On the use of Denoising Autoencoders and Deep Convolutional Adversarial Networks for Automated Removal of Date Stamps

Nicolas Anderson, Mikael Antero Paavola,
Johnny Sognnes

SUPERVISORS
Morten Goodwin, Xuan Zhang

Master's Thesis
University of Agder, 2019
Faculty of Engineering and Science
Department of ICT

UiA
University of Agder
Master's thesis

Faculty of Engineering and Science
Department of ICT

Abstract

This thesis investigates to what extent the deep learning models such as Denoising Autoencoder (DAE) and Deep Convolution General Adversarial Net (DCGAN) automate the removal of the date stamps from images with high resolution while preserving the rest of the images. Both DAE and DCGAN algorithms are implemented with Convolutional Neural Networks (CNN). The DAE algorithm can perform this task with entirely satisfactory results. The DAE can reconstruct the original images from corrupted inputs with date stamps. While DCGAN delivers poor yet interesting results. The images generated by the DCGAN are quite different from the reference images. All performed experiments in this thesis that the quality of output images produced by DAE is far superior to that of the results generated by DCGAN.

Keywords: Blind Image Inpainting, DAE, DCGAN, automated date stamp removal

Preface

On the use of Denoising Autoencoders and Deep Convolutional Adversarial Networks for automated removal of date stamps is the master thesis project done by Nicolas Anderson, Mikael Antero Paavola, and Johnny Sognnes for the course IKT590 at the University of Agder. The project was suggested by us as we are inspired by the interesting and challenging nature of blind image inpainting. It was supervised by Associate Professor Morten Goodwin and Dr. Xuan Zhang. We would like to thank our supervisors Associate Professor Morten Goodwin and Dr. Xuan Zhang for their advice, supervision and insights throughout the master thesis process.

Nicolas Anderson, Mikael Paavola and Johnny Sognnes
Grimstad, 23.05.2019

Table of Contents

Glossary

AE: Autoencoder
BCE: Binary Cross Entropy loss function
CAE: Contractive Autoencoder
CE: Context Encoder
CNN: Convolutional Neural Network
CV: Computer Vision
DAE: Denoising Autoencoder
DCGAN: Deep Convolutional General Adversarial Net
DNN: Deep Neural Network
DL: Deep Learning
GAN: Generative Adversarial Net
KSVD: A popular sparse coding technique
MAE: Mean Absolute Error
MOS: Mean Opinion Score
MSE: Mean Square Error
PSNR: Peak Signal-to-noise ratio
Relu: Rectifier Linear Unit
SSDA: Stacked Sparse Denoising Autoencoder
SSIM: Structured Similarity Index
VAE: Variational Autoencoder

List of Figures

List of Tables

Chapter 1

Introduction

Image inpainting is the method of removing the unwanted parts from an image or restoring the original image from a corrupted version such as an image with missing parts or scratches. This is typically done manually by using software tools like Adobe Photoshop, but this is time-consuming and does not scale well when dealing with large amounts of images.

In contrast to most existing inpainting techniques, blind inpainting methods deal with more challenging inpainting issues. Notably, no prior information about the original image and as well as no information about the location of the corrupted regions is given to blind inpainting algorithms. That makes blind inpainting tasks interesting to solve. Most blind inpainting techniques are built using Deep Neural Networks (DNNs). Some blind inpainting algorithms have weaknesses as they attempt to recover the image from a corrupted one by making assumptions about those corrupted regions, or they do not have enough information about those damaged locations. Whereas, other blinding algorithms do not have enough capacities to handle various types and sizes of corrupted geometry shapes or depraved high-resolution images.

This thesis focuses on solving one particular type of blind painting namely removing the date stamp from the high resolution images. This is done with using two algorithms namely Denoising Autoencoder (DAE) and Deep Convolutional Generative Adversarial Net (DCGAN).

1.1 Motivation

The primary motivation is to investigate if the deep learning models such as DAE and DCGAN could automate the task of removing the date stamps from a collection of images with high resolution. Another motivation is to discover whether DAE or DCGAN performs better for this specific task.

1.2 Goal

The goal of this thesis is to solve the task of automating the removal of the date stamps from the colored images with high resolutions while preserving the rest of the image. Particularly, this thesis focuses on developing the different deep learning models such as DAE and DCGAN to achieve the above-mentioned goal.

1.3 Problem Statement

The main issue addressed in this thesis is blind image inpainting where information about the location of date stamps on images are not given to inpainting algorithms. This makes the task of automatic removal of date stamps from the images more interesting and challenging. Additionally, the thesis explores different Deep Learning (DL) techniques and their ability to perform blind inpainting without reducing the image quality. Some of those techniques can create blurry images while some can leave missing holes or regions after date stamps are removed. The above-mentioned issue can be tackled using supervised deep learning techniques, especially DAE and DCGAN. Consequently, the desired result is that these two models learn to remove the date stamps without damaging the original quality, meaning that they output neither blurry images nor leave missing regions.

1.3.1 Research Questions

1. How can DAE and DCGAN models be used for removing the date stamps from high resolution images?

2. To what extent can DAE and DCGAN handle datasets with colored high resolution images?

3. How to evaluate objectively when the results achieved by DCGAN and DAE are compared?

1.3.2 Hypotheses

1. The DAE model can remove date stamps from colored images with high resolution.

2. The DCGAN model can remove date stamps from colored images with high resolution.

3. The output images produced after removing date stamps by DAE do not have missing regions or holes.

4. The output images produced by the DCGAN after removing date stamps do not have missing regions or holes.

5. The quality of output images produced by the DCGAN is better than that of output by the DAE or vice versa.

1.4 Assumptions and Limitations

1.4.1 Assumptions

- Enough computational power is available to train and test DAE and DCGAN models for high quality images.

1.4.2 Limitations

- The scope of the topic is huge in a sense that two different deep learning models are used for the inpainting, and a way to objectively evaluate the results achieved by both of these models is investigated.

1.5 Contributions

This thesis presents automated removal of date stamps, a particular type of blind image inpainting with the help of DAE and DCGAN algorithms. The DAE algorithm is able to process high resolution colored images. This thesis also explores the potential usefulness of DCGAN with different setups, and the possibility for the network to blindly remove date stamps. The results are of interest even if DCGAN does not produce desired results as stated by the hypotheses.

1.6 Report Outline

This Section briefly describes how the rest of the chapters in this report is organized.

- **Chapter 2**: Describes the various concepts and theories such as inpainting, blind inpainting, different types of Autoencoders (AEs) and General Adversarial Network (GAN).
- **Chapter 3**: Mentions distinct state-of-the art techniques.
- **Chapter 4**: Explains about the datasets, tools, the proposed solutions for DAE and DCGAN to remove the date stamps from the colored images with high resolution.
- **Chapter 5**: Presents various results of DAE and DCGAN , discusses and evaluates them.
- **Chapter 6**: Includes the conclusion for the thesis, the suggestion for improvements and suggestion of this thesis extension.

Chapter 2

Theoretical Background

This chapter presents summary knowledge about deep learning models such as AE and GAN and image inpainting.

2.1 Image Inpainting

Corruption in images can occur through a plethora of different reasons. These include noise introduced through acquisition channels such as Gaussian noise, to physical damages on photographs such as scratches, wear, and tear, to overlaid text or other graphics introduced by artificial editing. The goal of image restoration techniques is to recover an uncorrupted image from a noisy observation of it. Image inpainting, a type of image restoration, can be used to repair missing pixel values, such as holes, scratches and missing regions, or to remove intricate patterns such as text or other graphics placed on the image through artificial editing. A necessity of image inpainting is to have a mask that detects the locations of an image in which inpainting is required. The central idea of the image inpainting algorithms is to fill in the corrupted, missing regions with accessible information from their neighbors and eradicate the undesired objects. The goal is to modify a corrupted image in an undetectable way so that an observer is unable to observe that the image used to be noisy and has been restored. However, it is unfeasible without prior information of that image [1].

Image inpainting and denoising are typical image recovering problems that are both convenient by themselves and significant preprocessing steps of many other applications. Image denoising issues appear when an image is corrupted by additive white Gaussian noise which is typical result of many acquisition channels, while image inpainting issues arise when some pixel values are missing or when

we want to get rid of more practical patterns such as superimposed text or other objects from the image. Image inpainting methods can be split into two classes, namely non-blind inpainting and blind inpainting. In non-blind inpainting, the corrupted regions that need to be fixed are provided to the algorithm as a priori, while in blind inpainting, no information about the corrupted regions is given and the algorithm must automatically analyze the pixels that need inpainting [2].
A set of different inpainting and blind inpainting techniques are demonstrated in figures 2.1, 2.2, 2.3.

Figure 2.1: Image with scratch before and after inpainting [3].

Figure 2.2: Image with random superimposed text before and after inpainting [2].

Figure 2.3: Image with random holes before and after inpainting [4].

2.2 Autoencoders

Autoencoders are unsupervised neural networks that use backpropagation setting the output values to be identical to inputs, i.e., it uses $y^{(i)} = x^{(i)}$. The general structure of an autoencoder can be seen in Figure 2.4 where +1 refers to bias units, $LayerL_1$ represents the input layer via Encoder, $LayerL_2$ represents the hidden layer h via bottleneck and $LayerL_3$ represents the output layer via Decoder for the reconstruction of inputs [5].

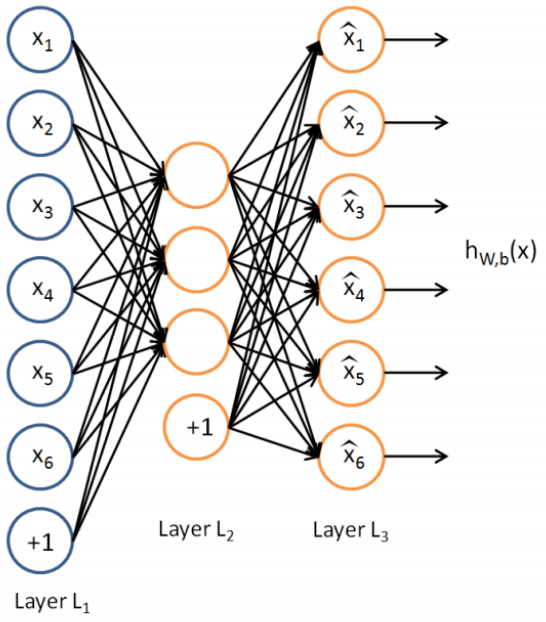


Figure 2.4: The structure of an autoencoder [5].

It contains two components namely an encoder function $h = f(x)$ and a decoder that represents a reconstruction $r = g(h)$. If an autoencoder accomplishes in learning to set $g(f(x)) = x$ everywhere, then it is not actually useful. Autoencoders are rather designed to be unable to learn to copy perfectly. Mostly they are limited in ways that let them to replicate only approximately, and to duplicate only input that features the training data. Since the model is compelled to prioritize which aspects of the input should be replicated, it often learns useful properties of the data. Autoencoders can be used for dimensional reduction or feature learning and information retrieval tasks [6].

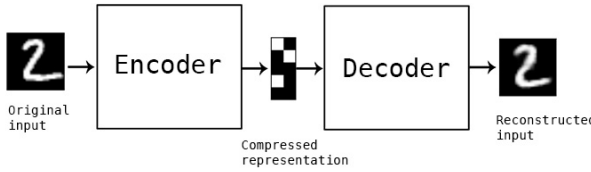


Figure 2.5: How an autoencoder works [7].

There exists a variety of autoencoders and among them, undercomplete autoencoder, sparse autoencoder, denoising autoencoder, contractive autoencoder and variational autoencoder are well-known. They are further described in the coming chapters.

2.2.1 Undercomplete Autoencoder

An autoencoder that has a smaller code dimension in the bottleneck than its input dimension is called **undercomplete**. The bottleneck layer is the layer with smallest code dimension. Learning an undercomplete representation forces the autoencoder to capture the most notable features of the training data. The learning process is simply expressed as minimizing a loss

$$L(x, g(f(x))) \tag{2.1}$$

in which $\mathbf{L}$ is a loss function that penalizes $\mathbf{g(f(x))}$for being different from $\mathbf{x}$, such as the mean squared error.

An autoencoder learns to extend the identical space as PCA (Principal Component Analysis) when the decoder is linear and $\mathbf{L}$ is the mean squared error. In this state, an autoencoder trained to carry out the replicating task has learned the principal subspace of the training data as a side effect. Therefore, autoencoders

with nonlinear encoder function f and nonlinear decoder functions g can learn a more effective nonlinear generalization of PCA. The difference between these two approaches can be viewed in the Figure 2.6.

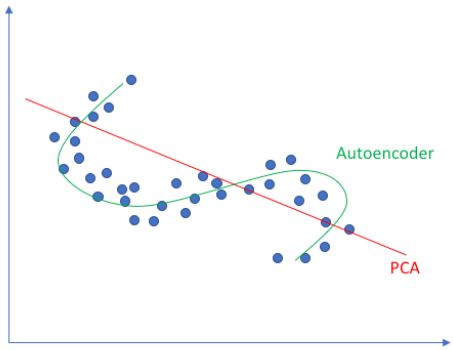


Figure 2.6: Linear vs nonlinear dimensionality reduction [8].

