Vint, D., Di Caterina, G., Soraghan, J., Lamb, R., & Humphreys, D. (2020). Analysis of deep learning architectures for turbulence mitigation in long-range imagery. In Proceedings of SPIE 11543, Artificial Intelligence and Machine Learning in Defense Applications II (Vol. 11543). SPIE. https://doi.org/10.1117/12.2573927

Analysis of deep learning architectures for turbulence mitigation in long-range imagery

David Vint[a], Gaetano Di Caterina[a], John Soraghan[a], Robert Lamb[b], and David Humphreys[b]

[a]Centre for Signal & Image Processing, University of Strathclyde, Glasgow, United Kingdom
[b]Airborne & Space Systems Division, Leonardo MW Ltd, Edinburgh, United Kingdom

ABSTRACT

In long range imagery, the atmosphere along the line of sight can result in unwanted visual effects. Random variations in the refractive index of the air causes light to shift and distort. When captured by a camera, this randomly induced variation results in blurred and spatially distorted images. The removal of such effects is greatly desired. Many traditional methods are able to reduce the effects of turbulence within images, however they require complex optimisation procedures or have large computational complexity. The use of deep learning for image processing has now become commonplace, with neural networks being able to outperform traditional methods in many fields. This paper presents an evaluation of various deep learning architectures on the task of turbulence mitigation. The core disadvantage of deep learning is the dependence on a large quantity of relevant data. For the task of turbulence mitigation, real life data is difficult to obtain, as a clean undistorted image is not always obtainable. Turbulent images were therefore generated with the use of a turbulence simulator. This was able to accurately represent atmospheric conditions and apply the resulting spatial distortions onto clean images. This paper provides a comparison between current state of the art image reconstruction convolutional neural networks. Each network is trained on simulated turbulence data. They are then assessed on a series of test images. It is shown that the networks are unable to provide high quality output images. However, they are shown to be able to reduce the effects of spatial warping within the test images. This paper provides critical analysis into the effectiveness of the application of deep learning. It is shown that deep learning has potential in this field, and can be used to make further improvements in the future.

Keywords: Turbulence Mitigation, Deep Learning, CNN, Turbulence Simulation

1. INTRODUCTION

The quality of optical imaging systems is constantly improving. This is especially true in the case of long distance imagery. Modern lenses are able to capture high quality images from very large distances. This is extremely useful in certain applications such as air to ground imagery or long distance object identification. In long range imaging, however, problems still occur: mainly that of diffraction limitations and turbulent atmospheres. Diffraction is an effect caused by the camera system's aperture. In order to ensure a suitable depth of field, the camera aperture must be small. This can cause the light to diffract within the camera and introduce blurring.

The effects of a turbulent atmosphere are more severe, and less controllable. A turbulent atmosphere can be caused by temperature and pressure changes, which lead to random changes in the refractive index of air[1]. In the case of optical turbulence, light that is propagating through the atmosphere can become severely distorted. This can result in imagery that contains a significant amount of spatial blur as well as spatial warping (or 'image dancing'). Due to the loss of information caused by such a turbulent atmosphere, it is highly desired that such effects could be reduced. This is the motivation for this work.

Turbulence mitigation in images can be approached in two ways. The first is the use of adaptive optics, which aim to reduce turbulence at the time of image capture. The second being to use post-acquisition processing, which aims to recover the lost high frequency details of the captured image. The field of turbulence mitigation in

Further author information: (Send correspondence to David Vint)
[a]E-mail:{david.vint, gaetano.di-caterina, j.soraghan}@strath.ac.uk
[b]E-mail:{robert.lamb, david.humphreys}@leonardocompany.com

imagery is a well researched field[2–8], where common techniques make use of temporally varying video sequences to obtain either a single high resolution image or a sequence of clean images.

Such techniques are well established and are able to mitigate turbulence within images to a high standard. However, it has been shown in recent years that the use of deep learning (DL) for image processing applications has resulted in higher performance, when compared to traditional image processing techniques. Examples of such fields are object classification, image segmentation, pose estimation and super resolution. This paper presents a review of the use of DL for turbulence mitigation. It is an investigation into the applicability of DL for removing complex turbulence within single images.

Such an application of DL to turbulence mitigation has been considered before[9–12]; however, this paper aims to assess the capabilities of several DL architectures that were not originally designed for turbulence mitigation. The motivation for this assessment is to evaluate which network will best adapt to the problem of turbulence mitigation. The networks under review were originally designed for tasks such as image denoising, super resolution or image deblurring. They are BRDNet[13], RDN[14], SuperSR[15], CAE-Unet-CAE[9], RCAN[16] and DnCNN[17]. Of these networks, CAE-Unet-CAE is the only architecture that was originally designed for the task of turbulence mitigation. The DnCNN architecture has however been applied to this task in various publications[10,12].

The most prevalent issue with most DL applications is the access to suitable data. Not only is suitably varied data required, but in order to accurately train a convolutional neural network (CNN), the amount of data required is great. In the case of image processing, each distorted image must have a high resolution, clean ground truth counterpart. For the case of turbulent imagery, this is a near impossible task, as it is extremely difficult to obtain a high resolution image without some form of control over the atmosphere. It is for this reason that the training of a CNN for turbulence mitigation is a difficult task. For the networks in this paper, the training data is simulated with an accurate turbulence simulator based on previous works in the literature[18,19]. This has allowed a large dataset to be generated, with both distorted and ground truth images.

The structure of the paper is as follows. Section 2 describes traditional techniques used for image restoration as well as common deep learning techniques. Section 3 describes the operation of the turbulence simulator developed for this work as well as the dataset generation process. Section 4 then presents the details of the chosen deep learning architectures. Section 5 describes the testing procedure that the networks are put through and presents the results. Section 6 presents a discussion of the effectiveness of deep learning to the field of turbulence mitigation. Section 7 finally concludes the paper.

2. RELATED WORK

2.1 Turbulence Mitigation

The degradation of images is commonly described with the following equation[20–24]:

$$y = Ax + n \tag{1}$$

where x is a clean image and y is the observed, degraded image. A is a matrix that denotes the degradation present within the image and n is additive noise. This model can describe a number of different imaging problems. For example, in the task of denoising, the matrix A is simply an identity matrix. In the case of deblurring, A represents a blurring matrix. In the case of turbulent interference, the matrix A represents the random spatial warping and blurring of the clean image x. The task of undoing the effects of A and n is extremely difficult, as the process is irreversible, unless the two parameters are accurately known. This results in an ill-posed problem, whereby, for a given an image y, an extremely large number of possible solutions for x can exist.

A technique that is widely used to attempt to solve (1) for the problem of turbulence mitigation is the use of multiple frames of the same turbulent scene. This allows methods to utilise the added information provided by subsequent frames to gain a better understanding of the clean latent image[2]. These methods can be loosely described by the diagram in Figure 1, where different approaches will follow different paths and will use different operations for each block. Common operations for each block are listed.

Image Selection refers to an analysis of each frame, to determine which frames are of the best quality. These frames are selected with the use of an image quality metric. These 'High Quality' frames are then passed on to the

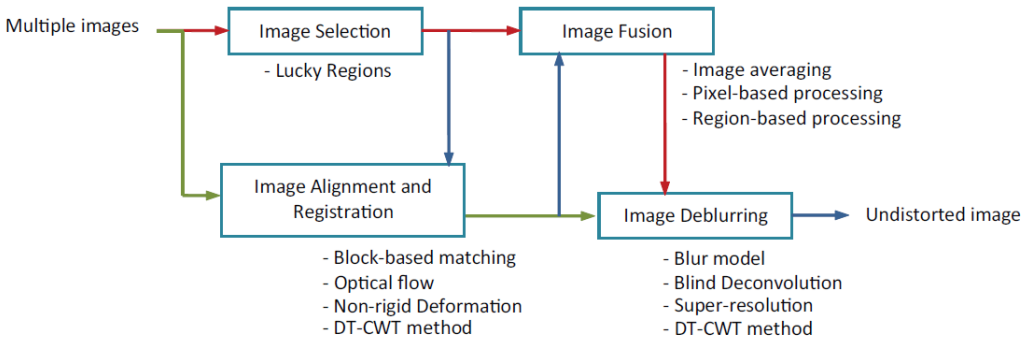


Figure 1. Flow of operations for traditional turbulence removal techniques[2]

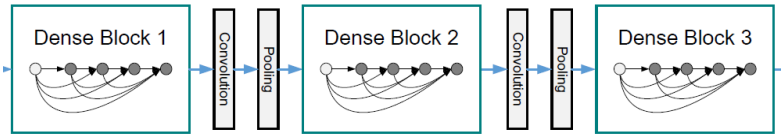


Figure 2. Example of densely connected layers[25]

