Generative Adversarial Network Based Visual Saliency Prediction with Cascaded Hierarchical Atrous Spatial Pyramid Pooling

Daniel Dufera Kenea

A Thesis Submitted to the Department of Computer Science and Engineering

School Of Electrical Engineering and Computing

Office of Graduate Studies

Adama Science and Technology University

September, 2022

Adama, Ethiopia

Generative Adversarial Network Based Visual Saliency Prediction with Cascaded Hierarchical Atrous Spatial Pyramid Pooling

Daniel Dufera Kenea

Advisor: Prof. Yun Koo Chung

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical and Computing

Office of Graduate Studies

Adama Science and Technology University

September, 2022

Adama, Ethiopia

DECLARATION

I declare that this thesis is entitled "**Generative Adversarial Network based Visual Saliency Prediction with Cascaded Hierarchical Atrous Spatial Pyramid Pooling**" is my own work and has not been submitted to any university for similar purposes. The references used in this thesis are duly recognized by proper citations.

_____	_____	_____
Name of student	Signature	Date

RECOMMENDATION

I, the major advisor/supervisor of this research, hereby certify that I have closely advised/supervised the student while developing this thesis and read the thesis entitled "**Generative Adversarial Network based Visual Saliency Prediction with Cascaded Hierarchical Atrous Spatial Pyramid Pooling**" prepared by Daniel Dufera. Therefore, I recommend the submission of the thesis to the department for further review and evaluation.

Major Advisor/Supervisor	Signature	Date

APPROVAL SHEET

We, the undersigned, members of the Board of Reviewers of the thesis open defense by Daniel Dufera have read and evaluated the thesis entitled "**Generative Adversarial Network based Visual Saliency Prediction with Cascaded Hierarchical Atrous Spatial Pyramid Pooling**, therefore, to certify that the thesis is accepted.

Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon the submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and

Department Head	Signature	
School Dean	Signature	
Office of Postgraduate Studies, Dean	Signature	

ACKNOWLEDGEMENT

First, my deepest thanks go to the Almighty God for his keeping with health within all my way. And then, I would like to thank my advisor **Prof. Yun Koo Chung** for providing me the great opportunity to work on this thesis and for providing continuous guidance throughout my study. I would also like to thank him for guiding me in the right direction, for timely feedback, and for his valuable suggestions to fulfill the thesis requirements.

And also, I would like to thank **Dr. Worku Jifara** for his willingness to help us whenever we want by giving us his time. I would like to thank **Dr. Mesfin Abebe** for his support and advice in reading my thesis. Next, I would like to thank Mr. Mifta Juniedi (M.Sc.) for his support.

Next, I would like to thank my parents for their countless support in my journey, without them I wouldn't be here today. And also, I would like to express my gratitude to all my friends and classmates for their unconditional support throughout my journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AA	Attention Area
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASPP	Atrous Spatial Pyramid Pooling
AUC	Area under Curve
BN	Batch Normalization
CHASPP	Cascaded Hierarchical Spatial Pyramid Pooling
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
DC	Dilated Convolution
DL	Deep Learning
EL	Edge Loss
GAN	Generative Adversarial Network
GPU	Graphical Processing Unit
HVS	Human Visual Saliency
I2I	Image to Image Translation
KLD	Kullback Leibler Divergence
L1	Manhattan Distance
L2	Euclidean Distance
LSGAN	Least Square Generative Adversarial Network
LSRGAN	Least Square Relativistic Generative Adversarial Network
LWGAN	Wasserstein Generative Adversarial Network Loss
MAE	Mean Absolute Error
ML	Machine Learning

MSE	Mean Square Error
NSS	Normalized Scanpath Saliency
PDF	Probability Density Function
PL	Perceptual Loss
RAM	Random Access Memory
ReLu	Rectified Linear Unit
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
SD	Shannon Divergence
SIM	Similarity or Histogram Intersection
VGG	Visual Geometry Group
VS	Visual Saliency
VSP	Visual Saliency Prediction
WGAN	Wasserstein Generative Adversarial Network

LIST OF NOTATION AND SYMBOLS

Notation	Meaning
A	Shows the given image is domain A
B	Shows the given image is domain B
D	Discriminator
X	Sample images from category A
θD	Discriminator Parameter
θG	Generator Parameter
G_{xy}	Indicates the generator make from domain A to domain B
et.al	and others
T_G	Indicate ground truth
Z	Noise data sample

ABSTRACT

Visual saliency refers to an area of an image that attracts human attention. The Human Visual System (HVS) can focus on specific parts of a scene, rather than the whole image. Visual attention describes a set of cognitive procedures that choose important information and filter out unnecessary information from cluttered visual scenes. Images become a soul in computer vision since it contains plenty of information and human being receives 80% of information through vision. In processing the whole image while only a certain part of an image is needed, more resources are consumed. Instead of processing the whole pixels of an image, specifying only the needed pixel is computationally efficient to minimize the efforts. This is achieved by using GAN with CHASPP module and EfficientNet-B7 which uniformly scales up all dimensions of the image (depth, width, and resolution) is selected as feature extractor in this study which improves the way of extracting features in visual saliency prediction. Different datasets like CAT2000, MIT1003, DUTOMRON, and PASCALS are used in this study to illustrate the efficiency of the selected models and techniques. Human attention modeling focuses on a bottom-up approach that computes the impact of visual stimuli popping from its surrounding. However different models and algorithms have different results in the prediction of the attention area of an image. In this study, we developed effective visual saliency prediction using GAN with CHASPP and other factors like edge loss and perceptual loss. CHASPP module scored the best result on the same datasets measured by different evaluation metrics. It improved the baseline work of SalGAN+ASPP from 3.356 ± 0.04 *to* 3.851 ± 0.01 *(SalGAN+CHASPP+e). This study concluded that the CHASPP module, edge loss, and perceptual loss have a great influence on visual saliency prediction using a generative model.*

Key words: *- Visual saliency prediction, Attention area, Generative Adversarial Network, Cascaded Hierarchical Atrous Spatial Pyramid Pooling, Low-Level Features, High-Level Features*

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

The role of human visual attention is crucial in visual saliency. The spatial placement of an image that attracts human attention is referred to as visual saliency. To focus on a special location instead of processing the entire image, we need to move our eyes to a specific location (Ghariba et al., 2019).

Sometimes, a processing function needs to be applied only to a portion of an image. The selective perception mechanism greatly reduces the size of data processed by the human visual system, thereby enabling humans to suppress unimportant stimuli when processing complex visual information. Visual attention describes a set of cognitive procedures that choose important information and filter out unnecessary information from cluttered visual scenes. Attention is important for anybody to perceive anything at all. Human performs different types of attention. The first, Focus Attention the ability to respond discretely to specific visual, auditory, or tactile stimuli (e.g., reading the overlaid captions in a movie). Second, Sustained Attention the ability to maintain a consistent behavioral response during continuous and repetitive activity (e.g., stay attentive during the movie). Third, Selective Attention is the ability to selectively maintain the behavioral or cognitive resource on specific stimuli while ignoring the distracting or competing stimuli. Forth, Alternating Attention the ability to switch between multiple tasks with different demands. And the last type of human attention that we emphasize in this study is Divide Attention is the ability to respond simultaneously to multiple tasks with different tasks.

First, we scan the visual spotlight surrounding the environment, focusing on things like words, faces, images on a package, reading books, and a variety of other objects. Most recently visual Saliency data are collected with mouse clicks or webcams. Via saliency detection, the key features of an image can be extracted, and the computing resources are assigned properly, it effectively improves the efficiency and accuracy of image processing. Therefore, Saliency prediction technology is commonly used in numerous visual tasks, like object detection, tracking, image segmentation, and retrieval. Other computer vision applications that VSP can

help with include salient object recognition (Liu & Han, 2016), image retrieval, multiresolution imaging, and scene categorization.

Generally, Human visual attention (HVA) runs on two approaches. The first is a bottom-up approach that focuses on low-level features, like intensity, color, edge orientation, and texture. This strategy helps in the isolation of areas that display basic environmental features. The second is a task-driven top-down strategy that requires explicit awareness of the visual scene's context. Moreover, it is founded on the qualities of the target object.

Different deep learning models are used for visual saliency prediction. Most of them like RNN, LSTM are computationally less performance than GAN (Prior, 2020). GAN model come with different module to improve its performance. First, GAN with a self-attention module is used in both generator and discriminator to fuse semantic features with different spatial resolutions for visual saliency prediction (Sun et al., 2022). Second, GAN with transformer module is used to strengthen the modeling ability of GAN model. Also, several novel multi-scale inspection schemes, including multi-scale weight mask, multi-scale adaptive thresholding and multi-scale image patch-based defect evaluation are used in the model (Liu et al., 2021). But, both models require high capacity of Graphical Processing Unit (GPU) to process which is very expensive. CHASPP module which required less amount of GPU is used with GAN in this study.

1.1.1. Generative Adversarial Networks (GAN)

GAN is an AI algorithm developed and designed in 2014 by Ian Goodfellow (Goodfellow et al., 2020) to solve the generative modeling problem. The training of a GAN involves the use of both neural networks known as the generator and discriminator. These models are competing with each other in a contest where the generator network produces new data samples from random noise close to that of the training set, while discriminator networks try to categorize the data as real and fake.

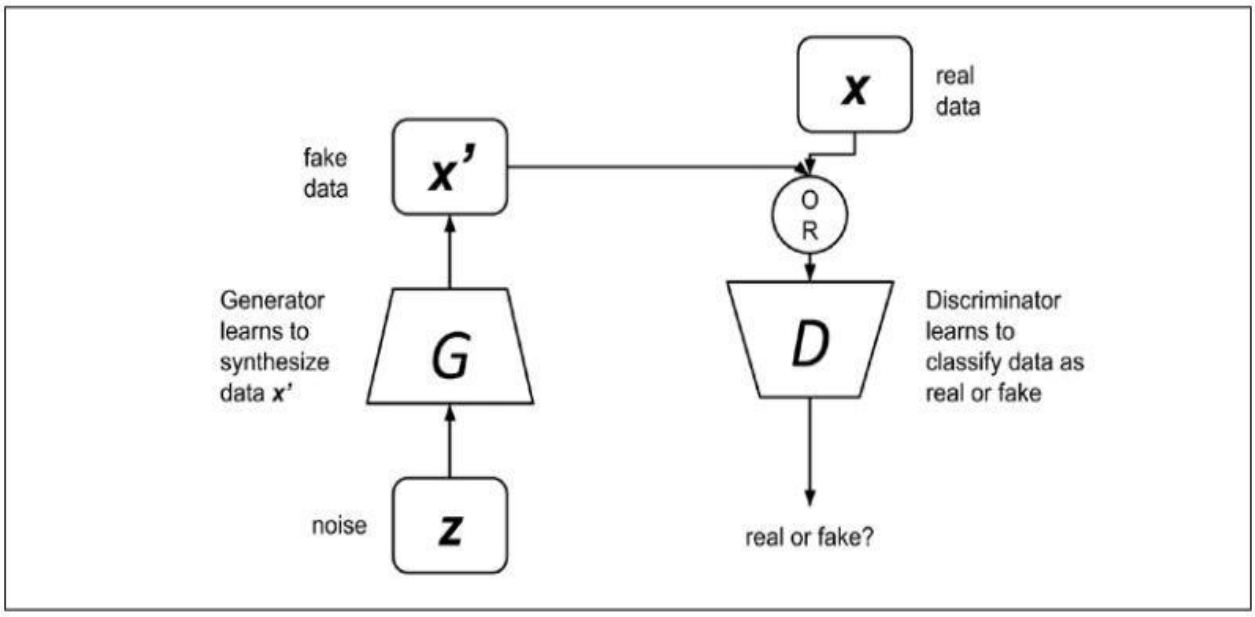


Figure 1.1 GAN Architecture (Source adapted from: *(Atienza, 2020)*)

1.1.2. Cascaded Hierarchical Atrous Spatial Pyramid Pooling (CHASPP)

ASPP is a module that can extract semantic information from a variety of scenes. (Lian et al., 2021) have proposed a CHASPP module consisting of two cascaded components and each component to densify the sampling distribution by utilizing dilation with variable rates, two Atrous convolutions were incorporated to create a hierarchical PP structure. The significant benefit of CNN is that it could learn hierarchical feature from the images automatically instead of handcrafted feature extractions that allows us to use CHASPP. Since it is composed of two series forms of ASPP, concatenation is applied at the end. The first layer's output becomes the second's input, and so on until the last layer.

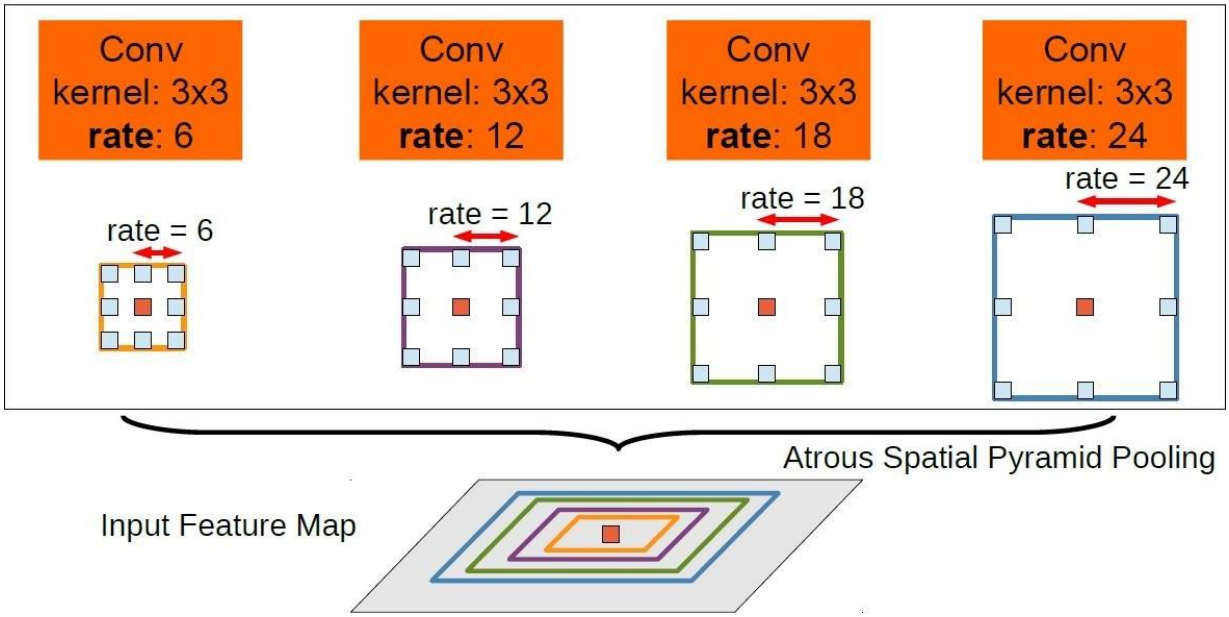


Figure 1.2 CHASPP Architecture (Source adapted from:(Lian et al., 2021))

1.2. Motivation of the Study

Visual saliency is now the hottest topic in computer vision (CV). This is due to its availability and versatility in application. Gaze-aware compression and summarization, activity recognition, object segmentation, recognition and detection, image captioning, image cropping, question answering, advertisement, novice training, patient diagnosis, and monitoring are some of the most important applications (B & Kamath, 2017). Checking only the part of images that attracts human attention and enhances the extraction of essential features from images and ignoring the irrelevant part makes our everyday life easier.