If the encoder and decoder of an autoencoder is given too much capabilities, it will learn to execute the copying task without extracting useful information about the distribution of the data. Thus, it is necessary to put constraints on autoencoders. The objective of constraining the capacity of an autoencoder is achievable in a variety of ways by adding different regularization terms which will be explained further in the coming chapters. However, an undercomplete autoencoder is designed in such a way that it does not require any regularization term as it has smaller code dimension than its input dimension [6].

2.2.2 Sparse Autoencoder

A sparse autoencoder is an autoencoder whose training contains a sparsity penalty $\Omega(\boldsymbol{h})$ on the hidden layer $\boldsymbol{h}$ in addition to the reconstruction error:

$$L(x, g(f(x))) + \Omega(h) \tag{2.2}$$

in which $g(\boldsymbol{h})$ is the decoder output, and $\boldsymbol{h} = f(x)$, is the encoder output. Furthermore, the sparsity constraint can further be defined as

$$\Omega(h) = \lambda \sum_i |h_i| \tag{2.3}$$

where λ is a hyperparameter and i is the index of the layer.

Sparse autoencoders are usually used to learn features for another task like classification. An autoencoder with sparse regularization must respond to particular statistical features of the dataset it has been trained on, rather than merely performing as an identity function. In such a way, training to perform the replicating task with a sparsity penalty can give a network that has learned useful features as a result. The penalty term $\Omega(\boldsymbol{h})$ can be considered as a regularizer term that is added to a feedforward network whose main task is to reconstruct the output which is similar to the input (unsupervised learning objective) and perhaps also perform some supervised task that depends on these sparse features [6].

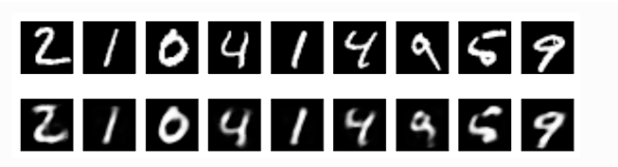


Figure 2.7: Images generated by a sparse autoencoder [7].

In Figure 2.7, the top row represents the original digits from the original MNIST dataset, which is used as input and the bottom row depicts the reconstructed digits generated by the sparse autoencoder. A more in-depth discussion on sparse autoencoders is presented by Goodfellow [6] and Andrew Ng [5].

2.2.3 Denoising Autoencoder (DAE)

The **denoising autoencoder (DAE)** is an autoencoder that uses a corrupted data point $\widetilde{\mathbf{x}}$ as input and is trained to recover the original, uncorrupted data point $\mathbf{x}$ as its output. That is illustrated in the Figure 2.8.
The loss function of the DAE can be described as

$$L(x, g(f(\widetilde{x}))) \tag{2.4}$$

in which $\widetilde{\mathbf{x}}$ is a replica of $\mathbf{x}$ that has been corrupted by some type of noise [6]. A deeper discussion on sparse autoencoders is presented by Goodfellow [6].

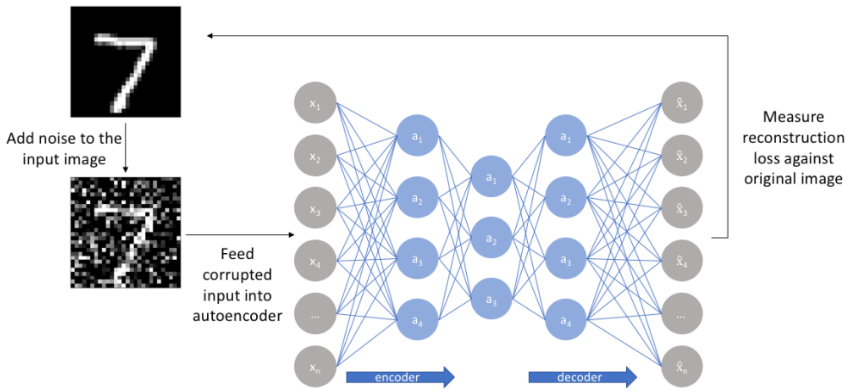


Figure 2.8: A denoising autoencoder [8].

2.2.4 Contractive Autoencoder (CAE)

The contractive autoencoder (CAE) model presents an explicit regularization on the code $\mathbf{h} = f(x)$, encouraging the derivatives of f to be as small as attainable:

$$\Omega(\mathbf{h}) = \lambda \left\| \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right\|_F^2 \tag{2.5}$$

The penalty $\Omega(\mathbf{h})$ is the squared Frobenius norm (sum of the squared elements) of the Jacobian matrix of partial derivatives related to the encoder function. The loss function of the CAE model is defined as:

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \lambda \sum_i \|\nabla_x h_i\|^2 \tag{2.6}$$

where λ is a hyperparameter and $\nabla_x h_i$ is the gradient field of hidden layer activations with respect to input x, summed over all i training samples.

This penalty $\Omega(\mathbf{h})$ causes the CAE network to learn a function that does not alter much when x changes to some extent. Since this penalty is used only at training examples, the CAE is forced to learn features that capture information about the training distribution. The name **contractive** emerges from the way that the CAE folds space. Since the CAE is trained to withstand perturbations of its input, it is encouraged to map a neighborhood of input points to a smaller neighborhood of output points. One can think of this as contracting the input neighborhood to a smaller output neighborhood [6].

2.2.5 Variational Autoencoder (VAE)

The VAE is a form of autoencoder that uses learned approximate inference and can be trained entirely with gradient-based methods. In order to generate a sample from the network, the VAE picks a sample **z** from the code distribution $p_{model}(x|z)$. Then, x is sampled from a distribution $p_{model}(x; g(z)) = p_{model}(x|z)$. Whereas the approximate inference network (or encoder) $q(z|x)$ is deployed to get **z** during training, and $p_{model}(x|z)$ is then viewed as a decoder network. This can be seen in Figure 2.9.

The key concept behind the VAE is to train a parametric encoder (an inference network or recognition model) which produces the parameters of q. Considering **z** is a continuous variable, one can back-propagate through samples of z chosen from $q(z|x) = q_{model}(z; f(x; \theta))$ to achieve a gradient with respect to θ. Learning then involves $\mathcal{L}$ (the variational lower bound) with respect to the parameters of the encoder and decoder. All the expectations in $\mathcal{L}$ may be estimated by Monte Carlo sampling [6].

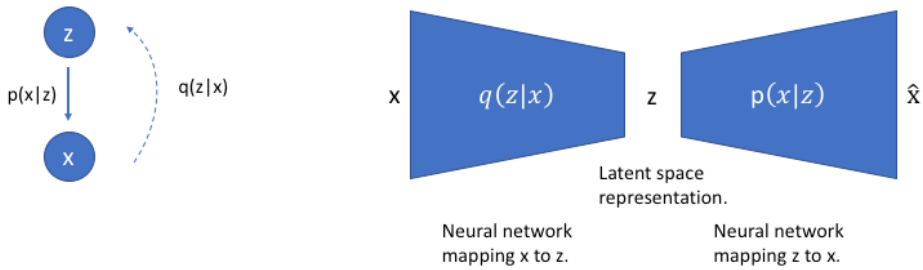


Figure 2.9: the network architecture of a vairational autoencoder [9].

2.3 Generative Adversarial Networks

GANs are based on a game-theoretic scheme where a generator network compete against a network called the discriminator. The generator network generates samples $x = g(z; \theta^{(g)})$. While the discriminator network differentiates between samples taken from the training data and samples chosen from the generator network. The discriminator emits a probability value given by $d(x; \theta^{(d)})$, pointing the probability that x is a real training sample rather than fake example picked from

the generator. This process is illustrated in Figure 2.10. The most straightforward way to define learning in GANs is as a zero-sum game where a function $v(\theta^{(g)}, \theta^{(d)})$ decides the payoff of the discriminator. The generator gets $-v(\theta^{(g)}, \theta^{(d)})$ as its own payoff. Both the generator and discriminator networks seek to maximize their payoff, so that at convergence

$$g* = \arg\min_{g} \max_{d} v(g, d) \tag{2.7}$$

The default choice for v is

$$v(\theta^{(g)}, \theta^{(d)}) = \mathrm{E}_{x \sim p_{data}} \log d(x) + \mathrm{E}_{x \sim p_{model}} \log(1 - d(x)) \tag{2.8}$$

This motivates the discriminator to classify samples as real or fake precisely. Concurrently, the generator tricks the discriminator into believing its examples are positive. At convergence, samples generated by the generator are identical with the actual data, and the discriminator yields $\frac{1}{2}$ in all places. Then, the discriminator may be discontinued. The major motive for introducing GANs is that the learning process needs neither estimate inference nor estimate of a partition function gradient. When $max_d v(g, d)$ is convex in θ^g, the method is guaranteed to converge and is asymptotically consistent. However, if $max_d v(g, d)$ is not convex and g and d are defined by neural networks, the nonconvergence emerges as a problem which further causes GAN to underfit.

Alternative approach to **zero-sum** introduced by Goodfellow (2014) involves the **maximum likelihood**. As maximum likelihood training converges, the reformulation of the GAN should also converge given enough samples. Whereas, this new formulation cannot improve convergence in reality, perhaps due to suboptimality of the discriminator or high variance around the expected gradient.

Unlike **zero-sum** or equivalent to **maximum likelihood**, the best-performing formulation of the GAN game in practical experiments is different. In that best-performing formulation brought in by Goodfellow et al. (2014c), the generator intends to enhance the log-probability that the discriminator makes an error rather than focusing to reduce the log-probability that the discriminator makes the correct prediction. This motivation is derived from the observation that it causes the derivative of the generator's cost function with respect to the discriminator's logits to last large even in the situation when the discriminator confidently dismisses all generator examples. Even though stabilization of GAN learning is still an open problem, GAN learning carries out well when the architecture and hyperparameters of the model are cautiously selected [6].

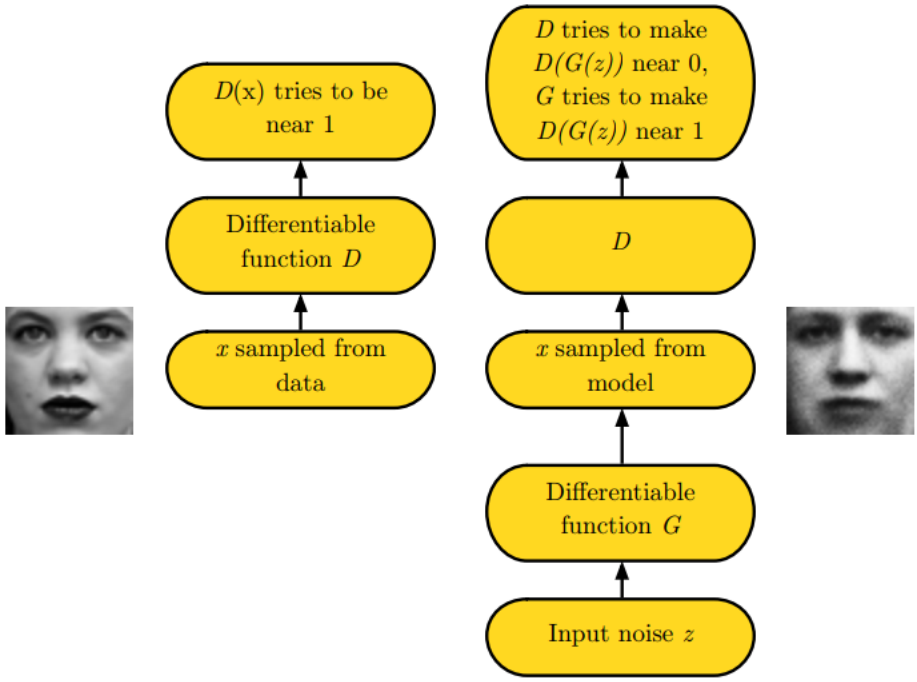


Figure 2.10: The GAN framework pits two adversaries against each other in a game [10].

Chapter 3

State of the Art

The performance of image inpainting and blind image inpainting algorithms steadily and progressively improve as the advancements in deep learning enhance. The recovering of images that are imposed by scratches, superimposed text, or graphics by these algorithms are becoming as good as their human counterparts in these recent years. Some of the remarkable researches that remove the unwanted text or objects can be seen in the sub-chapters below.

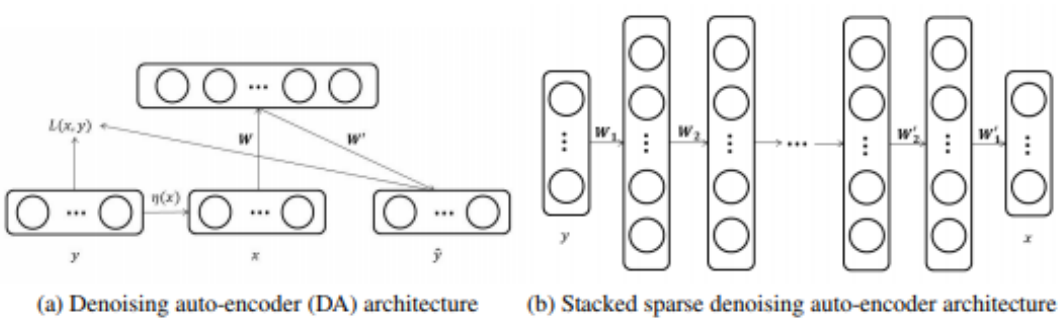
3.1 Image Denoising and Inpainting with Deep Neural Networks

Xie et al. [2] present a novel approach to low-level vision problems that fuses sparse coding and deep networks pretrained with DAE. The proposed approach can perform image denoising and complex blind inpainting tasks and can yield the results that are comparable to that of KSVD, a popular sparse coding technique. The proposed model in this research is known as Stacked Sparse Denoising Autoencoders (SSDA), and the architecture of the network is illustrated in Figure 3.1. A visual comparison of inpainting results between SSDA and KSVD can be seen in Figure 3.2.

