VISIBLE AND INFRARED IMAGE FUSION USING ENCODER-DECODER NEURAL NETWORK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FERHAT CAN ATAMAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2021

Approval of the thesis:

VISIBLE AND INFRARED IMAGE FUSION USING ENCODER-DECODER NEURAL NETWORK

submitted by **FERHAT CAN ATAMAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar Dean, Graduate School of **Natural and Applied Sciences**	__________
Prof. Dr. İlkay Ulusoy Head of Department, **Electrical and Electronics Engineering**	__________
Prof. Dr. Gözde Bozdağı Akar Supervisor, **Electrical and Electronics Engineering, METU**	__________

Examining Committee Members:

Prof. Dr. İlkay Ulusoy Electrical and Electronics Engineering, METU	__________
Prof. Dr. Gözde Bozdağı Akar Electrical and Electronics Engineering, METU	__________
Prof. Dr. A. Aydın Alatan Electrical and Electronics Engineering, METU	__________
Prof. Dr. Alptekin Temizel Graduate School of Informatics, METU	__________
Assoc. Prof. Dr. Seniha Esen Yüksel Electrical and Electronics Engineering, Hacettepe University	__________

Date: 07.09.2021

Name, Surname:	Ferhat Can Ataman
Signature :	

ABSTRACT

VISIBLE AND INFRARED IMAGE FUSION USING ENCODER-DECODER NEURAL NETWORK

Ataman, Ferhat Can
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Gözde Bozdağı Akar

September 2021, 129 pages

The image fusion aims to gather all important information from the source images into a single image. While the data is reduced, the fusion image has a high spatial and spectral resolution. It includes more informative and complete information. In this work, we reviewed state-of-the-art methods in the infrared and visible spectrum image fusion literature and we present a novel deep learning-based solution. Our proposed method is inspired by encoder-decoder network U-Net architecture [1]. Furthermore, we analyzed the fusion quality measurement metrics. We integrated fusion quality measurements into our proposed method's training step. In this way, we achieved superior performance. The analysis is performed qualitatively and quantitatively on TNO [2] and VIFB [3] datasets. The proposed method is compared with state-of-the-art methods and detailed experiments are conducted. It shows the best performance among deep learning-based methods. Project codes can be found at `https://github.com/ferhatcan/pyFusionSR`.

Keywords: image fusion, visible image, infrared image, encoder-decoder network

ÖZ

KODLAYICI-KOD ÇÖZÜCÜ SİNİR AĞI İLE KIZILÖTESİ VE GÖRÜNÜR SPEKTRUM GÖRÜNTÜLERDE FÜZYON

Ataman, Ferhat Can
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Eylül 2021 , 129 sayfa

Görüntü füzyonunun amacı kaynak olarak kullanılan görüntülerdeki bütün önemli bilgilerin tek bir görüntüde toplanması işlemidir. Data boyutu azalırken, üretilen füzyon görüntüsü yüksek uzamsal ve spektral çözünürlüğü sahip olur. Bu görüntü daha bilgi verici ve eksiksiz bilgileri içerir. Bu çalışmada literatürde en iyi olan methodları inceledik ve derin öğrenme tabanlı orjinal bir çözün sunduk. Ayrıca, füzyon kalite ölçümü için kullanılan metrikler analiz edildi. Sunduğumuz modelin eğitim aşamasına füzyon kalite metriklerini dahil ettik. Nitelikli ve nicelikli analizler TNO [2] ve VIFB [3] veri kümelerinde uygunlanmıştır. Sunulan yöntem literatürdeki en iyi metotlar ile karşılaştırılmıştır ve detaylı deneyler yapılmıştır. Derin öğrenme tabanlı yöntemler arasında en iyi sonucu vermektedir. Proje kodları şu linkte bulunabilir `https://github.com/ferhatcan/pyFusionSR`.

Anahtar Kelimeler: görüntü füzyon, görünür görüntü, kızılötesi görüntü, kodlayıcı-kod çözücü ağı

To my lovely wife Zehra, Ataman and Bektaş families...

ACKNOWLEDGMENTS

I would like to express my special thanks to gratitude to my supervisor Prof. Dr. Gözde Bozdağı AKAR for her guidance and encouragement. I am very proud to work with her. Also, I wish to thank my colleagues in ASELSAN A.Ş. and Innovita Technologies for their support and ideas to improve this work. Especially, Tolga AKSOY and Yunus Emre KILIÇER help me a lot while discussing and generating ideas and they deserve sincere thanking.

I am also grateful to my wife Zehra, my parents, and my friends for their patience, support, and belief in me.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
AI	Artificial Intelligence
DL	Deep Learning
MLP	Multilayer Perceptrons
MST	Multi-scale transform
BF	Bilateral filter
2DSVD	2D multi-resolution singular value decomposition
GF	Guided filter
DCT	Discrete Cosine Transforms
OMP	Orthogonal Matching Pursuit
GOMP	Group Orthogonal Matching Pursuit
PCA	Principal Component Analysis
ICA	Independent Component Analysis
NMF	Non-negative matrix factorization
LP	Laplacian Pyramid
GP	Gradient Pyramid
DWT	Discrete Wavelet Transform
SWT	Stationary Wavelet Transform
DTCWT	Dual-tree Complex Wavelet Transform
CVT	Curvelet Transform
NSCT	Nonsubsampled Contourlet Transform
VSM	Visual Saliency Map
SSIM	Structural Similarity
GAN	Generative Adversarial Network

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Imaging is a process that captures electromagnetic radiations in a spatial range. Visible spectrum imaging collects visible lights in the range 400-700 nm and converts them to digital values that are arranged and perceived as an image by humans. The visible imaging cameras are designed to replicate human vision by capturing red, green, and blue wavelengths of the visible electromagnetic spectrum. They collect reflected lights from the objects. Their performance is affected by atmospheric conditions such as fog, clouds, smoke, etc. They need good illumination to construct an image. Thus, in night conditions, they show poor performance. On the other hand, the source of thermal imaging is the heat of the objects. All objects that have a temperature higher than absolute zero in Kelvin (O°K of -273°C) emit electromagnetic radiation which cannot be seen by the naked eye [48]. When the heat of the object increases, the electromagnetic radiation increases as well. This radiation places mostly in the infrared wavelength. It can penetrate some atmospheric conditions like fog, smoke, rain, etc. Infrared imaging converts infrared radiation to the image that renders the spatial distribution of the temperature differences in the scene.

In imaging technology, there are two main resolutions definition that define the quality of the image. The first one is spatial resolution. It specifies the physical pixel size of the imaging device. Details and features of the objects in the scene can be seen with high spatial resolution more clearly [49]. The second one is spectral resolution. It specifies the sensitivity of the imaging device to the electromagnetic spectrum. The number of spectral bands and width of the spectral bands that the device is sensitive are the important parameters for the spectral resolution. Visible spectrum images use red, green, and blue spectral bands whereas infrared images use thermal spectral bands which the wavelengths are from the red spectral band edge around 700 nanometers to 1 millimeter. In current imaging technology, visible spectrum images have better spatial resolution than infrared images. However, thermal images use a different spectral band that reveals a different kinds of features and details of the objects. Due to it uses the infrared radiation of the scene, it can operate in day and night conditions without losing any information. Also, in bad atmospheric conditions such as fog, smoke, etc., it provides a sight because of the nature of infrared imaging. Thus, both visible and infrared images have advantages and they are required for different real-world scenarios.

The imaging technology is rapidly improving and sensors become more and more informative. The images are used in various applications so, the quality of the image is important. In surveillance, imaging technology is excessively used. The objects in the scene should be distinguishable. The operators should need to identify objects, read the important information in the image. For instance, they need to read the car plate from far distances. The spatial and spectral resolution is so important for them. Also, they need to keep track of the multiple-image sources which can be time-consuming and some important features can be skipped. Reducing the multiple images to a single image makes easy their jobs. On the other hand, images are used for machine perception applications which are called computer vision. Relevant applications work top on the image such as object classification, object detection, object tracking, etc. These applications need high spatial resolution and can benefit from high spectral resolution. The applications generally run through the multi-dimensional images that are stacking of the visible and infrared images. This behavior increases the time or memory consumption of the applications. Rather than work on multi-spectral images, the application can use a single image that includes all relevant information regarding the scene. To address these issues, image fusion is defined. Image fusion is a technique that combines different kinds of images to generate a single image that includes all important information without any artifact. In this work, we are focusing on infrared and visible image fusion applications. It gathers all important information from visible and infrared images to construct a complete single image. The aim is to create a new image that includes more understandable and compact features of the scene while reducing the amount of data. By the way, the spatial and spectral resolution of the fused image increase. It completes the weaknesses of both images. The solution decreases the need for complex physical solutions and costs. These two imaging modalities offer different aspects of the scene. For instance, visible spectrum imaging provides color and shape information by capturing reflected light from the scene. Whereas, infrared spectrum imaging constructs an image by capturing thermal radiation. The source of the image formation affects the visibility, shape, color, and other aspects of the objects in the digital image. The visible images can be influenced by the scene conditions such as fogging, raining. Thermal images are resistant to such conditions but, they have low resolution and poor texture information. Combining the advantages of these different imaging technologies can boost the performance of the computer vision applications mentioned above or present a quality image to the user, i.e. a human that reviews the camera footage.

The spatial and spectral resolutions of the image are important but, deciding which information is important and needs to be preserved when fusing images. Measuring the quality of the fused image is very challenging and it is changing in the direction of the requirements. Moreover, there is no exact true fusion result to understand and examine the solution. Thus, image quality is measured through important features for desired tasks. For instance, image contrast and brightness are important for the human visual system in surveillance applications. Considering these features and applications, image quality is measured with different metrics.

1.2 Contributions and Novelties

Our contributions are as follows:

- We propose a novel neural network architecture to address the fusion problem without needing any formulation or additional steps. All parameters in the proposed architecture are trained together. Thus, the process is optimized to learn to extract complex interactions and features, to fuse these features, and reconstruct the fused feature as a whole. Also, in the proposed architecture, we use only convolutional and pooling layers which decreases the computational load and makes it more suitable for real-time applications.

- We propose a novel loss function using no-reference quality metrics. The mean square error function is also integrated into the loss function to increase visual perception quality. In this way, generated color images become more realistic and natural. To measure the perceived qualities of our results, we use no reference perception quality metric mentioned in [42].

1.3 The Outline of the Thesis

In the Chapter 1, background information about deep leaning is given. It is required to understand the concepts of the proposed method. The chapter starts with the history of deep learning and continues to explain core concepts on it. In the end, common methods and novelties are explained.

In the Chapter A, image fusion literature is reviewed. The chapter consists of three parts: image fusion methods, fusion quality measurements, and applications of image fusion.

In the Chapter 3, our proposed method is presented with great details. The architecture and the training process are explained. The design choices of the proposed method are also given.

In the Chapter 4, experiment results are shared. In the first part, experiments conducted on the proposed method are given. In the second part, the best-proposed method is compared with the state-of-the-art methods quantitatively and qualitatively. The image quality measurements mentioned in Chapter A are used for the comparisons. Also, visual results are shared.

CHAPTER 2

CONVOLUTIONAL NEURAL NETWORKS (CNNS)

2.1 Introduction

Convolutional neural networks are a special kind of neural network for processing data. It uses convolutional operation instead of matrix multiplication. It becomes sparse relative to feed-forward networks also known as fully connected networks because weights are shared for input data. Thus, the network parameters are significantly decreased. Also, it is shown that CNNs are successful in various applications. It can extract good internal representations. These reasons make CNNs popular.

We continue with an explanation of some common architectures.

2.1.1 The Convolutional Operation

Convolution is an operation that takes two input function. It can be considered that it is a weighted average operation as defined in Eqn. 2.1. One of the input functions defines weights whereas the other one is input signal. The weight input is called as kernel and output of convolution operation is called as feature map in deep learning area. In CNNs, 2D convolution operations are used commonly which is given in Eqn. 2.2. The kernel is generally much more smaller than the input image which equals to kernel has non-zero elements in a a bounded region in 2D. For example, kernel size is generally taken as 3 by 3, 5 by 5 or similar, whereas the image size generally bigger than 32 by 32.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.1}$$

$$S(i,j) = (K * I)(i,j) = \sum_{m}\sum_{n} I(i-m, j-n)K(m,n) \tag{2.2}$$

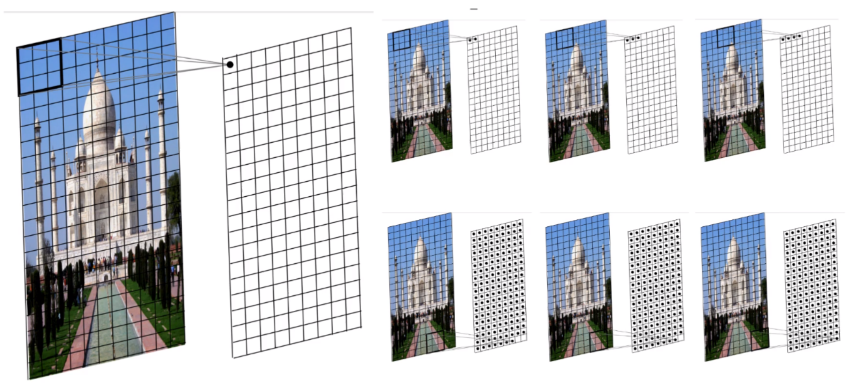


Figure 2.1: Convolution operation on an image. Image taken from [32]

2.2 Some Common Architectures of CNNs

The convolutional neural networks become popular when they show the best performance in various fields like image classification, object detection. There are some architectures that present innovations to improve the concepts. Thus, the innovative architectures should be reviewed to understand key points that bring success. The VGG, ResNet and U-Net architectures are fundamental methods in CNNs.

2.2.1 VGG [4]

VGG [4] architecture was proposed by Karen Simonyan and Andrew Zisserman from the University of Oxford in 2014. Their main contribution is to add smaller kernel-size convolution layers with deeper networks. They introduced CNN models that have 16 and 19 layers. Previously, in AlexNet, 8 layer CNN was proposed with 11 by 11 kernel sizes. In VGG, layers have 3 by 3 kernels. Decreasing kernel size of the convolution layer decreased the number of network parameters and increasing depth of the network increased accuracy as well. Thus, VGG showed that deeper networks with small receptive fields, i.e. small kernel size, achieve better results. VGG-16 architecture is shown in Figure 2.2.

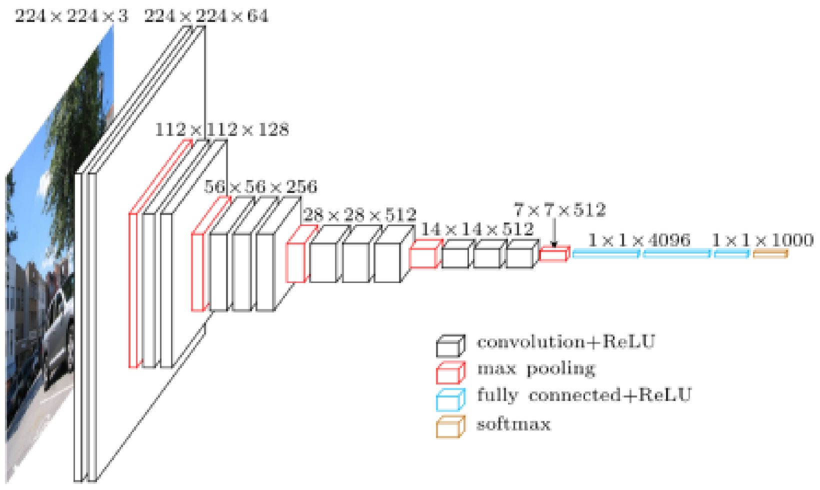


Figure 2.2: VGG-16 architecture taken from [33].

2.2.2 ResNet [5]

VGG network showed increasing the depth of the network increases accuracy as well. However, simply stacking layers do not work because vanishing gradient problem when training network. Thus, ResNet [5] presented a novel solution to prevent from vanishing gradient problem. The core idea is to add skip or shortcut connections between layers. With the help of shortcut layers, gradients can flow through the end of the network. The residual building block of the ResNet architecture is shown in Figure 2.3. The comparison between VGG network with ResNet is shown in Figure **??**. ResNet has only one fully connected layer with deeper architecture and the number of parameters is significantly dropped compared to VGG. Also, adding skip connections doesn't affect the size of the parameters but, training becomes more stable.

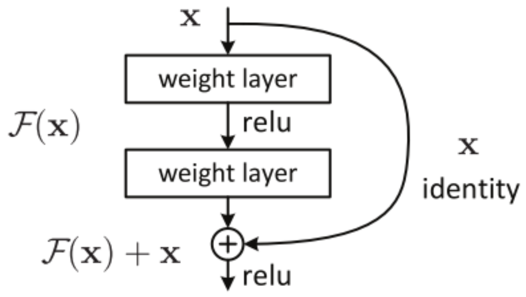


Figure 2.3: Residual building block for ResNet architecture taken from [5]

2.2.3 U-Net [1]

VGG and ResNet architectures are successful methods to extract features and classify images. To classify each pixel that is segmentation, the U-Net [1] architecture presented a novel method. It consists of two parts: encoder and decoder layers. The encoder layer extracts image features with convolution layers and downsamples input images with maximum pooling layers. The decoder layer reconstructs the segmentation result using extracted features from the encoder layer. It uses convolution transpose layers to upsample feature maps. The overall architecture is shown in Figure 2.4. A similar idea to the skip connection in ResNet is used between encoder and decoder layers. It becomes popular for image generation tasks such as segmentation and ResNet and VGG models are integrated into the encoder layer in further researches.

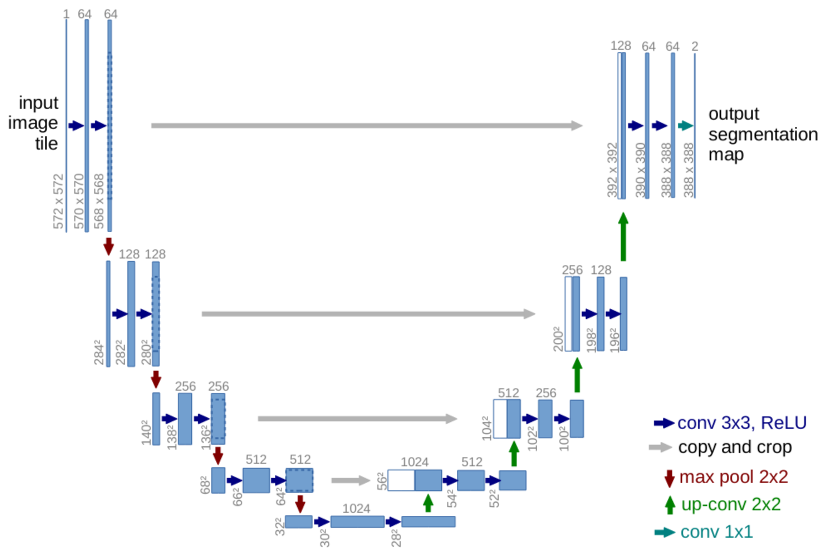


Figure 2.4: U-Net architecture. Gray arrows show skip connections of the architecture. The number of channels is denoted on top of the boxes. Image taken from [1].

CHAPTER 3

LITERATURE REVIEW

3.1 Introduction

In this chapter, we reviewed the existing methods and quality measurements for infrared and visible image fusion.

3.2 Infrared and Visible Image Fusion Methods

Fusion methods are divided into seven categories in [34]. These are multi-scale transform, sparse representation, neural network, subspace, saliency-based, hybrid, and other methods. Image fusion algorithms are composed of three steps: Extracting good representations of images, a fusion rule, and reconstruction of the fused image. Each method can be reviewed by following these steps. In this section, each category will be explained according to their theory briefly. Then, some common methods of the category will be reviewed.

3.2.1 Multi-scale Transform Based Methods

Multi-scale transform-based methods are the most common technique that is applied in various algorithms for image fusion. Input images are decomposed into several components of different scales. Thus, each component can represent different real-world object features. It is shown that multi-scale transforms are similar to the human visual system so, using this can give good visual effects in the final reconstructed image. One of the most fundamental transforms used in the multi-scale transform is pyramid transform and its concept dated back to the 1980s. In the Laplacian pyramid transform [50], the following steps are applied iteratively: low-pass filtering, sampling, interpolating, and differencing. Other transforms that is commonly used are wavelet transforms [51], nonsubsampled contourlet transform [52], edge-preserving filter [53], curvelet transform [54]. More details about these transforms can be found in [34]. After multi-scale decomposition, these representations are fused according to a fusion rule defined by the method. To combine the multi-scale representations, the global or regional fusion rule is defined. The methods compare the coefficients based on a metric that can extract image features like contrast, energy, and detail. According to the comparison result, multi-scale representations are combined. In the last step which is the reconstruction of fused image, the inverse transform is applied to fused multi-scale representations to generate the fused image. In Figure 3.1, the multi-scale transform-based infrared and visible image fusion methods' overall scheme is shown.

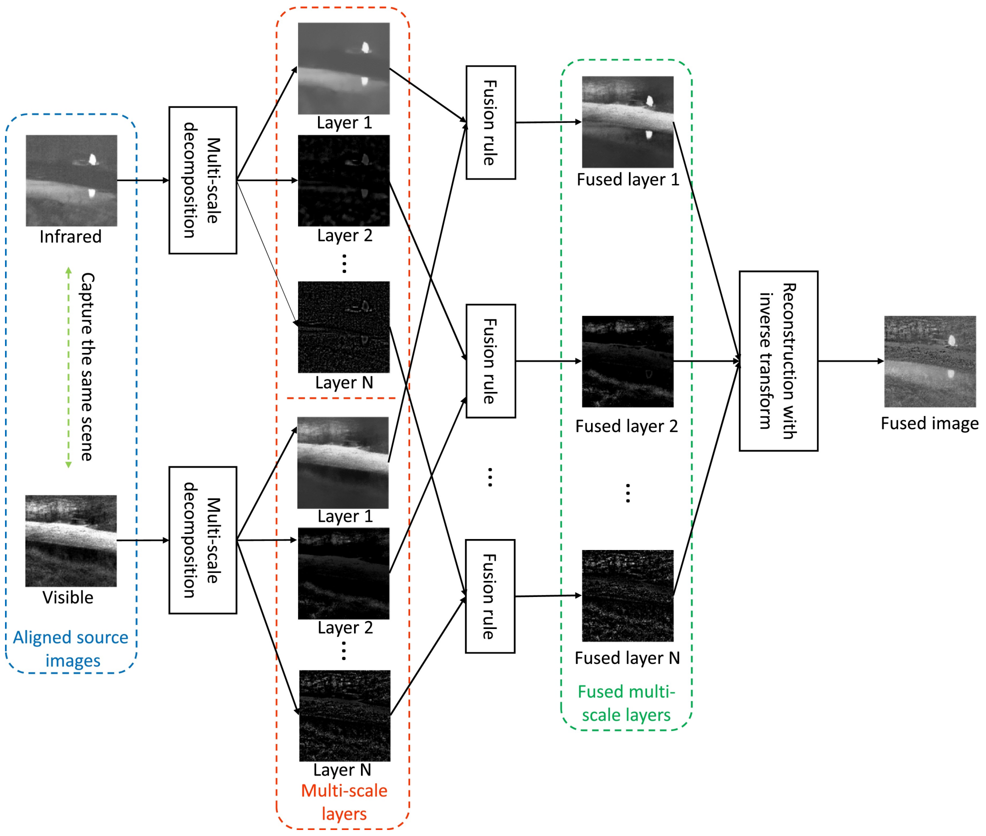


Figure 3.1: Multi-scale transform based infrared and visible image fusion scheme taken from [34].

3.2.1.1 ADF Method [6]

ADF [6] is shortening of Anisotropic diffusion-based image fusion. They use anisotropic diffusion to extract base and detail layers from the image. Then, they define Karhunen-Loeve transform [55] to fuse detail layers and weighted superposition to fuse base layers. In the final step, fused detail and base layers are superposed and the final fused image is generated. Proposed ADF method is shown in Figure 3.2.

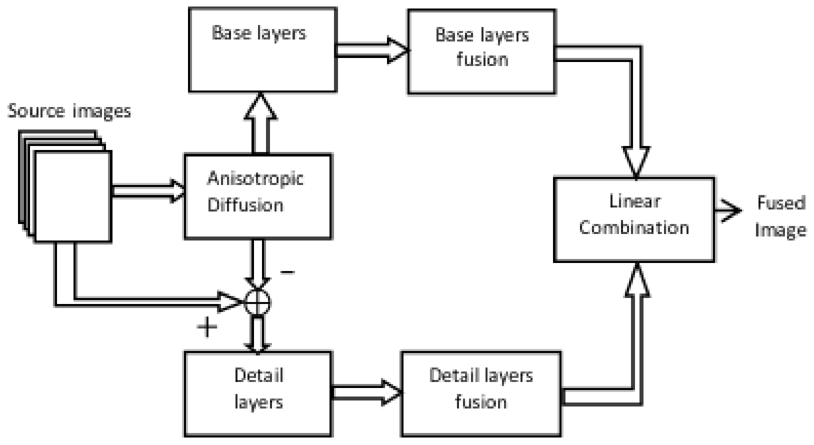


Figure 3.2: Proposed ADF Method taken from [6]

3.2.1.2 CBF Method [7]

CBF [7] uses cross bilateral filter to extract and fuse image features. The bilateral filter (BF) is introduced in [56]. The cross bilateral filter uses a second image to shape the filter kernel and operates in the first image. Thus, In CBF, the cross bilateral filter is used with both orders which are first and second images are visible and infrared and vice versa. Thus, representations for visible and infrared images are extracted and they generate pixel-based weights by measuring the strength of details for both of them. The images are combined with found weights to reconstruct the final fused image. Proposed CBF method is shown in Figure 3.3.

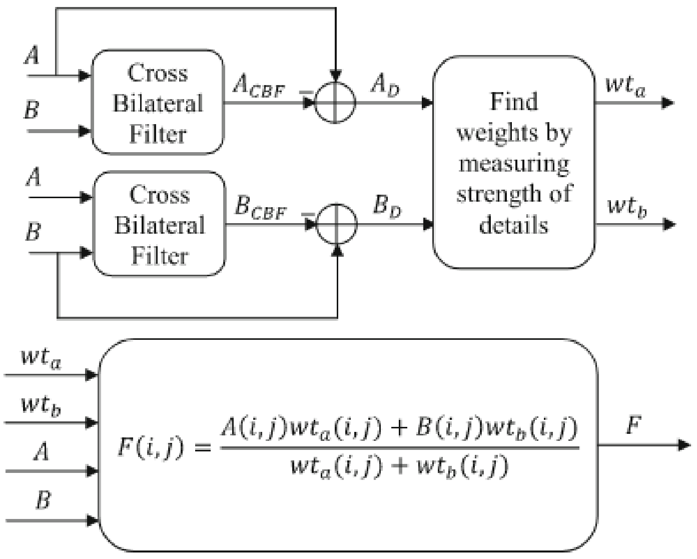

A
B
Cross Bilateral Filter
A_{CBF}
A_D
A
B
Cross Bilateral Filter
B_{CBF}
B_D
Find weights by measuring strength of details
wt_a
wt_b
wt_a
wt_b
A
B
$F(i,j) = \frac{A(i,j)wt_a(i,j) + B(i,j)wt_b(i,j)}{wt_a(i,j) + wt_b(i,j)}$
F

Figure 3.3: Proposed CBF Method taken from [7]

3.2.1.3 Hybrid_MSD Method [8]

Hybrid_MSD [8] uses Gaussian and bilateral filters [56] to extract representations of the input images. Changing bilateral filter parameter σ_r which is the standard deviation of the range Gaussians that control the influences of the neighboring pixels' intensity difference. With different σ_r values, they stated that large or small scale features of the input image can be extracted. They define three different levels which are small, large, and base levels. In Figure 3.4, decomposition of an input image for three-level Hybrid_MSD is shown. I_b^i shows the bilateral filter output where I_g^i shows the output of Gaussian filter with i^{th} level parameters. I^0 means input image whereas B means base level decomposition. After three-layer decomposition results are generated for infrared and visible images separately, corresponding decomposition layers are combined. Base layers and the other two layers are combined with different fusion rules. Detailed explanations can be found [8]. In the final step, fused image result is generated by applying inverse Hybrid_MSD transform to fused decomposition layers which are the sum of all levels of decomposed information with the base image. The overall architecture of the method is shown in Figure 3.5.

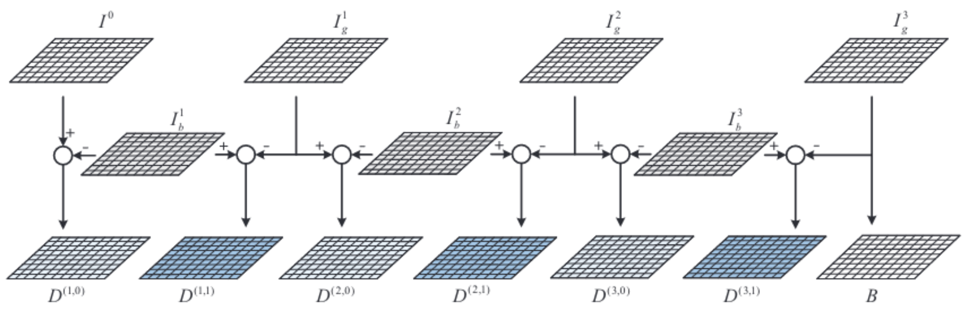


Figure 3.4: Construction of three level decomposition taken from [8]

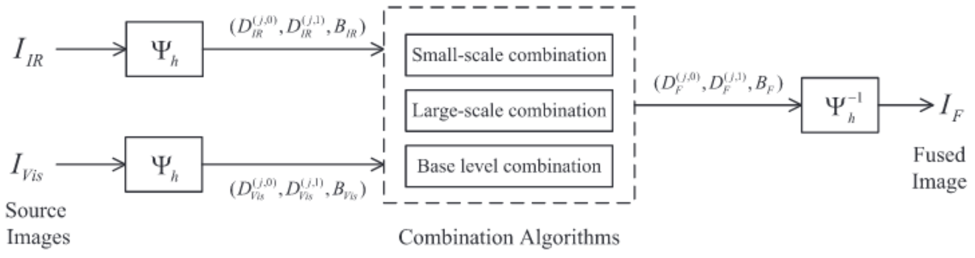


Figure 3.5: Proposed Hybrid_MSD Method taken from [8]

3.2.1.4 GFCE and HMSD_GF Methods [9]

In [9], they investigated that guided filter (GF) [57] can be used instead of bilateral filter [56] in the Hybrid_MSD method 3.2.1.3. They showed that GF can be a good edge-preserving filter or a good smoothing filter with adjusting related parameters of GF. Thus, the Gaussian filter and bilateral filter are replaced by GF. Proposed HMSD_GF method is shown in Figure 3.6. They grouped decomposed layers as Hybrid_MSD 3.2.1.3. Decomposed information combination is defined for each level separately and there is a regularization parameter that controls the relative amount of IR spectral information injected into the visible image to prevent some artifacts. The final fused image is reconstructed by summing all level combinations. The GFCE method enhances image quality with some additional steps to the HMSD_GF. Firstly, they apply a preprocessing step to enhance visible image quality. Also, they chose the regularization parameter with measuring the perceptual saliency of preprocessed input images. In Figure 3.7, steps of GFCE are shown.

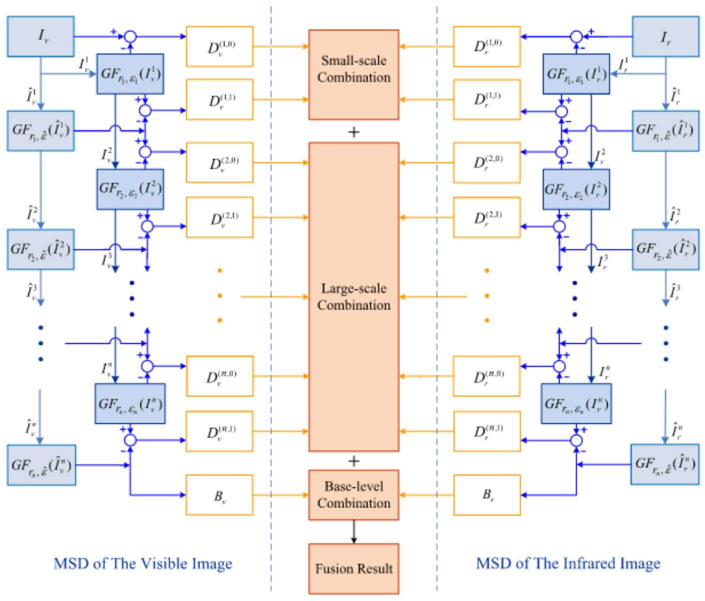


Figure 3.6: Proposed HMSD_GF Method taken from [9]

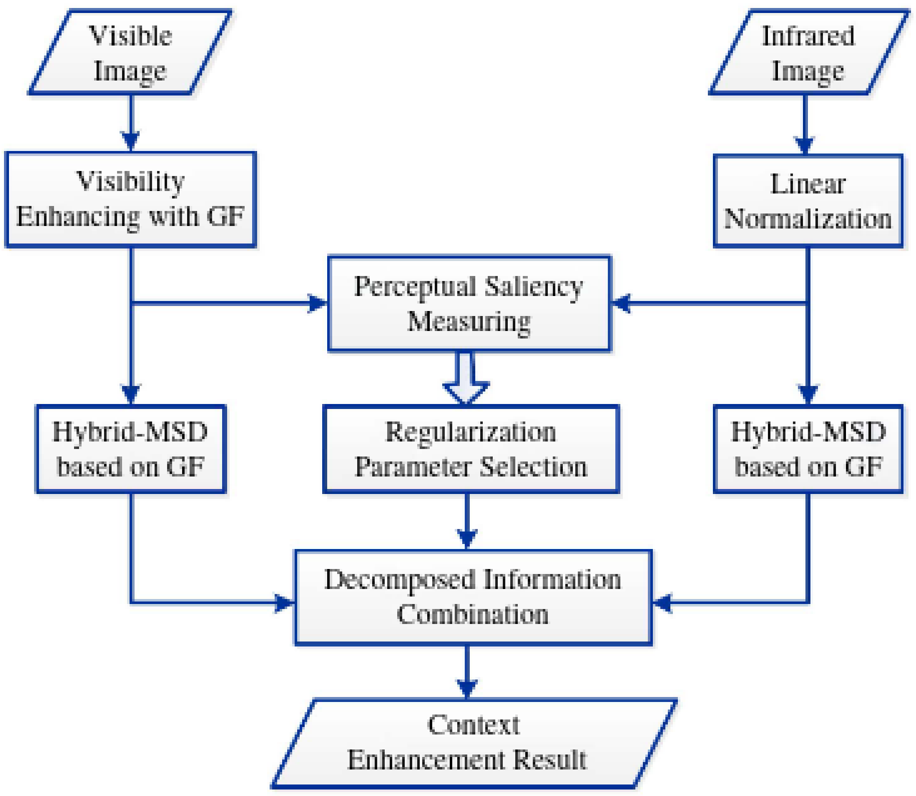

Visible Image
Infrared Image
Visibility Enhancing with GF
Linear Normalization
Perceptual Saliency Measuring
Hybrid-MSD based on GF
Regularization Parameter Selection
Hybrid-MSD based on GF
Decomposed Information Combination
Context Enhancement Result

Figure 3.7: Proposed GFCE Method taken from [9]

3.2.1.5 MSVD Method [10]

In MSVD [10], to decompose source images, 2D multi-resolution singular value decomposition (2DSVD) [58] is used. In the second step, according to the designed fusion rule, the larger absolute value of two corresponding detailed coefficients for each level. Only in the coarsest level (last level), an average of the coefficients is calculated. In the last step, inverse 2DSVD is applied since the steps are reversible. The proposed method is shown in Figure 3.8.

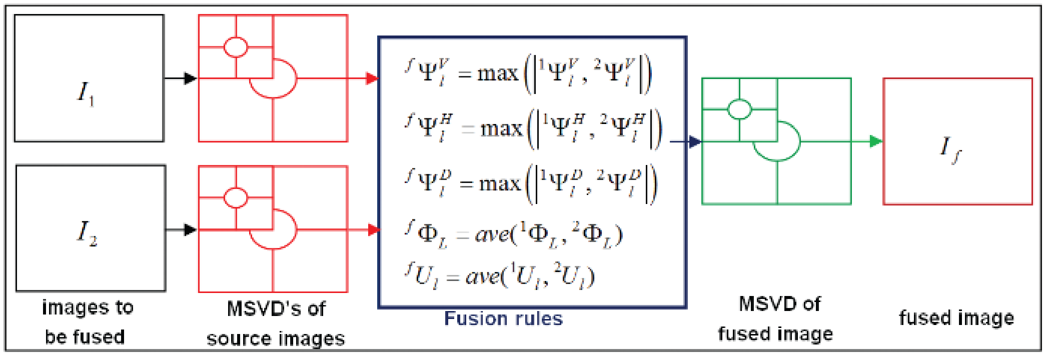

I_1
I_2
images to be fused
MSVD's of source images
${}^f\Psi_l^V = \max\left(\left|{}^1\Psi_l^V, {}^2\Psi_l^V\right|\right)$
${}^f\Psi_l^H = \max\left(\left|{}^1\Psi_l^H, {}^2\Psi_l^H\right|\right)$
${}^f\Psi_l^D = \max\left(\left|{}^1\Psi_l^D, {}^2\Psi_l^D\right|\right)$
${}^f\Phi_L = ave({}^1\Phi_L, {}^2\Phi_L)$
${}^fU_l = ave({}^1U_l, {}^2U_l)$
Fusion rules
MSVD of fused image
I_f
fused image

Figure 3.8: Proposed MSVD Method taken from [10]

3.2.2 Saliency Based Methods

Saliency regions are important for the human visual system because it attracts attention to itself. Furthermore, It can be distinguished by its neighborhood objects. To increase visual quality, the integrity of salient regions should be maintained for image fusion application. Generally, methods use the saliency information in their fusion method for example GFF 3.2.2.2 and MGFF 3.2.2.1.

