Pattern Recognition 129 (2022) 108712

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

SiameseFuse: A computationally efficient and a not-so-deep network to fuse visible and infrared images

Sedat Özer[a,*], Mert Ege[b,c], Mehmet Akif Özkanoglu[a,c]

[a] Ozer Lab., Department of Computer Science, Ozyegin University, Cekmekoy, Istanbul 34794, Turkey
[b] ASELSAN INC., Akyurt, Ankara 06750, Turkey
[c] Ozer Lab., Department of Computer Science, Bilkent University, Cankaya, Ankara 06800, Turkey

ARTICLE INFO

Article history:
Received 3 August 2021
Revised 1 April 2022
Accepted 13 April 2022
Available online 20 April 2022

MSC:
68T10
0000
1111

Keywords:
Multi-temporal fusion
Efficient learning
Multi-modal fusion

ABSTRACT

Recent developments in pattern analysis have motivated many researchers to focus on developing deep learning based solutions in various image processing applications. Fusing multi-modal images has been one such application area where the interest is combining different information coming from different modalities in a more visually meaningful and informative way. For that purpose, it is important to first extract salient features from each modality and then fuse them as efficiently and informatively as possible. Recent literature on fusing multi-modal images reports multiple deep solutions that combine both visible (RGB) and infra-red (IR) images. In this paper, we study the performance of various deep solutions available in the literature while seeking an answer to the question: "Do we really need deeper networks to fuse multi-modal images?" To have an answer for that question, we introduce a novel architecture based on Siamese networks to fuse RGB (visible) images with infrared (IR) images and report the state-of-the-art results. We present an extensive analysis on increasing the layer numbers in the architecture with the above-mentioned question in mind to see if using deeper networks (or adding additional layers) adds significant performance in our proposed solution. We report the state-of-the-art results on visually fusing given visible and IR image pairs in multiple performance metrics, while requiring the least number of trainable parameters. Our experimental results suggest that shallow networks (as in our proposed solutions in this paper) can fuse both visible and IR images as well as the deep networks that were previously proposed in the literature (we were able to reduce the total number of trainable parameters up to 96.5%, compare 2,625 trainable parameters to the 74,193 trainable parameters).

1. Introduction

With the ever-growing capabilities of deep networks as in Babaee et al. [1], Valiente et al. [2], Collobert and Weston [3], many industries around the world are now prioritizing deep learning in their solutions to keep up with the relevant state-of-the-art research. As a particular example, consider security and defence fields in which various surveillance applications require more sophisticated visual representations that will reveal the most information out of a given scene. A particular modality that is widely used in such pattern analysis applications is infrared (IR) imaging. An infrared camera operating at the thermal spectrum captures the thermal radiation emitted by the objects in the scene and an infrared image represents that information in a visual form. On the other hand, visible (VIS) image is taken in the visible domain of the spectrum (which is typically saved in RGB or in grayscale format) and it captures the information of the visible light bouncing from the materials or object surfaces in the scene. It carries more of the surface texture and shape information. Consequently, visible and IR images carry different properties of a scene. A recent trend in pattern recognition is efficiently fusing those two imaging modalities into a single and high quality visual summary (i.e., into a single "fused" image) that contains the important and descriptive information coming from both of those modalities.

Multi-modal image fusion has been studied in various fields including pattern recognition, image processing, computer vision and remote sensing. Sample works can be found in Sellami and Tabbone [4], Li et al. [5], Chen et al. [6]. For example, in Yang et al. [7] a 6-layer network was proposed where the first 3 layers were convolutional and the last 3 layers were fully connected layers to fuse hyperspectral image with the multispectral image. As another example, in Shao and Cai [8] a two-branch network was proposed to fuse two modalities (multispectral image and panchromatic image). The authors of Shao and Cai [8] proposed using different

* Corresponding author.
E-mail address: sedat.ozer@ozyegin.edu.tr (S. Özer).

https://doi.org/10.1016/j.patcog.2022.108712
0031-3203/

numbers of layers in different branches. In one branch, they proposed 2 (and 4) layers while in the other one, they proposed using 8 layers. At this point, it is important to note that the research on developing data fusion algorithms covers a broader spectrum of applications and there are significant differences between many different types of applications. For example, in fusion applications for object detection, it is important for such algorithms to focus on certain features of the objects of interests in images, as opposed to considering any and every detail in a given entire image. Such object detection algorithms typically require annotations in addition to multi-modal input image sets. Examples can be found in Chen et al. [9], Zhou et al. [10]. Typically, the goal in such applications is obtaining a more reliable bounding box information as output when the multi-modal image set is given as input. Consequently, they focus on learning the edge and texture information in those given bounding boxes. Another important application area is fusing different modalities into a single and informative image containing salient information from each modality. The difference in such applications from the object detection based applications is that salient image-fusion applications typically do not utilize annotations and therefore they consider all the pixels in given images as opposed to focusing only on the objects of interests. While multi-model techniques can be useful in many applications, the used modality and its properties can also be important, especially during the training stage of learning-based algorithms. For example, while structural similarity may be a good measure to fuse IR and visible images, it may not be a good one to combine other modalities. Therefore, in this paper, we limit our work with the line of works that combine particularly the IR images with the visible ones.

As the need for fusion algorithms increases in various markets, the trend in designing algorithms leans towards using deep (or even deeper) networks for increased accuracy, as in Li et al. [5], Shao and Cai [8]. However, another important aspect in various fields to fuse images is the computational complexity and the architecture's simplicity (referencing Occam's Razor as in Blumer et al. [11]). Computational complexity is directly related to the total number of trainable parameters in a neural network. The complexity becomes even more important when the algorithm runs on mobile and autonomous vehicles (drones, cars, etc.) due to the limited hardware and power constraints. Unfortunately, there is a trade-off between computational efficiency and accuracy and that aspect is yet to be studied in detail in the literature for networks fusing those two imaging modalities. Most relevant literature using deep learning predominantly focuses on how to get higher (better) accuracy in different metrics in terms of the output quality by ignoring the added computational burden. In many of those lines of work, however, it remains unclear how much they truly add to the computational performance by using deeper and deeper networks, i.e., it remains unclear if using additional layers would make any significant difference in accuracy, while certainly increasing the computational cost.

Our main goal in this paper, unlike the mainstream trend of using deeper and more complex networks to fuse modalities, is to look for how shallow we can go in a network to get comparable or even better accuracy while having less computational complexity when compared to the existing relevant (and possibly deeper) techniques in the literature. As also mentioned above, since multi-modal image fusion is a broad field used in many applications including object segmentation, tracking and detection, we limit ourselves with the literature that focuses on using two modalities (visible and IR) to combine salient features of each modality to obtain a new fused image in this paper. The task of fusing different modalities into a single and representative image that somehow represents both input images is a challenging task and it differs from other common vision-based tasks such as object segmentation and detection. However, the idea and the results presented in this paper may also help other vision tasks since our proposed solution, with its fused output image, can form the first block (in the form of a backbone network) in training of other networks for various other vision tasks such as segmentation, detection and tracking.

In general, when the goal is fusing multiple modalities, there are several concerns and challenges. Among those, some of the most important ones are:

1. what architecture to use to obtain a fused image as output with the state-of-the-art fusion performance;
2. what metrics to be used to check a given architecture's performance;
3. how to fuse those modalities in a more computationally efficient and in an end-to-end way;
4. how to compute (or what to use as) the ground truth for the fusion of two modalities.

Neural network based solutions have been proposed concerning the first question in literature as in Li et al. [5], Yang et al. [7], Shao and Cai [8], however it remains unclear what approach to use when considering the architecture: a deep network or a shallow network. Regarding the second question, multiple metrics such as SSIM, Entropy and many others have been used in the literature and a list of commonly used metrics in the relevant literature along with their definitions can be found in Li and Wu [12], Ma et al. [13,14], (also see Section 4.1 in this paper).