To provide the best result, many researchers propose a lot of approaches for visual saliency prediction network to identify the region of interest and extract relevant information from the image but their result still has limitation. This study improves this problem by extending the result obtained by the previous work.

1.3. Statement of Problem

Even if visual saliency prediction gets much attention in computer vision, there is a huge gap between attaining goals and expected results. A lot of challenges are phase in processing images for different purposes (Hegadi, 2010)(Riche et al., 2013). For example, we take an image as

input and use the whole image to get the output, but not the whole input contributes equally to the output. So, if we have huge data where only a few pixels are required to produce the result, processing the whole input is not computationally efficient. Instead of processing the whole pixels of an image, specifying only the needed pixel reduces the computational complexity. VSP, which helps to reduce the computational burden, used to solve such a problem.

In the paper we picked up as baseline work, architecture is designed with ASPP (Kroner et al., 2020) which has the following limitations. First, Because of the limited number of sampling ranges of ASPP, many valuable global features, and contextual information cannot be sufficiently sampled, and this degrades the representational ability of the segmentation network. Second, A large number of local detail characteristics is easily discarded because of the sparse distribution of the effective sampling points in the atrous convolution kernels of ASPP (Lian et al., 2021). And also, the baseline uses CNN in discriminator which feeds one input image to the network; it gives the probability of the whole input image size that they belong in the scalar vector as an output. This made the model inefficient. They used cross entropy loss that considers the similarities of prediction and ground truth and does not explicitly capture the structural differences between the prediction and its corresponding ground truth. The problem of using only selected potential scenes per image and including only relevant information for the recognition process still needs further study.

Generally, in this study, the problem of processing the whole image which consumes resources like time and memory solved by using fast processing for saliency prediction and improved accuracy.

1.4. Research Questions

The following are the research question answered by this research.

- ✓ **RQ1:** What are the challenges of visual saliency prediction in computer vision applications?
- ✓ **RQ2**: What is the impact of Cascaded Hierarchical Atrous Spatial Pyramid Pooling in visual saliency prediction with Generative Adversarial Network?
- ✓ **RQ3**: What are the determining factors to improve the visual saliency prediction?

1.5. Objective of the Study

1.5.1. General Objective

The general objective of this study is to design GAN based efficient Visual Saliency Prediction using generative models.

1.5.2. Specific Objectives

The research focuses on the following areas to satisfy the aforementioned goal:

- ✓ Reviewing journal, literature, workshop and magazine to understand VSP challenges and its application in CV.
- ✓ Integrate the CHASPP module with Generative Adversarial Network model.
- ✓ Train the proposed model on the available dataset.
- ✓ Include the edge loss and perceptual loss to the model to update the parameter during train.
- ✓ Build the model and search for the optimal hyperparameters to improve the model performance
- ✓ To evaluate the performance of developed VSP using various metrics for further improvement.

1.6. Significance of the Research

The image contains plenty of crucial information and humans acquired 80% of information through visuals(Yan et al., 2022). Processing the whole image could be costly as we have seen in the problem statement. This study should be applied to different scenarios and serve as a primary step for other applications such as image retrieval, health care, object detection, image cropping, Image captioning, photo editing, face recognition, and surveillance. Generally, the visual saliency prediction architecture presented in this study has been applied to different applications listed above with success in identifying the attention area of the input image. Future works might benefit from extending and customizing this work on a large dataset and utilizing it for different applications.

1.7. Scope and Limitation of the Study

1.7.1. Scope of the Study

This study concentrates on developing architecture for VSP on natural images. Implementing the GAN-based visual saliency prediction to achieves the best result than previous work by taking images as input and predicting part of the images which attract human attention. This will be done by feeding images to the generator and creating images that resemble the input one and pass to the discriminator which identifies whether the image comes from real or fake data and identify the area of interest by extracting feature from the image. Generally, this study emphasizes images.

1.7.2. Limitation of the Study

This work does not include the following due to tight schedules and resources limitations.

- ✓ Predicting video saliency is not included in this research.
- ✓ Forecasting an action and activity is not part of this study.

1.8. Application of the Study

Visual saliency prediction is a precondition to determining the area of interest of an image on which we concentrated. It has several applications (Borji, 2018) in different areas like:

- ✓ ***Image retrieval***: It is a method for accessing or seeking images in a database. If we want to get only relevant information from an image, we must identify the part of the images that are important for our purposes. To do so salient prediction is the absolute quick fix.
- ✓ ***Health care***: technology makes the life of human being simpler. Its improvement in health care is also reached an unexpected level. Visual saliency prediction helps doctors and nurses treat their patients effectively with minimum effort by focusing only on the infected patient's body that is specifically identified with the help of salient prediction and detection.
- ✓ ***Image captioning***: notice the object's image caption is very crucial.

- ✓ ***Image cropping***: It selects the region of interest and then we can crop the unnoticeable region. Resource problems can be handled by removing irrelevant parts and processing them efficiently.
- ✓ Surveillance
- ✓ Object segmentation, recognition, and detection.

1.9. Organization of the Thesis

Even though it is not a fully standardized format globally, this study adheres to the formats and standards outlined in the ASTU Guideline for Postgraduate Studies.

This thesis is organized by categorizing parts into chapter titles, subtitles, and sub-sub titles. Based on these ideas, the arrangement was as follows: acknowledgement, abstract, acronyms, introduction, motivation, statement of the problem, objectives, research question, scope, and limitation were included in chapter one.

The background literature and related works on visual saliency prediction and Generative Adversarial Network are discussed in chapter two.

Chapter three: represents the research methodology and techniques used to develop the solution, the data collection mechanism, preprocessing and annotation techniques, design tools, and model and evaluation methods.

Chapter Four: elaborates on the proposed model. The proposed architecture, loss functions, and training pseudocode have been discussed.

Chapter Five: discuss implementation detail of GAN-based visual saliency prediction with CHASPP.

Chapter Six: explained result and discussion on VSP and different evaluation metrics results are compared and the last chapter included conclusion and future work. Finally, references and appendix and sample of code were written at the last section.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORK

2.1 Introduction

CV is a field of computer science focused on developing digital systems that can process, analyze, and comprehend visual data (pictures or videos) in the same way as people do. The goal of a CV is to teach machines how to read and comprehend images down to the pixel level. This growth starts from processing at the pixel level to machine learning in recent years (Babich, 2020).

ML is an AI area that automates the process of developing analytical models and allows devices to adapt to new situations on their own. Intelligent big data management, smart gadgets, and engaging customer experiences are just a few of the benefits of machine learning. ML is a type of machine learning that creates computational objects that learn over time based on their experiences. It is divided into three types: supervised, unsupervised, and reinforcement learning. In a supervised situation, ML is given a collection of inputs as well as a set of target outputs. The algorithm then tries to learn patterns that can be used to automatically map input data points to their right target output while unsupervised learning has no labels associated with the input data and thus, we cannot correct our model if it makes an incorrect prediction (Rosebrock, 2017). The reinforcement learning technique is a reward-punishment-based aspect of ML. The evolution of computer vision applications is heavily based on ML. However, processing many tasks such as video and images is very difficult by using machine learning techniques because of their complexity. This problem brings another option that can handle complex data unlike machine learning called deep learning.

Deep learning is a discipline of ML that deals with ANN, which are algorithms inspired by the structure and function of the brain (Brownlee, Machine Learning Mastery, 2019) that represent complex data. Deep learning algorithms have a lot of layers and information routes through each of the layers. The term "deep" implies the network's number of hidden layers. The network has more layers to learn more. ANN is one of the deep learning algorithms that associated artificial neurons which are made up of perceptron’s in neural layers. Another deep-learning algorithm

that can process a sequence of inputs and keep its state while processing the following sequence of inputs is the RNN. CNN is the most widely used algorithm in recent years relative to other algorithms, and it solves the issue that existing ANNs have. It's a type of deep learning model that uses a grid layout, such as photographs, that is motivated by the coordination of the animal visual cortex and meant to learn spatial hierarchies of features, from low- to high-level patterns, automatically and adaptively (Yamashita, Nishio, Kinh, Do, & Toga, 22 June 2018).

As a result of the rapid growth of technology, many algorithms are developed from time to time to improve the application of AI in different areas. Deep learning is one hot area in computer vision that solves the problem faced in ML techniques. The new algorithm comes with the ability to handle and process complex data such as images and video in generative models. GAN and Auto encoders are the most commonly used ones. Let's see GAN in detail.

2.2 Generative Adversarial Network (GAN)

GAN is a generative model framework developed by Ian Goodfellow in 2014 (Goodfellow et al., 2020) to model two networks that compete with each other. The first one is called a generator which creates real resemble data from the given input and random noise. Another network known as a discriminator attempts to distinguish authentic data from data generated by the false generator network. To generate data, the generator must clearly understand the real data's pattern and create data very close to the real. Since both networks are competing with each other, one network improves itself based on the criticism from another network. The generator synthesizes a batch of samples, which are then given to the discriminator, along with real instances from the domain, to be classified as real or false. The discriminator is tweaked in the next round to improve its capacity to distinguish between true and false samples, while the generator is tweaked based on how well the generated samples tricked the discriminator. (Brownlee, Machine Learning Mastery, 2019).

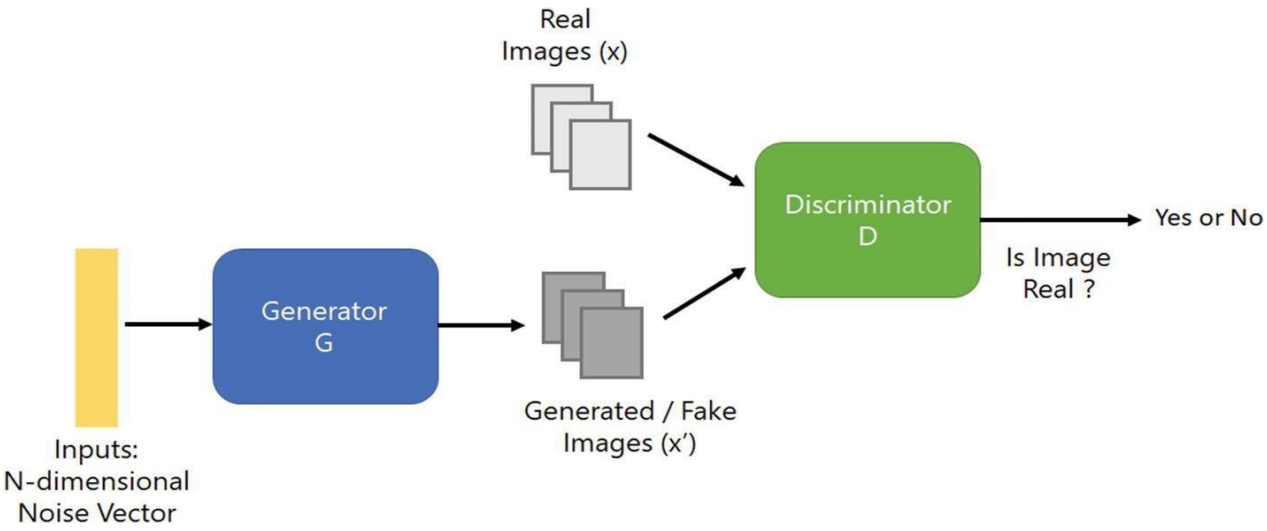


Figure 2.1Generative Adversarial Network Architecture Source: Adapted from(Goodfellow et al., 2020)

2.2.1. Training GAN

Training is the process of finding the weight and the loss of the given input which converge the network quickly. The well-trained model is the model which estimates the weight that gives the best result at the end of training. The training algorithm for a GAN is fairly complicated because it involves two independently learned networks. The most complex question in GAN is how to ensure that the generator-discriminator network is consistently trained. The goal of GANs training is to discover the parameters of the generator which confuse the discriminator and to find discriminator parameters that increase classification accuracy. During training when parameters of the generator are updated, the discriminator parameters are fixed and vice versa. Now let's see overall the training of GAN.

Pseudocode for training GAN

- Train Discriminator
  - ✓ Take random input from a real sample: x
  - ✓ Take arbitrary noise vector z and generated a group of fake samples: G(z)=x*
  - ✓ Compute the losses for D(x) and D(x*) and back propagate the total error to update parameters of Discriminator to reduce discriminator loss
- Train Generator
  - ✓ Take random noise vector z and generate a group of fake samples: G(z)=x*

- ✓ Compute classification loss for D(X*) and backpropagate the total error to update parameters generator to maximize the discriminator loss

2.2.3. Generative Adversarial Networks Architectures

DL algorithms have emerged as a result of the exponential growth of technology. Not only developing the new algorithm, updating the existing one is a familiar task to improve AI performance. GAN is the most often modified deep learning algorithm. Let's see some GAN architecture

***Conditional GAN (CGAN)*:** CGAN is a version of GAN that can be developed by simply feeding the data, *y*, wish to condition on to both networks i.e., generator and discriminator. It was proposed by Mirza et al. to tackle the issue of conditional image generation (Mirza & Osindero, 2014). Extending GAN to create a conditional model in which the discriminator and generator are both conditioned on new input y. y could be any kind of information like class labels or data from other formats. The adversarial training framework enables concrete flexibility to combine the generator's prior input noise $p(z)$and **y.** Researchers can now work on fascinating study fields including image-to-image translation, style transfer, and visual salient prediction thanks to this effort.