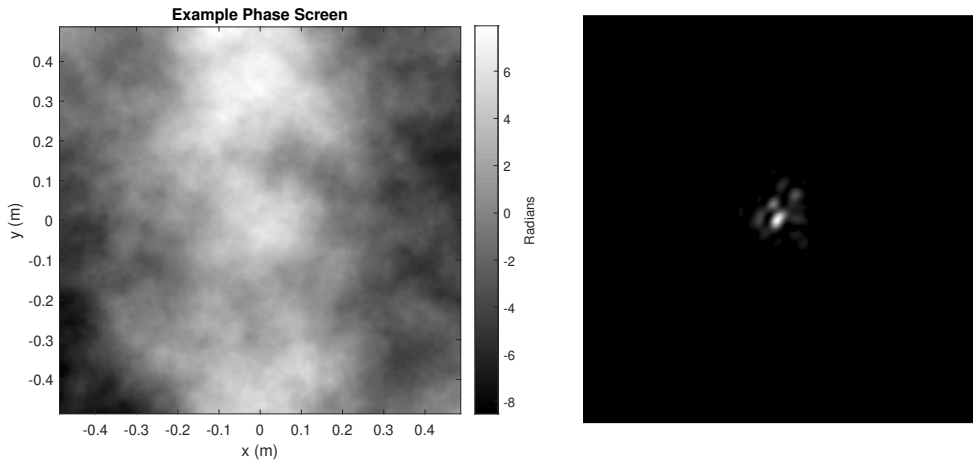
next stage of the model. Image alignment and registration is responsible for overcoming the spatial distortions in the video sequence. This block attempts to align the input frames to compensate for spatial inconsistencies between each frame. The image fusion block takes in a sequence of frames as an input and combines them together to form a single image. The purpose of this block is to reconstruct a latent image of the input frames. This is desirable as, by combining the distorted frames, the temporal difference between the frames is reduced, leaving a single image that represents the average distortion. This single image can then be fed into the image deblurring block. The task of this block is to remove any final blur present within the image. This block can also operate on individual frames if necessary. The output is then a clean image or a sequence of clean images.

2.2 Deep Learning

The task of image reconstruction is an extremely difficult task. This is due to the fact that a degraded image has already lost much of the high frequency content that is characteristic of high spatial resolution. It is then the task of the restoration algorithm to estimate these high frequency details. However, due to the ill-posed nature of this task, algorithms can struggle to recover these sharp details. The use of deep learning aims to approach the problem as a data driven learning problem. This can come in many different forms, as different image restoration tasks require different approaches. Areas of image reconstruction include denoising, super resolution, JPEG image deblocking and deblurring. Each of which has deep learning approaches in literature.

Denoising is a fundamental operation in image processing. The nature of image capture is extremely prone to additive noise, and commonly requires noise reduction. The use of of deep learning in this field is well established[26] and has been proven to outperform traditional techniques. Super resolution aims to accurately resize images, whilst also increasing high frequency detail. This field has seen many applications of deep learning. The most pertinent increase in quality was introduced by Dong et al.[27], who were the first authors to use CNNs for super resolution. In the years since, the use of deep learning has become the norm for super resolution, with an annual competition to assess super resolution networks on the same data[28–30]. Deblurring is another common problem in image processing. During image capture, each pixel on the sensor is exposed to a point spread function (PSF). If this PSF is larger than the pixel size on the sensor, blurring can take place. The aim of deblurring is to estimate this PSF and undo its effects. This can be achieved with methods such as deconvolution. However it can become challenging if the blurring in the image is spatially varying. In this case, deep learning provides a promising alternative approach.

It is common that any one deep learning architecture is trained on different datasets, allowing a single network architecture to provide multiple different image restoration tasks. For example, some authors use their networks



(a) (b)

Figure 3. Example of a Modified Von Karman Phase Screen and generated Point Spread Function after simulation: (a) example of a single phase screen; (b) resulting Point Spread Function after simulation.

on denoising, super resolution and deblurring[14,17]. These can be applied to different tasks, as the architecture of the networks is able to adapt to new data.

Most of networks in this paper make use of a number of commonly used CNN architecture designs. The first of which is residual learning, which is the inclusion of a skip connection that allows the input to the network to be combined with the output. Let the input to the network be a noisy/degraded image y:

$$y = x + n \tag{2}$$

where x is the ideal, sharp image and n is the noise/degredation applied to the image. The inclusion of a skip connection allows the network to simply learn n, which can then be subtracted from the original input x. This allows the network to gain a better understanding of the degradation process without any bias from the image contents. As an improvement on residual learning, networks can also employ residual 'blocks'[31], where the residual learning is performed over smaller blocks in the overall architecture. This allows the network to be far deeper, as the connections allow a 'highway' for the backpropagation of the loss. Inspired by the success of residual learning, another common practice in deep learning architecture is the use of densely connected layers first introduced by Huang et al.[25]. This type of architecture sees the output of each layer being fed as an input to each subsequent layer. This is illustrated in Figure 2. As with residual blocks, this aids backpropagation and also allows each layer to gain a better understanding of the layers preceding it.

2.3 Turbulence Simulation

In order to train the proposed CNN architectures, suitable data is required. In practice, it is extremely difficult to obtain a suitable dataset of turbulence affected imagery, as the high resolution, clean image, is unavailable. To obtain a suitable dataset, the conditions in which images are taken need to be controlled. Such as in Ref. 21, where turbulence is induced with gas hobs, allowing high resolution images to be captured when the hobs are switched off.

Another method of obtaining suitable data is that of data generation, whereby clean data is deliberately degraded to mimic effects of turbulence. In order to achieve this, an accurate model of how light propagates through turbulence is therefore needed.

In order to derive such a model, a number of methods have been proposed. These methods can be separated into 3 dimensional (3D) and 2 dimensional (2D) approaches. 3D approaches aim to fully simulate the propagation of light from one point to another, introducing the random distortions of turbulence during propagation. 2D approaches aim to avoid the computationally expense of a 3D simulation and apply the effects of turbulence onto

a 2D image. These 2D approaches are based on empirical data, from which they can extract suitable statistics. This allows the effects of turbulence to be reduced to a 2D representation which can then be applied to clean images[32,33].

More recently, deep learning has also been applied to turbulent data generation[34], where a Generative Adversarial Network (GAN) has been used to generate new turbulent data, providing a method of data generation that takes a fraction of the time of a statistical model.

In this paper, the turbulent imagery is simulated using a 3D propagation based turbulence simulator. This process simulates the full path of light from source to camera. The simulator is based on the works from Hardie et al.[35], as well as the MATLAB code provided in Schmidt[18].

3. DATA SIMULATION

The effect of turbulence is a strictly non-linear random phenomena. This makes the solving of the Navier-Stokes equations extremely challenging[36]. In order to simplify the process, Kolmogarov[37] developed a statistical model of turbulence which relies on certain assumptions.

A key idea used to model turbulent media is that of energy cascade. This states that in a turbulent medium, eddies form due to an injection of energy. These eddies then proceed to break up into smaller eddies and continue to reduce until the eddies dissipate completely as heat. This cascade of energy begins at a size L_0 and reduces in size to l_0. These two values are known as the inner and outer scales of turbulence. The group of eddies that lie between these two scales is known as the 'inertial subrange'.

It is within this inertial subrange that the assumptions of Kolmogarov are made. It is assumed that the eddies within the subrange are statistically homogeneous and isotropic. By using these assumptions and dimensional analysis, Kolmogarov was able to derive a power spectral density (PSD) for the changes in refractive index in air:

$$\Phi_n(\kappa) = 0.033 C_n^2 \kappa^{-11/3}, \quad 1/L_0 \ll \kappa \ll 1/l_0 \tag{3}$$

where κ is the angular spatial frequency in rad/m and C_n^2 is the refractive index structure parameter. It is a measure of the strength of the fluctuations of the refractive index in air. Typical values range from 1×10^{-16} (weak) to 1×10^{-13} (strong). Since the introduction of the Kolmogarov PSD, other models have been proposed. These introduce more flexibility in the choice of the inner and outer scale. Mainly, the modified Von Karman PSD given as:

$$S_{\phi_i}^{mvK}(\rho) = \frac{0.023 \mathrm{e}^{-\rho^2/\rho_m^2}}{r_{0_i}^{5/3} \left(\rho^2 + \rho_0^2\right)^{11/6}} \tag{4}$$

where $\rho_m = \frac{5.92}{2\pi l_0}$ and $\rho_0 = \frac{1}{L_0}$. r_{0_i} is the Fried parameter of the i^{th} phase screen and ρ is the radial spatial frequency. It can be noted, that by setting $l_0 = 0$ and $L_o = \infty$, the modified Von Karman PSD reduces to the Kolmogarov PSD.

The turbulence simulator model developed for this work is based off the works of Schmidt[18] and Hardie et al.[19]. These works develop turbulence models that make use of the modified Von Karman PSD to simulate the propagation of light through a turbulent atmosphere. The extended area of turbulent atmosphere is represented by a discrete number of phase screens. These phase screens are generated using the methods described in Ref. 18. An example of such a phase screen is shown in Figure 3a.

