ISTANBUL TECHNICAL UNIVERSITY
ELECTRICAL-ELECTRONICS FACULTY

IMAGE INPAINTING WITH DEEP LEARNING

SENIOR DESIGN PROJECT

Enes DEMİRAĞ
Halil İbrahim BENGÜ

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

January 2021

ISTANBUL TECHNICAL UNIVERSITY
ELECTRICAL-ELECTRONICS FACULTY

IMAGE INPAINTING WITH DEEP LEARNING

SENIOR DESIGN PROJECT

Enes DEMİRAĞ
040160027

Halil İbrahim BENGÜ
040160053

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

Project Advisor: Prof. Dr. Ender Mete EKŞİOĞLU

January 2021

İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

DERİN ÖĞRENME İLE RESİM BOYAMA

LİSANS BİTİRME TASARIM PROJESİ

Enes DEMİRAĞ
040160027

Halil İbrahim BENGÜ
040160053

Proje Danışmanı: Prof. Dr. Ender Mete EKŞİOĞLU

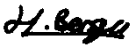
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

OCAK, 2021

We are submitting the Senior Design Project Report entitled as Image Inpainting with Deep Learning. The Senior Design Project Report has been prepared as to fulfill the relevant regulations of the Electronics and Communication Engineering Department of Istanbul Technical University. We hereby confirm that we have realized all stages of the Senior Design Project work by ourselves and we have abided by the ethical rules with respect to academic and professional integrity .

Enes DEMİRAĞ
040160027

Halil İbrahim BENGÜ
040160053



FOREWORD

We would like to thank our adviser, Prof. Dr. Ender Mete Ekşioğlu, for providing guidance and feedback throughout this project.

January 2021

Halil İbrahim BENGÜ
Enes DEMİRAĞ

TABLE OF CONTENTS

Page

ABBREVIATIONS

ANN	**:** Artificial Neural Network
CNN	**:** Convolutional Neural Network
DCGAN	**:** Deep Convolutional GAN
FID	**:** Fréchet Inception Distance
FMM	**:** Fast Marching Method
FPN	**:** Feature Pyramid Network
GAN	**:** Generative Adversarial Network
GMCNN	**:** Generative Multi-column Convolutional Neural Network
GPU	**:** Graphics Processing Unit
ID-MRF	**:** Implicit Diversified Markov Random Fields
MLP	**:** Milti Layer Perceptron
MS	**:** Mumford-Shah's
MSE	**:** Mean Squared Error
openCV	**:** Open Source Computer Vision Library
PCA	**:** Principal Component Analysis
PSNR	**:** Peak Signal-to-Noise Ratio
ReLU	**:** Rectified Linear Unit
ResNet	**:** Residual Network
RGB	**:** Red, Green, Blue
SSIM	**:** Structural Similarity Index
TV	**:** Total Variation
VGG	**:** Visual Geometry Group
WGAN	**:** Wasserstein GAN

SYMBOLS

dB	**:** Decibel
$\odot$	**:** Hadamard Product Operator
$\mathscr{L}$	**:** Loss
$\mathbb{E}$	**:** Expected Value

LIST OF TABLES

Page

LIST OF FIGURES

Page

Image Inpainting with Deep Learning

SUMMARY

There may be parts in the images with different features that are not desired. These undesired parts might be caused by aging of a photograph or these parts could include an object which is wanted to be removed from the image. Removing these parts and inpainting them with visually realistic features is both a research area and a commercial problem. There are many specialized tools for this purpose and use of these tools may also require professional knowledge.

The aim of the Image Inpainting with Deep Learning project is to examine the different structures in the pictures that fill these regions that are extracted from the pictures. For this purpose, many different work types were divided into groups and sample studies from these groups were compared. Inpainting methods for review purposes, are divided into two as traditional methods and deep methods. Although many methods were mentioned during our study only a few are chosen. Navier-strokes and fast-marching methods were chosen for comparison as example of traditional methods. EdgeConnect, Generative Contextual, Generative Multi-column Convolutional Neural Network and Deep Image Prior methods, which were found to give successful results among deep methods, were selected to be examined in detaial and implemented. In addition to all these, two novel methods are implemented which are inspired from all methods we examined throughout this project. Results of these two methods are also explained and examined in this project.

As examples of deep methods, our two methods, EdgeConnect and Generative Contextual are two GAN based studies which use deep artificial neural networks that work adversarial to each other. There are two stages in the EdgeConnect method and these two stages use the mentioned GAN structure. In the first stage, corrupted image is taken as input to the model and edge information is predicted, and in the second stage, it is desired to produce an image whose missing region is filled by using the both edge information and masked image. Also, in Generative Contextual study there are two stages that use GAN aswell. In the first stage, maske area is inpainted roughlt, and in the second stage, improvement operations are carried out on the created image by using a layer that aims to obtain contextual information.

Another deep image inpainting study chosen is the Generative Multi-column CNN method, which uses convolutional layers in parallel. This method passes the input image through parallel CNN structures using filters of different kernel sizes. Then, these structures, which provide different levels of information on the image, are brought together. And as a result, the inpainting process is completed.

Finally, another method that we have examined in detail is the Deep Image Prior method. This is a method that performs the inpainting process without the need to train a model using any dataset. This method, which can also be used in applications

such as super-resolution and image denoising, obtains iterative results by using various CNN structures.

Pytorch and Tensorflow deep learning libraries were used during our study. Our studies were run on the graphics card, especially since serious processing power was required for training and testing of deep learning methods. As a result of our project, different studies that we tested were compared and presented with PSNR and SSIM values and example images.

Derin Öğrenme ile Resim Boyama

ÖZET

Görsellerde, bulunulması istenmeyen değişik özelliklere sahip bölümler bulunabilmektedir. Bu bölümler eğer söz konusu görsel, fiziksel olarak bulunan bir görsel ise görselin eskimesi sebebiyle oluşmuş olabileceği gibi, dijital bir resimde istenmeyen bir obje veya bölge de olabilir. Görsellerden bu hataların düzeltilmesi veya bu bölümlerin görsellerden çıkartılıp, yerlerinin gerçekçi bir şekilde doldurulmak istenmesi günümüzde araştırma alanı olduğu gibi aynı zamanda ticari bir problemdir. Bu amaca yönelik çok sayıda özelleşmiş araçlar bulunmaktadır ve bu araçların kullanılması profesyonel kullanım da gerektirebilmektedir.

Derin Öğrenme ile Resim Boyama projesinde amaç, resimlerde bulunan ve görsellerden çıkarılan bu bölgeleri dolduran farklı yapıların incelenmesidir. Bu amaç doğrultusunda, bir çok farklı proje, türlerine göre gruplara ayrılmış ve bu gruplardan örnek çalışmalar karşılaştırılmıştır. İncelenmek üzere inpainting yöntemleri geleneksel yöntemler ve derin yöntemler olmak üzere ikiye ayrılmıştır. Çalışmamız sırasında fazla sayıda yöntemden bahsedilmiş olmakla birlikte, karşılaştırma için geleneksel yöntemlerden navier-strokes ve fast-marching yöntemleri seçilmiştir. Derin yöntemler arasından da başarılı sonuçlar verdiği tespit edilen, EdgeConnect, Generative Contextual, Generative Multi-column Convolutional Neural Network ve Deep Image Prior yöntemleri detaylı olarak incelenmek ve gerçeklemek üzere seçilmiştir. Bütün bunlara ek olarak, çalışmamızın sonunda incelemiş olduğumuz farklı tür çalışmalardan ilham alarak gerçeklemeye karar verdiğimiz iki yöntem de incelenmiş ve görsel sonuçlar elde edilmiştir.

Derin yöntemlere örnek olarak seçtiğimiz iki yöntemimiz EdgeConnect ve Generative Contextual, GAN adı verilen ve birbirlerine zıt çalışan iki derin yapay sinir ağı temelli çalışmalardır. EdgeConnect çalışmasında iki aşama bulunmaktadır ve bu iki aşama da bahsedilen GAN yapısını kullanmaktadır. Birinci aşamada girdi olarak alınan görselin eksik bölgelerindeki sınır çizgileri tahmin edilmeye çalışılmakta, ikinci aşamada oluşturulan sınır çizgileri ve maskelenmiş görsel kullanılarak eksik bölgesi doldurulmuş görsel üretilmek istenmektedir. Generative Contextual çalışmasında da GAN kullanan 2 aşama bulunmaktadır. İlk aşamada kabaca maskelenmiş bölge doldurulmakta, ikinci aşamada ise oluşturulan görsel üzerinde, anlam bilgisi elde etmeyi amaçlayan bir katman da kullanılarak iyileştirme çalışmaları yapılmaktadır.

Seçilmiş olan diğer bir derin resim boyama çalışması ise konvolusyon katmanlarını paralel olarak kullanan Generative Multi-column CNN yöntemidir. Bu yöntem girdi olarak aldığı görseli farklı boyutlarda filtreler kullanarak paralel CNN yapılarından geçirir. Ardından görsel üzerindeki farklı seviyede bilgi elde etmeyi sağlayan bu yapılar bir araya getirilir ve sonuç olarak resim boyama işlemi tamamlanmış olur.

Son olarak ayrıntılı incelediğimiz bir diğer yöntem ise Deep Image Prior yöntemidir. Bu yöntem herhangi bir dataset kullanılarak bir model eğitilmesi ihtiyacı olmadan resim boyama işlemini gerçekleştiren bir yöntemdir. Resim boyama dışında süper-çözünürlük ve gürültü giderme gibi uygulamalarda da kullanılabilen bu yöntem çeşitli CNN yapılarını kullanarak iteratif olarak sonuç elde eder.

Çalışmamız sırasında Pytorch ve Tensorflow derin öğrenme kütüphaneleri kullanılmıştır. Özellikle derin yöntemlerin eğitilmesi ve test aşaması için ciddi işlem gücü gerekmesi sebebiyle çalışmalarımız ekran kartı üzerinde çalıştırılmıştır. Çalışmalarımızın sonucunda, testlerini gerçeklediğimiz farklı çalışmalar örnek görsellerle birlikte, PSNR ve SSIM değerleri üzerinden karşılaştırılmış ve sunulmuştur.

1. INTRODUCTION

Inpainting is the process of reconstructing missing and corrupted regions in an image. Early works in this field started with manually repairing deformed photographs by a talented artists. With the transition from analog photography to digital images, algorithms that can perform this process in a much shorter time started to appear. With the development of deep learning technologies and the increase in image data resources, the use of artificial neural network models has become more popular to solve this problem. These methods perform the image inpainting process completely automatically without the need for people.

Image inpainting can be used for purposes such as removing an unwanted object from the image or restoring a noisy image by filling the missing data in images. Traditional image restoration methods are basically designed to fill missing pixel values in a way that similar to the neighboring pixel values. These methods usually give poor results in cases with large missing areas.

Modern methods are usually trained with millions of images from thousands of different labels using supervised machine learning. For example, generative adversarial network, which has the ability to generate new data, is frequently used in such studies. A neural network alone cannot understand the places that need to be filled in an given input image. Therefore, input image is given along with a mask indicating that the related corrupted parts in the image are missing and need to be filled. Then, output image is generated by passing through various network layers. The missing parts of the original image are filled by taking the masked parts from this generated output image.

2. TRADITIONAL INPAINTING METHODS

There are many different methods used for image inpainting. Each of these methods has mathematical foundations in itself. These methods are categorized according to the algorithms they use to get the desired output [31]. The first of these methods are the methods called traditional methods that try to correct the missing part on the main title picture only with the features in the picture to be corrected.

Machine learning methods, which have been used frequently in many different problems due to increased processing power and broad access to large data sets, have also started to be used for image inpainting problems. These ideas, which will be explained under the title of traditional methods, also formed the basis for machine learning and deep learning methods used for inpainting. The methods explained under this title were used in different deep learning methods, which will be examined later.

It would be more accurate to examine traditional methods mainly as exemplar-based and diffusion-based. Although Exemplar-based methods are divided into pixel-based and patch-based, patch-based methods will be examined mainly in exemplar-based methods, since there are more studies relating to patch-based methods and they give better results. [32] and [33] are some of the traditional method example works.

As can be understood from their names, patch-based methods try to fill the missing region in the picture with the information obtained from different parts of the picture. Diffusion-based methods try to fill the region by diffusing surrounding pixels from its boundary [34], [35].

Traditional methods are methods that do not require training. For this reason, it is an appropriate method to use for simpler problems. Traditional methods do not work well in problems where the masked area in the picture starts to form a proportionally large area. This result is quite normal given that these methods infer from pixels near the missing region.

2.1 Patch-based Methods

Patch-based image inpainting consists of methods that use the undamaged regions in the image to create the region to be filled. With this approach, it is aimed to have the highest possible patch similarity level [36].

In order to implement this method, an algorithm is needed on how to transfer the obtained regions to the areas planned to be filled. Another important point in this method is the selection of the algorithm that will compare the regions in different parts of the picture with the regions desired to be created [37].

While patch-based methods work well, they assume that the information in the missing part of the picture resides elsewhere in the picture. Another problem with patch-based methods is that they require relatively more processing power because they are methods that consistently involve searching and comparing.

2.1.1 Patch-based Texture Synthesis

In this method by Zhou et al. [1], it is aimed to fill the desired area by creating textures with patch-based algorithms for Image inpainting method. While implementing this targeted method, a road starting from the rough, proceeding in detail is aimed. While performing the filling process, targets and constraints were created in order to ensure the integrity of the newly created region with the rest of the picture.

In this study, a more efficient inpainting study has been achieved by basically using the successive elimination algorithm and based on the previously used methods.

Texture synthesis is performed by trying to imitate information about texture and structure integrity. At the same time, it is aimed to reduce the processing power used by using the successive elimination algorithm during texture comparison.

In this approach, the function given in equation 2.1 is tried to be minimized in order to achieve the desired result.

$$I_t = \frac{1}{k_p} \sum_{q \epsilon \Omega} I_q f(|p-q|).g(|I_p - I_q|) \tag{2.1}$$

However, minimizing this equation does not mean that the energy function decreases in general. To explain better, the region to be filled in the image can be completely filled

with a smaller patch than itself and gives a perfect result in the equation. However, for better results, coarse images as are used in equation 2.2 and equation 2.3.

$$\lambda(x,y)(\mu_1-\mu_0)=\mu\Delta^2\mu 1, where\frac{\partial\mu_1}{\partial n} \tag{2.2}$$

$$\lambda(x,y)(\mu_2-\mu_0)=\mu\Delta^2\mu 2, where\frac{\partial\mu_2}{\partial n} \tag{2.3}$$

In this way, the image inpainting process for the visual is corrected to give a good result from the rough approach. However, this process requires a high processing power due to 3 different partial differential equations.

$$E(c_k,C)=\sum_k\int_{\Omega_k}\lambda(x,y)(c_k,\mu_0)^2dxy+V|C| \tag{2.4}$$

It is logical to use an approach to solve this problem. With this approach, uniform visual densities can be accepted as constant for different regions. Hence, MS energy function can be reduced to equation 2.4. Also, an inpainted image example with its input image is given in figure 2.1.

Figure 2.1 : Inpainted Image with Patch Based Texture Synthesis [1]

2.1.2 Image Melding

The image melting method [2] creates a transition zone to transfer various information and properties within one image to another image. This method can be used for different purposes depending on the usage type.

This method, primarily uses many different transforms to increase the ability to search for patches. Next, image gradients are included in patch representation and Poisson equation solver is used for color averaging. Without causing blurry areas in the picture,

a new energy method is used based on L_2/L_0 norms for colors. Melding of 2 different image is illustrated below in figure 2.2.

Figure 2.2 : Image Melding Example [2]

Many different patch-based methods have been used for inpainting problems. Image melding is basically a patch-based algorithm. However, when based-patched methods are given inconsistent visuals, they can give insufficient results because they use a direct distance function between pixels.

Another result aimed with this study is that patch-based and gradient methods can be used together and thus more complex problems can be solved. Thanks to the methods used, Image Melding can be used to solve many different problems including inpainting.

Image synthesis using a single source is one of the ways image melding can be used. In this algorithm, a mask divides the image into two parts, S and T, respectively, as source and target. After this process, the energy function shown in equation 2.5 is obtained.

$$E(T,S) = \sum_{q \subset T} min_{p \subset S}(D(Q,P) + \lambda D(\Delta Q, \Delta P)) \tag{2.5}$$

In the above equation Q represents the targeted patch with w width and height and target pixel q on the upper left corner. P represents the transformation that is applied

to the neighbor pixels of the source pixel. Throughout the operations, all patches have 5 channels, which 3 of them are RGB and 2 channels are gradient channels. In the equation 2.5, RGB channels are shown with Q and P, gradient channels are shown with $\Delta Q \Delta P$Distances for all channels are expressed with D and, weight the gradient channels in according to RGB channels.

Two important steps during the implementation of the algorithm are searching and voting. For the search part, the PatchMatch algorithm is properly used. With using this algorithm, patches with sufficient proximity are detected for the target patch. Translation, rotation and scaling are used freely during this process. At the same time, ranges are defined on the channels in order to obtain better results.

$$T = argmin(D(I, \bar{T} + \lambda D(\Delta I, \Delta \bar{T})) \tag{2.6}$$

In the voting part, an algorithm is used in which every pixel in the patch is included, and the resulting output takes its form in eq 2.6. In figure 2.3, Inpainting example with image melding method is given.