The third question relates to obtaining more computationally efficient ways to fuse the given input modalities. However, the computational side of the fusion problem has not been addressed properly before in most of the relevant papers. We address that issue in this paper from the perspective of trainable parameters. The fourth question is related to the labeling problem. In a VIS and IR image pair, in general, we obtain different features for the same object from each modality. Since the output (fused) image should summarize all the content of both images, it should not focus on particular objects in the image (such as only cars, humans or buildings) and consequently, creating a ground truth itself becomes problematic issue for this task as it is not clear whether objects should be identified in advance in each modality or the whole images should be used as the ground truth somehow. However, in a more general sense, it probably is safe to state that the similarity (in a certain form) between both input images and the output image should be high. To avoid any bounding box information, the main trend in the previous work was using the entire image from a single modality to train the network first and then duplicate that network to apply it on each modality as in Li and Wu [12]. However, we believe that both modalities should be introduced to the network during the training stage. While one current issue is the lack of large IR image equivalents of existing large data sets consisting of visible images (such as ImageNet), there are recent domain transfer based works such as Özkanoğlu and Ozer [15] that attempt to provide solutions to that problem.

In this paper, we mainly tackle above-mentioned four problems and propose a novel Siamese architecture, which uses a form of auto-encoder based CNN architecture, to fuse the given multi-modal images including both thermal (IR) and visible images. We call our proposed solution: SiameseFuse. The input to our network is a pair of IR and VIS images and the output is a single fused image. We use different numbers of layers in our architecture and report their performance in our experiments for comparison. Furthermore, as opposed to using a single loss term based on only structural similarity, we consider multiple loss terms and study the performance of using various combinations of those loss terms. Finally, we report the state-of-the-art results on various data-sets (including FLIR, Vedai and TNO Human Factors datasets) when

compared to the previous work with extensive analysis. Our main contributions include:

- introduction of a novel Siamese-based network which yields competitive results when compared to the deeper solutions from the existing literature in multiple metrics (see Section 4.1 for metric definitions and Experiments section for our results);
- introduction of a shallow end-to-end network which reduces the required trainable parameters up to **96.5%**, while providing competitive results;
- the study of using various loss functions for fusion and providing an analysis on their effect on the final result;
- use of multiple datasets to demonstrate our network's performance to the previously proposed deep models;
- reporting an extensive analysis on comparing various loss functions and their performances in our network.

The rest of this paper is organized as follows. In Section 2, related work from the relevant literature is summarized. In Section 3, our approach is explained in detail. All of our experiments and results are presented in Section 4. Finally, we conclude this paper in Section 5.

2. Related work

Over the past few years, there is constant growth in developing algorithms utilizing data that comes from different sensors, as it gets cheaper and more effective to use multiple sensors in many domains as in Sellami and Tabbone [4], Amin-Naji et al. [16]. There are many applications of fusing infrared and visible images including object tracking [17], autonomous systems [18], object detection [19], surveillance [20] and remote sensing [21]. Each of those application areas includes its own dataset(s) and different ground truth types (such as bounding boxes, pixel IDs, object IDs or time steps) with the particular type of loss functions. In this paper, our goal is to find an end-to-end fusion method that takes two images obtained by two different imaging modalities as input and produces a single fused image as output as a combination of both modalities, while being computationally efficient. Therefore, we summarize the most relevant literature mainly from that perspective next. The most relevant work can be summarized under two categories: (i) non-deep learning based techniques and (ii) deep learning based techniques.

2.1. Non-deep learning based techniques

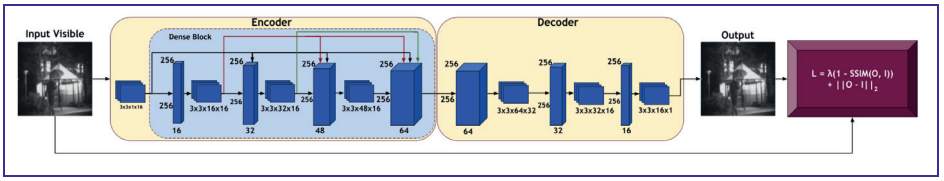
Non-deep learning (or non-neural networks) based techniques have been extensively studied in the literature. Those techniques can be classified under six main categories including: (i) multi-scale transform based, (ii) sparse representation based; (iii) subspace based; (iv) saliency based; (v) hybrid and (vi) other techniques. A detailed description on each of those categories (with many sample works from the literature) can be found in recent papers as in Ma et al. [14], Smith and Singh [17], Xing et al. [22], Ma et al. [23], Chen et al. [24].

2.2. Deep learning based techniques

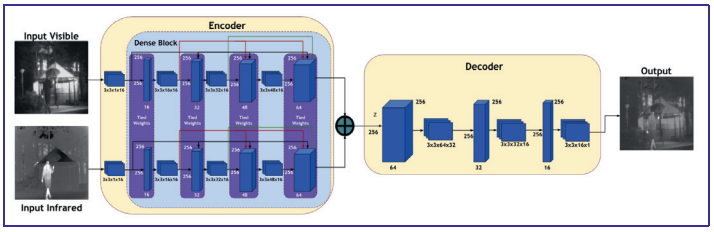
While the relevant fusion literature has studied deep architectures heavily, as in Zhang et al. [25], the inputs and the output format in each application differs. For example, the work in Zhang et al. [25] focused on fusing different features (image intensity and degree of linear polarization are used as inputs to obtain a fused image). They introduced a deep network with 12 layers to obtain the fused image. However, our main focus in this paper is fusing visible and infrared modalities to obtain a visually meaningful and representative image containing information from both input modalities. Therefore, in this section, we only provide the relevant literature with applications on that particular application.

The early neural networks based solutions include the pulse-coupled neural network (PCNN) architecture in the relevant literature as in Zhao et al. [26], Lu et al. [27]. Deep learning based techniques are more recent techniques that can provide end-to-end solutions in many applications. The success of deep learning in other fields has recently attracted the attention of the researchers working on image fusion problems as well. Since the deep learning field is a relatively new field in fusion applications, there is a limited amount of work available introducing deep learning based fusion techniques, therefore the most optimal solutions fusing IR and visible images are *probably* yet to be proposed. Convolutional Neural Network (CNN) based techniques have been recently introduced for fusion tasks. For example, as an early work, in Liu et al. [28], Liu et al. proposed a CNN-based network for fusion. In their proposed architecture, they used three convolutional layers for feature extraction and two fully connected (FC) layers where the last layer was a softmax layer. The model was trained only on visible images, however the main goal of that work was not to obtain a fused image as output. A common problem in many deep learning based methods is that as the network gets deeper, the detail information gained in each layer gradually gets lost. Li et al. proposed a CNN-based method to solve that problem of information loss in Li et al. [5]. They decomposed the image into two parts: the base part & the detail content. They used weighted-averaging strategy to obtain the fused base part. To extract the detail content, they used a CNN to extract multi-layer features to preserve as much of the information as possible. Then a soft-max operator in each layer was applied to obtain weight maps in order to get multiple candidates for the detailed content of the fused image. Finally, they obtained the final fused image by fusing those two parts. Their proposed solution uses the VGG-19 model [29] to extract deep features. Consequently, their model ends up being a very deep network (see our experiments section to compare its number of trainable parameters to others) and relies on extraction features that are trained on visible images.

Recently, a (relatively) smaller network was proposed by Li et al. [12]. They call their network as DenseFuse network to fuse infrared and visible images. DenseFuse's architecture consists of an encoder network and a decoder network. The encoder part contains one convolutional layer and three fully connected layers. The decoder part contains four convolutional layers. However that proposed solution (DenseFuse) contains slightly different architectures for training and for testing. Fig. 1 and Table 1 provides the details of those training and testing architectures of DenseFuse. During the training step, the fusion layer is eliminated and the network is trained on only the RGB images (on the MS-COCO dataset [30]). On the other hand, during the testing step, the same encoder part of the DenseFuse architecture is applied on both thermal and visible image pairs as a separate branch. The output of those two branches is then combined through a fusion layer and then fed into the decoder network. DenseFuse algorithm was originally tested on 20 image pairs (additionally the trained model was also tested on the dataset from [31]). In their fusion layer, two strategies were introduced: (i) adding the pixels directly and (ii) using the ℓ_1-norm strategy. In total, the DenseFuse architecture contains 74,193 trainable parameters and includes 5 convolutional layers and 3 fully connected layers with skip connections (see Fig. 1 for the architecture and Table 1 for the details of each used layer in the DenseFuse architecture). In its loss function, DenseFuse uses the sum of weighted pixel loss and structural similarity based loss terms [12]. Another recent work introduced a Generative Adversarial Network (GAN) architecture (called DDcGAN) in [13] to consider fusing IR and visible images where each image has a different



(a) DenseFuse Training Model: The model uses only visible (VIS) images for training (used MS-COCO dataset) and uses a single branch network to train. No IR information is incorporated (or introduced) into the network during the training step.