The main objective function of a two-player minimax game would be as:

$$Min_D Max_G V(D,G) = E_x \sim P_{data}[\log(D(x))] + E_z \sim P_z * \log(1 - D(G(z)))\dagger \qquad (2.1)$$

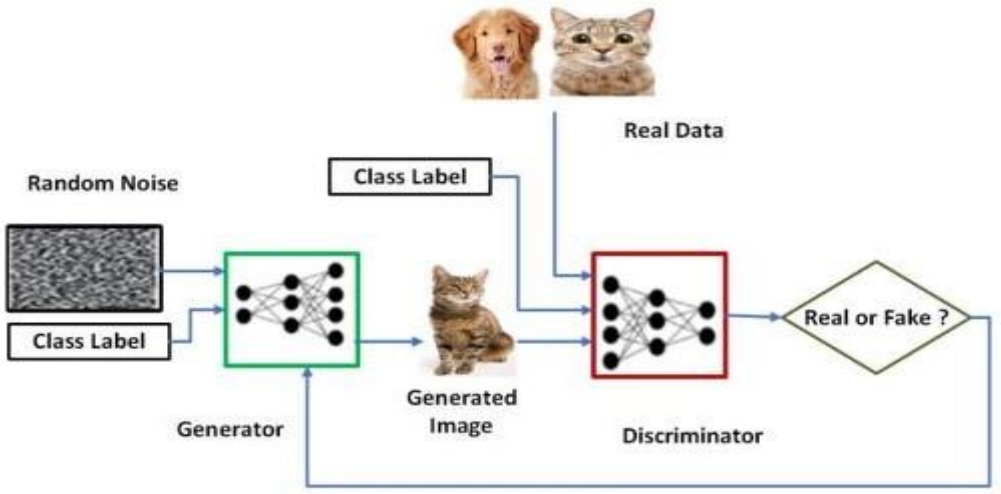


Figure 2.2 Architecture of Conditional GAN Source: Adapted from (Mifta, 2021)

2.2.4. GAN Objective Function

Everything in this world grows and changes at a high rate at the moment. The same is true for GAN. The first GAN developed by Goodfellow was not the complete model that satisfies the experts in the area of deep learning (Phan et al., 2020). That is why different researchers come up with a modified version of the GAN model from time to time to tackle the problem found in the old version. We have the various improved GAN model today. The goal functions employed in GAN, as we know, discover the gap between the true data distribution and the generated data distribution. But GAN faces a problem during training data. The vanishing gradient is the major one. The objective function can be reformulated to solve this difficulty. GAN's varied objective function is created to increase image quality and performance. Let's see some of them.

Wasserstein GAN: WGAN is a type of GAN that minimizes an approximation of the EM distance that is both reasonable and efficient. Proposed by (Arjovsky et al., 2017) to improve the problem that happens in the GAN model by training without the need of balancing between generator and discriminator. To stabilize the discriminator's training, Arjovsky uses Wasserstein distance instead of JS divergence. WGAN is developed to solve the problem of learning probability distribution with unsupervised learning by using generative samples rather than estimating the density. The goal of WGAN is to determine how well the model distribution matches the real distribution. Instead of Jensen Shannan divergence, WGAN calculates the similarity between actual and false data using Earth Mover's (EM) distance. SD is a comparison method for two probability distributions' similarity. Both Discriminator and generator have their own cost as the following.

$$LWGAN(D) = E_{x\sim pdata(x)}\,[D(x) + E_{z\sim p(z)}{}^{*}(1 - D(G(z)))+] \tag{2.2}$$

$$LWGAN(G) = E_{x\sim p(z)}[1 - D(G(z))] \tag{2.3}$$

***Least square relativistic GAN*:** LSRGAN is proposed by Zhang, Cheng, Jiang, & Kou (Zhang, Cheng, Jiang, & Kou, 2020) to generate a visually similar image with perceptual texture details in addition to avoiding the vanishing gradient problem by adopting least square loss. Most content loss function like cross-entropy loss is susceptible to vanishing gradient during updating

generator using generated samples. It gives better results in single image super-resolution by applying spectral normalization that allows discriminator trained stable. Design least-squares relativistic discriminator that enables to obtain of high-quality perceptual manifolds of an image. In the process of magnifying the quality of generated image's objective function has a great role. The least-square relative adversarial loss function used in LSGAN which was inspired by RaGAN (Jolicoeur-Martineau, 2018) can be described as follow.

$$L_D^{ADV} = E[(C(I^{HR}) - E[C(G_{\theta G}(I^{LR}))] - 1)2] + E[(C(G_{\theta G}(I^{LR})) - E[C(I^{HR})] + 1)^2)] \quad (2.4)$$

$$L_G^{ADV} = E[(C(G_{\theta G}(I^{LR})) - E[C((I^{HR}))] - 1)2] + E[(C((I^{HR})) - E[C(G_{\theta G}(I^{LR}))] + 1)^2)] \quad (2.5)$$

Where, $G_{\theta G}I^{LR}$ and I^{HR} are the produced image and the true original sample, respectively. $C(.)$ Signify the output of a non-transformed discriminator.

***Perceptual Loss*:** Perceptual loss is a concept in the loss function that promotes natural and perceptually attractive outcomes. The significant element that impacts the performance of visual salient prediction is the loss function. The purpose of perceptual loss in visual salient is to find the most important part of an image. Thus, choosing the loss that would adhere to that goal is the backbone of this work (Mikhailiuk, 2021). Losses on a feature-by-feature basis are neural networks that are used as feature extractors in a novel class of loss functions that have recently acquired a lot of traction. The network implementing the loss function is frequently not bijective due to pooling in the hidden layers, which means that various inputs to the function may result in identical latent representations. As a result, feature-wise losses are commonly combined with a regularization term like L2 or L1 norms, and the weights of each loss component must be carefully tuned.

2.3. Style Transfer

The act of changing an image's style from one pattern to the other genre image is called style transfer. It is the portion of image conversion and it was introduced by L. Gatys (Gatys, 2015) to alter the style of image. Style translation is combining the content image and style reference image to give output by painting it. Style transfer tries to keep the content of the image intact while applying the style of the other image. It extracts the content and style from the middle layers of the NN model. To tackle the problem of style transfer (L. Gatys, 2015) used a pre-

trained convolutional network with content and style loss. (L. Gatys, 2015) learns to translate styles in the following images.

Figure 2.4 Style Transfer. Source: Adapted from (L. Gatys et al, 2015).

2.4. Image to Image Translation

I2I translation is a category of vision and graphics problems in which the goal is to figure out how to transfer an input image to an output image. I2I was given enough training data, transforming one possible representation of a scene into another. It can be categorized as unsupervised and supervised image-to-image translation based on the training dataset. In supervised I2I translation (Heil, Vats, & Hast, 2022) there might be pairs of input-output and synthesized color images from features labeled. Unsupervised I2I translation, where the data for training is unpaired (Zhu et al., 2017).

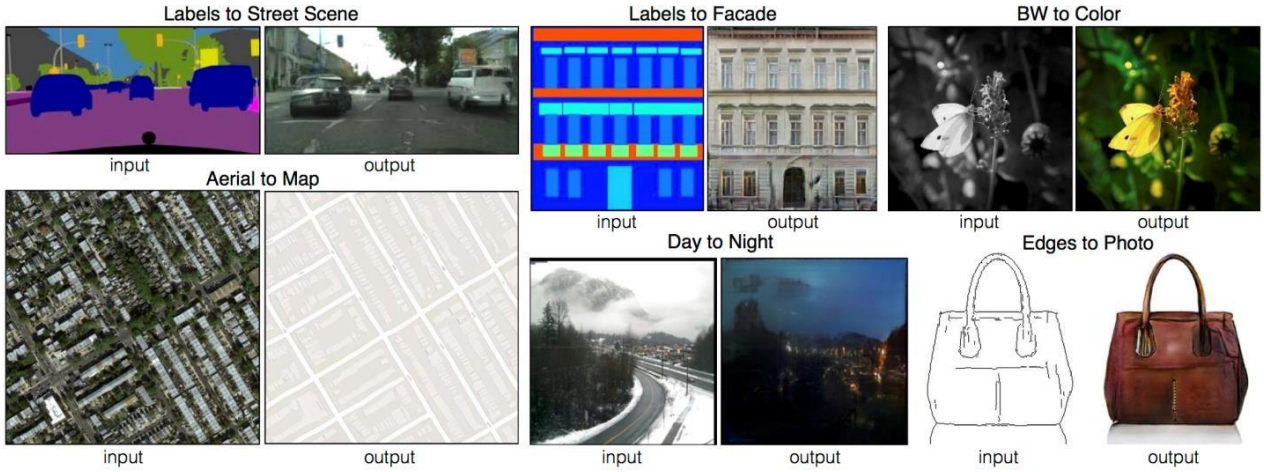


Figure 2.5 Examples of image-to-image translation applications. Source: Adapted from (Zhu et al., 2017).

2.5. Human Visual System

Both the eyes and the brain are a portion of the visual framework. When light enters the eye and hits the retina, photoreceptors send electrical signals through the optic nerve to the back of the brain, where the early stages of visual recognition occur. At this point, the brain gradually sends the filtered signals forward, allowing them to "see" and respond. There are many ways the brain can contribute to our visual framework. Various parts of the brain are contained within the visual framework. Let's take a look at some of them.

The superior colliculus (SC): The superior colliculus (SC) is a part of the brain that integrates visual, auditory, and somatosensory information to trigger motor orders. The major function of the superior colliculus is to physically guide the brain's sensory components toward interesting stimuli. The orienting activity allows the eyes, ears, and other organs to acquire comprehensive information about exterior stimuli. Orienting movements in humans and other primates may include head movements, but they are predominantly performed by redirecting the eyes utilizing quick eye movements known as saccades (Vanderah, 2021).

Ganglion Cells: Ganglion cells are the vertebrate retina's final output neurons. Bipolar cells and amacrine cells provide information about the visual world to ganglion cells (retinal interneurons). (NELSON, 2007). It contains 125 million photoreceptors. Retinal ganglion cells use their axons, which are long fibers that make up the optic nerve, to interpret visual information that begins with light entering the eye and transfers it to the brain.

***Lateral Geniculate Nucleus (LGN)*:** Forms the gateway to the visual cortex. Because of its tight interaction with the retina, the LGN is classified as a first-order thalamic nucleus. For each LGN layer, just one eye delivers input. On top of each other, the LGN layers are stacked like pancakes. Small, medium, and big cells are divided into distinct layers and convey different types of visual messages to the main visual cortex as a result of layer specialization.

In general, our visual perception process goes like this:

- ✓ The brain picks up the general attributes, like form or whether an image is divided into sections.
- ✓ Brain examines each section individually, scanning for patterns and noticing inconsistencies.
- ✓ Only then does the brain begin to analyze a picture in detail, committing elements of it to long-term memory.

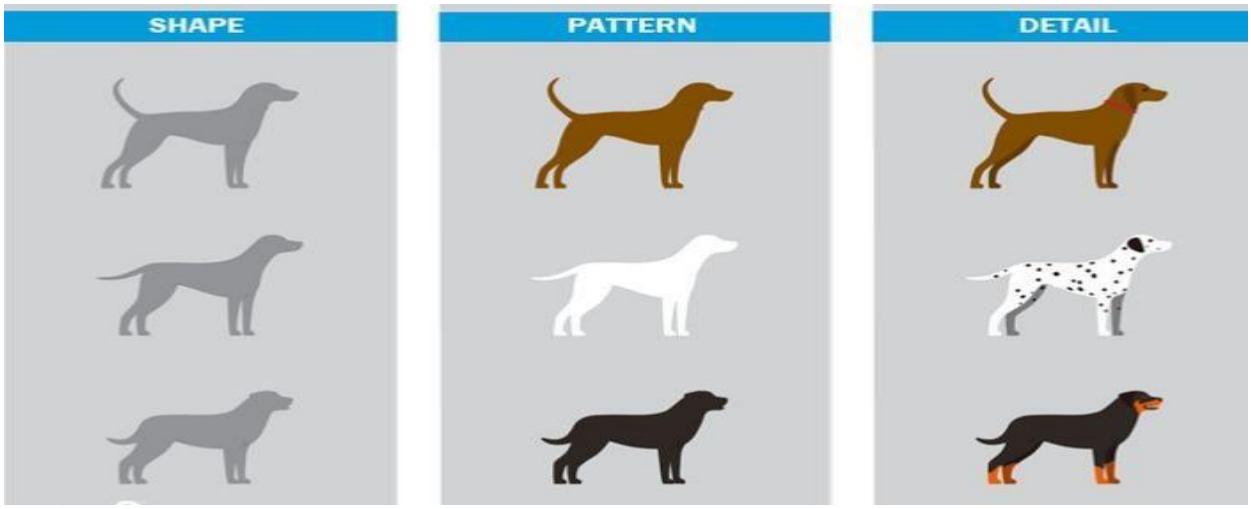


Figure 2.6 Human Visual Perception Procedures. Source: Adapted from ((NELSON, 2007)

Salience map: In the context of visual processing, saliency refers to the image's unique properties (pixels, resolution, etc.). Images that display the distinct features of each pixel in a given image are known as saliency maps (Ghariba B. M., 2020). These distinct features depict the image's aesthetically appealing regions. A topographical representation of them is a saliency map (Sharma, 2018). A salience computational model illustrates how low-level external visual features like color, direction, brightness, and motion are merged into a single global map that represents each point on the map's relative 'salience'. Koch and Ullman proposed the notion, which was later implemented by Itti et al. For this, Itti's saliency model is used to explain things.

The model takes into account three aspects of an image: colors, intensity, and orientations. The saliency map displays these combinations.

The steps are given below:

- ✓ The image that is fed into the system is used to extract the three features mentioned above.
- ✓ The images are converted to a red-green-blue-yellow color space for color conversion. For intensity, the images are transformed to gray scale.
- ✓ Gabor filters are used to translate the orientation feature concerning four angles.
- ✓ To construct feature maps, all of these processed images are used to create Gaussian pyramids.
- ✓ Each of the three features is used to create feature maps. The saliency map is a composite of all feature maps.