SSDA can automatically remove complicated patterns like an overlaid text from an image, rather than simple patterns like pixels missing at random. Besides, it does not need the location that requires inpainting to be given a priori.

Experimental results indicate the effectiveness of SSDA in the denoising and blind inpainting works and also SSDA can improve the performance of unsupervised fea-

ture learning. This research [2] also proposes a new training scheme for DAE that can both denoise and inpaint images within a unified framework. The weakness with SSDA is that it can remove only the noise patterns it has seen in the training data. In other words, it can generalize to unseen but similar noise patterns [2].



(a) Denoising auto-encoder (DA) architecture (b) Stacked sparse denoising auto-encoder architecture

Figure 3.1: Model Architectures [2].

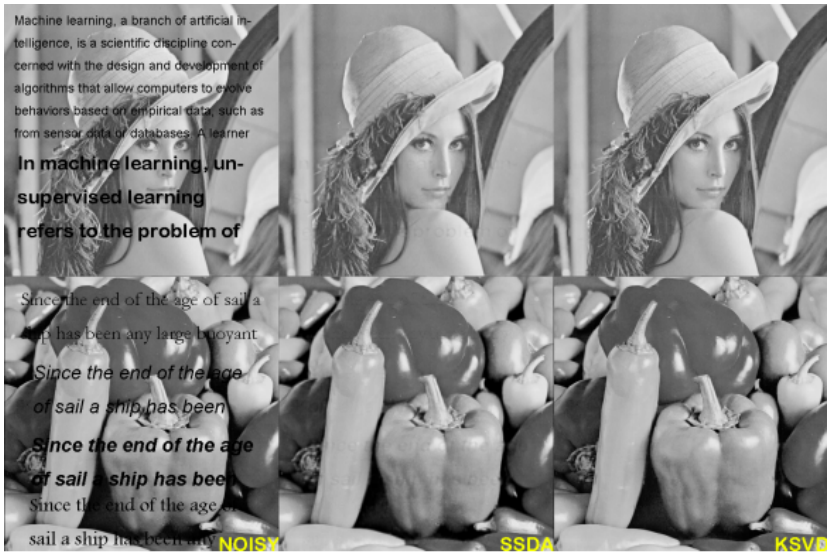


Figure 3.2: Visual comparison of inpainting results from Noisy, SSDA and KSVD. [2].

3.2 Deep Convolutional Generative Adversarial Nets

The adoption of supervised learning with Convolutional Neural Networks (CNNs) in computer vision (CV) applications has been popular during recent years while unsupervised learning with CNNs within the CV applications domain is less preferred. Motivated by that, Alec Radford et al. [11] introduced a new GAN approach built on CNN called DCGAN in order to bridge the gap between the success of CNNs for supervised learning and unsupervised learning. They also proposed a more stable set of architectures for training GANs, and they also yielded convincing evidence that adversarial networks learn useful representations of images for supervised learning and generative modeling. They also made an important observation which indicates that as models are trained longer they sometimes collapse a subset of filters to a single oscillating model [11].

Figure 3.3: Generated bedrooms by DCGAN model after one training pass through the LSUN dataset [11].

3.3 Deep Blind Image Inpainting

Liu et al. [12] introduce an efficient blind image inpainting algorithm to restore a clear picture from a corrupted input directly. Inspired by the residual learning algorithm, Deep blind image inpainting [12] involves an encoder and decoder architecture to capture the more useful information and formulate a robust loss

function to handle with outliers. For the network architecture of the proposed model, one can view Figure3.4.

The deep blind image inpainting algorithm can anticipate the missing information in the corrupted regions, which improves image restoration. Consider Figure 3.5 in which their method directly learns the missing information in the corrupted locations. By plugging the learned information into an input image, realistic images can be achieved.

Both qualitative and quantitative experiments indicate that the deep blind image inpainting algorithm can handle the corrupted regions of arbitrary shapes, and it can also execute positively against state-of-the-art techniques. However, the deep blind image inpainting technique does not perform well when significant structures or details that are unique in an image is corrupted. Figure 3.6 indicates a failure sample in which the nose and mouse are densely corrupted, and the proposed method fails to restore these two parts. Notably, their approach does not execute well when unique structures are corrupted by large square[12].

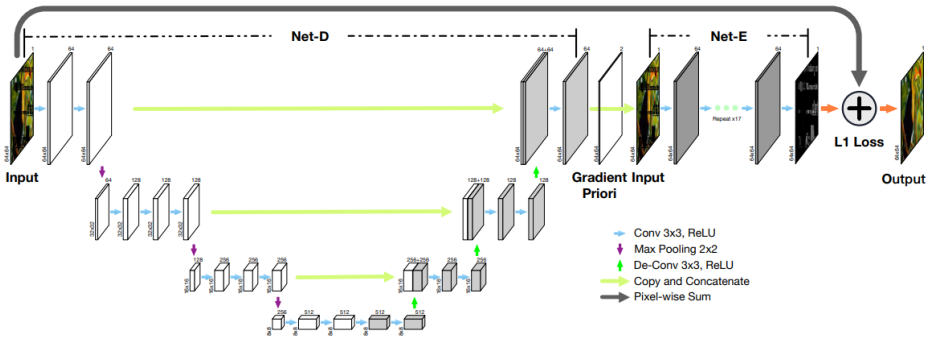


Figure 3.4: Network Architecture of the proposed model [12].

(a) Corrupted image (b) Our restoration result (c) Learned residual image

Figure 3.5: Image recovering using the deep blind image inpainting algorithm [12].

(a) Input (b) Output

Figure 3.6: A failure example of the deep blind image inpainting technique [12].

3.4 Image Inpainting and Object Removal with Deep Convolutional GAN

Fu et al. [13] present solving landscape picture inpaint problem with the DCGAN. The architecture of DCGAN in this project is formed with two networks, namely the Generator and the Discriminator, which can be seen in Figure 3.7 and Figure 3.8, respectively. Two reasons mainly motivate them. The first reason is to remove the undesired objects from the taken pictures, and the second reason is to explore the recently published DCGAN on image inpainting.

According to Fu et al.[13], the resulting model beats most of the existing published solutions so far. A comparison between the original image, the cropped image, and the reconstructed images produced by the Telea inpainting algorithm and the DCGAN algorithm is shown in Figure 3.9.

There were two major challenges while the research[13] was conducted. One of the challenges that made DCGAN difficult to train was that the discriminator and the generator learn with different speeds. To get good training results, both discriminator and generator have to be tuned so that they both are learning at the same pace. Another challenge was the high cost in both time and memory for the training of DCGAN. The GAN model used around 50 minutes to train 1 epoch, and it typically takes about 40 epochs to get decent results. Additionally, memory overhead is also quite immense. CUDA crashes from time to time, which leads to wasted training efforts[13].

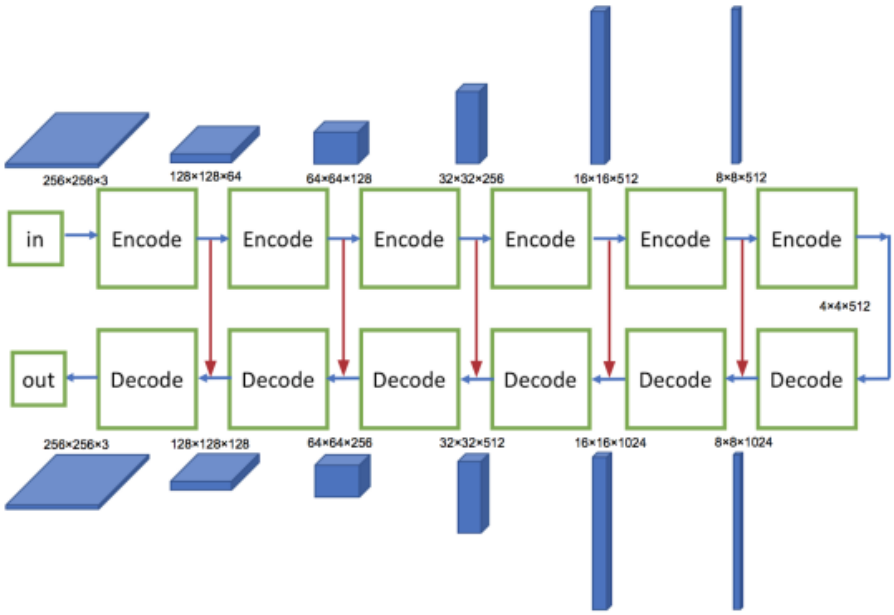


Figure 3.7: Network Architecture of the Generator [13].

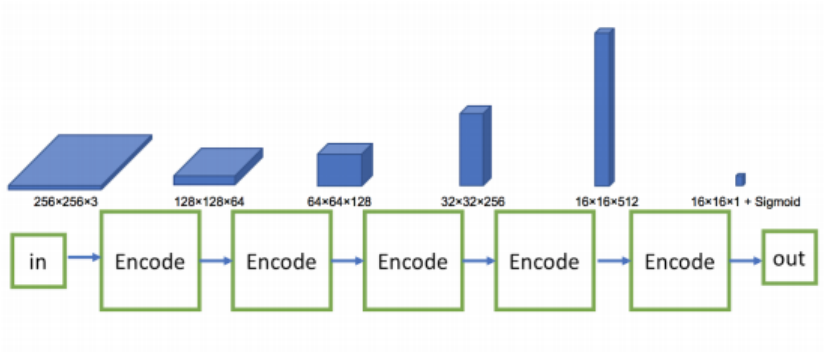


Figure 3.8: Network Architecture of the discriminator [13].

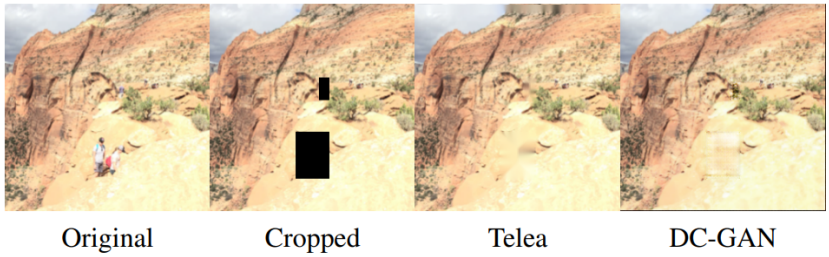


Figure 3.9: Image Yuxin, one of the authors of this research, took in Zion mountain [13].

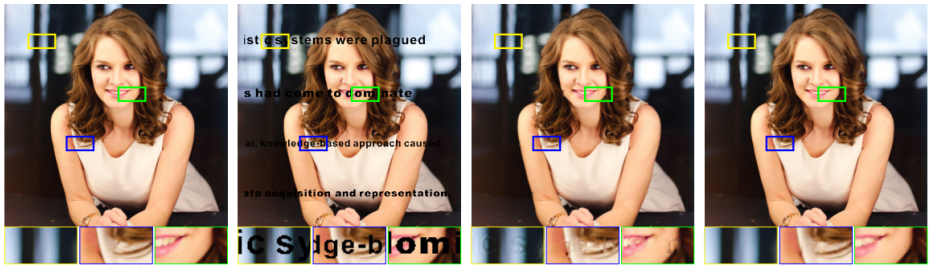
3.5 Deep Image Prior

Ulyanov et al. [14] presents that the structure of a generator network is adequate to capture a vast amount of low-level image statistics before any learning. To do that, they show that a randomly-initialized neural network can be applied as a handcrafted prior with excellent results in standard inverse problems like denoising, super-resolution, and inpainting. According to Ulyanov et al. [14], the same prior can also be adapted to invert deep neural representations to diagnose them and to recover images based on flash-no flash input pairs.

Deep Image Prior algorithm [14] focuses on the inductive bias captured by standard generator network architectures in addition to its diverse applications. It also

links the gap between two popular image restoration methods such as learning-based methods using deep convolutional networks and learning-free methods based on handcrafted image priors such as self-similarity.

The architecture of the network can be found in the supplementary material on their website[14]. The comparison of text image inpainting sample achieved by the Deep Image Prior and Shepard networks can be seen in Figure 3.10 [14].



(a) Original image (b) Corrupted image (c) Shepard networks [27] (d) Deep Image Prior

Figure 3.10: Text inpainting comparison between Deep Image Prior and Shepard Networks [14].