(b) DenseFuse Testing Model: After the network trained on VIS images only, during the inference step, the encoder part is duplicated for each modality yielding two separate branches processing each modality separately. Those two branches are then combined through a fusion layer (tensor addition) and fed into the decoder network to obtain the final output.

Fig. 1. The architecture of the previous work: the DenseFuse network, Li and Wu [12]. The different architectures of training and testing steps are shown individually.

Table 1
Details of each layer and the total number of trainable parameters for the DenseFuse algorithm [12] (see Fig. 1).

Process	Layer	Size	Stride	Channel (Input)	Channel (Output)	Activation	Parameters
Input	–	256 × 256	–	–	1	–	
Encoder	Conv Layer 1	3 × 3	1	1	16	ReLU	160
Encoder	Dense Layer 1	3 × 3	1	16	16	ReLU	2320
Encoder	Dense Layer 2	3 × 3	1	32	16	ReLU	4624
Encoder	Dense Layer 3	3 × 3	1	48	16	ReLU	6928
Decoder	Conv Layer 2	3 × 3	1	64	64	ReLU	36,928
Decoder	Conv Layer 3	3 × 3	1	64	32	ReLU	18,464
Decoder	Conv Layer 4	3 × 3	1	32	16	ReLU	4624
Decoder	Conv Layer 5	3 × 3	1	16	1	ReLU	145
Output	–	256 × 256	–	1	–	–	
Param. #:	–	–	–	-	–	–	74,193

resolution. Their architecture uses one generator network and two discriminator networks. In their generator network, they use 10 convolutional layers and in both of their discriminator networks, they use 3 convolutional layers followed by a fully connected layer. They tested their network on 36 pairs of IR and visible images from the TNO Human Factors dataset[1] and created smaller patches from those 36 image pairs. The mode collapse problem of GAN-based architectures [32] remains unexplored in that work. A more recent and relevant work was introduced in Fang et al. [33] which also depends on a deep network architecture using VGG19, and squeezeNet networks as well as channel attention and channel shuffle modules (making the network a very large and a very deep network).

As it can be seen in the recent works [5,12,34] and in Fang et al. [33], the trend in fusing IR and visible images is going towards utilizing more and more layers or using more complicated and pretrained architectures and possibly with skip connections. While such architectures can yield slightly higher performance, it remains unclear whether the added computational complexity (by using deeper and deeper networks) can justify the increase in performance. Different deep architectures may introduce different complexities to their approach. For example, mode collapse [32] is a well-known problem in GAN-based architectures and that may yield convergence to focus on particular shapes or textures in fusion. Similarly, in general, including fully connected layers with skip connections increases the total number of trainable parameters and introduces additional implementation based complexities. Consequently, at this point, the question: *"Do we need deeper networks to fuse IR and visible images?"* should also be asked. Our main goal in this paper, is looking for an answer to that question. We argue that we can get closer to those deeper networks' performance (or can get even better performance) by using not-so-deep networks and by utilizing only convolutional layers.

Our work differs from the previous work in multiple ways: (i) we focus on developing a network that is truly trained on two image modalities, while many earlier papers focused on training from a single modality (or using pretrained models that were trained only on visible images); (ii) we focus on developing a computationally efficient and fully convolutional network to fuse VIS and IR images while the most relevant works utilize networks containing both convolutional any fully connected layers as in Li et al. [5], Li and Wu [12]; (iii) we include multiple loss functions and evaluate their performances on extensive experiments; and (iv) the previous works were mainly tested on small numbers of image pairs (for example DenseFuse uses only 20 image pairs for testing). However, we utilize significantly larger datasets to perform our experiments (For example, in FLIR dataset, we use 9609 image pairs for training and 839 image pairs for testing).

[1] https://figshare.com/articles/TNO_Image_Fusion_Dataset/1008029.

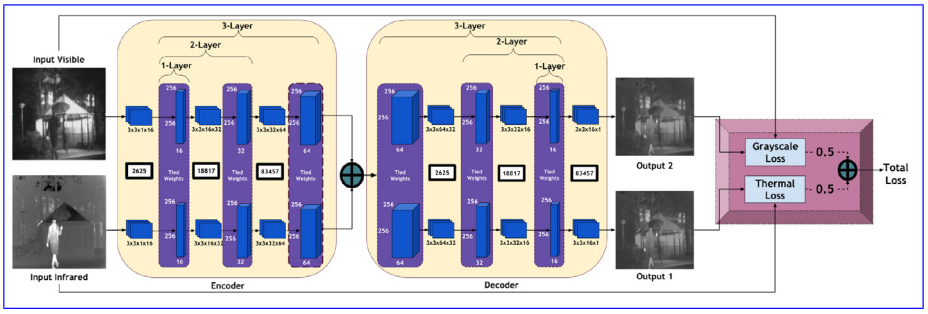


Fig. 2. An overview of our proposed SiameseFuse architecture including the loss unit. The details of each layer is given in Table 3. This figure shows all three of our proposed models together including: 1-layer, 2-layer and 3-layer SiameseFuse architectures.

Table 2
List of total number of trainable parameters is given for three different configurations of our proposed SiameseFuse.

Proposed models	Total number of trainable parameters
1-Layer Architecture	2625
2-Layer Architecture	18,817
3-Layer Architecture	83,457

3. Proposed method

In this paper, we propose a fully-convolutional Siamese network based solution whose input is an image pair (VIS and IR) obtained via two different imaging modalities and the output is a fused image. Our proposed architecture is given in Fig. 2, which contains only convolutional layers. Unlike the previous work (such as DenseFuse), we do not use any fully connected layer or skip connections in our proposed architecture to keep our model simple. We call our proposed architecture SiameseFuse.

Our proposed SiameseFuse architecture consists of three main blocks: (1) the *Extraction block* which extracts the salient features from each modality and it acts as an encoder network; (2) the *fusion block* in which the extracted salient features are element-wise added; and (3) the *Reduction block* which acts similar to a decoder network to map high dimensional salient features back to two dimensional and single channel image space. Our SiameseFuse architecture extracts the salient features from each modality through a series of convolutional layers (where the number of convolutional layers varies between 1 and 3 in this work) in the extraction block. After the extraction block, the image feature maps are fused in a linear fashion which is indicated as *Tensor Addition* in Fig. 2. In the final (third) section, we use another Siamese architecture: *Reduction* block to revert the initial encoding process. After the reduction block, both thermal and grayscale loss functions are calculated separately during the training of the network by using Eqs. (1) and (2), respectively. Finally, those two loss functions are combined by taking their weighted average with Eq. (3) (see next subsection for more information about the loss function).

In this paper, we define three different models (variants) for our SiameseFuse architecture, and between each of those models, only the total number of used-layers changes. Fig. 2 shows those three separate architectures (varying in layer numbers), namely: (1) 1-layer architecture, (2) 2-layer architecture, and (3) 3-layer architecture all-together. The 3-layer architecture uses all the shown layers in both extraction and reduction blocks in the figure. The 2-layer architecture uses the first two layers of the extraction block and the last two layers of the reduction block. Finally, the 1-layer architecture uses only the first layer of the extraction block and only the last layer of the reduction layer in the figure. Table 3 also includes further details for each layer shown in the figure. Furthermore, Table 2 also compares the total number of trainable parameters for each one of our three proposed models.