Figure 2.3 : Inpaingting Example with Image Melding [2]

2.2 Diffusion-based Methods

This method basically fills missing regions or unwanted objects in an image by diffusing neighboring pixel information. Diffusion process uses partial differential equations. Inpainting is not the only use of such diffusion-based methods. In addition, these type of methods are available in areas like image compression. Diffusion-based image inpainting can be used to remove undesirable objects in an image or to repair damaged parts of it. However, as a downside, these methods can leave artifacts after

inpainting process such as local variance difference and noise pattern [38]. In figure 2.4 an example of diffusion-based inpainting can be seen. The black region in 2.4.a is the unknown region and the 2.4.b shows the inpainting result, the region surrounded by the dashed curve is recovered by inpainting.

Basically, the inpainting process is nothing more than restoring the missing pixel information in an image and there are some common assumptions that most of the images have commonly share. In this way of thinking, Navier-Stokes based method [39] was introduced. Navier-Stokes equations known in the area of mechanics, especially fluid mechanics. This inpainting method emerged by combining ideas in the field of classical fluid dynamics with the help of partial equations and remains one of the most popular traditional methods to this day. The assumption that the edges in an image are continuous is used. The authors designed this inpainting method by considering the color information of the surrounding pixels and the this continuity constraint.

Figure 2.4 : Diffusion-based inpainting example [3]

Another traditional inpainting study based on fast marching method [35] proposed that, in order to estimate the color of the missing pixels, gradients of the neighbor regions can be used. In our work, these two methods used as a baseline when comparing the deep learning based inpainting methods with traditional inpainting methods.

2.2.1 Navier-Stokes Method

Navier-Strokes equations are partial differential equations for describing the motion of fluid substances. These equations used in variety of areas in different industries such

as modeling ocean currents, weather conditions and even when designing aircrafts and cars.

$$\omega_t + \upsilon \cdot \nabla\omega = \nu\nabla \cdot (g(|\nabla\omega|)\nabla\omega) \tag{2.7}$$

In this method [39], the stream function formulation used in fluid mechanics has been made suitable for the inpainting process. The counterpart to the vorticity-stream function for inpainting transform into equation 2.7 which corresponds to the solution of the intensity value of a pixel. [34] Similarly, the fluid velocity in fluid dynamics corresponds to the gradient of the intensity values also interpereted as isophote direction, vortocity corresponds to smoothness and viscosity corresponds to diffusion respectively in this problem.

The algorithm used in this study proceeds by following the edges in the picture from the known region to the unknown region. Since it is assumed that the edges will be continous during this progress, isophote lines are preserved. Isophote means the lines created by pixels with the same intensity value. During this process, the direction of the gradient vector in the boundaries preserved. After the isophotes are obtained, color information is calculated to reduce the minimum variance and by filling in the unknown pixels, the inpainting process is completed.

2.2.2 Fast Marching Method

In this study published in 2004 [35], Ahmet developed an inpainting technique based on the fast marching method. A filling process is designed starting from the boundary pixels of a region where the pixel information is missing and going step by step towards the inside. It takes a small neighborhood of pixels to calculate the unknown pixel value as the weighted sum of known pixels inside that boundary. For this weighted summation process, the weights selected according to the proximity and boundary normal. After all the pixels are calculated in each step, the next closest pixel is passed with the fast marching method and the same process is repeated again until everything inside the region is inpainted. FMM ensures that the pixels which are near to the known pixels inpainted first and inside pixels will be filled afterwards. This inpainting priciple

can be seen in figure 2.5. Pixel p in figure 2.5.a will be inpainted. Inside a small region ε, for any q known pixel, the gradient ∇I will be calculated as seen in figure 2.5.b.

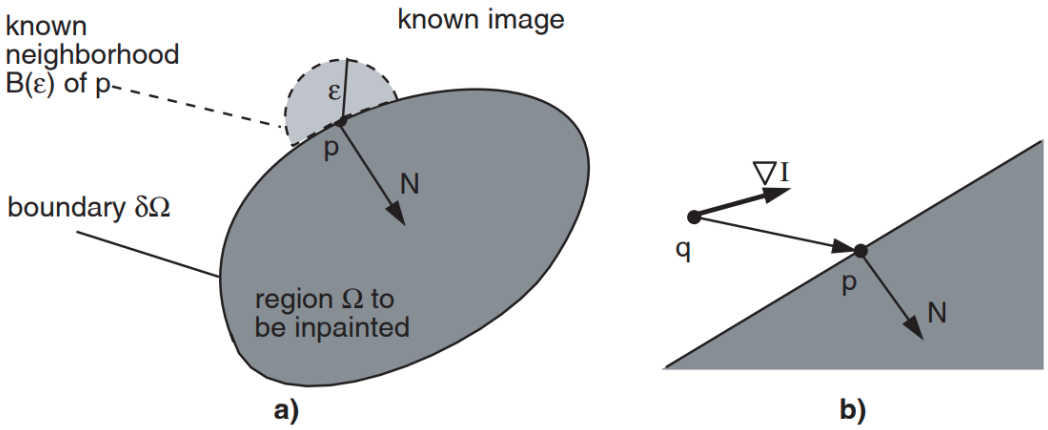


Figure 2.5 : Fast marching method inpainting priciple

In figure 2.6, comparison of the Navier-Stokes and Fast Marching approaches can bee seen. White colored pixels in 2.6.a are the missing areas and 2.6.d is the original image. In 2.6.b, output of the Navier-Stokes method and in 2.6.c, output of the Fast Marching method shown respectively.

Figure 2.6 : Comparison of Navier-Stokes and Fast Marching methods

3. INTRODUCTION TO MACHINE LEARNING

3.1 Machine Learning

Machine learning is the modeling of complex systems on mathematical and statistical operations that perform tasks such as regression, classification and clustering using data. Thus, with the help of these algorithms computers can perceive mixed patterns in data. Today, there are many different machine learning methods for variety of data types. These learning methods are generally divided into three categories. These categories are supervised learning, unsupervised learning and reinforcement learning respectively. Supervised learning methods use training data with corresponding labels which defined beforehand. Label also known as ground truth represents the expected output of the machine learning system. This type of learning algorithms try to come up with a mapping function between training data and the ground truth. As an example, speech recognition [40] or a regression problem such as market forecasting [41] can be solved using this type of learning. The second category is the methods implemented using only data without any corresponding ground truth. Two example of the areas of use of such unsupervised learning algorithms are the clustering problem in the recommender systems [42] and the dimension reduction [43]. The last category, reinforcement learning, is used in solving problems that require real-time decision making. For example, in the field of robotics, the methods under this category are used in the design of the systems to decide what action to take against any state in an environment [44].

Deep learning is a subclass of the much broader area of machine learning applications. In order to comprehend the methods mentioned in this thesis well, it is necessary to have a knowledge of statistics, calculus, optimization and computer vision as well as machine learning basics. For this reason, some machine learning basics are mentioned in the following chapters.

3.2 Neural Networks

The idea of a neural networks are although they could not be realized efficiently for a long time, is an idea in machine learning field that emerged more than 70 years ago, and represents a similarity to neurons in the human brain. First neural network is created by McCulloch and Pitts, however they did not have enough technology to run it. Later in 1954, Clark and Farley at MIT were able to run the first Neural Network.

The main feature that enables neural networks to work is that they are made of neurons. These neurons are similar to neurons in the human brain. The basic requirement for understanding how neural networks work is to understand how neurons in the human brain work and how it connects to Artificial Neural Networks (ANN).

There are billions of neurons in a human brain. These neurons communicate with each other using electrical signals. Neurons produce their own output according to the different types of signals they receive. If the value they receive is above a certain limit, it generates electrical impulse in response.

As with a real neuron, a neuron in the ANN also functions similarly. The neurons in the ANN simply take the input they receive and process this data and then give an output. Each neuron in the network has its own parameters called weights. This decision-making process produces an output yes or no. Hence, it is possible to call this process a classifier. Similar to a neuron function, a threshold function is illustrated in figure 3.1.

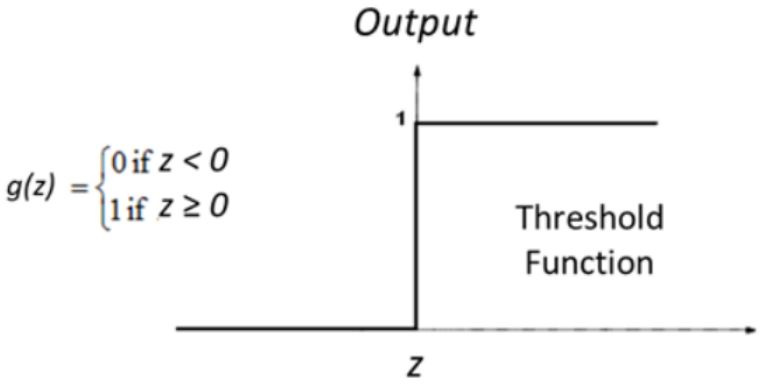


Figure 3.1 : Threshold Function

There are different numbers of layers in ANN depending on their size. Neurons transmit their output from each layer to the next layer. As will be explained later, the

output from each neuron is inserted into activation functions to ensure non-linearity within the system. Neuron structure is shown in figure 3.2.

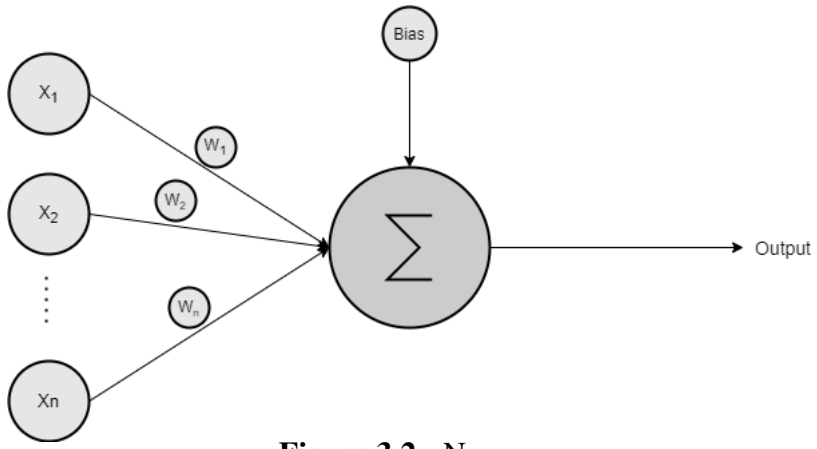


Figure 3.2 : Neuron

Based on the ideas previously announced, McCulloch and Pitts designed the first neural network. In the system they designed, many inputs are given to each neuron as input, and the output of the neuron can be obtained by adding these inputs after multiplying with weights. The output of this neuron is also directed to the activation function.Operations done in a neuron are shown in 3.1 and 3.2.

$$Out = \sum_{i=1}^{N} I_i W_i \tag{3.1}$$

$$Y = f(Out) \tag{3.2}$$

3.2.1 Perceptron

Mainly perceptron is a machine learning algorithm that classifies an input. Perceptron performs this classification with a linear function which uses weight and bias values. With this feature, perceptron forms the basis of all neural networks.

$$Out = \sum_{i=1}^{N} I_i W_i + B_i \tag{3.3}$$

Perceptron consists of 4 basic parts as input value, weight, bias and activation function. As can be seen, the innovation that perceptron adds to neural networks is that the bias value is added to each process as shown in 3.3.

As was the case when neural networks were first launched, the perceptron could not achieve the expected success due to technical limitations. There are 2 types of Perceptron models which are called single-layered perceptron model and multi-layered perceptron model. It contains a feedforward network within the single-layer perceptron model. It is the simplest neural network that can be implemented. An illustration of a Feed-Forward network structure is shown in 3.3

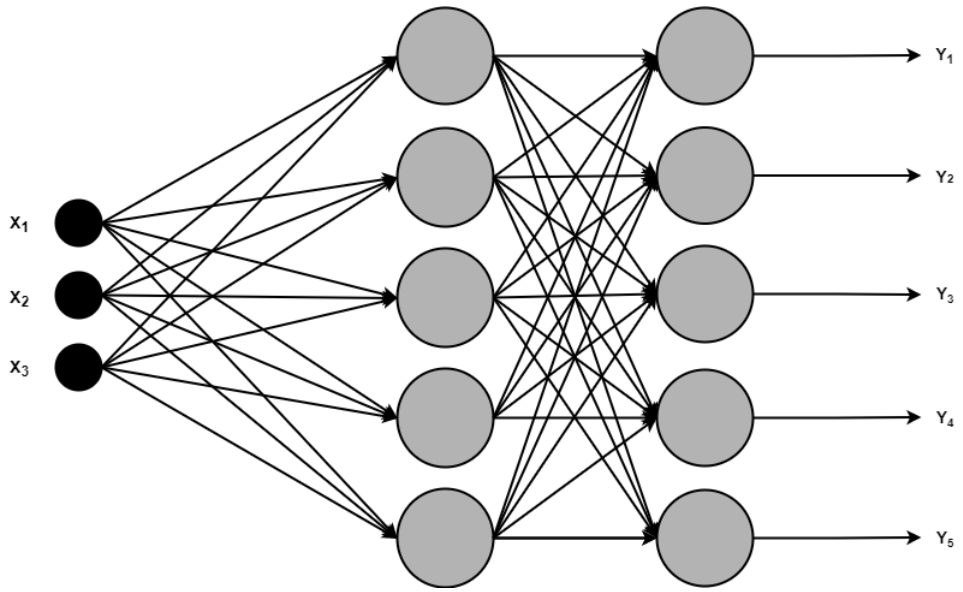


Figure 3.3 : Feed-Forward Network Structure

As seen in Figure 3.3, each neuron in the first stage is connected to the neurons in the other layer. Combinations between all neurons require considerable high processing power which allow the solution of more complex problems. All neurons in the input layer direct their result to the output layer. If all the neurons between the layers are connected in a network, the network that is obtained can be called a fully connected network. On the other hand, if it is stated that the weight value between two different neurons, it is understood that there is no connection between that two neurons.

It is also pretty significant to state that single-layered perceptron models are only applicable to problems with linearly separable classes. Later on in the future, it is discovered that multi-layered perceptron networks can be used for non-linearly separable problems. Data distributions for different problems are given in figure 3.4.

Even though multi-layered perceptron networks are similar to single-layered perceptron networks, they contain fundamental differences. There are many more

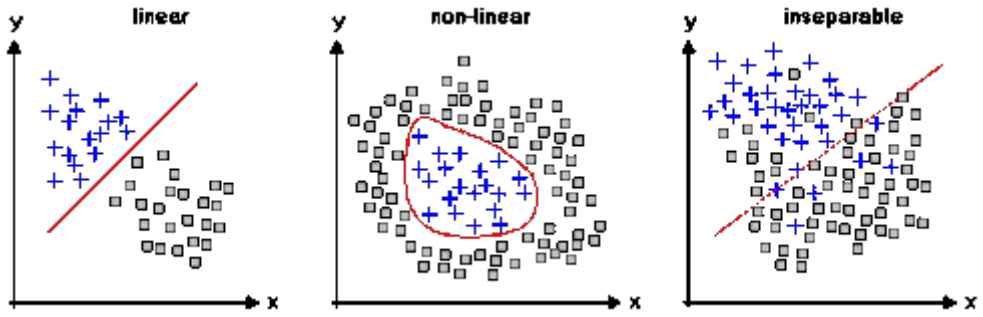


Figure 3.4 : Data with Different Distributions [4]

hidden layers in multi-layered perceptron. Also, another important difference is that the multi-layer perceptron has a backward stage in addition to the forward stage.

In forward stage, a path from the input layer to the output layer is followed. In oppose to that in the backward stage, an error is obtained from the value from the generated output and the value that should actually be obtained, and this error is used to change the parameters in the neurons which are weights and biases.

One other important addition is the usage of non-linear activation functions in multi-layer networks. With these improvements, multi-layered networks are able to work on non-linearly separable problems.

3.2.2 Loss Function

In mathematics, loss function also known as cost function is a function that represents the real "cost" of anyting to a real value. In essence, optimization problems try to minimize the output of the loss functions. They are designed to take one or more values as input and return a real number as output. Mean square error (MSE) is a simple loss function commonly used in neural network models. It is the sum of squared distances between the predicted values and the corresponding ground truth values. An optimizer will update the neural network parameterswhile trying to minimize this loss. Cross entropy loss [45], also known as log loss or logistic loss, mostly used in classification problems when the values of the output layer is either zero or one. For binary classification problems binary cross-entropy loss can be used. If there are more than two classes in the problem categorical cross-entropy loss is preferred. In equation 3.4 the binary cross-entropy formulation is presented where y is the binary indicator if label is correct classified or not and p is the predicted probability of that label. To sum

up, loss function is a mathematical way of measuring the error between the prediction and ground truth in order to see how the model is performing.

$$L = -(y \cdot log(p) + (1-y) \cdot log(1-p)) \tag{3.4}$$

3.2.3 Optimizer

Algorithms that try to minimize the output of the loss function by changing the model parameters during the training phase are called optimizers. Optimizers are algorithms that calculate how and how much should the values of parameters updated. One of the most popular optimizer used in neural network applications is the Gradient Descent algorithm [46]. Simply put, this algorithm looks at the gradient of the loss value effect of a small change in network parameters. In computer vision studies, advanced optimizers such as Adam [47] used.