In order to simulate the propagation of light through these phase screens, a 2D Gaussian windowed sinc function is used as a point source. This point source is then propagated through each screen with the Fresnel diffraction equation, defined as:

$$\begin{aligned} U\left(\boldsymbol{r}_n\right) = & Q\left[\frac{m_{n-1}-1}{m_{n-1}\Delta z_{n-1}}, \boldsymbol{r}_n\right] \\ & \prod_{i=1}^{n-1}\left\{\mathcal{T}\left[z_{i+1}\right]\mathcal{F}^{-1}\left[\boldsymbol{f}_i, \frac{\boldsymbol{r}_{i+1}}{m_i}\right] Q_2\left[-\frac{\Delta z_i}{m_i}, \boldsymbol{f}_i\right]\mathcal{F}\left[\boldsymbol{r}_i, \boldsymbol{f}_i\right]\left(\frac{1}{m_i}\right)\right\}\left\{Q\left[\frac{1-m_1}{\Delta z}, \boldsymbol{r}_1\right] U\left(\boldsymbol{r}_1\right)\right\} \end{aligned}$$

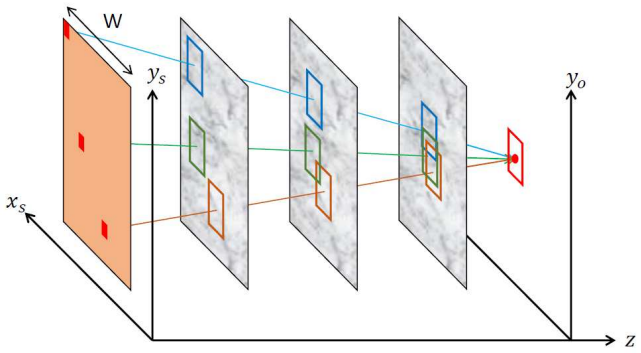


Figure 4. Illustration of application of isoplanatic imaging.

where $Q[\cdot]$ and $Q_2[\cdot]$ are quadratic phase factors and $F[\cdot]$ and $F^{-1}[\cdot]$ are the forward and inverse Fourier transforms respectively. $\mathcal{T}[z_{i+1}]$ is the phase induced by the phase screens. $U(\boldsymbol{r}_1)$ is the complex point source at the source plane and $U(\boldsymbol{r}_n)$ is the resulting complex plane at the observation plane.

Further processing to the final complex plane is then performed to obtain a point spread function of the imaging system. This is mainly the multiplication of a camera aperture mask, $a(x, y)$ and a collimation operation that allows the focusing of the image on the focal point. These operations are given as:

$$p(x, y) = a(x, y)U(\boldsymbol{r}_n) \exp\left[\frac{-j\pi\left(x^2 + y^2\right)}{\lambda L}\right] \tag{5}$$

The PSF can then be calculated according to Fourier optics[38] as:

$$h(x, y) = \left(|\mathrm{FT}\{p(x, y)\}|^2\right)\Big|_{u=\frac{x}{\lambda l}, v=\frac{y}{\lambda l}} \tag{6}$$

An example of such a PSF is shown in Figure 3b. In order to gain a more accurate simulation of real turbulence, the work in Ref. 19 is recreated, whereby the phase screens are generated to an extended size. This allows the optical path of each pixel to be traced through the atmosphere. Each pixel is then simulated separately. This is known as anisoplanatic imaging, and is the cause of image dancing. An illustration of this technique is shown in Figure 4.

This results in each pixel in the image to have its own bespoke PSF, dependant on its path through the extended phase screens. The simulator does include a 'skip' parameter that allows pixels in the simulation not to be processed. The missing PSFs are then interpolated from the surrounding pixels. This does not introduce any errors into the simulation, as pixels close together lie within the same isoplanatic angle. This angle denotes that any two sources within a certain angle propagate through the same patches of turbulent atmosphere. The skip parameter allows the computation of the simulation simulation to be greatly reduced.

Given these details, the simulator is able to accurately simulate anisoplanatic turbulent in a number of different situations. The inputs to the system allow a large number of scenarios to be simulated. These inputs are detailed in Table 1.

3.1 Dataset Generation

In order to train the CNN architectures with appropriate scenarios, the training data was to be suitable for the task at hand. This required a dataset of scenes that could potentially be corrupted by turbulent atmosphere. Given this, the places dataset[39] was to be used.

This dataset contains a total of 1,469,737 scene images within 205 different categories. Of these 205 categories, 31 were chosen as categories that could be prone to turbulent interference, where categories were excluded if the contents within the scenes were unlikely to be affected in a real scenario, such as indoor scenes or scenes at

Table 1. Turbulence Simulator Settings.

Setting	Description
C_n^2	Atmospheric structure parameter
D2	Diameter of the camera system
L	Propagation Distance
nscr	Number of phase screens
L_0, l_0	Inner and outer scale of turbulence
Skip	The number of pixels to be skipped in each simulation
λ	Wavelength of light

close ranges. The remaining categories contained images of outside scenes at varying ranges and environments. Although the dataset provides RGB colour images, the images used in this work were converted into greyscale, in order for their use in the turbulence simulator.

In order to fully test the trained networks, a three stage testing procedure was used, where each stage introduced more turbulent imagery. To facilitate this, three levels of distortion were defined as Low, Medium and High turbulence. In each case, the level of turbulence within the images directly relates to the Cn2 value of the simulator.

The value of Cn2 can vary from severe turbulence at a value of $1 \times 10^{-13} m^{-2/3}$, to weak turbulence at a value of 1×10^{-16}[1]. Therefore, the three levels of distortion are defined as:

- Low: $0.001 \times 10^{-13} m^{-2/3}$ –> $0.334 \times 10^{-13} m^{-2/3}$
- Medium: $0.334 \times 10^{-13} m^{-2/3}$ –> $0.667 \times 10^{-13} m^{-2/3}$
- High: $0.667 \times 10^{-13} m^{-2/3}$ –> $1 \times 10^{-13} m^{-2/3}$

Using these turbulence levels, four datasets were generated. These were labelled as 'Low', 'Medium', 'High' and 'Mixed'. Where the values of C_n^2 for 'Low', 'Medium' and 'High' corresponded to the definitions above. The fourth dataset, 'Mixed', was to be used to train the networks, and contained an even distribution of images from each of the levels of distortion. The other parameters needed for the simulation are describes in Table 2.

Table 2. Chosen Turbulence Simulator Settings.

Setting	Value
D2	0.1m
L	5km
nscr	5
l_0	0.01m
L_0	300m
Skip	4
λ	525nm

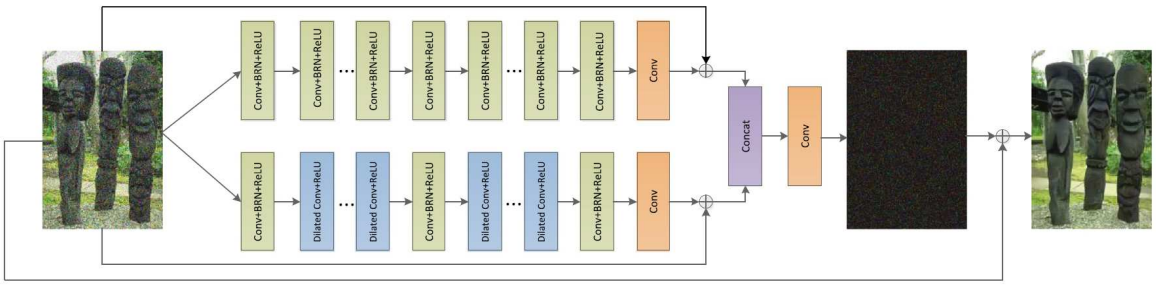


Figure 5. The CNN architecture of the BRDNet[13]

4. DEEP LEARNING ARCHITECTURES

4.1 BRDNet

Initially designed for the task of denoising, the architecture from Tian et al.[13] makes use of a combination of batch renormalisation and dilated convolutions in its design. Their overall network (Figure 5) consists of two subnetworks that run in parallel. The outputs of each subnetwork are then concatenated and processed further to provide the final output image. This network contains two main types of layer: 'Conv+BRN+ReLU' and 'Dilated Conv+ReLU'. The first of which contains a Convolutional layer with a kernel size of $3 \times 3 \times 64$. This is followed by a batch renormalisation layer followed by a ReLU activation function. The 'Dilated Conv+ReLU' layer consists of a dilated convolution layer with a kernel size of $3 \times 3 \times 64$ with a dilation factor of 2. This is then followed by a ReLU activation function. The top network consists of 16 'Conv+BRN+ReLU' layers, with a final convolutional layer with a kernel size of $3 \times 3 \times c$, where c is the depth of the image input. The output of this layer is then summed with the original image input. The lower network consists of a single 'Conv+BRN+ReLU' layer followed by 7 'Dilated Conv+ReLU' layers. These are then followed by another 'Conv+BRN+ReLU' and 6 more 'Dilated Conv+ReLU' layers. A final 'Conv+BRN+ReLU' layer then leads into a single convolutional layer that also uses a kernel size of $3 \times 3 \times c$. As with the top network, the output of this layer is summed with the image input. The output of the two summations are then concatenated together and passed through a final convolutional layer. This output can be summed with the original image to produce a clean output image. The use of the two networks in parallel allows a greater width to be obtained, and the use of dilated convolutions allow a greater receptive field to be captured, whilst keeping computational cost low. The use of batch renormalisation avoids potential issues with small batch size and internal covariate shift.