3.6 Context Encoders

Pathak et al. [15] present an unsupervised visual learning algorithm driven by context-based pixel prediction. This algorithm is called Context Encoders (CEs) where a convolutional neural network is trained to generate the contents of an arbitrary image region conditioned on its vicinity. To conduct successfully at this task, CEs need to both comprehend the content of the whole image, as well as produce a possible hypothesis for the missing part(s). While training CEs, the authors of this research experimented with a standard pixel-wise reconstruction loss, reconstruction, plus adversarial loss. The latter produced much better results since it can better deal with multiple modes in the output. They also discovered that a CE learns a representation that captures not just appearance but also the semantics of visual structures. The effectiveness of CEs' learned features for CNN pretraining was quantitatively demonstrated on classification, detection and segmentation tasks. Additionally, CEs can be applied for semantic inpainting tasks, either stand-alone or as initialization for non-parametric techniques. The Qualitative illustration of the inpainting task is shown in Figure 3.11 where an input image is given with a mission region (a), a human artist is inpainting it in (b), automatic inpainting using proposed context encoder trained with L2 reconstruction

loss in (c), and using both L2 and adversarial loss in (d) [15].

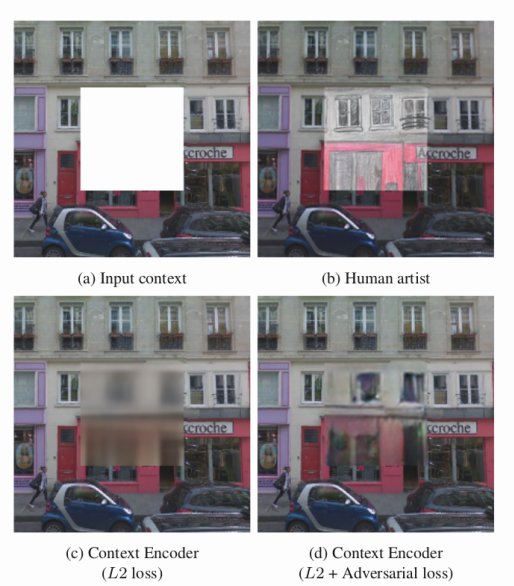


Figure 3.11: Qualitative illustration of the inpainting using context encoders [15].

3.7 Evaluation Methods

The quality of an image changes over time due to various distortions such as processing, compressing, and transmitting. There are two methods to evaluate image quality, and they are called subjective and objective methods. The subjective evaluation technique is considered as capital-intensive since it demands many observers and their opinion based judgments on large image dataset. Consider Mean Opinion Score (MOS), a popular approach for subjective image quality measurement. In MOS, a group of people is requested to compare original and corrupted images to predict the quality of the damaged image. The mean score is used as the image quality index. Even though this process reflects human perception, it is considered to be too time-consuming and unrealistic to apply compared to other image processing algorithms [16].

Whereas the objective evaluation technique applies automatic algorithms to check the quality of an image without human interference. These objective approaches are split into different classes based on the availability of the original image. They are known as **full-reference** where the original image is available, **reduced-**

reference where the original image exists partly in a group of extracted features as information that helps in the evaluation process, **no-reference** where there is no original image. This is also known as "blind quality assessment". Mean square error (MSE), peak signal-to-noise ratio (PSNR), and Structural Similarity Index (SSIM) are the most commonly used objective image quality measures[16].

Full-reference quality measures focus on original, corrupted and reconstruction images. Full reference image quality measures could be categorized into six classes of objective image assessment measures according to [16], that is

1. **Pixel difference-based measures** such as MSE, signal-to-noise ratio (SNR) and PSNR are simple to calculate.

2. **Correlation-based measures** where correlation is applied to calculate the difference between two digital images. In image quality assessment, correlation of pixels is used as a measure of the image quality.

3. **Edge-based measure** in which the edges in the original and the corrupted images are found, and then a measure of displacement of edge positions or there consistency are deployed to discover the image quality for the whole image.

4. **Spectral distance-based measures** where Discrete Fourier Transform is deployed on the original and the damaged images. Here, the difference of the Fourier magnitude or phase spectral is used as a measure of image quality,

5. **Context-based measures** where instead of comparing pixels in original and damaged images, pixel neighborhoods are compared against each other by finding the mutli-dimensional context probability to use it for calculating image quality.

6. **Human Visual System-based measures (HVS)** in which image quality is computed as the human eye would do. Typically, humans use contrast, color and frequency changes in their measures.

Chapter 4

Methodology

This chapter presents an overview of the dataset and tools used in this research and the proposed solution for DAE and DCGAN.

4.1 Dataset

The Food-11 dataset is an open source dataset available for research purposes only. It includes 16643 food images in 11 major food classes. Those 11 classes are Bread, Dairy product, Dessert, Egg, Fried food, Meat, Noodles/Pasta, Rice, Seafood, Soup, and Vegetable/Fruit. The images in the downloaded dataset are stored in three folders; training, validation, and evaluation. [17].
The images are all in excellent quality, and they are all colored and in high resolution. The dataset was chosen for this reason.

4.2 Proposed Solution

The proposed solution for removing the date stamps from the colored, high-resolution images involves the entire workflow of the experiments. The workflow is visualized in a model that gives a clear overview of the process, as in Figure 4.1. This workflow is formed with four major components namely **data pre-processing** that explains the corruption of dataset, the **network design** of DAE and DCGAN models, **image restoration** that defines the training and testing of DAE and DCGAN networks and finally **evaluation of the proposed solution** that further defines the visual and objective via quantitative evaluation of the results achieved in this thesis.

The proposed solution for DAE and DCGAN models are described in separate workflows in the sections 4.3 and 4.4, describing in detail the workflow for each algorithm.

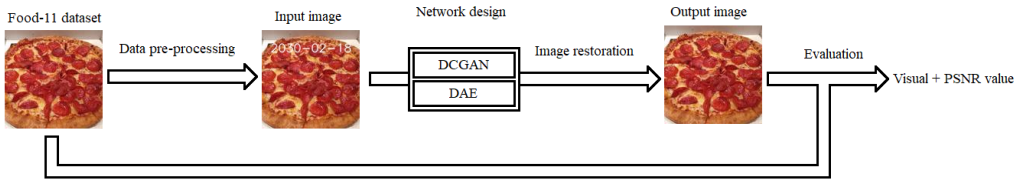


Figure 4.1: The proposed workflow

4.3 Proposed Denoising Autoencoder

This section explains how the DAE model can be designed to remove the date stamps from the high-resolution images. The section consists of the dataset corruption, network architecture of the DAE, how it is trained, and evaluation of the output images. This proposed DAE is used to conduct DAE Experiment.

4.3.1 Data Pre-processing

The data pre-processing for the DAE model involves resizing and adding corruption to the Food-11 dataset. No normalization is done on the data before training.

Resizing: The Food-11 dataset contains images of varying dimensions. The images vary from 0.05 to 56 megapixels. For these experiments, the images are cropped to a square, then resized to 256 to 256. This size is chosen for two reasons. First, due to the length and height of the images, parameters that are halved in the pooling layers, are a power of 2. Second, this resolution size is high enough to notice details and sharpness. Only 3 images in the original dataset have length or height smaller than 256, and they are not used in the experiments.

Corruption: The date stamps used as an overlay on the Food-11 dataset are supposed to imitate as the actual date stamps. The OpenCV library [18] is used to insert date stamps onto the images within the Food-11 dataset. Each image within this dataset is given a random date with white font color. The date stamp has a random font type, size, and location, so the variation in the corruption of the image will increase. The images with date stamps are considered corrupted

and are saved in separate folders. Some of the corrupted images can be seen in Figure 4.2.

Figure 4.2: Some of the corrupted data samples that were used as the input data for the proposed models in this research

4.3.2 Network Architecture

The proposed DAE model uses corrupted data as training samples and the ground truth images as labels. The desired outcome is an image with quality as close to ground truth as possible. The architecture of one of the proposed DAE models can be seen in Figure 4.3, which has 10 layers in total. Figure 4.3 is generated using [19]. The first 4 layers are called encoder and the middle layer via the bottleneck of the DAE model is formed with two layers. The last 4 layers are the decoder.

Throughout the entire network, batch normalization [20] and relu (rectifier linear unit) activation function [6] was used between the convolution [6] and max pooling [6] layers and the convolution and upsampling [6] layers. The output of the final layer is fitted with a sigmoid activation function [6]. The "same padding" and a batch size of 16 were used through the whole DAE network.

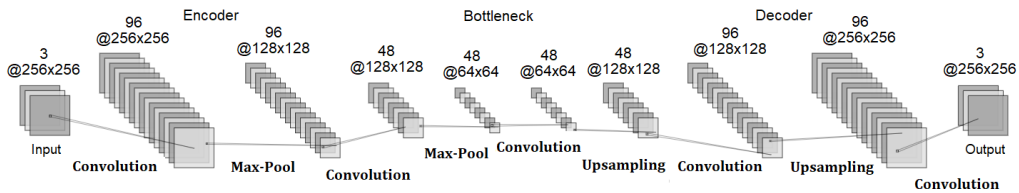


Figure 4.3: The network architecture of the DAE model.

Number Suffix layers	Types of layers	Output Shapes
1^{st}	Input	(256, 256, 3)
2^{nd}	Convolution	(256, 256, 96)
3^{rd}	Max pooling	(128, 128, 96)
4^{th}	Convolution	(128, 128, 48)
5^{th}	Max Pooling	(64, 64, 48)
6^{th}	Convolution	(64, 64, 48)
7^{th}	Upsampling	(128, 128, 48)
8^{th}	Convolution	(128, 128, 96)
9^{th}	Upsampling	(256, 256, 96)
10^{th}	Convolution/Output	(256, 256, 3)

Table 4.1: The description of network layers of DAE model

The 1^{st} layer of the 10 layered network takes the input of a $256*256$ corrupted and colored image from the Food-11 dataset. The 2^{nd} layer encodes the image with convolution and has 96 filters. The 3^{rd} layer reduces the filter sizes to $128 * 128$ with max pooling. The 4^{th} convolution layer with 48 filters encodes further the output from the 3^{rd} layer.

The two layers that are the bottleneck of the DAE namely the 5^{th} max pooling and 6^{th} convolution have the same number of filters, 48, and sizes are with the 5^{th} layer of the encoder part further encoded into the size of $64 * 64$.

The decoding process is the reverse of the encoding process. The 7^{th} layer increases the filter size into $128*128$ with upsampling. The 8^{th} layer decodes the image with convolution. The 9^{th} layer increases the filter size into $256 * 256$ with upsampling. The final convolution layer of DAE decodes the result of the previous layer back to original colored images without date stamps. Those output images are the reconstructed images. A summarized description of all network layers of the proposed DAE model can be seen in the table 4.1.

4.3.3 Image Restoration

This Section is split into two subsections that further define the training and testing workflow of the DAE model.

Training

The code implementing the DAE is based on existing DAE code [21]. The training of the DAE model is described in a workflow in Figure 4.4. The DAE network takes the images from the training folder of the corrupted Food-11 dataset as input images and learns to remove the date stamps by using the images from the training folder of the original Food-11 dataset as reference. Adam [22] is used as an optimizer, with the parameters of the optimizer set to default. Binary cross entropy [23] is used to measure the training loss. Validation is done after each epoch using the images from the validation folder. The training folders contain 9860 images each. The validation folders contain 3428 images each.

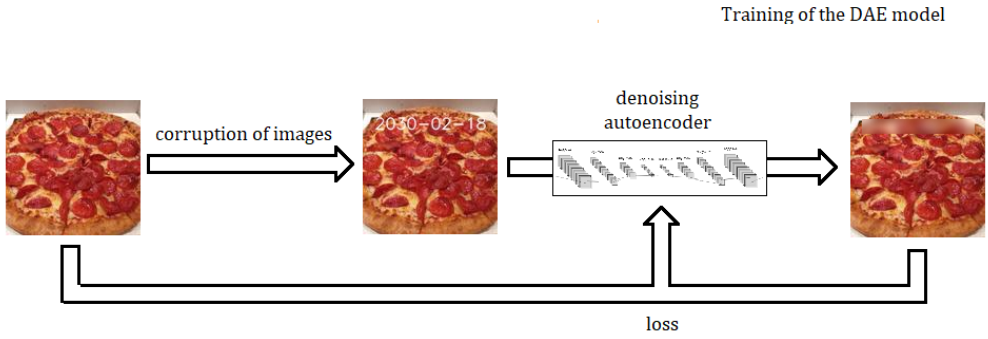


Figure 4.4: The training workflow of DAE

Testing

Figure 4.5 shows the testing workflow of the DAE model. After the DAE model is trained to reconstruct images and remove date stamps, it is tested using the images in the evaluation folder of the Food-11 dataset. This folder contains 3347 corrupted images. These images are reconstructed using the trained DAE model. After the DAE is tested, the quality of the results produced by the DAE model is further evaluated visually and quantitatively by using PSNR. The details of the evaluation approach are presented in Section 4.5.

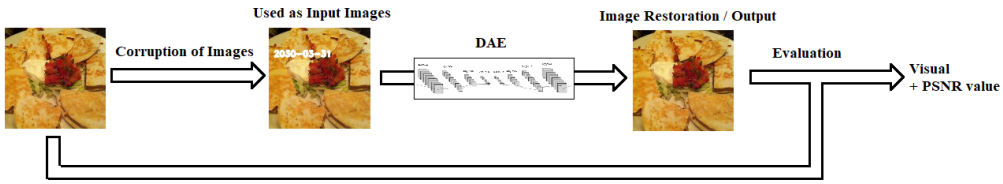


Figure 4.5: The testing workflow of DAE

4.4 Proposed Deep Convolutional General Adversarial Net

This section contains the preprocessing needed for the DCGAN model, its network architecture, the training procedure and the evaluation of the output images. This section also explains how the DCGAN model can be designed to remove date stamps from high-resolution images. Three experiments are done using the DCGAN model, each with different architectures or inputs:

1. DCGAN with date stamps Experiment.
2. DCGAN without date stamps Experiment.
3. DAE and Discriminator Experiment.

4.4.1 Data Pre-processing

The corrupted images introduced in the Section for DAE 4.3.1 are used as input images for both the DAE and DCGAN models. The images from the dataset are downsized further to 64x64 size, to reduce training times.