3.2.4 Activation Function

Activation function is a spesific function for calculating the output of a single neuron in the neural network. Problems which are linearly non-seperable can be solvable using non-linear activation functions such as sigmoid, hyperbolic tangent (tanh), and ReLU. In concolutional layers ReLU activation function is commonly used. The main advantage of this activation function is that while casting negative neuron output values to zero it allows a regularization in the network via not activating every neuron. These mentioned activation functions can be seen in figure 3.5.

3.3 Convolutional Neural Networks

Convolutional Neural Network is a widely used neural network architecture, especially thanks to its stunning success in computer vision studies [48]. In this architecture, structures such as convolutional layer, pooling layer, flattening layer are used. CNNs are the structures that achieve only important features of the images by applying the convolution process using two-dimensional filters and obtaining different level of information in the image [49]. CNN takes advantage of the hierarchical patterns in the data by following a path different than MLP structure. It brings together small and

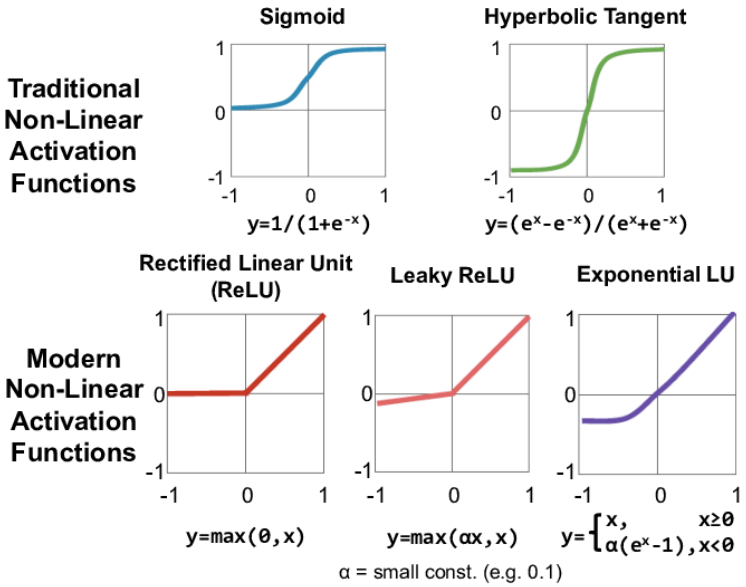


Figure 3.5 : Activation Functions [5]

simple patterns to obtain more complex patterns. They use convolution operation in place of general matrix multiplication.

In the convolution layers, as seen in the figure 3.6, 2D convolution is done between a tensor and the specified filter. In tasks such as inpainting where colored RGB images are used, these filters are applied separately to each color channel. During the convolution process while applying these filters which usually have 3x3, 5x5, or 7x7 size kernel onto matrix, on the borders padding process is applied. During this process, padding determines the course of action when values that are outside the limit and do not exist are desired to be accessed by convolution operation.

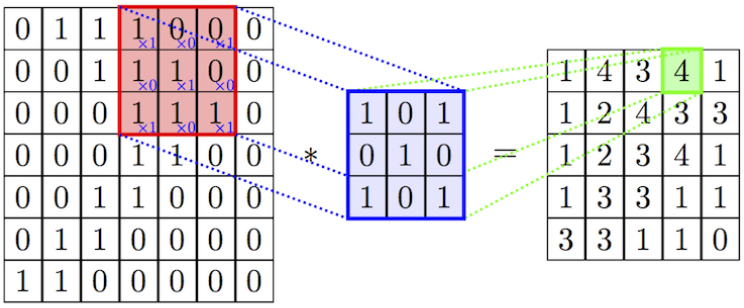


Figure 3.6 : 2D Convolution [6]

Pooling layers, generally following the convolutional layers, reduce the size of the resulting tensor. In many convolutional neural networks, max pooling is used, but there are also min pooling and average pooling layers. The given tensor is divided into smaller pieces and a new and smaller tensor is obtained by taking the maximum,

minimum or mean value of each sub-piece according to the chosen method. In figure 3.7, average-pooling and max-pooling methods are shown.

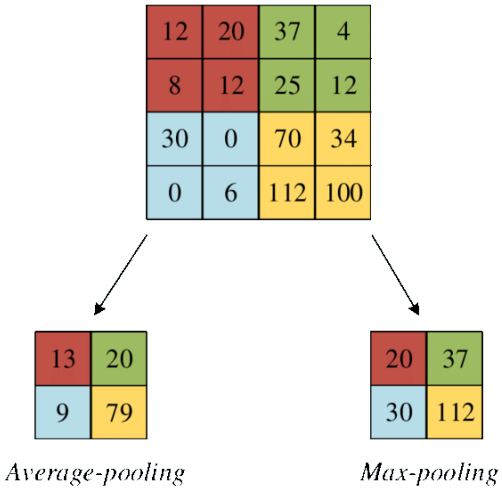


Figure 3.7 : Pooling [6]

Flattening layer is the layer that makes the tensors in CNN structures vectorized in order to use the fully connected layers which commonly used in MLP structures. This layer serialized the incoming tensor and fills all the values to the neurons in layer.

There are two CNN architectures commonly used for deep inpainting studies. First one is the encoder-decoder structures, and the other one is a derivation of the former one which called as autoencoder structure. These architectures performs well on many computer vision task and also shows promising results on other inverse problems in computer vision such as inpainting, denoising, and super-resolution. Moreover, this structures can work on different deep learning problems such as sentiment analysis, image captioning, and translation. For example, Google Translate is built upon an encoder-decoder structure [50].

3.3.1 Encoder-Decoder Networks

To explain briefly, encoder network can be in fully connected structure or convolutional structure or even as recurrent neural network (RNN) structure [51]. An encoder network will take input and output a feature map which is simply a vector or more generally a tensor. This feature vector holds the necessary and important feature information about the input. The decoder network is usually the same network architecture as an encoder but in reverse. It takes a feature vector and tries to generate

the input as closest as possible. Encoder-decoder network consist of two network parts which are an encoder and a decoder. Input processed through these structures. These network structures can be used in various areas such as language translation [50] and in computer vision applications as generative models. Mentioned model architecture can be seen in figure 3.8.

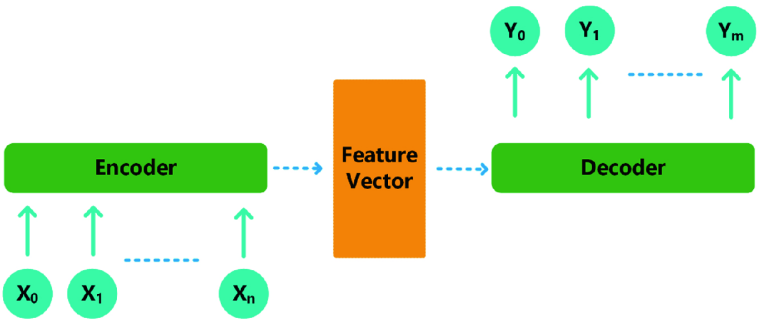


Figure 3.8 : Encoder-Decoder Network Structure [7]

3.3.2 Autoencoder Networks

Autoencoder is a structure that aims to establish a connection between input and output by obtaining a code in hidden layers. The code concept here can be considered similar to the feature vector in encoder-decoder network structures. The most typical use of autoencoders is dimension reduction. Autoencoder structured neural networks can be used in unsupervised non-linear dimension reduction applications. In figure 3.9 a comparison between a traditional dimension reduction method PCA [43] and a autoencoder structure.

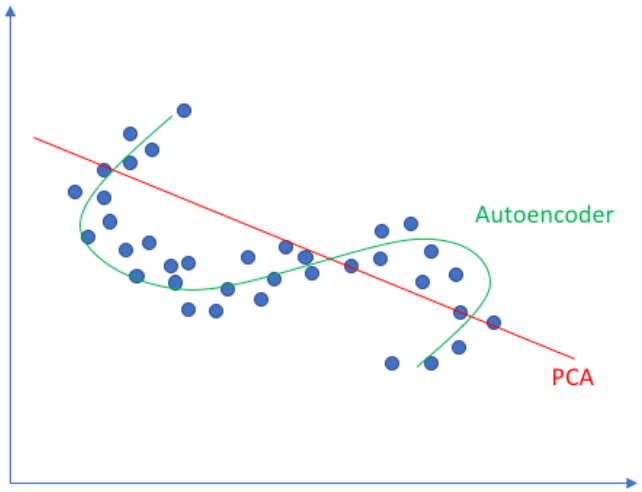


Figure 3.9 : Autoencoder vs. PCA dimension reduction task [8]

By training the autoencoders it becomes possible to ignore the noise in data. For instance, it can be used in anomaly detection application over any signal [52] [53]. Another example application would be to create a noiseless image from a noisy image by obtaining the most important features while ignoring the noise [54]. This application is called denoising which is a popular inverse problems in computer vision studies. Learning with autoencoder structures is actually possible with an encoder that can best describe the input image and a decoder that can give an output as similar as possible to the original image using this description from former encoder structure. A convolutional autoencoder network structure designed for the image classification application is shown in the figure 3.10.

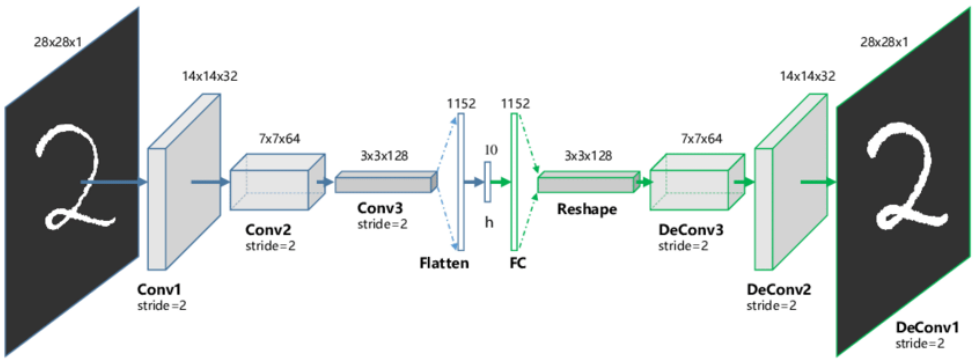


Figure 3.10 : Convolutional autoencoder network structure [9]

3.4 Generative Adversarial Networks

As time passed, many different machine learning methods and ideas emerged. As explained in the neural network and machine learning sections, these methods were methods that work to make operations on the input to obtain meaning, or methods that uses datasets with huge amount of data with expected outputs to converge the model to give correct result.

As machine learning methods and specifically neural network and deep learning methods began to be used more frequently and the number of areas in which they were used increased, the methods used continued to differ and develop. One of these ideas developed is Generative Adversarial Networks.

It is possible to categorize methods of machine learning under 2 main categories, which are supervised and unsupervised learning. Even, it is possible to obtain successful results with supervised learning labeling huge amount of data is not economical and

highly time consuming or in some cases it is not possible. Basically, the basic idea behind the emergence of the generative models is that there is no possible data that can be compared with output, or rather, the desired output does not have a correct answer to compare. Objectives of generator and discriminator models is given in figure 3.11.

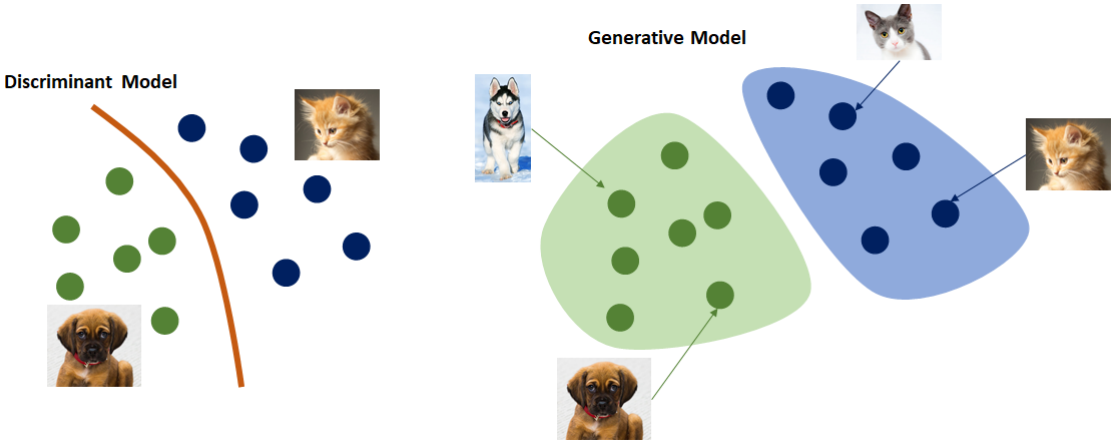


Figure 3.11 : Difference Between Generative and Discriminative Models [10]

There are different type of generative methods in machine learning. Most generative models developed over Markov chain and maximum likelihood estimation. Also, there are generative models such as Restricted Boltzmann Machine [55] and Deep Belief Network [56]. A new model called Generative Adversarial Networks (GANs) is proposed by Ian Goodfellow [57] and it has been used in various problems in the area of computer vision.

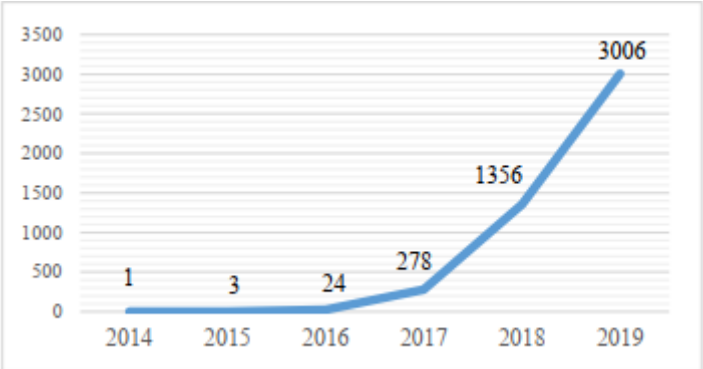


Figure 3.12 : GAN Citation Counts [11]

With the use and development of GANs in many different fields, GAN structures have become one of the leading methods of unsupervised learning. In figure 3.12, work counts related to GAN is illustrated.

There are 2 different networks in the GAN. The word adversarial in its name comes from the fact that these structures work against each other throughout the training procedure.

As explained by Goodfellow [57], within the GAN structure there are 2 networks being trained simultaneously. These networks are called generative and discriminative, respectively, G and D. In this structure, G is responsible for the creation of new data, while D tries to predict the probability of the data given as input to it.

From this point of view, it can be said that the Generative and Discriminator structures correspond to the minimax two-player game. Also, if G and D are constructed as perceptrons with multilayers, it can be said that the backpropagation can be used to train the whole structure. Simplified structure of GAN can be seen in figure 3.13.

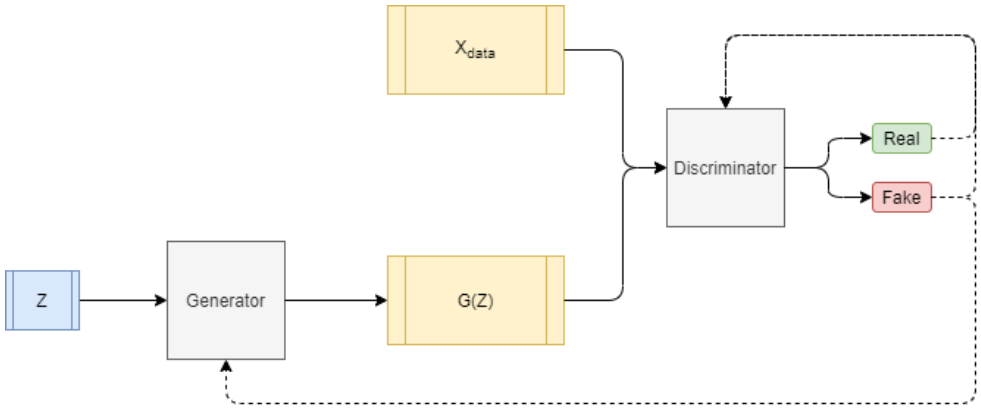


Figure 3.13 : GAN Model Explanation

When the mentioned adversarial model structure is implemented with perceptron, its application becomes straightforward. As a first step, generator structure G functions as a mapping to obtain data distribution from the input given to it. The discriminator structure created creates only a scalar as output. Structure D is trained to better label data from G and dataset. However, G is also trained to minimize log (1 - D (G (z))).

To explain, D and G play minimax with the function V (G, D) given in equation 3.5.

$$min_G max_D V(D,G) = E_{x\ P_{data(x)}}[logD(x)] + E_{z\ P_z(Z)}[log(1 - D(G(z)))] \tag{3.5}$$

When implemented, it can be seen that Equation 3.5 could not give sufficient results in the first steps of training. As can be guessed, at the beginning of the training phase, the Discriminator network could easily distinguish generated data from the real data which

causes log(1-D(G(Z)) to saturate. In this case, it is reasonable to train to minimize log(D(G(Z)) rather than train to minimize log(1-D(G(Z)).

To exemplify the success of GAN structures, examples of faces produced by this method in different years are shown in Figure 3.14.