4.2 RDN

The Residual Dense Network from Zhang et al.[14] present the use of Residual Dense Blocks (RDB) within their network. These blocks (Figure 7) employ both local residual learning (LRL) and densely connected convolutional layers. The combination of these techniques as well as local feature fusion (LFF) leads to a contiguous memory (CM) mechanism. As can be seen, the output of each layer is fed on as an input to each subsequent layer. This concatenation of feature maps can quickly lead to a significant number of parameters. The LFF mitigates this by reducing the large number of concatenated feature maps with the use of a 1×1 kernel. The output of which is then summed with the input of the current RDB block as local residual learning.

Within the main network, RDB blocks are stacked together as seen in Figure 6, where the output of each RDB block is fed to a concatenation block. This is known as Global Feature Fusion (GFF). As well as GFF, the main network consists of two skip connections, providing global residual learning.

Within this network, the number of convolutional layers per RDB is denoted as C, the number of RDB blocks within the main network is denoted as D and the number of feature maps per convolution layer is denoted by G. The authors show that for each of these variables, larger values leads to better performance. The authors also show that the use of CM, LFF, and FF provide performance increases with an ablation investigation.

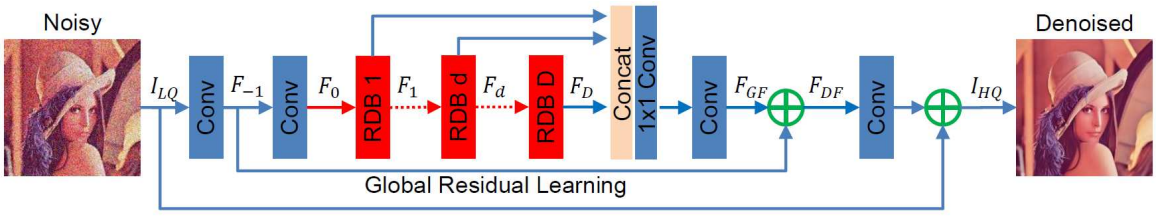


Figure 6. Architecture of RDN[14]

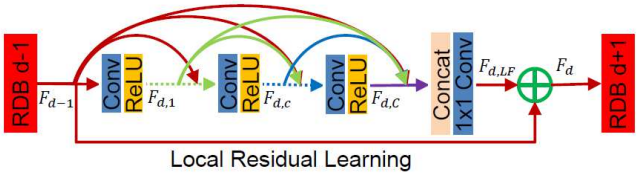


Figure 7. Residual Dense Block used in the RDN architecture[14]

4.3 CAE-Unet-CAE

The architecture from Chen et al.[9] is the only architecture in this work that was initially designed to perform turbulence mitigation. It consists of three networks, the first of which is a Convolutional Autoencoder (CAE). This is followed by a U-net structure, which is then followed by a second CAE, as seen in Figure 8. The U-net is utilised to remove the noise within images, and the pair of CAEs are used for feature extraction and image reconstruction respectively. In order to train their network, the authors use the theoretical PSF for atmospheric turbulence, given as:

$$h\left(u,v\right)=e^{\left\{-3.44\left(\frac{\lambda fU}{r_0}\right)^{\frac{5}{3}}\left[1-\beta\left(\frac{\lambda fU}{r_0}\right)^{\frac{1}{3}}\right]\right\}} \tag{7}$$

For each of the layers within the network a kernel size of 3×3 is used. In each of the CAEs, the number of feature maps for each layer is 64. Within the U-net, after each pooling layer, the number of feature maps is doubled. Given a total of 3 pooling layers, the number of feature maps follows the pattern: 64-128-256-512-256-128-64, where the upsample operations were achieved with transpose convolutions. The CAE-Unet-CAE architecture also employs residual learning within the U-net as well as symmetric skip connections between the two CAEs.

4.4 SuperSR

The model proposed by Feng et al.[15] was initially designed for the task of super resolution and was runner up in the 2019 NTIRE super resolution challenge[30]. As can be seen in Figure 9, their network consists of a U-net type structure, whereby the input resolutions is downsampled for processing, and then upsampled for image reconstruction. Within the main network, after downsampling, several 'Cascading blocks' are connected together, where the output of each cascade block is fed forward as an input to all proceeding blocks. Each of these cascade blocks contain a similar structure, this time with RCAB blocks. The RCAB block was initially introduced in the work by Zhang et al.[16]. They are designed such that the network is able to focus on more informative features. The original RCAB block is shown in Figure 11. For the SuperSR network, each of the kernels are of size 3×3 (with the exception of the 1×1 convolutions).

4.5 RCAN

The first network to introduce Channel Attention blocks, the architecture from Zhang et al.[16] was designed for the task of super resolution. It consists of residual in residual (RIR) structure, as can be seen in Figure 10. The main network originally consists of 10 residual groups, each containing 20 RCAB blocks (Figure 11). Before passing into the RIR structure, the input to the network is first passed through a single convolutional layer. In the original paper, the output of the RIR structure is upscaled with the use of ESPCNN (or Pixel Shuffle)[40]. This is omitted in this investigation, as the goal of the retrained network would be to remove turbulence from

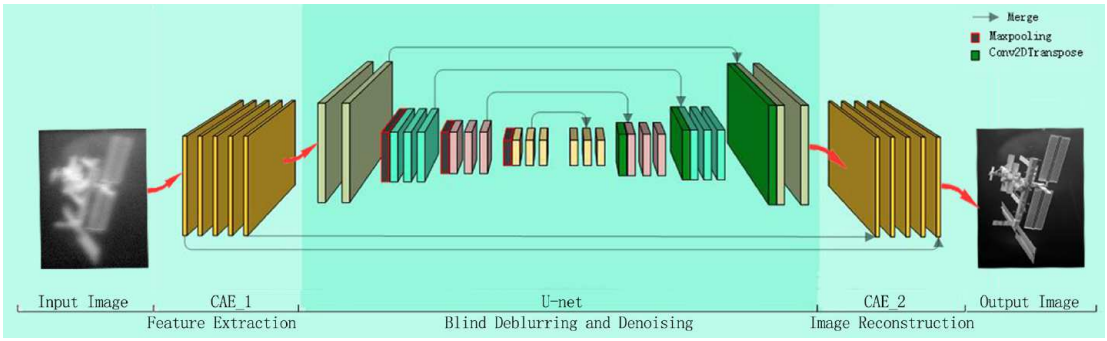


Figure 8. Architecture of CAE-Unet-CAE[9]

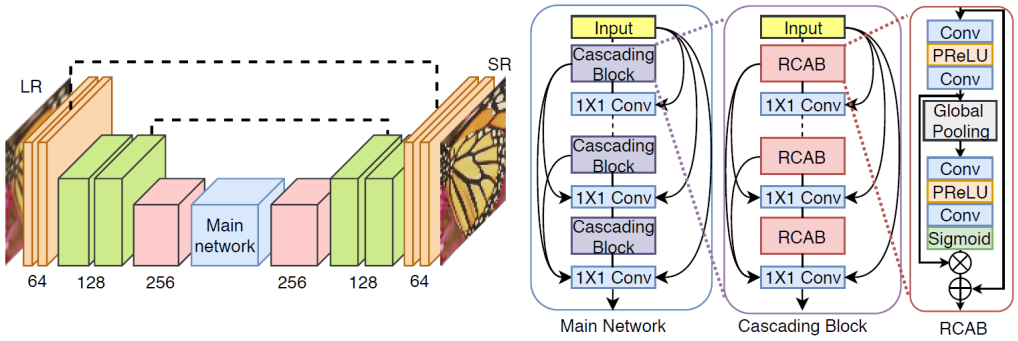


Figure 9. Architecture of SuperSR[15]

an image, without changing the spatial dimensions. Instead the output of the RIR is passed through a single convolutional layer. As with SuperSR, each of the kernels has a size of 3×3 with the exception of the downscaling and upscaling layers, which have a kernel size of 1×1.

4.6 DnCNN

The DnCNN architecture from Zhang et al.[17] is the oldest network in this investigation. It is commonly used as a comparison for newer networks[20,26,41,42].

It has been applied to the task of turbulence mitigation before. In Ref. 12, DnCNN is used within a traditional turbulence mitigation technique, where it is employed to replace traditional blur estimation and unsharp masking operations. In Ref. 10 the task of turbulence mitigation is completed solely with the use of DnCNN. Within this paper, they alter the kernel size of the network from 3×3 to 5×5 in order to capture the full receptive field of the turbulent image.

The network architecture is shown in Figure 12. The input is first passed through a convolutional layer with ReLU activation. This is then followed by 17 'Conv+BN+ReLU' layers, where BN is Batch normalisation. The final residual image is then constructed with a final convolutional layer. This residual can then be subtracted from the original input to provide a clean output image.

