects" test video. Manual inspection of the frames reveals that false positives are primarily caused by non-objects looking like objects in either the live or reference frames, but not in both. For example, a small white high-contrast patch of sky between tree branches can be present in the live frame

but not in the reference frame, due to an alignment error. Such types of false positives indeed cannot be filtered by comparing "objectness." Furthermore, certain shapes of sharp hard-shadow edges are sometimes detected as false positives, though in general the network is robust to lighting differences and shadows. These results appear to confirm our opinion that the network has primarily learned to be an "objectness" detector that simply compares the "objectness" score between two images to determine changes. This also explains the robustness of the system to most dynamic backgrounds.

There is a type of environmental change that is relevant in IED detection for which this "objectness" comparison may fail: replaced objects. These are objects that were initially present in the reference frame and were replaced by a different object at the exact same location in the live frame. However, this situation does not occur in any of our recorded datasets. To verify this potential failure mode, the replacement of objects in the real world is simulated by placing artificial objects, not in the live frame, but in the reference frame, at the same location as the real objects occurring in the live frame. This indeed causes the object F-1 score to drop from 0.66 to 0.36 on the validation set. Performing the same experiment but now also adding artificial "replaced objects" to the training set improves this score to 0.48. These relatively low scores reinforce the hypothesis that the normally trained network primarily works with a difference in "objectness" score at the same location in the image, which is small when an object is present in both the live and reference frame. The gain achieved with artificial "replaced objects" in the training set seems to indicate that the network *can* learn to find these types of changes, but the original training dataset simply lacks replaced objects.

3.8.2 *Change Object Generalization*

The main difficulty of detecting IEDs is that markers come in many forms, which means that the network should be capable of detecting object classes it has never seen during training. This generalization capability of the proposed network is investigated by splitting the changes into three broad categories: block, beverage object (bottles and cans), and other. To prevent the amount of training data of having an impact on the results, all training sets in this experiment contain an equal number of training frames. Training and testing on the same category is excluded, since it would require using a relatively small number of training image pairs to prevent overlap in train and test sets. The results are shown in Figure 10.

Blocks, bottles and cans are relatively similar in appearance, as can be seen from the examples in Figure 11. This is reflected by how well the network performs when training with one and testing on the other. The "other" class, which contains many different kinds of objects, suffers substantially when not enough similar-looking objects occur in the training set. Note that objects in the "other" set are often much harder to detect by a human as well, since it contains changes such as tree trunks and camouflage nets instead of colored blocks. It is not surprising that training with all classes performs best.

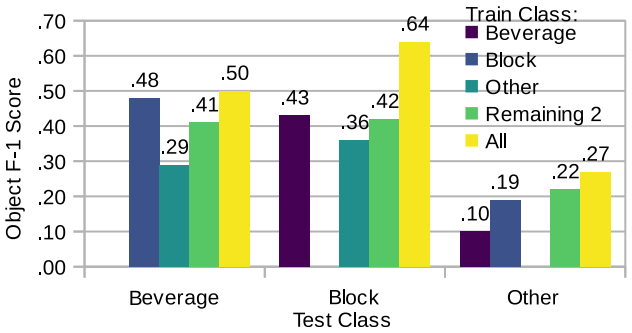


Figure 10. F-1 test scores on different types of changes using different change types for training. Bar colors refer to the classes used for training. "Remaining 2" refers to training with the other two classes than the test class and "All" refers to training with all three classes.

3.8.3 *Artificial Object Insertion*

To investigate whether training with artificial data can improve the generalization of the network, we add artificial data to the regular training set using the four different blending methods, discussed earlier and shown in Fig. 5. Three types of experiments are performed: training with purely artificial changes, training with both real and artificial data and training with only real data. Training with purely artificial changes means setting the labels of real changes to "don't care," since, we need to use the same frame pairs for a fair comparison. The results are shown in Figure 12. The four test datasets in this experiment are chosen to highlight different aspects of training with artificial data. First, the validation set provides a reference score for other experiments in this work. Second, the "Large Misalignment" test set shows how well the network performs when falling back to its "objectness" scores. Third, the "Dissimilar Real Class" test set is a subset of the "Hard Objects" test video with only the "other" subclass of objects, which shows how well the network generalizes to difficult never-seen object types in unknown scenes. Finally, the validation set with artificial changes shows that the method can reasonably well detect artificial object changes in scenes that are also present in the training set.