Figure 3.14 : GAN Success Over Years [11]

3.4.1 Deep Convolutional GAN

Deep Convolutional Generative Adversarial Network [12], shortly DCGAN, is a frequently used addition to GAN structures. The generator and discriminator used in this structure are in deep convolutional structure and thus a more stable training is performed. With this work, it is aimed to use the advantages of CNNs.

DCGAN is an important development in terms of having more successful and high quality generators. At the same time, this structure has helped the emergence of many different types of GAN structures. Overall structure of DCGAN is illustrated in figure 3.15.

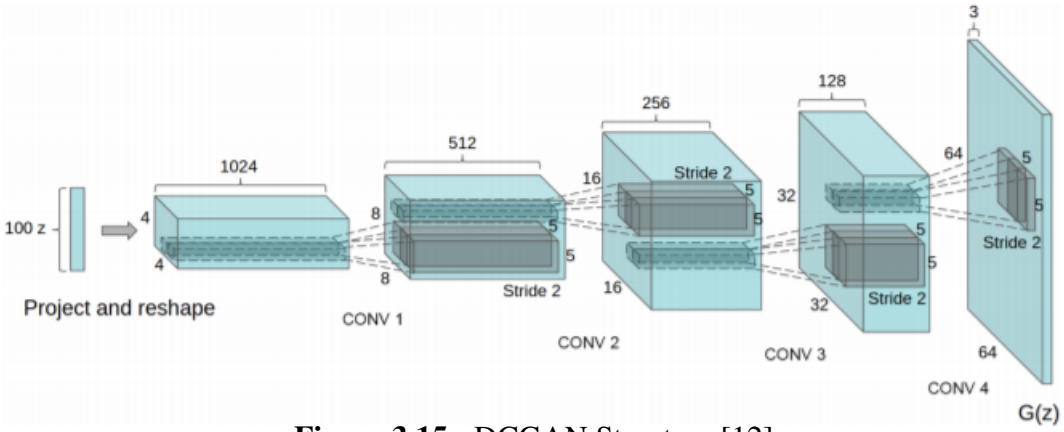


Figure 3.15 : DCGAN Structure [12]

3.4.2 Wasserstein GAN

As GAN applications become widespread, different types of GAN structures have started to emerge. One of them is the Wasserstein GAN structure. This GAN structure both increases the stability of the GAN and provides the use of a loss function better related to the quality of the images produced.

WGAN model is a study presented by Arjovsky et al. [58], that adds additional features to the GAN structure. As mentioned earlier, images in the WGAN structure, receive a score that expresses the reality of the image rather than simply being classified as real or fake by the discriminator.

Fundamentally plans to be a solution to Discriminator and Generator balancing, one of the main problems of GAN structures, with the new Earth Move distance using WGAN. Earth-Mover distance is shown in equation 3.6.

$$W(P_r, P_g) = inf_{y\varepsilon \prod(P_r,P_g)} \mathbb{E}_{(x,y)\ y}[||x-y||] \tag{3.6}$$

4. DEEP LEARNING BASED INPAINTING METHODS

As with many different computer vision problems, deep learning methods have been used for image inpainting. The reason why deep learning methods are being used more frequently as time progresses is that they provide more successful results in complex problems compared to traditional methods. The main reason for this is the creation of large-scale datasets that will enable the training of deep methods and the computational power that allow the training of these deep methods.

4.1 CNN-based Methods

Convolutional neural network structures, which are known to be very successful in computer vision studies thanks to their grid-like layer topology, are also used in image inpainting studies and give outstanding results. There are many architectures specially designed for inpainting work.

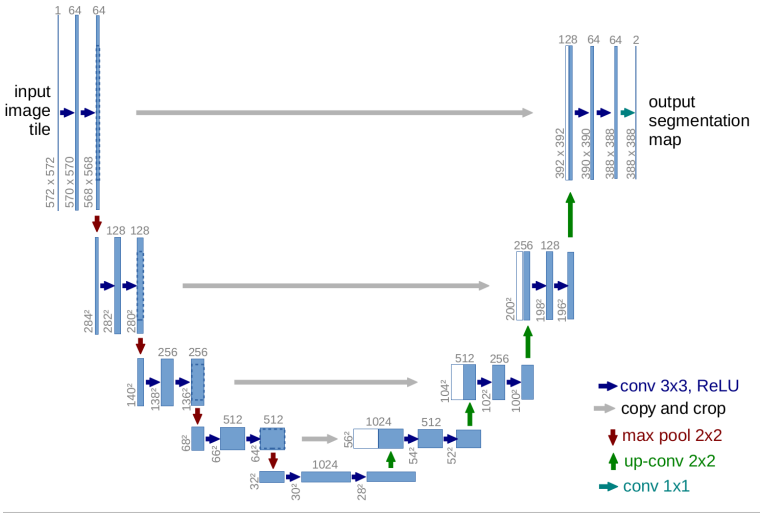


Figure 4.1 : U-Net Architecture [13]

The U-Net [13] architecture used in Shift-Net inpainting study [14] is one of them. The U-Net CNN architecture is shown in figure 4.1 This architecture takes an image and a mask which shows the missing areas and put them into convolutional layers. It concatenate the output of each layer with the corresponding layer of the same size

in symmetrical architecture. The results of this structure achieve excellent success in terms of generated image structure and fine detail. The architecture of the Shift-Net which is a U-Net with special shift-connections can be seen in figure 4.2

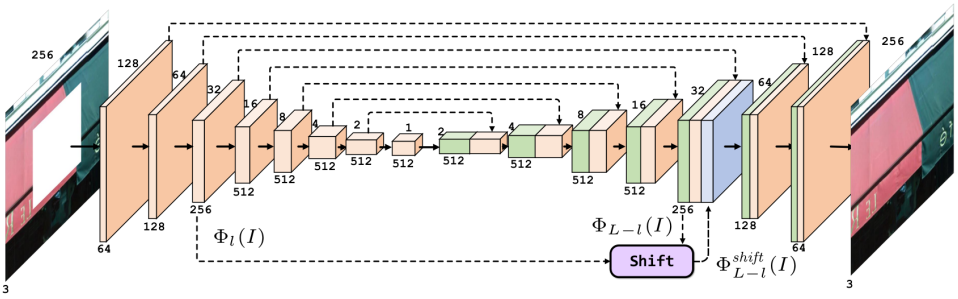


Figure 4.2 : Shift-Net Architecture [14]

Another popular architecture used in inpainting studies is the encoder-decoder network. Sidorov and Hardeberg [59] uses a 3D CNN encoder-decoder network architecture and can perform not only inpainting but also tasks such as denoising and super-resolution. In the study of Liu et al. [15] coherent semantic attention layer designed and used in an encoder-decoder structure called refinement network. This architecture can be seen in figure 4.3.

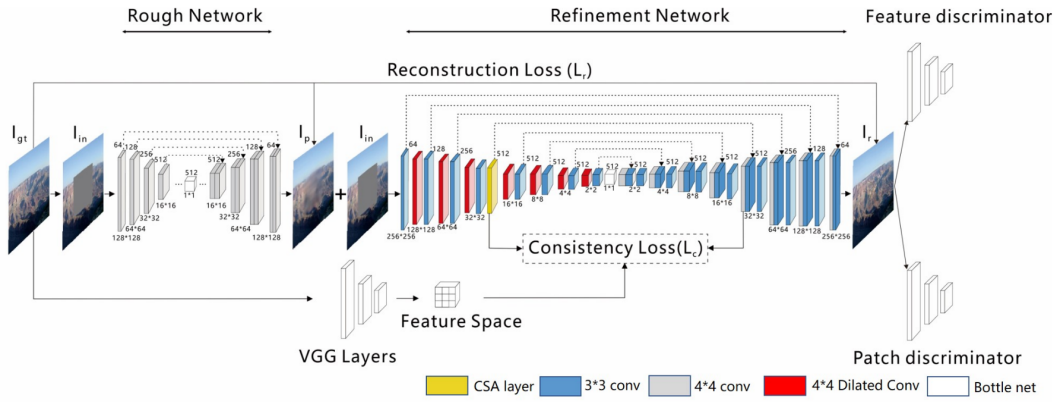


Figure 4.3 : CSA layer at the resolution of 32×32 in refinement network [15]

In Artist-Net [16] proposed by Liao et al. In order to achive inpainting, two different content encoders, a style encoder used with a joint decoder. Artist-Net architecture can be seen in figure 4.4. A similar structure is also available in CNN architecture developed by Cai et al. [60] for the purpose of semantic object removal. There are also context encoder [18] architectures used in inpainting studies. These context

encoders basically CNNs trying to generate the missing parts of an image from their surroundings.

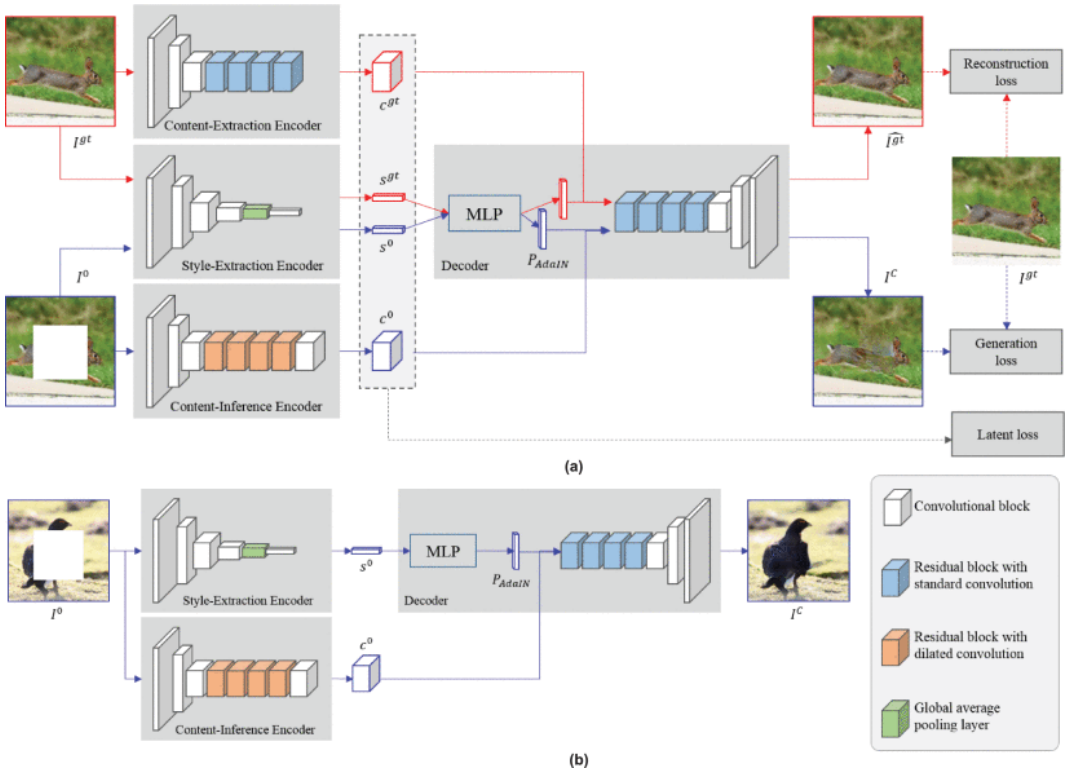


Figure 4.4 : Artist-Net Architecture [16]

Zeng et al. [17] used Pyramid-context encoder network to perform inpainting. This pyramidal architecture called PEN-Net performs a high quality inpainting operation. The network structure can be seen in figure 4.5. Nakamura et al. [61] aimed to remove the text from images and achieved excellent results using CNN.

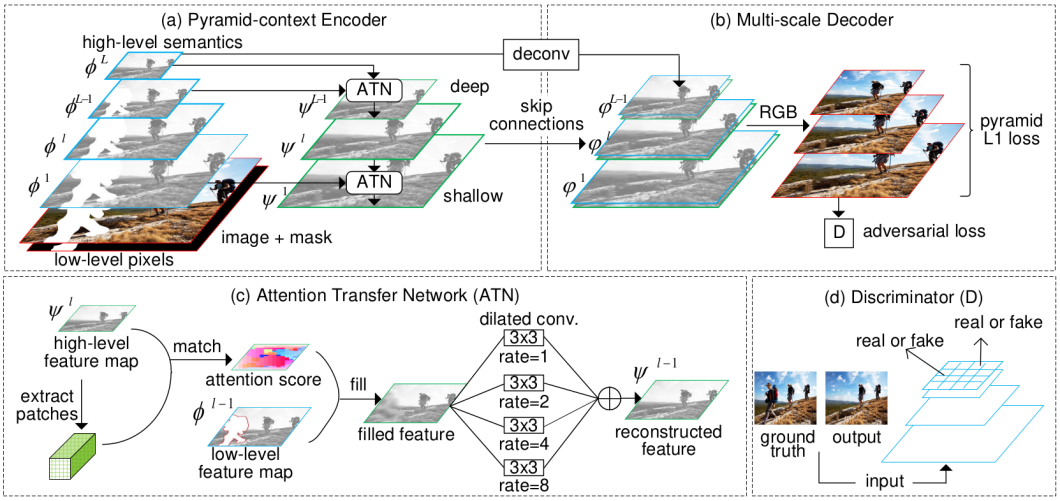


Figure 4.5 : The Pyramid-context Encoder Network (PEN-Net) [17]

4.2 GAN-based Methods

With different deep learning methods that have been tried over time, it has been analyzed that some methods give more successful results. It can be said that CNN and GAN based methods are better in evaluating realistic results in images. In this section, methods based on GAN idea will be explained.

In the paper, Context Encoder: Feature Learning by Inpainting [18], it is shown that an encoder-decoder architecture, first create a feature space with its encoder. Then, the decoder uses that space the create realistic inpainted output image. Moreover, using different losses altogether shows significantly better and more realistic results could be constructed. Figure 4.6 illustrates the structure of Context Encoder.

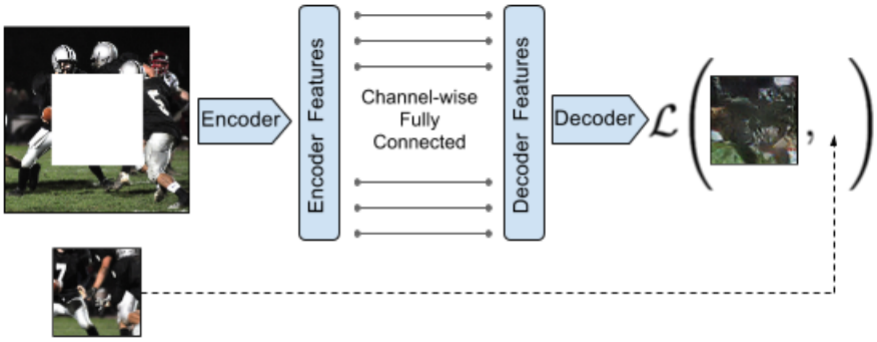


Figure 4.6 : Context Encoder Structure [18]

In addition to this study, different methods have been proposed to solve different problems. To prevent blurry parts in the created pictures, using less downsampling layers are proposed by lizuka et al. [27]. Furthermore, using dilated convolution layers instead of fully connected layers are proposed by Yu and Koltun [19]. Unfortunately, these changes led to longer training times because of highly sparse filters generated by dilation element. Structure example with dilated convolution layers are presented in figure 4.7. Gated and dilated gated convolutional layers used in an image completion study SC-FEGAN [62]. In this study authors built GAN-based system in order to complete masked input images with respect to the given sketches.

Different methods are also used such as using a pre-trained VGG network which leads to much shorter training time, proposed by Yang et al. [63]. Also, Liu et al. [64] came

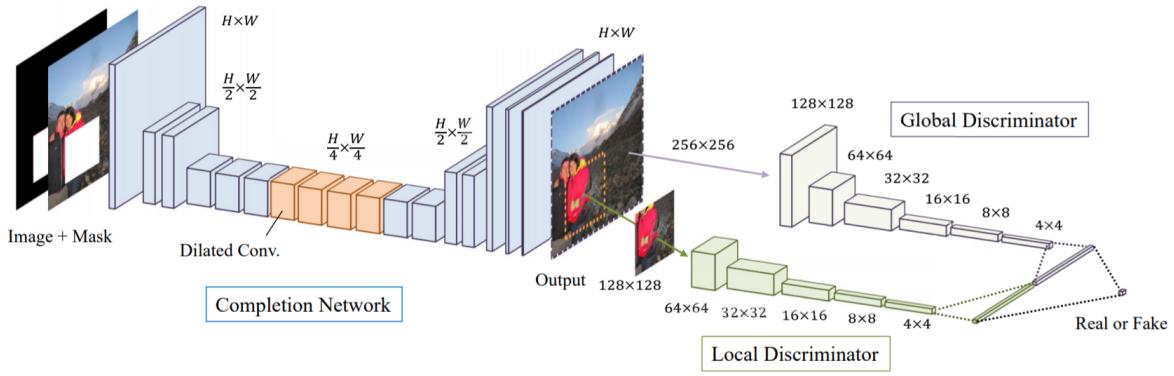


Figure 4.7 : Model with Dilated Convolution Layer [19]

up with partial convolution layers that saves convolution filters from taking too many zeros from masked regions. Partial convolution operation is shown in Figure 4.1.

$$x' = \begin{cases} W^T(X)\frac{sum(1)}{sum(M)} + b, & \text{if } sum(M) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

The novel idea that we actively use in our study and that provides the most successful results for the inpainting operation is the introduction of Generative Adversarial Networks asserted by Yeh et al. [20]. Fundamental idea behind GAN is illustrated in Figure 4.8 for an image related problem.