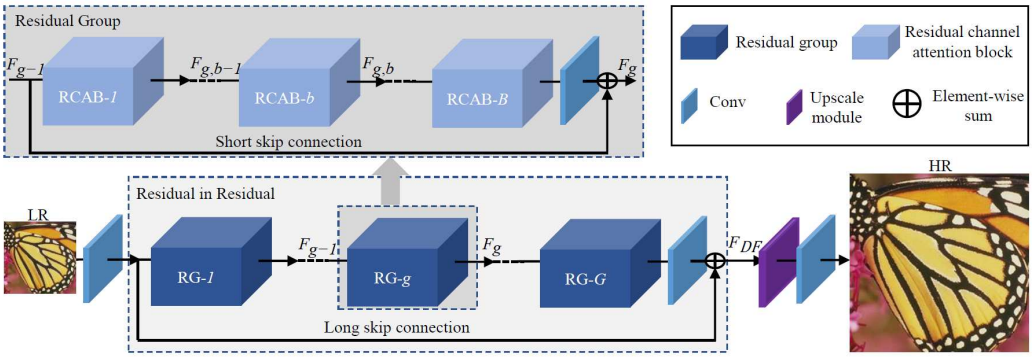


Figure 10. Architecture of RCAN[16]

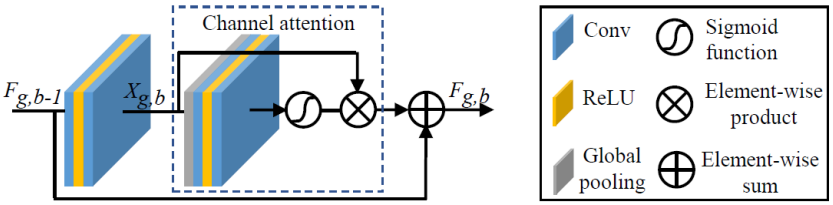


Figure 11. Architecture of the RCAB Block used in both RCAN and SuperSR[16]

5. ARCHITECTURE COMPARISON

5.1 Training Procedure

The construction of the networks was achieved by following the details provided in the corresponding papers. To our knowledge, each network has been faithfully recreated according to the original works. For each of the models under investigation, the basic architecture structure remained unaltered with the exception of the RDN and RCAN architectures. For the case of RDN, the feature size of each layer was reduced from 64 to 32 due to memory limitations. Within RCAN, the original model contained 10 Residual Groups (RG) and 20 RCAB blocks per RG. This was reduced to 5 RGs and 10 RCAB blocks per RG, also due to memory limitations. For the DnCNN model, a kernel size of 5×5 and network depth of 17 was used, as in Ref. 10.

The loss function for each network was kept as intended in the source papers and the ADAM optimiser was used in each network, with $B_1 = 0.9$, $B_2 = 0.999$ and $\delta = 10e^{-8}$.

To identify a suitable learning rate for each network, each was tested at different rates. The best learning rate for each network was then used for training. A similar process was performed for the batch size for each network. As well as the training settings, the training length was bespoke to each network. This was necessary, as different networks would train faster than others. Therefore, the epoch count was changed for each network. The final learning rates, loss functions, optimisers, batch sizes and epoch counts can be seen in Table 3.

Table 3. Network Training Settings.

Network	Learning Rate	Loss	Optimiser	Batch Size	Epoch Count
BRDNet	0.00001	L2	ADAM	16	10
RDN	0.0001	L1	ADAM	10	15
SuperSR	0.00001	L1	ADAM	16	10
CAE-Unet-CAE	0.001	L2	ADAM	16	15
RCAN	0.0001	L1	ADAM	10	15
DnCNN	0.00001	L2	ADAM	10	10

For the networks with 10 epochs of training, the learning rate was halved after the 5th and 8th epochs. For the networks with 15 epochs of training, the learning rate was halved after the 9th and 12th epochs.

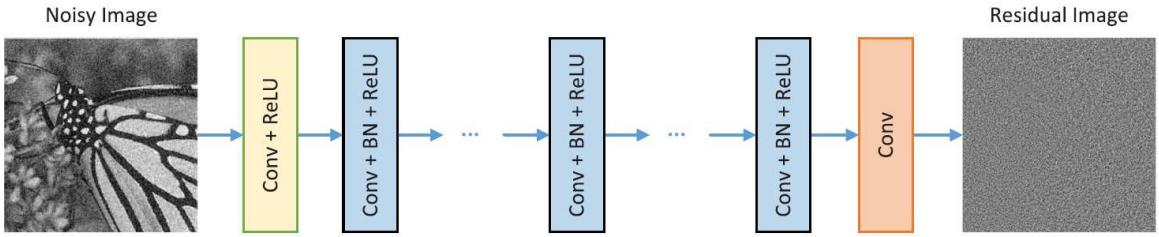


Figure 12. Architecture of DnCNN[17]

Each model was trained using the 'Mixed' turbulence dataset described in section 3.1, This consists of 3995 turbulent images. For training, 20 patches of size 80x80 were extracted from each image, providing a total dataset size of 79900 patches. This was then further split into training and validation sets with a ratio of 0.8. Resulting in 63920 training patches and 15980 validation patches.

5.2 Results

In order to gain a full understanding of the networks performances, a suitable testing procedure was required. As detailed in section 3.1, four datasets were generated. The models were trained on the 'Mixed' data. For testing, each of the networks were given the same 1000 images from the 'Low', 'Medium' and 'High' datasets respectively. For each image, the Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index[43] (SSIM) metrics were used to measure performance. The average of each metric was then taken. These results are shown in Table 4.

Table 4. Average PSNR and SSIM results of each network. Each test dataset contained 1000 images. (PSNR/SSIM).

Network	Low	Medium	High
Low Resolution Inputs	23.36/0.669	**20.90**/0.544	**20.19**/0.504
BRDNet	19.08/0.598	18.39/0.491	17.87/0.451
RDN	**23.77**/**0.704**	20.66/**0.568**	20.13/**0.534**
SuperSR	22.74/0.685	20.51/0.547	19.93/0.508
CAE-Unet-CAE	22.55/0.676	20.55/0.551	19.97/0.513
RCAN	22.14/0.688	19.75/0.545	19.21/0.503
DnCNN	23.35/0.669	*20.89*/0.545	*20.19*/0.505

It can be seen that, of the deep learning architectures, RDN provided the highest PSNR value for the 'Low' level of turbulence and DnCNN provided the highest PSNR for 'Medium' and 'High' turbulence. For all levels of turbulence, RDN provided the highest SSIM score. It can also be seen that, with the exception of the 'Low' level turbulence, none of the networks were able to produce images of better quality than the turbulent input. Figures 13-18 provide example outputs of each network. For each level of turbulence, two cases are presented. The first being an image that has successfully been improved by one or more networks. The second case is an image that the networks struggle to improve.

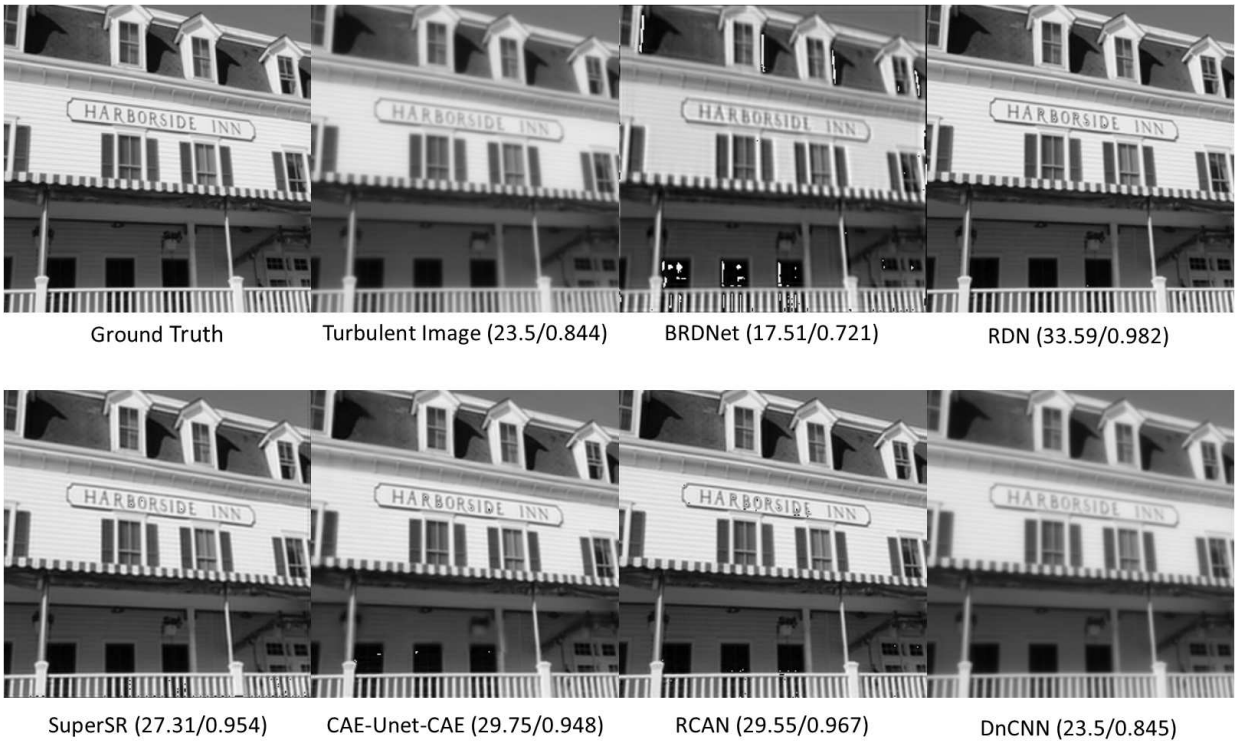


Figure 13. Successful outputs of 'Low' level turbulence. (PSNR/SSIM)