Our loss unit is shown at the end of the Fig. 2. One problem in supervised learning is what kind of labels (ground truth) to use in the loss function and that is also a problem in our application area. Some applications use bounding box information while others may use segmentation masks to focus on certain shapes and textures. However, in our application we do not have any annotation information. In this paper, our solution to the lack of annotations (or lack of available ground truth) problem is briefly explained as follows: we do not look for particular objects in images and we do not look for any bounding box in the images as our goal is combining the salient information from both images. Consequently, instead of looking for labelled data, we use each input image as the output for the other image (i.e., use each input modality as output for the other modality) within our Siamese architecture. That way, we can train our Siamese architecture to learn both modalities at the same time and to learn how to combine local structure & texture information from both input modalities by correlating one image modality to the other one. I.e., during the training step, for a given VIS and IR image pair, we use the VIS image as the output for the upper network (which uses the IR image as input) and we use the IR image as the output for the lower network (which uses the VIS image as input) in Fig. 2. Therefore, each channel learns to transform one modality to another one while they both share the same weights. Since they learn how to combine both images in the Reduction block (and since both channels yield the same output) during the testing stage, we consider only one output of those two channels of the reduction block.

$$\mathcal{L}(I_1, O) = \lambda_1(1 - \text{SSIM}(I_1, O)) + \lambda_2\|(I_1 - O)\| + \lambda_3(1 - \rho(I_1, O)) + \lambda_4(1 - \text{MS\_SSIM}(I_1, O)) \quad (1)$$

$$\mathcal{L}(I_2, O) = \lambda_1(1 - \text{SSIM}(I_2, O)) + \lambda_2\|(I_2 - O)\| + \lambda_3(1 - \rho(I_2, O)) + \lambda_4(1 - \text{MS\_SSIM}(I_2, O)) \quad (2)$$

$$\mathcal{L} = \alpha_1\mathcal{L}(I_1, O) + \alpha_2\mathcal{L}(I_2, O) \quad (3)$$

3.1. Loss function

In our loss block (see the "loss unit" in Fig. 2), there are two separate loss computations to be made. The first loss function uses visible and fused images as input, while the second loss function uses the thermal and fused images as input. Eq. (3) shows those two loss functions, where I_1 is the grayscale (VIS) image, I_2 is the thermal (IR) image, and O is the fused output image. In Eqs. (1) and (2), the parameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ represent the weights for each used loss term. In Eq. (3), α_1 and α_2 represent

Table 3
Details of each layer in our proposed SiameseFuse architecture is given for the extraction and for the reduction layers individually.

Process	Layer	(Filter) Size	Stride	Channel (Input)	Channel (Output)	Activation
Input	–	256×256	–	–	1	–
Extraction	Conv Layer 1	3×3	1	1	16	ReLU
Extraction	Conv Layer 2	3×3	1	16	32	ReLU
Extraction	Conv Layer 3	3×3	1	32	64	ReLU
Reduction	Conv Layer 3	3×3	1	64	32	ReLU
Reduction	Conv Layer 2	3×3	1	32	16	ReLU
Reduction	Conv Layer 1	3×3	1	16	1	ReLU
Output	–	256×256	–	1	–	–

the linear weights for the thermal and the grayscale losses. The final loss function is a weighted linear combination of both thermal and visible loss functions as defined in Eq. (3).

In our total loss function (Eq. (3)), instead of using only the structural similarity (SSIM) function as the loss function, we used a combination of multiple meaningful terms including l_2 norm (also known as the pixel loss), SSIM, multiscale structural similarity (MS-SSIM) and cross-correlation to study the effect of their performance on the fusion task. In our equations, the function: $\|.\|$ represents the l_2 norm and the function: $\rho(.)$ represents the cross-correlation. The definitions of the other two used functions: SSIM and MS_SSIM are given below.

3.1.1. SSIM

The structural similarity is a measurement of the perceived quality of images, based on the extraction of structural information from an input. This metric starts by defining x & y as two discrete non-negative signals (where x is the input image and y is the fused image), calculates their mean μ and variance σ, then obtains luminance l, contrast c, and structure comparison s as follows:

$$l(\mathrm{x,y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4}$$

$$c(\mathrm{x,y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{5}$$

$$s(\mathrm{x,y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{6}$$

where C_1, C_2, C_3 are the softening constants, given by

$$C_1 = (K_1L)^2,\ C_2 = (K_2L)^2 \ \text{ and } C_3 = C_2/2 \tag{7}$$

where K_1 & K_2 are constants $\ll 1$ and L is the dynamic range of the pixel values. Structural Similarity (SSIM) [35] between the input image and the fused image is then defined as follows:

$$\mathrm{SSIM(x,y)} = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma} \tag{8}$$

The parameters: α, β, and γ are used to adjust the importance of each component. Typically, those parameters are all set to 1, as in Ma et al. [35]. This way, the equation can be reorganized as,

$$\mathrm{SSIM(x,y)} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)(\sigma_{xy} + C_3)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)(\sigma_x\sigma_y + C_3)} \tag{9}$$

Consequently, the final SSIM value between the given two input images x_1, x_2 and the fused image y is defined as follows:

$$\mathrm{SSIM} = SSIM(x_1, y) + SSIM(x_2, y) \tag{10}$$

.

3.1.2. MS_SSIM

The multi-scale SSIM [35] method involves the input image being down-scaled by a factor of 2 while also having applied a low-pass filter. The original image is indexed as *1* and the max down-sample iteration is indexed as *M*.

$$\mathrm{MS\_SSIM(x,y)} = [l_M(x,y)]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(x,y)]^{\beta_j} [s(x,y)]^{\gamma_j} \tag{11}$$

4. Experiments

In our experiments, first bicubic interpolation is used to reduce the image dimensions to 256×256 and RGB images are converted into grayscale images as we use one channel inputs for each branch in our Siamese architecture. We assume that both VIS and IR images given in the input image pair are already registered. For that reason, we manually (externally) computed the parameters between the image pairs first and then registered all the image pairs by using those parameters before training the network.[2] Both of the α_1 and α_2 values of Eq. (3) are set to 0.5 to consider each modality equally in all of our experiments (except the experiments summarized in Table 8).

We use multiple datasets to compare our results to the existing work in the relevant literature and we evaluate the performance by using seven different metrics. Next, we define those metrics.

4.1. Metrics

In this paper, one of the algorithms that we compare our results to is DenseFuse algorithm [12] and since seven different metrics were used in that work, for comparison reasons, we use those same seven metrics in this work as well. The metrics used in this paper are briefly explained below.

4.1.1. SCD

The sum of the correlations of differences (SCD) metric is defined (in Aslantas and Bendes [36]) as follows:

$$\mathrm{SCD} = \mathrm{Corr}(F - A, B) + \mathrm{Corr}(F - B, A) \tag{12}$$

where A and B are the input images and F is the output.

4.1.2. $Q^{AB/F}$

$Q^{AB/F}$ [37] is a metric that computes the weighted average of $Q^{A/F}$ and $Q^{B/F}$. It is a measure of the perceptual loss of information in terms of orientation and strength. It involves computing the relation of edge information, obtained by a sobel edge detection algorithm, of the two source images: A and B, and the fused output image: F.

[2] We also tested some deep solutions for image registration as a preprocessing step, however, since we did not find their registration performance satisfactory, we decided to use manual solution which was more accurate.

4.1.3. FMI

Feature Mutual Information (FMI) is a metric proposed to measure how consistent the measured input with subjective measurements or human observations. It extends the concept of mutual information by extracting features of the image before the calculation [38]. Therefore, the computation of this metric is more complex and memory intensive. In our experiments, we use the Fast-FMI implementation from [39]. In our experiments as the features, we used two feature types: discrete cosine transform (FMI_{dct}) and wavelet transform (FMI_w) individually for this metric.