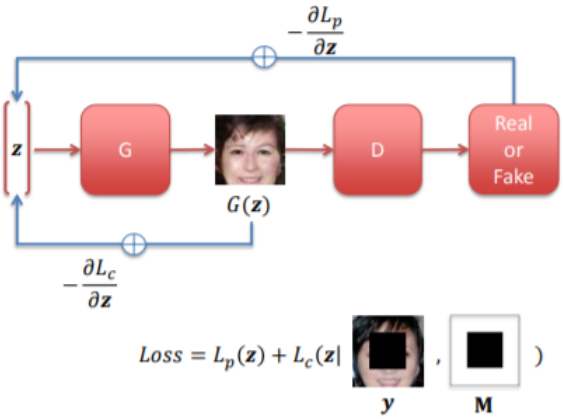


Figure 4.8 : GAN Idea [20]

Different methods that use GAN structures and add many innovations to this structure will be explained in detail in the following sections.

4.3 State-of-the-Art Methods

4.3.1 EdgeConnect

Nazeri et al. proposed a method [21] that uses edge information to produce realistic output images over missing zones. EdgeConnect consists of 2 different GANs placed one behind the other. The first of these GANs tries to create the missing edge information over the masked image, and the second GAN structure complements the missing image by using this created edge information and the original masked image. Fundamental idea of the structure is shown in Figure 4.9.

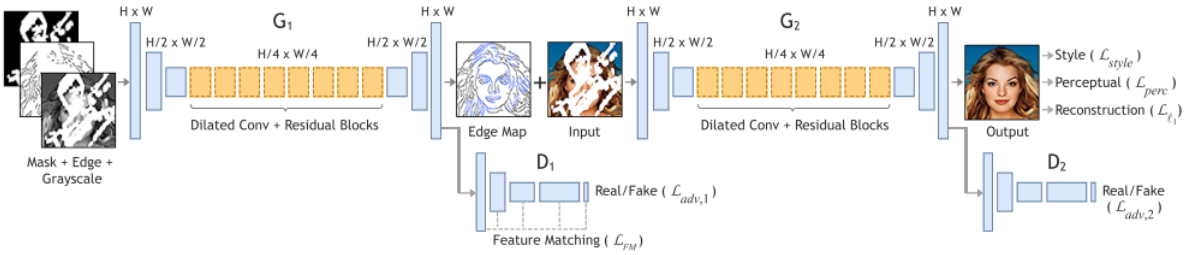


Figure 4.9 : EdgeConnect Network [21]

With this method, Yu et al. want to solve some of the basic problems that arise in the inpainting process. Using the edge information, it is desired to create visuals with sharper borders. Figure 4.10 demonstrates a few examples of inpainted images with EdgeConnect.

Similar features are available for both structures that create edge information and final image. Generator Network consists of 3 parts. As mentioned before, an encoder structure is used primarily for feature extraction from the image. Residual network has been added after the encoder structure. Finally, there is a decoder to produce the image from features. Convolution layers are used adjacently for the discriminator structure. The methods, transformations and tools we use to implement these structures will be explained in detail in the following sections.

The working logic of the model basically consists of 4 steps. Briefly, the image selected as input in each iteration is directed to the dataset script. In this dataset script, the process of finding an edge and separating the image from the color channels are

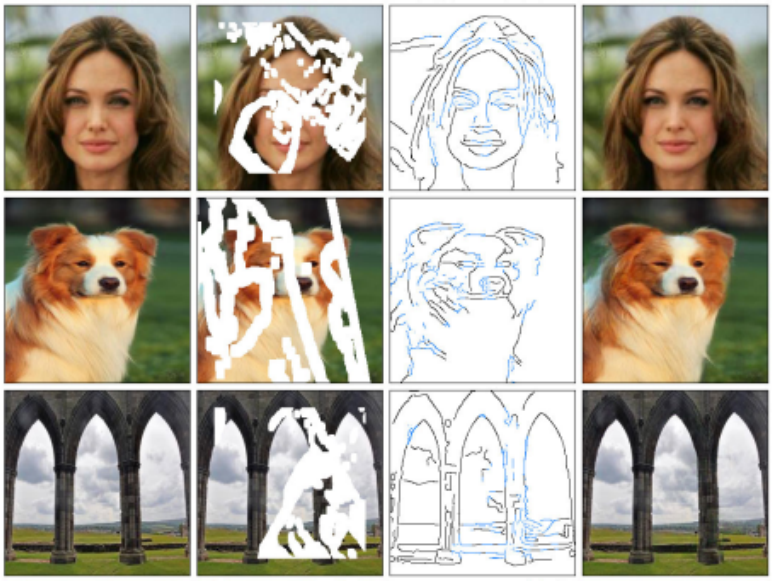


Figure 4.10 : EdgeConnect Inpainting Example [21]

performed. From the parts obtained in this step, the edge and gray image are given to the edge model as input. The output is reverse masked and summed with the original parts. In this way, we only get the part produced about the masked region of the output picture. Then, with the edge information obtained from this section, the RGB image is given to the inpainted model and again reversely masked.

Our goal was to implement the same model into our framework. To achieve this purpose, we got inspired by the example models that are publicly shared on internet. This was one of the main challenges of our project.

4.3.1.1 Edge Generator

Edge generation is the fundamental idea behind EdgeConnect. In this section, generating edge will be explained. Input image that is given to the system will be denoted as I_{gt}, which stands for ground truth image. Also, edge information of input image and gray scale version of input image are denoted as C_{gt}, I_{gray} respectively. Hence, while masked version of edge information and gray scale input images are represented as $I^m_{gray} = I_{gray} \odot (1-M)$ and $C^m_{gt} = C_{gt} \odot (1-M)$, edge generation can be expressed as equation 4.2. In these operations, $\odot$ represents Hadamard product.

$$C_{pred} = G_1(I^m_{gray}, C^m_{gt}, M) \tag{4.2}$$

C_{gt} and C_{pred} is given to discriminator network along with I_{gray}, which tries to predict if the given edge map is real. In a greater scale whole network is trained using an objective function which is given in equation 4.3 with adversarial loss and feature loss.

$$min_{G_1} max_{G_2} \mathscr{L}_{G_1} = min_{G_1} (\lambda_{adv,1} max_{D_1} (\mathscr{L}_{adv,1}) + \lambda_{FM} \mathscr{L}_{FM}) \tag{4.3}$$

In above equation, $\lambda_{adv,1}$ and λ_{FM} are called regularization parameters and they are adversarial loss and feature matching loss respectively. Adversarial loss is given equation 4.4.

$$\mathscr{L}_{adv,1} = \mathbb{E}_{(C_{gt}, I_{gt})}[logD_1(C_{gt}, I_{gray})] + \mathbb{E}_{I_{gray}} log[1 - D_1(C_{pred}, I_{gray})] \tag{4.4}$$

Feature-matching loss, pushes network to generate images more similar to real images from dataset which helps network to be more stable. Feature matching loss is given in equation 4.5 below.

$$\mathscr{L}_{FM} = \mathbb{E}[\sum_{i=1}^{L} \frac{1}{N_i} ||D_1^(i)(C_{gt} - D_1^i(C_{pred}))||_1] \tag{4.5}$$

In given equation, N_i represents the number of elements in the activate layer numbered with i. $D_1^{(i)}$ is discriminator's activation in layer i and L is the discriminator's last layer.

4.3.1.2 Image Completion Network

Inpainting network uses masked image which is expressed as $I_{gt}^m = I_{gt} \bigodot (1 - M)$ to produce inpainted images over masked areas. Inpainting network also uses composite edge map which is defined as $C_{comp} = C_{gt} \bigodot (1 - M) + C_{pred} \bigodot M$. Using these inputs, network produces an output with filled missing regions shown in equation 4.6.

$$I_{pred} = G_2(I_{gt}^m, C_{comp}) \tag{4.6}$$

Overall, image completion network is trained with joint losses which are ι_1 loss, adversarial loss, perceptual loss and style loss. Similar in edge generator adversarial loss is shown in equation 4.7.

$$\mathscr{L}_{adv,2} = \mathbb{E}_{(I_{gt},C_{comp})}[logD_2(I_{gt},C_{comp})] + \mathbb{E}_{C_{comp}} log[1 - D_2(I_{pred},C_{comp})] \tag{4.7}$$

Other two losses that is used in image completion network is proposed in [65] and [66] known as L_{perc} and L_{style}. Perceptual loss, penalizes the generated images if they are not similar perceptually to activation maps of a network that is trained before. Perceptual loss is shown in equation 4.8, where ϕ_i is the activation map of pre-trained network on i'th layer.

$$\mathscr{L}_{perc} = \mathbb{E}[\sum_{i=1} \frac{1}{N_i} ||\phi_i(I_{gt} - \phi_i(I_{pred}))||_1] \tag{4.8}$$

Activation maps that are used in perceptual loss are also used in style loss which is expressed in equation 4.9.

$$\mathscr{L}_{style} = \mathbb{E}_j[||G_j^{\phi}(I_{pred}^m) - G_j^{\phi}(I_{gt}^m)||_1] \tag{4.9}$$

G_j^{ϕ} is Gram matrix that is obtained from previously shown activation matrix. Finally, overall loss function of the system is given in equation 4.10.

$$\mathscr{L}_{G_2} = \lambda_{l_1}\mathscr{L}_{l_1} + \lambda_{adv,2}\mathscr{L}_{adv,2} + \lambda_{perc}\mathscr{L}_{perc} + \lambda_s\mathscr{L}_{style} \tag{4.10}$$

4.3.2 Generative Image Inpainting with Contextual Model

One of the main problems in inpainting methods is that they create blurry and distorted parts in the missing regions. To prevent this, Yu et al. came up with Generative Image inpainting with Contextual Attention [22]. Generative image inpainting with contextual attention consists of 2 main stages which are called Coarse Network and Refinement Network. Model structure is illustrated in figure 4.11.

Coarse Network uses a method similar to the encoder and decoder structure like many methods previously described. In this part, an inpainted image is created over the

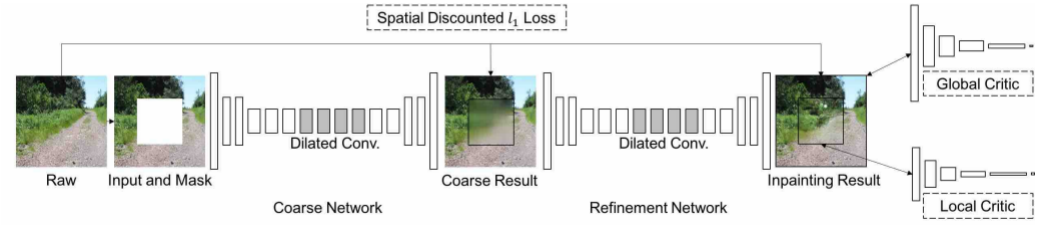


Figure 4.11 : Contextual Attention Model Structure [22]

missing part by using the GAN structure.

The main idea of this study lies in Refinement network. The image created in this section progresses on 2 different paths, then the outputs of these 2 different paths are combined to achieve the desired improved result. Refinement network is shown in more detail in figure 4.12.

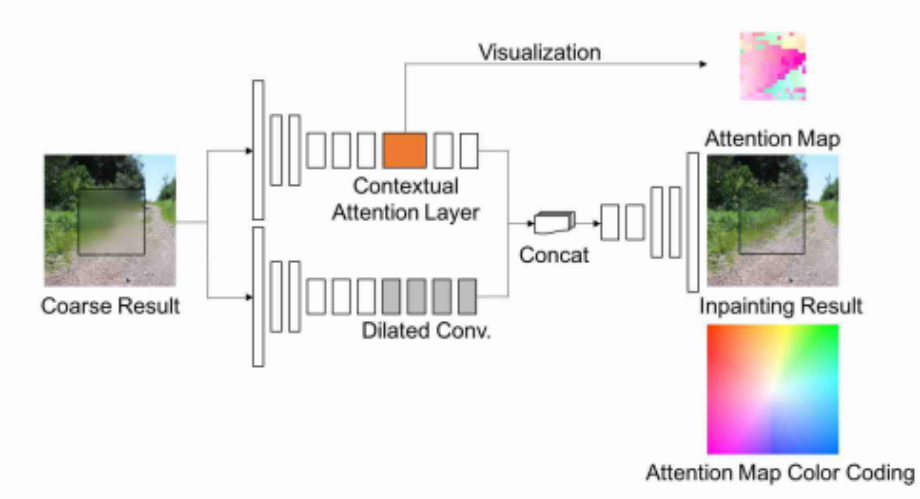


Figure 4.12 : Refinement Network [22]

In the first of these 2 paths, there is a structure consisting of dilated convolution and normal convolution layers. In the other and more important structure, there is a layer called Contextual Attention Layer that performs attention perception through the color map on the picture. In this layer, the picture is divided into patches and compared and evaluated. Algorithm in attention scoring is explained in Figure 4.13.

The Generative Contextual Attention Model has common similarities with the Edge connect model in terms of training. But there are critical differences due to the model type as expected.

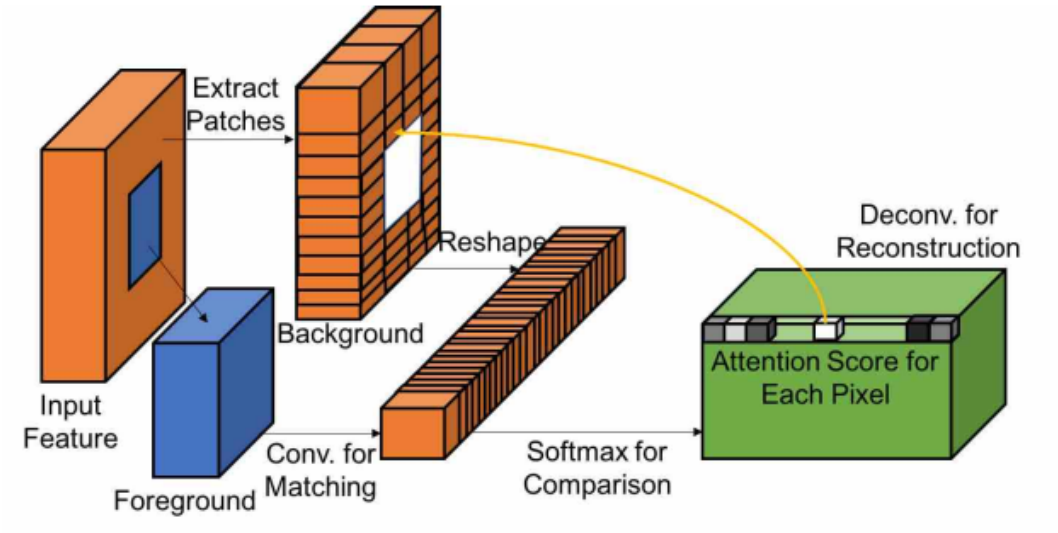


Figure 4.13 : Refinement Network [22]

4.3.2.1 Contextual Attention

Contextual attention layer can fill the masked areas by using the feature information obtained from the unchanged parts of the image. For this, firstly, it divides the unmasked region expressed as background into patches and organizes it as a convolutional filter. Then it is put into the cosine similarity operation shown in equation [] with patches in the masked region expressed as foreground.

$$s_{x,y,x',y'} = < \frac{f_{x,y}}{||f_{x,y}||}, \frac{b_{x',y'}}{||b_{x',y'}||} > \tag{4.11}$$

Where $s_{x,y,x',y'}$ is patch similarity between background and foreground. Following this step, softmax function is used to obtain a similarity score with constant value λ, as $s^*_{x,y,x',y'} = softmax_{x',y'}(\lambda_{s_{x,y,x',y'}})$. In final step, used background patches are reused as deconvolutional filters to reconstruct foregrounds. For better results, attention coherency could be increased by calculating a new attention score by summing scores in a shifted direction in desired axis.

4.3.3 Image Inpainting via Generative Multi-column CNN

In the article [23], the inpainting process is presented with a multi column CNN structure. This network acquires different image features in parallel. A new reconstruction loss function has been developed to better characterize global objects.

Also, a loss function called ID-MRF loss has been developed to increase the local details and improve the texture quality. With these loss functions and a adversarial loss, the model trained on the image dataset can fill the masked pixels in an image with local and global information. Trained model of this architecture shows realistic results without even the need for post-processing, which is often seen on similar works.

This inpainting system has a structure that can be trained from beginning to end. It first receives an image and a binary mask as input. The pixels to be masked are denoted by 1 and known pixels by 0. Then the image masked by the operation and inserted into the network. The result of the network is a completed image. Network structure is as shown in figure 4.14.