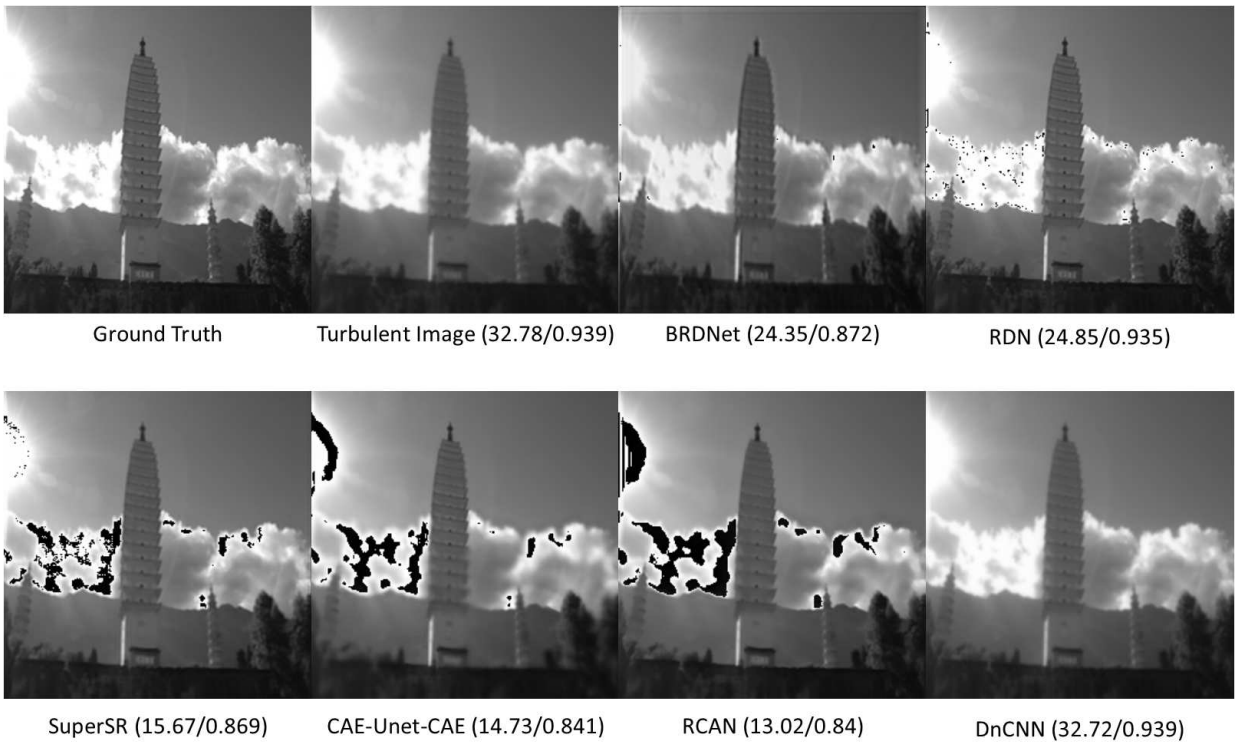


Figure 14. Unsuccessful outputs of 'Low' level turbulence. (PSNR/SSIM)

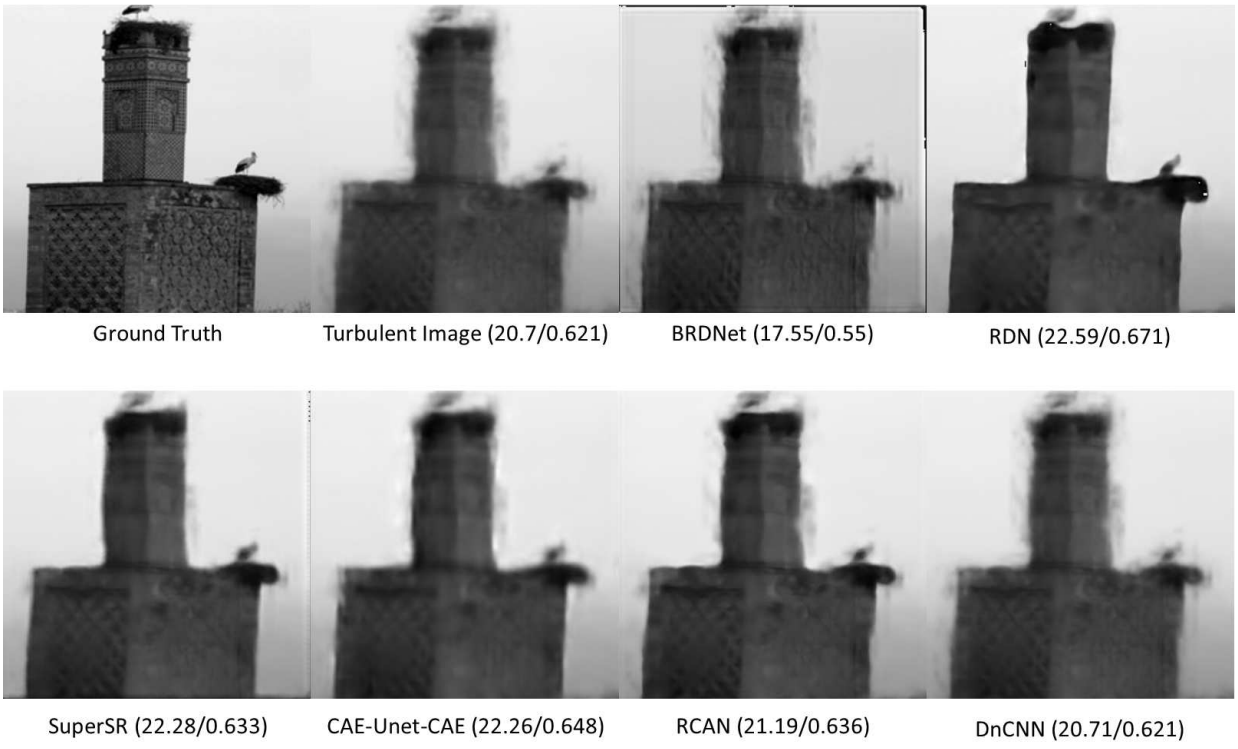


Figure 15. Successful outputs of 'Medium' level turbulence. (PSNR/SSIM)

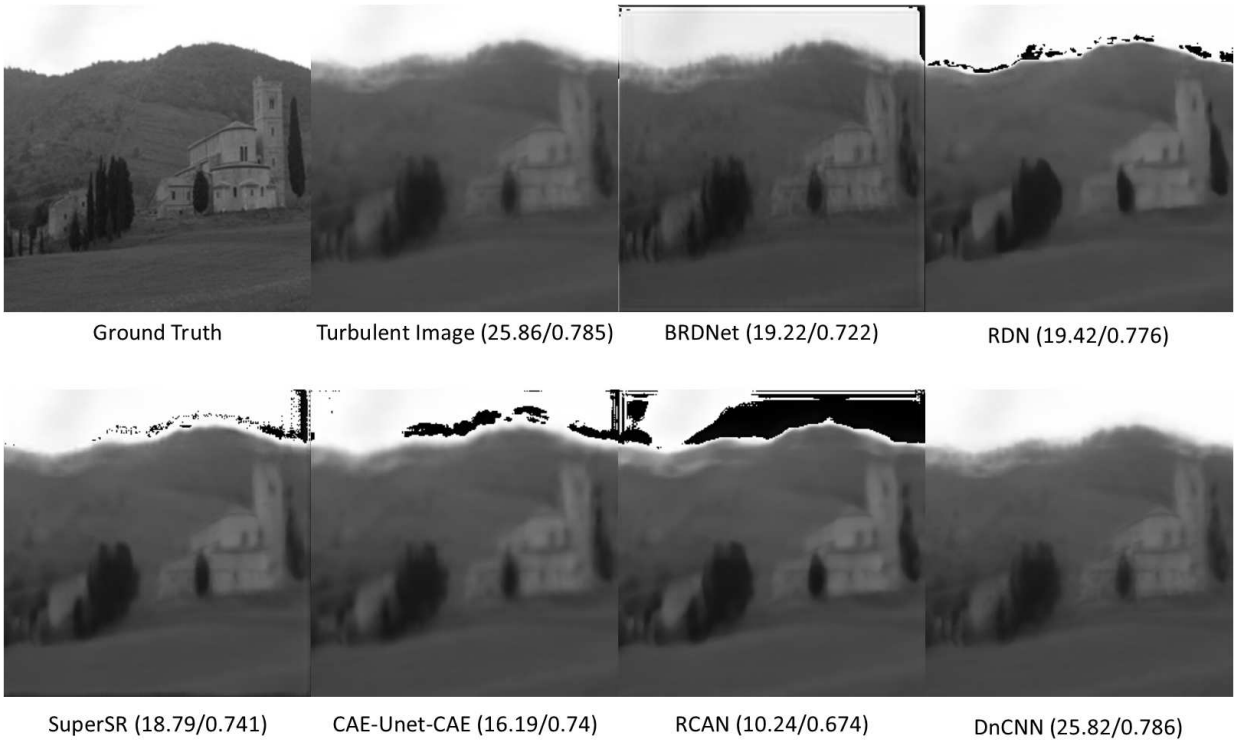


Figure 16. Unsuccessful outputs of 'Medium' level turbulence. (PSNR/SSIM)