3.2.2.1 MGFF [11]

MGFF [11] uses guided filter (GF) [57] to construct multi-scale decomposition layers. The approach is close to GFF 3.2.2.2 but it uses more than two scales and it uses guided filter when decomposing source images. Source images are regarded as 0^th base layer decomposition result. To generate the next base layer decomposition, GF is applied to the previous layer with the guidance of another image spectrum decomposition layer that is for visible image decomposition, infrared image decomposition is used as guidance vice versa. Furthermore, detail layer decomposition is calculated as consecutive base layer differences. In the second step of the method, the fusion rule is defined to reconstruct the fused image. The detail layers are combined with a weighted summation. The corresponding weights are calculated from salience maps which are the absolute value of the detail layers. Each corresponding detail layer of the source images is combined with these weights and all detail layers are superposed to produce the final detail layer. Also, the final base layer is calculated by superposing of last base layers of source images. In the last step, the fused image is reconstructed by superposing the final base layer and final detail layer. In Figure 3.9, methods steps are shown.

3.2.2.2 GFF [12]

GFF [12] is another method that integrates guided filter [57] to image fusion. The difference from other multi-scale methods is GFF uses only two scales to construct the fused image. In the first step, image representations are generated. These are called as base layer and detail layer. The base layer is generated using average filtering and the detail layer is generated by subtracting the base layer from source images. In the second step, they define a fusion rule. The Laplacian filter is applied to the source image to obtain a high pass image. Then, a Gaussian filter is applied to a high pass image to smooth the image and the result is called a saliency map. Saliency maps of source images are compared. For each pixel, if a saliency value is higher than the other's value, the pixel value becomes 1, otherwise becomes 0. Thus, saliency maps become binary images, called weight maps and they become complementary of each other, i.e, corresponding pixel sum becomes 1. The guided filter is applied to the weight maps with the guidance of the source image. The refined weight maps are normalized to provide a summation of corresponding pixel values equal to 1. In the last step, base and detail layers are combined with refined weight maps separately. Then, combined base and detail layers were superposed to construct the fused image. In Figure 3.10, steps are visualized.

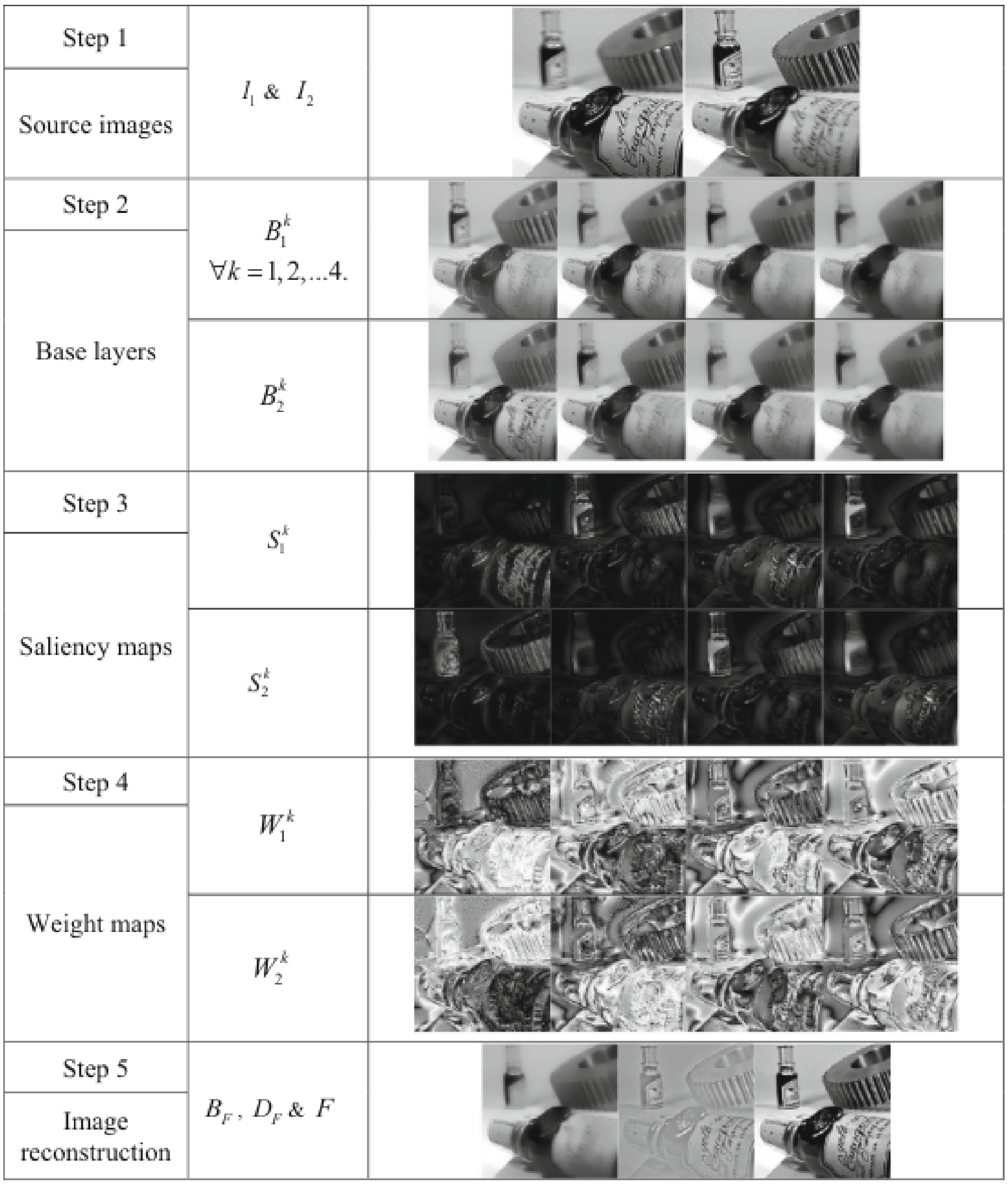
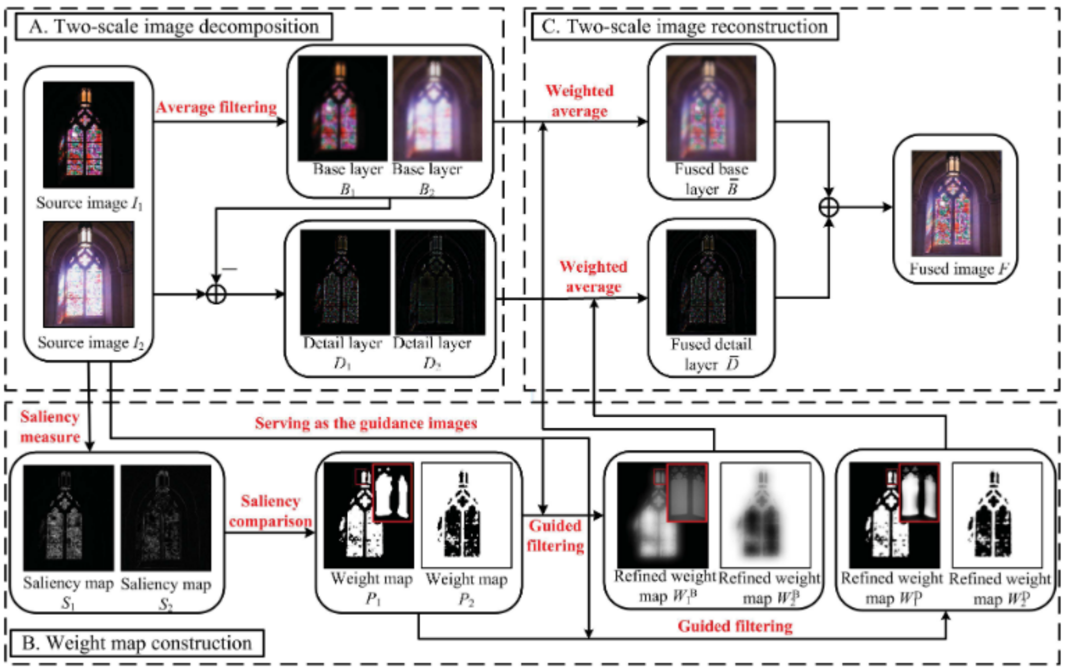


Figure 3.9: Proposed MGFF Method taken from [11]

3.2.2.3 TIF [13]

TIF [13] is similar to GFF 3.2.2.2. It uses two scales to construct fused images like GFF. In the first step, it extracts base layers of source images applying an average filter. In addition, the detail layers are extracted by subtracting base layers from source images. In the second step, the visual saliency detection algorithm is defined to define weight maps used for the fusion of detail layers. Saliency regions are found by the



A. Two-scale image decomposition
C. Two-scale image reconstruction
Average filtering
Source image I_1
Source image I_2
Base layer B_1
Base layer B_2
Detail layer D_1
Detail layer D_2
Weighted average
Fused base layer $\bar{B}$
Fused detail layer $\bar{D}$
Fused image F
Saliency measure
Serving as the guidance images
Saliency map S_1
Saliency map S_2
Saliency comparison
Weight map P_1
Weight map P_2
Guided filtering
Refined weight map W_1^B
Refined weight map W_2^B
Refined weight map W_1^D
Refined weight map W_2^D
B. Weight map construction

Figure 3.10: Proposed GFF Method taken from [12]

following process. A mean and median filter is applied to source images. Saliency maps are calculated by taking the absolute value of the differences between these filters applied for each source image separately. The weights maps are calculated by normalizing saliency maps that the summation of each corresponding pixel should be equal to 1. Detail layers are combined using calculated weight maps and base layers are combined simply taking average. In the last step, fused detail and base layers are superposed to construct the fused image. Figure 3.11 shows the schematic of explained method TIF.

3.2.3 Sparse Representation Based Methods

Sparse representation aims to learn a dictionary that can define images sparsely. Natural images can be defined by sparse features. Sparse representation reduces the dimension of images with the help of a learned dictionary. An example of sparse representation of image can be seen in Figure 3.12. Before sparse coding, the image is divided into several patches. The patches are defined by sparse coefficients and reconstructed from these coefficients. In this way, visual artifacts such as noise

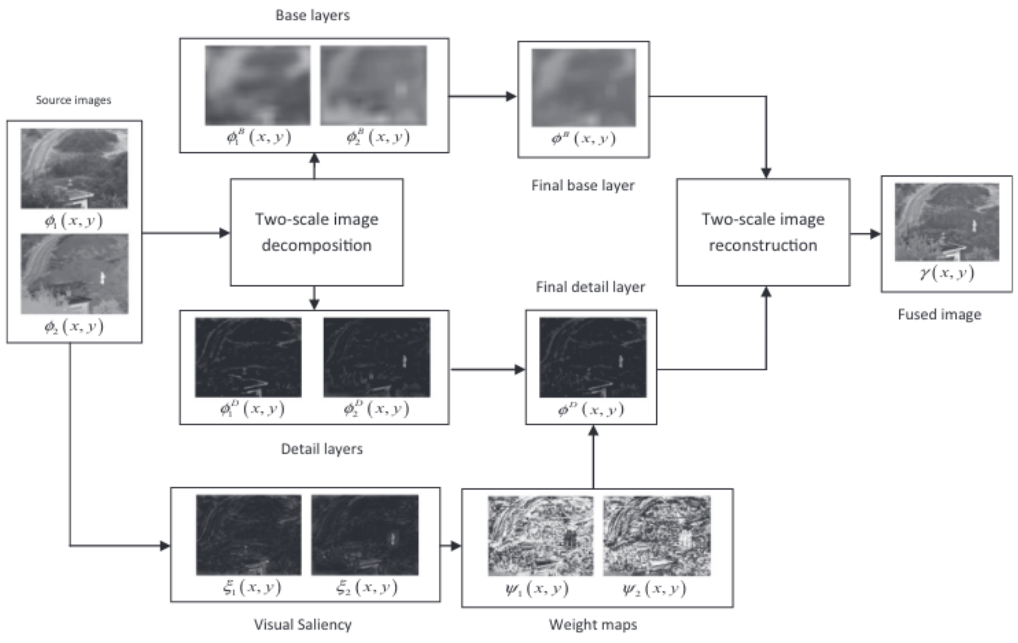


Figure 3.11: Proposed TIF Method taken from [13]

decreases, and the quality of the image potentially increases whenever the image is reconstructed from sparse coding [59]. In Figure 3.13, general architecture applied for sparse representation-based methods is shown. Algorithm steps are organized as:

1. Vectorize input image using sliding window strategy and apply sparse coding using learned dictionary
2. Define a fusion rule to combine sparse coefficients
3. Reconstruct image from fused sparse coefficients

3.2.3.1 SR Method [14]

In sparse representation, the source image is divided into patches because the dictionary size is growing exponentially when the patch size increases. Thus, smaller size patches lead to smaller size overcomplete dictionary and required computational power decreases as well. In SR method [14], source image is divided into patches with size of nxn. These patches are lexicographically ordered as a vector. All vectors are combined in a matrix in which each column corresponds to a vector. Then, sparse

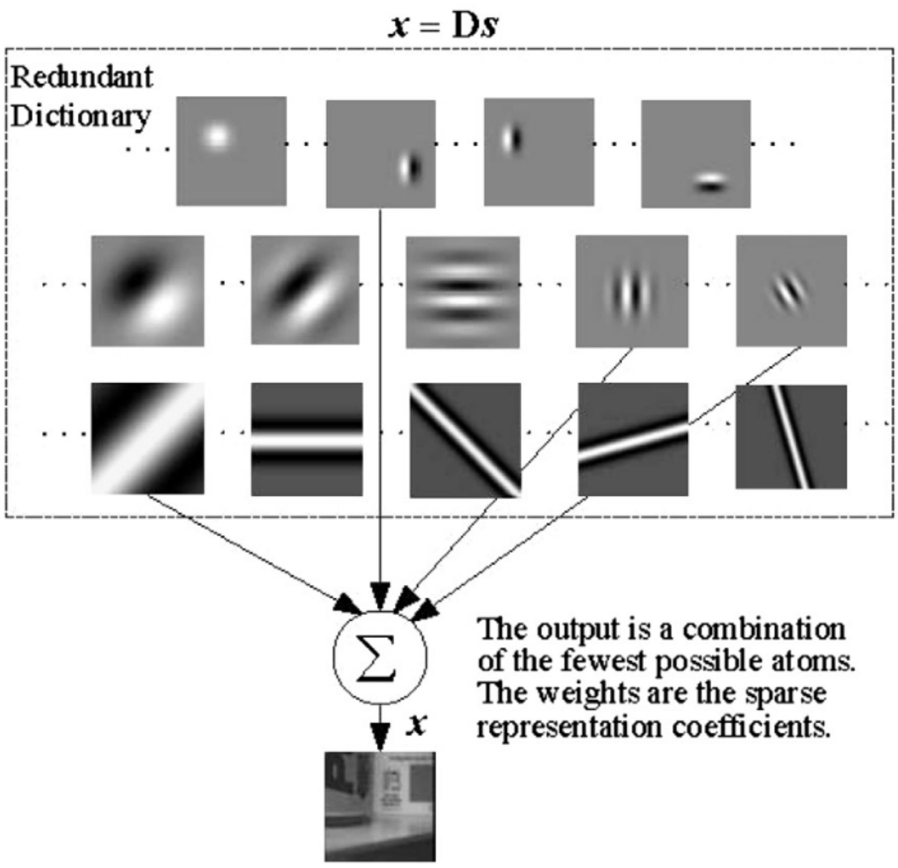

$x = Ds$
Redundant Dictionary
The output is a combination of the fewest possible atoms. The weights are the sparse representation coefficients.
Σ
x

Figure 3.12: Sparse representation of an image taken from [35]

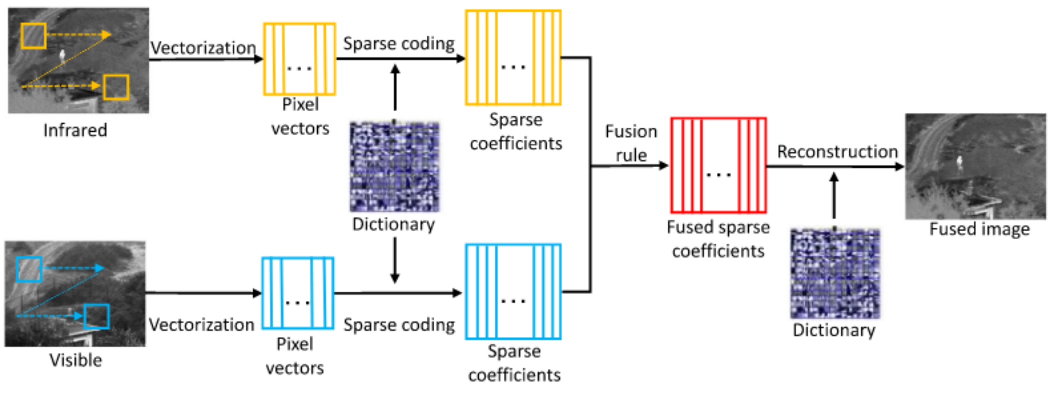


Figure 3.13: Sparse representation based fusion scheme taken from [34]

coefficients are found using given overcomplete dictionary. The dictionary that is used in this method is constructed with discrete cosine transforms (DCT). Sampling the cosine wave in different frequencies is adopted for the DCT dictionary. To calculate best matching sparse representation, orthogonal matching pursuit (OMP) [60] which is a greedy algorithm that sequentially selects the dictionary elements. In the second step of the image fusion, sparse coefficients of the source images are combined choosing the largest activity level in each pixel location. In the last step, fused sparse coefficient converted to lexicographically ordered vector matrix. Then, the final fused image is constructed inverse process of vectorization explained above. If any pixel value comes from different patches, the average of them is taken. The overall architecture of the proposed method is shown in Figure 3.14.

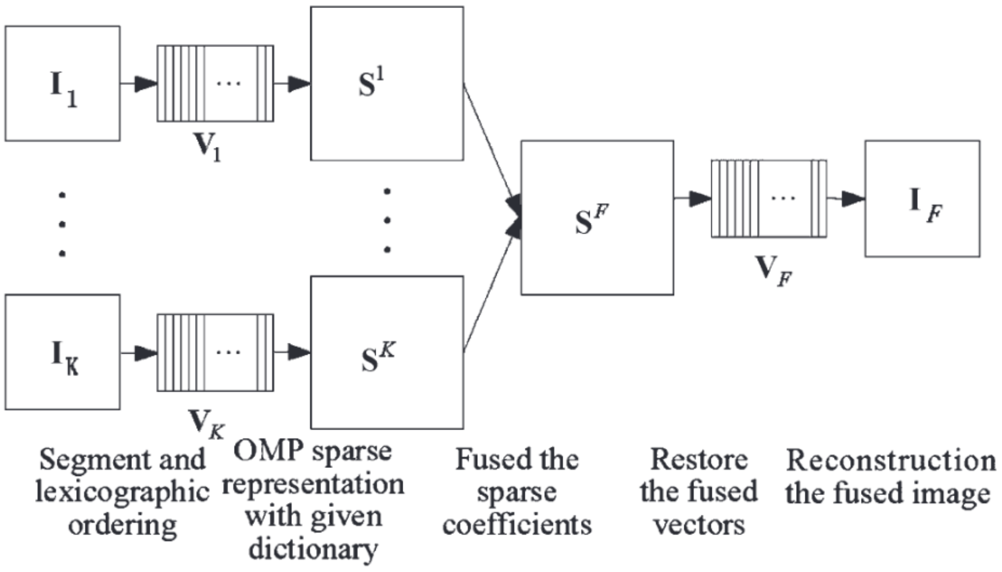

I_1
V_1
S^1
I_K
V_K
S^K
S^F
V_F
I_F
Segment and lexicographic ordering
OMP sparse representation with given dictionary
Fused the sparse coefficients
Restore the fused vectors
Reconstruction the fused image

Figure 3.14: Overall architecture of SR method taken from [14]

3.2.3.2 SOMP [15]

Similar to the SR method 3.2.3.1, the SOMP method uses a given dictionary to construct sparse coefficients of the source images. Before applying the proposed method to the source images, a dictionary is constructed to define sparse representation. They use three kinds of overcomplete dictionaries. The first one is an overcomplete DCT dictionary. The second one is the hybrid overcomplete dictionary that consists of DCT bases, wavelet 'db1' bases, Gabort bases, and ridgelet bases functions. The last one is the trained overcomplete dictionary that uses 50000 8 by 8 sample blocks from 50 natural images to learn dictionary atoms. In Figure 3.15, dictionary types are shown.

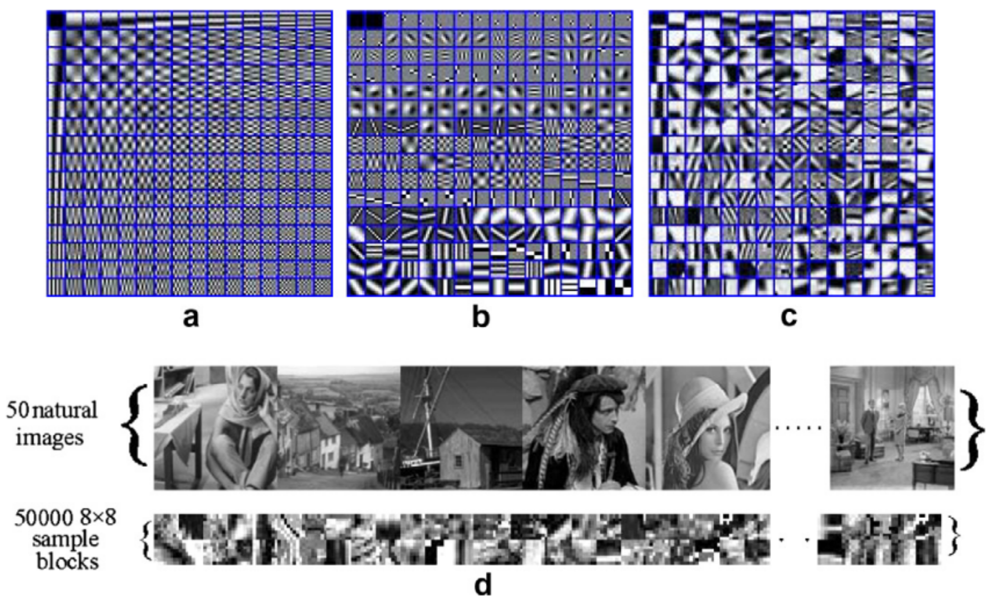


Figure 3.15: Over-complete dictionaries and the training data. The overcomplete DCT dictionary, the hybrid overcomplete dictionary, and the trained overcomplete dictionary from left to right respectively. The bottom row shows the training data. Credit to [15]

In the first step, source images are divided into patches using a sliding window. Patch size is the same as the atom size of the given dictionary. The patches are transformed into vectors via lexicographical ordering and all of them are combined in a vector matrix. To extract sparse representations, a simultaneous orthogonal matching pursuit algorithm [35] is applied. In the second step, sparse coefficients are combined using the absolute maximum rule. The maximum absolute value of coefficients is chosen to define fused sparse coefficients. Lastly, fused sparse coefficients are converted to fused vectors. Each vector is reshaped into the original patch size. Each reshaped patch is added to the corresponding region in the image. Thus, each pixel consists of several patch values. The final pixel value is divided by the number of patches used to construct the pixel. Thus, the final fused image is reconstructed. The overall architecture of the proposed method is shown in Figure 3.16.

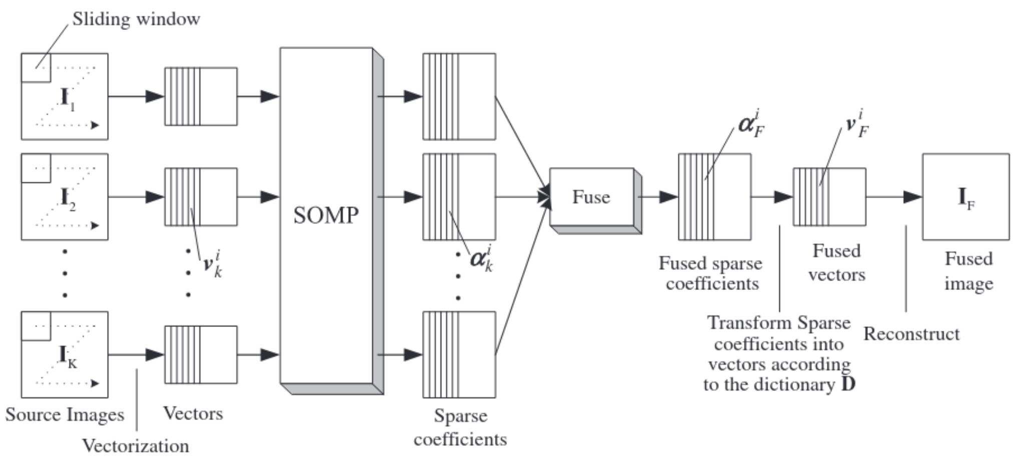


Figure 3.16: Overall architecture of SOMP method taken from [15]

3.2.3.3 GSR Method [16]

In GSR method [16], fused image reconstruction steps are similar to other sparse representation-based methods. The difference of the proposed method, it uses group sparse representation to define a sparse representation of the image. It combines different types of dictionaries. It uses group orthogonal matching pursuit (GOMP) to derive sparse coefficients of vectorized source image patches. In the first step of the proposed method, source images are divided into overlapping patches with a square area. Each path is normalized by extracting the mean value of the patch from each pixel value in the patch. Then, The normalized patches are transformed into vectors via lexicographical ordering and all of them are combined in a vector matrix. The sparse coefficients are calculated using GOMP. In the second step, a fusion rule is defined to combine group sparse coefficients. Maximum l2-norm of sparse coefficients are selected. Also, to reconstructed a fused image, the mean value of paths is required. Thus, the final mean value of the patches is selected as the maximum mean value of the source images' corresponding patches. In the final step, fused mean values and fused group sparse representation are combined to the reconstructed fused image.

3.2.3.4 NNSR Method

NSSR method [36] uses non-negative sparse representation. The overcomplete dictionary is learned from source images using an online dictionary learning algorithm [61]. Also, a traditional sparse representation which is orthogonal matching pursuit (OMP) [60] is used.

In the first step, source images are divided into overlapping patches and vector matrices are constructed from these patches. Vector matrices are used to train dictionaries for non-negative sparse representation. The non-negative sparse coding and traditional sparse coding (OMP) vectors of vector matrices are constructed. In the second step, they define some fusion rules for sparse coding. Salient features of both visible and infrared images are extracted using activity level. Target features (IT) and specific contour (IC) features of infrared image and texture features (VT) of the visible image are extracted from sparse coefficients. Then, a regional consistency rule is defined to combine traditional sparse coefficients using non-negative sparse representation-based features, called the fusion guide map and weights to help the combination of the visible and infrared coefficients. The regional consistency function defines which traditional sparse coefficients are used in a region. In some regions, infrared features are much more important than visible features vice versa. If a region consists of related features of both images, the weight map is used to combine the traditional sparse coefficients. The fused sparse coefficient is generated at the end of this step. In the last step, the fused image is reconstructed using a traditional sparse dictionary. Non-negative sparse representation is only used in extracting features and generating the fusion guide and weight maps. Overall schematic diagram of the proposed method is shown in Figure 3.17.

3.2.4 Sub-space Based Methods

Dimensionality reduction is commonly used to compress input high dimensional data to low dimensional spaces or sub-spaces while preserving data's variation as much as possible. This method eliminates redundant information without losing structural information. Moreover, low-dimensional space consumes less time and memory to

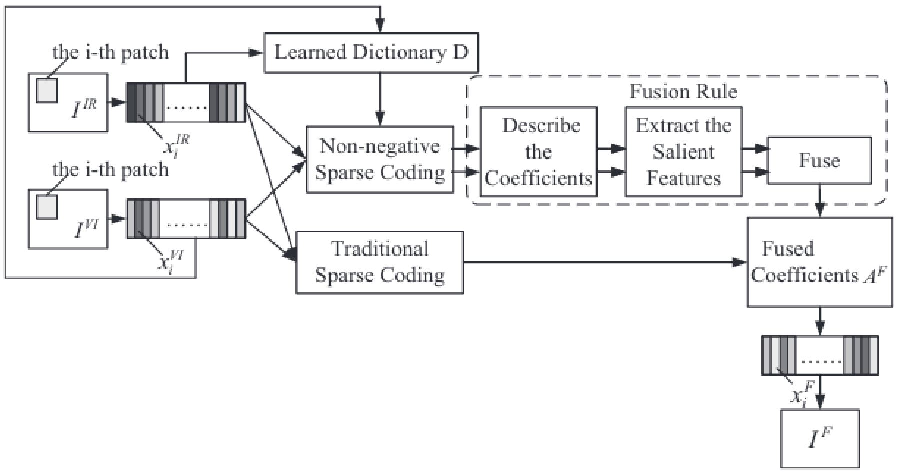


Figure 3.17: Overall architecture of NNSR method taken from [36]

process. PCA, ICA, and NMF, detailed explanations can be found in [62], are common methods for dimensionality reduction.

3.2.4.1 MDLatLRR [17]

MDLatLRR [17] use LatLRR [63] decomposition method to find low dimensional space representation of source images. To calculate low dimensional decomposition V_d, projection matrix L should be learned from training data patches. The LatLRR method defines $P(.)$ and $R(.)$ functions that extract features from the source image and reconstruct detail images from decomposition respectively. In the first step, multi-level decomposition maps are extracted for each source image separately. The low dimensional decomposition V_d^i, i corresponds to layer number, holds detail information called as detail layer. Reconstructed detail image subtracted from the input image to obtain first base layer I_b^1. The process continues iteratively, V_d^{i+1} is calculated using I_b^i, then I_b^{i+1} is calculated using V_d^{i+1} and I_b^i. In the second step, V_d^i layers for $i = 1, ..., r$ are combined using detail part fusion rule $FS(.)$. The weights are calculated by normalizing V_d^i layers by dividing each to the element-wise summation of all of them. In the final step, The low dimensional decomposition V_d^i layers are combined using calculated weights and each one is reconstructed and superposed. Then, the final base layers and constructed detail layer are superposed to generate the fused image. Figure 3.18 shows the overall architecture of the MDLatLRR.

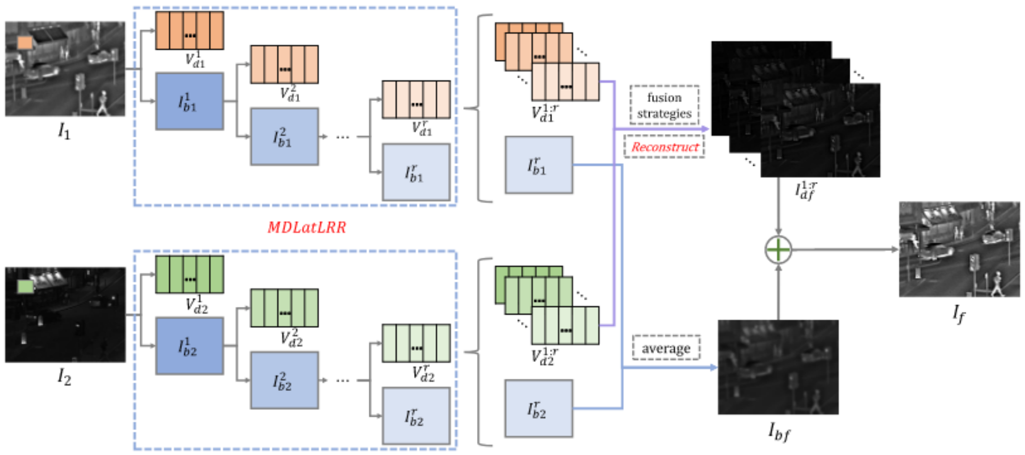


Figure 3.18: Proposed MDLatLRR Method taken from [17]

3.2.4.2 FPDE [18]

FPDE uses fourth-order differential equation [64] which gives name to this method to decompose image to detail and approximation layers. In the first step, FPDE is applied to extract approximation layers. Detail layers are computed by subtracting the approximation layers from source images. In the second step, two different fusion rule is defined for detail and approximation layers. For detail layers, detail layers are considered as a column vector and their principal components are computed. These computations are used as a weight to combine detail layers. For approximation layers, the average fusion rule is applied. In the final step, the element-wise summation is applied to fused layers to calculate the fused image. Block diagram of the approach is shown in Figure 3.19.

3.2.5 Hybrid Methods

All other image fusion methods have their advantages and disadvantages. Hybrid methods try to combine the other methods to improve fusion performance. Thus, some hybrid methods are explained in this chapter.

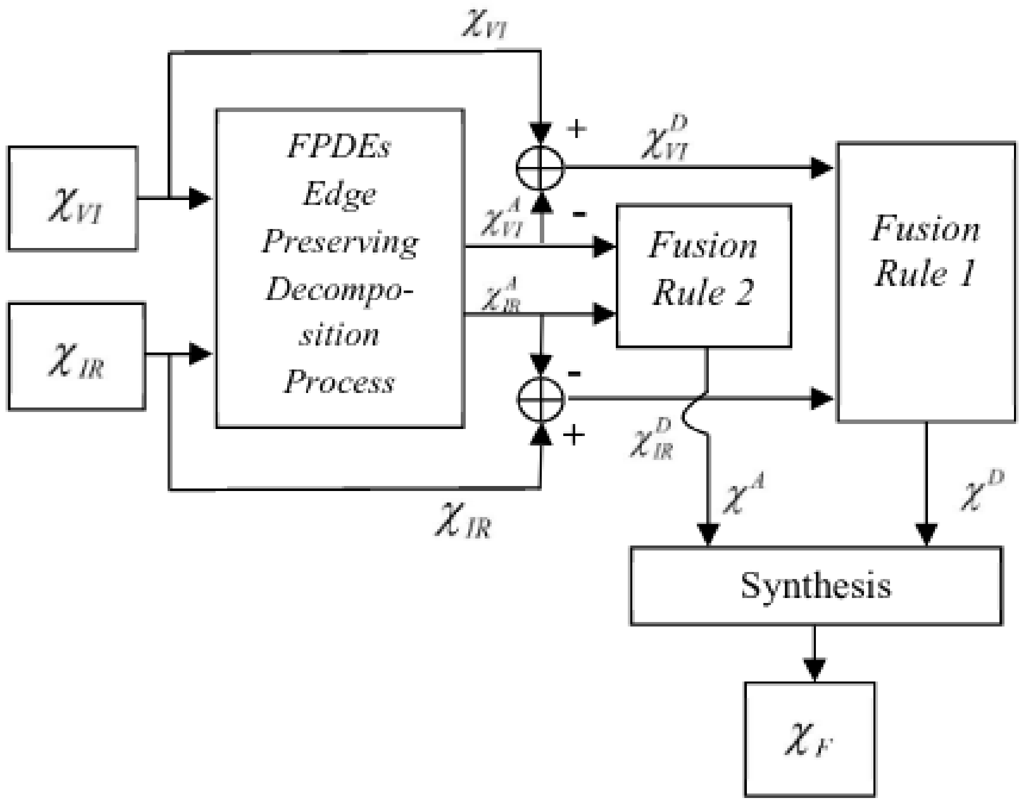
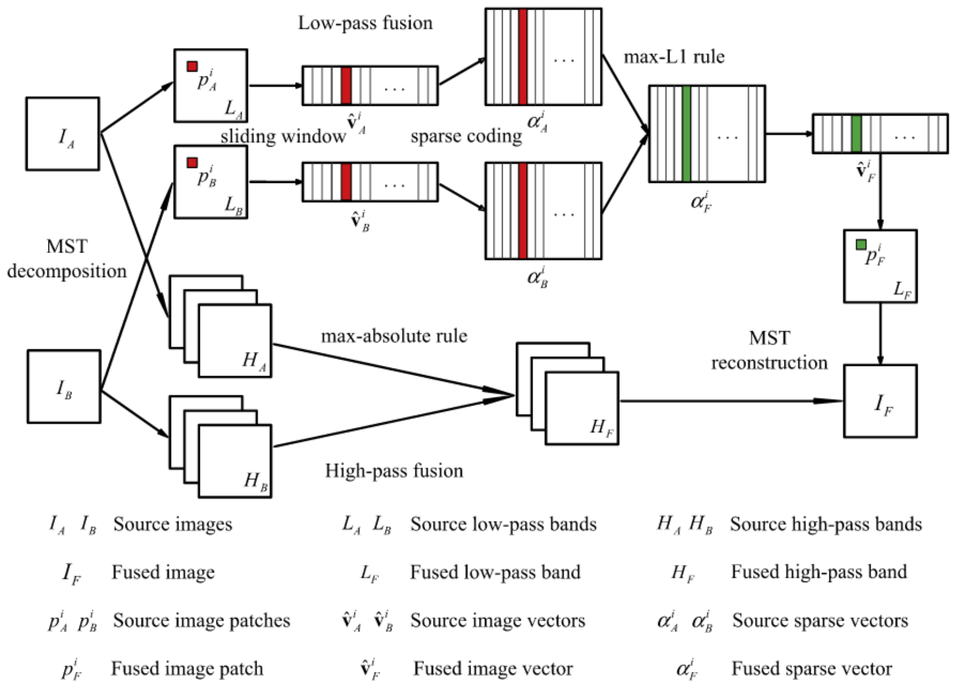


Figure 3.19: Block diagram of proposed FPDE method taken from [18]

3.2.5.1 MST_SR Methods [19]

All other image fusion methods have their advantages and disadvantages. Hybrid methods try to combine the other methods to improve fusion performance. Thus, some hybrid methods are explained in this chapter. MST_SR [19] tries to combine multi-scale transform (Chapter 3.2.1) and sparse representation (Chapter 3.2.3). Firstly, source images are decomposed into low-pass bands and high-pass bands. As the default option, they tried the Laplacian pyramid (LP) [50] which is a pyramid-based decomposition method. Other methods that are used to decompose images are ratio of low-pass pyramid (RP) [65], gradient pyramid (GP) [66], discrete wavelet transform (DWT) [67], stationary wavelet transform (SWT) [68], dual-tree complex wavelet transform (DTCWT) [69], curvelet transform (CVT) [70] and nonsubsampled contourlet transform (NSCT) [71]. In the second step, for each band, different fusion rules are defined. For low-pass bands, a sparse representation approach is applied to fuse low-pass bands. Low-pass response images are divided into patches. These patches are lexicographically ordered as a vector. All vectors are combined in a matrix in which each column corresponds to a vector. Then, sparse coefficients are found using given overcomplete dictionary. Using the maximum L1 value rule, sparse coefficients are combined. Then, fused coefficients are converted to fused vector matrix and each vector in the vector-matrix is reshaped into the original patch size. Overlapped patches are averaged over their accumulation times to construct fused low-pass images. For high-pass bands, the maximum absolute value fusion rule is applied to construct the fused high-pass image. In the last step, inverse multi-scale transform is applied to construct a fused image from fused high-pass and low-pass images. The schematic diagram of the proposed method is shown in Figure 3.20.