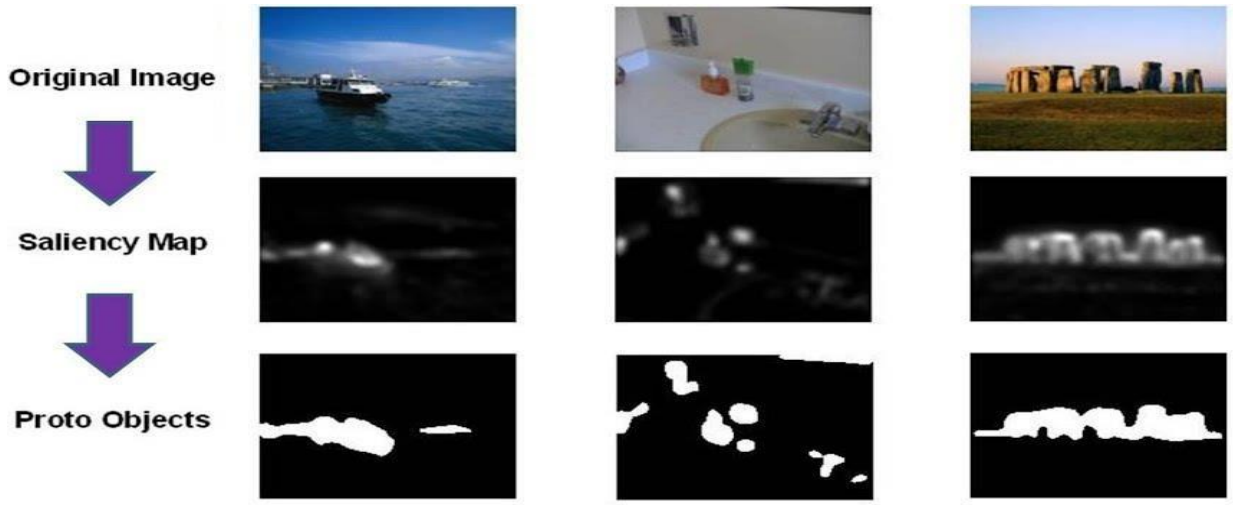


Figure 2.7 Depiction of saliency maps Source: Adapted from (Yan Li, 2019)

2.5.1. Priority Map

Regardless of whether the prioritization is based on bottom-up or top-down considerations, a priority map is a topographic space in the brain in which activity codes for the priority of locations in that space. Priority maps in different parts of the brain can be combined to draw attention to certain aspects of visual stimuli. The visual cortex has a stimulus-driven salience map, the midbrain has a learned value-based map (favoring the bottom right corner in this case), the frontal cortex has an instruction-based map ("attend top left"), and the frontal eye fields have

a resultant integrated priority map (Christiaan Klink et al., 2014). This is depicted in the following diagram.

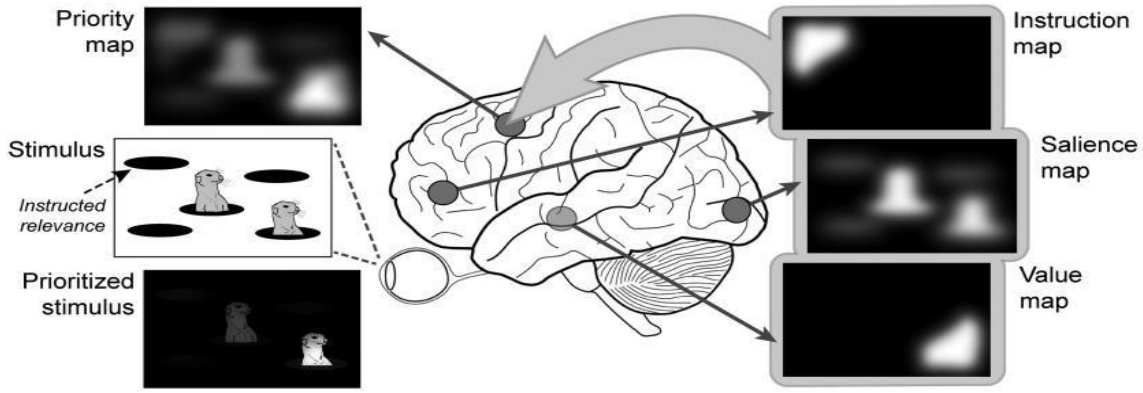


Figure 2.8 Priority Map. Source: Adapted from (Christiaan Klink et al., 2014)

2.5.2. Saliency Detection

The purpose of saliency identification is to identify the important aspects of natural images that attract people's attention. Ting Zhao and Xiangqi Wu suggest Pyramid Feature Attention (PFA) as a new salient object recognition method. Saliency detection is a technique for detecting what is most important in an image with minimum time and effort. To collect rich context information, they use a context-aware pyramid feature extraction module and a channel-wise attention module for high-level features. They use the spatial attention module to filter out some background details for low-level features (Zhao & Wu, 2019).

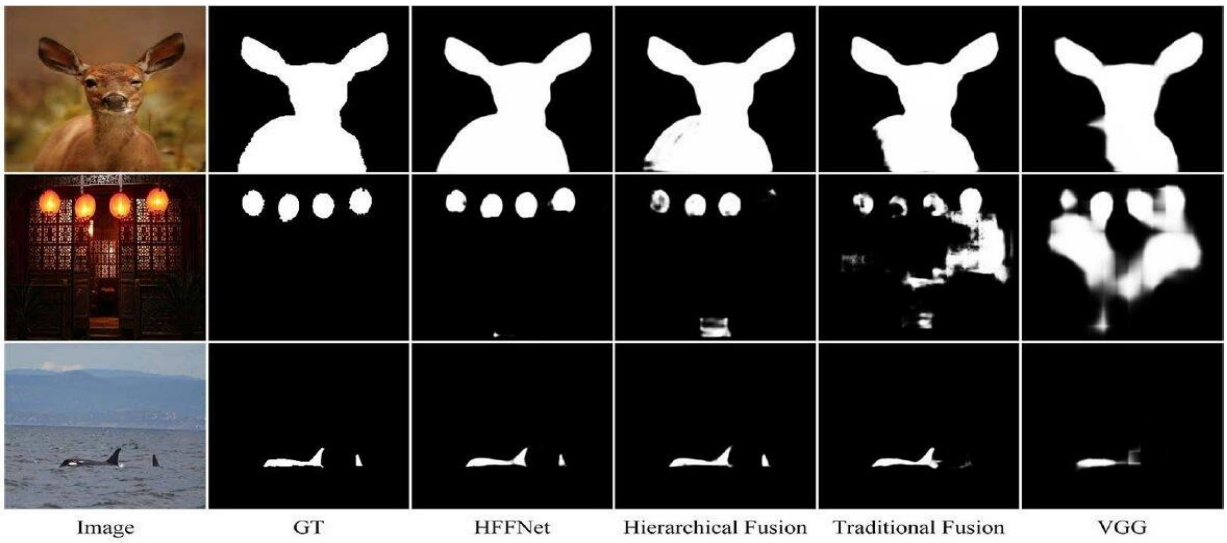


Figure 2.9 Example of salient object Detection. Source: Adapted from (Zhao & Wu, 2019)

2.6. Attention Mechanism

Attention mechanisms have developed from studies of human vision. In the human visual system due to certain reasons, not all fractions of data are processed. This bottleneck motivates many AI experts to find attention mechanisms by mimicking and broadening the dispersion of human attention throughout the observation of pictures and videos. The attention mechanism allows the human to prioritize the perception to deal with others effectively (Zhang, C. Lipton, Li, & J. Smola, 2020).

Most significant improvements in the area of computer vision and natural language handling have been achieved throughout previous times. It was described that processes of attention might function on models and also mostly focused on the relationship between the perception system of the mental and the human sight. The concept is introduced by Bahdanau et al to deal with the process of a sequence-to-sequence model that can't process long input well. To improve the problem of processing long sequence input various attention mechanisms, emerge. It is an excellent fit for saliency detection because attention systems have a strong ability to choose features.

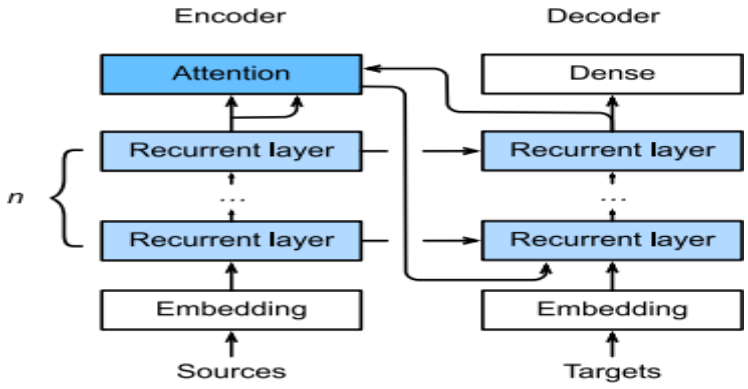


Figure 2.10 Attention mechanisms with layers in the s2s model. (Source: Adapted from (Zhang, C. Lipton, Li, & J. Smola, 2020)

2.7. Related Work

Even though many models have shown good results had achieved past saliency predictions still in challenging problem for some reason.

(Pan et al., 2017) Proposed SalGAN for the saliency prediction method, and trained with adversarial loss function with two networks: one predicts saliency maps from raw pixels of an input image; the other uses the first one's output to determine whether a saliency map is anticipated or ground truth. However, their work has three limitations. The first one is they used bilinear interpolation to restore the original image resolution, which reduces the performance of the model. Secondly, the absence of evaluation metrics that are very crucial for saliency prediction like DL (Bylinskii et al., 2019). The SalGAN model's third flaw is that it is trained without employing a discriminator to forecast the saliency map, instead relying solely on content loss, i.e., BCE.

Assens, Giro-i-Nieto, McGuinness, & E. O'Connor (Assens et al., 2019) Have proposed pathGAN that explores Omni-directional Scanpath prediction and improves the state of the art of visual Scanpath prediction on iSUN and Salient360 dataset. It consists of a generator and discriminator that use off-the-shelf networks to extract features from images and train an iterative layer. It was defined as predicting the likelihood of fixation based on the conditioned preceding fixation. However, there are certain limitations to this paper, such as the use of solely content loss and the lack of explicitly defined evaluation indicators.

Qin, et al. proposed a pre-defined architecture, BasNet (Qin, Zhang, Huang, Gao, Dehghan, & Jagersand, 2019) that focus on boundary quality to detect salient region in an image and it is composed of a densely supervised Encoder-Decoder network and a residual refinement module, which is in charge of saliency map refining and saliency prediction, respectively. But this paper focuses on low-level features only (Sethi, Rahurkar, & S. Huang, 2007) to predict the interest area of an image. That is its limitation.

Wenguan Wang & Shen (Wang & Shen, 2018), Proposed the CNN-based attention model to capture hierarchical saliency information from deep with global saliency information based on a skip-layer network to avoid vanishing gradient. Their model integrates multi-level saliency prediction inside a single network, reducing the redundancy of earlier approaches dramatically.

However, the model has limitations and inaccuracy in the result obtained as compared with other papers.

Kerkouri, Tliba, Chetouani, & Harba, proposed an end-to-end deep-based model named SALYPATH (Kerkouri et al., 2021) that efficiently predicts the Scanpath of images by leveraging features from the trained saliency model. They predicted the Scanpath of an image by using a VGG-based encoder-decoder equivalent to SalGAN's (Pan, et al., 2018) generator part through the fully convolutional network. Different metrics can be used to get efficient prediction like MulitMatch (MM) which compares Scanpath through different criteria, the fixation map created from the predicted fixation point is compared to the ground truth saliency map using NSS, and the percentage of anticipated fixation locations within the threshold saliency map is measured by congruency.

(He, R. Tavakoli, Borji, Mi, & Pugeault) Have developed a deep visual saliency model that analyzes how individual neurons learned at intermediate layers of the deep saliency model (Bruckert et al., 2021). They tried to improve the visual saliency based on the most commonly known and large dataset for visual saliency (SALICON and MIT300). The trained on pre-trained VGG16 model to get better accuracy as compared with the previous saliency model. However, the result they scored is not that good because of the limitation of data.

Table 2.1 A summary of previous works on saliency prediction

No	***Title***	***Objective***	***Model***	***Gap***
1	Contextual encoder-decoder network for visual saliency prediction (2020)	Identifying the attention area of an image with little effort	VGG16 with ASPP	▪ Much valuable global features and contextual information cannot be sufficiently sampled ▪ A large amount of local detail characteristics is easily discarded.
2	SalGAN: visual saliency prediction with adversarial networks (2018)	Predicting saliency maps from raw pixels of an input image and mapping it to the ground truth.	SalGAN: Adopting Encoder-Decoder architecture with original VGG 16 model	▪ Use bilinear interpolation to restore the original image resolution which reduces the performance of the model.
3	BASNet: Boundary Aware Salient Object Detection	Detecting salient region in an image focus on boundary quality	BASNet: Creating an Encoder-Decoder network with dense supervision and a residual refinement module	▪ Only focus on low-level features to predict the interest area of an image
4	PathGAN: Visual Scanpath Prediction	On the iSUN and Salient360! Datasets,	PathGAN: Using off-the-shelf networks extract characteristics	▪ Use only content loss

	with Generative Adversarial Networks 2018)	they improve the state of the art in visual scan path prediction.	from images and train recurrent layers to produce or discriminate scan paths accordingly.	
5	SALYPATH: A deep-based architecture for visual attention prediction (2021)	Through saliency model characteristics, efficiently predicts the scanpath of an image.	SALYPATH: consists of a VGG-based encoder and two predictor networks for predicting saliency and scanpath, respectively.	▪ The loss function is not used.

As shown in table 2.1, several types of research are conducted by different researchers to predict accurate attentive area in an image. However, their study has the limitation listed in table 2.1 like focusing only on low-level features, using only content loss, and using the module with a limited sampling range of numbers that bring the loss of valuable global information and local characteristics. And also, the use of inefficient methods to retain the size of results is the major limitation of the previous study. Therefore, this study solve those problems by substituting ASPP with CHASPP modules that have a higher sampling range with different dilation rate sizes which in turn helps to cover large distribution scope and adds high-level features for capturing semantic information. Since it has a large sampling range and a small number of parameters CHASPP is suitable to overcome the limitation of ASPP explained in the above table. The decoder is used instead of bilinear interpolation in the process of restoring the size to improve the visual saliency prediction and adding loss function particularly edge loss and perceptual loss to the existing one is also another method used. Including patchGAN discriminator in this study reduces the problem of finding salient region processing input as an NxN array of a vector rather than yielding a single scalar output that didn't use yet for VSP.