Labeling: The file name of the images in the dataset is of format "xx_yyyy.jpg", with "xx" being the category of the image. In Experiment 2 and 3, the categories are used to separate the images so that only one category is used. This is done to get less variation in the dataset, because it is suspected that a dataset with large variation will be more challenging to train. The image category is chosen before training. The output images of the generator are based on the specified category. The "Bread" class of the Food-11 dataset is the category that is used in these experiments.

Augmentation: Data augmentation is performed on the dataset by flipping each image, so both the original and the flipped image is in the dataset. This new dataset is used for experiment 3.

4.4.2 Network Architecture

The architecture of the DCGAN model is similar to the original GAN model [24] and uses convolutional neural networks to transform the layers inside the network. There are no pooling layers in the network, and the pixel dimensions of the image are reduced with convolution layers and "Same" padding. A batch size of 32 is used.

Figure 4.6 shows the proposed architecture of the discriminator of the DCGAN model. Figure 4.7 shows the proposed architecture of the generator of the DCGAN model. Both figures are generated using [19].

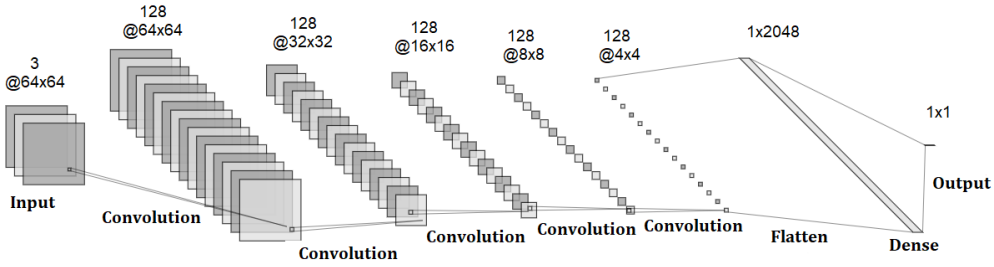


Figure 4.6: The network architecture of the Discriminator network of the DCGAN model.

Table 4.2 shows a summarized description of all network layers of the proposed discriminator of DCGAN network. Throughout the discriminator network, batch normalization with momentum of 0.9 and LeakyReLU activation function with alpha value 0.1 was used as hyper-parameters between the convolution and convolution layers. The dropout function is used between the 7^{th} flatten layer and the dense output layer of the network. The final dense layer is fitted with a Sigmoid activation function.

Number Suffix layers	Types of layers	Output Shapes
1st	Input	(64, 64, 3)
2nd	Convolution	(64, 64, 128)
3rd	Convolution	(32, 32, 128)
4th	Convolution	(16, 16, 128)
5th	Convolution	(8, 8, 128)
6th	Convolution	(4, 4, 128)
7th	Flatten	(1, 2048)
8th	Dense/Output	(1, 1)

Table 4.2: The description of network layers of Discriminator network of the proposed DCGAN model

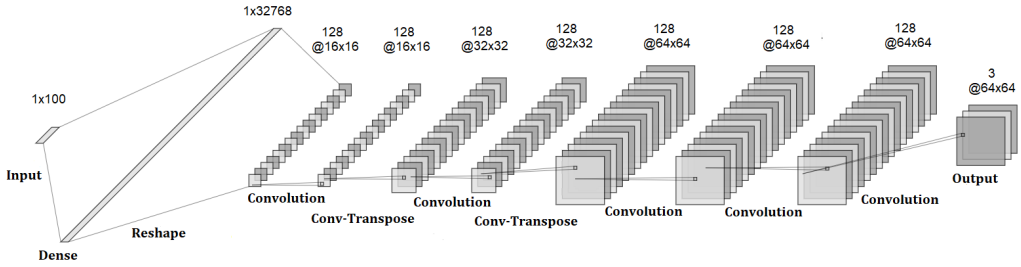


Figure 4.7: The network architecture of the Generator network of the DCGAN model.

Table 4.4.3 shows a summarized description of all network layers in the proposed generator of the DCGAN network. Throughout the generator network, batch normalization with momentum of 0.9 and LeakyReLU activation function with alpha value 0.1 were used as hyper-parameters between the Dense and Reshape layers, between convolution and convolution layers, and between the convolution and conv-transpose layers. The final convolution layer is fitted with a Tanh activation function.

Number Suffix layers	Types of layers	Output Shapes
1st	Input	(1, 100)
2nd	Dense	(1, 32768)
3rd	Reshape	(16, 16, 128)
4th	Convolution	(16, 16, 128)
5th	Conv-Transpose	(32, 32, 128)
6th	Convolution	(32, 32, 128)
7th	Conv-Transpose	(64, 64, 128)
8th	Convolution	(64, 64, 128)
9th	Convolution	(64, 64, 128)
10th	Convolution/Output	(64, 64, 3)

Table 4.3: The description of network layers of Generator network of the proposed DCGAN model

4.4.3 Image Restoration

Image restoration of DCGAN involves training and testing workflow of the DCGAN model.

Training

The code implementing the DCGAN is based on existing code [25]. Each epoch of DCGAN is first set up to train the discriminator, then the generator. The balancing is essential in order not to overtrain one of them, and make both able to catch up with each other. Figures 4.8 and 4.9 show the training workflow of the discriminator and generator of the DCGAN model, respectively.

The discriminator is trained by creating two batches of images and using these as input to the discriminator model. One batch contains reference images, and the other contains generated images. The output of the discriminator model is a prediction on whether the generated image is real or false. First, the discriminator trains on reference images. The correct prediction is that the images are real. Then, the discriminator trains on generated images. The losses from both batches are averaged, and the weights are updated.

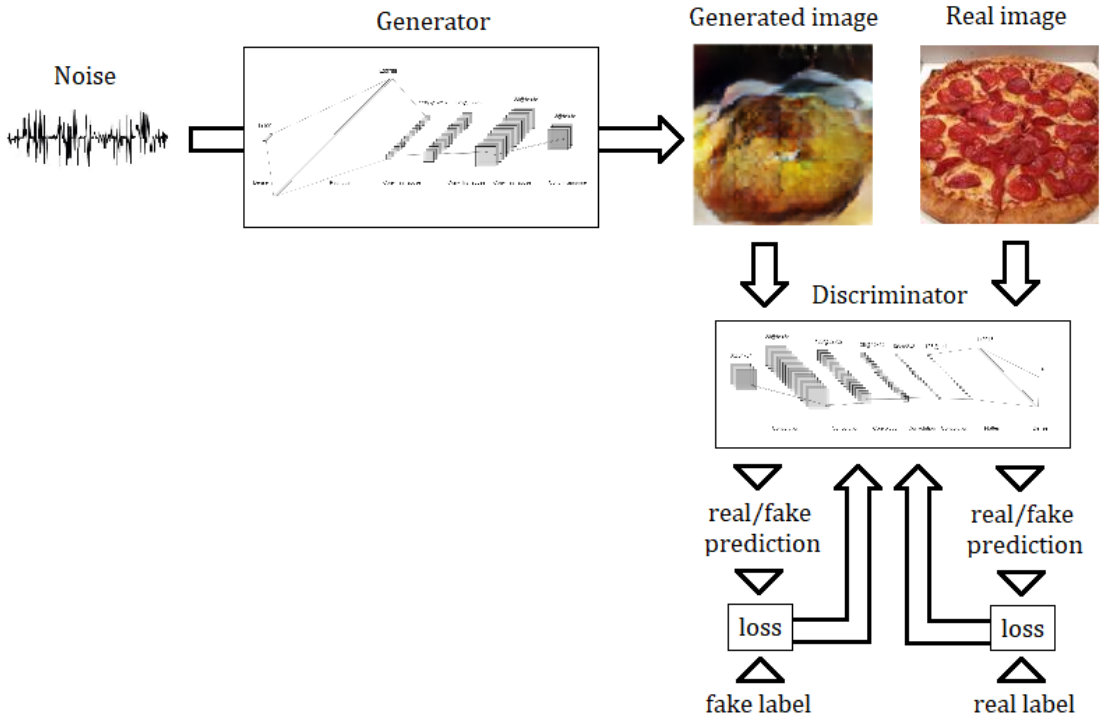


Figure 4.8: The training workflow of Discriminator of the DCGAN

The generator is trained by turning off training for the discriminator, and training the full DCGAN model. The generator input is a batch of random noise samples. The generator output is a batch of generated images. The loss of the DCGAN is low when many generated images are predicted as real and high when many generated images are predicted as false. After the predictions, the weights of the generator are updated to minimize the loss, while the weights of the discriminator remain unchanged.

The generator is initially creating images from random noise, then after a while, the weights in the generator layers turn the noisy input into a value that it assumes the discriminator will predict to be true. An effect of the random input of the generator is that the output image will not resemble any image in the dataset. This will result in images without date stamps when trained with ground truth images, but these images are not restored images, but completely new generated images of food not seen before. Since the goal of the thesis is to remove date stamps from images, this result is not useful.

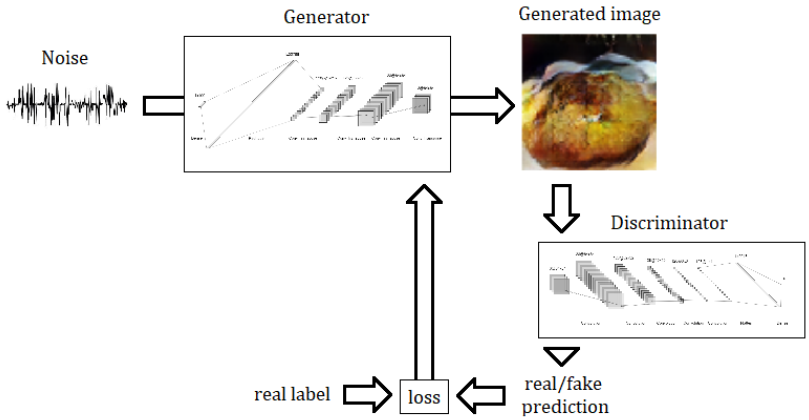


Figure 4.9: The training workflow of Generator of the DCGAN

Testing

After the network has trained the desired number of epochs, the quality of generated images are evaluated. Since the generator and discriminator are trained in parallel, evaluating the images using the network's discriminator can result in the same accuracy with a poorly performing network and with a well-performing network. One possibility is using a classifier trained on all classes of the food-11 dataset.

In Experiment 2, the generated images are compared with the images with date stamps, and in Experiment 3, the generated images are compared with the images without date stamps. The evaluation method used is PSNR. The value can be directly compared with the value of the DAE test.

The compared images are chosen randomly. This gives a worse PSNR value than if the images had been compared directly. The result can still be of interest compared to the worst case scenario, discussed in 4.5.

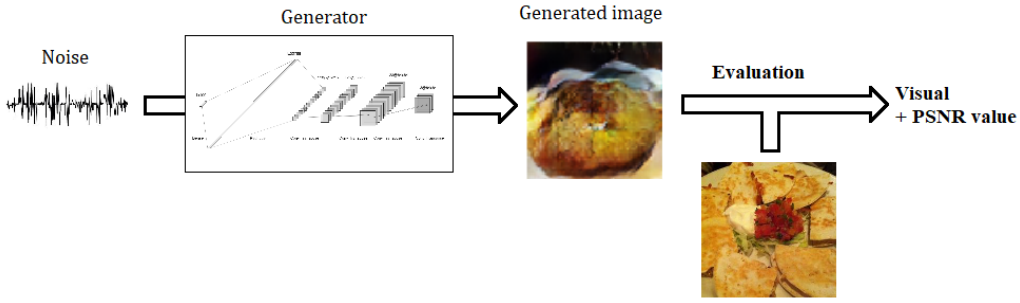


Figure 4.10: The testing workflow of DCGAN

4.4.4 Proposed DAE and Discriminator

The output generated in the experiment above, in Section 4.4.2 does not correspond to any specific ground truth image because it uses random noise as input. A proposed experiment for using DCGAN to restore a specific input image using blind inpainting, is to use the whole date-stamped image as an input to the DCGAN network instead of the random vector that is the input of a generator. This experiment requires a network that is able to do this. A possible solution is to replace the generator with a DAE and train it together with a discriminator, using the discriminator as an additional loss function of the GAN. This will be referred to as a DAE-discriminator network.

4.5 Evaluation for the proposed solutions

The evaluation for the performed experimental results is done in two parts. The first part is a visual evaluation, and the second part is an objective evaluation. The details of the evaluation part can be seen in 5.5. The visual evaluation is performed in Section 5.5.1, where the image quality is assessed by visually comparing the results generated by the proposed models DAE and DCGAN and an opinion based judgment. The objective via quantitative evaluation is performed in Section 5.5.2 by using PSNR to objectively and quantitatively compare the results produced by DAE and DCGAN. PSNR is chosen as it is the most commonly used method to objectively and quantitatively compare the quality of the images. It is preferred to use PSNR and not MSE because PSNR does not result in very small numbers. In both the visual and objective approaches, the quality of the original images is compared to the quality of the predicted images, the reconstruction images in the case of DAE model, and the generated images in the case of DCGAN model.