Low-pass fusion
max-L1 rule
sliding window
sparse coding
MST decomposition
max-absolute rule
MST reconstruction
High-pass fusion
I_A I_B Source images
L_A L_B Source low-pass bands
H_A H_B Source high-pass bands
I_F Fused image
L_F Fused low-pass band
H_F Fused high-pass band
p_A^i p_B^i Source image patches
$\hat{\mathbf{v}}_A^i$ $\hat{\mathbf{v}}_B^i$ Source image vectors
α_A^i α_B^i Source sparse vectors
p_F^i Fused image patch
$\hat{\mathbf{v}}_F^i$ Fused image vector
α_F^i Fused sparse vector

Figure 3.20: The schematic diagram of proposed general MST_SR method taken from [19]

3.2.5.2 VSMWLS Method [20]

VSMWLS method combine multi-scale transform (Chapter 3.2.1) and saliency-based (Chapter 3.2.2) approaches. In the first step, source images are decomposed into base and detail layer using rolling guidance filter [72] and Gaussian filter. In the second step, a different fusion rule is defined for detail and base layers. For base layers, in the beginning, a visual saliency map (VSM) [73] is computed. Then, base layers are combined using weighted averaging which weights are computed from VSMs. For detail layers, weighted least square optimization is applied. Weights used in optimization are calculated using the maximum absolute rule for each corresponding layer (infrared and visible layers pair). In the last step, fused detail layers and fused base layers are superposed to reconstruct the fused image. Overall algorithm steps are shown in Figure 3.21

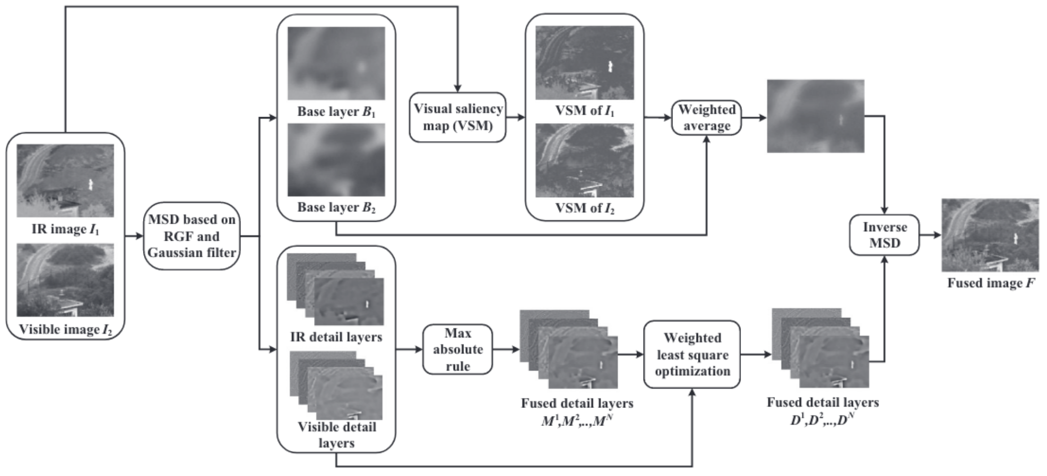


Figure 3.21: The schematic diagram of proposed VSMWLS method taken from [20]

3.2.6 Deep Learning Based Methods

The deep learning area inspired biological neuron systems to solve problems that are easy for human beings but hard to formulate mathematically. For instance, in the real-life, a human can easily classify, recognize, detect and track a cat in a video sequence. However, these problems are challenging for computers and with the help of mimicking the neural system of humans can present solutions closer or better from humans. Like all of us learn gradually in our whole life, the deep learning models are trained using samples. This approach gives promising results for computer vision tasks like image generation, segmentation, etc. Thus, image fusion applications also take advantage of deep learning. In the last years, these methods dominate the image fusion field in both quantitative and qualitative comparisons. Some of them present a solution only deep learning-based while some of them combine the previous methods such as multi-scale transform 3.2.1. In this work, we are mainly focusing on deep learning area-based solutions so, we collect all methods that use the deep learning approach in any part of the solution to this chapter.

3.2.6.1 CNN Method [21]

This method integrates deep learning to the fusion step in the multi-scale transform using the Laplacian pyramid. They find weight maps for each decomposition layer. It is used to fuse the decomposition layers that will be explained in the next paragraph. The remaining procedure is the same with multi-scale transform methods 3.2.1. The decomposed images are fused and inverse multi-scale transform is applied to reconstruct the fused image. Figure 3.22 shows the overall architecture of the proposed CNN method.

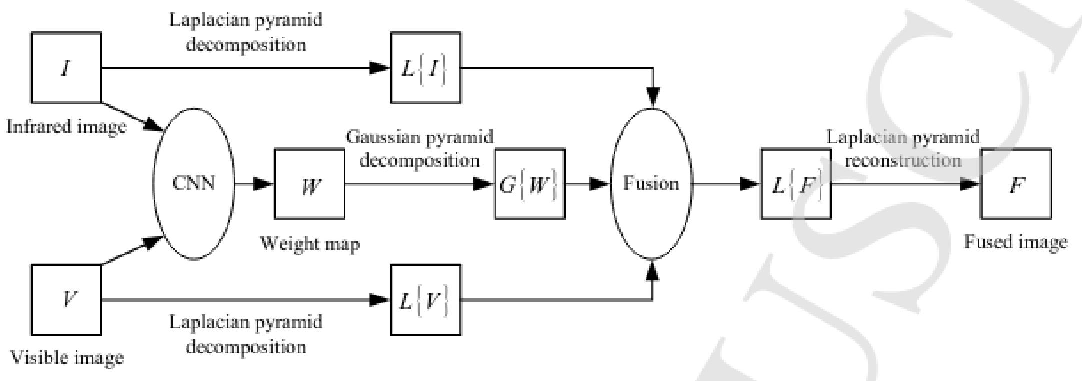

Laplacian pyramid decomposition
I
$L\{I\}$
Infrared image
CNN
W
Gaussian pyramid decomposition
$G\{W\}$
Fusion
$L\{F\}$
Laplacian pyramid reconstruction
F
Weight map
Fused image
V
$L\{V\}$
Laplacian pyramid decomposition
Visible image

Figure 3.22: The schematic diagram of proposed CNN method taken from [21]

The coefficient fusion step is defined as follows in CNN [21]. First of all, a Siamese network is used to generate a weight map. The designed network shown in Figure 3.23. A 16 by 16 image patches taken from source images are given to the network as inputs. The network includes convolutional and max-pooling layers. Fully connected layers are used to generate two outputs that give the probability of the source patch pairs at the corresponding location. If the probability of the infrared image patch is larger, the infrared patch has more valuable information than the corresponding visible image patch. They improve the Siamese network to compute weight maps in the whole image by replacing a fully connected layer with an equivalent convolution layer with an 8x8x512 kernel size. Thus, any arbitrary input patch size can be used as input. The whole network is trained using high-resolution image patches with different Gaussian blur ratios. The details about training can be found in [37]. Gaussian pyramid decomposition is applied to the output weight map. In the next part, they define a fusion rule. In this rule, they calculate the local energy map and similarity measure. The Sum of squares of the coefficients in a small region gives a local energy map. The similarity measure is computed as follows. Two times of sum of the local region in the decomposition map is divided by the sum of corresponding local energy map values. The similarity measure is between -1 and 1, a higher value means higher similarity. The fusion rule is a conditional function. In the first condition, if the similarity measure is higher than a defined threshold, coefficients are combined by weighted average using decomposed weight map. Otherwise, the 'choose-max' fusion rule is applied.

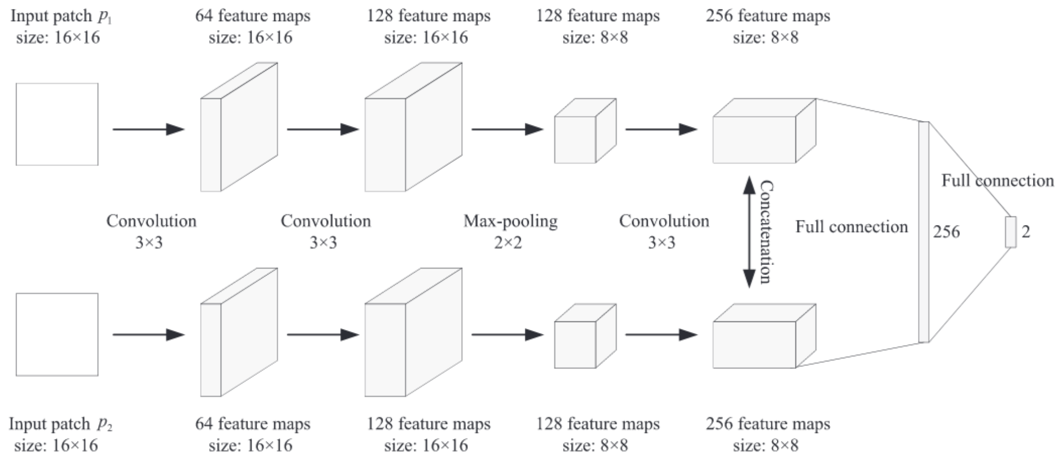

Input patch p_1 size: 16×16
64 feature maps size: 16×16
128 feature maps size: 16×16
128 feature maps size: 8×8
256 feature maps size: 8×8
Convolution 3×3
Convolution 3×3
Max-pooling 2×2
Convolution 3×3
Concatenation
Full connection
256
Full connection
2
Input patch p_2 size: 16×16
64 feature maps size: 16×16
128 feature maps size: 16×16
128 feature maps size: 8×8
256 feature maps size: 8×8

Figure 3.23: The proposed Siamese network to find weight map taken from [37]

3.2.6.2 DLF Method [22]

This method integrates deep learning to the fusion step in the multi-scale transform method GFF 3.2.2.2. They used a pre-trained VGG Network [4] to extract features from decomposed detail layers of the source images. On the outside of detail layer fusion, the proposed method is the same as GFF. In the first step, the source images are decomposed into base and detail layers. Secondly, base layers are combined with averaging strategy, and detail layers are combined using deep learning included method that will be explained later. In the last step, fused base and detail layers are superposed to reconstruct the fused image. The overall architecture of the proposed method is shown in Figure 3.24.

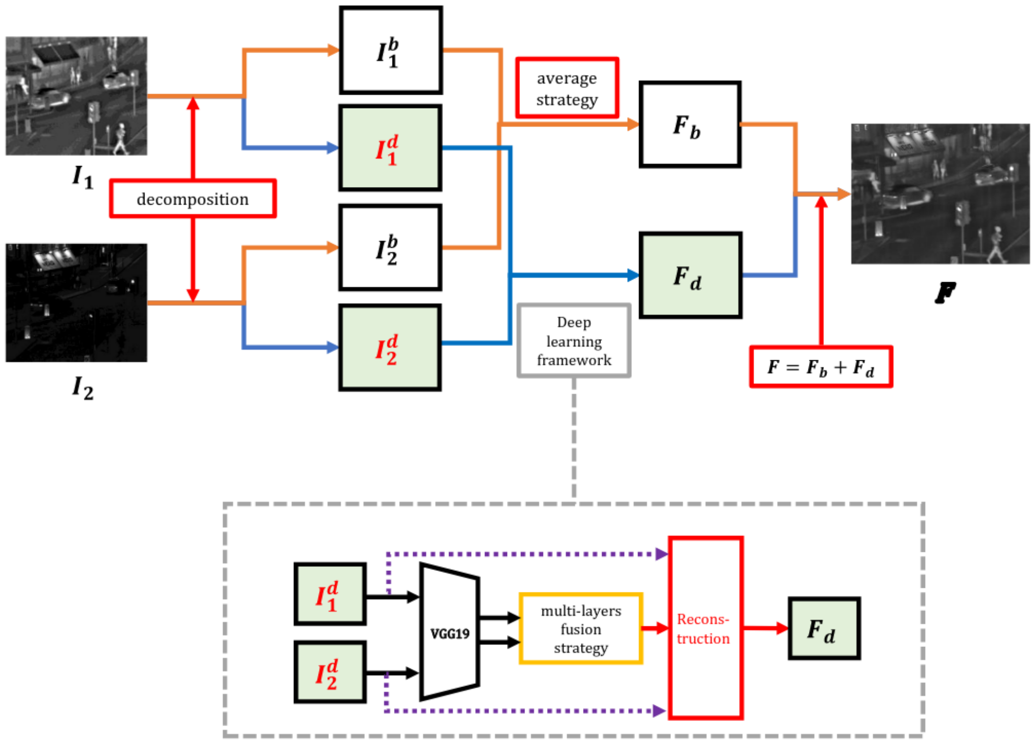


Figure 3.24: The schematic diagram of proposed DLF method taken from [22]

The detailed content fusion procedure includes the following steps, also shown in Figure 3.25. Firstly, the detail layers are given as input to the VGG network to extract deep features called feature maps. The feature maps are obtained from four different layers in the VGG network. Each feature layer is normalized using $l1$-norm across the channel dimension. The normalized feature maps called the activity level maps are smoothed using the block-based average operator. The initial weight maps are computed by the soft-max operator using corresponding activity level maps of the source images. Subsampling of inputs caused by the pooling layer in the VGG network decreases the feature map size. Thus, the initial weight maps should be upsampled to equalize dimensions with input detail layers. The detail layers are combined with the resized weight maps that generate four fused detail images. These four fused detail images are fused using maximum pixel value in each pixel location.

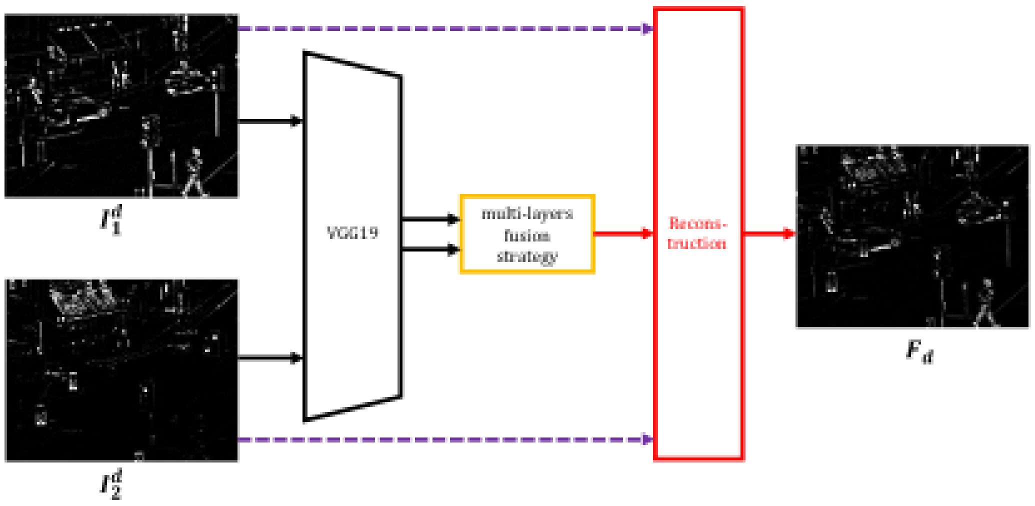


Figure 3.25: The procedure of detail content fusion taken from [22]

3.2.6.3 ResNet Method [23]

In ResNet method [23], deep residual architecture [5] is used to extract features. They design a simple architecture that fused image is reconstructed as the weighted average of the source images. The weight maps are found using fixed ResNet50 architecture [5] that is trained by ImageNet [74]. First of all, source images are given to ResNet50 as input, and different feature layer outputs are extracted called feature maps. For each feature map, zero-phase component analysis (ZCA) [75] is used to project features to the same space. Then, a block-based l_1-norm through the feature map channels is applied and the result is called the initial weight maps. The initial weight maps should be resized to match input source image size. Furthermore, the resized weight maps of the source images are combined using a soft-max operation. Thus, the sum of the corresponding pixel weights becomes one. Then, the weight maps are generated and the fused image is reconstructed by the weighting average method mentioned above. In Figure 3.26, the overall architecture of the proposed method is shown.

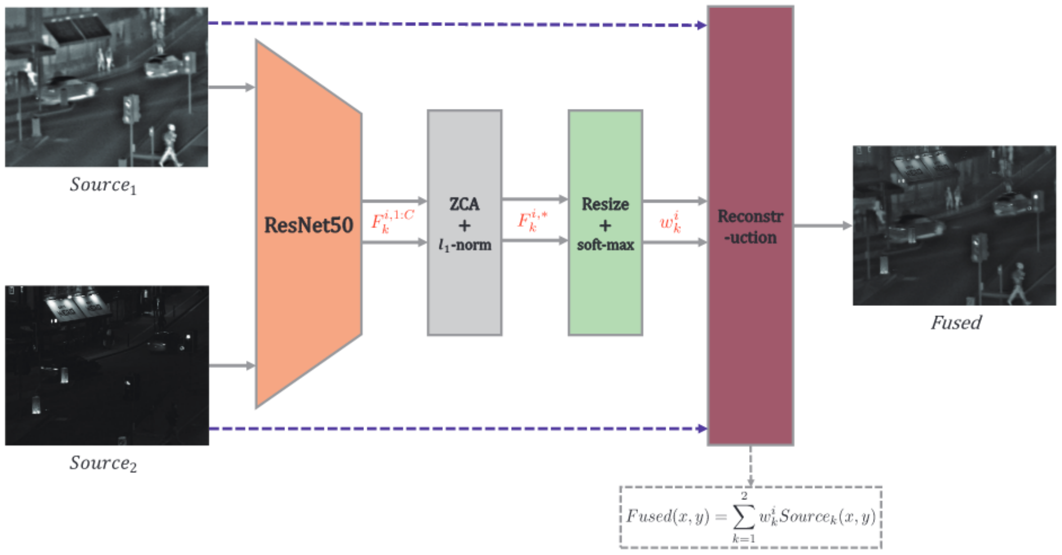


Figure 3.26: The procedure of the ResNet method taken from [23]

3.2.6.4 DenseFuse Method [24]

DenseFuse method [24] present an approach that uses only neural network to reconstruct fused image. They divide the architecture into three main parts: encoder, fusion layer, and decoder. In the encoder layer, image features are extracted using only convolutional layers. Skip connection strategy is used to preserve deep features as much as possible. The channel dimension increases throughout the encoder layer. In the decoder layer, four convolutional layers are used. There are no skip connections. The channel dimension decreases throughout the decoder layer and finally, the output becomes a single channel fused image. The training phase of the architecture does not include a fusion layer. The encoder and decoder layers are directly connected and a single input single output model is used. There is no absolute ground truth for image fusion applications so, they need to train networks in a single image base. The training image is processed through the encoder-decoder network. The generated output is compared with the input image. The loss function uses pixel loss L_p (l_2-norm between images) and structural similarity (SSIM) loss L_{ssim} (SSIM is computed between images and subtracted from one). These loss functions are combined with weight. The aim is to teach the network to extract features and construct an image from extracted features. In Figure 3.27, The training phase is shown. The trained network is ready to use in fusion applications. The source image is propagated through the encoder layer. The feature maps of the source images are fused in the fusion layer. Then, the fused image is reconstructed by the fused feature map.

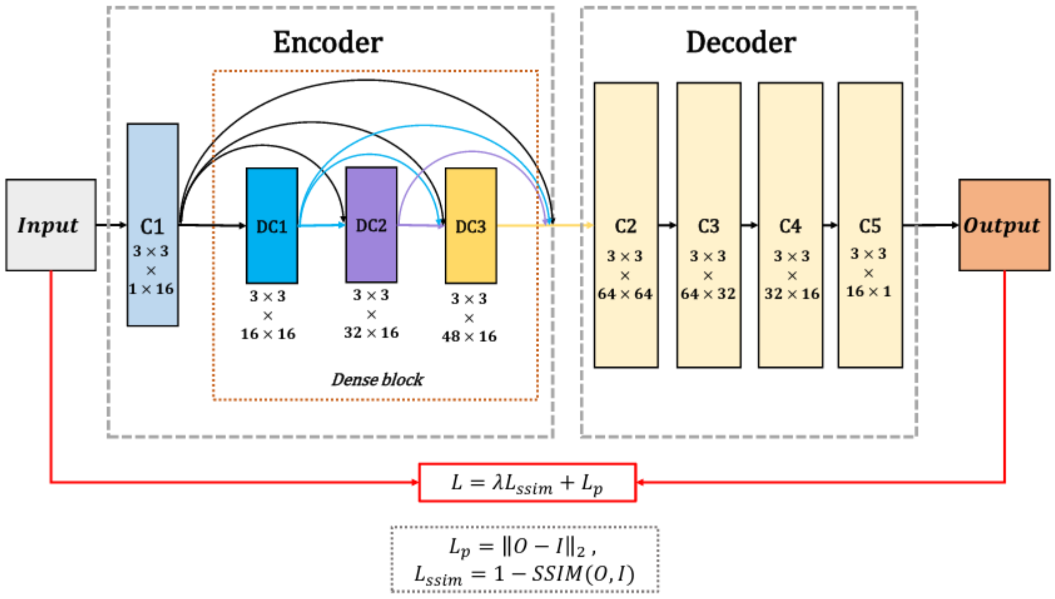


Figure 3.27: The training procedure of the proposed method taken from [24]

They introduce two fusion strategies. The first one is simply adding the feature maps. The second one is called l_1-norm strategy. the l_1-norm of each pixel through the channel is used to calculate the initial activity level map. The initial activity level maps smoothed using the block-based average operator. The soft-max is applied to the final activity level maps to find weight maps that are used to combine the feature maps of the source images. The overall architecture of the proposed method is shown in Figure 3.28.

3.2.6.5 DeepFuse Method [25]

DeepFuse method [25] introduces a completely neural network-based solution that can be trained end-to-end without any additional operation. The training and inference phases of the neural network are also the same. The whole architecture has three components: a feature extraction layer, a fusion layer, and a reconstruction layer. The first part of the network consists of two convolution layers and the weights of these

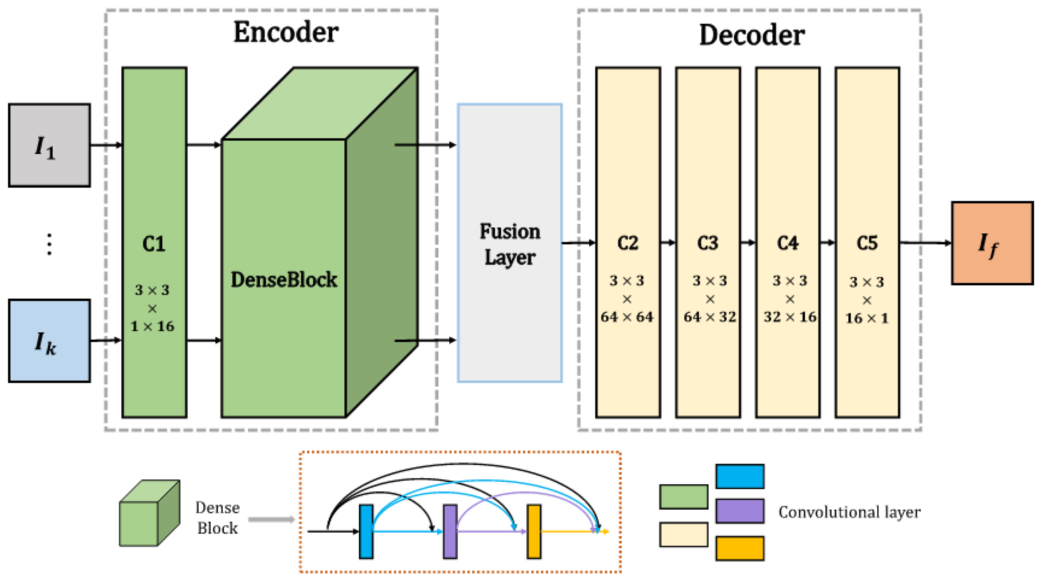


Figure 3.28: The overall architecture of the proposed method taken from [24]

layers are shared by all source images. Thus, the first part of the network is forced to learn common features from source images and they claim that this helps to fuse features better. The second component of the network is the fusion layer. The feature maps are superposed, also known as tensor addition. They tried several approaches to combine features such as concatenation, max, mean, and product operations. However, the best fusion result is obtained with addition operation among all other fusion approaches. In the last part, there is there convolutional layer used to reconstruct the fused image. The neural network architecture is shown in Figure 3.29. One of the main advantages of the proposed method is to include only convolutional layers that have a low number of parameters. To train the network, MEF-SSIM [76] image quality measure is used. MEF-SSIM compares source images with fused images in patches. The final score is the average of all patch scores. The final score is in the range [0,1]. Thus, the loss function is computed by subtracting MEF-SSIM from 1. It gives zero when the fused image takes the maximum score from MEF-SSIM.

In the overall architecture, input images are converted to YCbCr color space. The Y(luminance)-channels of the source images are fused using the DeepFuse method and other(chroma) channels are combined by weighting fusion. The fusion weights have a higher value for good color components. The schematic diagram of the overall

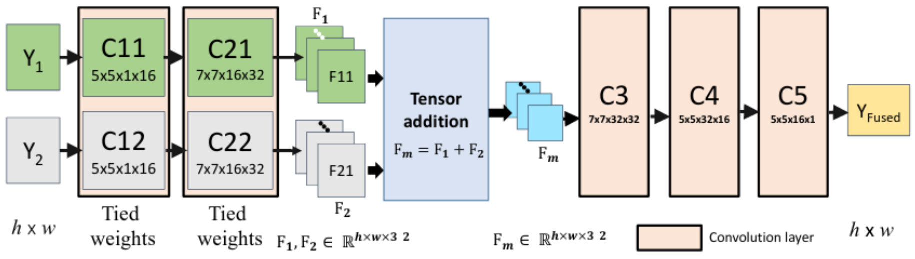


Figure 3.29: The proposed neural network to fuse input images taken from [24]

architecture is shown in Figure 3.30.

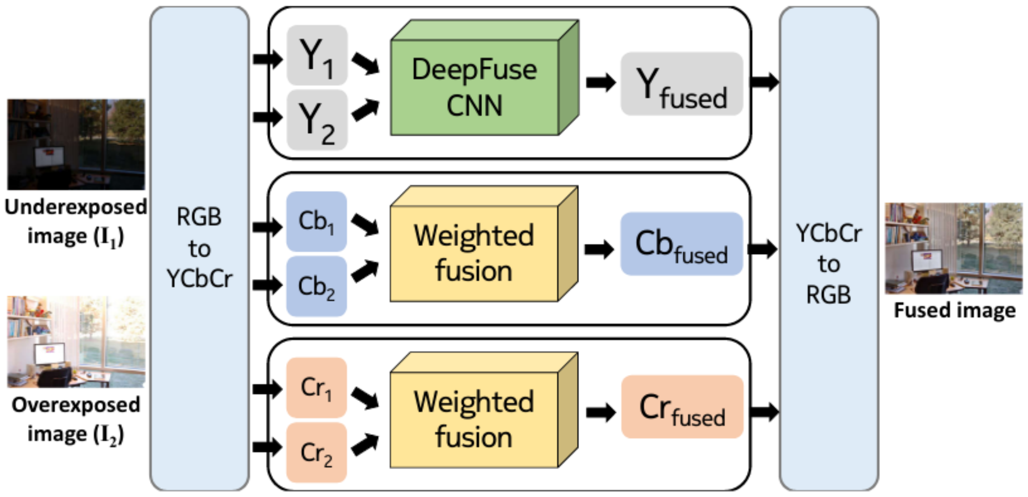


Figure 3.30: The overall architecture of the proposed method taken from [25]

3.2.6.6 FusionGAN Method [26]

FusionGAN method [26] uses a powerful deep learning-based method called generative adversarial network (GAN) which is invented by Goodfellow et al. [77]. GAN is based on a minimax algorithm in game theory. It consists of two parts: generator and discriminator. The discriminator part tries to distinguish which input is real or fake. The generator part tries to generate a fake output that fools the discriminator. Convolutional layers are used in both parts. The aim of the FusionGAN is to train a generator that can reconstruct the fused image from source images directly. To train the generator, they need a discriminator and a few training samples. Visible and infrared source images are concatenated through the channel layer. It is fed into the generator and The generator outputs fused image. The loss function of the generator has two terms: adversarial loss that tries to fool the discriminator to believe the generator output is real and, context loss which enforces fused image to have similar intensities as the infrared image and similar gradients as the visible image. To train discriminator, the fused image and visible image are fed into the discriminator and expected to predict fused image as fake and visible image as real. After the training process finishes, only a trained generator is used to reconstruct the fused image. The detailed architecture of the generator is shown in Figure 3.32. Also, training and test processes are shown in Figure 3.31.

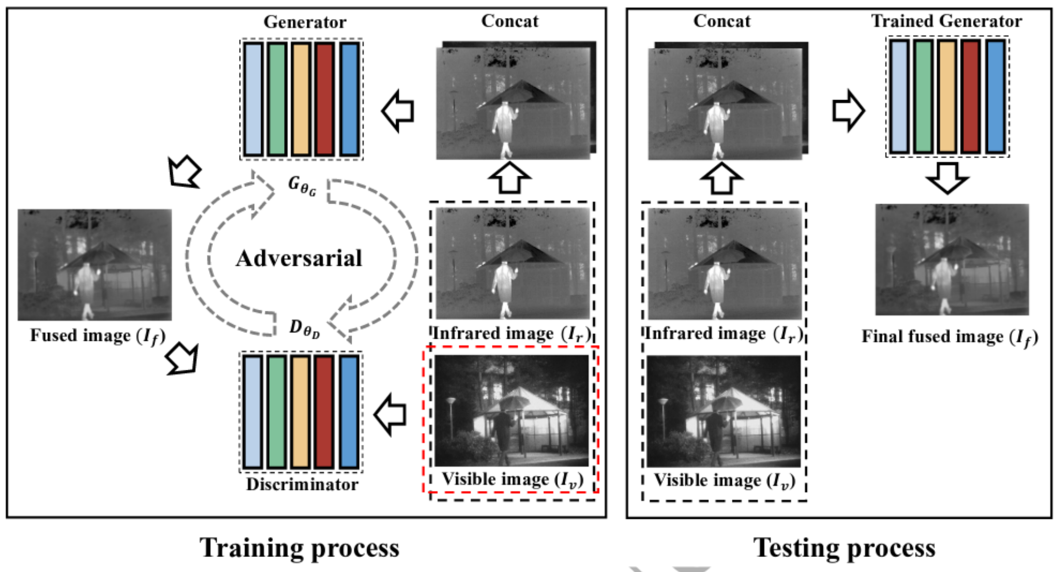


Figure 3.31: The overall architecture of the proposed method taken from [26]

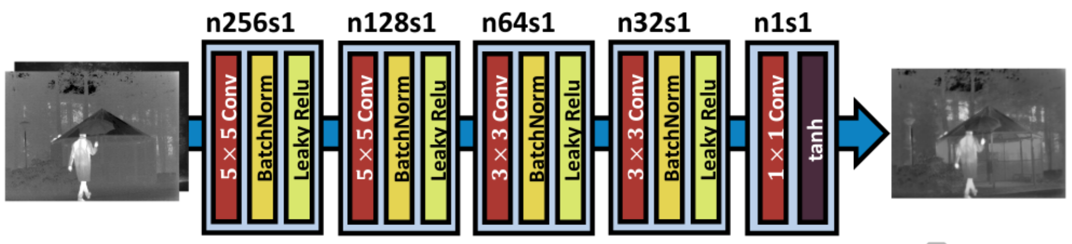


Figure 3.32: The overall architecture of the proposed method taken from [26]

3.2.6.7 RFN-Nest Method [27]

RFN-Nest method [27] differs from other deep learning-based methods in some aspects. The most important contribution of the method is that source image features are combined using a residual convolutional layer. The architecture consists of three parts: encoder layer, fusion layer, and decoder layer. The encoder layer extracts input image features while the input downsamples. Extracted source image features are fused by a fusion layer called RFN shown in Figure 3.34. Outputs of RFN layers are fed into the decoder layer to generate fused images demonstrated in Figure 3.35. The overall architecture is shown in Figure 3.33.

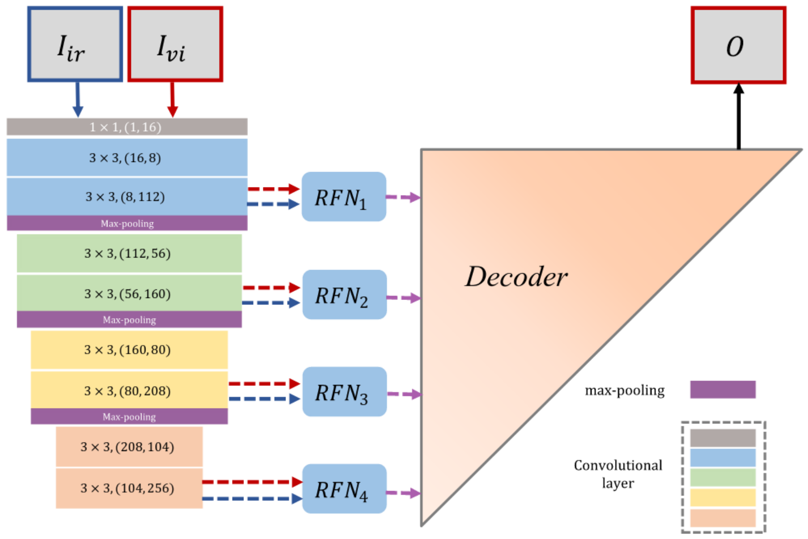


Figure 3.33: The overall architecture of the proposed method taken from [27]

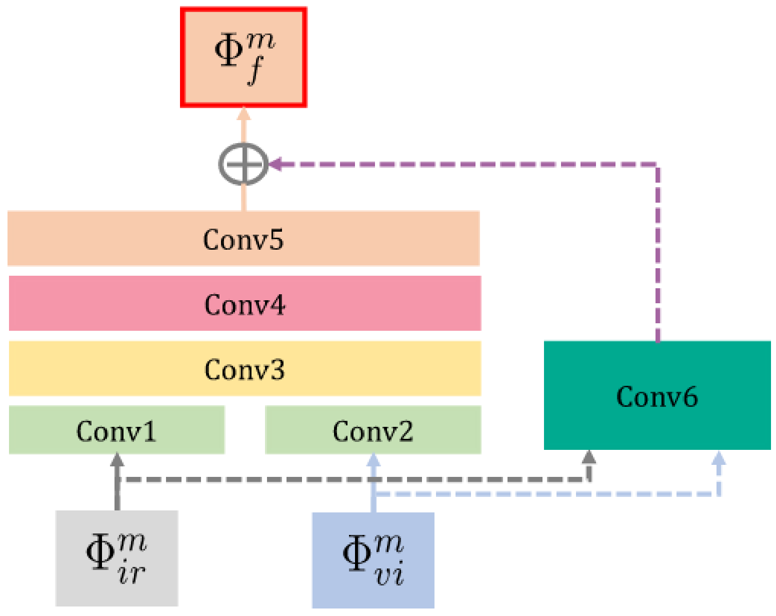


Figure 3.34: The residual fusion layer called RFN taken from [27]

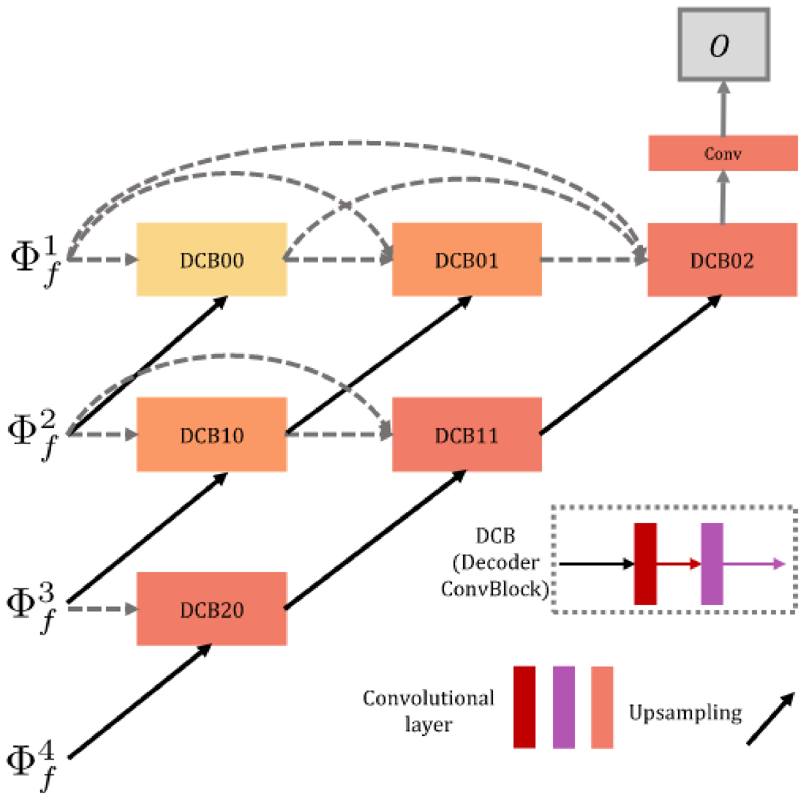


Figure 3.35: The decoder layer inspired from UNet++ [38]. Image is taken from [27]

The training process includes two stages. In the first stage, RFN layers are dropped and encoder layer features are directly connected to the decoder layer. The network is trained like auto-encoder networks. The network tries to reconstruct the input image. Thus, the encoder layer learns to extract good features while the decoder layer learns to reconstruct the image from features. They define a loss function that combines pixel loss and structural similarity loss. Pixel loss is square of the Frobenius norm and structural similarity loss is computed by subtracting SSIM measure [78] from 1. In the second stage of the training process, trained encoder and decoder layer weights are fixed. RFN layers are connected as the proposed method. With the infrared and visible image pairs, only RFN layers are trained. The loss function to train RFN consists of two parts. In the first part, detailed information from the visible image is considered and SSIM metric [78] between fused image and the visible image is used. In the second part, the output of RFN is compared with the weighted combination of the source image features i.e. input of the RFN. The Frobenius norm is calculated between them. The combination weights of the features are the hyper-parameter of the training process.