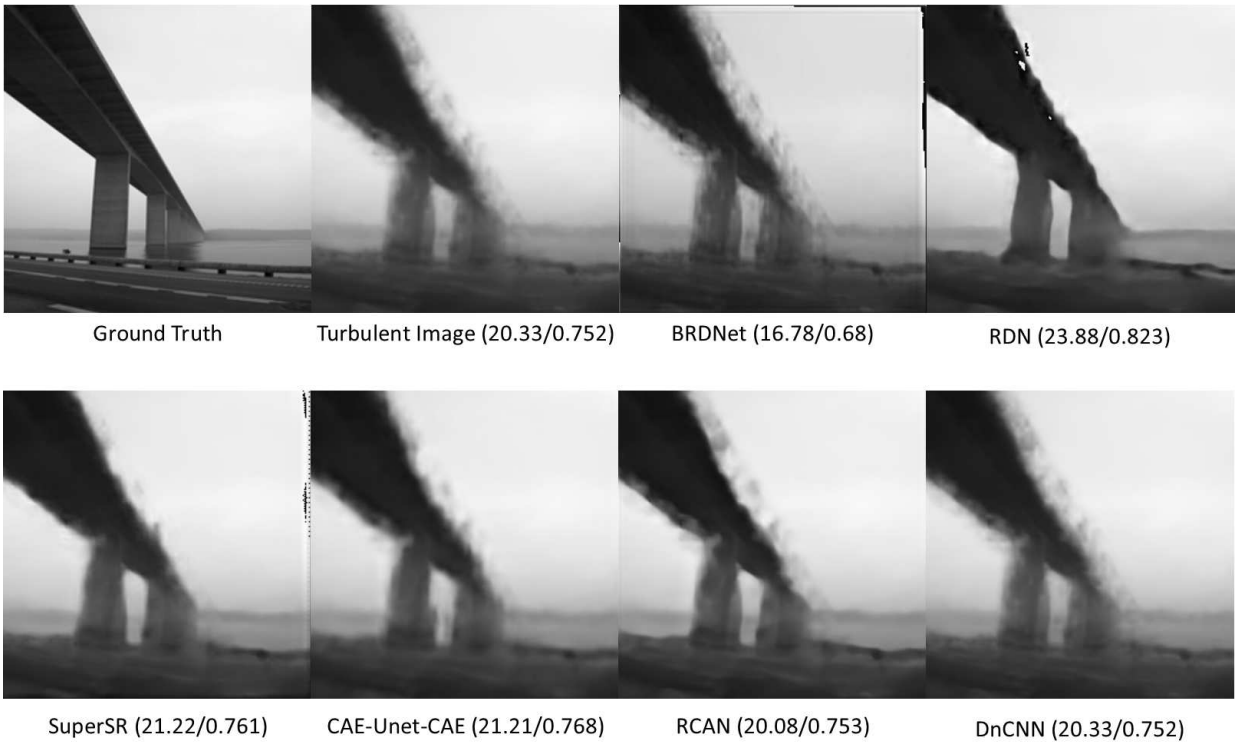


Figure 17. Successful outputs of 'High' level turbulence. (PSNR/SSIM)

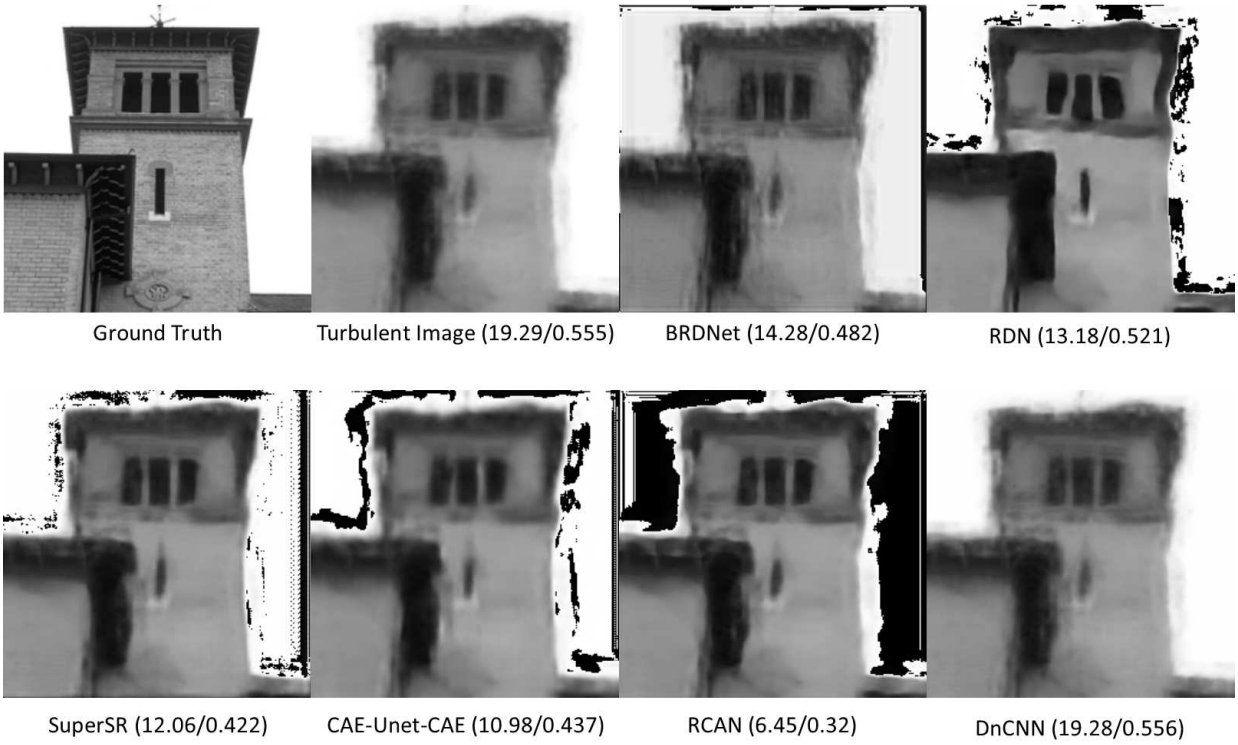


Figure 18. Unsuccessful outputs of 'High' level turbulence. (PSNR/SSIM)

6. DISCUSSION

The results in Table 4 suggest that the networks are unable to fully mitigate the effects of turbulence in images. Especially in the case of high turbulence levels. This is further shown by the analysis of individual images, where the only cases of success occur at very low turbulence levels. This shows that the task of turbulence mitigation is indeed a complex one. With the exception of CAE-Unet-CAE, the networks used in this review were not initially designed for turbulence mitigation. The fact that CAE-Unet-CAE was outperformed in this review shows that network architectures designed for other tasks, can indeed be retrained on different datasets to good effect.

It is believed that the cause of the results shown is due to two main factors. Firstly, the turbulence present within any two images is never the same. This means that the network has to understand the underlying effects of the turbulence, as opposed to simply learning the degradation process and undoing it. The second factor is the fact that the networks in this review were trained on single images. This increases the difficulty in restoration, as the high level of distortion is extremely difficult to remove with only one example of each scene. A method that is adopted by traditional turbulence mitigation techniques is the use of video sequences. This allows multiple variations of a single scene, allowing methods to make use of the additional temporal information in order to gain a better understanding of the clean latent image. Given the results of single image turbulence mitigation, it can be seen that in the case of turbulent imagery, a great deal can be gained from temporally related data.

This is not to say, however, that the use of deep learning for single image turbulence mitigation is unachievable. By analysis of individual images, it can be seen that the networks are able to provide a slight visual improvement in some cases. Such a case can be seen in Figure 15, where the RDN network is mostly able to remove the spatial distortions in the image. It is however unable to recover the high frequency textures. A similar result can be seen in Figure 17, where both RDN and SuperSR are able to significantly reduce the spatial warping. Even within the cases where the PSNR is reduced, as in Figure 18, the warp present on the building has been reduced. This shows that, although the networks are unable to recover high frequency detail, they are able to reduce the effects of image dancing. This is promising, as it denotes that, if the networks were retrained on lower levels of turbulence, they might be able to provide better, sharper results.

Another point of interest is the effect that can be seen in Figures 18, 16 and 14, where the background of the image suffers from a type of 'clipping'. This shows that the networks struggle to cope with very bright areas in the images, which has resulted in the significant decrease in the PSNR value. However, it seems that the foreground of these effected images can show positive improvements over the input image. This may indicate an issue in the training procedure, whereby these areas of 'clipping' have introduced an unexpected error.