CHAPTER THREE

3. RESEARCH METHODOLOGY OF GAN-BASED VSP WITH CHASPP

3.1. Overview

The methodology utilized in conducting GAN-based visual saliency prediction with CHASPP is explained in this chapter. Dataset, preprocessing, feature extraction algorithm, baseline work, and patchGAN discriminator are explained in this section. Tools that are used to conduct the research and evaluation methods used to measure the result of the study are included in this chapter. The following diagram depicts the general process that flows through this study.

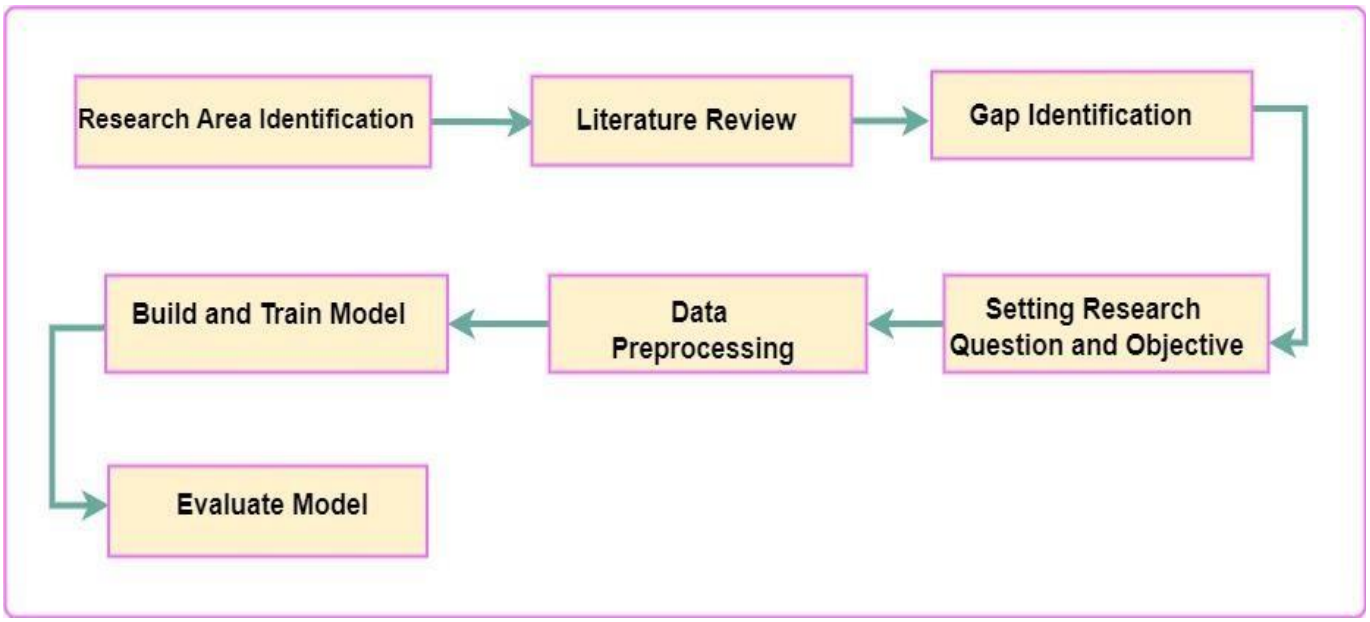


Figure 3.1 The Proposed Model Process flow

3.2. Data Preprocessing

Before starting training any deep learning model, the collected raw data are converted to the format that fit the model. Several strategies were employed to preprocess data to build a better dataset.

A. **Batch-Normalization:** The technique was just proposed a few years ago in 2015 (Chatterjee, Kiao , & Tushti, 2021), but it has now become the most often used deep learning model

strategy**.** Normalization has a considerable positive impact on the performance of your model. Normalization is scaling the data to be analyzed to a specific range such as [-1.0, 1.0] to provide better results. This eliminates the problems associated with floating point number precision. Because the activations of one layer of a neural network provide inputs to the next layer, normalizing them also help the network learn the parameters in the next layer more efficiently. As a result, by normalizing the activations of each layer, the entire learning process can be simplified (Chatterjee, Kiao , & Tushti, 2021).

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij} \tag{3.1}$$

$$\delta_j^2 = \frac{1}{m}\sum_{x=1}^{m}(x_{ij} - \mu_j)^2 \tag{3.2}$$

Where x_{ij} is i, jth element of the input, the i dimension is the batch, and j is the feature dimension, m corresponds to the training samples.

Then, the normalized image x can be defined as:

$$\times = \left(\frac{z-\mu}{\sigma}\right) \tag{3.3}$$

Where, z is the input image that has a value between 0 and 1,μ, and δ are mean and variance respectively.

B. Resize Image: images in the dataset could be of various sizes. For instance, some image has a size of 223×320 and others have 350×350 so this should be converted into the same size as the image. In deep learning, image scaling is a crucial preprocessing step that allows input data to be adjusted to the network's size. To save time, the input images are usually scaled to a low spatial resolution (Talebi & Milanfar, 2021).

3.3. Feature Extractions

Feature extraction is a mechanism that reduces the dimension without losing relevant information (Tan & Le, 2019). Processing raw data is very expensive and challenging in terms of resource consumption. The aim of extracting features is to get the finest feature from the large dataset by selecting the most influential feature(Chollet, 2017). These extracted features are easy to process and they can represent the original data as it is without losing any relevant

information. Different researchers proposed several deep CNN architectures for different purposes like Xception (Chollet, 2017), DenseNetI2I (Huang et al., 2017), and EfficientNet (Tan & Le, 2019). EfficientNet outperforms other CNN architectures with fewer parameters with improved accuracy. EfficientNet-B7 is employed as the backbone in the feature extraction generator in this study to increase network performance with low computational complexity. The top1 accuracy, as well as the number of parameters, is shown in Figure 3.2

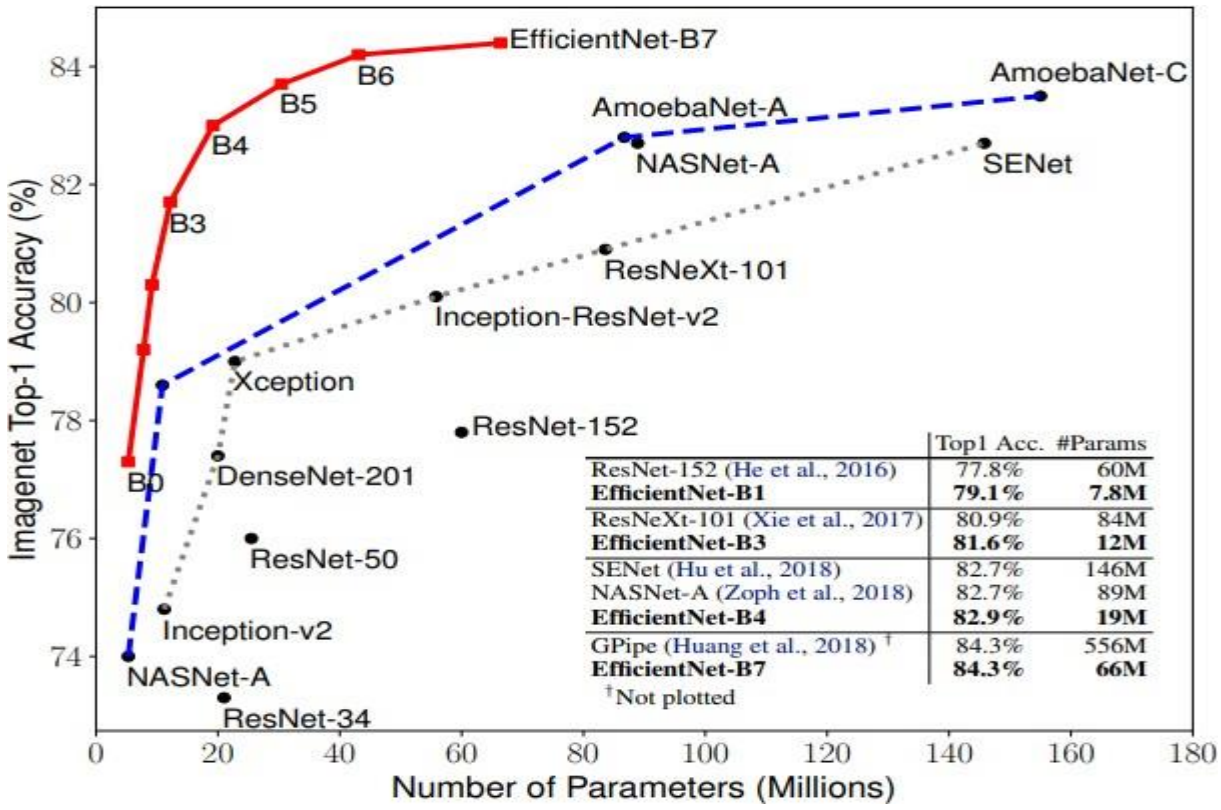


Figure 3.2 Model Size vs ImageNet Accuracy. Source: Adapted from (Tan & Le, 2019)

Using a compound coefficient, the convolutional neural network design and scaling method known as EfficientNet equally scales all depth (increasing the number of a layer), width (increasing the number of channels or feature map), and resolution dimensions. The EfficientNet scaling method evenly increases network width, depth, and resolution with a set of preset scaling coefficients, in contrast to standard practice, which scales these elements arbitrarily.

3.4. Datasets

To get the successful application of deep learning techniques good annotated data is extremely important. Due to the availability, accessibility and efficiency employ freely online available datasets such as MIT1003, CAT2000, DUTORMON and PASCAL-S are selected and each datasets with their corresponding ground truth dataset.

***MIT1003*:** is a dataset that contains 1003 natural indoor and outdoor scenes commonly used as training data based on the MIT300 benchmark which was released in 2012 (Le, 2019).[1]

***CAT2000*:** introduced by (Borji & Itti, 2015) which contains 4000 images 200 from each of the 20 categories, which cover a variety of scenes such as cartoons, art, objects, low-resolution images, indoor, outdoor, jumbled, noisy, pattern, random, and line drawings.[2]

***DUTOMRON*:** is introduced by Yang et al.in (Yang, Zhang, Lu, Ruan, & Yang, 2013) used for the evaluation of the Salient Object Detection task and it contains 5,168 images with high quality. The images have one or more salient objects with a somewhat cluttered background.[3]

***PASCAL-S*:** is a dataset introduced by Li et al. (Li, Hou, Koch, M. Rehg, & Yuille, 2014) for salient object detection consisting of a set of 850 images from the PASCAL VOC 2010 validation set with multiple salient objects on the scenes.[4]

3.5. Development Tools

Development tools are the tools used during the process of research conduct. To implement the designed work, several types of hardware and software development tools are used.

[1] https://people.csail.mit.edu/tjudd/WherePeopleLook/ALLSTIMULI.zip.

[2] http://saliency.mit.edu/trainSet.zip

[3] http://saliencydetection.net/dut-omron/download/DUT-OMRON-image.zip

[4] http://cbs.ic.gatech.edu/salobj/download/salObj.zip

3.5.1. Hardware Tools

The followings are hardware tools we used in this research processes for different functions.

Table 3.1 Hardware Tools

No	*Device Name*	*Purpose*
1	RAM	To store a large dataset
2	Flash Disk	To transfer data
3	Hard Disk	To increase the computation and to fasten the training
4	GPU	To accelerate the computation and training process

3.5.2. Software Tools

Software tools are used for writing the code, dubbing, visualizing results, and documenting the research process for reporting. Software tools are programming tools and the library we used are explained as follows:

***Jupyter notebook*:** It's an open-source web application that simplifies loading, configuring, and writing Python API code. It is an ecosystem that consists of many modules that assist us in quickly handling data and is very user pleasant. Providing the package and ensuring that the management is extremely appropriate.

***TensorFlow*:** It's an open-source machine learning platform for building models in client environments. This platform develops and deploys machine learning-based applications. It consists of many libraries that assist researchers in developing ml-powered applications.

Keras: is a python-based high-level deep learning API for neural networks. It supports multiple backend neural network computations and makes implementing neural networks easy. We used Keras in this study because of its simplicity, flexibility, and power.

3.6. Base Line Work

To improve the effectiveness of this study, the implemented model is compared to recent models on visual salient with GAN. The contextual encoder-decoder network (Kroner et al., 2020) was

used as a baseline in this study since the Contextual encoder-decoder network for visual saliency prediction achieves better quality salient images with small epochs, scored high values measured with different evaluation metrics described in section 3.7, and small training and testing time as compared to other previous visual saliency prediction studies.

The contextual encoder-decoder network (Kroner et al., 2020) uses bilinear interpolation to restore the image and ASPP for the representation of an area image that contains relevant information. This research shows the predicted output from the synthesized image is closer to the ground truth.

***PatchGAN Discriminator*:** PatchGAN is a convolutional neural network. The sole difference is that it maps an input image to an NxN array rather than a single scalar output. Each element of the NxN array corresponds to a patch in the input image (Kang & atul., 2019). (Bylinskii et al., 2019) The discriminator loss and the L1 loss are both used to train the generator model. L1 losses are well-known for causing fuzzy visuals. In many circumstances, L1 losses are unable to acquire high frequencies in images, but they can capture low frequencies. The discriminator's responsibility now is limited to capturing high frequency. PatchGAN was used to focus the model's attention on local image patches, which assisted in capturing high frequencies in the image. PatchGAN process the input image as an NxN vector that makes it has small parameters which in turn enable PatchGAN to run faster.

3.7. Evaluation Metrics

In recent years, many articles have compared models that predict human fixation using various criteria. Location-based and distribution-based metrics are the two types of metrics (Bylinskii et al., 2019). These groups of metrics are described as the following.