3.2.6.8 Dual-Branch Method [28]

Dual-Branch method [28] designs a encoder-decoder neural network. The encoder layer extracts source image features and the decoder layer reconstructs the fused image using the features extracted. The network has a fusion layer during inference only. The fusion layer does not include any learnable parameters, so it is thrown in the training phase. The overall architecture of the proposed method in the training phase is shown in Figure 3.36. The encoder layer is separated into two branches called detail branch and semantic branch after the initial feature map generated which is a blue box in the encoder layer shown in Figure 3.36. In the detailed branch, a dense block structure is used to extract good detail and texture information. The detail branch architecture is inspired from DenseFuse method's encoder layer 3.2.6.4. The semantic branch downsamples the input feature map to extract global features of the image. To concatenate branch outputs, the semantic branch upsamples the feature map. The concatenated feature map is fed into the decoder layer to reconstruct the fused image. To train a network, the loss function consists of four different loss functions. The first one called pixel loss finds mean square error (MSE) between input and output images. The second one called gradient loss is MSE between gradients which are computed by the Laplacian operator of the input and output images. The third one called color loss is calculated by l_2-norm between histograms of input and output images. The last one called perceptual loss compares feature maps of the input and output images. The feature maps are obtained by feeding the images into VGG [4] pre-trained network. All four-loss functions are combined with balancing hyper-parameters. It is important to note that in the training phase, only one input image is used.

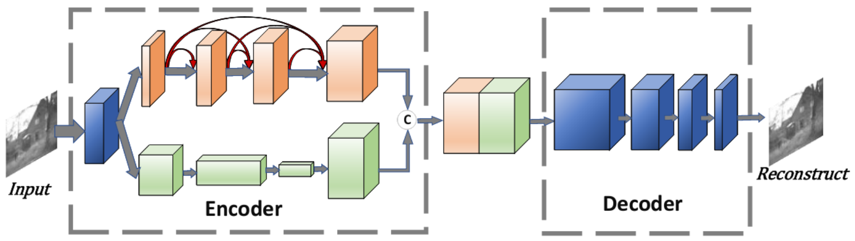


Figure 3.36: The overall architecture of the Dual-Branch method in the training phase. Image is taken from [28]

In the inference phase, visible and infrared images are fed into weighted shared Siamese encoder layers shown in Figure 3.37. The fusion layer combines the feature maps that will be explained later. Then, the fused image is reconstructed by a trained decoder layer using a fused feature map.

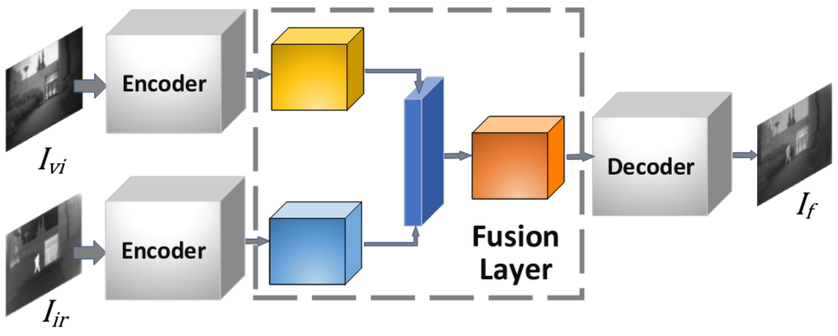


Figure 3.37: The architecture of the Dual-Branch method in the inference phase. Image is taken from [28]

They choose two fusion strategy which is addition and channel strategy. The addition strategy is simply adding corresponding pixel values. The channel strategy is a weighting addition that calculates weights from feature maps. In each channel, a global average is applied. Thus, each channel has an initial weight. Then, initial weight maps from different source images are normalized which the corresponding weight map summation for a channel becomes one. The normalized weights are multiplied with the feature maps and the results are summed element-wise. These two strategies are shown in Figure 3.38.

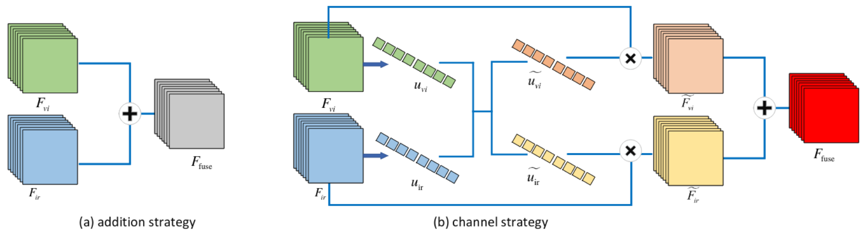


Figure 3.38: The fusion strategies of the Dual-Branch method. Image is taken from [28]

3.2.6.9 TFSNet Method [29]

In this method, they also proposed encoder-decoder architecture that encoder layer is responsible of extracting features of the source images while decoder layer reconstruct image from fused features. The difference of the method is adaptive weight allocation strategy and an end-to-end trainable network. The network consists of two identical encoder layers structures but their parameters are learned. In the encoder layer, the source image are fed into a shallower convolutional network, then a three layer cascade-connected dense blocks are used. The corresponding extracted four feature maps from source images are fused and given to decoder layer. In the fusion step and feature extraction, adaptive weight allocation strategy (AWA) is used. AWA is based on channel attentions. Briefly, each channel are averaged and the results constructs a one dimensional vector, Z. Then, a 1 by 1 convolution operation is performed on Z and the sigmoid function is applied to normalize weights. Normalized weights, Z', are multiplied with corresponding feature layer's channel to complete

AWA operation. In the fusion layer, respective feature maps comes from encoder layers are added after applying AWA to each of them. Then, the combined value is multiplied with a constant, α that called intensity stretch factor set as 2.7. In the decoder layer, the deconvolution layers are used to reconstruct fused image. The overall architecture of the TFSNet method is shown in Figure 3.39.

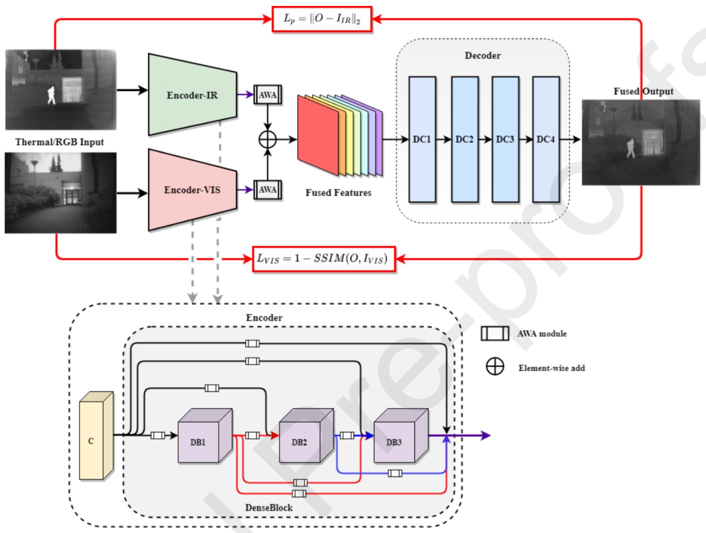


Figure 3.39: The overall architecture of TFSNet method. Image is taken from [29].

In the training phase, loss function composed of two metric. For infrared image, pixel-wise image loss (MSE) between input infrared image and fused image. For visible image, SSIM is used to preserve image structure similar to visible image. The VT5000 dataset [79] is used for training.

3.2.7 Other Methods

Other methods are generally inspired by new ideas and present new approaches. The methods that have success in other computer vision applications are tried to integrate into the image fusion area and it becomes important to improve the perspective to the problem. Even if the performance of these methods is poor, the approach leads to new ideas and improvements over successful methods.

3.2.7.1 GTF Method [30]

GTF method [30] tries to preserve thermal radiation information and detailed appearance information on both source images. The method estimates fused images directly using an objective function. The objective function is composed of two main parts. The first one defines the relationship between the source infrared image and the fused image. They stated that the fused image should have a similar pixel intensity distribution with the source infrared image. This constraint is measured by l^p norm ($p >= 1$). The second constraint defines the source visible image should have similar pixel gradients rather than pixel intensity to extract detailed appearance information. They present an optimization algorithm to minimize the objective function defined above. Total variation minimization is used for optimization. The fused image is directly obtained after applying optimization.

3.2.7.2 IFEVIP Method [31]

IFEVIP method [31] tries to extract useful infrared image features and add these features to the visible image to construct the fused image. The proposed method consists of three procedures. In the first part, infrared image background is extracted using quadtree decomposition [80] and Bézier interpolation [81]. The background image is smoothed using Gaussian blur operation and it is extracted from the source infrared image to extract infrared bright features. Secondly, the infrared bright features are refined using a source visible image. Even if background information is eliminated, there are some artifacts that are related to the background. Thus, the source infrared image is subtracted from the source visible image, and the negative values are equalized to zero. They define a hyper-parameter called suppression ratio that is multiplied by the previous result and subtracted from the infrared bright features. In this way, redundant background information is eliminated. In the last part, the final infrared bright features are added to the source visible image to reconstruct the fused image. In Figure 3.40, the proposed method is shown with a visual example.

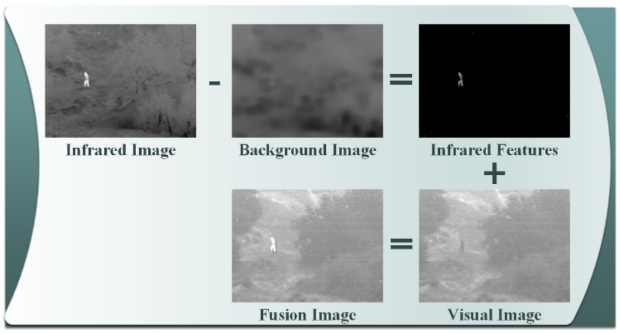


Figure 3.40: The schematic diagram of proposed IFEVIP method taken from [31]

3.3 Quality Measurements for Image Fusion Applications

In image fusion applications, there is no absolute ground truth to measure performance. Thus, the internal information of the source images and fused image should be compared to evaluate the performance of the methods. Furthermore, subjective evaluation is required because it bases on the human visual system. Humans compare several criteria, such as image details, contrast and brightness, object completeness, image distortions, and natural-looking of the scene. These qualitative and quantitative comparisons are used to evaluation of the performances of the methods. However, all evaluation metrics have good sides and drawbacks. Thus, the comparisons should be included as many metrics as possible for the robust evaluation. In the following parts, commonly used evaluation metrics and their pros and cons will be reviewed.

3.3.1 Entropy (EN)

Entropy (EN) is used in information theory to measure the amount of information contained in the signal. In [82], they introduced a procedure for the assessment of fused image quality. They defined EN as Eqn. 3.1 where L denotes the number of gray levels in the histogram of the image and p_l is the corresponding normalized value of the histogram at gray level l. The larger value of EN means more information contained in the image. However, noise can affect greatly the result.

$$EN = -\sum_{l=0}^{L-1} p_l \log_2 p_l \tag{3.1}$$

3.3.2 Mutual Information (MI)

Mutual Information (MI) measures mutual dependence between two random variables. To be more specific, MI quantifies the shared amount of information on two images. In [83], the amount of information transferred to the fused image from the source image is measured. The MI between two random variables is calculated by the Kullback-Leibler measure defined in Eqn. 3.2 where $p_X(x)$ and $p_F(f)$ denote the marginal histogram of source image X and fused image F respectively. $p_{X,F}(x, f)$ defines the joint histogram of the source image X and fused image F.

$$MI_{X,F} = \sum_{x,f} p_{X,F}(x, f) \log \frac{p_{X,F}(x, f)}{p_X(x)p_F(f)} \tag{3.2}$$

MI is calculated separately with source images A and B. The results are summed to obtain the final MI score as given in Eqn. 3.3. The higher MI score indicates that more information is transferred to the fused image from the source images.

$$MI = MI_{A,F} + MI_{B,F} \tag{3.3}$$

3.3.3 Feature Mutual Information (FMI)

Feature mutual information (FMI) is introduced by [84] and they considered to calculate MI metric between feature maps of the source images and the fused image. The feature map can include edges, details, and contrast of the images. In the computation of the metric, Eqn. 3.2 is used. The final FMI score is computed as Eqn. 3.4 where $\hat{A}$, $\hat{B}$ and $\hat{F}$ denote the feature maps of the infrared, visible and fused image respectively.

The higher FMI score is better as indicated in MI score 3.3.2.

$$FMI = MI_{\hat{A},\hat{F}} + MI_{\hat{B},\hat{F}} \tag{3.4}$$

3.3.4 Structural Similarity Index Measure (SSIM)

Image distortion and structural losses in the image take the attention of the human visual system. Thus, measuring image distortion and structural losses can give intuition about the quality of the image. In [78], they proposed a metric called SSIM to model loss and distortion. SSIM composes of three components: luminance (l), structure (s) and contrast (c). The product of these components gives the final SSIM score. The score is in the range [0, 1]. The score of one means the compared images are identical. A higher SSIM score means more similarity in the input images. SSIM score is calculated as follows:

$$SSIM_{X,F} = \sum_{x,f} l(x,f) \cdot c(x,f) \cdot s(x,f) \tag{3.5}$$

where $l(x,f)$, $c(x,f)$ and $s(x,f)$ denote luminance, contrast and structure components and computed as given in Eqns. 3.6, 3.7 and 3.8 respectively. x and f denotes the patches of the source image and fused image. μ_x and μ_f are the mean value of the patches. σ_x, σ_f and σ_{xf} are the variance of the source image patch x, variance of the fused image patch f and covariance of source and image patches. The $c1$, $c2$ and $c3$ is used to make metric stable. Final SSIM score is computed by summing SSIM scores between fused and source images as stated in Eqn. 3.9.

$$l(x,f) = \frac{2\mu_x\mu_f + c1}{\mu_x^2 + \mu_f^2 + c1} \tag{3.6}$$

$$c(x,f) = \frac{2\sigma_x\sigma_f + c2}{\sigma_x^2 + \sigma_f^2 + c2} \tag{3.7}$$

$$s(x,f) = \frac{\sigma_{xf} + c3}{\sigma_x \sigma_f + c3} \tag{3.8}$$

$$SSIM = SSIM_{A,F} + SSIM_{B,F} \tag{3.9}$$

3.3.5 Standard Deviation (SD)

Standard deviation (SD) measures the amount of variation of data. It is commonly used in statistical concepts. In the image, SD gives the contrast ratio of the image. It is mathematically defined as follow:

$$SD = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(F(i,j) - \mu)^2} \tag{3.10}$$

where μ is the mean value of the whole image, $F(i,j)$ is the corresponding pixel in the image The idea behind this metric is that high contrast attracts human attention. Large SD means more contrast image and the only fused image is used to compute the SD metric.

3.3.6 Spatial Frequency (SF)

Spatial Frequency (SF) is based on horizontal and vertical image gradients which are also called spatial row frequency (RF) and spatial column frequency (CF). It measures the gradient distribution of an image. A higher SF score means the image has more details and texture. RF and CF are computed as follows:

$$RF = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(F(i,j) - F(i,j-1))^2} \tag{3.11}$$

$$CF = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(F(i,j) - F(i-1,j))^2} \tag{3.12}$$

The SF score is l_2 norm of the RF and CF shown in the Eqn. 3.13.

$$SF = \sqrt{RF^2 + CF^2} \tag{3.13}$$

3.3.7 Average Gradient (AG)

Average gradient (AG) uses gradient information to define a metric. Gradient represents details and texture information so, it can be used for quality assessment of an image. In AG score calculation, the only fused image is used.

$$AG = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\sqrt{\frac{\nabla F_x^2(i,j) + \nabla F_y^2(i,j)}{2}} \tag{3.14}$$

where $\nabla F_x^2(i,j) = F(i,j) - F(i+1,j)$ and $\nabla F_y^2(i,j) = F(i,j) - F(i,j+1)$. The larger AG score means better gradient information exists in the image.

3.3.8 Mean Squared Error (MSE)

Mean squared error (MSE) measures the dissimilarity of the images at the pixel level. Most basic evaluation metric to find the distance of the images. MSE is calculated separately for source images with the fused image. The MSE score is defined as follows:

$$MSE = \frac{MSE_{AF} + MSE_{BF}}{2} \tag{3.15}$$

where $MSE_{XF} = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(X(i,j) - F(i,j))^2$. The higher MSE means the dissimilarity between source images and fused image is high. Thus, a good MSE score should be small.

3.3.9 L1 error

L1 error measures the dissimilarity of the images at the pixel level. Most basic evaluation metric to find the distance of the images. L1 is calculated separately for source images with the fused image. The L1 score is defined as follows:

$$L1 = \frac{L1_{AF} + L1_{BF}}{2} \tag{3.16}$$

where $L1_{XF} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |(X(i,j) - F(i,j))|$. The higher L1 means the dissimilarity between source images and fused image is high. Thus, a good L1 score should be small.

3.3.10 Root Mean Squared Error (RMSE)

Root mean squared error is similar to MSE 3.3.8. When calculating MSE_{XF}, the squared of the result is calculated to find $RMSE_{XF}$. Then, Eqn. 3.17 is used to compute overall RMSE score. As stated in MSE, small scores are desired.

$$RMSE = \frac{RMSE_{AF} + RMSE_{BF}}{2} \tag{3.17}$$

3.3.11 Peak Signal To Noise Ratio (PSNR)

Peak Signal To Noise Ratio (PSNR) uses MSE 3.3.8 between source images and fused image and peak value of the fused image. The larger PSNR value means the fused image is closer to the source images and has less distortion. PSNR is computed as shown in Eqn. 3.18 where r denotes the peak value of the fused image.

$$PSNR = 10 \log_{10} \frac{r^2}{MSE} \tag{3.18}$$

3.3.12 Correlation Coefficient (CC)

The correlation coefficient (CC) measures how strong the relationship has in the source images and fused images. It is in the range [-1, 1]. The zero CC score means the images do not have any relationship. A negative valued relationship shows correlation is opposite. The CC score is calculated using Eqn. 3.20 where r_{AF} and r_{BF} are calculated using Eqn. 3.19.

$$r_{XF} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(X(i,j)-\mu_X)(F(i,j)-\mu_F)}{\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(X(i,j)-\mu_X)^2\big(\sum_{i=1}^{M}\sum_{j=1}^{N}(F(i,j)-\mu_F)^2\big)}} \tag{3.19}$$

$$CC = \frac{r_{AF}+r_{BF}}{2} \tag{3.20}$$

3.3.13 Edge Intensity (EI)

The edge intensity (EI) metric simply computes the edge strength of the fused image. If the fused image has a higher EI score, it is assumed that the image is more clear and has a higher quality. The Sobel edge operator is used to extract vertical and horizontal gradients. In Eqn. 3.21, s_x and s_y represent the horizontal and vertical edges calculated using Sobel edge operator respectively.

$$EI = \sqrt{s_x^2 + s_y^2} \tag{3.21}$$

3.3.14 Q, $\mathrm{Q_W}$, and $\mathrm{Q_E}$

Universal quality index (Q) is a special case of the SSIM metric 3.3.4 that $c1 = c2 = 0$. In the [85], the sliding window approach is used to overall image quality rather than comparing whole images as given in Eqn. 3.22. Thus, the mean score of all regions is calculated. It is more appropriate because the image is non-stationary and space-variant.

$$Q(a,b) = \frac{1}{W}\sum_{w\epsilon W} Q(a,b|w) \tag{3.22}$$

In [86], a fusion quality index (Q_0) is introduced using Q metric. They use local weights $\lambda(w)$ to combine Q scores in each image window. Local weights $\lambda(w)$ are calculated using local salience where these are selected as variances of the local image patches. Q_0 is computed as follows:

$$Q_0(a,b,f) = \frac{1}{|W|}\sum_{w\epsilon W}(\lambda(w)Q(a,f|w)) + (1-\lambda(w))Q(b,f|w) \tag{3.23}$$

where the local weights $\lambda(w)$ are computed in Eqn. 3.24 using $s(a|w), s(b|w)$ saliency metrics of the input image a and b respectively.

$$\lambda(w) = \frac{s(a|w)}{s(a|w)+s(b|w)} \tag{3.24}$$

Furthermore, each window in the image is treated equally but, it is not correct in the image. Some regions have more important information than others. Thus, they present a local region weight $c(w)$ which is computed as follows:

$$c(w) = \frac{C(w)}{\sum_{w'\epsilon W} C(w')} \tag{3.25}$$

where $C(w) = max(s(a|w), s(b|w))$. Final weighted fusion quality index (Q_W) becomes,

$$Q_W(a,b,f) = \frac{1}{|W|}\sum_{w\epsilon W} c(w) * ((\lambda(w)Q(a,f|w)) + (1-\lambda(w))Q(b,f|w)) \tag{3.26}$$

They proposed a final modification that considers the edge response of the images. The inputs of the Q_W score can be the edge response of the images a, b, and f. The edge-dependent Q_W score is combined with the original Q_W score to construct edge-dependent fusion quality index Q_E which is given in Eqn. 3.27.

$$Q_E(a,b,f) = Q_W(a,b,f)\cdot Q_W(a',b',f')^{\alpha} \tag{3.27}$$

3.3.15 $\mathbf{Q^{AB/F}}$, $\mathbf{L^{AB/F}}$, $\mathbf{N^{AB/F}}$

$Q_{AB/F}$ [87] is a fusion quality metric that uses edge information that is associated with Human Visual System. First of all, horizontal s_x and vertical s_y edges are computed using the Sobel edge operator. In each pixel location, edge strength $g(n,m)$ and orientation $\alpha(n,m)$ are computed as follows:

$$g(n,m) = \sqrt{s_x(n,m)^2 + s_y(n,m)^2} \tag{3.28}$$

$$\alpha(n,m) = tan^{-1}(\frac{s_y(n,m)}{s_x(n,m)}) \tag{3.29}$$

Relative edge strength $G^{XF}(n,m)$ and orientation $A^{XF}(n,m)$ is calculated with respect to the fused image as given in Eqns. 3.30 and3.31 which X refers to input images.

$$G^{XF}(n,m) = \begin{cases} \frac{g_F(n,m)}{g_X(n,m)} & \text{if } g_X(n,m) > g_F(n,m) \\ \frac{g_X(n,m)}{g_F(n,m)} & \text{otherwise} \end{cases} \tag{3.30}$$

$$A^{XF}(n,m) = \frac{||\alpha_X(n,m) - \alpha_F(n,m)| - \pi/2|}{\pi/2} \tag{3.31}$$

These relative maps and sigmoid function are used to derive $Q_g^{XF}(n,m)$ and $Q_\alpha^{XF}(n,m)$ which are perceptual loss information in terms of edge strength and orientation. Edge information preservation value is defined as follows:

$$Q^{XF}(n,m) = Q_g^{XF}(n,m) \cdot Q_\alpha^{XF}(n,m) \tag{3.32}$$

where $0 <= Q^{XF}(n,m) <= 1$ that is higher is better preservation of the edge information on the fused image. Final fusion quality score is calculated using Eqn. 3.33 where $w^A(n,m) = [g_A(n,m)]^L$ and $w^B(n,m) = [g_B(n,m)]^L$, L is constant. Also, $0 <= Q^{AB/F} <= 1$ which $Q^{AB/F} = 1$ means perfect preservation of the edge information in the fused image F from input images A and B.

$$Q^{AB/F} = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M} Q^{AF}(n,m)w^{A}(n,m) + Q^{BF}(n,m)w^{B}(n,m)}{\sum_{n=1}^{N}\sum_{m=1}^{M} w^{A}(n,m) + w^{B}(n,m)} \tag{3.33}$$

With the $Q^{AB/F}$ that is the information transferred from the source images, $L^{AB/F}$ and $N^{AB/F}$ metrics can be also calculated which are total loss of information and noise or artifacts added to fused image respectively. The information loss metric, $L^{AB/F}$, is calculated using Eqn. 3.34. The lower score is better for the fusion quality. Also, the score that measures additive noise or artifacts to the fused image is calculated using Eqns. 3.35 and 3.36. This score also should be as low as possible and in the theory, the sum of three scores should be equal to 1: $Q^{AB/F} + L^{AB/F} + N^{AB/F} = 1$.

$$L^{AB/F} = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M}(1 - Q^{AF}(n,m))w^{A}(n,m) + (1 - Q^{BF}(n,m))w^{B}(n,m)}{\sum_{n=1}^{N}\sum_{m=1}^{M} w^{A}(n,m) + w^{B}(n,m)} \tag{3.34}$$

$$N_{n,m} = \begin{cases} 2 - Q^{AF}(n,m) - Q^{BF}(n,m) & \text{, if } g_F(n,m) > (g_A(n,m) \,\&\, g_B(n,m)) \\ 0 & \text{, otherwise} \end{cases} \tag{3.35}$$

$$N^{AB/F} = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M} N_{n,m}(w^{A}(n,m) + w^{B}(n,m))}{\sum_{n=1}^{N}\sum_{m=1}^{M} w^{A}(n,m) + w^{B}(n,m)} \tag{3.36}$$

3.3.16 Q$_{\text{CB}}$

Q_{CB} [88] is a fusion quality metric that is inspired by the human visual system. It includes five steps that contrast sensitivity filtering, the local contrast computation, the contrast preservation calculation, the saliency map generation, and the global quality measure computation. In the first step, Fourier transform is applied to the source and fused images. Let $\mathcal{I}_A(u,v)$ is the Fourier transform of an image where A can be the source or fused images. The results are multiplied by a $S(r)$ function that $r = \sqrt{u^2 + v^2}$. Then, inverse Fourier transform is applied to find filtered images. In the second step, local contrast computation is calculated. The result is called C_A. Detailed calculations can be found in [88]. After C_A is calculated, the masked result C'_A is computed as seen in Eqn. 3.37. The masked results are used in contrast preservation calculation and saliency map generation. The third step is defined in Eqn. 3.38.

$$C'_A = \frac{k(C_A)^p}{k(C_A)^q + Z} \tag{3.37}$$

$$Q_{AF} = \begin{cases} \frac{C'_A(x,y)}{C'_F(x,y)} & \text{, if } C'_A(x,y) < C'_F(x,y) \\ \frac{C'_F(x,y)}{C'_A(x,y)} & \text{, otherwise} \end{cases} \tag{3.38}$$

In the fourth step, saliency maps are generated to combine contrast preservation values in the last step as shown in Eqn. 3.39. In the last step, the contrast preservation values are combined using saliency maps as shown in Eqn. 3.40. Higher scores are better.

$$\lambda_A(x,y) = \frac{C'^2_A}{C'^2_A + C'^2_B} \tag{3.39}$$

$$Q_{CB} = \lambda_A(x,y) * Q_{AF} + \lambda_B(x,y) * Q_{BF} \tag{3.40}$$

3.3.17 Q_{CV}

Q_{CV} [89] is also inspired from human visual system similar to Q_{CB} 3.3.16. However, the approach is quite different than Q_{CB} and the lower scores are better for fusion quality. The approach consists of five steps that are extracting edge information, partitioning into non-overlapping local regions, local region saliency, similarity measure of the local regions, and global quality measure computation. In the first step, the edge responses of each image are computed. Lets say edge responses as $G_X(i,j)$

where X can be source images A, B, and fusion image F. In the next step, images are divided into non-overlapping local regions W_i. Window size is the hyper-parameter of this quality metric. Then, the local saliency of each window is calculated. The saliency of the local region $\lambda_A(X^W)$ is the basic summation of the edge intensities of the region. In the fourth step, local region differences f_k^W are found that the fused image is extracted from source images. The similarity of local regions is measured by the mean squared value of contrast sensitivity function (CSF) filtered images $\hat{f}_k^W$. In the last step, the global quality measure is calculated by a weighted summation of the similarity of the local regions. $\lambda_A(X^W)$ is used as weights of each region to compute the final score. Selecting CSF filter and window size are the most important parameters to be decided.

3.3.18 $\mathbf{Q_Y}$

Q_Y [90] is the fusion quality metric that is calculated based on SSIM [78]. As mentioned in the SSIM metric 3.3.4, the images are divided into square regions and each corresponding region in the input images are compared in structure, luminance, and contrast components. Luminance comparison is insignificant for local patches and it is discarded from the calculation. The SSIM result for each region is called a ssim-map that is calculated using the formula given in Eqn. 3.41 which uses Eqns. 3.7 and 3.8. First of all, the following ssim-maps are calculated: ssim-map between infrared and visible image (ssim-map-xy), ssim-map between infrared and fused image (ssim-map-xf), and ssim-map between visible and fused image (ssim-map-yf). These metrics are combined to construct the final Q_Y score as given in Eqn. 3.43. To combine the ssim-maps, the lambda λ weights are calculated as seen in Eqn. 3.42. The λ aims to specify the relative importance of the regions. In Eqn 3.43, r shows the region number and total region number is R. After summing all regions' scores, it is divided to R to find the average score.

$$ssim - map - xy = c(x, y) * s(x, y) \tag{3.41}$$

$$\begin{aligned}\lambda_x &= \frac{\sigma_x}{\sigma_x + \sigma_y} \\ \lambda_y &= 1 - \lambda_x\end{aligned} \tag{3.42}$$

$$Q_Y = \frac{1}{R}\sum_{r}^{R} \lambda_x(r) * (ssim - map - xf)(r) + \lambda_y(r) * (ssim - map - yf)(r) \tag{3.43}$$

3.4 Applications

Image fusion is used by the various application. Both infrared and visible spectrum images have strengths and weaknesses in some imaging conditions. The most basic one is that visible images can have color information while infrared images can see objects through smoke or fog for example.

There is two main application area for the image fusion: surveillance and computer vision tasks. In surveillance, clear, robust, high contrast, and bright images that express the details and object features properly are needed. Rather than keeping track of multiple images on two screens, it is easier to keep track of one screen that combines images. It makes objects easily identified, more clear and sharp. The fused image has a high spatial resolution that makes these possible. In computer vision applications such as object detection, tracking, and recognition, the features extracted from image for an object is used to complete these tasks. Some of the applications use multiple spectrum images without fusing images and that increases the processing time of the application significantly. Thus, a fused image that reconstructs the useful information better can give benefit to these applications.

CHAPTER 4

PROPOSED ARCHITECTURE

4.1 Introduction

An important issue in image fusion is that there is no absolute correct reference to evaluate the results. Thus, no reference image fusion quality metrics are presented to measure the accuracy of the methods applied mentioned in Chapter 3.3. However, these evaluation metrics suffer from distinguishing results' strengths and weaknesses. In addition to objective metrics, subjective evaluations are required to examine experiments more accurately. The solutions generated for the image fusion problem are examined using these metrics. To overcome the issues related to the fusion quality, the problem is generally divided into three parts. However, this leads to another problem that the relationships between these parts may not be built up as desired. There can be some artifacts and related to connections between the parts. An artificial neural network presents a solution that handles the connection problems between parts. Furthermore, Each part can learn to represent its task better with data.

Deep learning methods need defining a metric called loss function to train the network in the desired way. The network should differentiate results' accuracy. In image fusion applications, we have no reference image to compare so, this makes it difficult to solve the problem with this approach. Commonly, pre-trained networks are used to extract features and classical methods are preferred to fuse these features. We try to learn extracting features of visible and infrared images, combining and reconstructing a fused image using an end-to-end deep neural network that handles all of them. To address both of these issues, we present a deep convolutional neural network(DCNN) architecture. Our main contributions are designing a solution that can represent the problem as a whole and can be trained using a combination of no-reference quality metrics.

4.2 Network Architecture

The proposed architecture is mainly inspired by U-Net architecture. The reason behind this choice is U-Net architecture [1] tries to find representation in lower-dimensional space. Information represented in the lower dimensions should include compressed features of the input. Figure 4.1 shows the top-down explanation of the autoencoder architecture. U-Net architecture can be considered as autoencoder but it has some improvements over it to overcome bottleneck problems. Image fusion applications need a good internal representation of input images to create a single image that includes all important information of all of the inputs. For instance, infrared images are successful in distinguish objects that have different thermal radiation whereas color images are successful in gathering color information of the object. When these two images are fused, the desired result should include both color information and thermal visibility information. Thus, intuitively, the architecture can extract rich features of all imaging conditions separately and we can combine them and recover desired fused images.

The U-Net architecture takes one input image and generates a single output image. In our case, we have two input images and we desire to obtain a single output image. To achieve this, we use two encoder layer that is exactly same structure but, each one has different parameters specializations. After the encoder layers, we get two feature maps in each level and we need to fuse them to give decoder layer. Thus, a fusion layer is added to the design. In the end, U-Net architecture is adapted to image fusion applications adding an extra encoder layer and new fusion layer.

The proposed architecture is composed of 3 parts: Encoder, Decoder, and Fusion. The complete overall architecture is shown in Figure 4.7. Encoder, Decoder, and Fusion layers are represented by yellow, blue, and green boxes respectively. Each part can be explained separately to understand overall architecture better. Thus, we are going to explain these parts in great detail.

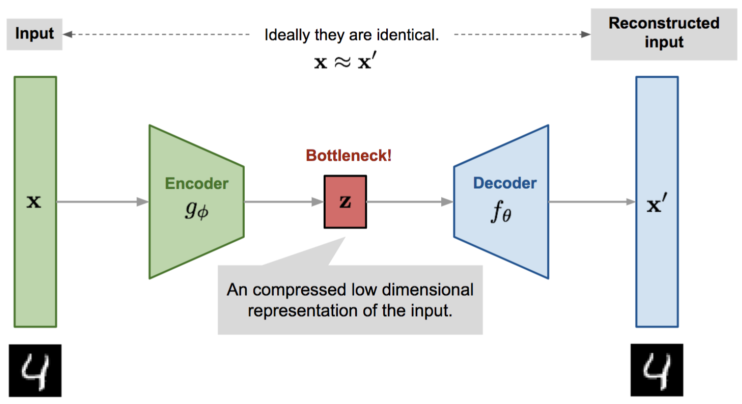

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 4.1: Autoencoder architecture. U-Net architecture can be considered as autoencoder but it has some improvements over it to overcome bottleneck problem. Image is taken from [39].

4.2.1 Encoder

The encoder can be considered as feature extraction and dimensionality reduction part of the architecture. It takes input and downscales in each layer to generate compressed features at the end. To downscale input, convolution with stride or pooling layer can be used. To recap knowledge, convolution needs two input functions. One is called kernel and another one is the input. In the deep learning area, kernel size is generally much smaller than input size which is typically 3 by 3 or 5 by 5. In Figure 4.2, convolution with stride 2 can downscale input to its half. The kernel moves along the input with step size 2. In each step, the kernel and corresponding input part are multiplied and summed. If the example input size is 6 by 6, the 2 by 2 kernel takes 3 positions across and down the input, so output becomes 3 by 3.

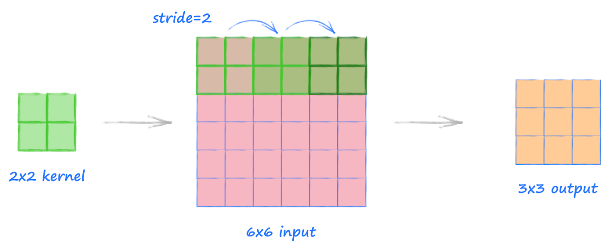


Figure 4.2: Convolution operation with stride 2 and kernel size 2 by 2. Image is taken from [40].