The whole image is used when comparing images for these experiments, and only a portion of the image is corrupted. The reason for this choice is to include any potential quality loss in locations where there is no text in the corrupted image. This means that in this case, the PSNR value will be higher compared to other experiments where the whole image is corrupted, as long as the rest of the output image stays the same. The worst case scenario is a black image compared with a white image, where MSE=1, PSNR=48.13dB. The average PSNR value between ground truth images and date stamped images is 67.8.

Given an original image f and a predicted image g, both of the size $M * N$, the PSNR between f and g is defined by:

$$PSNR(f,g) = 10\log_{10}(\frac{255^2}{MSE(f,g)}) \tag{4.1}$$

where

$$MSE(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - g_{ij})^2 \tag{4.2}$$

in which i represents the horizontal pixel value and j represents the vertical pixel value The PSNR value approaches infinity as the MSE approaches zero: this means that a higher PSNR value provides a higher image quality. Whereas, a small PSNR value indicates high numerical differences between images [26].

4.6 Tools

4.6.1 Software Tools

Developing the different deep learning models such as DAE and DCGAN were done in python 3 programming language and main libraries such as Tensorflow and Keras were used.

- Tensorflow is an open source machine learning system that runs at large scale and in heterogeneous environments. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multi-core CPUs (Central Processing Units), general-purpose GPUs (Graphical Processing Units), and TPUs (Tensor Processing Units). TensorFlow enables developers to test with novel optimizations and training algorithms. It also supports a variety of applications, with a focus on training and inference on deep neural networks [27].

- Keras is a high-level neural networks API which is developed in python and can run on top of software libraries such as TensorFlow, CNTK or Theano. It can run smoothly on both CPU and GPU. Keras was implemented with a focus on doing fast experiments [23].

4.6.2 Hardware Tools

- cair-gpu01.uia.no: this is one of the gpu servers from the University of Agder that is used in this thesis. Its hardware specifications are as as follows [28]:

  DELL PowerEdge R730

  - 2x NVIDIA Tesla K80
  - 2x Intel Xeon E5-2660 2.0GHz 14C/28T
  - 128GB 2400MHz DDR4 RAM
  - 800GB Internal SSD, 20TB Shared storage.

  this server use the following Software Stack.

  - Kubernetes on NVIDIA GPUs (KONG)
  - Ceph (ceph-ansible)
  - JupyterHub

Chapter 5

Results

This Chapter is divided into five Sections. Each of the first four Sections describe a distinct conducted experiment. The fifth Section is a visual and quantitative comparison of the results where the quality of the images from experiment 2, 3, and 4 are compared with the images from experiment 1.
The experiments in this research are as follows:

1. Experiment 1, in Section 5.1 evaluates setups of the various DAE tests and show the performance for removing the date stamps over loss and PSNR.

2. Experiment 2, in Section 5.2 evaluates the setup of DCGAN with corrupted images as the reference, and show the performance for removing the date stamps over loss and PSNR.

3. Experiment 3, in Section 5.3 evaluates the setup of DCGAN with corrupted images as the reference, and show the performance for removing the date stamps over loss and PSNR.

4. Experiment 4, in Section 5.4 evaluates the setup of DAE-Discriminator, and show the performance for removing the date stamps over the loss.

The purpose of these experiments is to see if the models can remove date stamps from colored, high-resolution images or not, to evaluate the quality of the reconstructed images, and to see how stable the training is from the loss curve. Experiment 1 is designed with different DAE setups and different parameters, which make up the various tests. The purpose is to see for each test how much of the date stamps are left after training and compare the performance between the tests to find the best setup. The purpose of Experiment 2 is to produce a result that shows whether DCGAN can return uncorrupted images with good

quality from corrupted images. The purpose of Experiment 3 is to find out how well the DCGAN model from Experiment 2 can perform with input data without corruption. The images are expected to be of higher quality than the images from Experiment 2 because there are fewer features to learn without corruption. The purpose of Experiment 4 is to combine elements of the DCGAN model and the DAE model and see whether the resulting hybrid model can produce images that are close to the ground truth, instead of generating new images. Furthermore, it is interesting to see how the output of a traditional DCGAN network changes when the generator part is replaced by a DAE network.

5.1 Denoising Autoencoder Experiment

Experiment 1 uses the DAE architecture and contains 6 different tests with different parameters and code dimensions. In Test 1 the model is set to train for 20 epochs first and then set to train for 100 epochs. The resulting images are then demonstrated in Figures 5.1 and 5.2. As for Tests 2-6, the DAE models are set to train for 100 epochs. The images produced from tests 1-6 vary slightly in quality as presented by the PSNR results in table 5.3. For visual brevity, only image samples from Test 1 are presented. All tests use parameters as previously stated in chapter 4.3 unless otherwise specified.

The DAE networks that are used in Test 1, 2, and 4 are overcomplete autoencoders with different convolution filters. BCE loss via optimization function is used for Tests 1, 2, and 4. Whereas in Test 3, undercomplete autoencoder with BCE loss function is used. While overcomplete autoencoder with MSE loss function is utilized in Test 5 and overcomplete autoencoder with mean absolute error (MAE) loss is deployed in Test 6.

Typically, an overcomplete autoencoder means the bottleneck of the network has an equal or larger dimension than the input dimension. Whereas an undercomplete autoencoder means the bottleneck has a smaller dimension than the input dimension. A typical autoencoder is an undercomplete network that replicates the input images. A typical DAE network is overcomplete as it takes corrupted input images, and the corruption ensures that the network does not learn the identity function. Hence it is necessary to formulate various versions of overcomplete DAE networks in order to discover which overcomplete DAE variant can deliver the best results. Therefore, Tests 1, 2, 4, 5, and 6 were designed and tested. Compared to those tests, Test 3 is designed and tested as an undercomplete network in order to discover if an undercomplete DAE model can outperform an overcomplete DAE.

5.1.1 Test 1

Figure 5.1 shows the results produced by the DAE model, where the network is trained for 20 epochs. The results indicate the DAE can get rid of the date stamps, but some images show remains of the date stamp still visible after removal. However, after the network has trained for 100 epochs, the resulting output by the DAE model 5.2 shows no remains of date stamps over images, the regions where date stamps are removed are not noticeably blurry, and the results are very satisfactory.

Figure 5.1: Examples of results generated after the DAE is trained for 20 epochs. Top Row: the corrupted samples used as input samples. Bottom Row: the output samples by the DAE model after the date stamps removal.

Figure 5.2: Results generated after the DAE is trained for 100 epochs. Top Row: the corrupted samples used as input images. Bottom row:the output samples by the DAE model after the date stamps removal.

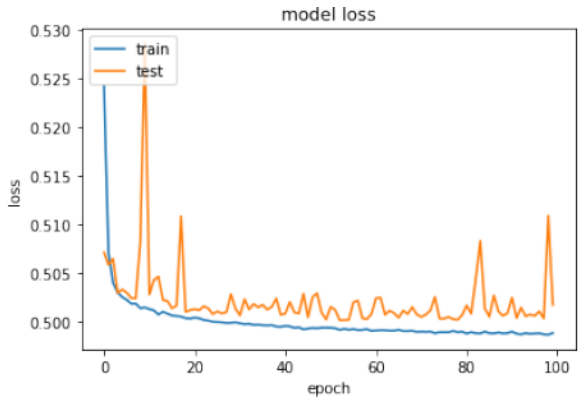


Figure 5.3: Test 1 loss curve.

Observations: Results from Test 1 demonstrate that the output images produced after removing the date stamp by the DAE model do not leave missing regions or holes. The images have left some corrupted regions when the network is trained for 20 epochs, as shown in Figure 5.1. However, when the network is trained for 100 epochs, the corrupted regions disappear. The PSNR value for Test 1 is 77.7, which is computed after the network is trained for 100 epochs. Figure 5.3 shows how much the test loss curve of the network is converging to the training loss curve. The nature of the loss test curve indicates that the network is stable between 20 to 80 epochs while it is unstable on some parts between 80 to 100 epochs. Overall, DAE model can perform well.

5.1.2 Test 2

The filters of the convolution layer 2 and 8 are reduced from 96 to 12. This modification of the DAE network is done to find out the best version of overcomplete DAE that can deliver the best performance.

Observations: Figure 5.4 shows that the test loss is closer to the training loss, with less variation. PSNR value for Test 2 is 78.9, which is computed after the network is trained for 100 epochs. Compared to Test 1, the test loss of Test 2 is smoother, and also the PSNR of Test 2 is slightly higher. This confirms that this DAE variant works better than the DAE variant in Test 1.

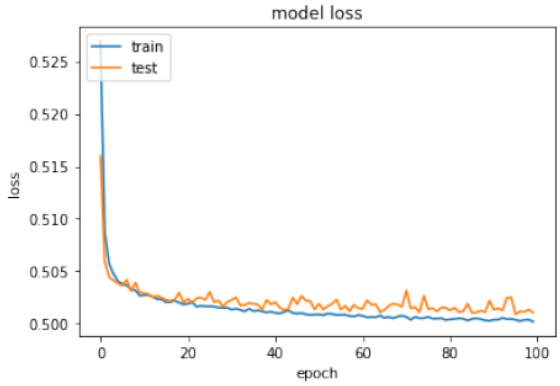


Figure 5.4: Test 2 loss curve.

5.1.3 Test 3

The same changes from test 2 are carried over to test 3. Furthermore, four additional layers are added to the network. The dimension of the bottleneck is 32x32x96, compressing the input with a factor of 2, making the encoder undercomplete. This modification of the DAE network is done to discover if an undercomplete DAE can perform better than an overcomplete DAE.

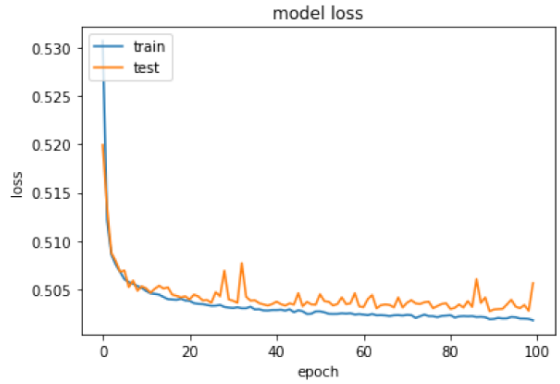


Figure 5.5: Test 3 loss curve.

Observations: Figure 5.5 shows how much test loss is converging against the training loss. PSNR value for Test 3 is 75.1, which is computed after the network is trained for 100 epochs. Unlike loss curve Test 1, the test loss curve of Test

3 is smoother. Compared to tests 1 and 2, the PSNR value of test 3 is lower. Interestingly, the test loss curve of Test 3 is similar to that of Test 2. Shortly, it can be concluded by comparing other tests in Experiment 1 that an overcomplete DAE performs better than that of an undercomplete DAE

5.1.4 Test 4

The filters of the convolution layer 4 and 6 are increased from 48 to 196. This results in a bottleneck that is 4 times larger. This modification of the DAE network is done to find out the best version of an overcomplete DAE that can deliver the best performance.

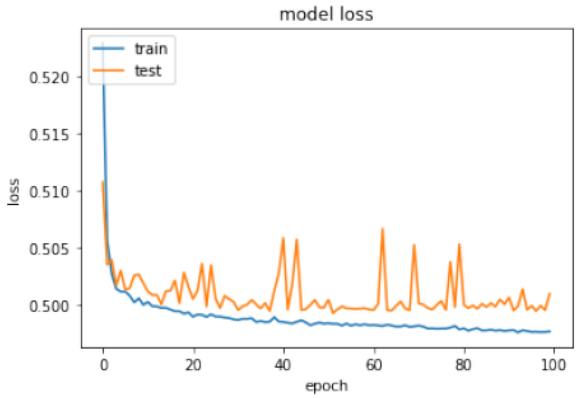


Figure 5.6: Test 4 loss curve.

Observations: Figure 5.6 shows how much the test loss curve is converging against the training loss. PSNR value for Test 4 is 78.8, which is computed after the network is trained for 100 epochs. PSNR for Test 4 is higher than that of tests 1 and 3. The improvement of the results compared with the results in test 1 is minimal. Potential additional tests with networks built deeper than the network in this test will likely yield results that are much higher.

5.1.5 Test 5

MSE loss is used instead of BCE or MAE loss to determine if the DAE model performs better with a different loss function. Apart from this modification, the same network structure as Test 1 is used.

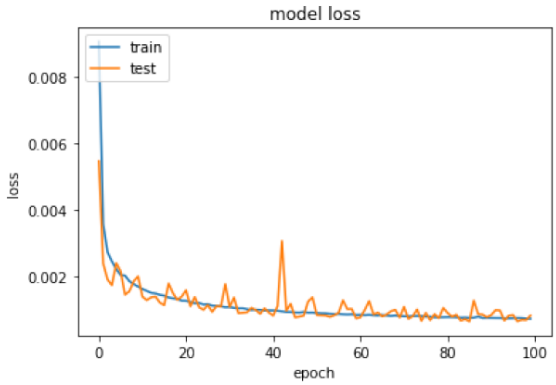


Figure 5.7: Test 5 loss curve.