We have made four observations from Fig. 12. First, training with only real data allows the network to perform almost as well on detecting artificially inserted changes as real changes (compare the leftmost and rightmost bars of "All Real"). This implies reasonable generalization to unseen objects of classes that are slightly different from the training classes, in known scenes. Second, the score on "Dissimilar Real Class" is not improved by training with artificial data (the green bar of "All Real" shows a higher score than any of the blending methods). Possibly, the artificially inserted changes are still not sufficiently representative for these unseen object types. For example, the camouflage net change in this set is unlike any of the classes in the COCO dataset.

Figure 11. Examples of the object classes from the dataset. Top: Beverage objects. Middle: Block objects. Bottom: Other.

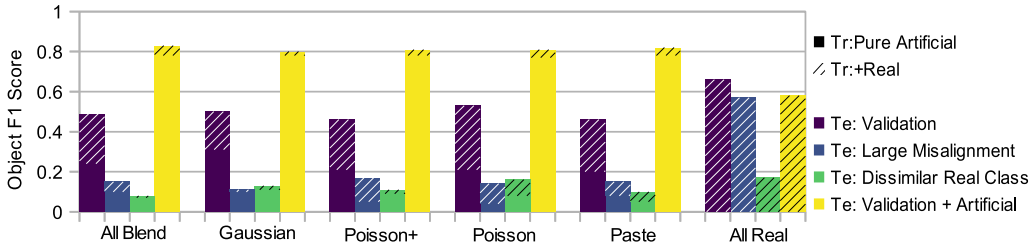


Figure 12. Results of training and testing on artificially inserted changes in the scene, for various different blending methods. Colors with no hatching refer to training with a dataset with purely artificial changes, while hatching shows the score improvement of using real data on top of artificial data. The "All Real" experiment is the only experiment where no artificial data has been used. "All Blend" refers to training with artificial objects blended randomly using one of the other blending methods.

Third, the "Large Misalignment" video suffers severely when training with artificial data. This can be due to more environmental changes causing false positives, since they may look similar to some of the artificial object types and different from the original real object types. From this, we conclude that our method is only powerful for poorly aligned images if the object types in the training set are representative for the true object types to be detected. Detections on well-aligned images are more robust to this problem, as "objectness" detections in the same location in both frames cancel out. Fourth, training with purely artificial data always yields poor performance on detecting real changes.

4. DISCUSSION

Despite its good performance on our dataset, the trained network also contains several limitations. First, the network is closer to an object detector than a pure change detector, since the decision for change is mostly based on "objectness." This causes the network to fail in recognizing the change type "object replaced by a different object," because both objects will have a high "objectness" score. The network has never learned this change type, as it is not present in the training set. An experiment of adding artificially inserted replacement objects into the training set already achieves some improvement in the score (Object F-1 score from 0.36 to 0.48), thereby indicating the network is able to learn these changes as well, given sufficient representative training data.

Second, almost all objects in the dataset are non-natural objects that would normally not belong to the scene setting. This is not necessarily the case for general IED markers in practice. It is unlikely that the trained network will perform well on more natural changes, since the current ground-truth annotations mark all natural changes as "no change," which means that the network is rewarded for suppressing them actively. Future work could investigate this hypothesis and test whether training with more natural objects in the dataset addresses this problem.

Third, the performance difference between a broadly general method (YUV-space image differencing) and our trained neural network becomes smaller, the more the test data deviates from the training data. For example, the current training data are recorded in forest and dune environments, under sunny or overcast weather, with a specific set of representative objects. When the system is employed in a different environment (e.g., urban), under weather conditions that strongly change the appearance and contrast of the objects compared to the training data (e.g., heavy rain, night, snow) or when applied on different

objects, the performance will deteriorate. This type of issues can largely be resolved by gathering training data in the corresponding settings, which should make the network invariant to variations that are not an inherent property of the objects, given sufficient data. This means that representative training data is crucial. Purely artificial object data appears to be insufficient for this purpose. In conclusion, expensive manual collection of real-world data is still on short term required to achieve the high performance generalized over new object classes reported in this work. However, in the future, artificial data in the form of extensive data augmentation by mimicking weather conditions and their corresponding luminance and color variations, could still offer significant value for improving the robustness of the detection system.

Finally, while some of our improvements to the contrastive loss are powerful, the addition of the second margin is non-contributing for change detection. Since the single-margin loss is both simpler and achieves equal results (for pixel and object F-1 scores), employing a single margin is recommended.