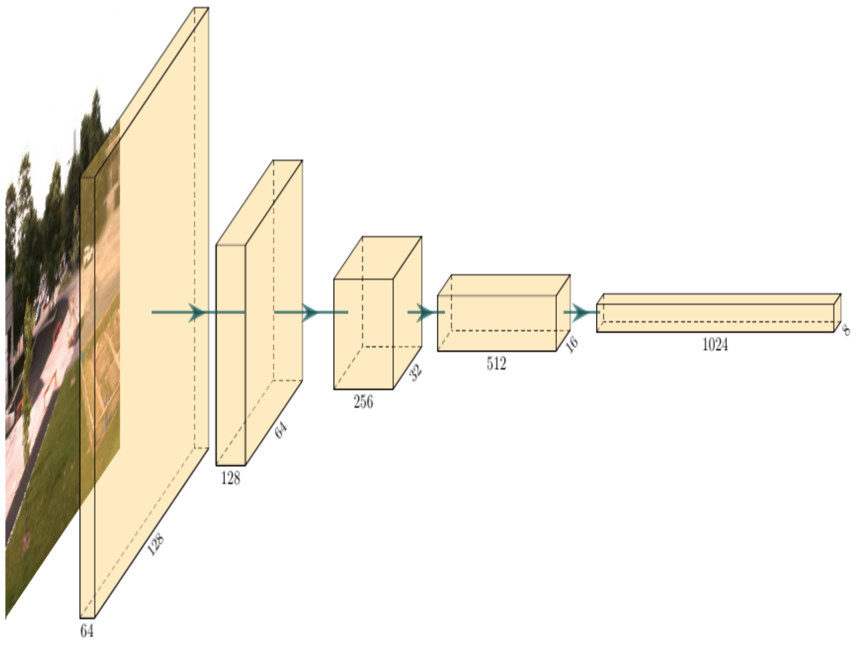


Figure 4.3: Basic encoder architecture. Includes only convolution layers.

Another downsampling method is pooling. It is generally quite a simple function that is the maximum or average operator. The pooling size specifies the downsampling rate. Input is divided into regions specified with the pooling size. The pooling operator is applied to each region separately.

A simpler encoder can be defined series of convolutional layers with stride 2. In each stage, feature map dimensions fold to 2 whereas width and height are cut in half. In Figure 4.3, the basic architecture is shown. Yellow boxes show that feature maps after convolutional operation applied. While the dimension of the feature map increases, the width and height of the input decrease. The advantage of such a design is that including fewer parameters thanks to the parameter sharing feature of the convolutional layer. Another advantage is that the input shape is not strictly determined. The only constraint exists for input's shape is they should be dividable by 32 because the network has 5 convolution layer with stride 2.

ResNet architecture [5] is commonly used in various tasks and it can extract good features of the input image. In chapter 2.2.2, detailed explanation and used areas of the ResNet can be found. It is generally used as a backbone network that extracts useful features because it is trained on a large dataset that is called ImageNet [74] that contains more than a billion images for a thousand object classes. Thus, this pre-trained network can decrease training time and also, can achieve better results. ResNet [5] uses both downsampling method mentioned in above. In Figure 4.4, an alternative view of the architecture that shows downsampling in ResNet [5] clearly. In the Conv1 layer, the max-pooling operation is used to downsample feature size. The remaining parts use convolution with stride to downsample the input feature map. The input image is reduced to one in 32. That is if the input image is a shape of 224 by 224, the last layer output will be 7 by 7. The last layers of the ResNet [5] are extracted and pre-trained layers are used as an encoder for our proposed architecture.

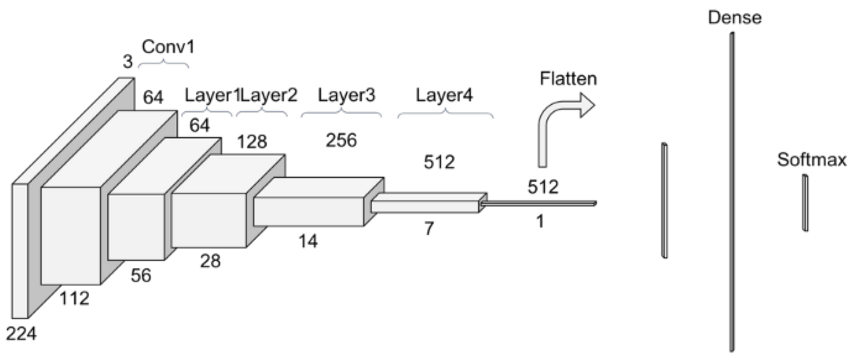


Figure 4.4: ResNet 34 architecture. Image is taken from [41]

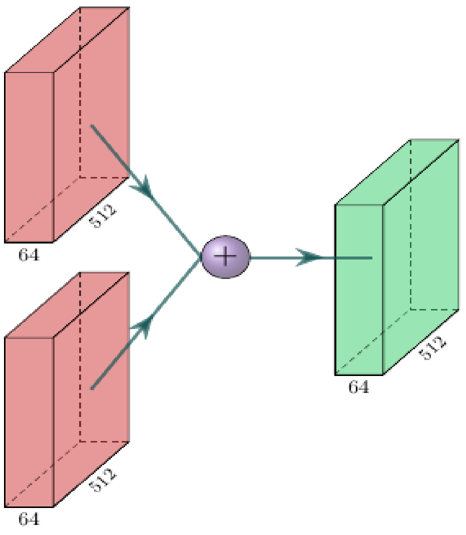
4.2.2 Fusion

The fusion layer combines feature maps of different spectrum images that are infrared and visible spectrum images in the proposed case. We need a design choice to unify feature maps. Thus, we come up with combination ideas that are used in neural network designs commonly to downsample feature maps such as in the ResNet model. There are 3 different methods to combine feature maps that we tried. The first one is to add feature maps. The feature map size is strictly equal and they can be added element-wise. The second one is to concatenate feature maps in the channel dimension and then applying a convolution layer to cut channel dimension to half without changing the width and height of the feature maps. The last method is the reverse order of the second method. The convolution operation is applied before concatenation. In Figure 4.5, methods are visualized. Our final proposed method recommends the second method that will be explained with great detail in Chapter 5 with experimental results.

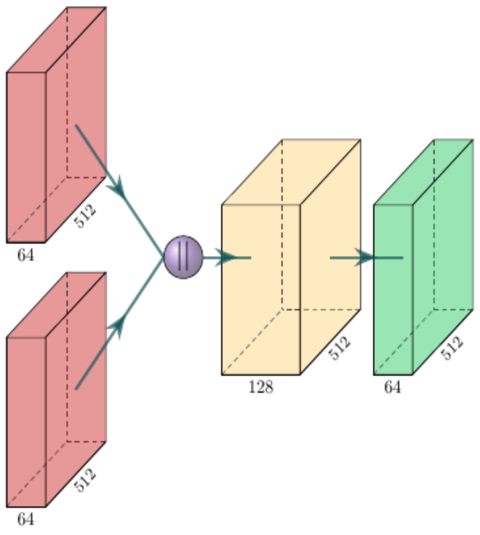
4.2.3 Decoder

The decoder needs to upsample given feature maps and generate a result that should have the same dimensions as the input image given to the encoder. The transposed convolution operation is used to upscale input feature maps. Before going into detail about the architecture of the decoder layer, transposed convolution operation should be understood better. First of all, mostly, the term deconvolution is used instead of transposed convolution. However, deconvolution is mathematically defined as the inverse of the convolution which is not related to transposed convolution. The main idea behind transposed convolution is to transform the convolution result to its original input.

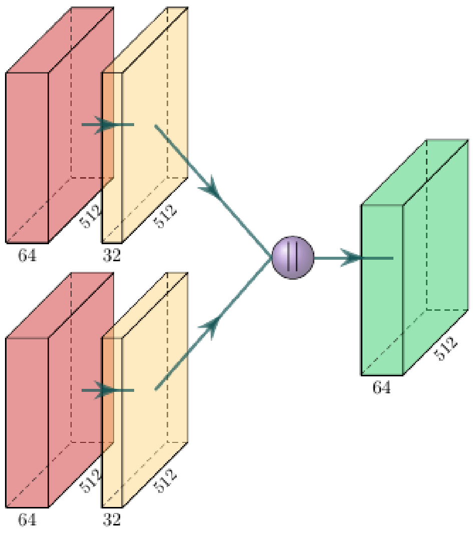
In our architecture, the decoder takes inputs from the fusion layer that includes 5 feature maps. Each one should be upsampled and concatenated with the previous feature layer until reaching the input image's dimensions. In Figure 4.6, decoder architecture is visualized. Each box drawn in the figure is the feature map after the corresponding operation. The width of the boxes represents the depth of the feature maps whereas the other two dimensions correspond to the width and height of the feature maps. Blue boxes show the result of deconvolution operation, green boxes are the outputs of the fusion layer and yellow box is the convolution layer to reduce the depth of the feature map to meet desired output channel size, for example, the channel size should be 3 for color(RGB) output image. Also, the output of the last layer should be normalized to view the image. Thus, after the last convolution layer, the yellow box in the figure, hyperbolic tangent, tanh, function applied to bound result between -1 and 1. In Eqn. A.2, the applied formula is given. In each transposed convolution layer, we used a 4 by 4 kernel with stride 2 and padding 1. Also, we used grouped convolutions to become a wider network to extract better representations of the inputs. In each transposed convolution layer, 32 groups are used. In the AlexNet [91], groups are used to train networks on multiple GPUs with smaller memory. Their methods require approximately 3 GB of GPU RAM to train, whereas their GPU, Nvidia GTX 580,



(a) Method 1: Add



(b) Method 2: Concatenate + Convolution



(c) Method 3: Convolution + Concatenate

Figure 4.5: Fusion method visualizations. Yellow boxes are the result of the convolution layer. Green boxes are the results of the fusion layer. Red boxes are the inputs that came from encoder layers. Purple circles define the operation used. '+' means element-wise adding while '||' means concatenation in channel dimension.

has 1.5 GB RAM. Although they were used for practical reasons, Deep Roots publication [92] investigates the effect of grouping in various architecture and they found that group convolution increases computational efficiency via decreasing parameter number and floating-point operations and achieve lower error rates.

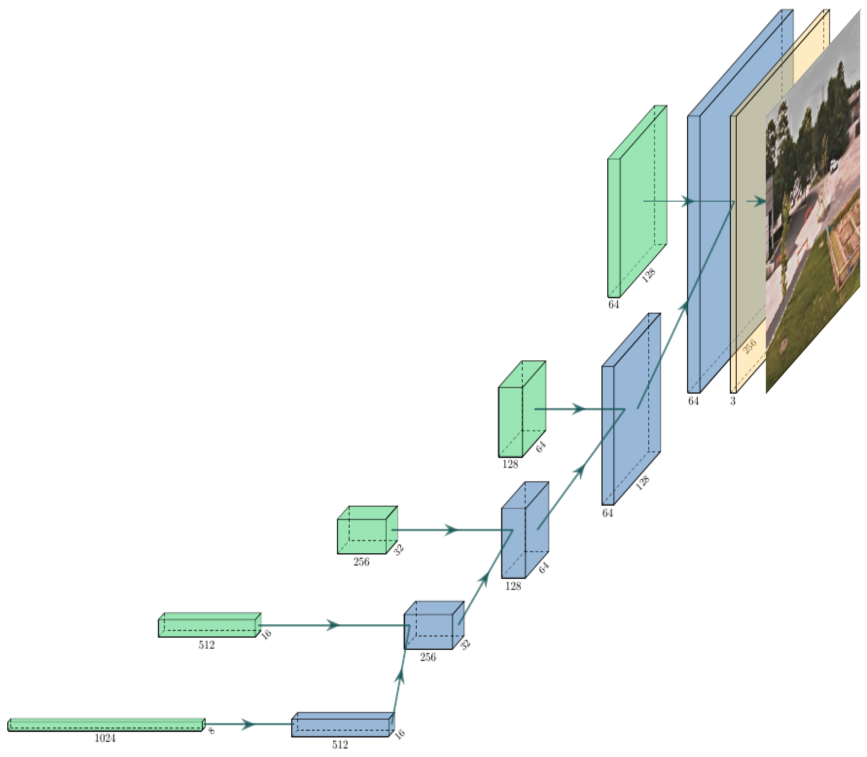


Figure 4.6: Decoder architecture. Blue boxes show transposed convolution results while green boxes are feature maps coming from the fusion layer. The yellow box is the convolution layer. If more than one arrow exists for any layer, these feature maps are concatenated at depth dimension which is indicated as the width of the boxes.

4.3 Loss Functions

There is no reference or ground truth images for the image fusion task, thus, it is crucial to choose a suitable metric that can measure the quality of the fused image by comparing input images. There exist some fusion quality metrics in the literature and our aim is to use these metrics as loss functions to train our network. We use Q_w [86], that compares two input images with result image in local regions and mean square

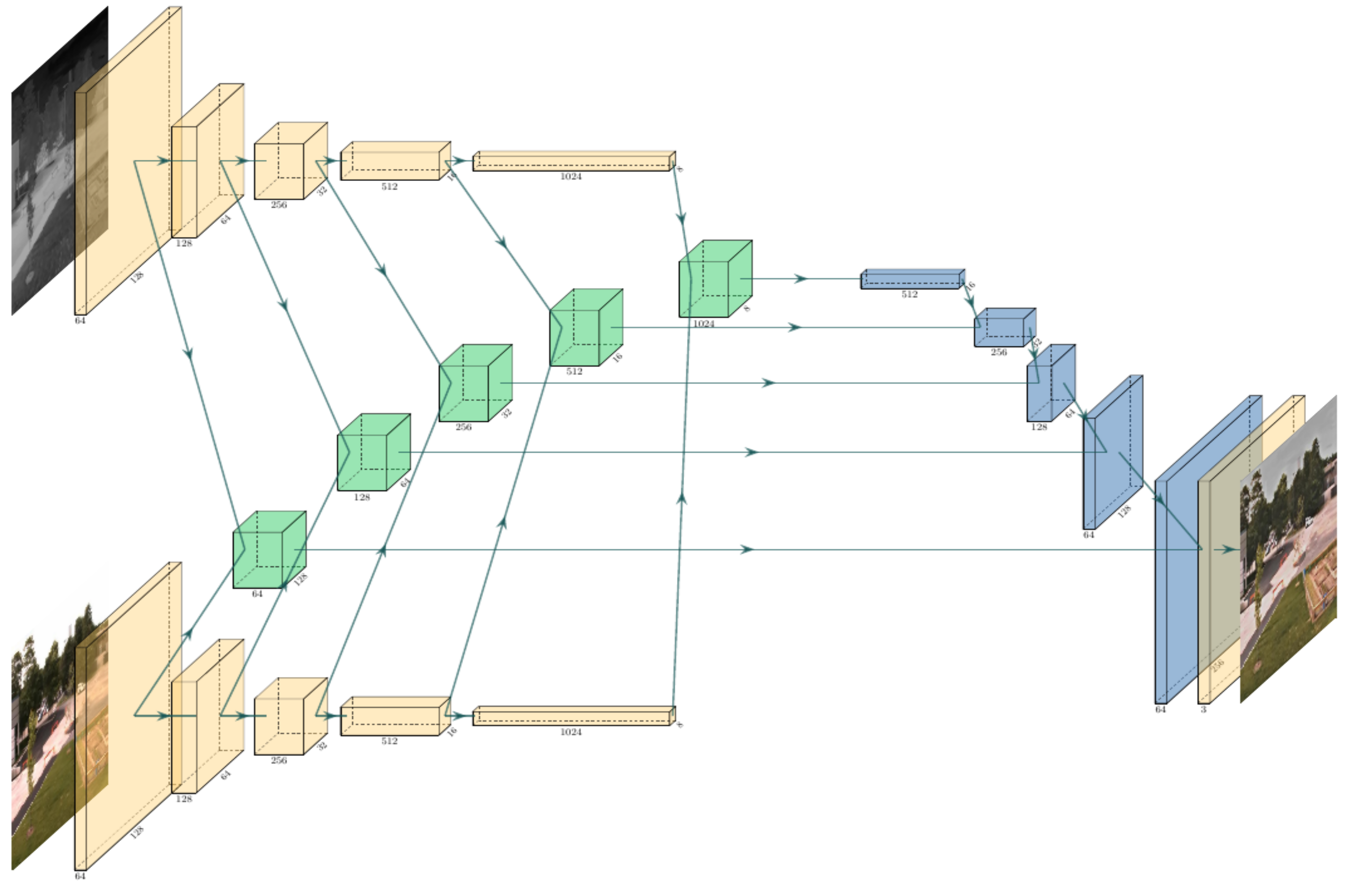


Figure 4.7: Overall architecture. Yellow boxes represent the encoder layers. Blue boxes represent the decoder layer. Green boxes represent the fusion layer. The architecture is called as Encoder-Decoder-Fusion.

errors(MSE) between result and input images. Our choice depends on the idea that the quality and edge information can be beneficial to extract object information more clearly. Image quality measurements are considered and compared in Chapter 3.

Let $\bar{x}$ is the mean of x, $\bar{y}$ is the mean of y, σ_x denote square-root of variance, σ_{xy} is the covariance of input image patches x and y, i.e.,

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2, \qquad \sigma_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

The quality assessment measure can be computed as follows between two input x and y, the equation can be decomposed and as Eqn. 4.1 which is introduced by Wang and Bovik in [85].

$$Q_0 = \frac{4\sigma_{xy}\bar{x}\bar{y}}{(\bar{x}^2 + \bar{y}^2)(\sigma_x^2 + \sigma_y^2)}$$

In the each comparison, it takes into account correlation coefficient $\frac{\sigma_{xy}}{\sigma_x \sigma_y}$, luminance distortion $\frac{2\bar{x}\bar{y}}{\bar{x}^2+\bar{y}^2}$ and contrast distortion $\frac{2\sigma_x\sigma_y}{\sigma_x^2+\sigma_y^2}$ between images given in Eqn.4.1. It's range is between -1 and 1.

$$Q_0 = \frac{\sigma_{xy}}{\sigma_x \sigma_y} * \frac{2\bar{x}\bar{y}}{\bar{x}^2 + \bar{y}^2} * \frac{2\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2} \tag{4.1}$$

The Q_0 metric compares only two inputs in patches. In the fusion quality metric, a function should be defined to take inputs of more than 2, in our case, it is 3, infrared, visible, and result image. To calculate the fusion quality index, Q_w, the image is divided into square regions. For all these regions, Q_0 scores are calculated. The final score is calculated as an average of all local scores shown in Eqn. 4.2. Fusion quality score compares the fused image with input images separately in local regions. To combine them, the local relevance of each region should be determined. If the infrared image includes a feature that does not exist in the visible image, the infrared image's local region score should be more important than the visible image's. This feature can be contrast, sharpness or entropy, etc. Thus, a local weight $\lambda(w)$ is calculated using local saliency $s(a|w)$ and $s(b|w)$ of the images a and b. The $\lambda(w)$ is between 0 and 1 that indicates relative importance of the image a over image b. That is the larger $\lambda(w)$ means image a has more weight when calculating score for this local region. Eqn.

4.3 shows the calculation of the weight for region w.

$$Q_0(a,b) = \frac{1}{W} \sum_{w \in W} Q_0(a,b|w) \tag{4.2}$$

$$\lambda(w) = \frac{s(a|w)}{s(a|w) + s(b|w)} \tag{4.3}$$

In the [86], they define the local saliency $s(a|w)$ and $s(b|w)$ as variances of input image a and b respectively so, Eqn. 4.3 becomes

$$\lambda w = \frac{\sigma^2_{(a|w)}}{\sigma^2_{(a|w)} + \sigma^2_{(b|w)}}$$

and patch size is taken as 8 by 8. Also, to combine all patches' scores, [85] defines a weight $c(w)$ for each local region. Thus, each local region has a different impact on the final quality index score. $c(w)$ is calculated as

$$c(w) = \frac{max(s(a|w), s(b|w))}{\sum_{w' \epsilon W} max(s(a|w), s(b|w))}$$

.

The final score is defined in the Eqn. 4.4 which is called a fusion quality index. The score becomes higher if the fused image is constructed better and it bounds between -1 and 1. Thus, to use Q_0 in the loss function, we extract it from 1 to give zero loss when the fusion result is excellent. The Q_0 score can be calculated with edge images instead of original images. Thus, it can compare the edge responses of the images which are beneficial to us to construct fused results better. Let a', b' and f' is the edge responses of a, b and f respectively. Thus edge score Q_E equals to $Q_0(a', b', f')$. Edge responses are calculated using Canny Edge detection algorithm [93] and GPU supported code is implemented with following guide given by Thevenot [94].

$$Q_w(a,b,f) = \frac{1}{W} \sum_{w \in W} c(w) \Bigg(\lambda(w) Q_0(a,f|w) + (1 - \lambda(w)) Q_0(b,f|w) \Bigg) \tag{4.4}$$

Besides Q_0 score, mean square error MSE is implemented to recover image similarities better. The MSE calculation is

$$MSE(a,b) = \frac{1}{N}\sum_{i=1}^{N}(a_i - b_i)^2$$

.

MSE compares only two inputs so, MSE scores are calculated between fused image f and input images a and b separately and the average of total MSE score is computed. This score is strictly positive and gives smaller outputs when the images are similar. The overall loss function is defined in the Eqn.4.5. α, β, and γ are the loss weights to normalize losses to make them equally important.

$$\begin{aligned} L(a,b,f) = \alpha * (1 - Q_w(a,b,f)) + \beta * (1 - Q_e(a,b,f)) \\ +\gamma * 0.5 * (MSE(a,f), MSE(b,f)) \end{aligned} \tag{4.5}$$

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Introduction

The fusion methods are compared quantitatively and qualitatively to show weaknesses and strengths. The performance evaluation metrics explained in Chapter 3.3 are used to compare the methods quantitatively. Also, the mean opinion score metric is used to specify the overall image quality of the fused images.

In this chapter, the experiments conducted on the proposed method and performance evaluation are explained and quantitative results are shared. First of all, the training and inference details of our neural network are shared and the design choices of the proposed method are investigated. Then, the performance evaluation of our best-proposed method is done. The results are compared with deep learning-based methods to see the success in its group. Finally, all methods are compared. Furthermore, in all comparisons, the fused images obtained using some sequences of TNO [2] and VIFB [3] datasets are shown. It is important to see the differences between the methods visually.

5.2 Training and Inference Details

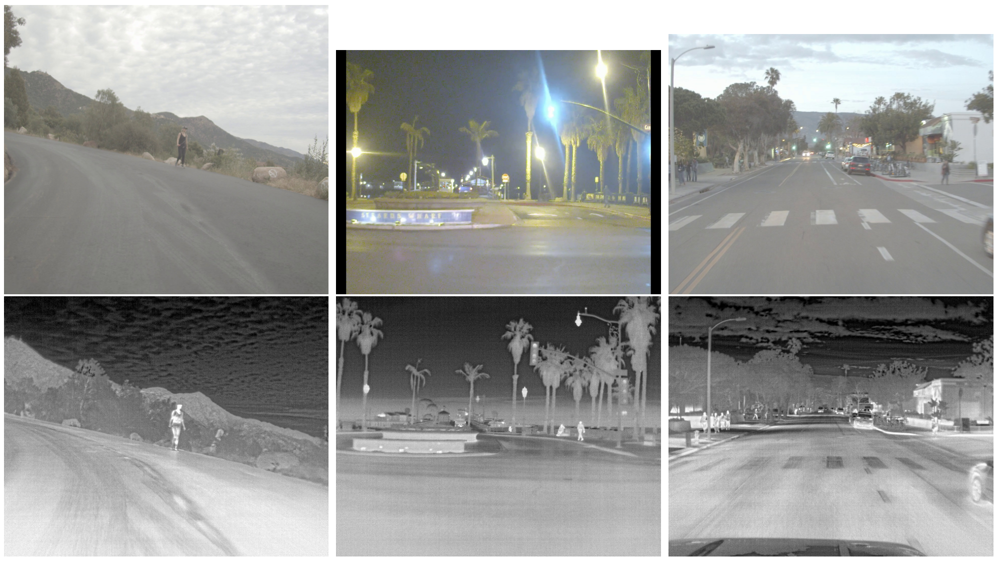
The entire network described in the previous sections is trained with Flir ADAS [95] and KAIST [96] datasets which both of them include day and night conditions. In Figure 5.1, some sample images from the training datasets are shown. There are 8363 image pairs in FLIR ADAS [95] dataset and 12539 image pairs in KAIST [96] dataset. In total, there are 20902 image pairs are used in the training. In addition to

Table 5.1: Number of parameters of the different types of the proposed method.

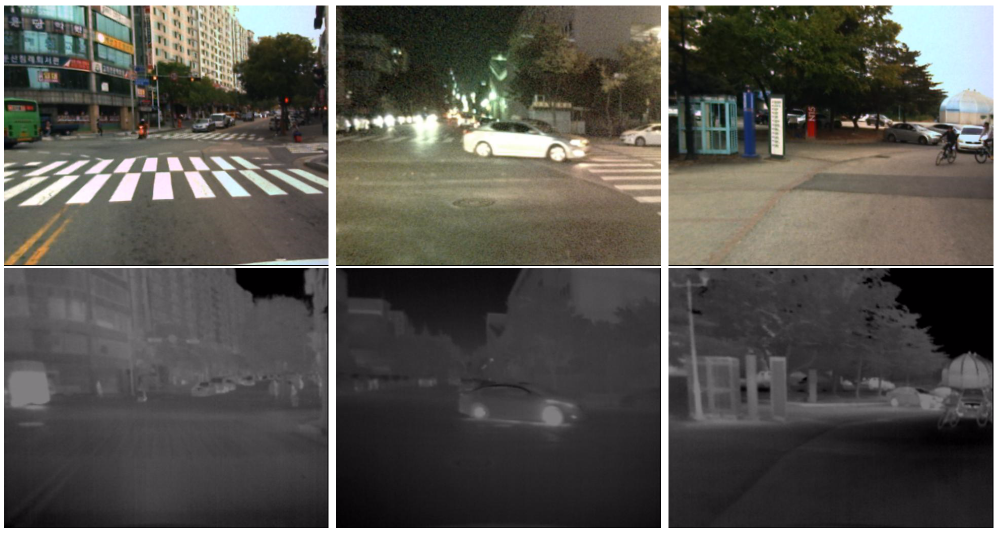
Model Types	Fusion Types		
	Add	Concatenate + Convolution	Convolution + Concatenate
Encoder with ResNet50 pre-trained	100,000,467	111,161,939	102,791,827
Basic Encoder	13,414,912	16,210,368	14,114,272

training datasets, we have two visible and infrared image pairs datasets that are TNO [2] and VIFB [3] datasets that include various scenes but smaller samples. Most of the fusion algorithms are compared using these datasets. In TNO [2], visible images are grayscaled whereas, in VIFB [3], most of the visible images are colored. In both datasets, there are 21 visible and infrared image pairs. In the training phase, Gaussian blur and Gaussian noise are added to the input images. Input image size is fixed to 256x256 and input images are normalized between -1 and 1. The initial learning rate is 0.001. ADAM optimizer is used with $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$ as mentioned in the [97]. Also, a multi-step learning rate scheduler is used. Steps are decided as 10, 20, and 30, and the decay factor for the learning rate is 0.5. That is after reaching the above epoch numbers in the training, the learning rate is cut in half. 30 epochs are generally enough to train the network with stated data sets. In Table 5.1, the number of parameters used in the different types of the proposed method is given. The number of parameters affects the inference and training times.

All system is implemented in Pytorch 1.6.0 with CUDA 10.2. The computer has Intel i5 4460 CPU, Nvidia GTX 970 4GB GPU, and 8 GB system memory. Approximately, whole training datasets include 20K samples. One epoch, i.e. whole training dataset used once, takes 161 mins with the above setup for Encoder with ResNet50 pre-trained model using Concatenate + Convolution fusion type which is the maximum parameter size among all proposed methods given in Table 5.1.



(a) Sample images from Flir ADAS [95] dataset. Each column shows the visible and infrared image pairs exist in the dataset.



(b) Sample images from KAIST [96] dataset. Each column shows the visible and infrared image pairs exist in the dataset.

Figure 5.1: Sample images that is used in the training. They include night and day conditions.

5.3 Experiments on Proposed Method

Experiments are conducted to construct the best model that achieves the best scores both qualitatively and quantitatively on image fusion and analyze the effects of design choices. Fusion layer type, pre-trained encoder layer, and loss function are the design choices of the proposed method. To find the best model, the following experiments are made. First of all, the learning capability of the basic network with different loss functions is analyzed. The basic network model means that the encoder layer is composed of the only convolutional layer without any pre-training phase and the fusion layer is simply add. Then, the effect of the pre-trained encoder layer that uses the ResNet-50 network is investigated. Lastly, the effect of the fusion layer type is examined.

5.3.1 Effect of Loss Function

In the proposed method, loss function is composed of three parts: Q [86], Q_E [86] and MSE mentioned in the Section 4.5. Firstly, only Q_E that measures fusion quality with respect to edge response is used. Then, Q that measures fusion quality with respect to visual silences is added to loss function and trained with this setup. Lastly, MSE that measures the distance of the source and fused images pixel-based is added to see if there are further improvements. Thus, intuitions about the fusion quality score Q, Q_E, and MSE are as follows. Q measures the transformation quality of objects and their features like texture, shape, and color that are important for the scene. Q_E measures the edge quality of the fusion image relative to source images. MSE measures the closeness of the fused image to the source images in pixel value in order to understand how a natural looking image is obtained because, the mathematically optimal solution for other fusion quality metrics can achieve a result that has unrealistic, dark, or bright images.

The results are compared both quantitatively and qualitatively on TNO and VIFB datasets. MSE, $L1$, Q_W, Q_E, and Q_Y scores are compared. Table 5.2 and 5.5 show the average scores of the proposed methods that are trained with different loss functions. It is seen that combining MSE, Q_E, and Q in the loss function gives the best result while holding other design choices the same.

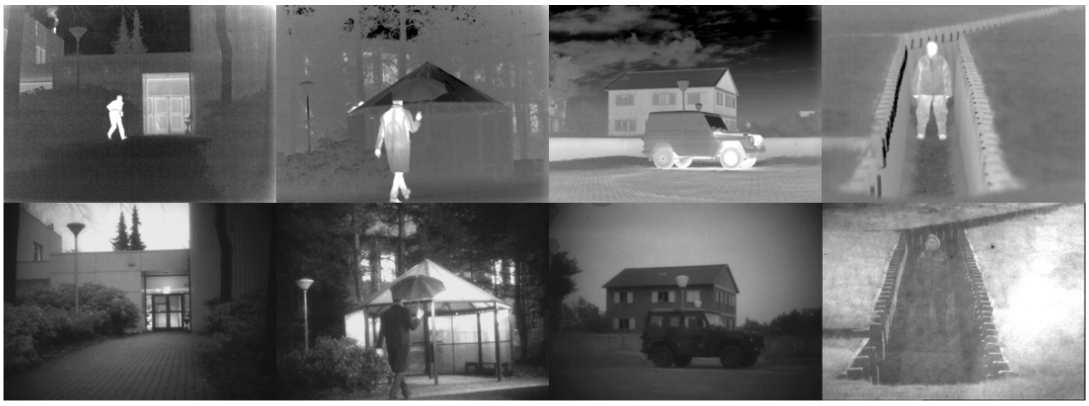
Table 5.2: MSE, $L1$, Q, Q_W, Q_E and Q_Y scores of the proposed method with indicated loss functions. Scores are generated on TNO Dataset [2]. Green results indicates the best result in the corresponding metric among all of the methods.

Quality Metrics / Loss Functions	MSE	$L1$	Q_E	Q	Q_W	Q_Y
$MSE + Q_E + Q$	0.128	0.002	0.836	0.864	0.725	0.876
Q_E	0.153	0.014	0.807	0.739	0.599	0.774
$Q_E + Q$	0.181	0.008	0.792	0.839	0.669	0.805

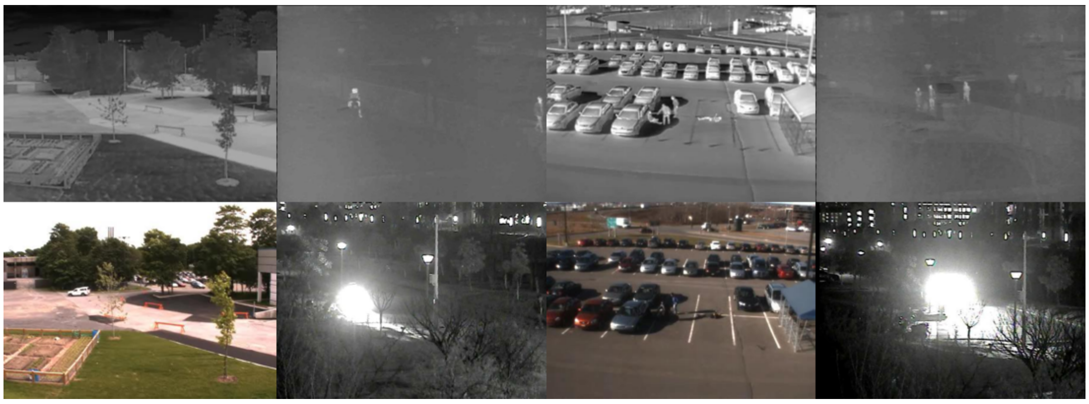
Table 5.3: MSE, $L1$, Q, Q_W, Q_E and Q_Y scores of the proposed method with indicated loss functions. Scores are generated on VIFB Dataset [3]. Green results indicates the best result in the corresponding metric among all of the methods.

Quality Metrics / Loss Functions	MSE	$L1$	Q_E	Q	Q_W	Q_Y
$MSE + Q_E + Q$	0.109	0.008	0.797	0.796	0.634	0.882
Q_E	0.146	0.021	0.743	0.566	0.423	0.756
$Q_E + Q$	0.175	0.021	0.722	0.735	0.533	0.806

Furthermore, the results are compared qualitatively. Figure 5.3 shows the qualitative results for input image pairs given in Figure 5.2. For TNO [2] dataset input pairs, results of the proposed method that is trained with $MSE + Q_E + Q$ loss have more contrast and clarity. The method trained with only Q_E loss function can help to recover edges better but, visual salient regions are not reconstructed well. The method used $Q_E + Q$ loss function give better result than only Q_E loss function used method, however, some information in infrared images are not recovered well. For example, in the third row of the Figure 5.3a, The jeep is hot, that is probably parked yet, and the information only exists in the first row. On the other hand, using VIFB [3] dataset input pairs, we can see the effect of loss function type to recover colors in Figure 5.3b. Using only Q_E loss function couldn't reconstruct colors well. Thus, in the below experiments, the loss function is taken as a combination of MSE, Q_E, and Q



(a) Visible and infrared image pairs in TNO Dataset [2]. The sequence numbers are 12, 13, 16 and 21 from left to right respectively. Top row is infrared images while bottom row is visible images.



(b) Visible and infrared image pairs in VIFB Dataset [3]. The names of the image pairs are 'carWhite', 'elecbike', 'fight' and 'manlight' from left to right respectively. Top row is infrared images while bottom row is visible images.

Figure 5.2: Input images that is used to compare methods qualitatively.

that is explained in Chapter 4.3.

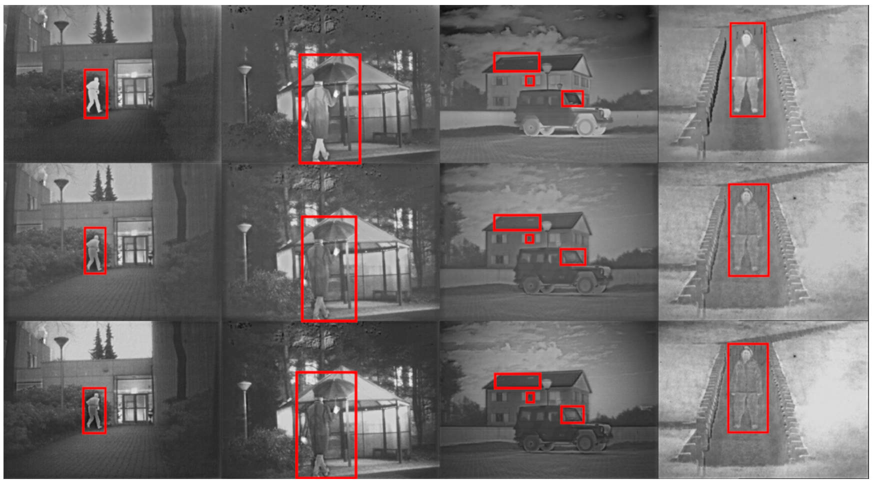
After choosing loss function combination, we investigated effects of the weights of the each part to the training accuracy. These weights α, β, and γ are given in Eqn. 4.5. We tried to change weights of the method. At the end of the first epoch, the quality metric that has higher weight than others have higher score than other training setups. When importance of the quality metric in the loss function increases, the proposed architecture converges to lowest error for this quality metric faster. Thus, the results of the first epoch change. However, continuing training until the loss converges, the changes weight does not effect the overall score of the method.

Table 5.4: MSE, $L1$, Q, Q_W, Q_E and Q_Y scores of the proposed method with indicated loss functions weights. Scores are generated on VIFB and TNO datasets. The network are trained 1 epoch using the Eqn. 4.5 as loss function. Green results indicates the best result in the corresponding metric among all of the methods.

Loss Function Weights			Quality Metrics					
α	β	γ	MSE	$L1$	Q_E	Q	Q_W	Q_Y
1	1	1	0.186	0.008	0.722	0.703	0.525	0.812
10	1	1	0.177	0.007	0.727	0.734	0.552	0.820
1	10	1	0.190	0.010	0.738	0.704	0.535	0.819
1	1	10	0.198	0.023	0.728	0.748	0.557	0.813

Table 5.5: MSE, $L1$, Q, Q_W, Q_E and Q_Y scores of the proposed method with indicated loss functions weights. Scores are generated on VIFB and TNO datasets. The network are trained 10 epoch using the Eqn. 4.5 as loss function. Green results indicates the best result in the corresponding metric among all of the methods.