Overall, the results of this review has allowed a foothold in the field of single image turbulence mitigation using deep learning, and has highlighted the challenges of such a task. Future research into this field can build upon these findings and provide better results. Such as the use of temporally relevant video sequences or building a single image architecture that is better suited to the complex nature of turbulent data.

7. CONCLUSION

This paper has provided a review of the use of deep learning for turbulence mitigation in long range imagery. A detailed simulator has been developed for this process, to allow the proposed networks to train on a sufficient amount of data. The networks were then trained on data containing multiple different levels of turbulent imagery. Each network was then tested on a series of datasets to ascertain their ability to remove the effects of three levels of turbulence within images. It was discovered, that the networks in question were not able to produce a considerable improvement in each of the turbulent levels. It is believed that this is due to the complex nature of the turbulence. However, it was observed that in some cases, the architectures were able to provide some form of improvement in image structure, but are unable to recover high frequency information. Of the chosen networks under review, the RDN architecture was able to produce the best results.

ACKNOWLEDGMENTS

D.V. acknowledged support from the UK EPSRC and from Leonardo MW Ltd, Edinburgh. D.V. would also like to thank Aidan McFadden for his help in the development of the turbulence simulator.

REFERENCES

[1] Tunick, A., Tikhonov, N., Vorontsov, M., and Carhart, G., "Characterization of optical turbulence (cn2) data measured at the arl alot facilitys," in [*Army Research Laboratory*], (2005).

[2] Anantrasirichai, N., Achim, A., and Bull, D., "Mitigating the effects of atmospheric distortion on video imagery: A review," (2011).

[3] Anantrasirichai, N., Achim, A., and Bull, D., "Atmospheric turbulence mitigation for sequences with moving objects using recursive image fusion," in [*2018 25th IEEE International Conference on Image Processing (ICIP)*], 2895–2899, IEEE (2018).

[4] Lau, C. P., Lai, Y. H., and Lui, L. M., "Variational models for joint subsampling and reconstruction of turbulence-degraded images," *Journal of Scientific Computing* **78**(3), 1488–1525 (2019).

[5] Hardie, R. C., Rucci, M., Karch, B. K., Dapore, A. J., Droege, D. R., and French, J. C., "Fusion of interpolated frames superresolution in the presence of atmospheric optical turbulence," in [*Optical Engineering*], **58**, 1–1 (2019).

[6] Chimitt, N., Mao, Z., Hong, G., and Chan, S. H., "Rethinking atmospheric turbulence mitigation," *arXiv preprint arXiv:1905.07498* (2019).

[7] Hardie, R. C., Rucci, M. A., Karch, B. K., Dapore, A. J., and Droege, D. R., "Super-resolution in the presence of atmospheric optical turbulence," in [*Long-Range Imaging III*], **10650**, 106500H, International Society for Optics and Photonics (2018).

[8] Rucci, M. A., Hardie, R. C., and Dapore, A. J., "Comparing multiple turbulence restoration algorithms performance on noisy anisoplanatic imagery," in [*Long-Range Imaging II*], **10204**, 1020409, International Society for Optics and Photonics (2017).

[9] Chen, G., Gao, Z., Wang, Q., and Luo, Q., "U-net like deep autoencoders for deblurring atmospheric turbulence," *Journal of Electronic Imaging* **28**(5), 053024 (2019).

[10] Gao, J., Anantrasirichai, N., and Bull, D., "Atmospheric turbulence removal using convolutional neural network," *arXiv preprint arXiv:1912.11350* (2019).

[11] Chak, W. H., Lau, C. P., and Lui, L. M., "Subsampled turbulence removal network," in [*CVPR*], **abs/1807.04418** (2018).

[12] Nieuwenhuizen, R. and Schutte, K., "Deep learning for software-based turbulence mitigation in long-range imaging," in [*Artificial Intelligence and Machine Learning in Defense Applications*], **11169**, 111690J, International Society for Optics and Photonics (2019).

[13] Tian, C., Xu, Y., and Zuo, W., "Image denoising using deep cnn with batch renormalization," *Neural Networks* **121**, 461–473 (2020).

[14] Zhang, Y., Tian, Y., Kong, Y., Zhong, B., and Fu, Y., "Residual dense network for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[15] Feng, R., Gu, J., Qiao, Y., and Dong, C., "Suppressing model overfitting for image super-resolution networks," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*], 0–0 (2019).

[16] Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., and Fu, Y., "Image super-resolution using very deep residual channel attention networks," in [*Proceedings of the European Conference on Computer Vision (ECCV)*], 286–301 (2018).

[17] Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L., "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing* **26**(7), 3142–3155 (2017).

[18] Schmidt, J. D., [*Numerical Simulation of Optical Wave Propagation with examples in MATLAB*], SPIE (2010).

[19] Hardie, R. C. and LeMaster, D. A., "On the simulation and mitigation of anisoplanatic optical turbulence for long range imaging," in [*Long-Range Imaging II*], **10204**, 102040B, International Society for Optics and Photonics (2017).

[20] Dong, W., Wang, P., Yin, W., Shi, G., Wu, F., and Lu, X., "Denoising prior driven deep neural network for image restoration," *IEEE transactions on pattern analysis and machine intelligence* **41**(10), 2305–2318 (2018).

[21] Anantrasirichai, N., Achim, A., Kingsbury, N. G., and Bull, D. R., "Atmospheric turbulence mitigation using complex wavelet-based fusion," *IEEE Transactions on Image Processing* **22**(6), 2398–2408 (2013).

[22] Zhang, K., Zuo, W., Gu, S., and Zhang, L., "Learning deep cnn denoiser prior for image restoration," in [*Proceedings of the IEEE conference on computer vision and pattern recognition*], 3929–3938 (2017).

[23] Zhang, Y., Xiang, Y., and Bai, L., "Generative adversarial network for deblurring of remote sensing image," in [*2018 26th International Conference on Geoinformatics*], 1–4, IEEE (2018).

[24] Tai, Y., Yang, J., and Liu, X., "Image super-resolution via deep recursive residual network," in [*2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], IEEE (2017).

[25] Huang, G., Liu, Z., and Weinberger, K. Q., "Densely connected convolutional networks," in [*CVPR*], **abs/1608.06993** (2017).

[26] Thakur, R. S., Yadav, R. N., and Gupta, L., "State-of-art analysis of image denoising methods using convolutional neural networks," *IET Image Processing* **13**(13), 2367–2380 (2019).

[27] Dong, C., Loy, C. C., He, K., and Tang, X., "Image super-resolution using deep convolutional networks," in [*IEEE Transactions on Pattern Analysis and Machine Intelligence*], **38**, 295–307 (2016).

[28] Timofte, R., Agustsson, E., Gool, L. V., Ming, H., and Zhang, Y. L., "Ntire 2017 challenge on single image super-resolution: Methods and results," in [*NTIRE 2017*], (2017).

[29] Timofte, R., Gu, S., Wu, J., Gool, L. V., Zhang, L., and Yang, M.-H., "Ntire 2018 challenge on single image super-resolution: Methods and results," in [*NTIRE 2018*], (2018).

[30] Timofte, R., Cai, J., Gu, S., and Zhang, L., "Ntire 2019 challenge on real image super-resolution: Methods and results," in [*NTIRE 2019*], (2019).

[31] Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi, W., "Photo-realistic single image super-resolution using a generative adversarial network," in [*2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], IEEE (2017).

[32] Repasi, E. and Weiss, R., "Computer simulation of image degradations by atmospheric turbulence for horizontal views," in [*Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XXII*], **8014**, 80140U, International Society for Optics and Photonics (2011).

[33] Schwartzman, A., Alterman, M., Zamir, R., and Schechner, Y. Y., "Turbulence-induced 2d correlated image distortion," in [*2017 IEEE International Conference on Computational Photography (ICCP)*], 1–13, IEEE (2017).

[34] Rai, S. N. and Jawahar, C., "Learning to generate atmospheric turbulent images," (2019).

[35] Hardie, R. C., Power, J. D., LeMaster, D. A., Droege, D. R., Gladysz, S., and Bose-Pillai, S., "Simulation of anisoplanatic imaging through optical turbulence using numerical wave propagation with new validation analysis," in [*Optical Engineering*], **56**, 071502–071502 (2017).

[36] McFadden, A., *Creating a Simulator of Visual Turbulence in Long Distance Imaging*, thesis (2019).

[37] Kolmogorov, A. N., "The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers," in [*Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*], **434**, 9–13 (1991).

[38] Goodman, J. W., [*Introduction to Fourier optics*], Roberts and Company Publishers (2005).

[39] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A., "Learning deep features for scene recognition using places database," in [*Advances in neural information processing systems*], 487–495 (2014).

[40] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z., "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in [*CVPR*], **abs/1609.05158** (2016).

[41] Zhang, K., Zuo, W., and Zhang, L., "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *IEEE Transactions on Image Processing* **27**(9), 4608–4622 (2018).

[42] Isogawa, K., Ida, T., Shiodera, T., and Takeguchi, T., "Deep shrinkage convolutional neural network for adaptive noise reduction," *IEEE Signal Processing Letters* **25**(2), 224–228 (2017).

[43] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image quality assessment: From error visibility to structural similarity," in [*IEEE Transactions on Image Processing*], **13**, 600–612 (2004).