5. CONCLUSION

We are the first to propose a neural network architecture for small object-like change detection from high-resolution ground-based vehicle imagery in real time, including the following contributions. First, we have designed an efficient Siamese encoder–decoder network based on ResNet-18 and ERFNet that significantly outperforms state-of-the-art change detection networks for IED marker detection. The network is also applicable to general small-object change detection use cases. Second, we have developed extensions to the contrastive loss function, by making it suitable for pixel-level problems, adding class balancing, and allowing "don't care" label processing. These extensions are appropriate for improving performance of any Siamese change detection network. Third, the generalization strength of the network has been extensively evaluated on both real and artificial data and its execution speed has been optimized to allow for real-world usage. Our architecture can be useful for general mobile change detection platforms with imperfect alignment, as long as representative training data can be obtained.

Our experimental results have shown that for small-object change detection, our network outperforms the state of the art considerably on our datasets, both in terms of pixel F-1 score and object F-1 score. Furthermore, after speed optimization, the network is more than a factor two faster than both a non-CNN baseline and two other CNN approaches.

As long-term future work, an interesting and challenging possibility is to create a single end-to-end trainable network, that no longer requires the alignment phase of the system, but efficiently finds all changes based only on stereo input images and the GPS coordinates of the cameras. Overall, we conclude that the application of deep learning for roadside IED detection can significantly improve robustness of existing counter-IED change detection systems.

REFERENCES

[1] D. W. J. M. van de Wouw, G. Dubbelman, and P. H. N. D. With, "On improving IED object detection by exploiting scene geometry using stereo processing," Proc. SPIE **9407** (2015).

[2] M. Wathen, N. Link, P. Iles, J. Jinkerson, P. Mrstik, K. Kusevic, and D. Kovats, "Real-time 3D change detection of IEDs," Proc. SPIE **8379** (2012).

[3] T. Müller, "Change detection on UGV patrols with respect to a reference tour using VIS imagery," Proc. SPIE **9460** (2015).

[4] M. Tektonidis and D. Monnin, "Color consistency and local contrast enhancement for a mobile image-based change detection system," J. Imaging **3** (2017).

[5] D. W. J. M. van de Wouw, K. van Rens, H. van Lint, E. G. T. Jaspers, and P. H. N. de With, "Real-time change detection for countering improvised explosive devices," Proc. SPIE **9026** (2014).

[6] D. W. J. M. van de Wouw, F. B. ter Haar, G. Dubbelman, and P. H. N. de With, "Development analysis of a real-time demonstrator for automated detection of IED from ground vehicles," J. Electron. Imaging **28** (2019).

[7] H. Haberdar and S. K. Shah, "Disparity map refinement for video based scene change detection using a mobile stereo camera platform," *20th Int'l. Conf. on Pattern Recognition* (IEEE, Piscataway, NJ, 2010), pp. 3890–3893.

[8] G. Saur and W. Krüger, "Change detection in UAV video mosaics combining a feature based approach and extended image differencing," *Int'l. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (International Society for Photogrammetry and Remote Sensing, 2016), pp. 557–562.

[9] D. Zamalieva and A. Yilmaz, "Background subtraction for the moving camera: A geometric approach," Comput. Vis. Image Underst. **127**, 73–85 (2014).

[10] H. Sajid and S.-C. S. Cheung, "Background subtraction for static and moving camera," *Int'l. Conf. on Image Processing (ICIP)* (IEEE, Piscataway, NJ, 2015), pp. 4530–4534.

[11] K. Sakurada and T. Okatani, "Change detection from a street image pair using CNN features and superpixel segmentation," *BMVC* (BMVA Press, Durham, UK, 2015).

[12] K. Sakurada, W. Wang, N. Kawaguchi, and R. Nakamura, "Dense optical flow based change detection network robust to difference of camera viewpoints," arXiv:1712.02941, 2017.

[13] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi, "Street-view change detection with deconvolutional networks," *Int'l. Conf. on Robotics Science and Systems* (Springer, Cambridge, MA, 2016).

[14] J. W. Hsieh, C. H. Chuang, S. Alghyaline, H. F. Chiang, and C. H. Chiang, "Abnormal scene change detection from a moving camera using bags of patches and spider-web map," IEEE Sensors J. **15** 2866–2881 (2015).