Loss Function Weights			Quality Metrics					
α	β	γ	MSE	$L1$	Q_E	Q	Q_W	Q_Y
1	1	1	0.175	0.005	0.749	0.755	0.563	0.825
10	1	1	0.168	0.004	0.748	0.752	0.574	0.827
1	10	1	0.176	0.009	0.751	0.744	0.549	0.822
1	1	10	0.171	0.003	0.748	0.774	0.590	0.823


(a) The fused images using TNO Dataset [2] inputs in Figure 5.2a.

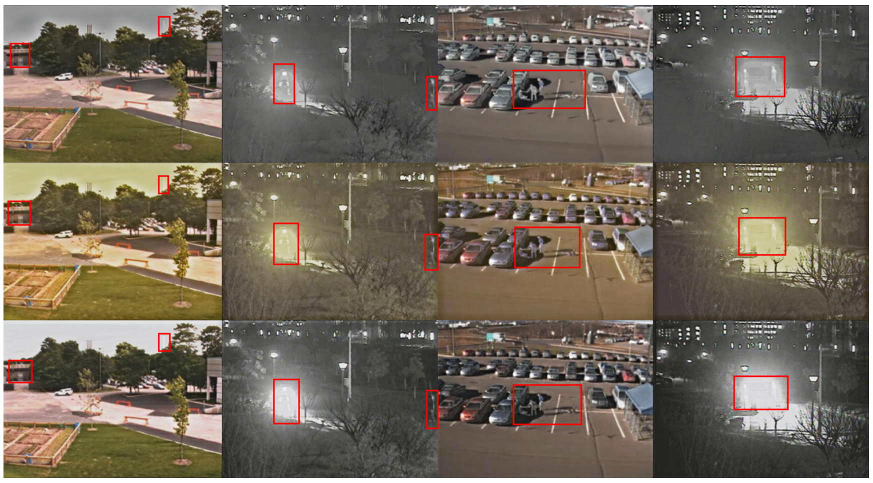

(b) The fused images using VIFB Dataset [3] inputs in Figure 5.2b

Figure 5.3: The fusion results of the different loss functions given in Tables 5.2 and 5.5. The proposed method is trained with $MSE + Q_E + Q$, Q_E, and $Q_E + Q$ loss functions separately and results are generated from trained neural networks in the mentioned order from top to bottom. Red squares show the regions that are some important visual information that is used to check overall quality with the naked eye. Better viewed in digital zoom these red rectangles.

5.3.2 Effect of Pre-trained Encoder Layer

In this part, we compare the effect of using a pre-trained network before training started. Using pre-trained weights in the object classification task can extract better features and it is obtained through larger datasets. In the Table 5.6 and 5.7, the effect of pre-trained network is clearly seen. Pre-trained ResNet network boosts performance greater in fusion quality metrics Q_W, Q_E, and Q_Y. In MSE, $L1$ metrics, performance is not affected much or even worse for the pre-trained option because these metrics compare the distance of the pixel values. However, the other metrics show significant improvements.

Table 5.6: MSE , $L1$, Q, Q_W, Q_E and Q_Y scores of the proposed method with indicated training networks. Scores are generated on VIFB Dataset [3]. Green results indicates the best result in the corresponding metric.

Quality Metrics / Training Networks	MSE	$L1$	Q_E	Q	Q_W	Q_Y
Pre-trained	0.128	0.002	0.836	0.864	0.725	0.876
Random Initialized	0.146	0.001	0.804	0.834	0.675	0.819

Table 5.7: MSE , $L1$, Q, Q_W, Q_E and Q_Y scores of the proposed method with indicated training networks. Scores are generated on TNO Dataset [2]. Green results indicates the best result in the corresponding metric.

Quality Metrics / Training Networks	MSE	$L1$	Q_E	Q	Q_W	Q_Y
Pre-trained	0.109	0.008	0.797	0.796	0.634	0.882
Random Initialized	0.134	0.007	0.745	0.753	0.563	0.822

5.3.3 Effect of Fusion Layer

There are 3 different methods to combine feature maps: Add, Concatenate + Convolution, and Convolution + Concatenate. In Chapter 4, how and why we decide these fusion methods. To explain briefly, these methods are used to downsample feature maps and various applications obtain benefits from these methods. To see their effectiveness, we conducted the following experiments to decide our last design choice. Thus, previously selected best design practices were applied to this part. Pre-trained ResNet-50 network is used in encoder part and combined $MSE + Q_E + Q$ loss function is used to train all experiments below. Table 5.8 and 5.9 show the MSE, $L1$, Q, Q_W, Q_E and Q_Y scores for TNO and VIFB datasets respectively. According to these scores, Concatenate + Convolution fusion method is the best among all three.

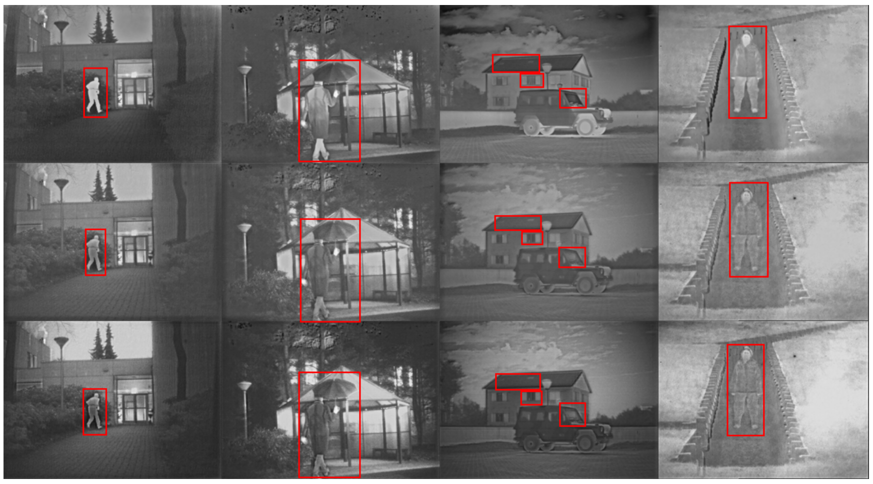
Table 5.8: MSE , $L1$, Q, Q_W, Q_E and Q_Y scores of the proposed method with indicated fusion methods. Scores are generated on VIFB Dataset [3]. Green results indicates the best result in the corresponding metric among all of the methods.

Quality Metrics / Fusion Methods	MSE	$L1$	Q_E	Q	Q_W	Q_Y
Concatenate + Convolution	0.128	0.002	0.836	0.864	0.725	0.876
Convolution + Concatenate	0.135	0.003	0.822	0.851	0.703	0.850
Add	0.130	0.001	0.819	0.849	0.698	0.853

Table 5.9: MSE , $L1$, Q, Q_W, Q_E and Q_Y scores of the proposed method with indicated fusion methods. Scores are generated on TNO Dataset [2]. Green results indicates the best result in the corresponding metric among all of the methods.

Quality Metrics / Fusion Methods	MSE	$L1$	Q_E	Q	Q_W	Q_Y
Concatenation + Convolution	0.109	0.008	0.797	0.796	0.634	0.882
Convolution + Concatenation	0.116	0.010	0.765	0.775	0.594	0.857
Add	0.110	0.005	0.774	0.780	0.604	0.858

Apart from quantitative experiments, we analyzed visual results to obtain better intuition. In Figure 5.4, the results regarding to both TNO and VIFB dataset image pairs given in Figure 5.2 are shown. The results are so close to each other but, some minor differences can be noticed when the images are reviewed with digital zoom. In TNO dataset results, the 3rd column that includes Jeep has better contrast in cloud regions for Concatenate + Convolution fusion method. Also, a similar observation can be noticed in the VIFB dataset results' 2nd column. A person who drives an electric bike is more distinguishable in the first row. Using quantitative and qualitative results, It is decided to use Concatenate + Convolution fusion method.


(a) The fused images using TNO Dataset [2] inputs in Figure 5.2a.

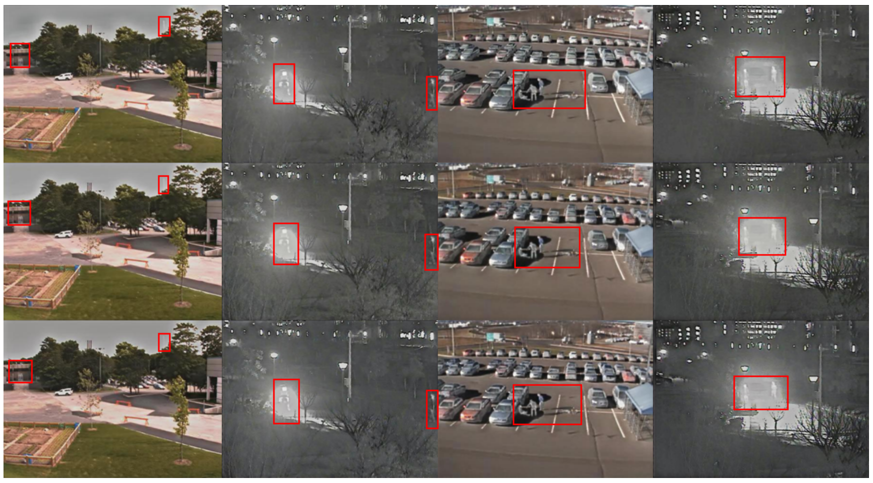

(b) The fused images using VIFB Dataset [3] inputs in Figure 5.2b

Figure 5.4: The fusion results of the different fusion methods given in Tables 5.8 and 5.9. The proposed method is constructed using these fusion methods. Concatenate + Convolution, Convolution + Concatenate and Add fusion methods are used from top to bottom respectively. Red squares show the regions that are some important visual information that is used to check overall quality with the naked eye. Better viewed in digital zoom these red rectangles.

5.4 Performance Evaluation

To understand the success of the proposed method, it should be compared with the existing state-of-the-art methods. In Chapter 3, existing methods are reviewed and performance metrics are explained briefly. In the literature, each performance metric has strengths and weaknesses. However, there does not exist a universal quality metric that can measure fusion quality perfectly. Thus, it is common to use a mean opinion score (MOS) that takes into account real individuals' opinions. MOS experiments are out of the scope of this thesis work but, we tried to predict MOS using a state-of-the-art image quality assessment metric that mimics the real human raters. The name of the metric is PaQ-2-PiQ. Images are compared with both fusion quality measures and perceptual quality metric (PaQ-2-PiQ). Apart from quantitative comparisons, visual results are given to explore differences in the image fusion methods.

Firstly, We compare the proposed method in its category that is deep learning-based methods. Then, all methods are compared to see the overall picture. TNO and VIFB datasets are used to investigate the strength and weaknesses of the methods. Also, VIFB has some color images that are used to compare the color information recovery of the methods. In Table 5.10, method names used in TNO and VIFB datasets are given in corresponding order. Rfn-Nest, Dual-Branch, and FusionGAN methods cannot perform color image recovery that is they generate single channel output. They should be trained with changing the output condition of the design to generate a three-channel image. In all of the experiments conducted in this part, pre-trained models are used. The pre-trained weights are obtained through their project pages. Most of the output fused images of the other methods are given on their projects' page. The remaining methods' output fused images are computed with the help of their project pages.

5.4.1 Only Deep Learning Based Methods Comparisons

There are currently 8 state-of-the-art deep learning-based methods that exist and they are reviewed in Chapter 3.2.6. Pre-trained weights shared by authors are used for these methods. Thus, some methods require a training network because the pre-

Table 5.10: Method names and corresponding numbers used in TNO and VIFB dataset comparisons.

Number	TNO Dataset Method Names	VIFB Dataset Method Names
1	ADF	ADF
2	CBF	CBF
3	CNN	CNN
4	DeepFuse	DeepFuse
5	DenseFuse	DenseFuse
6	DLF	DLF
7	Dual-Branch-addFusion	Dual-Branch-addFusion-grayscaled
8	Dual-Branch-channelFusion	Dual-Branch-channelFusion-grayscaled
9	Dual-Branch-l1Fusion	Dual-Branch-l1Fusion-grayscaled
10	FPDE	FPDE
11	FusionGan	FusionGan-grayscaled
12	GFCE	FusionGan-YChannelRGB
13	GFF	GFCE
14	GTF	GFF
15	HMSD_GF	GTF
16	Hybrid-MSD	HMSD_GF
17	IFEVIP	Hybrid_MSD
18	LatLRR	IFEVIP
19	MDLatLRR-level-1	LatLRR
20	MDLatLRR-level-2	MDLatLRR-level-1
21	MDLatLRR-level-3	MDLatLRR-level-2
22	MDLatLRR-level-4	MDLatLRR-level-3
23	MGFF	MDLatLRR-level-4
24	MST_SR	MGFF
25	MSVD	MST_SR
26	NSCT_SR	MSVD
27	Proposed-Best	NSCT_SR
28	ResNet-l1Zca4	Proposed_Best
29	ResNet-l1Zca5	ResNet
30	RFN-Nest	Rfn-Nest-grayscaled
31	RP_SR	RP_SR
32	TIF	TIF
33	VSMWLS	VSMWLS

trained network does not support three-channel image generation. For these methods, either YCbCr image channel is used to recover colors or gray-scaled versions are used for comparisons. Y channel of the visible image is used to generate the output image and the result is combined with input's Cb and Cr channels to construct colors. However, the results of this method didn't give satisfying results and gray-scaled options are used most of them. Rather than colors, we can compare all of the information regarding visible and infrared imaging physics. The TNO and VIFB datasets will be reviewed individually to be more clear. First of all, MSE, $L1$, Q, Q_W, and Q_Y scores of the methods will be given. Then, scores of all metrics mentioned in Chapter 3.3 will be given. The perceptual quality measurements versus Q_W, Q_E, and Q_Y scores are given as figures. Lastly, fusion results of the images to some input pairs will be shown.

5.4.1.1 TNO Dataset Comparisons

TNO dataset's visible spectrum images are grayscaled. Thus, we cannot compare the color recovery performance of the metrics unless the gray tones. First of all, deep learning-based metrics are compared quantitatively. The performance metrics Q_E, Q, Q_W, and Q_Y give more reliable results that are consistent and give better intuition about picture quality. Thus, in Table 5.11, only these scores with the most fundamental scores MSE and $L1$ are given. MSE and $L1$ scores give clues about how well the input images resemble the fused image. However, in image fusion, information of the input images should be preserved well and this makes fused images can recover information well while it cannot numerically resemble well to the input images. According to Table 5.11, CNN, DeepFuse and our proposed methods give best results in the Q_E, Q, Q_W metrics.

Table 5.11: Reliable performance metrics with the MSE and $L1$ scores that show numerically distance to the input images. Top three scores are highlighted with green, blue and red respectively for each metric. Results are obtained in TNO dataset.

Quality Metrics / Method Names	MSE	$L1$	Q_E	Q	Q_W	Q_Y
CNN [21]	0.161	0.029	0.803	0.799	0.642	0.820
DeepFuse [25]	0.123	0.019	0.710	0.725	0.519	0.809
DenseFuse [24]	0.105	0.005	0.609	0.655	0.399	0.830
DLF [22]	0.106	0.007	0.621	0.673	0.418	0.833
Dual-Branch-addFusion [28]	0.117	0.012	0.650	0.682	0.444	0.839
Dual-Branch-channelFusion [28]	0.123	0.004	0.601	0.535	0.349	0.794
Dual-Branch-l1Fusion [28]	0.117	0.012	0.650	0.682	0.444	0.839
FusionGan [26]	0.101	0.061	0.479	0.434	0.212	0.730
Proposed Best	0.109	0.008	0.797	0.796	0.634	0.882
ResNet-l1Zca4 [23]	0.107	0.007	0.635	0.682	0.433	0.837
ResNet-l1Zca5 [23]	0.107	0.007	0.616	0.663	0.409	0.832
RFN-Nest [27]	0.134	0.028	0.651	0.661	0.437	0.772

On the other hand, we compare the deep learning-based methods with all performance metrics in Tables 5.12 and 5.13. When we look at overall successes over the performance metrics, CNN, DeepFuse, and our proposed method beat other methods in most of the performance metrics. In our first comparison in Table 5.11, these three methods give the best scores. This shows that Q_E, Q, Q_W and Q_Y performance metrics are more reliable compared to others.

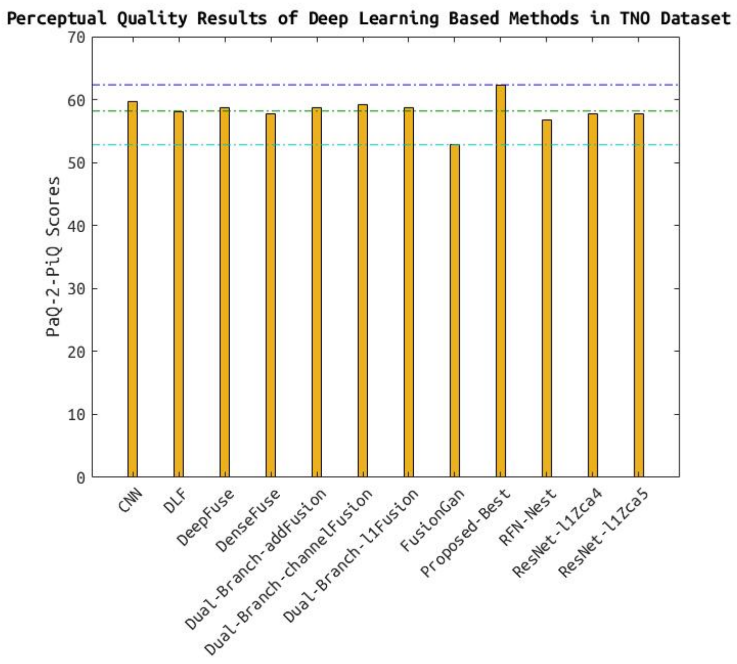


Figure 5.5: Average perceptual quality (PaQ-2-PiQ) [42] scores for deep learning-based methods. Results are obtained in the TNO dataset. Our proposed method obtains the best score among all other methods.

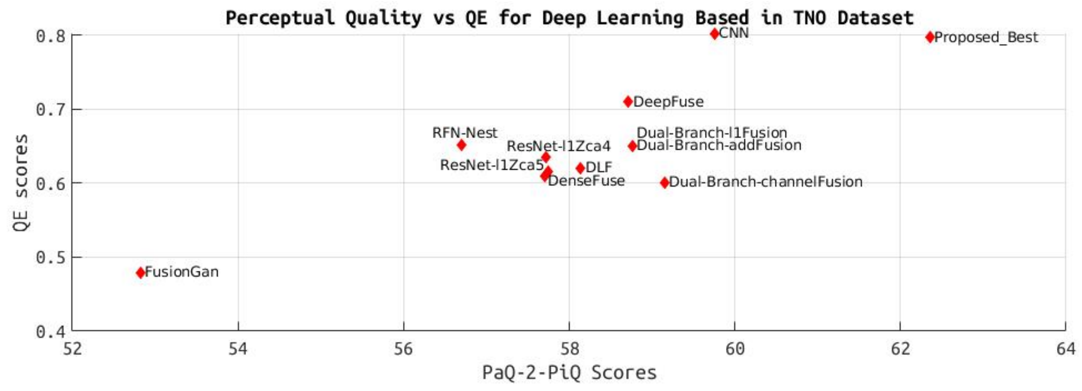
Furthermore, we tried to predict the mean opinion score,i.e. perceptual quality, using PaQ-2-PiQ [42] deep learning model. In Figure 5.5, average perceptual quality scores of all deep learning-based methods are compared. Our Proposed-Best method achieves the best average score in TNO [2] dataset among all deep learning-based methods. Moreover, In Figure 5.6, perceptual quality scores are reviewed with respect to Q_E, Q_W, and Q_Y scores to show our proposed method is reliable in both perceptual and quantitative scores. Our proposed model stays at the best possible region in these graphs, that is upper right corner. Even if the performance metric score of our proposed method has worse results than other methods, the comparison folder shows the difference clearly.

Table 5.12: Part 1 of other performance metrics that is not mentioned in the Table 5.11. Top three scores are highlighted with green, blue and red respectively for each metric. Results are obtained in TNO dataset.

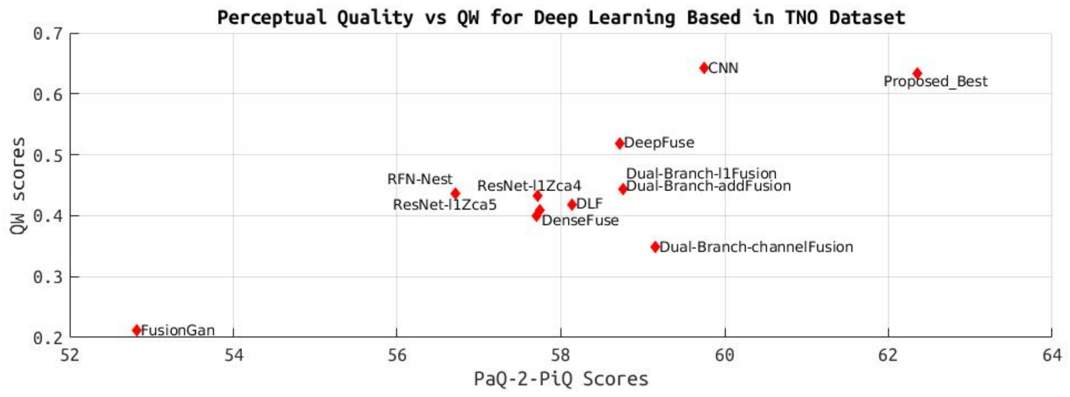
Quality Metrics / Method Names	AG	CrossEN	EI	EN	MI	PSNR	RMSE
CNN [21]	4.107	1.037	41.766	7.078	1.645	58.731	0.091
DeepFuse [25]	3.513	1.815	34.737	6.699	1.549	58.717	0.092
DenseFuse [24]	2.353	1.428	23.306	6.174	1.480	59.471	0.077
DLF [22]	2.426	1.559	24.005	6.183	1.459	59.471	0.077
Dual-Branch-addFusion [28]	2.471	1.521	25.079	6.332	1.505	59.426	0.077
Dual-Branch-channelFusion [28]	4.303	1.668	41.341	6.408	1.651	59.061	0.087
Dual-Branch-l1Fusion [28]	2.471	1.521	25.079	6.332	1.505	59.426	0.077
FusionGan [26]	2.205	2.633	22.148	6.363	1.559	57.871	0.115
Proposed-Best	3.586	1.606	35.986	6.414	1.291	59.363	0.079
ResNet-l1Zca4 [23]	2.425	1.488	24.057	6.235	1.349	59.456	0.077
ResNet-l1Zca5 [23]	2.372	1.495	23.499	6.195	1.403	59.469	0.077
RFN-Nest [27]	2.734	1.733	29.147	6.841	1.405	58.501	0.098

Table 5.13: Part 2 of other performance metrics that is not mentioned in the Table 5.11. Top three scores are highlighted with green, blue and red respectively for each metric. Results are obtained in TNO dataset.

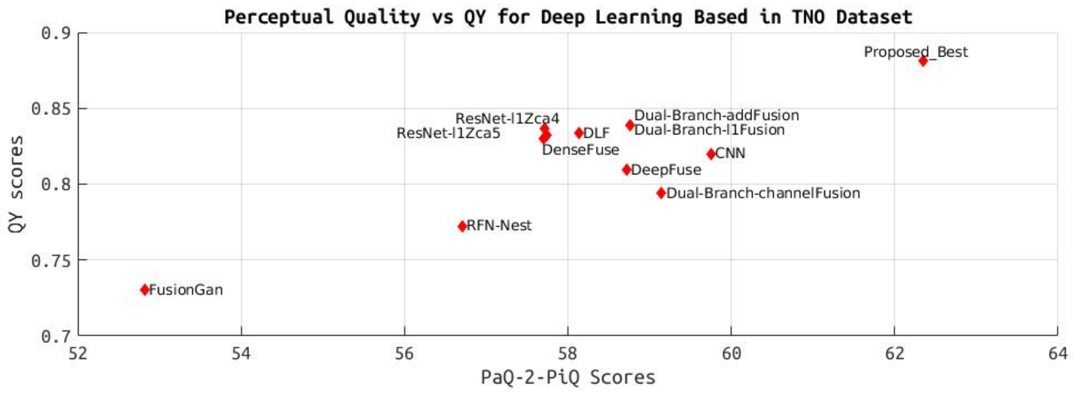
Quality Metrics / Method Names	SF	SSIM	SD	Q_{CB}	Q_{CV}	$Q^{AB/F}$	$L^{AB/F}$	$N^{AB/F}$
CNN [21]	10.705	1.395	45.496	0.556	360.765	0.559	0.100	0.371
DeepFuse [25]	8.918	1.462	33.653	0.506	484.688	0.506	0.146	0.233
DenseFuse [24]	6.041	1.562	22.546	0.496	471.968	0.406	0.232	0.001
DLF [22]	6.409	1.561	22.707	0.493	479.489	0.426	0.226	0.001
Dual-Branch-addFusion [28]	6.214	1.550	27.023	0.480	362.319	0.446	0.212	0.007
Dual-Branch-channelFusion [28]	13.146	1.407	32.200	0.486	840.068	0.422	0.190	0.172
Dual-Branch-l1Fusion [28]	6.214	1.550	27.023	0.480	362.319	0.446	0.212	0.007
FusionGan [26]	5.791	1.312	26.067	0.428	1061.593	0.280	0.331	0.104
Proposed-Best	9.827	1.501	26.968	0.522	408.588	0.594	0.129	0.146
ResNet-l1Zca4 [23]	6.328	1.560	23.742	0.502	447.528	0.430	0.225	0.001
ResNet-l1Zca5 [23]	6.135	1.561	22.940	0.496	461.457	0.412	0.230	0.001
RFN-Nest [27]	6.127	1.403	35.270	0.508	534.248	0.425	0.201	0.140



(a) Perceptual quality vs Q_E score graph.



(b) Perceptual quality vs Q_W score graph.



(c) Perceptual quality vs Q_Y score graph.

Figure 5.6: Perceptual quality scores with respect to Q_E, Q_W and Q_Y scores. Upper right corner in the graphs is the best place for a method. Results are obtained in TNO dataset.

In addition, to make visual comparisons of the deep learning-based methods, we shared some fused image samples shown in Figure 5.7. The results can be viewed better with digital zoom. There are small detail differences in the methods. For example, In the Jeep sequence which is the 4th column of the Figure 5.7, CNN and our proposed method results are sharper and details more clear. However, in the smoke sequence which is the 5th sequence, CNN shows poor performance. The soldier in the smoke is not distinguishable. In our proposed method, both smoke and soldier are seen.

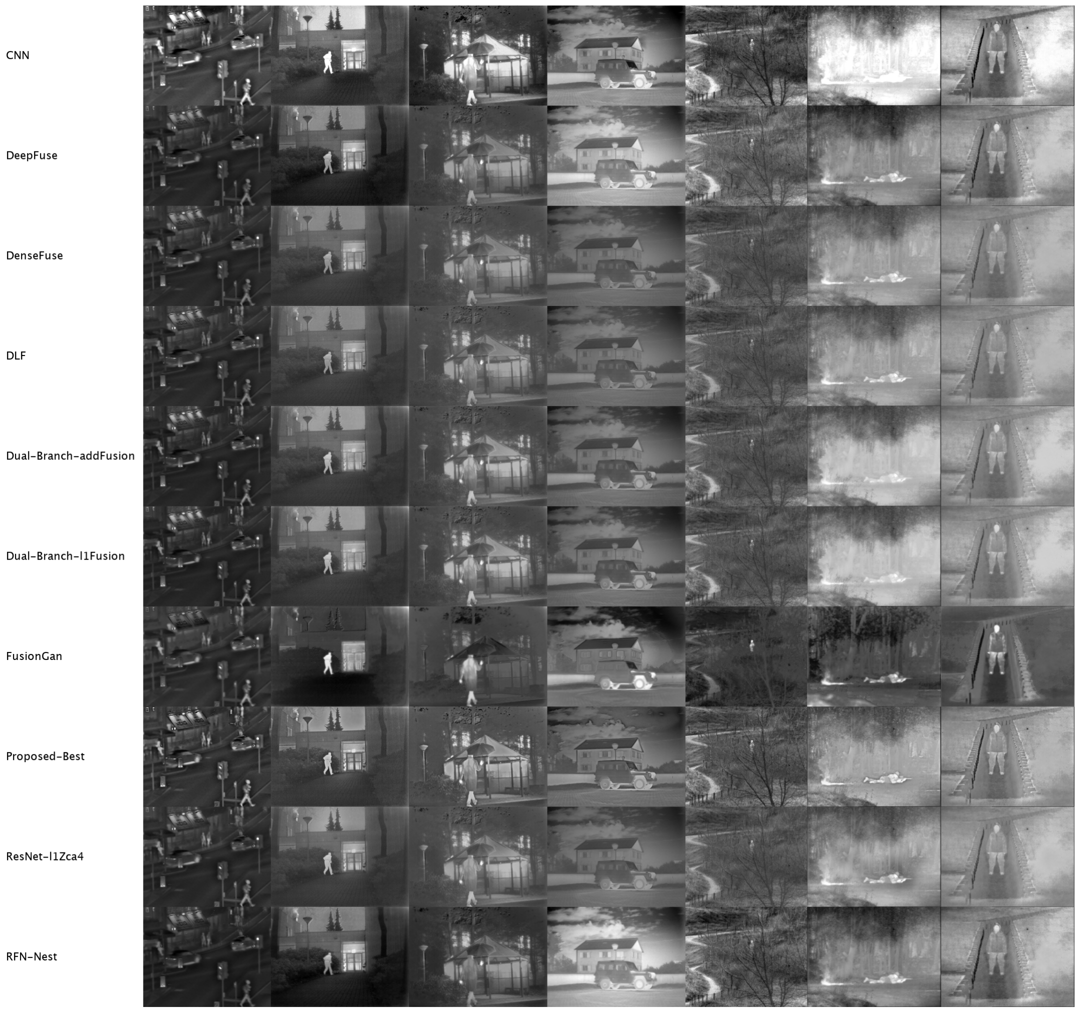


Figure 5.7: Some examples of the fused images of the deep learning based methos on TNO dataset.

5.4.1.2 VIFB Dataset Comparisons

In VIFB [3] dataset, some visible spectrum images have color channels. Thus, we can compare the color recovery performance of the metrics. As mentioned in the above chapter, we started to compare deep learning-based methods with more reliable performance metrics Q_E, Q, Q_W, and Q_Y. This gives better intuition about to overall picture. In Table 5.14, as in TNO [2] dataset comparisons, CNN and our proposed method give the best scores among all of the methods. However, Dual-Channel [28] and Rfn-Nest [27] methods only provide gray-scaled results, and this affects the overall scores but we can compare edge details and information not related to color.

Table 5.14: Reliable performance metrics with the MSE and $L1$ scores that show numerically distance to the input images. Top three scores are highlighted with green, blue and red respectively for each metric. Results are obtained in VIFB dataset.

Quality Metrics / Method Names	MSE	$L1$	Q_E	Q	Q_W	Q_Y
CNN [21]	0.176	0.015	0.836	0.841	0.709	0.849
DeepFuse [25]	0.148	0.076	0.494	0.595	0.297	0.724
DenseFuse [24]	0.112	0.002	0.644	0.705	0.456	0.810
DLF [22]	0.113	0.000	0.678	0.740	0.504	0.821
Dual-Branch-addFusion-grayscaled [28]	0.109	0.003	0.635	0.580	0.369	0.817
Dual-Branch-channelFusion-grayscaled [28]	0.143	0.004	0.611	0.545	0.358	0.775
Dual-Branch-l1Fusion-grayscaled [28]	0.123	0.005	0.685	0.624	0.430	0.829
FusionGan-grayscaled [26]	0.094	0.004	0.452	0.417	0.191	0.753
FusionGan-YChannelRGB [26]	0.220	0.140	0.313	0.379	0.128	0.681
Proposed Best	0.128	0.002	0.836	0.864	0.725	0.876
ResNet [23]	0.115	0.002	0.666	0.729	0.488	0.815
Rfn-Nest-grayscaled [27]	0.135	0.006	0.695	0.607	0.428	0.788

In Table 5.15 and 5.16, all performance metrics for deep learning-based methods can be seen. CNN and our proposed method take the best scores from most of the metrics. However, the results are not enough to compare methods. We need to show perceptual qualities. In Figure 5.12, perceptual quality scores of each method in the VIFB dataset are shown. Our proposed method achieves the best score. To review the methods, perceptual quality versus Q_E, Q_W, and Q_Y score figures are generated. In both perspectives, the success of our proposed method is seen. In Figure 5.8, the deep learning-based methods can be compared more accurately. The figure supports the comparison metrics give a good intuition about the success of the method, but further experiments are needed to justify final thoughts.

Table 5.15: Part 1 of other performance metrics that is not mentioned in the Table 5.14. Top three scores are highlighted with green, blue and red respectively for each metric. Results are obtained in VIFB [3] dataset.

Quality Metrics Method Names	AG	CrossEN	EI	EN	MI	PSNR	RMSE
CNN [21]	5.775	1.119	60.037	7.318	2.564	57.676	0.130
DeepFuse [25]	2.651	1.495	27.717	6.362	1.868	56.959	0.144
DenseFuse [24]	3.520	1.375	36.037	6.698	1.939	58.193	0.113
DLF [22]	3.797	1.444	38.422	6.722	1.943	58.190	0.114
Dual-Branch-addFusion-grayscaled [28]	3.195	1.388	33.537	6.678	2.043	58.164	0.114
Dual-Branch-channelFusion-grayscaled [28]	5.164	1.387	51.686	6.890	2.121	57.856	0.126
Dual-Branch-l1Fusion-grayscaled [28]	3.475	1.333	36.592	6.811	2.100	58.107	0.117
FusionGan-grayscaled [26]	2.950	2.442	30.613	6.370	1.592	57.344	0.133
FusionGan-YChannelRGB [26]	1.577	0.923	16.688	5.560	1.348	55.019	0.219
Proposed Best	5.246	1.594	54.778	6.942	1.732	58.005	0.119
ResNet [23]	3.647	1.391	37.110	6.732	1.899	58.184	0.114
Rfn-Nest-grayscaled [27]	3.650	1.556	39.314	7.147	1.992	57.903	0.120

Table 5.16: Part 2 of other performance metrics that is not mentioned in the Table 5.14. Top three scores are highlighted with green, blue and red respectively for each metric. Results are obtained in VIFB [3] dataset.

Quality Metrics Method Names	SF	SSIM	SD	Q_{CB}	Q_{CV}	$Q^{AB/F}$	$L^{AB/F}$	$N^{AB/F}$
CNN [21]	18.558	1.316	60.007	0.603	729.367	0.672	0.091	0.267
DeepFuse [25]	7.929	1.295	27.660	0.497	1063.807	0.290	0.234	0.031
DenseFuse [24]	10.850	1.389	34.182	0.429	932.323	0.403	0.169	0.072
DLF [22]	12.284	1.390	34.652	0.435	932.190	0.459	0.161	0.075
Dual-Branch-addFusion-grayscaled [28]	9.811	1.412	33.341	0.425	978.830	0.411	0.183	0.031
Dual-Branch-channelFusion-grayscaled [28]	18.212	1.342	41.935	0.445	1269.351	0.441	0.140	0.146
Dual-Branch-l1Fusion-grayscaled [28]	10.827	1.407	39.098	0.446	758.580	0.473	0.169	0.038
FusionGan-grayscaled [26]	9.261	1.267	26.864	0.342	1767.965	0.279	0.235	0.123
FusionGan-YChannelRGB [26]	4.788	1.127	15.222	0.341	1751.603	0.155	0.389	0.072
Proposed Best	16.737	1.360	40.405	0.497	768.403	0.599	0.117	0.229
ResNet [23]	11.538	1.389	34.877	0.435	896.176	0.437	0.166	0.071
Rfn-Nest-grayscaled [27]	9.987	1.343	45.276	0.471	997.649	0.447	0.165	0.122

Finally, some fused results from the VIFB dataset are shown in Figure 5.10. In the first column of the figure, a sequence called 'carWhite', CNN shows poor performance that our proposed method. When the sky region is reviewed, the clouds and the mast seen in the infrared image only cannot be recovered in the CNN method. Also, in the sixth column of the same figure, a sequence called 'kettle', the human that holds a bag under the building is more distinct in our proposed method. The details can be viewed in digital zoom.