4.1.4. Entropy

Entropy is defined to be:

$$\mathrm{En} = -\sum_{i=0}^{n-1} p_i \log_2 p_i \tag{13}$$

where parameters n and p denote the parameters that arise from the histogram created from the image. n is the number of gray levels, while p_i is the probability of a pixel having the i'th gray level. In our experiments, entropy is calculated using only the output image, which should have high entropy due to increased detail resulting from the merging of two sources.

For the remaining two metrics: SSIM and *MS_SSIM*, please refer to Sections 3.1.1 and 3.1.2 for the detailed definition, respectively.

4.2. Datasets

We have included three different datasets in this paper; namely (i) FLIR dataset[3] (contains VIS and IR image pairs), (ii) TNO-Human Factors (TNO-HF) datasets[4], and (iii) VEDAI dataset [40]. Our used baseline networks are trained on their individual datasets. For example, pretrained DenseFuse was trained on a single modality (on 79,000 MS-COCO [30] visible images), The pretrained version of the model from [5] is trained on ImageNet dataset [41] using 1.2 Million grayscale images (single modality), the pretrained DDcGAN [13] was trained on the TNO-HF dataset and the pretrained U2Fusion model was trained on the FLIR dataset. Our networks are trained on the FLIR dataset for both training and testing (the image pairs in the FLIR dataset do not come as registered. To avoid confusion and errors caused by the misalignment of the FLIR image pairs, we included a registration process on those image pairs as a pre-processing step in our experiments). In the training step, we used 9609 image pairs and we used an additional 839 image pairs for the testing.

4.3. Results

In terms of performance evaluation, we have conducted various tests with different configurations of our architecture and studied the different aspects of our proposed network. We compared our algorithm's performance to DenseFuse, DDcGAN, U2Fusion, ResNetFusion and to the work from [5] which we call *DeepLearningFramework* in our tables. The ResNetFusion algorithm does not come with a pretrained model and therefore, we trained it on the same FLIR data as we trained our SiameseFuse algorithm.

4.3.1. Training process

During the training of SiameseFuse, we used Adam optimizer [42] and the learning rate was set to 0.001. We trained SiameseFuse with random initialization and noticed that our network converges rapidly after a few iterations. Figs. 3–5 demonstrate the fast convergence during the training process on FLIR dataset. In each figure, the x-axis shows the epoch number.

[3] https://www.flir.in/oem/adas/adas-dataset-form.

[4] https://figshare.com/articles/TNO_Image_Fusion_Dataset/1008029.

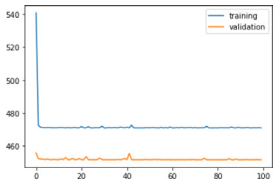


Fig. 3. Loss Function vs. Epochs.

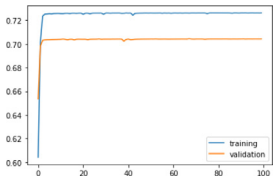


Fig. 4. $SSIM_a$ vs. Epochs.

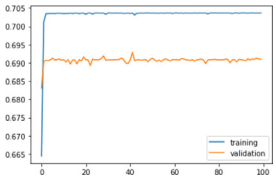


Fig. 5. MS_{SSIM} vs. Epochs.

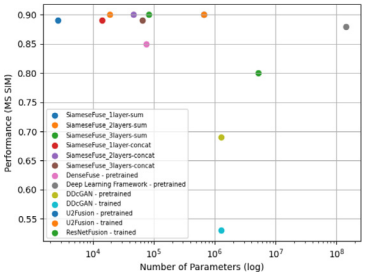


Fig. 6. MS_SSIM vs. Total number of Trainable Parameters.

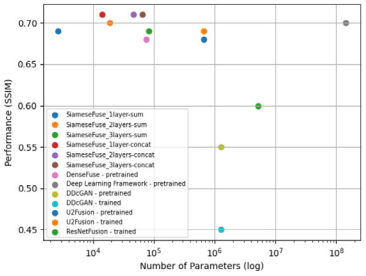


Fig. 7. SSIM vs. Total number of Trainable Parameters.

4.3.2. Comparison of the total number of trainable parameters

Figs. 7 and 6 compare the total number of trainable parameters required by each of our proposed models (with different numbers of layers) and compare those models to multiple recent works. In the figures, the total number of trainable parameters for each network is shown in *logarithmic scale*. We compared the performance of our proposed Siamese Network under different configurations (we used different filter numbers for our 2-layer model and each of those filter numbers for each layer is given in the parenthesis). As the figure shows, there is only a slight difference in performance between 1-layer and 2-layer SiameseFuse networks. Note that in the figures, the x-axis is shown in logarithmic scale. The model from [5] is the largest model having the highest number of trainable parameters as it includes a (pre)trained VGG-19

Table 4
This table shows how changing the filter dimensions in 2-Layer SiameseFuse network affects its performance.

Filter dimensions	En	$Q^{AB/F}$	SCD	FMI_w	FMI_{dct}	$SSIM_a$	MS_{SSIM}
3×3 - 3×3	**7.304**	0.439	**1.436**	**0.428**	**0.395**	0.704	0.894
5×5 - 3×3	7.298	0.437	1.428	0.424	0.392	**0.705**	**0.895**
5×5 - 5×5	7.298	**0.442**	1.426	0.423	0.393	0.702	**0.895**
3×3 - 5×5	7.290	0.433	1.420	0.419	0.389	0.703	0.892
DenseFuse – pretrained	7.124	0.417	1.207	0.422	0.392	0.681	0.846

Table 5
This table shows how changing the total number of filters in 2-Layer SiameseFuse affects its performance.

Filter numbers	En	$Q^{AB/F}$	SCD	FMI_w	FMI_{dct}	$SSIM_a$	MS_{SSIM}
8 - 16	7.298	0.438	1.430	0.423	0.392	0.703	0.894
16 - 32	**7.304**	0.439	**1.436**	**0.428**	**0.395**	0.704	0.894
32 - 64	7.303	**0.440**	1.435	0.427	**0.395**	**0.705**	**0.895**
DenseFuse – pretrained	7.124	0.417	1.207	0.422	0.392	0.681	0.846

model [29] on ImageNet dataset as a backbone in its architecture. As Fig. 6 shows, our models yield the least trainable numbers with the highest SSIM performance.

4.3.3. The effect of changing the filter dimensions

We studied how changing the filter dimensions in our architecture affects the performance. For this, we first fixed the total number of layers in our network and used the 2-layer SiameseFuse architecture. As for the filter dimensions, we used different combinations of 3×3 and 5×5 filters in each layer of our 2-layer SiameseFuse architecture. The obtained results are computed for seven metrics and summarized in Table 4. In the table, the first filter dimension represents the filter dimension that is used in the first layer in the "extraction" block and its equivalent layer in the "reduction" block, and the second number represents the filter dimension that is used in the second layer of the "extraction" block and its equivalent layer in the "reduction" block in our 2-Layer architecture. There is only a slight difference between the maximum obtained values in both the first and second rows, consequently, we kept using the 3×3 filter size. As the table shows, all the combinations yielded better SSIM and MS_SSIM results than the previous work: DenseFuse.

4.3.4. The effect of changing total number of used filters

Next, we studied how the total number of filters used in each layer affects the performance while the filter dimensions are fixed. For that purpose, we first kept the filter dimensions as 3×3 for each filter (in each layer). Then we changed the total number of used filters in our 2-layer architecture. We used the following combinations of filter numbers: (8–16), (16–32) and (32–64) where the first number represents the filter number used in the first layer and the second number represents the filter number used in the second layer. Table 5 compares the results obtained from our 2-layer architecture for 6 different metrics. While we obtain slight increase in performance by changing the filter numbers, we think that increase is not significantly important to use more filters. We also compared SiameseFuse results to the DenseFuse result in the table. All the tested combinations yielded higher (better) SSIM and MS_SSIM results than the previous work: DenseFuse.