3.7.1. Location-Based Metrics

Location-based metrics are not dependent on the distribution built Gaussian process, but consider the fixation locations as positive and negative (Jia & Bruce, 2020) (Riche, Duvinage, Mancas, Gosselin, & Dutoit, 2013). Some of these metrics are:

- ✓ ***Area under ROC Curve***: evaluating saliency as a classifier of fixation. It is the metrics used to measure the performance of the saliency map by considering which pixels are fixated or

not at the various thresholds. The saliency map is treated as binary classification under Area under ROC and the curve is drawn out by measuring TP and FP rates under each classifier.

- ✓ ***Normalized Scanpath Saliency (NSS)*:** Measuring the normalized saliency at fixation. It calculated the average normalized fixated location correspondence between ground truth and saliency map by involving the absolute saliency values as part of normalization. NSS may be calculated using a saliency map P and a binary map of fixation locations Q^B :

$$NSS(P, Q^B) = \frac{1}{N}\sum_i \bar{P}_i * Q^B \tag{3.4}$$

Where, $N = \sum_i Q^B_i \ and \ \bar{P} = \frac{p - \mu(P)}{\eth(P)}$ (3.5)

3.7.2. Distribution Based Metrics

A distribution-based metrics deal with both ground truth fixation maps and saliency maps as continuous distributions (Jia & Bruce, 2020) (Riche, Duvinage, Mancas, Gosselin, & Dutoit, 2013). These metrics are:

- ✓ ***Similarity (SIM)***: measuring the intersection between distributions. It computes how two distributions are similar and represent the result as a histogram. That is why it is called histogram intersection. The computation process is done by describing metrics as color-based and content-based image matching. Given a saliency map P and a continuous fixation map Q^D can be calculated as:

$$SIM\ (P, Q^D) = \sum_i min(P_i, Q^D_i) \tag{3.6}$$

Where, $\sum_i P_i = \sum_i Q^D_i = 1$ SIM =1 means the distributions are the same, while SIM = 0 describes no overlap. It's useful for evaluating partial matches in which the ground truth fixation map is represented by a subset of the saliency map.

- ✓ ***Person's Correlation Coefficient (CC)*:** evaluating the linear relationship between distributions. It is the most commonly used measure to test if linear relation exists between two scale variables and it is changing between -1 and +1. -1 means that there is a perfect negative linear relationship, 0 shows no linear relationship and +1 describes the exact positive linear relationship. P and Q^D, the saliency and fixation maps, can be interpreted as random variables to assess the linear connection between them using CC as:

$$CC(P, Q^D) = \frac{\eth(P, Q^D)}{\eth(P) \times \eth(Q^D)} \tag{3.7}$$

Where, $\delta(P;\ Q^D)$ is the covariance of P and Q^D. CC penalizes both false positives and negatives equally.

- ✓ ***Kullback-Leibler divergence (KL)*:** probability interpolation is used in the process of evaluating saliency with KL. It allows us to measure how two distributions are far apart. KL is the dissimilarity metric that measures the gap between two distributions that focuses on how saliency prediction and ground truth are performed as distribution. A saliency map P and a ground truth fixation map Q^D can be taken as input, and computes the loss of information when P is used to guess Q^D:

$$D_{KL}(P \,||\, Q) = \sum_{\mathrm{i}} Q_{\mathrm{i}} \ln(\mathrm{c} + \frac{\mathrm{Qi}}{\mathrm{g}+P\mathrm{i}}) \tag{3.8}$$

Where, Q target distribution, P its approximation, I is each pixel index, c $regularization\ constant$.

- ✓ ***Earth Mover's Distance (EMD)*:** Including spatial distance in evaluation. EMD is the metric used to find the difference between the predicted salient result and ground truth with the help of spatial distance.

3.7.3. Inception Score (IS)

The IS a metric for assessing the quality of image generative models automatically. It employs an Inception v3 Network that has been pre-trained on ImageNet and computes a statistic based on the network's outputs when applied to generated images (Barratt & Sharma, 2018).

$$IS(G) = exp\,(\frac{1}{N}\sum_{\mathrm{i}=1}^{N} D_{KL}\,(P\,(y|x^{\mathrm{i}}\,||p^{-}(y)))) \tag{3.9}$$

Where, where x ∼ pg indicates that x is an image sampled from pg, N the sizes of the data sample, and Dkl is Kullback-Leibler divergence and $p^{-} = \frac{1}{N}\sum_{\mathrm{i}=1}^{N} p(y|x^{\mathrm{i}})$

CHAPTER FOUR

4. PROPOSED ARCHITECTURE OF GAN-BASED VSP WITH CHASPP

4.1. Introduction

The detailed proposed architecture of GAN-based visual saliency prediction with CHASPP is explained in this chapter. Before passing to the implementation the architecture of the system must be defined and explained in detail. Architecture represents the whole details of what is going to be done diagrammatically. This section can be decomposed into three parts. The first part explained the detailed proposed model architecture; the second parts describe an overview of the model learning function in detail. The last part explains how the model is trained and what pseudocode is utilized in training the model.

4.2. Proposed Architecture

The model architecture is developed by combining generative models specifically GAN by adopting SalGAN developed by Pana et.al (Pan et al., 2017) and the CHASPP module proposed by (Liana, Panga, Hanb, & Pan, 2021) to obtain multiple-scale information distributed in different scopes. The architecture contains a module with many convolutional layers at varying dilation rates to collect multi-scale features in series and form an encoder-decoder structure. To predict the area of attention in an image accurately, a different method is used in this architecture like dilated convolution, the combination of ASPP module, and the like.

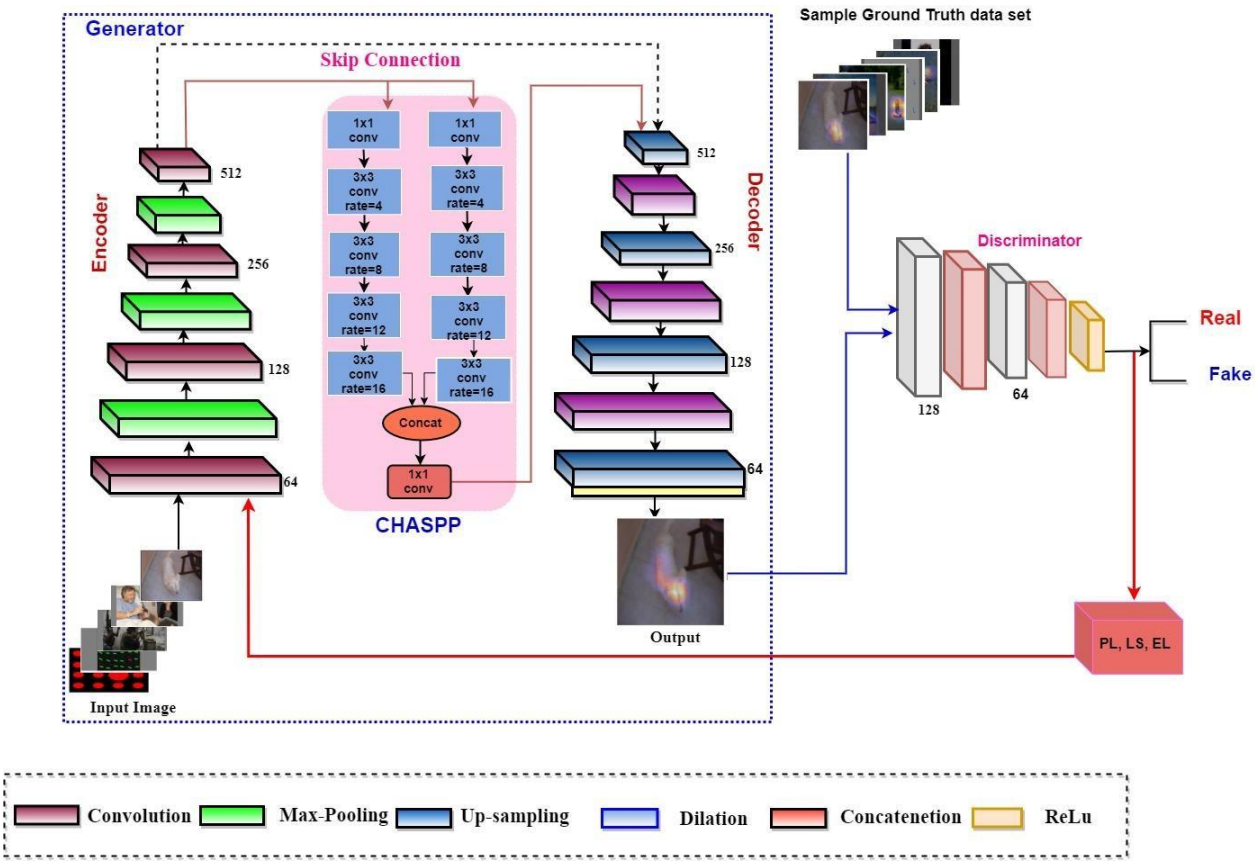


Figure4.1 Proposed Architecture

4.2.1. Generator Architecture

The generator is used to create a realistic-looking image, and it achieves this by training, back-propagating, and updating weights to obtain a realistic synthesized image. The proposed architecture in figure 4.1 contains CHASPP in between encoder and decoder as well as down sampling and up sampling implicitly. The contents of each block can be described as follows.

Down sampling blocks

- ✓ Convolutions
- ✓ Instance normalization
- ✓ ReLu activation function
- ✓ Convolution
- ✓ Instance normalization

CHASPP blocks

- ✓ Convolution
- ✓ Batch normalization
- ✓ Dilated convolution
- ✓ Concatenation

Up sampling block

- ✓ Convolutions
- ✓ Transpose convolution
- ✓ ReLu activation function

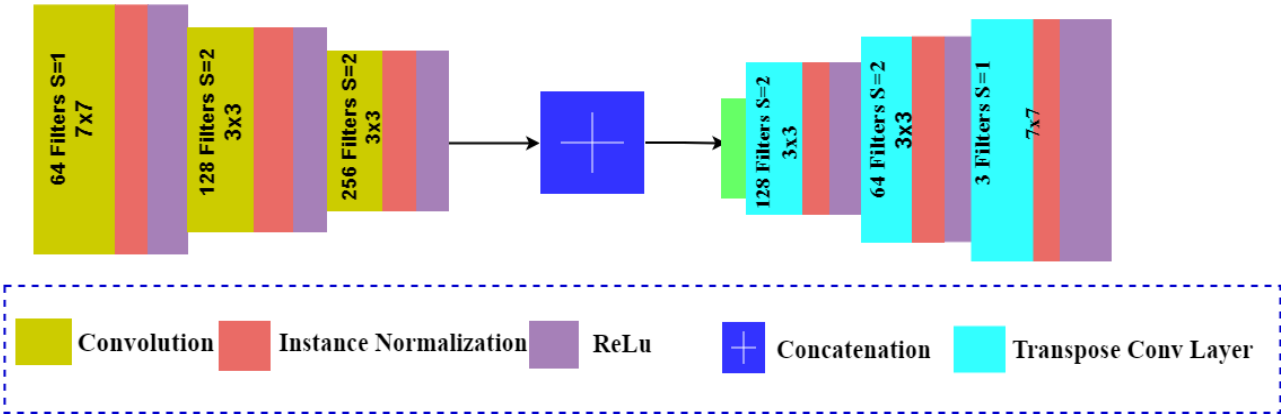


Figure4.2 Generator Architectures

A generator consists of different parts to achieve its functions as depicted in the above diagram. The main parts are:

- ✓ ***Encoder*:** It is the part in which feature extraction takes place through convolutions. To accomplish this process of extracting features, it takes an image as input, as well as the size of the filter window, and the stride size defines how far we move the filter window after each step as we go over the input image to extract features.
- ✓ ***Decoder*:** It reverses the process of an encoder to retain the original size of the input. While the encoder extracts the feature some information may be excluded that can

change either the size or shape of the image. To overcome such an information loss problem decoder is included in the generator to decode the output.

4.2.2. Cascaded Hierarchical Atrous Spatial Pyramid Pooling (CHASPP)

CHASPP captures multi-scale image information by running numerous convolutional layers with varying dilation factors simultaneously(Lian et al., 2021). It demonstrated and improved the performance of the sematic segmentation task by combining two ASPP modules each containing four 3x3 convolution layers lay with cascaded dilation rates of 4, 8, 12, and 16, followed by a 1x1 convolutional layer that can learn nonlinearly combined existing feature maps. And the two ASPPs concatenated each other before element-wise convolution takes place at the last. The main goal of CHASPP is to solve the limitation of ASPP by reducing the number of parameters and using dilation rates that enable coverage of large areas of the given image which help to obtain semantic information distributed in different scopes. It has a large number of sampling ranges as compared with other modules that are applicable to obtain relevant information. It captures multi-scale image information by running numerous convolutional layers with varying dilation factors. The convolutional filters of the output signal are calculated as the following.

$$y[\mathrm{i}] = \sum_{k=1}^{K} x[\mathrm{i} + rk].\, w[k] \qquad (4.1)$$

Where r is the rate at which the Atrous Convolution is dilated, w[k] is the filter of length K, x[i] is the input, and y[i] is the output of a pixel.

To achieves these goals different functions take place in CHASPP as depicted in the following diagram.