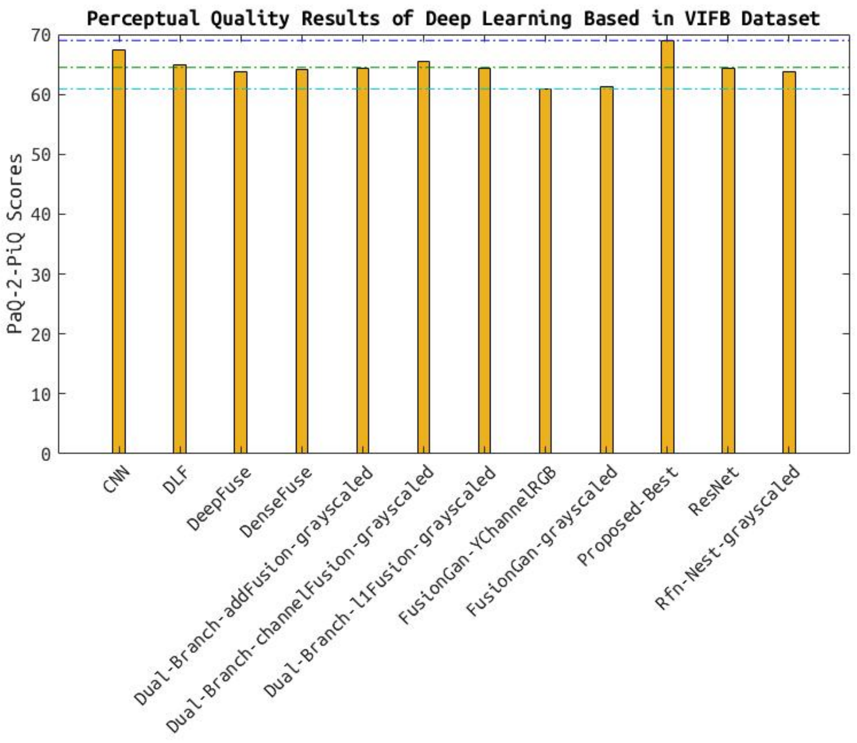
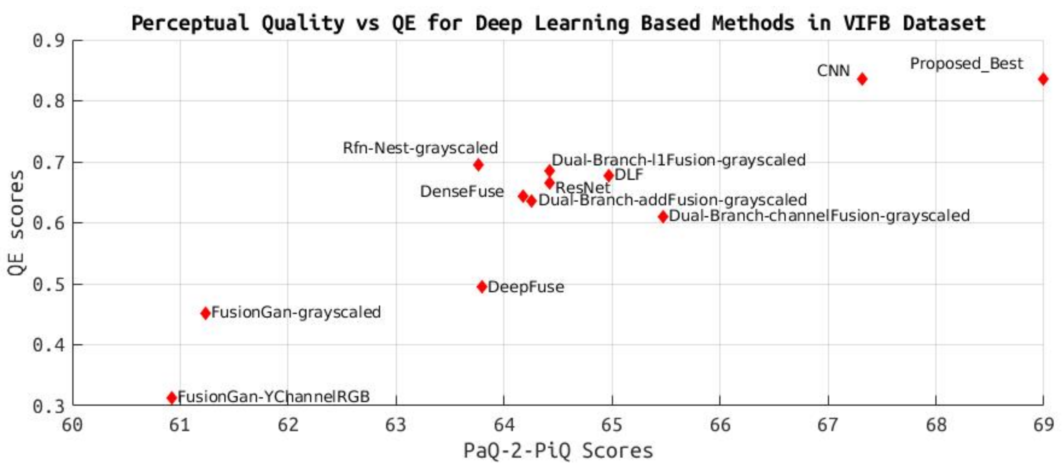
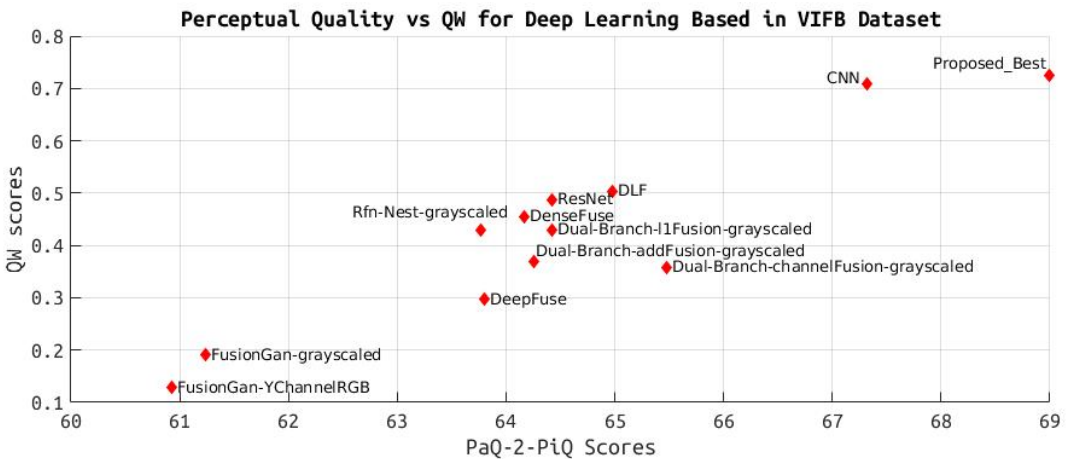


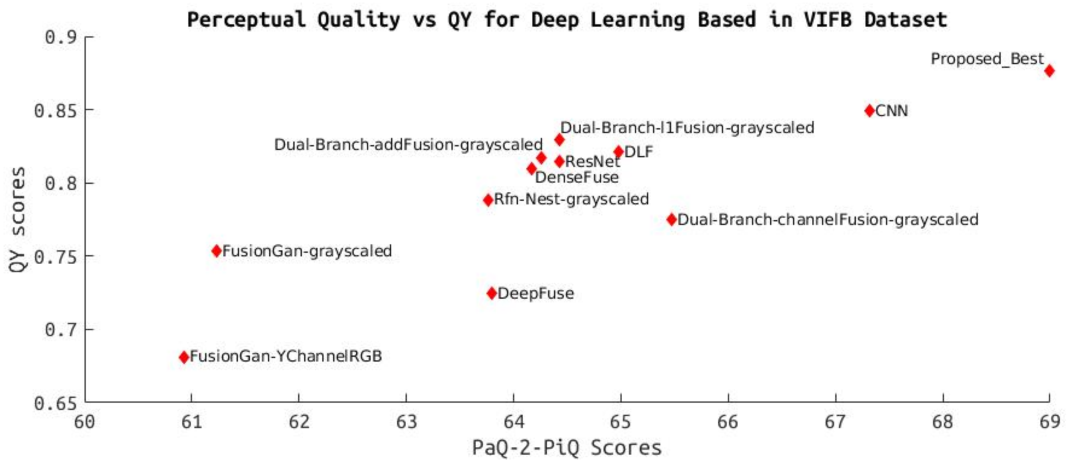
Figure 5.8: Average perceptual quality (PaQ-2-PiQ) scores for deep learning based methods. Results are obtained in VIFB dataset.



(a) Perceptual quality vs Q_E score graph.



(b) Perceptual quality vs Q_W score graph.



(c) Perceptual quality vs Q_Y score graph.

Figure 5.9: Perceptual quality scores with respect to Q_E, Q_W and Q_Y scores. Upper right corner in the graphs is the best place for a method. Results are obtained in VIFB dataset.

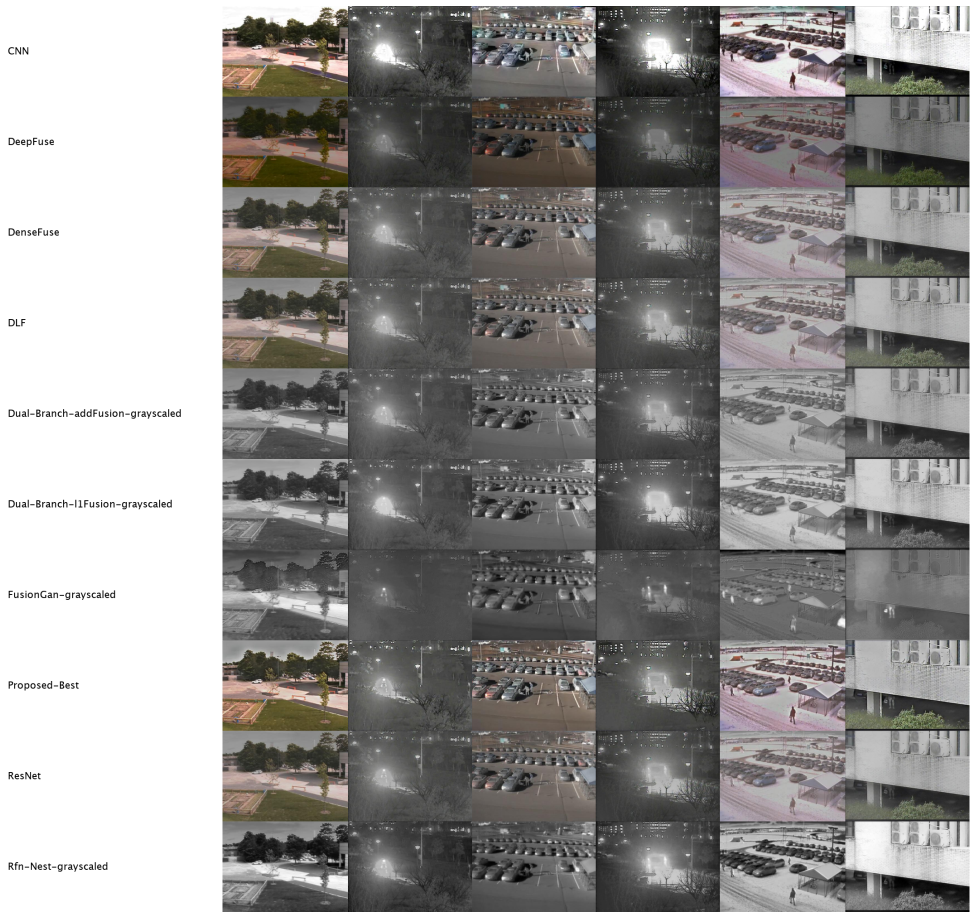


Figure 5.10: Some examples of the fused images of the deep learning based methos on VIFB dataset.

5.4.2 Overall Comparisons

Our proposed method is compared with all other existing state-of-the-art methods in both TNO and VIFB datasets. First of all, all methods' perceptual scores in TNO and VIFB datasets are given in Figure 5.11 and 5.12 respectively. To express benchmark results more clearly, in Figure 5.13 and 5.14, perceptual quality is compared with performance evaluation metrics Q_E, Q_W and Q_Y which are more stable for comparison. In these figure, we can see that even if our proposed method has lower scores on both perceptual and evaluation metrics, it stays in the upper right corner which beats existing methods. Furthermore, 'carWhite' and 'elecbike' image pairs in VIFB dataset and sequence 12 and 21 in TNO dataset is used to compare all methods visually. The fused image results are given in Figures 5.15, 5.16, 5.17 and, 5.18.

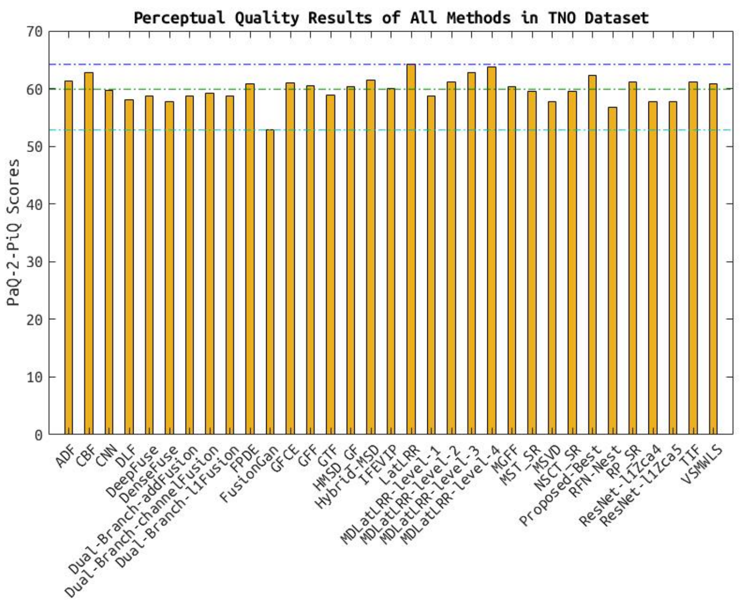


Figure 5.11: Average perceptual quality (PaQ-2-PiQ) scores for all methods. Results are obtained in TNO dataset.

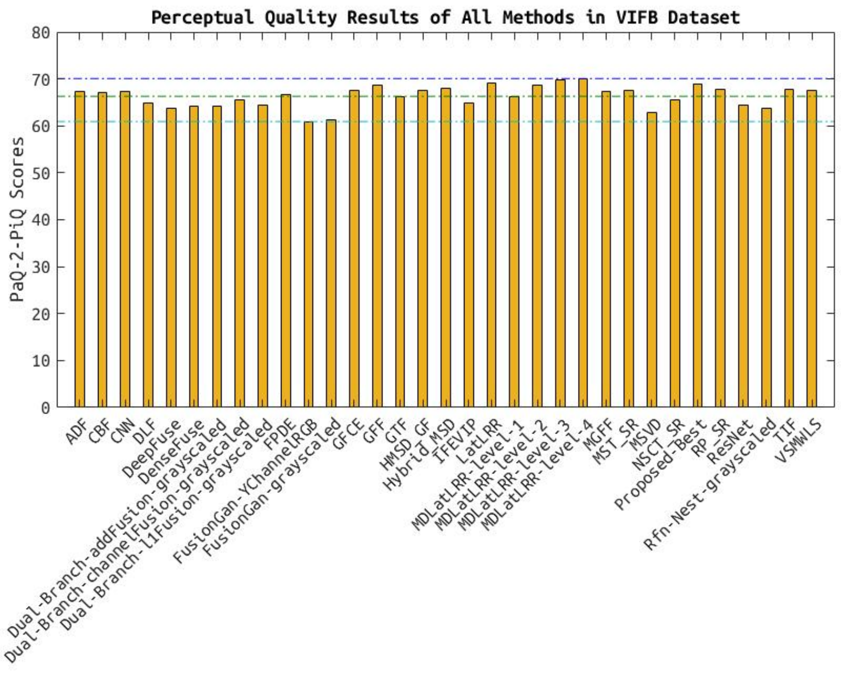
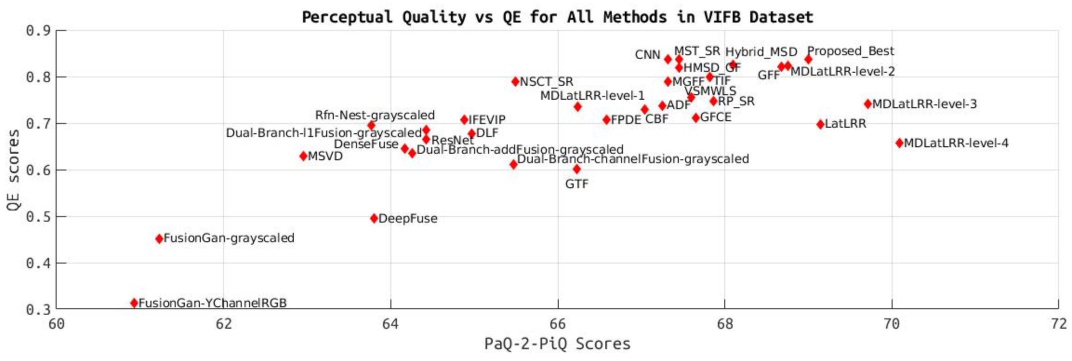
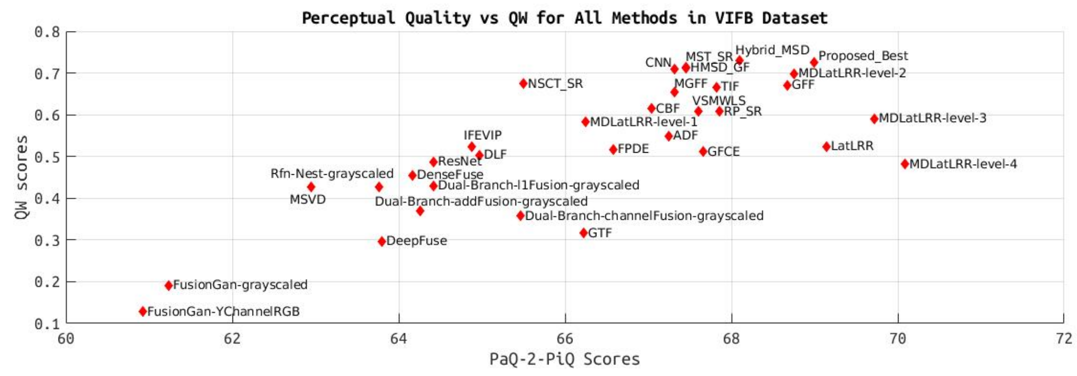


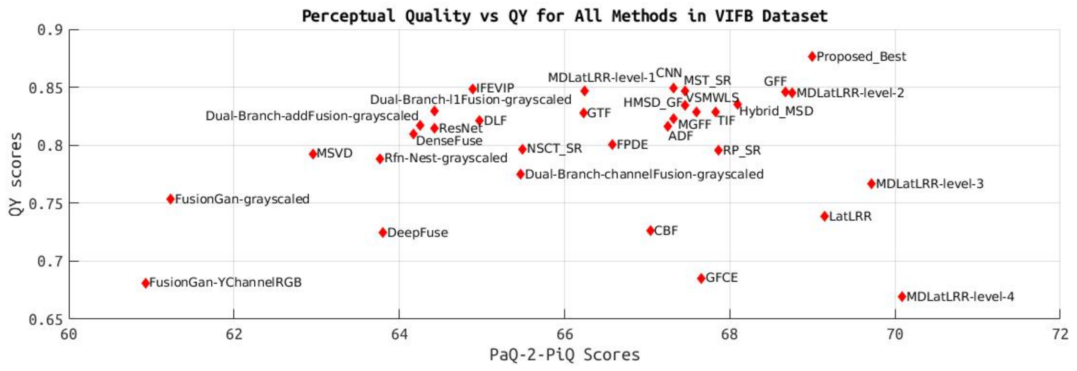
Figure 5.12: Average perceptual quality (PaQ-2-PiQ) scores for all methods. Results are obtained in VIFB dataset.



(a) Perceptual quality vs Q_E score graph.

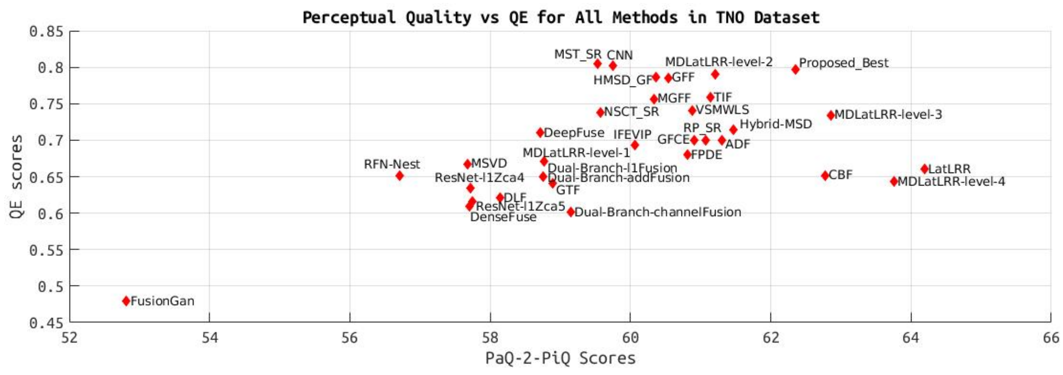


(b) Perceptual quality vs Q_W score graph.

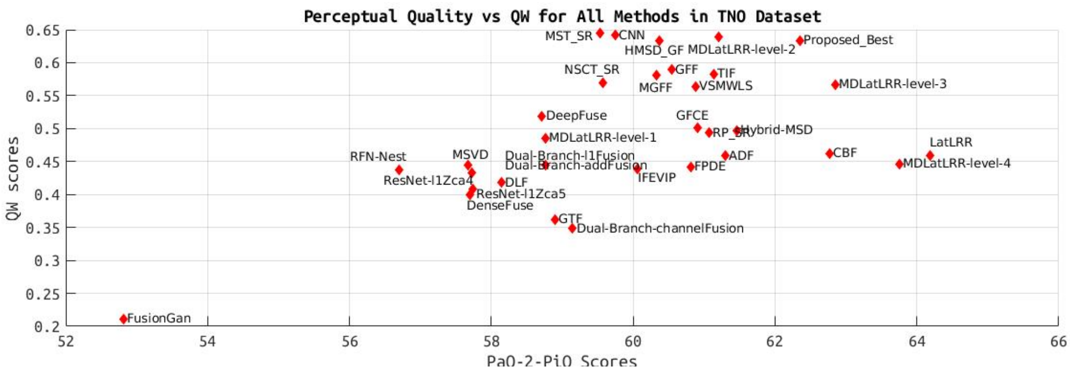


(c) Perceptual quality vs Q_Y score graph.

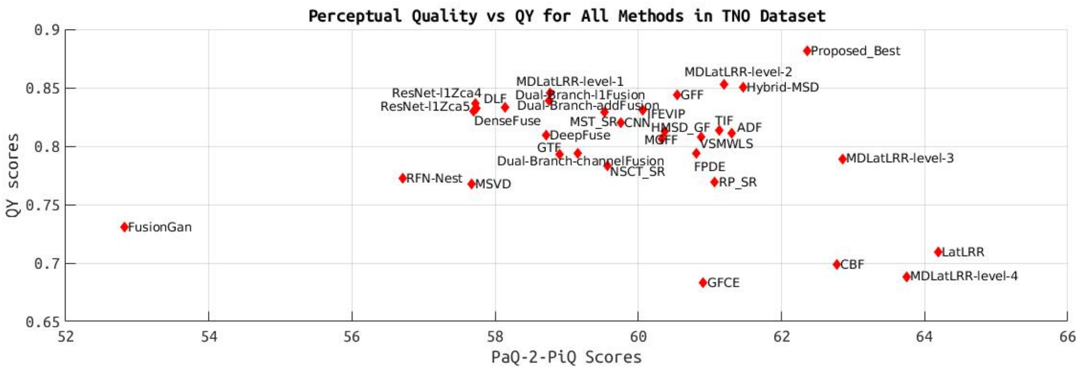
Figure 5.13: Perceptual quality scores with respect to Q_E, Q_W and Q_Y scores. Upper right corner in the graphs is the best place for a method. Results are obtained in VIFB dataset.



(a) Perceptual quality vs Q_E score graph.



(b) Perceptual quality vs Q_W score graph.



(c) Perceptual quality vs Q_Y score graph.

Figure 5.14: Perceptual quality scores with respect to Q_E, Q_W and Q_Y scores. Upper right corner in the graphs is the best place for a method. Results are obtained in TNO dataset.

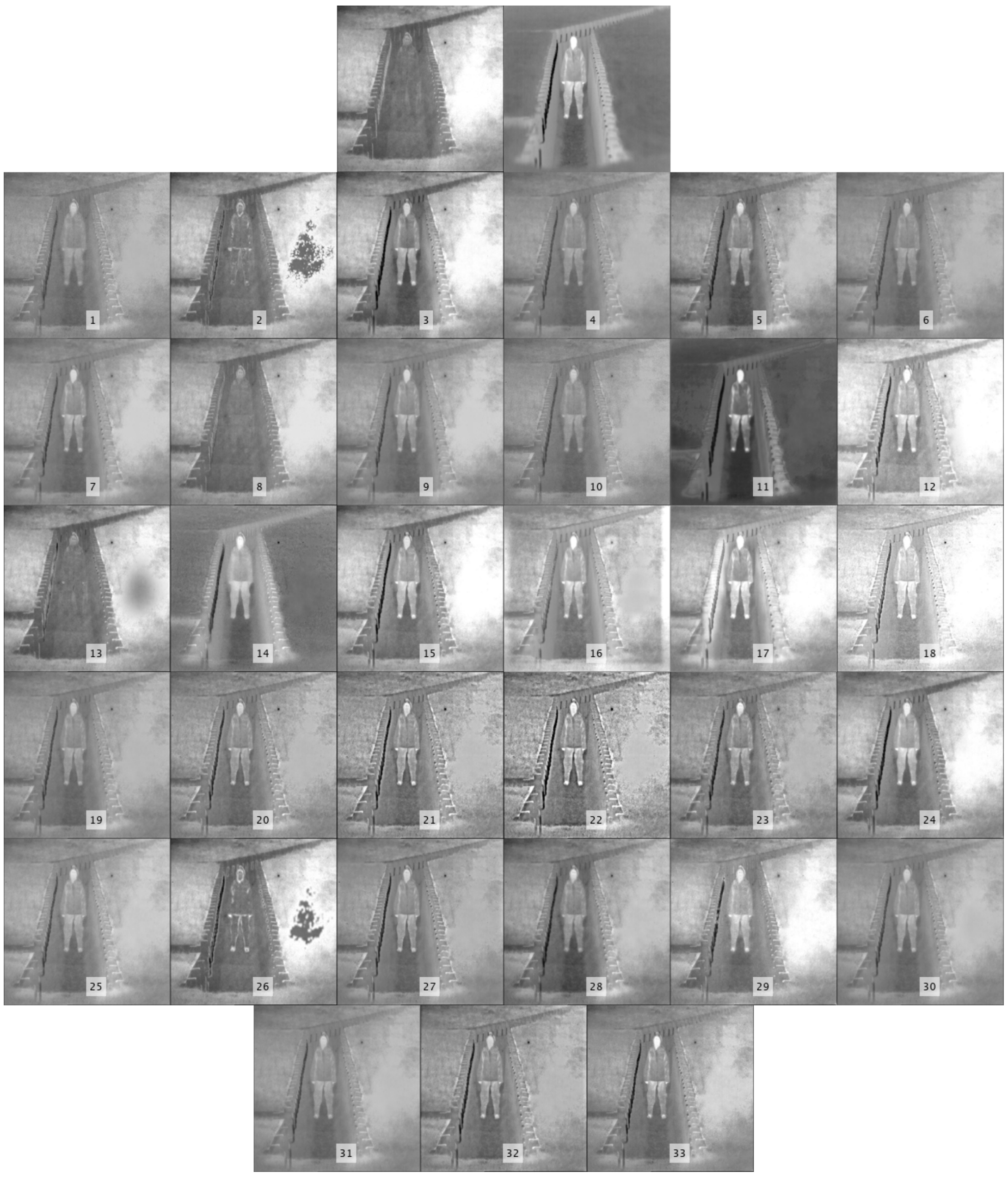


Figure 5.15: Sequence 21 in TNO dataset. First row is the visible and infrared images respectively. Each fused image has a number that can be checked from Table 5.10. The number 27 is our proposed method.

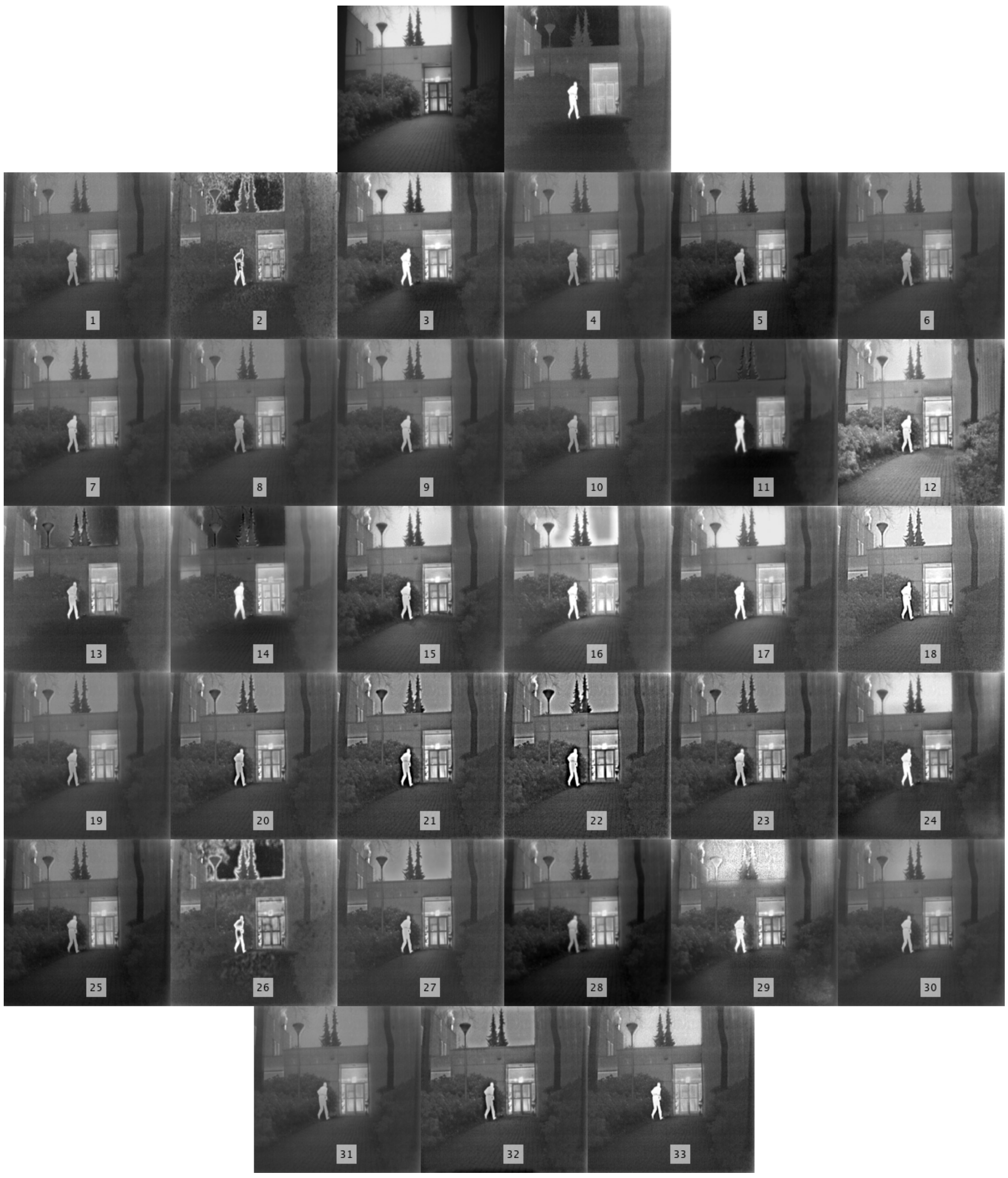


Figure 5.16: Sequence 12 in TNO dataset. First row is the visible and infrared images respectively. Each fused image has a number that can be checked from Table 5.10. The number 27 is our proposed method.

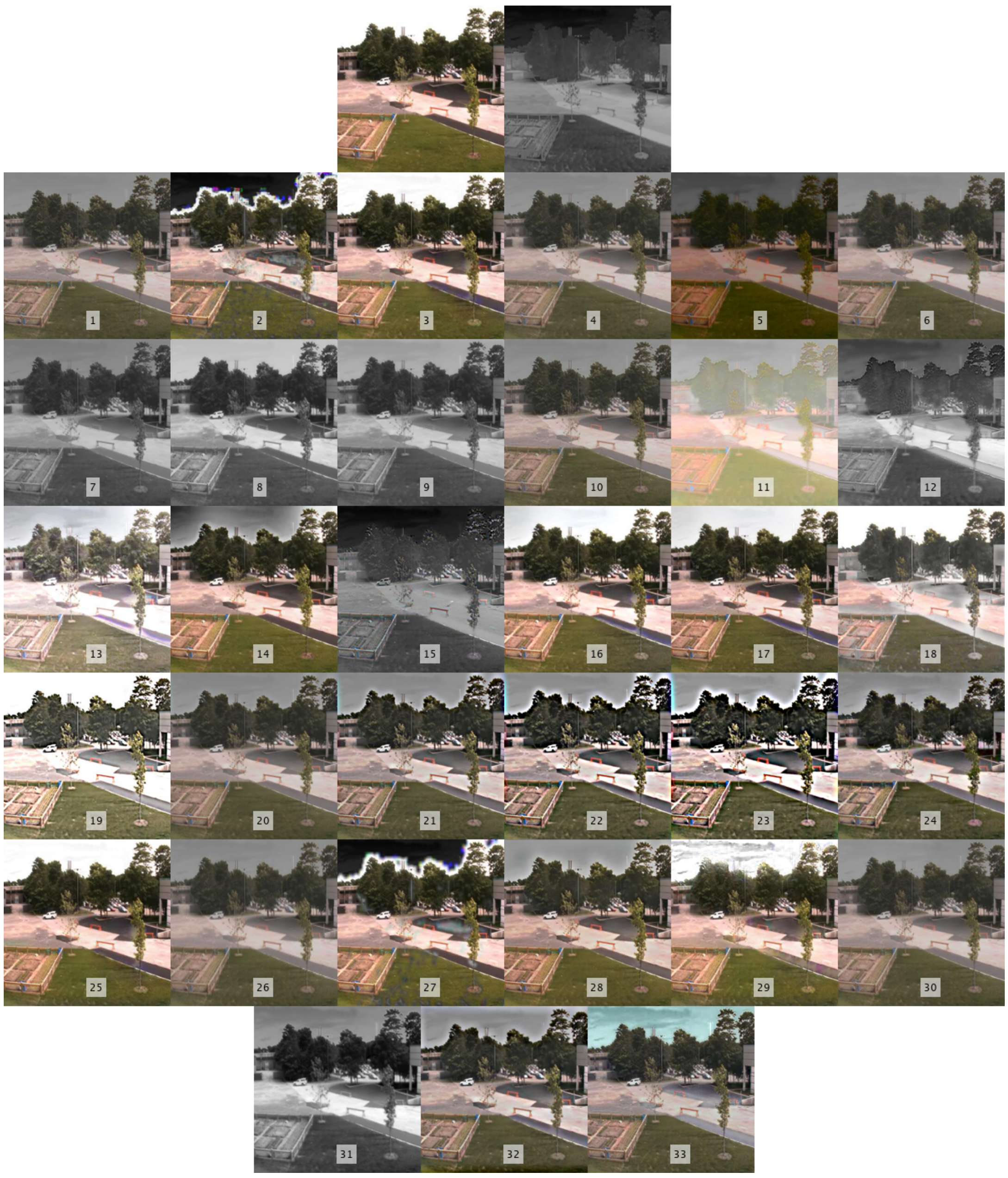


Figure 5.17: ’carWhite’ image pairs in VIFB dataset. First row is the visible and infrared images respectively. Each fused image has a number that can be checked from Table 5.10. The number 28 is our proposed method.

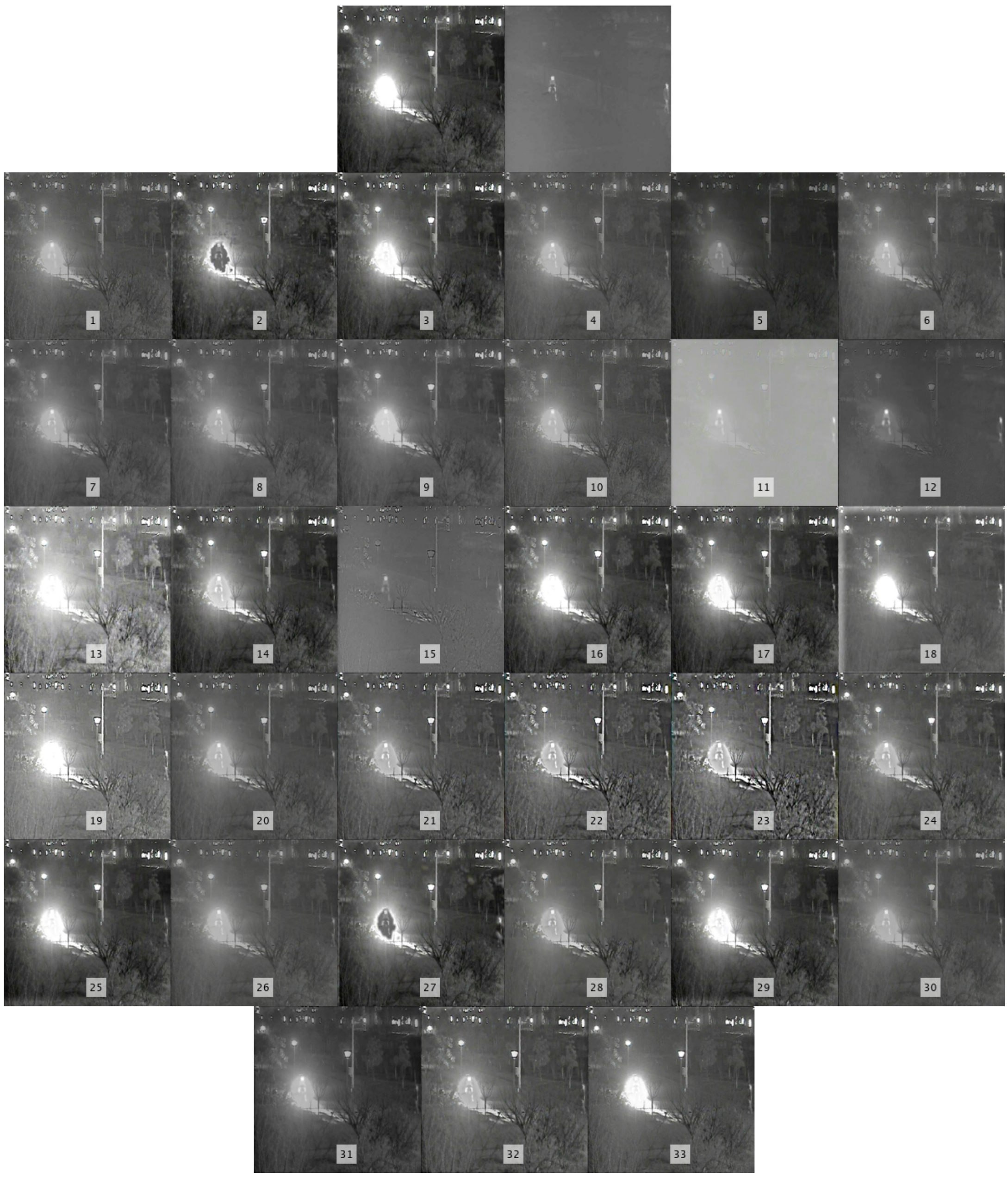


Figure 5.18: 'elecbike' image pairs in VIFB dataset. First row is the visible and infrared images respectively. Each fused image has a number that can be checked from Table 5.10. The number 28 is our proposed method.