4.3.5. Comparison of two fusion layer strategies

Fusing the information coming from each branch in the Siamese architecture can be done by using two main strategies: (i) through an element-wise addition process, and (ii) through a concatenation process. Table 6 shows how varying the total number of layers used in both "extraction" and "reduction" blocks affects the performance. In our experiments with this part, we also studied the performance of using element-wise summation and concatenation operations in the middle fusion block of our DenseFuse architecture. As the table shows, since there is no significant difference is obtained between those two operations, in our final architecture, we used the summation operation. The table shows the results of using 1 layer only, 2 layers and 3 layers in each "extraction" and "reduction" blocks in seven different metrics. While all the combinations showed significant improvement from the previous work: DenseFuse, we did not observe significant difference between using higher numbers of layers. The last column in the table shows the total number of trainable parameters in each model. Since the 1-layer architecture has significantly less number of parameters than all other architectures while yielding similar or better results than our other models, we decided to propose that one for applications where efficient computation is a constraint.

4.3.6. The effect of λ values of Eq. (3) in results

In order to test the contribution of each term used in Eq. (3) in SiameseFuse, we use different values for different weights: λ. The results are presented in Table 7. In this experiment, we kept the layer numbers fixed (2 Layer architecture is used) and the filter dimensions kept at (3,3) and (3,3). As expected, at different weight combinations, different results were obtained. For comparison (as a baseline) we also included the results obtained by the pretrained DenseFuse algorithm.

4.3.7. Qualitative results

Figs. A1–A4 show qualitative results obtained from our network to the previous work: DenseFuse. The image pairs are from the FLIR dataset. Column (d) shows the results obtained with addition strategy in the fusion layer and column (e) is obtained by using the concatenation strategy in the fusion layer. All the SiameseFuse results are obtained from 2-layer model. In all SiameseFuse models, we set $\lambda_1 = 0$, $\lambda_2 = 1$, $\lambda_3 = 1$ and $\lambda_4 = 1$.

4.3.8. The effect of changing α_1 and α_2 values in Eq. (3)

In order to test the contribution of each modality in our results (in terms of each metric), we study the effect of changing α_1 and α_2 values in Eq. (3). The results are presented in Table 8. In this experiment, we kept the layer numbers fixed (2 Layer architecture is used) and the filter dimensions kept at (3,3) and (3,3) and used FLIR dataset. As expected, at different weight combinations, different results were obtained. For comparison (as a baseline) we also included the results obtained by the pretrained DenseFuse algorithm.

Table 6
This table shows the performance of using different layers in our proposed network: SiameseFuse. At the fusing step, we tested the performance of both element-wise addition and concatenation of the feature-maps obtained from each modality. *DeeplearningFramework* was pretrained on ImageNet dataset. *DenseFuse* was pretrained on MS-COCO. DDCGAN was pretrained on TNO-HF and U2Fusion was pretrained on FLIR. All the newly trained versions of DDcGAN, U2Fusion and ResNetFusion used FLIR dataset. All of the listed architectures were *tested* on the same 839 image pairs. The last column lists the total number of trainable parameters needed during the training for each algorithm.

FLIR-train	En	$Q^{AB/F}$	SCD	FMI_w	FMI_{dct}	$SSIM_a$	MS_{SSIM}	#ofParams
$SiameseFuse_{1layer-sum}$	7.31	0.44	1.42	0.43	**0.40**	0.69	0.89	**2627**
$SiameseFuse_{2layers-sum}$	7.30	0.44	**1.44**	0.43	**0.40**	0.70	**0.90**	18,817
$SiameseFuse_{3layers-sum}$	7.32	0.44	1.43	0.43	**0.40**	0.69	**0.90**	83,457
$SiameseFuse_{1layer-concat}$	7.25	0.43	1.43	0.42	0.39	**0.71**	0.89	14,177
$SiameseFuse_{2layers-concat}$	7.27	0.43	**1.44**	0.42	0.39	**0.71**	**0.90**	46,529
$SiameseFuse_{3layers-concat}$	7.23	0.42	1.41	0.42	0.39	**0.71**	0.89	64,961
DenseFuse – pretrained [12]	7.12	0.42	1.21	0.42	0.39	0.68	0.85	74,193
DeepLearningFramework – pretrained [5]	7.03	0.45	1.29	0.43	0.39	0.70	0.88	143,667,240
DDcGAN – pretrained [13]	7,52	0,35	1,35	0,38	**0,40**	0,55	0,69	1,275,716
DDcGAN – trained [13]	7,69	0,28	1,09	0,35	0,34	0,45	0,53	1,275,716
U2Fusion – pretrained [43]	7,08	**0,49**	1,26	0,36	0,34	0,68	**0,90**	659,217
U2Fusion – trained [43]	7,21	0,48	**1,44**	0,42	0,38	0,69	**0,90**	659,217
ResNetFusion – trained [34]	**7,81**	**0,49**	0,97	**0,44**	0,33	0,60	0,80	5,108,105

Table 7
This table shows the effect of using different weights in our used loss function in seven different metrics. In the early rows, a different hyperparameter were set to 0 to see the significance of using that loss term in our results with each metric. In this paper, we use SSIM as our main performance metric. DenseFuse results are also included for comparison.

HyperParameters	En	$Q^{AB/F}$	SCD	FMI_w	FMI_{dct}	$SSIM_a$	MS_{SSIM}
$\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 1$	7.30	0.44	1.43	0.42	**0.40**	0.70	0.89
$\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 1$	7.30	0.44	**1.44**	**0.43**	**0.40**	0.70	0.90
$\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 3$	**7.31**	0.44	**1.44**	**0.43**	0.39	**0.71**	0.89
$\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 5$	7.30	0.44	1.43	**0.43**	**0.40**	**0.71**	0.89
$\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 10$	**7.31**	0.44	**1.44**	**0.43**	**0.40**	0.70	0.89
$\lambda_1 = 1, \lambda_2 = 0, \lambda_3 = 1, \lambda_4 = 1$	7.27	0.43	1.40	0.42	**0.40**	**0.71**	0.89
$\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 0, \lambda_4 = 1$	7.29	0.44	**1.44**	0.42	0.39	0.69	0.90
$\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 0$	7.27	0.44	1.43	0.42	0.39	**0.71**	0.90
$\lambda_1 = 1000, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 1000$	7.21	**0.45**	1.39	0.42	0.39	0.70	**0.91**
$\lambda_1 = 1000, \lambda_2 = 1, \lambda_3 = 0.1, \lambda_4 = 1000$	7.18	0.43	1.34	0.41	0.38	**0.71**	0.90
$\lambda_1 = 1000, \lambda_2 = 0.1, \lambda_3 = 1, \lambda_4 = 1000$	7.18	0.40	1.36	0.41	0.38	**0.71**	0.89
$\lambda_1 = 10, \lambda_2 = 10, \lambda_3 = 0.1, \lambda_4 = 10$	7.27	0.44	**1.44**	0.42	0.39	**0.71**	0.90
$\lambda_1 = 10, \lambda_2 = 1, \lambda_3 = 0.1, \lambda_4 = 10$	7.26	0.44	1.42	**0.43**	0.39	**0.71**	0.90
$\lambda_1 = 100, \lambda_2 = 1, \lambda_3 = 0.1, \lambda_4 = 100$	7.25	0.44	1.39	0.42	0.39	**0.71**	0.89
$\lambda_1 = 100, \lambda_2 = 10, \lambda_3 = 0.1, \lambda_4 = 100$	7.27	0.44	1.43	0.42	0.39	**0.71**	0.90
$\lambda_1 = 100, \lambda_2 = 100, \lambda_3 = 0.1, \lambda_4 = 100$	7.27	0.44	**1.44**	0.42	0.39	**0.71**	0.90
DenseFuse – pretrained	7.12	0.42	1.21	0.42	0.39	0.68	0.85

Table 8
This table shows the effect of using different α_1 and α_2 values of Eq. (3) in SiameseFuse (using FLIR data). DenseFuse results are also included for comparison.