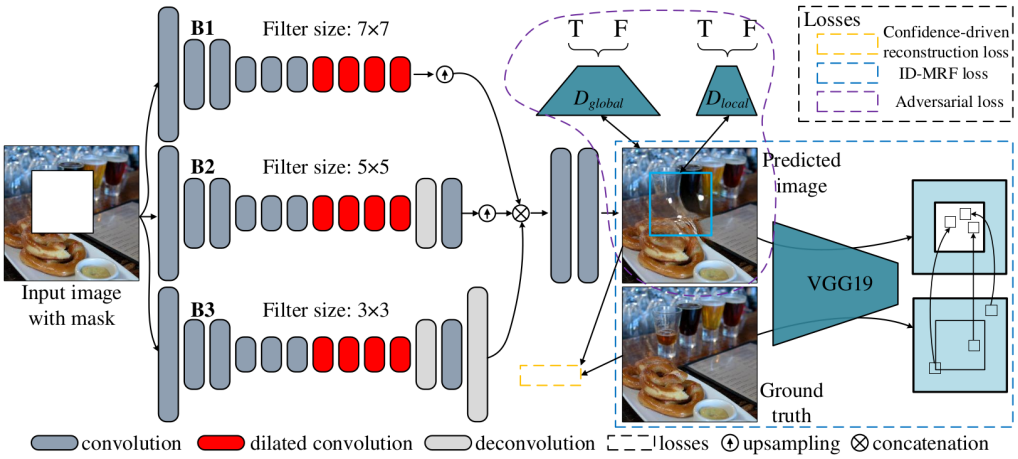


Figure 4.14 : Network structure of the GMCNN [23]

The network structure shown in figure 4.14 consists of three sub-networks. The first one is the generator network that creates the output inpainted image and is the main network which will used after the training phase. The other two sub-networks are networks that are only needed during the training phase respectively, local and global discriminator networks which used to calculate the adversarial loss and a pre-trained VGG network [67] in order to calculate the designed ID-MRF loss. Generator network consists of n parallel encoder-decoder branches (in figure 4.14, n = 3). These structures consist of convolution layers that apply filters of different kernel sizes to the input tensor. These convolution layers obtain features at different frequency levels according to the kernel size. Next, these branches which have different levels of information about input image upsampled bi-linearly to the original image size. Then these same size branches combined together and forms a feature map. Continued with two more

convolution layers, this feature map is transformed into the image. This image is the generated output image of the network and it is used to calculate the network loss when compared to the ground truth image. Even though these branches seem to be independent from each other, they are actually affected by the backpropagation phase in the training process. This framework is different from the commonly used encoder-decoder structures. In an encoder-decoder network, every layer inherits its information from its previous layer. However, in this architecture, different structures complement each other instead of just inheriting. With these different representation levels, the inpainting process is carried out.

To calculate loss during model training, a mixture of three loss functions used. ID-MRF loss, reconstruction loss, and adversarial loss.

ID-MRF stands for implicit diversified Markov random fields. This loss function is used to train with a path similarity based approach within a VGG structure fed with generated results. This method, which is used only during the training phase, is a metric used to measure the similarity between the randomly selected patches between generated pixel areas and the ground truth area pixel areas. A simple similarity measure like cosine similarity can be used to compute this process. This metric measures the similarity between the generated content and the random selected nearest-neighbor patches. But using cosine similarity measurement has some limitations. As seen in the figure 4.15.a, the results often become like blurry textures and for large masking areas this can become a serious problem.

(a) (b) (c)

Figure 4.15 : Cosine similarity vs. Relative similarity [23]

In this article, in order to eliminate this problem, a relative similarity measurement has been developed and used instead of a direct similarity such as cosine similarity. As seen in figures 4.15.b and 4.15.c, the ground truth image pathches compared with

the generated image patches. This relative similarity equation can be seen in equation 4.12.

$$RS(v,s) = exp((\frac{\mu(v,s)}{\max_{r\in\rho_v(Y^L))}\mu(v,r)+\varepsilon})/h) \tag{4.12}$$

$\mu(v,s)$ in the equation represents cosine similarity between patch v from generated area and patch s from ground truth area, the and values are constants, and represents the features on the L^{th} layer of the pre-trained VGG network. To be more precise, let generated content denoted as and is the corresponding ground truth, for any layer L, ID-RMF loss can be calculated as equation in the equation 4.13.

$$\mathscr{L}_M(L) = -log(\frac{1}{z}\sum_{s\in Y^L}\max_{v\in\widehat{Y}_g^L}\overline{RS}(v,s)) \tag{4.13}$$

The $\overline{RS}$ is the normalized relative similarity as in the equation 4.14 and Z is normalization factor. r and s represents the patches from generated image and ground truth image respectively.

$$\overline{RS}(v,s) = RS(v,s)/\sum_{r\in\rho_v(Y^L)}RS(v,r) \tag{4.14}$$

If all r patches in the $\widehat{Y}_g^L$ are close to only one spesific patch and far from other patches in ground truth, relative similarity becomes small and leads to a big loss value $\mathscr{L}_M(L)$. On the other hand if every r patch from generated image has a similar corresponding s patch in ground truth image, relative similarity becomes big and leads to a small loss value. One of the biggest contributions of this measure is to increase the similarity between the Y^L and $\widehat{Y}_g^L$ features. While minimizing the loss, generated feature distrubution and ground truth feature distrubution approach each other. As a result, in the inpainting regions blurry texture problem is overcome and variation is achieved.

$$\mathscr{L}_{mrf} = \mathscr{L}_M(conv4\_2) + \sum_{t=3}^{4}\mathscr{L}_M(conv\mathbf{t}\_2) \tag{4.15}$$

With this loss active, during the training phase, a realistic texture was obtained both locally and globally. The features layers mentioned and shown in the equation 4.15 are taken from the VGG19 pre-trained model [67].

A pixel-wise working reconstruction loss has been developed as a spacial variant reconstruction loss. In this designed confidence-driven reconstruction loss, known pixel values assigned with the value of 1 and others missing pixels confidence values gradually decays with respect to the distance from the mask border. Thus the pixels on the boundary make the most impact on the loss. As a result, a smooth transition between the masked area and the rest of the image is achieved. Increasing distance away from the boundary is obtained by convolving the mask with a gaussian filter and formulized like equation 4.16.

$$M_{\omega}^{i} = (g * \overline{M}^{i}) \odot M \tag{4.16}$$

g represents a 64×64 gaussian filter with standart deviation of 40 and $\odot$ is the Hadamard product operator. Hadamard product means element-wise multiplication of the matrices. Loss weight mask M_{ω} is obtained by repeating the equation 4.16 several times. Lastly the reconstruction loss calculated with the formula in 4.17.

$$\mathscr{L}_{c} = \|(Y - G([X,M];\theta)) \odot M_{\omega}\|_{1} \tag{4.17}$$

In equation 4.17, $G([X,M];\theta)$ represents the generated output. With the help of this loss, the results are softened from the mask boundaries towards the center.

Adversarial loss is used frequently in many tasks such as inpainting missing pixels. In this architecture, improved Wasserstein GAN [68] is used as a generator and one local and one global structure are used together as a discriminator. Reason for using two discriminators is to overcome the problem when generated areas are realistic and low loss but ill-matched with the rest of the image. The adversarial loss function is defined as in equation 4.18, where $\widehat{X} = tG([X,M];\theta) + (1-t)Y$ and $t \in [0,1]$.

$$\mathscr{L}_{adv} = -E_{X \sim \mathbb{P}_{x}}[D(G(X;\theta))] + \lambda_{gp} E_{\widehat{X} \sim \mathbb{P}_{\widehat{x}}}[(\left\| \nabla_{\widehat{X}} D(\widehat{X}) \odot M_{\omega} \right\|)_{2} - 1)^{2}] \tag{4.18}$$

Finally, the three loss functions mentioned above are combined together with different weights denoted as λ_{mrf} and λ_{adv} in equation 4.19. Regularization of the model achieved in training process.

$$\mathscr{L} = \mathscr{L}_c + \lambda_{mrf}\mathscr{L}_{mrf} + \lambda_{adv}\mathscr{L}_{adv} \tag{4.19}$$

As a comparison with GMCNN, two other CNN-based inpainting architectures selected under same conditions with the same losses and same hyperparameters. A single encoder-decoder network and the previously mentioned contextual attention coarse network [22] tested alongside with the GMCNN. Results are shown in figure 4.16. In 4.16.b, output of a single encoder-decoder network is shown. In 4.16.c, output of the contextual attention coarse network is shown. Lastly, in figures 4.16.d and 4.16.e output of the GMCNN can be seen. The former is the network with the fixed receptive fields and the latter is with varied receptive fields.

Figure 4.16 : Comparison between GMCNN and CNN-based methods [23]

Also our results with the GMCNN using a pretrained network, achieved great looking outputs with two different datasets. Places [28] dataset and CelebA [69] dataset outputs can be seen in the figures 4.17 and 4.18 respectively.

Figure 4.17 : Result of the model trained on Places2 dataset

As a limitation, like most of the generative neural networks this network also struggle with large datasets with diverse classes such as ImageNet. It is a harsh challange to inpaint different objects and places with one pre-trained model.

Figure 4.18 : Result of the model trained on CelebAHQ dataset

4.3.4 Deep Image Prior

Deep convolutional networks have become very popular for image generation and restoration problems. In general, the reason behind the good results of such networks is the ability to obtain realistic image priors from datasets containing large number of images. To explain briefly what image prior is, it is a general definiton for the information on an image that can be used for any kind of image processing tasks to enhance results, to choose the processing parameters, and resolve indeterminacies. For instance, some information about an image may known such as color distrubution, and this information or its approximation can be used as a prior to a spesific task. Image priors can be represented with math form and merged into processing steps such as filtering, deconvolution and segmentation which helps to reduce the feasible solutions.

In the article [25], the authors show that a generator network structure can get enough low-level image statistics without a long training phase on any dataset. To prove this hypothesis, designed deep image prior method with a randomly-initialized convolutional neural network was used as a prior and highly competitive results were obtained in various inverse computer vision problems such as denoising, super-resolution, and image inpainting. What makes this method important except that it can be used in a wide range of computer vision research areas, it can also forms a new branch between learning-based networks and learning-free networks with fixed explicit priors like self-similarity etc. To be declared in a single sentence, in deep image prior,

authors tries to bridge the gap between two popular methods by constructing a new explicit prior using convolution neural network.

To perform the task of image restoration like inpainting, learned-prior and explicit-prior are the two common methods utilized by researchers. To explain with an example, learned-prior is a straight forward approach to train a deep convolution network to learn about the world through the dataset. On the other hand, explicit-prior or hand-crafted prior method, is embedding constrains and distinctly teaching what types of images are natural. However in real word it is extremely difficult to express constraints mathematically. As a result, most of the explicit-prior methods works poorly compared to a state-of-the-art pre-trained neural network, yet deep image prior method offers surprisingly good results which can compete with learned-prior methods.

Deep convolutional networks are architectures that on of the best for solving inverse image problems. Architectures like generative adversarial networks and variational auto-encoders achieve state-of-the-art solutions on this tasks. However, it is not entirely correct to assume that the reason for the amazing results of these models trained on large datasets is that they can capture realistic image priors while learning over the data. Because only a good learning curve on a selected datasets does not prove that a network performs good on an image from a different dataset. A good network requires generalization which means that the structure of the network should resonate with the structure of the given input data [25]. For example, an image classification network which successfully generalized for real data can overfit when a random labels are presented [70].

To put it more clearly, in this article the authors shows that learning process is not a requirement and it is possible to obtain quite good image priors with a convolutional generative structure that does not need training. In order to prove this claim, an untrained, randomly-initialized convolutional generator neural network is fitted on a given corrupted image. Generator network initialized with a random distrubution conditioned to the input corrupted image. The reconstruction task is expressed as a conditional image generation problem. Randomly initialized network weights

optimized without needing any other data than input image and network itself. Only prior information is in the structure of the network itself. [25] as authors states.

$$x = f_\theta(z), \quad x \in R^{3\times H\times W}, \quad z \in R^{C'\times H'\times W'} \tag{4.20}$$

In equation 4.20, x is corrupted input image, z is a code tensor and represents the network parameters which randomly initialized in the beginning and will map code tensor z to the image x while resolving inverse problem through iterations. The network is switching between filtering operations such as non-linear activation, convolution, and upsampling. For most of the experiments, a hourglass type U-Net architecture with two million parameters used [25], [71].

In image restoration problems the goal is to recover original image x, when having a corrupted image x_0. To solve such tasks, problem is often formulated as an optimization like in equation 4.21.

$$x^* = \min_x E(x;x_0) + R(x) \tag{4.21}$$

In equation 4.21, $E(x;x_0)$ is a data term dependent to the task and $R(x)$ is an image prior. For a wide range of problems, such as super-resolution, denoising, and inpainting, the data term is typically simple to design, while designing the image prior is a challenging. For inpainting problems, the corresponding data term is defined as in the equation 4.22.

$$E(x;x_0) = \|(x-x_0)\odot m\|^2, \quad m \in \{0,1\}^{H\times W} \tag{4.22}$$

The symbol $\odot$ is the Hadamard product operator which was explained in former sections and m is the binary mask applied onto ground truth image x. The goal is to reconstruct the ground truth image x from the given image x_0 with missing pixel values.

As mentioned, choosing a regularizer $R(x)$ that can catch general priors in image x is a difficult and very extensive research subject within itself. To give an example, if total variation (TV) [72] is selected as a regularizer, the results are often have uniform

regions. In this study, as seen in the equation 4.23, implicit prior which gathered from neural network was used as a regularizer.

$$\theta_* = \arg\min_{\theta} E(f_\theta(z); x_0), \quad x^* = f_{\theta^*}(z) \tag{4.23}$$

θ^* is the minimizer achieved with the help of the gradient descent optimizer from random parameter values. Result of the inpainting process denoted as x^* is given in equation 4.23. It is also possible to apply minimazation on the code tensor z. However here z is a fixed three dimensional tensor containing 32 feature maps of uniform noise with the same spatial size as x.

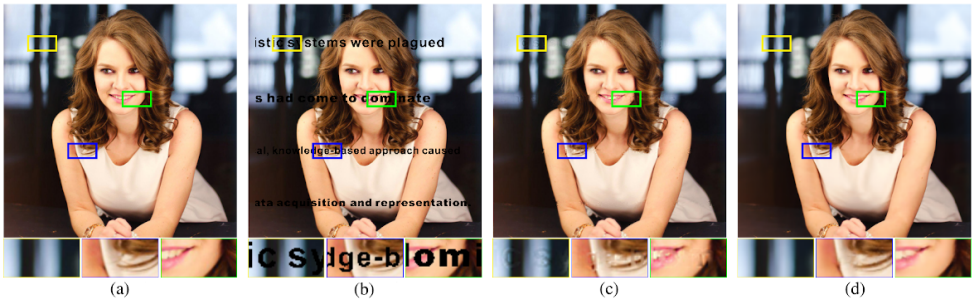


Figure 4.19 : Comparison between Shepard Network and Deep Image Prior [24], [25]

In the figure 4.19 deep image prior method compared with Shepard Neural Networks [24] which is particularly designed for inpainting applications. Using a black text as a mask, this method demonstrates astonishing results. While in the output of the Shepard network, remaining of the mask can be seen, in the output of the deep image prior method there are almost no artifacts.

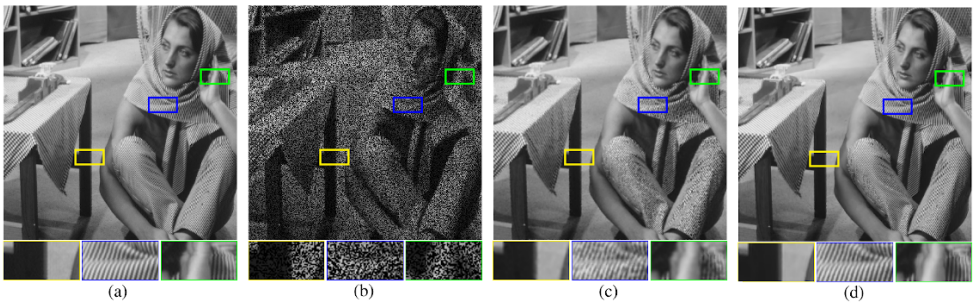


Figure 4.20 : Comparison between Convolutional Dictionary Learning and Deep Image Prior [26], [25]

In another experiment, deep image prior is compared with Convolutional Dictionary Learning method [26] using an image with masked half of its pixels by applying a

Bernoulli distrubution [73] with random noise. Comparison results are given in figure 4.20.

Apart from the inpainting study of small pixel regions, a large region inpainting process was also tested and deep image prior method has shown great success again. Although deep image prior method works great for variety of images, as a limitation it works poorly while inpainting large masked areas of the images which have highly semantic information such as face images due to being a learning-free method.

(a) (b) (c)

Figure 4.21 : Comparison between Global-Local GAN and Deep Image Prior [27], [25]

Result of the inpainting of a large masked region with deep image prior method is compared with a learning-based generative adversarial network model [27]. As shown in the figure 4.21, the pretrained GAN structure performs the inpainting process with the information learned from the dataset it trained on. On the other hand deep image prior fills the missing region with texture information learned from known regions.

As mentioned, deep image prior is a method which can be applied onto any network architecture, the results obtained are highly dependent on the network structure used. In figure 4.22, deep image prior method applied on different neural networks. From the results, it can be concluded that deeper networks have positive effect on the inpainting solution. However, having skip-connections which can significantly improve image recognition tasks shows extremely harmful behavior in inpainting process as seen in figure 4.22.e and 4.22.f where skip-connections added to the ResNet [74] and U-Net [13] architectures.