[15] S. R. Klomp, D. W. J. M. van de Wouw, and P. H. N. de With, "ECDNet: Efficient siamese convolutional network for real-time small object change detection from ground vehicles," *IS&T Electronic Imaging: Intelligent Robotics and Industrial Applications using Computer Vision Proceedings* (IS&T, Springfield, VA, 2019), pp. 458-1–458-7.

[16] F. Gustafsson, *Adaptive Filtering and Change Detection* (John Wiley & Sons Ltd., Sweden, 2000).

[17] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: a systematic survey," IEEE Trans. Image Process. **14**, 294–307 (2005).

[18] A. P. Tewkesbury, A. J. Comber, N. J. Tate, A. Lamb, and P. F. Fisher, "A critical synthesis of remotely sensed optical image change detection techniques," Remote Sens. Environ. **160**, 1–14 (2015).

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2009), pp. 248–255.

[20] A. M. El Amin, Q. Liu, and Y. Wang, "Zoom out CNNs features for optical remote sensing change detection," *2nd Int'l. Conf. on Image, Vision and Computing, ICIVC 2017* (IEEE, Piscataway, NJ, 2017), pp. 812–817.

[21] B. Hou, Y. Wang, and Q. Liu, "Change detection based on deep features and low rank," IEEE Geosci. Remote Sens. Lett. **14**, 2418–2422 (2017).

[22] A. M. El Amin, Q. Liu, and Y. Wang, "Convolutional neural network features based change detection in satellite images," Proc. SPIE **10011** (2016).

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR* (ICLR, San Diego, USA, 2015).

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *NIPS'12 Proc. 25th Int'l. Conf. on Neural Information Processing Systems* (Neural Information Processing Systems, San Diego, CA, 2012), pp. 1097–1105.

[25] P. Zhang, M. Gong, L. Su, J. Liu, and Z. Li, "Change detection based on deep feature representation and mapping transformation for multi-spatial-resolution remote sensing images," ISPRS J. Photogramm. Remote Sens. **116**, 24–41 (2016).

[26] W. Zhao, Z. Wang, M. Gong, and J. Liu, "Discriminative feature learning for unsupervised change detection in heterogeneous images based on a coupled neural network," IEEE Trans. Geosci. Remote Sens. **55**, 7066–7080 (2017).

[27] G. Cao, B. Wang, H.-C. Xavier, D. Yang, and J. Southworth, "A new difference image creation method based on deep neural networks for change detection in remote-sensing images," Int. J. Remote Sens. **38**, 7161–7175 (2017).

[28] K. Lim, W. D. Jang, and C. S. Kim, "Background subtraction using encoder-decoder structured convolutional neural network," *2017 14th IEEE Int'l. Conf. on Advanced Video and Signal Based Surveillance (AVSS)* (IEEE, Piscataway, NJ, 2017).

[29] Y. Wang, P.-m. J. Fatih, P. Janusz, K. Yannick, and B. Prakash, "CDnet 2014: an expanded change detection benchmark dataset," *Proc. IEEE Workshop on Change Detection (CDW-2014) at CVPR-2014* (IEEE, Piscataway, NJ, 2014), pp. 387–394.

[30] T. Liu, Y. Li, Y. Cao, and Q. Shen, "Change detection in multitemporal synthetic aperture radar images using dual-channel convolutional neural network," J. Appl. Remote Sens. **11** (2017).

[31] M. Gong, J. Zhao, J. Liu, Q. Miao, and L. Jiao, "Change detection in synthetic aperture radar images based on deep neural networks," IEEE Trans. Neural Netw. Learning Systems **27**, 125–138 (2016).

[32] X. X. Zhu, D. Tuia, L. Mou, G. S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," IEEE Geosci. Remote Sens. Magazine **5**, 8–36 (2017).

[33] K. Nemoto, T. Imaizumi, S. Hikosaka, R. Hamaguchi, M. Sato, and A. Fujita, "Building change detection via a combination of CNNs using only RGB aerial imageries," Proc. SPIE **10431** (2017).

[34] Y. Zhan, S. Member, K. Fu, M. Yan, X. Sun, H. Wang, and X. Qiu, "Change detection based on deep siamese convolutional network for optical aerial images," IEEE Geosci. Remote Sens. Lett. **14**, 1845–1849 (2017).

[35] F. Faiz, U. Rahman, B. Vasu, J. V. Cor, and J. P. Kerekes, "Siamese network with multi-level features for patch-based change detection in satellite imagery," *6th IEEE Global Conf. on Signal and Information Processing* (IEEE, Piscataway, NJ, 2018), pp. 958–962.