α_1 (Vis) - α_2 (IR) values	En	$Q^{AB/F}$	SCD	FMI_w	FMI_{dct}	$SSIM_a$	MS_{SSIM}
Vis:0.3 - IR:0.7	**7.472**	**0.443**	1.324	0.393	0.369	0.660	0.835
Vis:0.4 - IR:0.6	7.355	0.441	**1.354**	0.408	0.380	0.678	0.863
Vis:0.5 - IR:0.5	7.013	0.393	1.281	**0.431**	**0.397**	**0.702**	**0.870**
Vis:0.6 - IR:0.4	6.886	0.306	1.140	0.378	0.361	0.670	0.767
Vis:0.7 - IR:0.3	6.852	0.282	0.911	0.329	0.327	0.637	0.699
DenseFuse – pretrained	7.124	0.417	1.207	0.422	0.392	0.681	0.846

Table 9
Comparing the results of SiameseFuse to various algorithms on two additional datasets: VEDAI and TNO Human Factors datasets.

VEDAI dataset	En	$Q^{AB/F}$	SCD	FMI_w	FMI_{dct}	$SSIM_a$	MS_{SSIM}
SiameseFuse	6,53	0,71	0,86	0,73	0,77	**0,89**	0,94
DenseFuse - pretrained	6,68	**0,72**	**1,25**	**0,74**	**0,78**	**0,89**	**0,96**
DDcGAN - pretrained	**7,54**	0,47	0,92	0,59	0,58	0,66	0,76
U2Fusion - pretrained	6,66	0,64	0,82	0,58	0,64	0,82	0,93
DeepLearningFramework – pretrained [5]	6.54	**0,72**	0.89	0.73	**0,78**	**0.89**	0.95
TNO-HF dataset	En	$Q^{AB/F}$	SCD	FMI_w	FMI_{dct}	$SSIM_a$	MS_{SSIM}
SiameseFuse	6,17	0,36	1,63	**0,42**	**0,40**	**0,72**	0,84
DenseFuse - pretrained	6,73	0,43	**1,84**	0,37	0,34	0,66	**0,91**
DDcGAN - pretrained	**7,41**	0,37	1,58	0,41	0,39	0,54	0,73
U2Fusion - pretrained	6,27	**0,44**	1,64	0,35	0,34	0,70	0,87
DeepLearningFramework – pretrained [5]	6.20	0.42	1.65	**0.42**	**0,40**	**0,72**	0.85

4.3.9. Experimental results on additional datasets

In order to check our results further, we experimented on two additional datasets including VEDAI [40] and TNO Human Factors (TNO-HF) datasets. In these experiments, we compared our SiameseFuse to multiple recent algorithms and reported the results in seven different metrics. In the experiment using VEDAI dataset, except the metric: *En*, in all other metrics, pretrained DenseFuse yielded higher results. We think that the main reason for that is the thermal images of the Vedai dataset [40] being short-wave infrared. However, we trained our model with long-wave infrared dataset (using FLIR dataset) and the pretrained model of DenseFuse is trained with MS-COCO. Consequently, the used data in training DenseFuse is more similar to the VEDAI dataset. See Table 9 for the results.

VEDAI dataset includes 302 short-wave infrared and visible image pairs. The resolution of these images is 256×256. All of the images are used for testing that SiameseFuse is trained with FLIR dataset (with 9609 training image pairs) and DenseFuse is trained with MS-COCO as previously stated.

TNO-HF dataset contains 20 thermal and visible image pairs. Therefore it is a small dataset. As the results demonstrate, our (shallow) model gives similar results to the results of the deeper models, while SiameseFuse uses only two convolutional layers in each encoder and decoder part of its architecture. In some metrics, SiameseFuse still provides the best results, while in others, it still provides a close performance to the other algorithms' results (see Table). In particular, SiameseFuse yielded the best results in FMI_w, FMI_{dct} and $SSIM_a$ metrics, (see Table 9).

5. Conclusion

In this paper, we propose a novel and end-to-end architecture (with its three variant models) based on Siamese networks to fuse infrared (IR) and visible (VIS) image pairs into a single output image and compare its performance to two most relevant deep architectures (DenseFuse [12] and DeepLearningFramework [5]) from the literature. Our proposed solution is a fully convolutional network (that does not use any dense or fully connected layers) and it yields comparable or better results by requiring significantly fewer trainable parameters. Fewer trainable parameters means less computational requirements and less power consumption in mobile platforms. In our experiments, we showed that using multiple structural terms in the loss function can be beneficial. Furthermore, as we also demonstrated in our experiments, our proposed solution can yield a superior performance when compared to the multiple algorithms from the relevant recent literature in different metrics.

As the mainstream trend is using and designing deeper and deeper architectures for many pattern recognition applications, a main and fundamental aspect of pattern analysis should also be kept considering: *simplicity in model design*. In this paper, we show that we can use as little as two layers to fuse visible and IR images in our architecture. To achieve success with such a shallow network, we carefully design the network and choose the appropriate loss functions. Such simplicity in modeling introduces mobility to be used in other pattern recognition applications. For example, in many fusion applications a not-so-shallow network can be used as a backbone to fuse the images coming from different sensors to obtain a single output image, which can, then, be used as input for other applications such as object detection or tracking. Furthermore, in many pattern analysis applications and in many mobile robot systems, it is important to use computationally efficient networks that are as compact as possible due to the limited memory and limited computing power.

In this work, we show that SiameseFuse can provide better or similar performance when compared to the related work using deeper networks such as DenseFuse, while significantly reducing the network size by using **% 96,5 less trainable parameters**, when our 1-layer SiameseFuse network is used (compare the 2625 trainable parameters of our 1-layer SiameseFuse vs. the 74,193 trainable parameters of DenseFuse). Therefore, our 1-layer SiameseFuse solution remains a good choice that can be used not only on strong computers, but also on various mobile platforms with its significantly smaller number of trainable parameters, while yielding comparable fusing results to that of existing very deep architectures. Future work may include using SiameseFuse in different computer vision applications as a backbone fusing both modalities (VIS and IR).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper has been produced benefiting from the 2232 International Fellowship for Outstanding Researchers Program of TÜBITAK (Project no. 118C356). However, the entire responsibility of the paper belongs to the owner of the paper. The financial support received from TÜBITAK does not mean that the content of the publication is approved in a scientific sense by TÜBITAK. This work was originally done in early 2020 and experiments section was updated in January 2022.

Appendix A. Sample appendix section

(a) Visible Image (b) Infrared Image (c) DenseFuse (d) SiameseFuse-addition (e) SiameseFuse-concat.

Fig. A1. A car was heated under the sun and, therefore, it emits infrared radiation while under the bridge.

(a) Visible Image (b) Infrared Image (c) DenseFuse (d) SiameseFuse-addition (e) SiameseFuse-concat.

Fig. A2. The text on the awning and the man in dark can be seen clearly in the fused image.

(a) Visible Image (b) Infrared Image (c) DenseFuse (d) SiameseFuse-addition (e) SiameseFuse-concat.

Fig. A3. In the fused image, the shine of the headlines can be seen applied onto the infrared heat glow of the man.

(a) Visible Image (b) Infrared Image (c) DenseFuse (d) SiameseFuse-addition (e) SiameseFuse-concat.

Fig. A4. Notice that, in this sample image, the blurry effect of the thermal camera is fixed in the fused images.

References

[1] M. Babaee, D.T. Dinh, G. Rigoll, A deep convolutional neural network for video sequence background subtraction, Pattern Recognit. 76 (2018) 635–649.

[2] R. Valiente, M. Zaman, S. Ozer, Y.P. Fallah, Controlling steering angle for cooperative self-driving vehicles utilizing CNN and LSTM-based deep networks, in: 2019 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2019, pp. 2423–2428.

[3] R. Collobert, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 160–167.

[4] A. Sellami, S. Tabbone, Deep neural networks-based relevant latent representation learning for hyperspectral image classification, Pattern Recognit. (2021), doi:10.1016/j.patcog.2021.108224.

[5] H. Li, X.-J. Wu, J. Kittler, Infrared and visible image fusion using a deep learning framework, in: 2018 24th International Conference on Pattern Recognition (ICPR), IEEE, 2018, pp. 2705–2710.

[6] H. Chen, Y. Li, D. Su, Multi-modal fusion network with multi-scale multi-path and cross-modal interactions for RGB-D salient object detection, Pattern Recognit. 86 (2019) 376–385.