In figures 4.22.b, 4.22.c and 4.22.d an encoder-decoder architectures used with depth size of 6, 4, and 2 respectively. Figure 4.22.e is the output of the ResNet with skip layers and with depth of 8. Figure 4.22.f is the output of the U-Net structure with

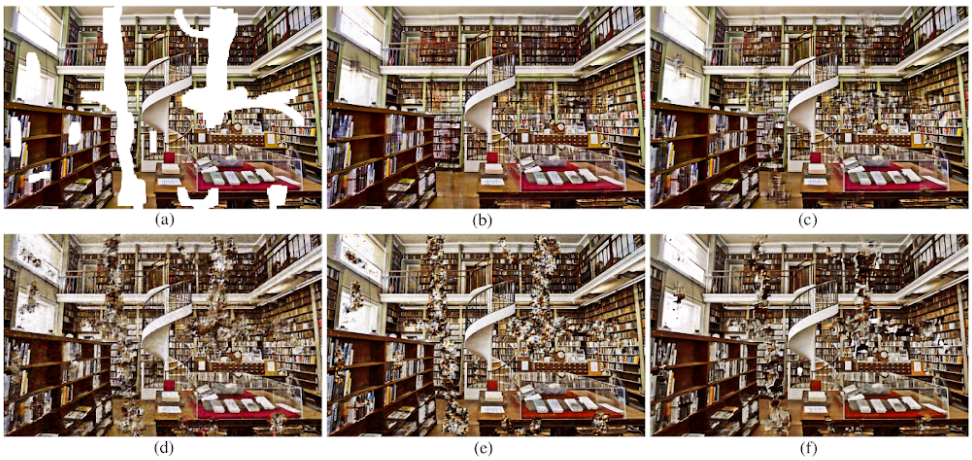


Figure 4.22 : Deep Image Prior results on different networks [25]

skip layers and depth of 5. Finally, we tested the Deep Image Prior method using an encoder-decoder network and achieved quite enough results. Figure 4.23 shows the output of the image 512x512 image with mask we generated after 1000 iteration.

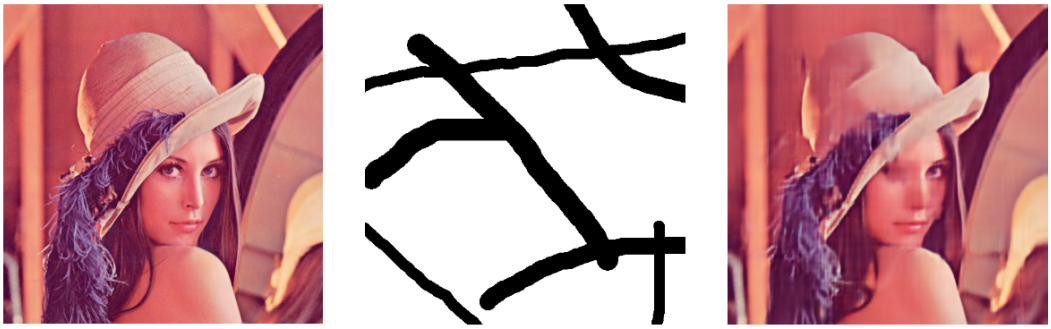


Figure 4.23 : Example result of the Deep Image Prior method

5. EVALUATION, RESULTS AND OUR CONTRIBUTION

The most important part of our study is to implement these models we have examined and to make comparisons of these models after the implementation. While implementing these models, we basically used the python software language, but we also used many different libraries and frameworks. The main 2 libraries we used are pytorch and tensorflow. However, we also used libraries like matplotlib, openCV, and numpy. The structures and codes we use can be accessed on the internet in an open source manner.

Implementing Deep Learning models is always a challenge. For this purpose, we used frameworks that allow us to use graphic cards for training and test phases. Similar to all studies on inpainting we used places2 dataset [28] for training and test. However due to high computational power and relatively weak GPUs we had, we needed to decrase the size of the dataset we used. An example batch from places2 dataset is shown in figure 5.1

Figure 5.1 : Places2 Dataset [28]

Example visual results and mathematical results over a validation dataset for different types of masks are represented in the following sections.

5.1 Framework

One of the key points in our study was to create a model that allows many different models to work together. In order to achieve this, we had to create a framework that is easily accessible by different models and consists of parts that generate standard types of feedback. In this way, we would have obtained a modular structure where it is easy to make changes and incorporate new ideas. General structure of our framework is shown in Figure 5.2.

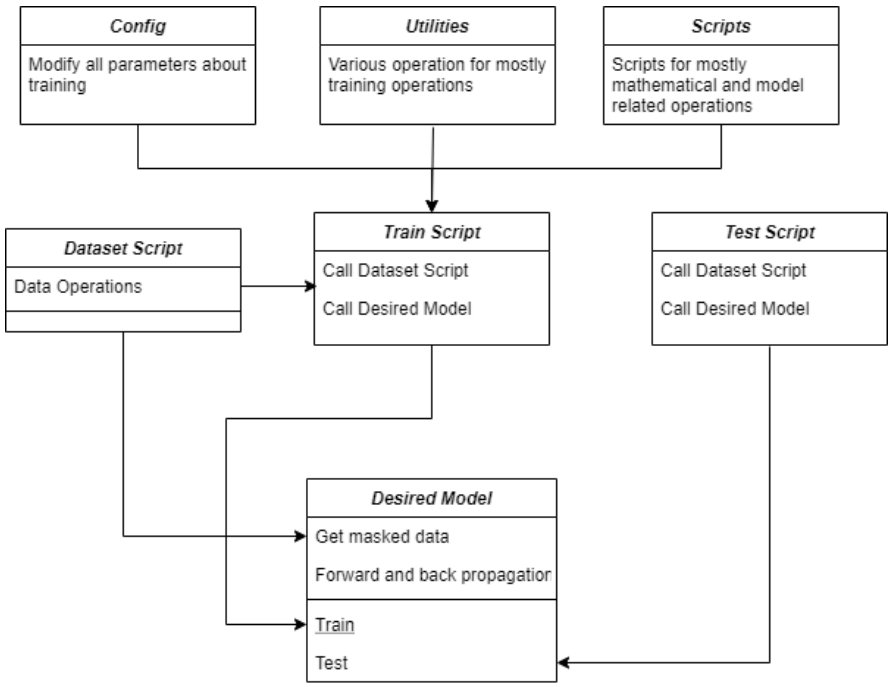
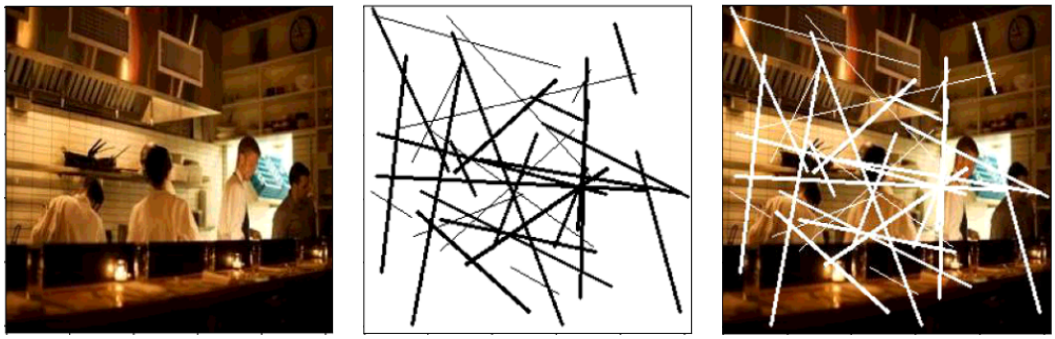


Figure 5.2 : Framework

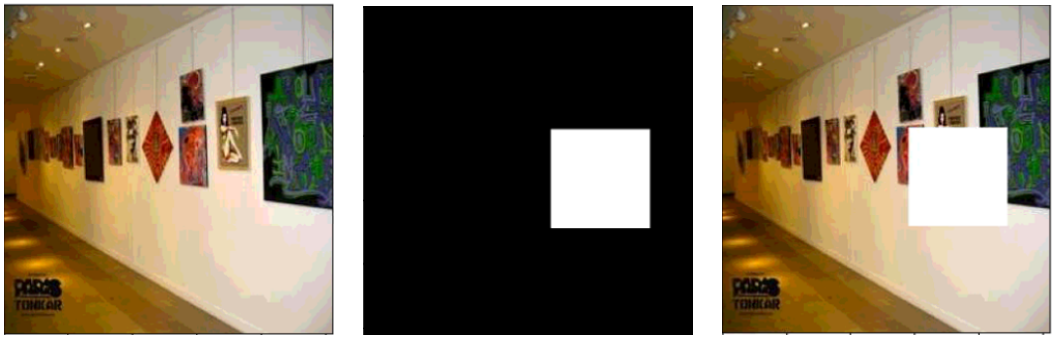
In order to provide these, we created scripts that will serve as small building blocks. The most important of these building blocks are the dataset and utilities scripts, which are also actively used by models and can include important features of the models.
Masked images or transformed images that models use as input are also generated by the dataset script. We created 2 different types of masking. The first masking type we have created is the masking type that creates scattered lines on the picture which we call line masking. With this masking type, we are able to simulate smaller imperfections rather than large missing regions in the picture. Line masking example is shown in figure 5.3.



(a) Image (b) Mask (c) Masked Image

Figure 5.3 : Line Masking Example

The other masking function we use is a function that masks a region of the given ratio value, starting from a random point on the picture. We preferred to use this masking type more to show that generative models are more successful. Our second masking type is presented in figure 5.4.



(a) Image (b) Mask (c) Masked Image

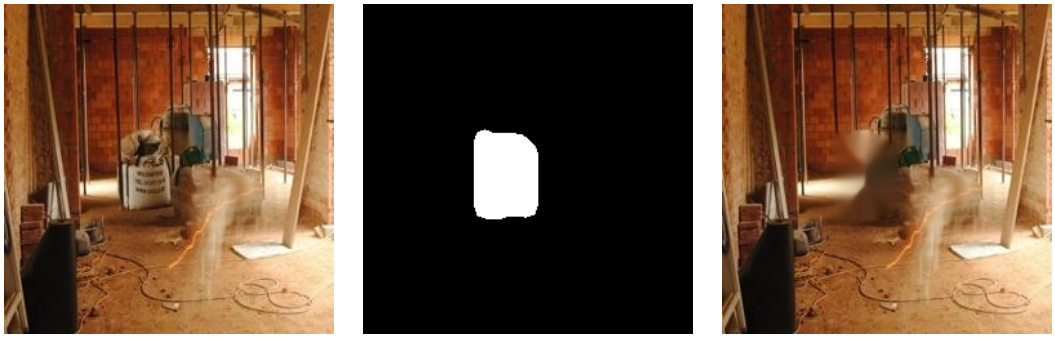
Figure 5.4 : Percentage Masking Example

Image inpainting, unfortunately, is not a method whose success can be fully controlled with a mathematical criterion. However, PSNR and SSIM values will be used throughout the project as a criterion in addition to achieving personal realistic results. Also for out comparisons mathematical methods are used throughout our project. An example of image inpainting with one of our mathematical method which uses navier-strokes method is shown in Figure 5.5.

5.2 Our Contribution

In addition to the examined methods, two new methods are proposed and implemented in our project called, Unified Contextual-Edge Model and FPN GAN Model.

(a) Input Image (b) Mask (c) Navier-Strokes Output



(d) Input Image (e) Mask (f) Navier-Strokes Output

Figure 5.5 : Mathematical Model Example

5.2.1 Unified Contextual-Edge Model

During our literature review we studied a model by Yu et al. which is called Free-Form Image Inpainting with Gated Convolution [29]. In this model, there are 3 main stages as shown in figure 5.6.

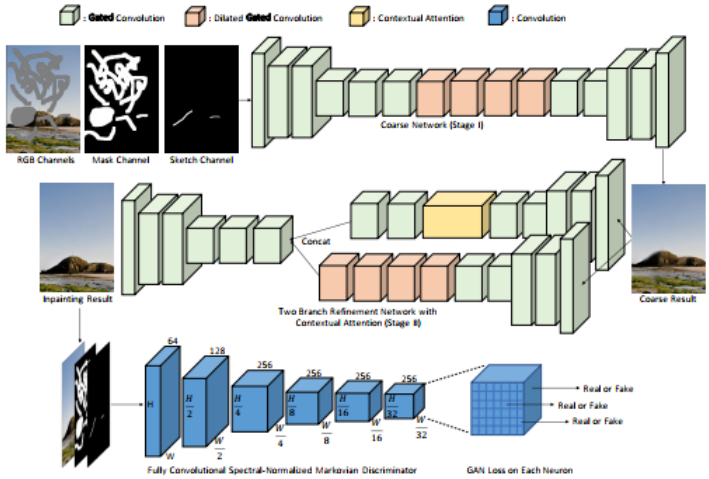


Figure 5.6 : Free-form Image Inpainting Example [29]

Based on this, we wanted to try EdgeConnect and Generative Contextual Attention models in a cascaded network. Hence, there will be 3 steps in the new model. Which consists of following steps. figure 5.7 shows the steps of this model.

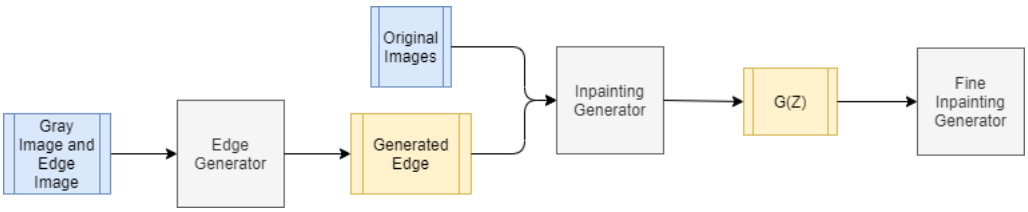


Figure 5.7 : Unified Model Structure

• Edge information prediction

• Inpainting operation with edge and RGB image as input

• Refinement network

We were able to implement this idea. However, size of the model is nearly doubled and training time is significantly increased, but it was also possible to use independently trained parameters for our models. Figure 5.8 shows the obtained result with unified model.

(a) Input Image

(b) Inpainted Image

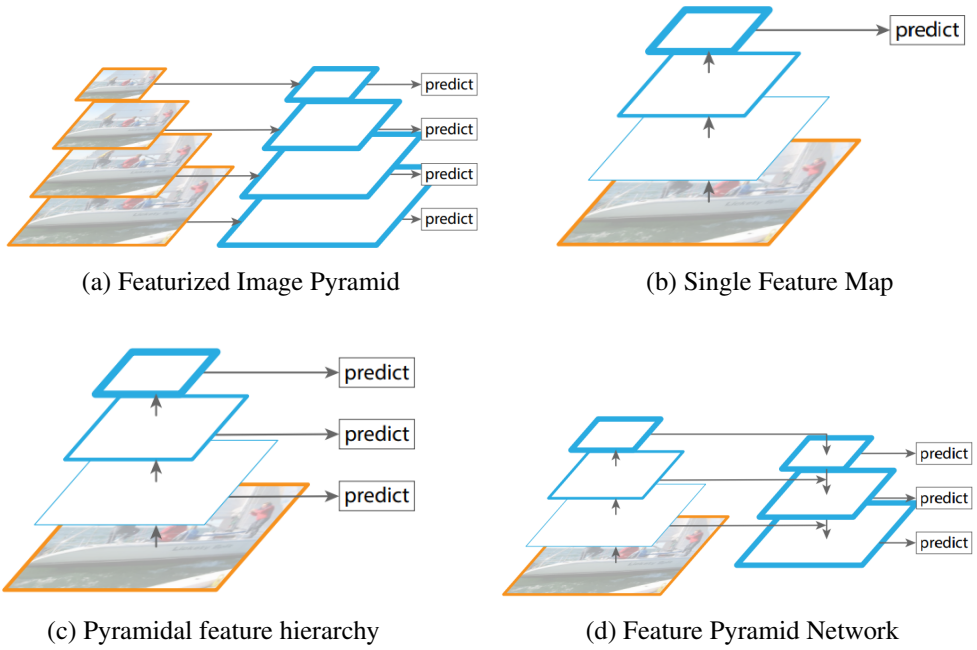
(c) Input Image

(d) Inpainted Image

Figure 5.8 : Unified Model Inpainting Example

5.2.2 Feature Pyramid GAN Model

Feature Pyramid Networks is proposed in paper Feature Pyramid Networks for Object Detection [30]. FPN is commonly used in object detection problem. The basic idea behind FPN is to combine the feature information of different sizes of the same image to obtain a more inferential output. During the execution of the model, the input image is inserted into downsampling layers. The results obtained from each stage are re-included in the upsampling process in a horizontal way. Different versions of feature structures in object detection are given in 5.9



(a) Featurized Image Pyramid

(b) Single Feature Map

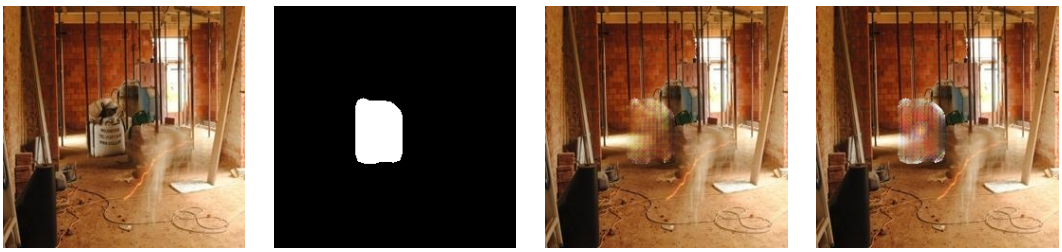
(c) Pyramidal feature hierarchy

(d) Feature Pyramid Network

Figure 5.9 : Feature Pyramid Structures [30]

In the models we use throughout this project uses single feature map structure given in figure 5.9. We wanted to try out this proposed feature pyramid network for inpainting. For this purpose, a GAN model for inpainting is created which is similar to the one that is used in EdgeConnect [21] and Generative Contextual [22] models. However, generative networks of this GAN is transformed into a feature pyramid network. For comparison purposes, we also used the same network GAN network without FPN which we call Vanilla GAN. Results of these models are given in figure 5.10

(a) Input Image (b) Mask (c) Vanilla GAN Output (d) FPN GAN Output



(e) Input Image (f) Mask (g) Vanilla GAN Output (h) FPN GGAN Output

Figure 5.10 : FPN and Vanilla GAN Inpainting Example

5.3 Test Results

In this section, visual and analytical comparisons of implemented models are shown.