Observations: PSNR value for Test 5 is 79.6, which is calculated after the network is trained for 100 epochs. Figure 5.7 shows the performance of test loss against the training loss in Test 5. It can be seen from this figure that somewhere between 0 to 40 epochs, the test loss converges better than training loss.

5.1.6 Test 6

MAE loss is used instead of BCE or MSE loss to decide if DAE performs better. Apart from this modification, the same network structure as Test 1 is used.

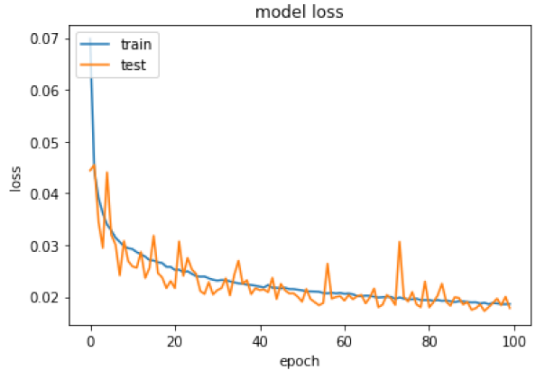


Figure 5.8: Test 6 loss curve.

Observations: PSNR value for Test 5 is 79.7, which is calculated after the network is trained for 100 epochs. Figure 5.8 shows the performance of test loss

against the training loss in Test 6. It can be seen from this figure that somewhere between 0 to 50 epochs, the test loss converges better than training loss.

5.2 Deep Convolutional Generative Adversarial Net

This experiment is about the test with DCGAN with date stamps where the binary cross entropy loss function is used. The results in 5.9, 5.10 and 5.11 are generated after the network has trained for 100 epochs. This experiment yields interesting results. This experiment can be split into three different types of results. Figure 5.9 shows the generated images where new date stamps such as 2 or 3 over the images appeared. Whereas, it can be seen from the generated Figure 5.10 that date stamps disappeared. DCGAN also generated images 5.11 where date stamps are preserved.

Figure 5.9: Generated images where additional date stamps appeared

Figure 5.10: Generated images where date stamps disappeared

Figure 5.11: Generated images where date stamps are preserved

Observations Unlike Experiment 1, the quality of generated results by DCGAN in the Experiment 5.2 are challenging to identify, but it seems DCGAN can get rid of date stamps on some occasions. Therefore, it is hard to decide whether the output images produced after removing the date stamp by DCGAN model does not leave mission regions or holes. That is DCGAN sometimes generates with new stamps appeared on the output. In some occasion, it produces images where the date stamps disappeared. Other times, it generates images where the date stamps are preserved. PSNR value for Experiment 2 is 51.2 which is computed after the network is trained for 100 epochs.

Figure 5.12 shows a trend of increasing the cost for the generator, meaning it is struggling to create convincing samples as the number of epochs increases. This agrees with Figure 5.13, which shows that the discriminator has better than random guessing from the start. Also, it continues to improve throughout the training session. Near the end, it averages on 0.3, meaning the generator is only able to create convincing images 30% of the time. This results in what is known as the diminished gradient or vanishing gradient problem. This means that the generator gradient gradually diminishes and learns slower, and in some cases vanishes to zero. A visual analysis Figure 5.12 shows that the cost seems to jump less from epoch to epoch in the range epochs 60 to 100 than in the range 0 to 60. However, this cost is still higher than ever before, indicating that the model is unable to change.

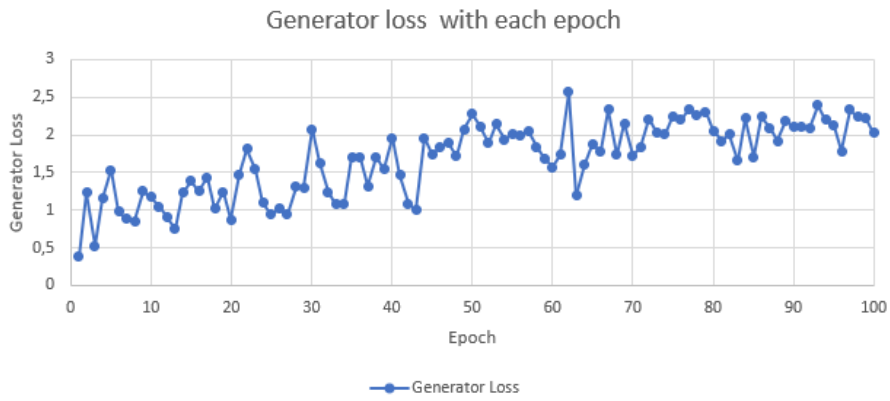


Figure 5.12: Loss for each epoch of Generator network during training of the DCGAN model in experiment 2.

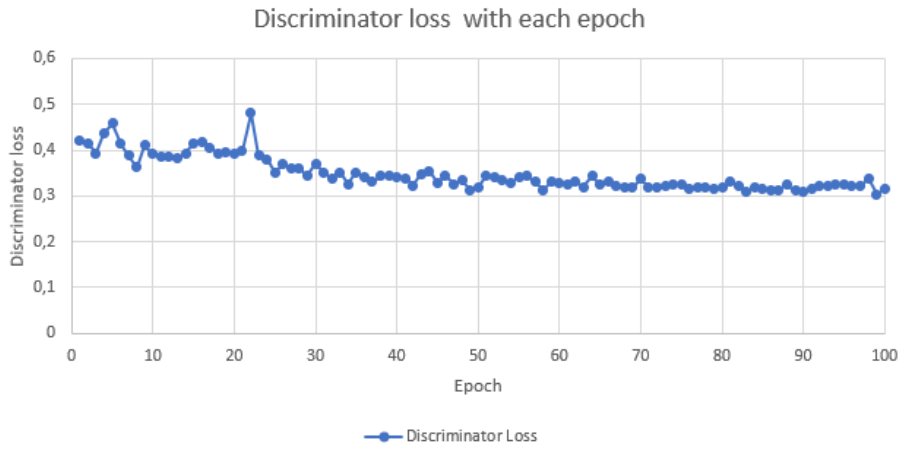


Figure 5.13: Loss for each epoch of Discriminator network during training of the DCGAN model in experiment 2

5.3 Deep Convolutional General Adversarial Net without date stamp Experiment

This experiment is about the test with DCGAN without date stamps where the binary cross entropy loss function is used. The objective of this experiment is to discover how well the proposed DCGAN can handle the data without corruption

and to compare it to the results where corruption was added. The results in the Figure 5.14 is generated after 100 epochs. Compared to results from Experiment 2, the results from experiment 3 are more pleasing to see.

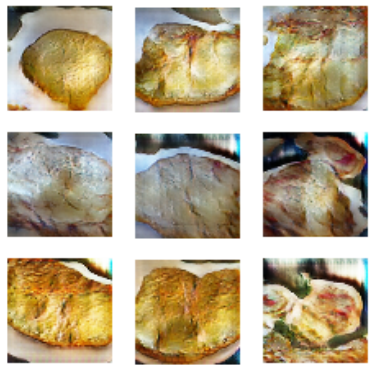


Figure 5.14: Generated images after 100 epochs

Observations: PSNR value for Experiment 3 is 51.2, which is computed after the network is trained for 100 epochs. Results from this experiment demonstrate that DCGAN can generate visually pleasing results. However, the generated results are still hard to identify. Figure 5.16 shows that the discriminator loss lies on average in the range of 0.4 to 0.5. This means that the discriminator is only slightly better than random guessing at separating the generated images from the real images. Additionally, during the last 10 epochs, the cost shows little variation and follows an almost straight horizontal line. Figure 5.15 shows the loss of the generator, which varies significantly during the training period. The change in cost values change less drastically as the training progresses, and both models become more confident, up until epoch 90, where the data points are tightly grouped. Since the discriminator from epoch 90 to the end is close to random guessing, the cost values for the generator are close to optimal and have learned an image representation that it only makes small changes to.

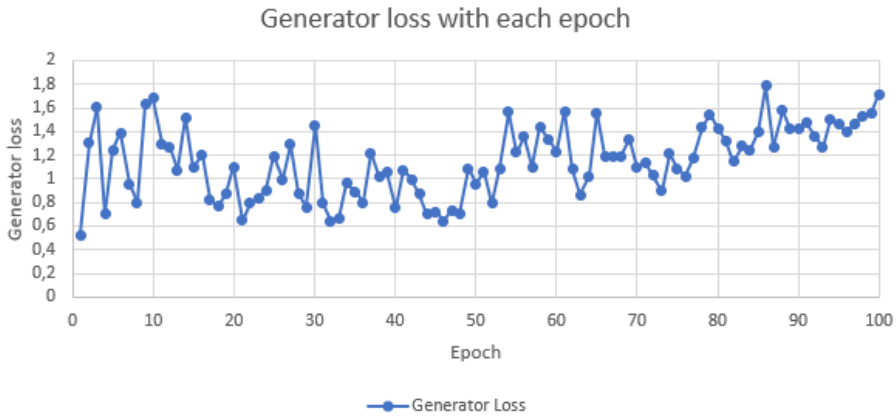


Figure 5.15: Loss for each epoch of Generator network during training of the DCGAN model in experiment 3

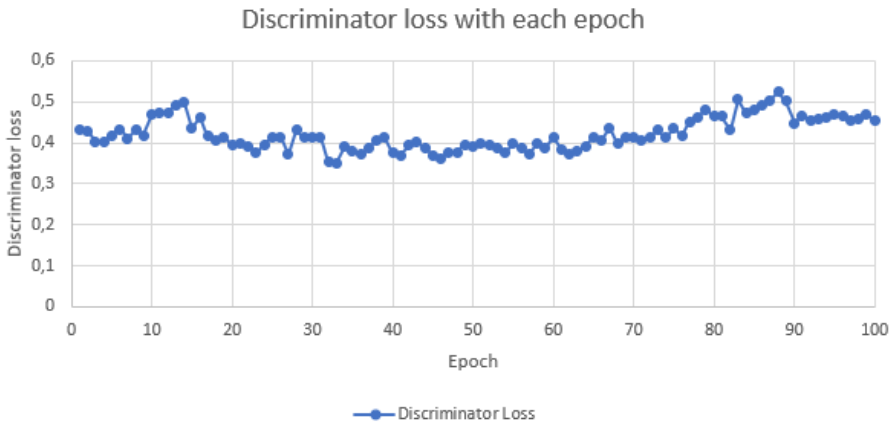


Figure 5.16: Loss for each epoch of Discriminator network during training of the DCGAN model in experiment 3

5.4 Denoising Autoencoder and Discriminator Experiment

This experiment is about the test with DAE and Discriminator model with date stamps where the binary cross entropy loss function is applied. Here DAE is used as a generator. The DAE-discriminator combination results in images that are

different from the inputs even outside the area of the date stamp. The colors have changed. The generated results are unsatisfactory compared to the results from experiment 1, 2, and 3. The entire image is changed, while the goal is only to change the region with the date stamp. This can be seen in Figure 5.17 where the network is trained for 1 epoch and in Figure 5.18 where the network is trained for 100 epochs.

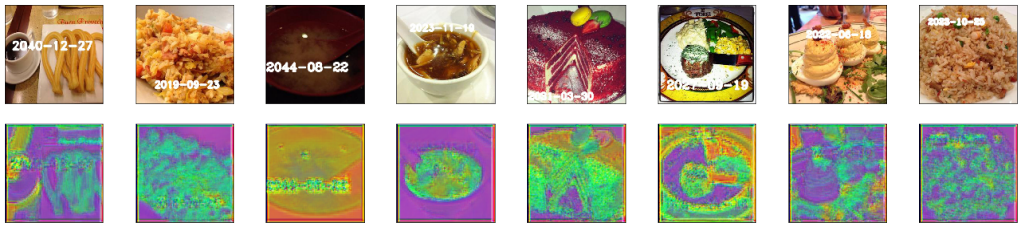


Figure 5.17: Generated image by DAE-Discriminator network after trained 1 epoch. Here, Top row: corrupted images with date stamps. Bottom row: decoded images.

Figure 5.18: Generated image by DAE-Discriminator network after trained 100 epoch. Here, Top row: corrupted images with date stamps. Bottom row: decoded images.

Observations: The DAE-Discriminator model generates worse images than the DCGAN model in Experiment 2. That is the whole output image changes entirely in color. A PSNR for Experiment 4 is unnecessary, as a visual comparison between the results in Experiment 1 and 4 show that the DAE model is superior. Figure 5.20 shows that the cost for the discriminator varies greatly from epoch to epoch throughout most of the experiment. It does, however, stabilize a bit near the end, jumping from loss values in-between 0.2 to 0.325 from epoch 90 to 100. The general trend shows that the discriminator performance is good, making correct predictions up to 70% - 80% of the time (near the end). Figure 5.19 shows

some interesting behavior on the DAE. At first, the model has a low cost and little changes in cost from epoch to epoch in the range epochs 0 to 20. From this point forward, as the DAE tries to optimize, or the discriminator learns to tell the generated images apart from the actual images, the cost values jump widely throughout the training period, until stabilizing again during the last 10 epochs. The cost remains low near the end, which seems counter-intuitive when compared against the discriminator loss, which is also low.