[7] J. Yang, Y.Q. Zhao, J.C.W. Chan, Hyperspectral and multispectral image fusion via deep two-branches convolutional neural network, Remote Sens. 10 (5) (2018) 800.

[8] Z. Shao, J. Cai, Remote sensing image fusion with deep convolutional neural network, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 11 (5) (2018) 1656–1669.

[9] Z. Chen, R. Cong, Q. Xu, Q. Huang, DPANet: depth potentiality-aware gated attention network for RGB-D salient object detection, IEEE Trans. Image Process. 30 (2020) 7012–7024.

[10] W. Zhou, Q. Guo, J. Lei, L. Yu, J.N. Hwang, ECFFNet: effective and consistent feature fusion network for RGB-T salient object detection, IEEE Trans. Circuits Syst. Video Technol. 32 (3) (2021) 1224–1235.

[11] A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth, Occam's razor, Inf. Process. Lett. 24 (6) (1987) 377–380.

[12] H. Li, X.J. Wu, Densefuse: a fusion approach to infrared and visible images, IEEE Trans. Image Process. 28 (5) (2018) 2614–2623.

[13] J. Ma, H. Xu, J. Jiang, X. Mei, X.P. Zhang, DDcGAN: a dual-discriminator conditional generative adversarial network for multi-resolution image fusion, IEEE Trans. Image Process. 29 (2020) 4980–4995.

[14] J. Ma, Y. Ma, C. Li, Infrared and visible image fusion methods and applications: a survey, Inf. Fusion 45 (2019) 153–178.

[15] M.A. Özkanoğlu, S. Ozer, InfraGAN: a GAN architecture to transfer visible images to infrared domain, Pattern Recognit. Lett. (2022), doi:10.1016/j.patrec.2022.01.026. https://www.sciencedirect.com/science/article/pii/S0167865522000332

[16] M. Amin-Naji, A. Aghagolzadeh, M. Ezoji, Ensemble of CNN for multi-focus image fusion, Inf. Fusion 51 (2019) 201–214.

[17] D. Smith, S. Singh, Approaches to multisensor data fusion in target tracking: a survey, IEEE Trans. Knowl. Data Eng. 18 (12) (2006) 1696–1710.

[18] Z. Wang, Y. Wu, Q. Niu, Multi-sensor fusion in automated driving: a survey, IEEE Access 8 (2019) 2847–2868.

[19] M. Eslami, A. Mohammadzadeh, Developing a spectral-based strategy for urban object detection from airborne hyperspectral TIR and visible data, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 9 (5) (2015) 1808–1816.

[20] C. O'Conaire, N.E. O'Connor, E. Cooke, A.F. Smeaton, Comparison of fusion methods for thermo-visual surveillance tracking, in: 2006 9th International Conference on Information Fusion, IEEE, 2006, pp. 1–7.
[21] H. Li, W. Ding, X. Cao, C. Liu, Image registration and fusion of visible and infrared integrated camera for medium-altitude unmanned aerial vehicle remote sensing, Remote Sens. 9 (5) (2017) 441.
[22] C. Xing, M. Wang, C. Dong, C. Duan, Z. Wang, Using taylor expansion and convolutional sparse representation for image fusion, Neurocomputing 402 (2020) 437–455.
[23] J. Ma, W. Yu, P. Liang, C. Li, J. Jiang, Fusiongan: a generative adversarial network for infrared and visible image fusion, Inf. Fusion 48 (2019) 11–26.
[24] J. Chen, K. Wu, Z. Cheng, L. Luo, A saliency-based multiscale approach for infrared and visible image fusion, Signal Process. 182 (2021) 107936.
[25] J. Zhang, J. Shao, J. Chen, D. Yang, B. Liang, Polarization image fusion with self-learned fusion strategy, Pattern Recognit. 118 (2021) 108045.
[26] C. Zhao, G. Shao, L. Ma, X. Zhang, Image fusion algorithm based on redundant-lifting NSWMDA and adaptive PCNN, Optik 125 (20) (2014) 6247–6255.
[27] X. Lu, B. Zhang, Y. Zhao, H. Liu, H. Pei, The infrared and visible image fusion algorithm based on target separation and sparse representation, Infrared Phys. Technol. 67 (2014) 397–407.
[28] Y. Liu, X. Chen, H. Peng, Z. Wang, Multi-focus image fusion with a deep convolutional neural network, Inf. Fusion 36 (2017) 191–207.
[29] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Conference Track Proceedings of the 3rd International Conference on Learning Representations, ICLR, IEEE, San Diego, CA, USA, 2015.
[30] T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.
[31] S. Hwang, J. Park, N. Kim, Y. Choi, I. So Kweon, Multispectral pedestrian detection: benchmark dataset and baseline, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1037–1045.
[32] A. Srivastava, L. Valkov, C. Russell, M.U. Gutmann, C. Sutton, VEEGAN: reducing mode collapse in GANs using implicit variational learning, Adv. Neural Inf. Process. Syst. 30 (2017) 3308–3318.
[33] A. Fang, X. Zhao, J. Yang, Y. Zhang, X. Zheng, Non-linear and selective fusion of cross-modal images, Pattern Recognit. 119 (2021) 108042.
[34] J. Ma, P. Liang, W. Yu, C. Chen, X. Guo, J. Wu, J. Jiang, Infrared and visible image fusion via detail preserving adversarial learning, Inf. Fusion 54 (2020) 85–98, doi:10.1016/j.inffus.2019.07.005. https://www.sciencedirect.com/science/article/pii/S1566253519300314.
[35] K. Ma, K. Zeng, Z. Wang, Perceptual quality assessment for multi-exposure image fusion, IEEE Trans. Image Process. 24 (11) (2015) 3345–3356.
[36] V. Aslantas, E. Bendes, A new image quality metric for image fusion: the sum of the correlations of differences, AEU Int. J. Electron. Commun. 69 (12) (2015) 1890–1896.
[37] C. Xydeas, V. Petrovic, Objective image fusion performance measure, Electron. Lett. 36 (4) (2000) 308–309.
[38] M.B.A. Haghighat, A. Aghagolzadeh, H. Seyedarabi, A non-reference image fusion metric based on mutual information of image features, Comput. Electr. Eng. 37 (5) (2011) 744–756.
[39] M. Haghighat, M.A. Razian, Fast-FMI: non-reference image fusion metric, in: 2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT), 2014, pp. 1–3, doi:10.1109/ICAICT.2014.7036000.
[40] S. Razakarivony, F. Jurie, Vehicle detection in aerial imagery: a small target detection benchmark, J. Vis. Commun. Image Represent. 34 (2016) 187–203.
[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[42] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, 2015. http://arxiv.org/abs/1412.6980
[43] H. Xu, J. Ma, J. Jiang, X. Guo, H. Ling, U2fusion: a unified unsupervised image fusion network, IEEE Trans. Pattern Anal. Mach. Intell. 44 (1) (2022) 502–518, doi:10.1109/TPAMI.2020.3012548.

Sedat Ozer received his M.Sc. degree from Univ. of Massachusetts, Dartmouth and his Ph.D. degree from Rutgers University, NJ. He has worked as a research associate at Univ. of Virginia and Massachusetts Institute of Technology. His research interests include pattern analysis, object detection & segmentation, object tracking, visual data analysis, geometric and explainable AI algorithms and explainable fusion algorithms. As a recipient of TUBITAK's international outstanding research fellow and as an Assistant Professor, he is currently at Ozyegin University.

Mert Ege is currently a M.Sc. student at Bilkent University and is working on designing efficient algorithms for embedded systems in ASELSAN Inc. He received his B.Sc. degree from Bilkent University. His research interests include radar data analysis, deep learning and image fusion.

Mehmet Akif Özkanoğlu is currently a M.Sc. student at Bilkent University and is working on designing efficient algorithms for autonomous systems. He received his B.Sc. degree from Istanbul Technical University and his research interests include deep learning, object detection, tracking and robotics.