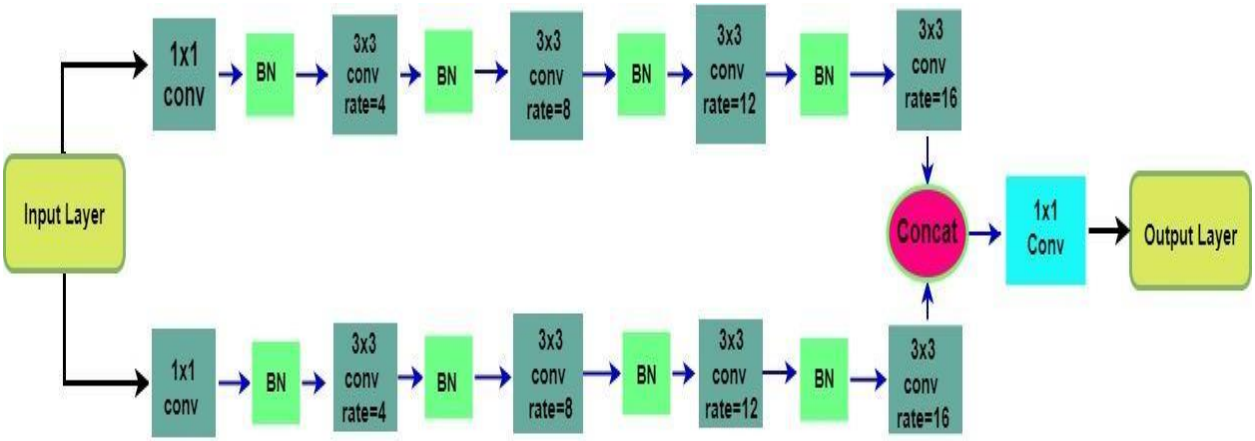


Figure4.3 Implicit CHASPP Architecture

***Dilated Convolution*:** It is a mechanism that increases the kernel (input) by inserting holes between the two consecutive elements involving pixels skipping to cover a large area of an input image. DC offers a wider field of view at the same computational cost and enables us to obtain more information without increasing the number of parameters. The dilated convolution can be calculated as the following formula.

$$(I *_l O)(q) = \sum_{(s+lt=p)} I(z)k(x) \tag{4.2}$$

Where, ***I(z)*** = Input, ***k(x)*** = Applied Filter, ****l*** = *l*-dilated convolution and $(I_{*l0)(q)}$ = Output

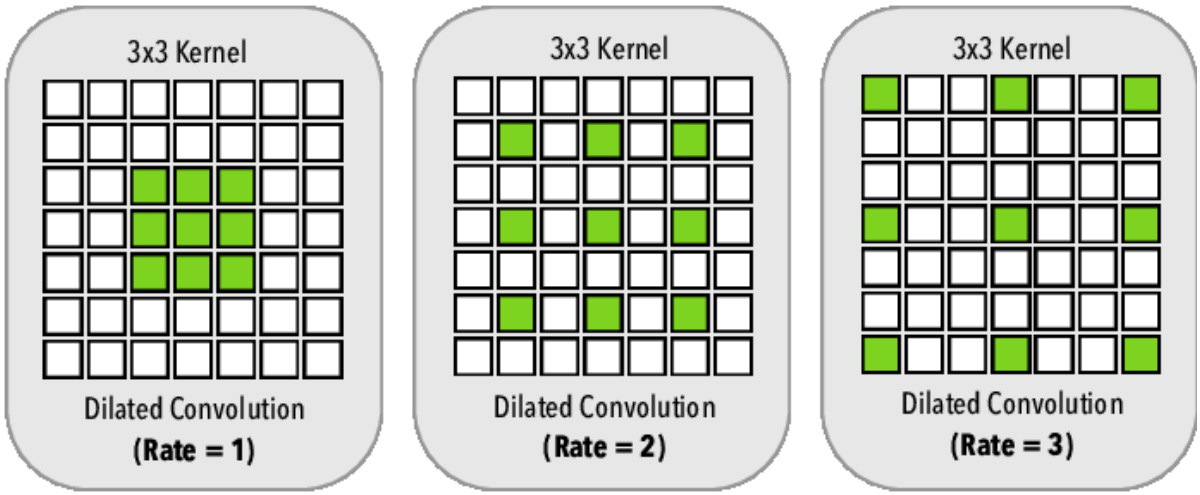


Figure 4.4 Dilated Convolutions

The configuration of the proposed model is summarized in the following table.

Table 4.1 Summarized Configuration of GAN with CHASPP model

	Layer type	*Filter size*
Encoder	Convolution	3x3, 64
	Instance normalization	(*), 64
	Convolution	3x3,128
	Max_pooling	2x2
	Convolution	3x3, 256
	Max_pooling	2x2
CHASPP	Convolution	1x1
	Batch normalization	
	Convolution	3x3, r=8
	Convolution	3x3, r=12, r=16, r=32
	Concatenation	1x1

Decoder	Convolution	3x3, 256
	Transposed convolution	3x3, 256
	Convolution	3x3, 128
	ReLu activation	
	Convolution	3x3, 64

4.3. Model Learning Functions

It is obvious, that a good learning function is a guarantee of yielding better results. The gap between the synthesized image and the real image is assessed by the loss function which is the optimization protest of the GAN. A synthesized image with a smaller loss function value is more plausible to the real samples and gives superior results. The learning function is a sensible factor that has a big impact on the performance of the model during the network training. Different learning functions can be added to this to improve the prediction of the visual salient of an image.

Because this study follows GAN architecture, the learning function like least square GAN loss, perceptual loss, and edge loss is used. The total loss function of this model is calculated as:

$$Ptotal = PLS + Pedge + Ppl \tag{4.3}$$

Least square loss (LS): Least square generative loss is developed by (Mao, Li, Xie, & Lau, 2017) to solve the problem of regular GAN that is employed with a sigmoid cross entropy loss function that is sensitive to vanishing gradient and LSGAN capable to generate high-quality results. The least-square loss is used to reduce the error, which is defined as the total of all squared discrepancies between the true and predicted values. The least-square loss is adopted for discriminator training in LSGAN to minimize the Pearson x^2 divergence with comparing regular GAN. These LS is calculated as:

$$PLSGAN_{(}G_{)} = \frac{1}{z} E_{z\sim p_z(z)}[(D(G(z))-c)^2] \tag{4.4}$$

Where c shows the value that the generator used to trick the discriminator and Pz(z) is the probability of fake data distribution.

***Perceptual loss (PL)*:** Distinguishing two very similar images is a difficult task. The perceptual loss brings the solution for such complex problem that comparing two images that almost look the same but they are different. The difference between the two images may be shifted by one

pixel only. It allowed us to find per pixel loss between actual and predicted images and it measures similarities of images robustly. Perceptual loss uses mean square error (MSE) or mean absolute error (MAE) loss to compare the similarity between synthesizes and actual images. The main goal of this is to preserve the detail of images.

***Edge loss (EL)*:** To distinguish the difference between neighboring pixel values in the image edge detection has a large impact and plays a great role to identify the attention area of an image. It uses Laplacian, which is the sum of second derivatives and is an isotropic filter. Edge is the most important feature that contains plenty of information about the image (Logan, 2005). Edge allows us to decide the character of an image by comparing the distance between pixels. i.e., if the distance between pixels is wide, there may be an edge at that location. This in turn helps in identifying the attention area of an image in visual saliency prediction. Different techniques are used to implement edge detection that detects both horizontal or along the x-axis and vertical or along the y-axis. One of the most common is the Prewitt operator.

$$Pedge = \frac{1}{WH}\sum_{x=1}^{W}\sum_{x=1}^{H}\left(||A_h(real)_{x,y} - A_h(G(pr))_{x,y}|| + ||A_v\ (real)_{x,y} - A_v\ (G(pr))_{x,y}||\right) \quad (4.5)$$

Where A_h and A_v Prewitt operator along horizontal and vertical direction respectively, W and H are weight and height of the input and $G(real)$ and $G(pr)$ are ground truth and generated image respectively.

4.4. Training Pseudocode

The whole configuration and the process of training are aimed at maximizing the similarity between the saliency prediction and ground truth.

Algorithm 1: The proposed architecture is with total loss and discriminator

***Input*:** , h

***Output*:** Æ, ρ

1. Load the batches of data from a dataset ($f \rightarrow h$)
2. Give batch of input to generator network ($G_z \rightarrow$)
3. Train D

   3.1. Classify the batch of input as real and fake: G(z)➛x*

3.2. Calculate discriminator least square loss from predicted saliency and ground truth

3.3. Update discriminator parameter.

4. Train generator

4.1. Predicting saliency from the batch of input ($\rho \rightarrow$)

4.2. Feature loss is computed with the ground truth and the predicted saliency image

4.3. Perceptual loss of predicted saliency is computed

4.4. Edge loss is computed with the ground truth and the predicted saliency images.

4.5. Update generator parameters based on least square, cycle consistency, feature, and edge loss to reduce the loss.

The bold one indicates the additional learning function to be added.

CHAPTER FIVE

5. IMPLEMENTATION OF GAN-BASED VSP WITH CHASPP

5.1. Introduction

This section explains how to put the proposed model into action. Implementation detail includes the setting of the working environment and implementation process of GAN-based VSP with CHASPP. Important code portions were displayed and experiments undertaken are discussed in this section. Complete implementations are outlined in the appendix section.

5.2. Working Environment

- Laptop: The Laptop is utilized for fostering a model design.
  - ✓ Operating system: Windows 10
  - ✓ Processor: Intel ® Core™ i5-7200U CPU @ 2.50GHz 2.70GHz
  - ✓ Graphics: Intel ® Graphics 3000
  - ✓ Primary Memory (RAM): 8.00 GB
  - ✓ System Type: 64-bit Operating System, x64-based Processor
- Desktop: the desktop computer is used for developing a saliency prediction.
  - ✓ Model: DELL OptiPlex 9020
  - ✓ Operating system: Windows 10 pro
  - ✓ Processor: Intel ® Core™ i5-4580 CPU @ 3.29GHz x 4
  - ✓ Graphics: Intel ® HD Graphics 4600
  - ✓ GPU: GeForce RTX 2070 Super 6 GB RAM
  - ✓ Primary Memory (RAM): 8.00 GB
  - ✓ System Type: 64-bit Operating System, x64-based Processor

5.3. Setup Environment

In the process of conducting this research different kinds of development environments and editors are used.

Jupyter notebook: Jupyter Notebook is an open-source, web-based interactive platform for creating and sharing documents with live code, mathematical equations, graphics, maps, charts,

visualizations, and narrative text. Python, PHP, R, C#, and more programming languages are all supported.

Colab: is a free Jupyter notebook environment that runs completely in the cloud. Most importantly, Colab requires no setup and allows you to edit the notebooks you created simultaneously with your team members in the same way that Google Docs documents can be edited.

Visual Studio Code, or VS Code, is a source-code editor for Windows, Linux, and macOS developed by Microsoft that assists us with debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git are just a few of the features available. Users can customize the theme, keyboard shortcuts, and preferences, as well as install extensions to add new features.

5.3.1. Programming Language and Libraries used

Different programming languages and libraries are available now. Among them, the followings are the best that was used in this study to improve the performance in the process of predicting VS.

- ✓ ***Python***: on python, many packages and libraries for scientific computing are available that are used in this study.
- ✓ ***Keras*:** Keras is an open-source software library for artificial neural networks that include a Python interface. It serves as a user interface for TensorFlow. TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML were some of the backends supported by Keras.
- ✓ ***TensorFlow*:** is used for computations and to connect the pre-trained model to the graphs.

5.4. Training Procedures

5.4.1. GAN-based VSP with CHASPP implementation

As described earlier GAN based VSP with CHASPP is based on SalGAN to predict the attractive part of an image from the given input images. GAN based VSP with CHASPP consists encoder and decoder to extract the feature and retain the constructed image. The encoder upscales the size of an image and extracts relevant semantic information. And it utilizes the VGG16 to extract key

features of an image using a pre-trained model to reduce computational complexity for the implementation of the selected baseline work as shown in the following code snippet.

```
from keras.applications.vgg16 import VGG16
vgg=VGG16(include_top=False  weights='imagenet',input_shape=( 256,256,3))
vgg.trainable = False
loss_model = KM.Model(inputs=vgg.input,outputs=vgg.get_layer('block4_conv3').ou
tput)
```

Code snippet VGG16 pre-trained model.

The model's encoder is aVGG16 architecture with 13 convolutional layers that have been pre-trained. To adjust for the omitted down-sampling, all thick levels are removed, and the last three layers are dilated at a rate of two. Finally, the three layers' activations are concatenated. The code below is an example.

```
 def _generator():
  input_image = input(256,256,3)
 #define encoder network
def _encoder(self, images):
  imagenet_mean = tf.constant([103.939, 116.779, 123.68])
        imagenet_mean = tf.reshape(imagenet_mean, [1, 1, 1, 3])
        images -= imagenet_mean
        if self._data_format == "channels_first":
            images = tf.transpose(images, (0, 3, 1, 2))

        layer01 = tf.layers.conv2d(images, 64, 3,
                                   padding="same",
                                   activation=tf.nn.relu,
                                   data_format=self._data_format
                                   name="conv1/conv1_1")
        layer02 = tf.layers.conv2d(layer01, 64, 3,
                                   padding="same",
                                   activation=tf.nn.relu,
                                   data_format=self._data_format,
                                   name="conv1/conv1_2")
```

Code snippet for the encoder

5.4.2. PatchGAN Implementation

As explained in section 3.6 PatchGAN process the input image in the form of an NxN array instead of single scalar output. Since it classifies the NxN patch as real and fake, it increases the probability of getting the attention area of an image. Since it maps from 256x256 to NxN array of outputs x and compares it with each other patchGAN plays a crucial role in identifying the

attention area of an image. The following is the code snippet of the patchGAN discriminator.

```
def patch_discriminator ():
    input_shape = KL.Input (shape = (256,256,3))
    kw = 4
    pad = int (np.ceil((kw-1)/2))
    l1 = tf.pad(input_shape, [[0, 0], [pad, pad], [pad, pad], [0, 0]], mode='REFLECT')
    l1 = KL.Conv2D(64, kernel_size=kw, strides=2, padding='valid')(l1)
    l1 = KL.ReLU(0.2)(l1)
    l2 = tf.pad(l1, [[0,0], [pad, pad],[pad, pad],[0,0]], mode = 'REFLECT')
    l2 = KL.Conv2D(128, kernel_size=kw, strides=2, padding='valid')(l2)
    l2 = InstanceNormalization(axis=3)(l2)
    l2 = KL.ReLU(0.2)(l2)
    l3 = tf.pad(l2, [[0,0],[pad, pad],[pad, pad],[0,0]], mode = 'REFLECT')
    l3 = KL.Conv2D(256, kernel_size=kw, strides=2, padding='valid')(l3)
    l3 = InstanceNormalization(axis=3)(l3)
    l3 = KL.ReLU(0.2)(l3)
    l5 = tf.pad(l3, [[0,0],[pad, pad],[pad, pad],[0,0]], mode = 'REFLECT')
    l5 = KL.Conv2D(512, kernel_size=kw, strides=1, padding='valid')(l5)
    l5 = InstanceNormalization(axis=3)(l5)
    l5 = KL.ReLU(0.2)(l5)
    final = tf.pad(c6, [[0,0],[pad, pad],[pad, pad],[0,0]], mode ='REFLECT')
    final = KL.Conv2D(1, kernel_size=kw, strides=1, padding='valid')(final)
    model = KM.Model(input_shape, final)
    return model
```