Looking at the decoded images reveals that they retain the shape and structure of the food items, but change the colors of the images in drastic ways. For both training sessions, this color shift is identical across the decoded images for each epoch, which implies partial mode collapse. Meaning that while the images retain their structure and shape, the colors become unvaried and similar in all decoded examples.

The high performance of the discriminator is likely due to it detecting typical patterns in the decoded images, and learning to predict them based on this, correctly. The variation in cost seen in both Figures is possibly due to the difference in how much the color shifts each epoch. Some color compositions highlight the structure of the image more than others, causing the discriminator to fluctuate between false predictions from the similar structure of the images and correct predictions based on color.

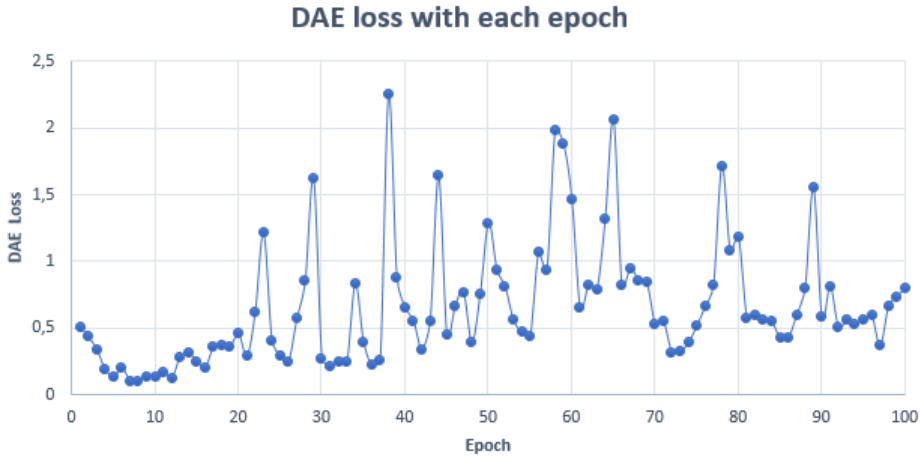


Figure 5.19: Loss for each epoch of DAE network during training of the DCGAN model in experiment 4

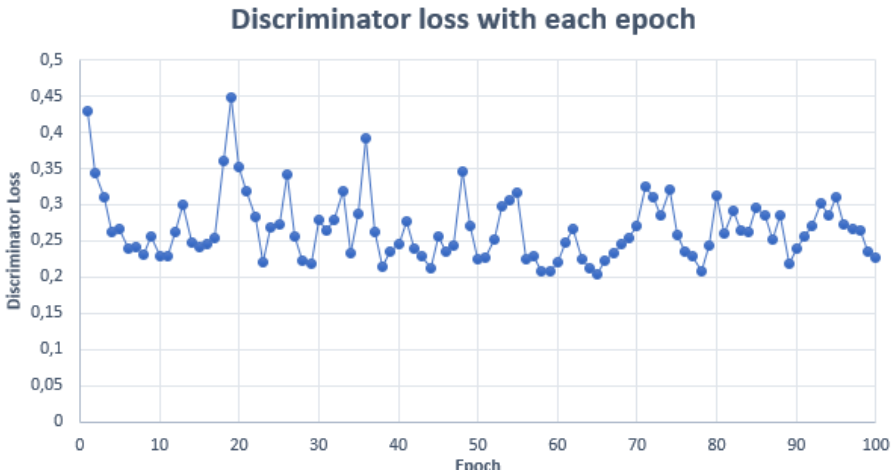


Figure 5.20: Loss for each epoch of Discriminator network during training of the DCGAN model in experiment 4

5.5 Comparisons

This Section involves two parts, namely visual evaluation and the quantitative evaluation of the results.

5.5.1 Visual evaluation of the results

The objective of the visual evaluation is to visually compare the results yielded by the DAE and DCGAN models based on opinion-based judgment. Figure 5.21 shows the results, where the top row shows the results of the DAE model from Experiment 1, and the bottom row shows the result of the DCGAN model from Experiment 2. By looking at this Figure, it can be clearly stated that the overall quality of the images from the top row is superior to the images from the bottom row. Top row images are produced after the DAE network is trained for just 20 epochs while the bottom row images are generated after the DCGAN network is trained for 100 epochs. It is hard to say what sorts of food are shown in the bottom row images while it is quite easy to identify that the top row images represent rice, soup, and spaghetti.

Figure 5.21: Top row: reconstructed images by DAE model after date stamp removal which has taken from 5.1 and Bottom row: generated images by DCGAN model where date stamps disappeared and the image is taken from 5.10

Compared to the DAE model, the DCGAN model produces poor results in general. Moreover, the DAE can completely get rid of the date stamps from the colored, high-resolution images while the DCGAN struggles to remove the date stamps and introduce much noise in the process. A visual comparison shows that the DAE performs better than the DCGAN for automated date stamp removal from high-resolution images.

5.5.2 Quantitative evaluation of the results

This Section addresses the qualitative evaluation of the results yielded by the DAE and DCGAN models. PSNR metric is used to assess the results achieved by the DAE and DCGAN models objectively. The reason why PSNR metrics are chosen for this task is explained in Section 4.5. The table 5.1 shows the PSNR values for the DAE model that is used for Test 1 in Experiment 1 and the DCGAN model that is used for Experiment 2. PSNR values are calculated for both models after they are trained for 100 epochs. It can be seen from this table that the PSNR value of DAE is higher than that of DCGAN. The results show that the DAE model performs better than the DCGAN in the task of removing date stamps from high-resolution images.

Dataset	DAE/PSNR	DCGAN/PSNR
Food11	77.7	51.2

Table 5.1: Quantitative evaluations for DAE Vs DCGAN. Here PSNR value of DAE is for test 1 from the experiment 1 and PSNR value for DCGAN represents for experiment 2.

The table 5.2 shows the PSNR value of the DCGAN model that is tested in Experiment 3, where the PSNR is computed after the network has trained for 100 epochs. The purpose of computing PSNR value for Experiment 3 is to find out the performance of DCGAN when it is tested with uncorrupted images. It can be seen from the tables 5.1 and 5.2 that there is a small difference between when DCGAN is tested with corrupted images and when it is tested with uncorrupted images.

Dataset	DCGAN/PSNR
Food11	52.1

Table 5.2: Quantitative evaluation for DCGAN without date stamps. PSNR value for DCGAN represents for experiment 3

Tests	Network type	Loss	PSNR value
Test 1	overcomplete	BCE	77.7
Test 2	overcomplete	BCE	78.9
Test 3	undercomplete	BCE	75.1
Test 4	overcomplete	BCE	78.8
Test 5	overcomplete	MSE	79.6
Test 6	overcomplete	MAE	79.7

Table 5.3: Summary of Quantitative evaluations for different DAE tests in Experiment 1. Here BCE means binary cross entropy, MSE means mean squared error and MAE means mean absolute error. DAE network is trained for 100 epochs in all these tests.

The table 5.3 shows the summary of quantitative evaluations for various DAE tests in Experiment 1. Here DAE is trained for 100 epochs in all tests, and PSNR value for each test is computed after the network is trained for 100 epochs. By comparing all PSNR values of tests in this table, it can be concluded that an overcomplete DAE performs better than an undercomplete DAE. Hence there is no motivation in designing an undercomplete DAE for the task of removing date

stamps from colored, high-resolution images. It can also be seen that the PSNR value of Test 6 is the highest. Therefore, overcomplete DAE with MAE loss via optimization performs best among all the conducted tests.

Overall, both visual and quantitative evaluation of the results indicates that the DAE model is superior to the DCGAN model in eradicating the date stamps from the colored, high-resolution images.

Chapter 6

Conclusion and Future Work

This chapter contains the Conclusion and Future Work Sections. The Conclusion Section presents a review of the results in the experiments. The Future Work Section presents various suggestions for improvements and suggestions for experiments for the extension of this thesis.

6.1 Conclusion

The two neural network types used in this thesis are networks that are widely used for image manipulation tasks, namely Denoising Autoencoder (DAE) and Deep Convolutional Generative Adversarial Network (DCGAN). The experiments in this thesis use and compare these networks for image inpainting tasks, specifically the removal of date stamps from high resolution colored images.

Both visual and quantitative results clearly show that DAE performs better than DCGAN for the automated removal of date stamps, and is able to remove date stamps with very little blur and artifacts in the corrupted regions. The solution with the best performance achieves a PSNR value of 79.7. DAE can handle $256 * 256$ colored images. With just 20 epochs, it yields outstanding results. The reconstruction images without date stamps from the DAE output are converging to the reference images. With the increasing epochs, the reconstruction images by DAE become even better. The resulting images are less blurry and usable in practice. The overall impression of the DAE model is that it yields entirely satisfactory results. Considering the minimal improvement of the results in tests with increased layers, potential networks that are built even deeper is likely to yield the same results.

DCGAN can handle $64 * 64$ colored images. However, the generated results by DCGAN are of poor quality and does not meet the hypothesis. However, these results are interesting since the generator network of DCGAN sometimes generates images with one date stamp, other times with two or three stamps or other times without any date stamps. The particular case where no date stamps are generated can be useful. For example, a generated image with the same features as the ground truth image and without a date stamp can be used to generate uncorrupted datasets. The DAE-DCGAN model is unfit for automated removal of date stamps from the high-resolution images.

6.2 Future Work

Further improvement of the DAE model can be done by searching for and implementing other technologies that reduce the blurriness of the restored regions. As for DCGAN, the challenge is to make the generated image look as similar to the non-generated images as possible. The model can be trained on images with larger resolution.

In the future, this thesis can be extended into blind video inpainting or solving more complicated image inpainting tasks such as a combination of super-resolution, and removal of sketches, text or watermarks with different colors that are blending in with the image to a larger degree than the corrupted images in this thesis. Hence, one can find out how far the algorithms can go to perform said tasks without damaging the original quality of the images.

Bibliography

[1] S Shivaranjani and R Priyadharsini. "A survey on inpainting techniques". In: *Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on*. IEEE. 2016, pp. 2934–2937.

[2] Junyuan Xie, Linli Xu, and Enhong Chen. "Image denoising and inpainting with deep neural networks". In: *Advances in neural information processing systems*. 2012, pp. 341–349.

[3] Jeena Joshua and Gopu Darsan. "Digital inpainting techniques: A survey". In: *Intern. J. of Latest Research in Engineering and Technology* 2 (2016), pp. 34–36.

[4] Jiahui Yu et al. "Generative image inpainting with contextual attention". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5505–5514.

[5] Andrew Ng et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[7] Francois Chollet. *Building Autoencoders in Keras*. 2016. URL: `https://blog.keras.io/building-autoencoders-in-keras.html` (visited on 03/30/2019).

[8] Jeremy Jordan. *Introduction to autoencoders*. 2018. URL: `https://www.jeremyjordan.me/autoencoders/` (visited on 03/03/2019).

[9] Jeremy Jordan. *Variational autoencoders*. 2018. URL: `https://www.jeremyjordan.me/variational-autoencoders/` (visited on 03/03/2019).

[10] Ian Goodfellow. "NIPS 2016 tutorial: Generative adversarial networks". In: *arXiv preprint arXiv:1701.00160* (2016).

[11] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[12] Yang Liu, Jinshan Pan, and Zhixun Su. "Deep Blind Image Inpainting". In: *arXiv preprint arXiv:1712.09078* (2017).

[13] Qiwen Fu, You Guan, and Yuxin Yang. *Image Inpainting and Object Removal with Deep Convolutional GAN*. Tech. rep. URL: `http://stanford.edu/class/ee367/Winter2018/fu_guan_yang_ee367_win18_report.pdf`.

[14] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep Image Prior". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[15] Deepak Pathak et al. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[16] Yusra AY Al-Najjar, D Chen Soong, et al. "Comparison of image quality assessment: PSNR, HVS, SSIM, UIQI". In: *Int. J. Sci. Eng. Res* 3.8 (2012), pp. 1–5.

[17] *Food-11: Food Image Dataset*. [Online; accessed 30. January 2019]. URL: `https://mmspg.epfl.ch/food-image-datasets`.

[18] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[19] *Publication-ready NN-architecture schematics*. [Online; accessed 24. April 2019]. Apr. 2019. URL: `http://alexlenail.me/NN-SVG/LeNet.html`.

[20] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[21] *The background resource for Denoising Autoencoder*. [Online; accessed 02. February 2019]. Feb. 2019. URL: `https://github.com/shibuiwilliam/Keras_Autoencoder/blob/master/Cifar_Conv_AutoEncoder.ipynb`.

[22] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[23] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[24] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[25] *The background resource for Deep Convolution General Adversarial Net.* [Online; accessed 11. March 2019]. Feb. 2019. URL: https://github.com/utkd/gans/blob/master/cifar10dcgan.ipynb.

[26] Alain Hore and Djemel Ziou. "Image quality metrics: PSNR vs. SSIM". In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 2366–2369.

[27] Martin Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[28] *cair-gpu01.uia.no.* [Online; accessed 09 February 2019]. Jan. 2019. URL: https://tools.uia.no/bitbucket/projects/CAIR/repos/cair-gpu/browse.

Appendices

Code for this thesis is available at https://github.com/Nicolas-31/IKT590

UiA University of Agder
Master's thesis
Faculty of Engineering and Science
Department of ICT