[36] Q. Wang, X. Zhang, G. Chen, F. Dai, Y. Gong, and K. Zhu, "Change detection based on Faster R-CNN for high-resolution remote sensing images," Remote Sens. Lett. **9**, 923–932 (2018).

[37] R. Caye Daudt, B. Le Saux, and A. Boulch, "Fully convolutional siamese networks for change detection," *Int'l. Conf. on Image Processing, ICIP* (IEEE, Piscataway, NJ, 2018), pp. 4063–4067.

[38] D. Peng, Y. Zhang, and H. Guan, "End-to-end change detection for high resolution satellite images using improved UNet++," Remote Sens. **11**, 1382 (2019).

[39] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Neural Information Processing Systems Conf.* (MIT Press, Cambridge, MA, 2014).

[40] S. Khan, X. He, F. Porikli, M. Bennamoun, F. Sohel, and R. Togneri, "Learning deep structured network for weakly supervised change detection," *IJCAI Int'l. Joint Conf. on Artificial Intelligence* (AAAI Press, Palo Alto, CA, 2017), pp. 2008–2015.

[41] M. Yang, L. Jiao, F. Liu, B. Hou, and S. Yang, "Transferred deep learning-based change detection in remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing* (IEEE, Piscataway, NJ, 2019), pp. 1–14.

[42] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," arXiv:1605.07678v4, 2017.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2016), pp. 770–778.

[44] C. Eggert, S. Brehm, A. Winschel, D. Zecha, and R. Lienhart, "A closer look: Small object detection in faster R-CNN," *IEEE Multimedia and Expo (ICME)* (IEEE, Piscataway, NJ, 2017).

[45] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," *IEEE Visual Communications and Image Processing (VCIP)* (IEEE, Piscataway, NJ, 2017).

[46] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," IEEE Trans. Intelligent Transportation Systems **19**, 263–272 (2018).

[47] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2017), pp. 5168–5177.

[48] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," IEEE Trans. Pattern Anal. Mach. Intell. **40**, 834–848 (2018).

[49] G. L. Oliveira, W. Burgard, and T. Brox, "DPDB-Net: Exploiting dense connections for convolutional encoders," *IEEE Int'l. Conf. on Robotics and Automation (ICRA)* (IEEE, Piscataway, NJ, 2018).

[50] S. Zagoruyko and N. Komodakis, "Deep compare: A study on using convolutional neural networks to compare image patches," Computer Vision Image Understanding **164**, 38–55 (2017).

[51] R. Hadsell, S. Chopra, and Y. Lecun, "Dimensionality reduction by learning an invariant mapping," *CVPR'06* (IEEE, Piscataway, NJ, 2006), Vol. 2.

[52] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," ACM Trans. Graph. **34**, 1–98 (2015).

[53] J. Lin, O. Morere, V. Chandrasekhar, A. Veillard, and H. Goh, "DeepHash: Getting regularization, depth and fine-tuning right," arXiv:1501.04711, 2015.

[54] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR* (ICLR, San Diego, California, USA, 2015).

[55] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proc. Thirteenth Int'l. Conf. on Artificial Intelligence and Statistics* (PMLR, Sardinia, Italy, 2010), pp. 249–256.

[56] K. He, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," *ICCV* (IEEE Computer Society, Washington, DC, 2015), pp. 1026–1034.

[57] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, 'Caffe: Convolutional architecture for fast feature embedding," arXiv:1408.5093, 2014.

[58] R. Achanta and S. Sabine, "Superpixels and polygons using simple non-iterative clustering," *2017 IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2017), pp. 4895–4904.

[59] D. Debidatta, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," *2017 IEEE Int'l. Conf. on Computer Vision* (IEEE, Piscataway, NJ, 2017), pp. 1310–1319.

[60] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *European Conf. on Computer Vision* (Springer, Cham, Switzerland, 2014), pp. 740–755.

[61] M. Tanaka, R. Kamio, and M. Okutomi, "Seamless image cloning by a closed form solution of a modified poisson problem," *Proc. 5th ACM SIGGRAPH Conf. and Exhibition on Computer Graphics and Interactive Techniques* (ACM, New York, NY, 2012).

[62] P. Patrick, M. Gangnet, and A. Blake, "Poisson image editing," ACM Trans. Graph. **22**, 313–318 (2003).

[63] M. Afifi and K. F. Hussain, "MPB: A modified poisson blending technique," Computational Visual Media **1**, 331–341 (2015).