(a) MaskedImage

(b) Fast Marching Method

(c) Deep Image Prior

(d) GMCNN

(e) Contextual Attention

(f) EdgeConnect

(g) Unified

Figure 5.11 : Example Inpainting Comparison 1

(a) MaskedImage

(b) Fast Marching Method

(c) Deep Image Prior

(d) GMCNN

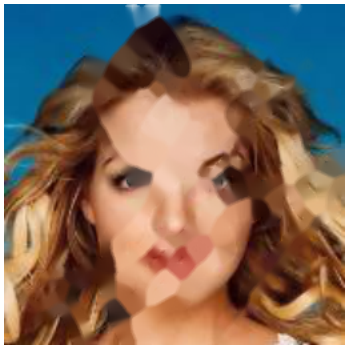
(e) Contextual Attention

(f) EdgeConnect

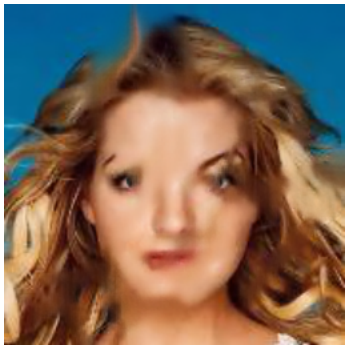
(g) Unified

Figure 5.12 : Example Inpainting Comparison 2

(a) MaskedImage

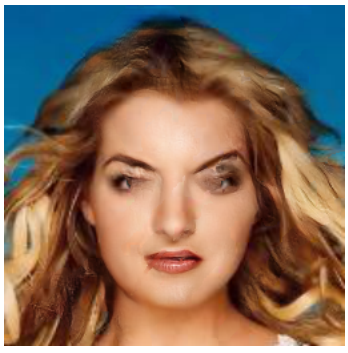


(b) Fast Marching Method

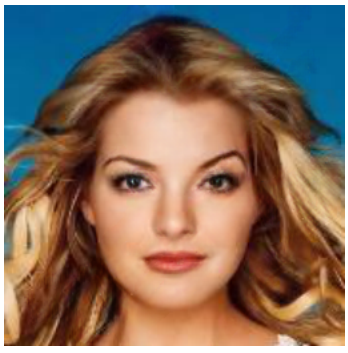


(c) Deep Image Prior

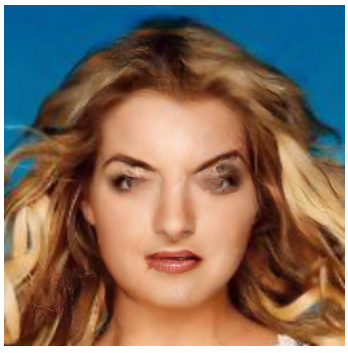
(d) GMCNN



(e) Contextual Attention



(f) EdgeConnect



(g) Unified

Figure 5.13 : Example Inpainting Comparison 3

(a) Input Image (b) Mask (c) Fast Marching Method (d) Deep Image Prior

(e) GMCNN (f) Contextual Attention (g) EdgeConnect (h) Unified

Figure 5.14 : Example Inpainting Comparison 4

(a) Input Image (b) Mask (c) Fast Marching Method (d) Deep Image Prior

(e) GMCNN (f) Contextual Attention (g) EdgeConnect (h) Unified

Figure 5.15 : Example Inpainting Comparison 5

(a) Input Image (b) Mask (c) Fast Marching Method (d) Deep Image Prior

(e) GMCNN (f) Contextual Attention (g) EdgeConnect (h) Unified

Figure 5.16 : Example Inpainting Comparison 6

(a) Input Image (b) Mask (c) Fast Marching Method (d) Deep Image Prior

(e) GMCNN (f) Contextual Attention (g) EdgeConnect (h) Unified

Figure 5.17 : Example Inpainting Comparison 7

(a) Input Image (b) Mask (c) Fast Marching Method (d) Deep Image Prior

(e) GMCNN (f) Contextual Attention (g) EdgeConnect (h) Unified

Figure 5.18 : Example Inpainting Comparison 8

Mentioned models are tested with Place2 validation data over 3 different masking types. One of the masks is a mask with an arbitrary line goes over the image. The other 2 masks are rectangles that cover a certain percentage over the image which are 10% and 20%. Analytical comparisons according to PSNR and SSIM values are given in 5.1 and 5.2 while NS, FM, EC, CA, GC, DP, U represents Navier-Strokes, Fast-Marching, EdgeConnect, Contextual Attention, GMCNN, Deep Prior and Unified models respectively.

PSNR	NS	FM	EC	CA	GC	DP	U
Lines	31.087	30.620	29.049	31.956	32.143	29.181	28.981
10%	25.108	25.284	26.148	24.850	24.844	20.460	24.743
20%	21.379	21.504	22.508	21.162	20.395	17.012	20.504

Table 5.1 : PSNR Results

SSIM	NS	FM	EC	CA	GC	DP	U
Lines	0.949	0.940	0.916	0.945	0.942	0.909	0.907
10%	0.902	0.902	0.911	0.910	0.906	0.761	0.908
20%	0.813	0.813	0.824	0.826	0.818	0.710	0.824

Table 5.2 : SSIM Results

FID	NS	FM	EC	CA	GC	DP	U
Lines	22.820	28.140	19.311	16.664	14.851	27.956	25.900
10%	55.689	62.013	26.202	27.093	30.084	65.562	28.025
20%	89.081	96.290	40.449	42.656	53.499	81.721	43.478

Table 5.3 : FID Results

Furthermore, PSNR, SSIM and FID values of our proposed method with 10% rectangle masks, Feature Pyramid GAN and Vanilla GAN for comparison are given in table 5.4.

	Vanilla	FPN
PSNR	25.277	23.331
SSIM	0.905	0.900
FID	40.534	59.544

Table 5.4 : SSIM, PSNR and FID results for proposed methods

6. CONCLUSION

Various models trying to achieve the same result by following different methods were examined, applied and compared throughout the the project. The obtained results are presented visually and mathematically in the project. On the other hand, it has been tried to add new methods to image inpainting method.

6.1 Possible Applications of this Project

Image related problems are getting more important each day in our society. Image inpainting is also one of these important image related areas. Image inpainting operation might be needed for various reasons, hence it's possible applications may vary.

- Correction of photos: As it is mentioned before, distortions in images may happen in various ways. One of the important usage of the image inpainting is correcting the distorted parts of target image.
- Removing unwanted parts from photos: Image inpainting operation might be needed to remove an unwanted region rather than a distorted area. Many people want to manipulate their photos for a lot of reason such as sharing them online. A lot of software are used for this purpose. It is also possible to use deep learning for the same reason.

6.2 Realistic Constraints

Training deep learning models are always hard to deal with. Even though it is easy to reach high computational power and huge datasets in recent years, it still take a lot of time to train these deep learning models. One of the main challenges of the image inpainting is to produce visually realistic images. It is possible to get better results mathematically, while output images are not visually realistic.

6.2.1 Social, environmental and economic impact

As it is mentioned before, for purposes like image inpainting, commercial softwares are commonly used and professionals might work in this area. Image inpainting with deep learning could be a powerful alternative for these software. On the other hand, getting better results with image inpainting could effect social media usage.

6.2.2 Cost analysis

Cost of this project is only restricted with training equipments for the networks an engineers' wages.

6.2.3 Standards

IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology is used in this thesis work.

6.2.4 Health and Safety Concerns

There are no health and safety concerns in this project.

6.3 Future Work and Recommendations

Image inpainting is an area that is still actively being developed. As computational power continues to improve and deep learning methods continue to be developed, it is natural to expect better results from image inpainting. In particular to our study, with the development of other areas where deep learning is used, better results can be obtained with multi-disciplinary studies. Our proposed methods could be improved by using bigger datasets or deeper network models which can be trained on better GPUs. Also, it is possible to add a fine tuning layer or a mathematical operation that smooths the output of our models.

REFERENCES

[1] T. Zhou, B. Johnson, and R. Li, "Patch-based texture synthesis for image inpainting," 2016.

[2] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: Combining inconsistent images using patch-based synthesis," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–10, 2012.

[3] H. Li, W. Luo, and J. Huang, "Localization of diffusion-based inpainting in digital images," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 3050–3064, 2017.

[4] D. Li, "Deep neural network approach for single channel speech enhancement processing," 2016.

[5] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, 03 2017.

[6] I. S. Mohamed, "Detection and tracking of pallets using a laser rangefinder and machine learning techniques," Ph.D. dissertation, 09 2017.

[7] A. Asadi and R. Safabakhsh, *The Encoder-Decoder Framework and Its Applications*. Cham: Springer International Publishing, 2020, pp. 133–167. [Online]. Available: https://doi.org/10.1007/978-3-030-31756-0_5

[8] H. Nugroho, "Fully convolutional variational autoencoder for feature extraction of fire detection system," *Jurnal Ilmu Komputer dan Informasi*, vol. 13, 03 2020.

[9] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," 10 2017, pp. 373–382.

[10] J. E. Sorribas, "About generative and discriminative models," Jul 2020. [Online]. Available: https://medium.com/@jordi299/about-generative-and-discriminative-models-d8958b67ad32

[11] P. Salehi, A. Chalechale, and M. Taghizadeh, "Generative adversarial networks (gans): An overview of theoretical model, evaluation metrics, and recent developments," 2020.

[12] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," 2015.

[14] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: Image inpainting via deep feature rearrangement," 2018.

[15] H. Liu, B. Jiang, Y. Xiao, and C. Yang, "Coherent semantic attention for image inpainting," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019.

[16] L. Liao, R. Hu, J. Xiao, and Z. Wang, "Artist-net: Decorating the inferred content with unified style for image inpainting," *IEEE Access*, pp. 1–1, 2019.

[17] Y. Zeng, J. Fu, H. Chao, and B. Guo, "Learning pyramid-context encoder network for high-quality image inpainting," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019.

[18] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[19] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015.

[20] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," 2016.

[21] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, "EdgeConnect: Generative image inpainting with adversarial edge learning," 2019.

[22] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," 2018.

[23] Y. Wang, X. Tao, X. Qi, X. Shen, and J. Jia, "Image inpainting via generative multi-column convolutional neural networks," 2018.

[24] J. S. J. Ren, L. Xu, Q. Yan, and W. Sun, "Shepard convolutional neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA: MIT Press, 2015, pp. 901–909.

[25] V. Lempitsky, A. Vedaldi, and D. Ulyanov, "Deep image prior," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018.

[26] V. Papyan, Y. Romano, J. Sulam, and M. Elad, "Convolutional dictionary learning via local processing," 2017.

[27] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–14, 2017.

[28] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, 2018.

[29] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, "Free-form image inpainting with gated convolution," 2018.

[30] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016.

[31] N. Zhang, H. Ji, L. Liu, and G. Wang, "Exemplar-based image inpainting using angle-aware patch matching," *EURASIP Journal on Image and Video Processing*, vol. 2019, no. 1, p. 70, Jul 2019. [Online]. Available: https://doi.org/10.1186/s13640-019-0471-2

[32] A. C. Kokaram, R. D. Morris, W. J. Fitzgerald, and P. J. W. Rayner, "Interpolation of missing data in image sequences," *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1509–1519, 1995.

[33] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1033–1038 vol.2.

[34] R. Takei and W. Au, "Image inpainting with the navier-stokes equations," 2005.

[35] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of Graphics Tools*, vol. 9, pp. 23 – 34, 2004.

[36] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari, "Image inpainting: A review," *Neural Processing Letters*, vol. 51, no. 2, p. 2007–2028, Dec 2019. [Online]. Available: http://dx.doi.org/10.1007/s11063-019-10163-0

[37] "PatchMatch: A randomized correspondence algorithm for structural image editing," https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/, accessed: 2021-1-9.

[38] L. Dou, Z. Qian, C. Qin, G. Feng, and X. Zhang, "Anti-forensics of diffusion-based image inpainting," *J. Electron. Imaging*, vol. 29, no. 04, 2020.

[39] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[40] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 5, pp. 1060–1089, 2013.

[41] A. M. Khattak, H. Ullah, H. A. Khalid, A. Habib, M. Z. Asghar, and F. M. Kundi, "Stock market trend prediction using supervised learning," in *Proceedings of the Tenth International Symposium on Information and Communication Technology - SoICT 2019*. New York, New York, USA: ACM Press, 2019.

[42] D. Cintia Ganesha Putri, J.-S. Leu, and P. Seda, "Design of an unsupervised machine learning-based movie recommender system," *Symmetry (Basel)*, vol. 12, no. 2, p. 185, 2020.

[43] I. Jolliffe, "Principal component analysis," in *International Encyclopedia of Statistical Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1094–1096.

[44] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[45] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," 2018.

[46] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

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

[48] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE Inst. Electr. Electron. Eng.*, vol. 86, no. 11, pp. 2278–2324, 1998.

[49] A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," 02 2020, pp. 1–5.

[50] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016.

[51] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[52] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 665–674. [Online]. Available: https://doi.org/10.1145/3097983.3098052

[53] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Anomaly detection using autoencoders in high performance computing systems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, p. 9428–9433, Jul 2019. [Online]. Available: http://dx.doi.org/10.1609/aaai.v33i01.33019428

[54] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1096–1103. [Online]. Available: https://doi.org/10.1145/1390156.1390294

[55] G. Montufar, "Restricted boltzmann machines: Introduction and review," 2018.

[56] Y. Huang, A. Panahi, H. Krim, Y. Yu, and S. L. Smith, "Deep adversarial belief networks," 2019.

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

[58] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017.

[59] O. Sidorov and J. Y. Hardeberg, "Deep hyperspectral prior: Single-Image denoising, inpainting, Super-Resolution," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019.

[60] X. Cai and B. Song, "Semantic object removal with convolutional neural network feature-based inpainting approach," *Multimed. Syst.*, vol. 24, no. 5, pp. 597–609, 2018.

[61] T. Nakamura, A. Zhu, K. Yanai, and S. Uchida, "Scene text eraser," 2017.

[62] Y. Jo and J. Park, "Sc-fegan: Face editing generative adversarial network with user's sketch and color," 2019.

[63] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," 2016.

[64] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," 2016.

[65] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[66] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016. [Online]. Available: http://arxiv.org/abs/1603.08155

[67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[68] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein GANs," 2017.

[69] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[70] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," 2016.

[71] Y. Liu, D. Nguyen, N. Deligiannis, W. Ding, and A. Munteanu, "Hourglass-shapenetwork based semantic segmentation for high resolution aerial imagery," *Remote Sensing*, vol. 9, p. 522, 05 2017.

[72] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," 2014.

[73] *Bernoulli Distribution*. New York, NY: Springer New York, 2008, pp. 36–37. [Online]. Available: https://doi.org/10.1007/978-0-387-32833-1_24

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

CURRICULUM VITAE

Name Surname: Enes Demirağ

Place and Date of Birth: Istanbul, 28/05/1998

E-Mail: demirag16@itu.edu.tr

EDUCATION:

- **B.Eng.:** 2016 - 2021, Istanbul Technical University, Faculty of Electrical and Electronics Engineering, Electronics and Communication Engineering
- **Erasmus Program:** Jan. - June 2019, Liverpool John Moores University, UK

PROFESSIONAL EXPERIENCE AND REWARDS:

- Baykar Defence *Aug 2020 - Present*
- Ravinspect Tech *July 2018 - Sept. 2018*
- DESI Security & Lock Systems *July 2017 - Aug. 2017*
- ITU Department of Information Technology *Apr. 2017 - July 2017*

Name Surname: Halil İbrahim Bengü

Place and Date of Birth: Istanbul, 27/04/1998

E-Mail: bengu16@itu.edu.tr

EDUCATION:

- **B.Eng.:** 2016 - 2021, Istanbul Technical University, Faculty of Electrical and Electronics Engineering, Electronics and Communication Engineering
- **Erasmus Program:** Jan. - June 2019, Linköping University, SWE

PROFESSIONAL EXPERIENCE AND REWARDS:

- Pavelsis *Nov. 2020 - Present*
- Aselsan *July 2020 - Sep. 2020*
- Kaizen Telecommunication *July 2018 - Sep. 2018*