CHAPTER 6

CONCLUSIONS

We present a novel method that is called Encoder-Decoder Network for Fusion. It is inspired from U-Net [1] and ResNet [5] architectures. It takes a visible three-channel image, i.e. RGB image, an infrared image as input and produces a three-channel output that merges the information on the input images. The proposed architecture is an end-to-end trainable neural network that is composed of convolution and pooling layers only. Most of the state-of-the-art methods propose a three separate step solution: feature extraction, fusion, and reconstruction. Each step is studied separately. However, our proposed method combines all three steps and optimizes all the steps together. That enables the creation of more strong connections between steps and the method becomes more stable. Moreover, the proposed method is trained with a custom loss function that takes into account the fusion quality and information transferred from input images. Thus, the fused image becomes of high quality both quantitatively and qualitatively. To show the success of the proposed method, it is compared with both deep learning-based methods and state-of-the-art methods.

As future work, the fusion result can be compared in further computer vision applications such as object detection, object tracking, and object recognition. Also, the perceived quality of the fused images could be rated by real human raters to see the overall opinion score of the fusion methods clearly. For real-time applications, the proposed method can be applied in embedded devices such as NVIDIA Jetson boards.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.

[2] A. Toet, “Tno image fusion dataset,” Apr 2014.

[3] X. Zhang, P. Ye, and G. Xiao, “Vifb: A visible and infrared image fusion benchmark,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015.

[6] D. P. Bavirisetti and R. Dhuli, “Fusion of infrared and visible sensor images based on anisotropic diffusion and karhunen-loeve transform,” *IEEE Sensors Journal*, vol. 16, no. 1, pp. 203–209, 2016.

[7] B. S. Kumar, “Image fusion based on pixel significance using cross bilateral filter,” *Signal, image and video processing*, vol. 9, no. 5, pp. 1193–1204, 2015.

[8] Z. Zhou, B. Wang, S. Li, and M. Dong, “Perceptual fusion of infrared and visible images through a hybrid multi-scale decomposition with gaussian and bilateral filters,” *Information Fusion*, vol. 30, pp. 15–26, 2016.

[9] Z. Zhou, M. Dong, X. Xie, and Z. Gao, “Fusion of infrared and visible images for night-vision context enhancement,” *Appl. Opt.*, vol. 55, pp. 6480–6490, Aug 2016.

[10] V. Naidu, “Image fusion technique using multi-resolution singular value decomposition,” *Defence Science Journal*, vol. 61, no. 5, p. 479, 2011.

[11] D. P. Bavirisetti, G. Xiao, J. Zhao, R. Dhuli, and G. Liu, “Multi-scale guided image and video fusion: A fast and efficient approach,” *Circuits, Systems, and Signal Processing*, vol. 38, no. 12, pp. 5576–5605, 2019.

[12] S. Li, X. Kang, and J. Hu, “Image fusion with guided filtering,” *IEEE Transactions on Image Processing*, vol. 22, no. 7, pp. 2864–2875, 2013.

[13] D. P. Bavirisetti and R. Dhuli, “Two-scale image fusion of visible and infrared images using saliency detection,” *Infrared Physics & Technology*, vol. 76, pp. 52–64, 2016.

[14] B. Yang and S. Li, “Multifocus image fusion and restoration with sparse representation,” *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 4, pp. 884–892, 2010.

[15] B. Yang and S. Li, “Pixel-level image fusion with simultaneous orthogonal matching pursuit,” *Information Fusion*, vol. 13, no. 1, pp. 10–19, 2012.

[16] S. Li, H. Yin, and L. Fang, “Group-sparse representation with dictionary learning for medical image denoising and fusion,” *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 12, pp. 3450–3459, 2012.

[17] H. Li, X.-J. Wu, and J. Kittler, “Mdlatlrr: A novel decomposition method for infrared and visible image fusion,” *IEEE Transactions on Image Processing*, vol. 29, p. 4733–4746, 2020.

[18] D. P. Bavirisetti, G. Xiao, and G. Liu, “Multi-sensor image fusion based on fourth order partial differential equations,” in *2017 20th International Conference on Information Fusion (Fusion)*, pp. 1–9, 2017.

[19] Y. Liu, S. Liu, and Z. Wang, “A general framework for image fusion based on multi-scale transform and sparse representation,” *Information Fusion*, vol. 24, pp. 147–164, 2015.

[20] J. Ma, Z. Zhou, B. Wang, and H. Zong, “Infrared and visible image fusion based on visual saliency map and weighted least square optimization,” *Infrared Physics & Technology*, vol. 82, pp. 8–17, 2017.

[21] Y. Liu, X. Chen, J. Cheng, H. Peng, and Z. Wang, “Infrared and visible image fusion with convolutional neural networks,” *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 16, no. 03, p. 1850018, 2018.

[22] H. Li, X.-J. Wu, and J. Kittler, “Infrared and visible image fusion using a deep learning framework,” in *2018 24th international conference on pattern recognition (ICPR)*, pp. 2705–2710, IEEE, 2018.

[23] H. Li, X.-j. Wu, and T. S. Durrani, “Infrared and visible image fusion with resnet and zero-phase component analysis,” *Infrared Physics & Technology*, vol. 102, p. 103039, Nov 2019.

[24] H. Li and X.-J. Wu, “Densefuse: A fusion approach to infrared and visible images,” *IEEE Transactions on Image Processing*, vol. 28, p. 2614–2623, May 2019.

[25] K. R. Prabhakar, V. S. Srikar, and R. V. Babu, “Deepfuse: A deep unsupervised approach for exposure fusion with extreme exposure image pairs,” 2017.

[26] J. Ma, W. Yu, P. Liang, C. Li, and J. Jiang, “Fusiongan: A generative adversarial network for infrared and visible image fusion,” *Information Fusion*, vol. 48, pp. 11–26, 2019.

[27] H. Li, X.-J. Wu, and J. Kittler, “Rfn-nest: An end-to-end residual fusion network for infrared and visible images,” *Information Fusion*, vol. 73, p. 72–86, Sep 2021.

[28] Y. Fu and X.-J. Wu, “A dual-branch network for infrared and visible image fusion,” 2021.

[29] L. Liu, M. Chen, M. Xu, and X. Li, “Two-stream network for infrared and visible images fusion,” *Neurocomputing*, vol. 460, pp. 50–58, 2021.

[30] J. Ma, C. Chen, C. Li, and J. Huang, “Infrared and visible image fusion via gradient transfer and total variation minimization,” *Information Fusion*, vol. 31, pp. 100–109, 2016.

[31] Y. Zhang, L. Zhang, X. Bai, and L. Zhang, “Infrared and visual image fusion through infrared feature extraction and visual information preservation,” *Infrared Physics & Technology*, vol. 83, pp. 227–237, 2017.

[32] P. Khurana, “Swir resolution+ unlocks potential.” `https://prvnk10.medium.com/the-convolution-operation-48d72a382f5a`, 2020. Accessed: 2021-10-01.

[33] D. Frossard, “Vgg in tensorflow.” `https://www.cs.toronto.edu/~frossard/post/vgg16/`, 2016. Accessed: 2021-05-06.

[34] J. Ma, Y. Ma, and C. Li, “Infrared and visible image fusion methods and applications: A survey,” *Information Fusion*, vol. 45, pp. 153–178, 2019.

[35] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, “Algorithms for simultaneous sparse approximation. part i: Greedy pursuit,” *Signal Processing*, vol. 86, no. 3, pp. 572–588, 2006. Sparse Approximations in Signal and Image Processing.

[36] J. Wang, J. Peng, X. Feng, G. He, and J. Fan, “Fusion method for infrared and visible images by using non-negative sparse representation,” *Infrared Physics & Technology*, vol. 67, pp. 477–489, 2014.

[37] Y. Liu, X. Chen, H. Peng, and Z. Wang, “Multi-focus image fusion with a deep convolutional neural network,” *Information Fusion*, vol. 36, pp. 191–207, 2017.

[38] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, “Unet++: A nested u-net architecture for medical image segmentation,” 2018.

[39] P. Singh, “How to ___ variational autoencoder ?.” `https://spraphul.github.io/blog/VAE`, 2020. Accessed: 2021-05-07.

[40] M. NeuralNet, “Calculating the output size of convolutions and transpose convolutions.” `http://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html`, 2020. Accessed: 2021-05-06.

[41] P. Ruiz, "Understanding and visualizing resnets." https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8, 2020. Accessed: 2021-05-13.

[42] Z. Ying, H. Niu, P. Gupta, D. Mahajan, D. Ghadiyaram, and A. Bovik, "From patches to pictures (paq-2-piq): Mapping the perceptual space of picture quality," 2019.

[43] "Biological neuron model." https://en.wikipedia.org/wiki/Biological_neuron_model, 2021. Accessed: 2021-04-30.

[44] "Single layer perceptron in tensorflow." https://www.javatpoint.com/single-layer-perceptron-in-tensorflow, 2021. Accessed: 2021-04-30.

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[46] J. Hoffman, "Cramnet: Layer-wise deep neural network compression with knowledge transfer from a teacher network," 2019.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[48] C. Gomes, "8 - designing military uniforms with high-tech materials," in *Military Textiles* (E. Wilusz, ed.), Woodhead Publishing Series in Textiles, pp. 183–203, Woodhead Publishing, 2008.

[49] "Swir resolution+ unlocks potential." https://www.geoimage.com.au/SWIR%20Series/resolution, 2018. Accessed: 2021-04-30.

[50] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, 1983.

[51] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[52] M. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2091–2106, 2005.

[53] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graph.*, vol. 27, p. 1–10, Aug. 2008.

[54] E. C and D. Donoho, "Curvelets - a surprisingly effective nonadaptive representation for objects with edges," *Curves and Surfaces*, 04 2000.

[55] R. Dony *et al.*, "Karhunen-loeve transform," *The transform and data compression handbook*, vol. 1, pp. 1–34, 2001.

[56] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pp. 839–846, 1998.

[57] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 6, pp. 1397–1409, 2012.

[58] R. Kakarala and P. O. Ogunbona, "Signal analysis using a multiresolution form of the singular value decomposition," *IEEE Transactions on Image processing*, vol. 10, no. 5, pp. 724–735, 2001.

[59] Q. Zhang, Y. Liu, R. S. Blum, J. Han, and D. Tao, "Sparse representation based multi-sensor image fusion for multi-focus and multi-modality images: A review," *Information Fusion*, vol. 40, pp. 57–75, 2018.

[60] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[61] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding.," *Journal of Machine Learning Research*, vol. 11, no. 1, 2010.

[62] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning:*

data mining, inference, and prediction. Springer Science & Business Media, 2009.

[63] G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," in *2011 International Conference on Computer Vision*, pp. 1615–1622, 2011.

[64] Y.-L. You and M. Kaveh, "Fourth-order partial differential equations for noise removal," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 9 10, pp. 1723–30, 2000.

[65] A. Toet, "Image fusion by a ratio of low-pass pyramid," *Pattern Recognition Letters*, vol. 9, no. 4, pp. 245–253, 1989.

[66] V. S. Petrovic and C. S. Xydeas, "Gradient-based multiresolution image fusion," *IEEE Transactions on Image processing*, vol. 13, no. 2, pp. 228–237, 2004.

[67] H. Li, B. Manjunath, and S. Mitra, "Multisensor image fusion using the wavelet transform," *Graphical Models and Image Processing*, vol. 57, no. 3, pp. 235–245, 1995.

[68] M. Beaulieu, S. Foucher, and L. Gagnon, "Multi-spectral image resolution refinement using stationary wavelet transform," in *IGARSS 2003. 2003 IEEE International Geoscience and Remote Sensing Symposium. Proceedings (IEEE Cat. No. 03CH37477)*, vol. 6, pp. 4032–4034, IEEE, 2003.

[69] J. J. Lewis, R. J. O'Callaghan, S. G. Nikolov, D. R. Bull, and N. Canagarajah, "Pixel- and region-based image fusion with complex wavelets," *Information Fusion*, vol. 8, no. 2, pp. 119–130, 2007. Special Issue on Image Fusion: Advances in the State of the Art.

[70] F. Nencini, A. Garzelli, S. Baronti, and L. Alparone, "Remote sensing image fusion using the curvelet transform," *Information Fusion*, vol. 8, no. 2, pp. 143–156, 2007. Special Issue on Image Fusion: Advances in the State of the Art.

[71] Q. Zhang and B. long Guo, "Multifocus image fusion using the nonsubsampled contourlet transform," *Signal Processing*, vol. 89, no. 7, pp. 1334–1346, 2009.

[72] Q. Zhang, X. Shen, L. Xu, and J. Jia, “Rolling guidance filter,” in *European conference on computer vision*, pp. 815–830, Springer, 2014.

[73] Y. Zhai and M. Shah, “Visual attention detection in video sequences using spatiotemporal cues,” in *Proceedings of the 14th ACM International Conference on Multimedia*, MM ’06, (New York, NY, USA), p. 815–824, Association for Computing Machinery, 2006.

[74] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[75] A. Kessy, A. Lewin, and K. Strimmer, “Optimal whitening and decorrelation,” *The American Statistician*, vol. 72, p. 309–314, Jan 2018.

[76] K. Ma, K. Zeng, and Z. Wang, “Perceptual quality assessment for multi-exposure image fusion,” *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3345–3356, 2015.

[77] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014.

[78] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[79] Z. Tu, Y. Ma, Z. Li, C. Li, J. Xu, and Y. Liu, “Rgbt salient object detection: A large-scale dataset and benchmark,” *arXiv preprint arXiv:2007.03262*, 2020.

[80] X. Bai, Y. Zhang, F. Zhou, and B. Xue, “Quadtree-based multi-focus image fusion using a weighted focus-measure,” *Information Fusion*, vol. 22, pp. 105–118, 2015.

[81] L. Zhang, “In situ image segmentation using the convexity of illumination distribution of the light sources,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 10, pp. 1786–1799, 2008.

[82] J. W. Roberts, J. A. Van Aardt, and F. B. Ahmed, "Assessment of image fusion procedures using entropy, image quality, and multispectral classification," *Journal of Applied Remote Sensing*, vol. 2, no. 1, p. 023522, 2008.

[83] G. Qu, D. Zhang, and P. Yan, "Information measure for performance of image fusion," *Electronics letters*, vol. 38, no. 7, pp. 313–315, 2002.

[84] M. B. A. Haghighat, A. Aghagolzadeh, and H. Seyedarabi, "A non-reference image fusion metric based on mutual information of image features," *Computers & Electrical Engineering*, vol. 37, no. 5, pp. 744–756, 2011.

[85] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE signal processing letters*, vol. 9, no. 3, pp. 81–84, 2002.

[86] G. Piella and H. Heijmans, "A new quality metric for image fusion," in *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, vol. 3, pp. III–173, IEEE, 2003.

[87] V. Petrovic and C. Xydeas, "Objective image fusion performance characterisation," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1866–1871, IEEE, 2005.

[88] Y. Chen and R. S. Blum, "A new automated quality assessment algorithm for image fusion," *Image and vision computing*, vol. 27, no. 10, pp. 1421–1432, 2009.

[89] H. Chen and P. K. Varshney, "A human perception inspired quality metric for image fusion based on regional information," *Information fusion*, vol. 8, no. 2, pp. 193–207, 2007.

[90] C. Yang, J.-Q. Zhang, X.-R. Wang, and X. Liu, "A novel similarity based quality metric for image fusion," *Information Fusion*, vol. 9, no. 2, pp. 156–160, 2008.

[91] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[92] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi, "Deep roots: Improving cnn efficiency with hierarchical filter groups," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1231–1240, 2017.

[93] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[94] A. Thevenot, "Implement canny edge detection from scratch with pytorch." `https://towardsdatascience.com/implement-canny-edge-detection-from-scratch-with-pytorch-a1cccfa58bed`, 2020. Accessed: 2021-05-17.

[95] "Free flir thermal dataset for algorithm training." =https://www.flir.com/oem/adas/adas-dataset-form/. Accessed: 2021-06-27.

[96] S. Hwang, J. Park, N. Kim, Y. Choi, and I. S. Kweon, "Multispectral pedestrian detection: Benchmark dataset and baselines," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[97] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[98] M. Minsky and S. Papert, *Perceptrons: An introduction to computational geometry*. MIT Press, 1969.

[99] J. McGonagle, G. Shaikouski, C. Williams, A. Hsu, J. Khim, and A. Miller, "Backpropagation." `https://brilliant.org/wiki/backpropagation/`, 2021. Accessed: 2021-04-30.

[100] C. Hansen, "Neural networks: Feedforward and backpropagation explained & optimization." `https://mlfromscratch.com/neural-networks-explained/#/`, 2019. Accessed: 2021-05-2.

[101] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[102] A. Oppermann, "Regularization in deep learning — l1, l2, and dropout." `https://towardsdatascience.com/regularization-in-`

`deep-learning-l1-l2-and-dropout-377e75acc036`, 2020. Accessed: 2021-05-06.

[103] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

APPENDIX A

BACKGROUND INFORMATION ON DEEP LEARNING

A.1 Introduction

Humans have desired to create machines that are programmed to think and mimic actions of human intelligence. This is called Artificial Intelligence (AI). Our aim is to automate routines, learn, understand and make a diagnosis in various topics that humans are good at. For instance, AI may tracks objects in security cameras, scans camera footage to find anomalies in real-time. It supports humans in various ways. Development on both hardware and software technologies enables to speed up studies on AI. Computers are straightforward. That is they should be programmed by following formal and mathematical rules. However, there exists some problems that are hard to describe formally but they are easy for humans.

Before AI solves problems, they need to learn relevant representations of data. Each information given in data is a feature. However, how features are extracted is important and it affects the performance of the AI system. To decide whether a person has cancer or not, relevant information like the presence of some type of scars on their MRI result or blood test results is needed. It can be challenging to describe a feature related to a task. For instance, to detect cats in pictures, we can define some features like cats have tails, paws, etc. However, we do not know exactly how the tail is represented in terms of pixel values. It is hard to describe because the tail can change shape and color. Even if these are stable, the picture may be taken in different light conditions, some pictures may include shadows, and so on.

AI systems need to acquire their knowledge from raw data. They should figure out patterns from data. This is known as Machine Learning. This enabled computers to solve problems that require making decisions relative to knowledge of the real world. A simple machine learning algorithm called logistic regression can predict house prices in a region by considering some features like overall quality, total living area, garage area, etc. AI predicts house prices with given some knowledge that called features. Obtaining these simpler feature representations can be solved via Deep Learning that tries to represent simpler and more efficient features from input raw data. We said that Deep Learning because a solution to problems consists of simpler blocks that build complicated blocks and this can be shown as a graph and it becomes deep. A simple deep learning algorithm Multilayer Perceptron can classify raw visible images whether includes cat or dog.

A.2 History

Deep learning can be thought that is a relatively new technology, but its foundations date back to the 1940s. Throughout history, it has had different names like cybernetics, connectionism, artificial neural network. Lastly, the deep learning (DL) name becomes popular. Even though the names were changed, the idea behind the name stays the same. DL tries to mimic biological neurons to compute and store information related to any task desired. Like humans, DL should have a training phase to master its application. The training phase is the most challenging part of DL because it needs strong hardware and software with a large amount of training data. Improvements in computational power enable DL to learn simple tasks like recognizing handwritten numbers. Over time the overall accuracy for applied applications has been increased and this leads to a revolution that transformed the AI industry according to researchers.

A.3 Perceptron

The perceptron is an algorithm that tries to solve binary classification problems in a supervised way. It is composed of three pieces: input layer, weights, and activation function. Simply, it learns a function that maps input x to an output f(x) which is a binary value. It is inspired by the biological neuron that includes multiple connections (inputs of the perceptron) in synapses and each synapse has a different connection amplitude (weights in the perceptron) and output response is generated with combining connections to decide whether a spike, i.e. neuronal action potential, is generated or not (activation function in the perceptron). Figure A.1 shows complete biological neuron model.

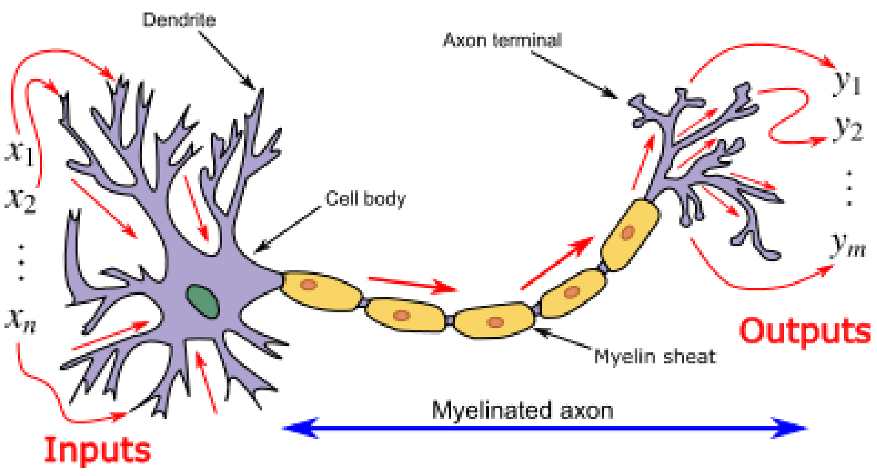


Figure A.1: Biological Neuron Model taken from Wikipedia[43]

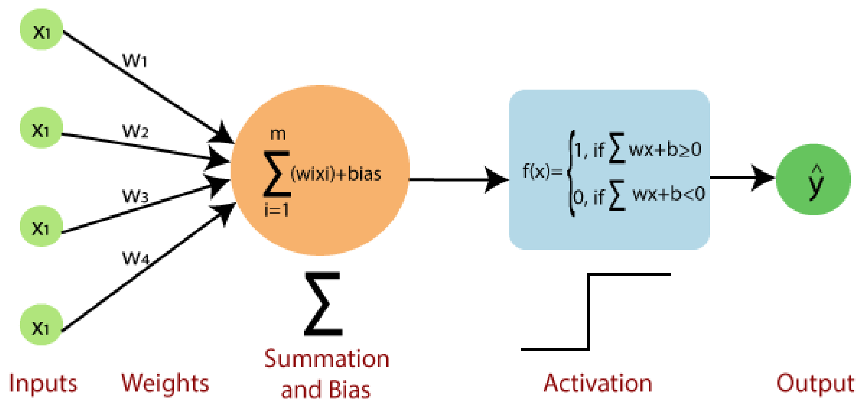


Figure A.2: Detailed explained parts of single layer perceptron[44]

The input layer is connected to the output neuron with a connection weight. Each input multiplies with corresponding weights and their total summation is sent to the activation function. The activation function is simply a thresholding function that outputs the binary result. Binary classification is completed with these steps as shown in Figure A.2

The perceptron learns to classify two separate classes with updating weights. There is only one layer to classify, so the perceptron is called a single-layer perceptron. Single-layer perceptrons are only capable of solving linearly-separable patterns. In 1969, it is proved in Perceptrons(book) [98] that single-layer perceptron can not learn XOR function which is not linearly separable.

Although it is achieved good results in binary classification for linearly-separable data, it could not be used efficiently for multi-class classification. Until a multilayer perceptron or feed-forward network that consists of multiple layers is invented, neural network research stagnates for years.

After the invention of the back-propagation algorithm, explained with great detail in section A.4, it made it possible to update network parameters, i.e. connection weights, efficiently. This enabled to use of multiple layers in addition to the input and output layer of a perceptron. These additional layers are called hidden layers. In hidden layers, each neuron has an activation function as an output layer. If a linear activation function is used in all neurons, it can be shown that this model can be reduced to a single-layer perceptron model. Thus, In Multilayer Perceptrons (MLP), there are used some nonlinear activation functions. Sigmoid and hyperbolic tangent functions are commonly used nonlinear functions in MLPs. Sigmoid ranges from 0 to 1 whereas hyperbolic tangent ranges from -1 to 1. These functions are described in Eqn. A.1 and A.2.

$$sigmoid(x) = \frac{1}{1+\epsilon^{-x}} \tag{A.1}$$

$$hyperbolic\_tangent(x) = \tanh x = \frac{\epsilon^{x} - \epsilon^{-x}}{\epsilon^{x} + \epsilon^{-x}} \tag{A.2}$$

A.4 Back-propagation Algorithm

The back-propagation algorithm is a supervised algorithm to train artificial neural networks using gradient descent. The error of network output is calculated and it is propagated through network layers in reverse. The neural network's weights are updated according to its' gradients. Calculating gradients of final layer and use these to calculate previous layers gradients until reaching out the first layer. Thus, gradients flow through the final layer to the first layer and it makes it system efficient because there is no need to calculate the gradient of all layers separately at one computationally expensive step. Detailed explanations can be found in [99] and [100].

It is a general optimization method to differentiate complex nested functions. It is invented in the 1970s, however, it did not become popular until 1986, a paper published by Rumelhart, Hinton, and Williams, "Learning Representations by Back-Propagating Errors" [101], shows that artificial neural networks, for instance, multilayer perceptrons, could learn good internal representations as seen in Figure A.3. With this algorithm, it was shown that artificial neural networks can learn features that are hard to extract with classical image processing approaches. This enabled the generation of solutions to problems that could not be solved or limited due to time or computational cost constraints.

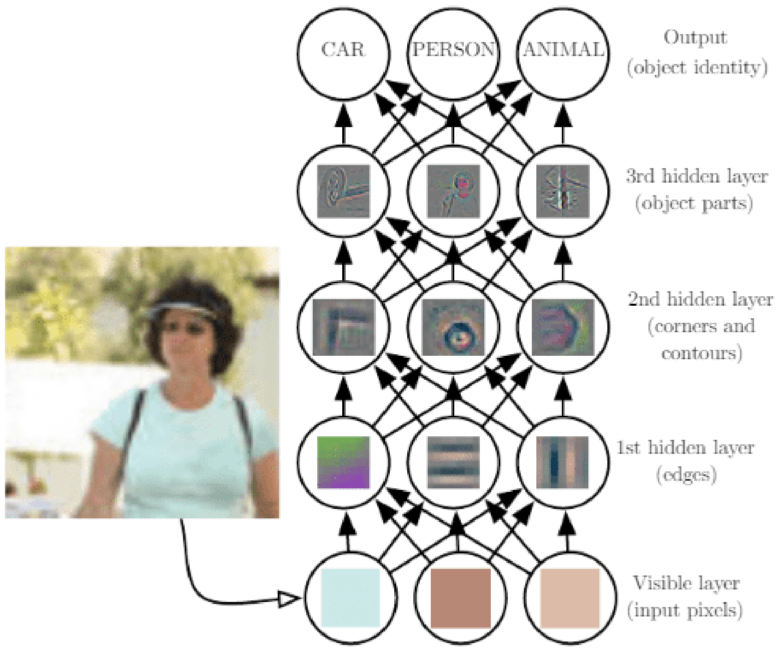


Figure A.3: Illustration of a deep learning model. It is taken from [45]. It shows that each hidden layer learns different features. First layers learn simple representations like edges, corners, etc. Through the final layers, the network learns more abstract features like object parts.

The method requires three things to work: A neural network that defines the weights and activation functions, a dataset that includes input and desired output pair, and an error function to calculate gradients.

1. ***Dataset*** is defined as $X = \{(x_i, y_i)|i = 1, 2, .., N\}$ where x_i is input and y_i is desired output.
2. A ***neural network*** defines the parameters θ and activation functions.
3. A ***cost function***, $E(X, \theta)$, defines relationship between desired output, y_i and generated output, $\hat{y}_i$ on input x_i. Thus, it is depending on network parameters and dataset.

The idea behind the algorithm requires to calculate gradients of the error function in the given sample (or samples, before running algorithm more than one result can be computed) with respect to network parameters θ. The opposite direction of the calculated gradients points minimum error region. In each update of the parameters, Ideally, the network becomes closer to the optimal point that gives the minimum error that is called the global minimum. Eqn. A.3 shows the gradient descent update rule. θ^t denotes the parameters at iteration t and α is the learning rate that specifies the change rate of the parameters.

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta} \tag{A.3}$$

As mentioned above, gradients flow reversely in the network. Before calculating gradients, network should calculate output, $\hat{y}_i$ for given input x_i. It is called forward pass. Then, the error function can be calculated. Forward pass for the network can be defined as Eqns. A.4 and A.5. Definition of the mathematical expressions given in Table A.1.

$$z^{(l)} = \theta^{(l)} \times a^{(l-1)} \tag{A.4}$$

$$a^{(l)} = \sigma(z^{(l)}) \tag{A.5}$$

Table A.1: Expressions and definitions used

Expressions	**Definitions**
L	Last Layer
l	current layer
$\hat{y}$	Predicted Output
y	Desired Output
z^l	Output of layer l
z^L	Output of Last layer ($\hat{y}$)
a^l	Activation result of layer l
$a^L = \hat{y}$	Activation result of output layer L
$\sigma(.)$	Activation function of nodes

To explain more clearly, error function can be defined as Eqn. A.6. The chain rule is used to calculate the gradients as given in Eqn. A.7. Starting from last layer Eqn. A.7 is calculated. According to gradients, last layer's parameters are updated using gradient descent in Eqn. A.3. The process continues with previous layer, $L-1$. To calculate gradients of layer $L-1$, we need to compute $\frac{\partial E}{\partial a^{(L-1)}}$ which is given in Eqn. A.8. First two part of the equation, $\frac{\partial E}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}$, is computed in the final layer gradient calculation which can be extracted in Eqn. A.7 putting $l = L$. Thus, previous calculations are reused. It can be imagined that gradient flows through layers reversely that is final to initial layer.

$$E(X,\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y^i} - y_i)^2 \tag{A.6}$$

$$\frac{\partial E}{\partial \theta^{(l)}} = \frac{\partial E}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial \theta^{(l)}} \tag{A.7}$$

$$\frac{\partial E}{\partial a^{(l-1)}} = \frac{\partial E}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial a^{(l-1)}} \tag{A.8}$$

Sample calculation for output layer, L is given in Eqn. A.9.

$$\frac{\partial E}{\partial \theta^{(L)}} = \frac{\partial E}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial \theta^{(L)}} = (\frac{1}{N}(\hat{y} - y))(\sigma^{'}(z^L))(a^{L-1}) \tag{A.9}$$

A.5 Regularization

It is desired that the artificial neural network architecture performs well not only in training data but also on new inputs. It is called overfitting. It is caused by noise in the training data. Artificial neural networks are too complex an architecture that is designed to learn any function. Thus, It can fit training data noise and it causes to fail in test examples that do not include the noise. Figure A.4 shows this phenomena with a simple example.

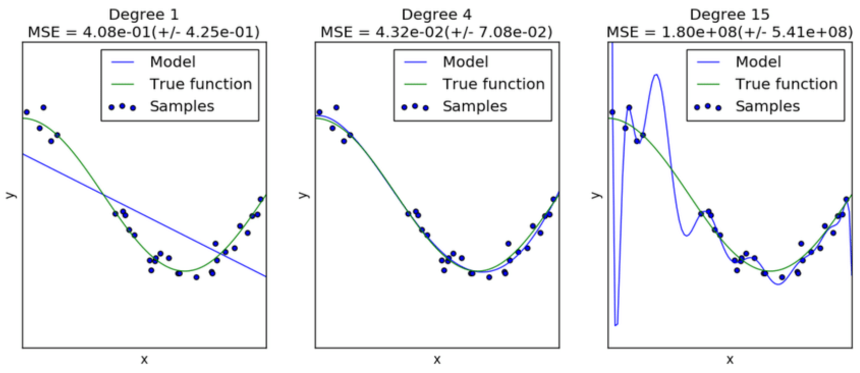


Figure A.4: Demonstration of underfitting, correct fit and overfitting from left to right respectively . It is taken from [46]

To avoid overfitting, there are several strategies and these are called regularization. Detailed explanations can be found in [102], [45].

A.5.1 Norm Penalties: L1 and L2 Regularization

These are also known as weight decay regularization. During L1 or L2 regularization, the regularization term is added to the loss function. This term Ω is equal to the L1 or L2 norm of the network parameters as defined in the Eqn. A.10. The term Ω is weighted by scalar α divided by 2 and added to the loss function as shown in Eqn. A.11. The $alpha$ is called as regularization rate. In the next step, to update network parameters according to the loss function, the gradient of the loss function is calculated. When the gradient of the Eqn. A.11 is computed and gradient descent update rule is applied, the Eqn. A.12 is obtained. The term $\epsilon \nabla_W L(W_{old}))$ in Eqn. A.12 is the penalty added to weight update equation. Thus, norm penalties try to minimize network parameters whereas the accuracy is maximized. Intuitively, smaller parameters reduce the impact of a neuron in the architecture. The overall complexity is reduced in this way. It is important to note that if the regularization term is weighted too much, the model becomes less complex and the accuracy of the model can not achieve certain goals.

$$\begin{aligned}\Omega(W) &= \|W\|_2^2 = \sum_i \sum_j w_{ij}^2 \quad L2norm \\ &= \|W\|_1 = \sum_i \sum_j w_{ij} \quad L1norm\end{aligned} \tag{A.10}$$

$$\hat{L}(W) = \frac{\alpha}{2}\Omega(W) + L(W) \tag{A.11}$$

$$\begin{aligned}W_{new} &= W_{old} - \epsilon(\alpha W_{old} + \nabla_W L(W_{old})) \\ &= (1 - \epsilon\alpha)W_{old} - \epsilon\nabla_W L(W_{old}))\end{aligned} \tag{A.12}$$

A.5.2 Dataset Augmentation

To make our trained model more stable and resistant to noise, the best way is to obtain more data that includes different aspects of the task and cover most of the input space. If the dataset is broad enough, our model can generalize the task better. Dataset augmentation is a method to create new fake data to increase dataset size. However, it can be difficult for various tasks to generate fake data unless mathematical solutions exist for this task. It is important to know task-dependent invariant transformations. For instance, the image classification task is independent of rotation, scaling, mirroring, etc. Thus, these transformations can be used to generate new fake data from the existing dataset. Furthermore, adding random noise to input images can be useful but the magnitude of the noise is carefully tuned. Further details can be found in [103].

A.5.3 Dropout

Dropout is a famous and powerful method for the regularization of artificial neural networks. The idea behind is quite simple. With a probability P, a neuron of the model is turned off during the training phase. Figure A.5 shows a feed-forward network with 2 hidden layers with and without dropout application. As mentioned earlier, closing some neurons make architecture simpler so, it can generalize task better and not overfit the training dataset.

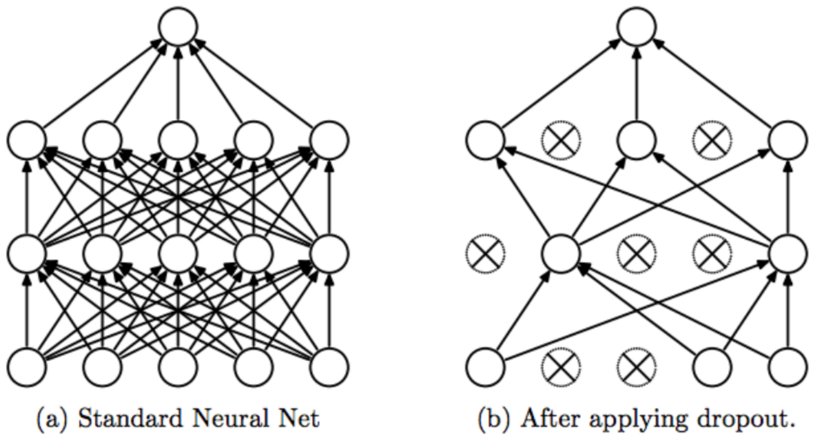


Figure A.5: Artificial neural network with and without dropout. With random probability some neurons are closed. Crossed units have been dropped. Image is taken from [47]

A.5.4 Early Stopping

When training artificial neural networks, the error rate decreases steadily over time while the validation set error does not. Thus, the aim is to find the point that validation set error minimum which is more probably to show better performance in the real test set. It is important to note that the validation set is not used for training. It is used to validate the model performance in a relatively smaller dataset. This strategy is known as early stopping. The Figure A.6 shows early stopping visually.

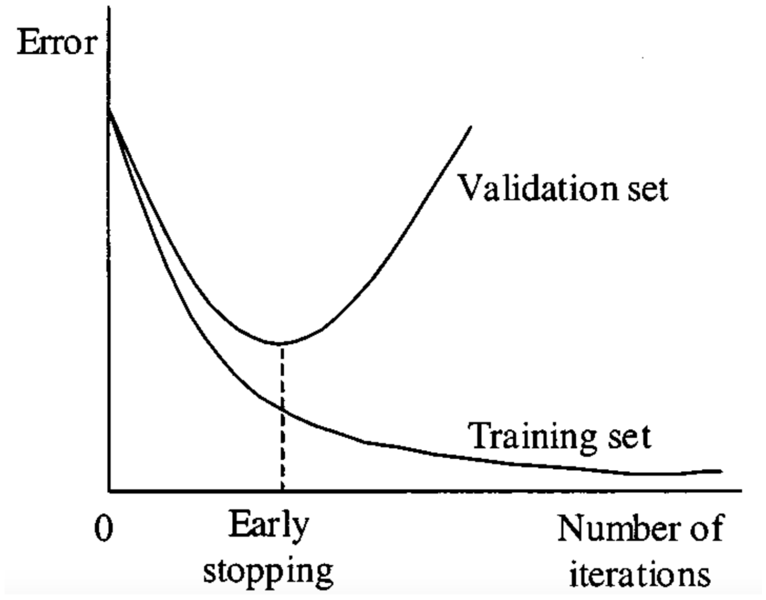


Figure A.6: Loss curves of the artificial neural network. This shows that from a point validation loss started to increase even if training loss continues to decrease.