5.4.3. CHASPP Implementation

To improve generator training CHASPP was introduced in this study. A pair of five 3x3 convolutions are performed and concatenated before passing to piece-wise convolution (1x1 convolution) in series (cascaded) form. The 3x3 filter size is suitable to get many feature maps from an image. Through all convolution layers, the dilation rate increases starting from 4, 8, 12, 16, and 18 in both pairs. Increased dilation rate enlarges the coverage area of an image. With dilation rate 4 we cover a certain area and compare it with an area covered with dilation rate 8

and go on. Finding their differences and comparing with each other based on metrics. If they are highly different, it considers the area as the attention part, unless going on compare all the distance between pixels throughout the images. The major function that we used in the CHASPP module is CategoricalScore which is used to find the attention area of an image by randomly changing the parameters. Since CHASPP has fewer parameters, it updates the parameter frequently to access more information about the image. The ReLu activations are then integrated with the overall scene context to form a single tensor. This shows how the CHASPP module is applied and improves the prediction of visual saliency. The code snippet is as follows:

```
def _chaspp(self, features):
          branch1 = tf.layers.conv2d(features, 256, 1,
                                     padding="same",
                                     activation=tf.nn.relu,
                                     data_format=self._data_format,
                                     name="chaspp/conv1_1")&&

        tf.layers.conv2d(features, 256, 3,
                                     padding="same"
                                     activation=tf.nn.relu,
                                     dilation_rate=4,
                                     data_format=self._data_format,
                                     name="chaspp/conv1_2")

        branch2 = tf.layers.conv2d(features, 256, 3,
                                     padding="same",
                                     activation=tf.nn.relu,
                                     dilation_rate=8,
                                     data_format=self._data_format,
                                     name="chaspp/conv1_3")&&
                  tf.layers.conv2d(features, 256, 3,
                                     padding="same",
                                     activation=tf.nn.relu,
                                     dilation_rate=12,
                                     data_format=self._data_format,
                                     name="chaspp/conv1_4")
```

Code snippet CHASPP implementation

5.4.4. Decoder Implementation

The decoder model applies a series of 3 up sampling blocks that each performs bilinear up sampling followed by a 3x3 convolution to avoid checkerboard artifacts in the image space. Unlike all other layers, the output of the model is modified by a sigmoid activation function.

```
def _decoder(self  features):
       shape = tf.shape(features)

        layer1 = self._upsample(features, shape, 2)
        layer2 = tf.layers.conv2d(layer1, 128, 3,
                                  padding="same",
                                  activation=tf.nn.relu,
                                  data_format=self._data_format,
                                  name="decoder/conv1")
        shape = tf.shape(layer2)
        layer3 = self._upsample(layer2, shape, 2)
        layer4 = tf.layers.conv2d(layer3, 64, 3,
                                  padding="same",
                                  activation=tf.nn.relu,
                                  data_format=self._data_format,
                                  name="decoder/conv2")
```

5.4.5. Perceptual Loss Implementation

A function named perceptual loss is designed to measure and return the feature value of the predicted and real image. The perceptual loss function first extracts the feature map of the predicted (fake) and real images. The L2 distance between the feature maps of the real and predicted images can then be used to calculate feature loss between paired real and predicted images. The loss was then used to change the network's weight.

```
#Computes feature loss for real and generated images
@tf.function()
def perceptual_loss(fake, real):
    vggX = loss_model(real)
    vggY = loss_model(fake)
    return tf.reduce_mean(tf.square(vggY-vggX))
```

Code snippet Feature Loss Function Implementation

5.4.6. Edge Loss Implementation

Edge loss is a measurement of the difference in the edge between generated and ground truth images, which aids the generator in producing a correct predicted image. Prewitt kernel operators can be used as edge extractors along the horizontal and vertical axes to yield edge loss. There are two ways to calculate edge loss. Prewitt kernel operators can extract first edge information from both a synthetic and a real image along the horizontal and vertical axes. The edge loss between real and synthetic picture pairings is then calculated by L1 distance as the following snippet deployed.

```
#Computes edge loss for real and generated images
def Edge_loss(fake, real):
    real_edge = tf.image.prewitt_edges(real)
    fake_edge = tf.image.prewitt_edges(fake)
    real_y = real_edge[0, :, :, :, 0]
    real_x = real_edge[0, :, :, :, 1]
    fake_y = fake_edge[0, :, :, :, 0]
    fake_x = fake_edge[0, :, :, :, 1]
    edge_loss = tf.reduce_sum(tf.abs( real_y - fake_y )) +
    tf.reduce_sum(tf.abs(real_x- fake_x))
    return edge_loss
```

Code snippet, Edge Loss Implementation

5.5. Hyperparameter Configuration

Before starting the learning process, hyperparameters configurations must be set. These parameters are batch size, epochs, optimization, learning rate, dilation rate, and weights. The configurations of these hyperparameters directly influence how the models are trained. The main objective of the Hyperparameters is to change the network attributes that give the best result by minimizing the losses (Panda, 2020). To achieve this aim Adam optimization is used in this research since it comes with the best properties of both AdaGrad and RMSProp. And it also requires less memory and is efficient since it is based on first order gradient as well as suitable for the problem with very noisy and that is large in terms of data/parameters. Dilation rate is also used in this study to obtain the sparse distribution of information in different scopes. Most relevant information can be collected by changing the rates to circulate throughout all pixels of an image which are very crucial in the prediction process. The learning rate approaches

0.0000679 after 150 epochs to stabilize the learning process of the model. Dilation rates are changed by the multiple of two starting with $\gamma = 4, 8, 16, ...$ is used in this study.

Table 5.1 Hyperparameter Configuration

Hyperparameter	***Value***
Batch size	*1*
Epoch	*200*
Optimization	*Adam*
Learning rate	*0.001*
α	*0.5*
γ	*4, 8, 16, 18 (through iteration)*

5.6. Experimental Class

To improve VSP the whole process is categorized into different experimental classes based on the factors that affect the VSP process and results in a different model. These classifications are helped to identify the factor affecting VSP. For this study, four experimental classes are conducted. In the first class using SalGAN used EfficientNet-B7 as a feature extractor and try to generate a salient region from the given input with the ASPP module used as baseline work. Edge loss and perceptual loss are also other factors added to SalGAN showed its effect on the result obtained from the network in the second experimental class. Third, CHASPP is incorporated to the model to obtain information distributed in sparse scope instead of ASPP made the result more accurate. Finally, both edge loss and the CHASPP module are added together in the fourth experimental class with SalGAN which comes with the advantages of the two. These all-experimental classes are summarized in the following table.

Table 1.2 Experimental Classes

Notation	*Experiment*	*Dataset*	*Model*
salGAN+ ASPP	Baseline work	CAT2000, MIT1003, PASCAL-S, DUTOMRON	VGG16
salGAN+e	Initial work of proposed model with edge loss	CAT2000, MIT1003, PASCAL-S, DUTOMRON	EfficientNet-B7
salGAN+CHASPP	VSP with CHASPP	CAT2000, MIT1003, PASCAL-S, DUTOMRON	EfficientNet-B7
salGAN+CHASPP+e	VSP with CHASPP and edge loss	CAT2000, MIT1003, PASCAL-S, DUTOMRON	EfficientNet-B7

5.7. CHASPP vs. ASPP

CHASPP overcomes the problem of ASPP explained in section 1.3 while we use ASPP we phase the problem by requiring a larger number of parameters and more memory for implementation. CHASPP handles this problem by reducing the number of parameters. But in the process of cascaded execution, more time may be consumed. The following table shows the comparison of CHASPP and ASPP in terms of consumed time and the number of parameters tested on the MIT1003 dataset.

Table 5.3 Comparison of ASPP and CHASPP in terms of consuming time

Module	*Overall training time/hrs.*	*Overall testing time/hrs.*	*# Of parameters in millions*
ASPP	44.40’	16.30’	59.3
CHASPP	45.06’	18.00’	33.5

As we observed from table 5.3, the overall training time consumed by CHASPP is nearly the same as ASPP, this is because the number of parameters has been reduced. The difference is not as big as compared with the observed result discussed in section 6.3.

CHAPTER SIX

6. RESULTS AND DISCUSSIONS

6.1. Results and Discussions

This chapter gives the outcomes of the experiments after discussing the details of the implementations and experiments to be undertaken. The performance of the proposed model was evaluated using the evaluation measures discussed in the third chapter. In addition, the suggested model's performance is compared to those of current salGAN designs.

6.2. Visual Saliency Prediction

VSP is a process of taking images as input and identifying attention to that image. This research uses a unique training experiment to improve the subjective and quantitative results of comparing baselines on four independent datasets used to train and evaluate the proposed model. These datasets are CAT2000, MIT1003, DUTOMRON, and PASCAL-S. After training the model, the result was tested for its ability to predict the correct attention part of an image. To measure the similarity between the result and the ground truth test set was used. Table 5 shows the four conducted experiments. All four experiments were conducted on the same hardware and software, hyperparameters, and dataset configuration as described in section 3.9. All models are trained for 200 epochs with a batch size of 1 and it takes around two days.

6.3. Experimental Results

Figure 6.1 below showed the results obtained from baseline work, SalGAN with ASPP which used the VGG16 pre-trained model. As shown in the figure it predicts the area of images that attract attention but still, there is a huge difference between the obtained result and ground truth when compared.

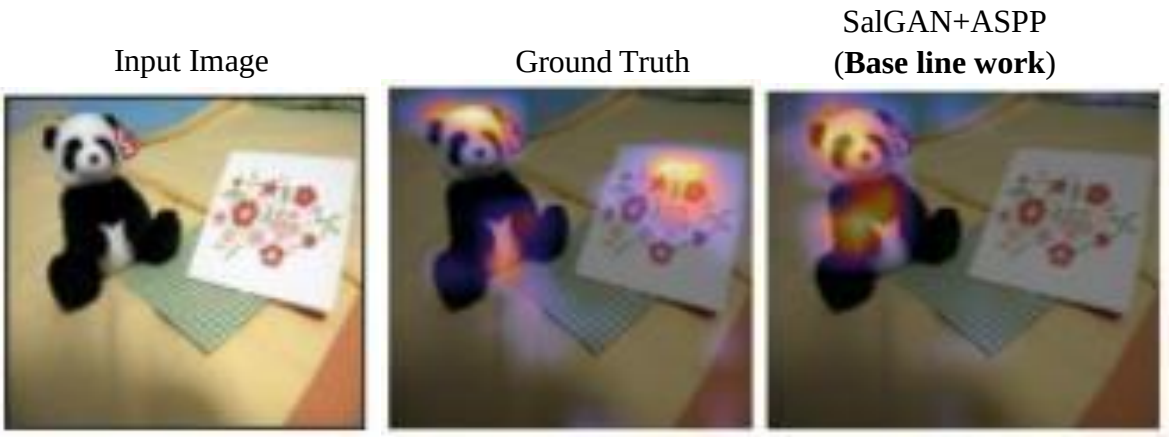


Figure 6.1 Sample test for SalGAN with ASPP on MIT1003 dataset.

The following figure 6.2 describes the result obtained by testing SalGAN+e (SalGAN with edge loss, and Perceptual loss) and using probability density functions (PDF) like color, and intensity to identify the odd part of an image and EfficientNet-B7 used as feature extractor to improve the result shown above. As seen from the image these two loss function (edge loss and perceptual) didn't cover all relevant part of an image that included as the attention part in figure 6.1 above. This implied that the two factors have not much effect on VSP without the module. Here, there is a gap between obtained result and the ground truth.

Figure 6.2 Sample test for SalGAN with edge loss on MIT1003 dataset.

Figure 6.3 shows the result of SalGAN with CHASPP. It depicts how this experiment improves the above two experiments when compared with them. As seen from the figure 6.3, the result is closer to the ground truth by omitting irrelevant parts that are included in SalGAN+ASPP and SalGAN+e. Most computer vision tasks usually revolve around low-level features like contrast, edges, and color only(Le, n.d.). This is the main challenges of VSP in CV application solved by the help of CHASPP module. It divides the images into sections based on both low-level and high-level features based on dilation rates and finds the difference between them consecutively.

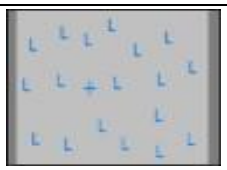
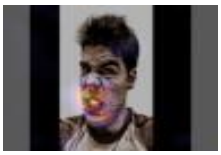
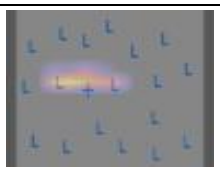
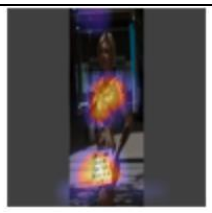
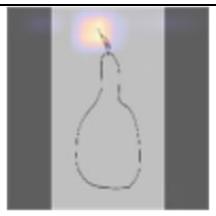
If the difference is high, it considers and marks the area as the attention part and goes throughout all the pixels and paints it as described in the following result.

Figure 6.3 Sample test for SalGAN with CHASPP on MIT1003 dataset.

Figure 6.4 depicts the result on SalGAN+CHASPP+e. Different factors are added in each of the experimental classes to improve the results. The addition of the CHASPP to the previous work has shown an improvement in the image being predicted. The above three experiments improve something that is an addition to the previous classes. Figure 6.4 shows the most accurate results than the other by combining the edge loss, perceptual loss, and CHASPP with SalGAN. In processing the image, at the time getting the edge it changes abruptly and thus helps us to obtain information from attention part of an image. Add the two factors alone to SalGAN have reduces the performance of the model on MIT1003 dataset as we have seen figure 6.2 above. But, as the diagram shows the result obtained from the experimental class which combined both edge loss, perceptual loss and CHASPP module is closer to the ground truth than others. This explained that how much these two factors are affecting VSP with CHASPP module.

Figure 6.4 Sample test for SalGAN with CHASPP on MIT1003 dataset

Table 6.1 Sample Results from all datasets using SalGAN with CHASPP model.

	CAT2000	***MIT1003***	***PASCAL-S***	***DUTOMRON***
II				
GT				
R				
II				
GT				
R				

Where, II=Input Image, GT=Ground Truth, and R=Result.

6.4. Evaluation Metrics

6.4.1. Objective Evaluation Metrics

Many objective evaluation metrics are utilized in the process of evaluating the execution of the model explained in section 3.6 Tables 6.2 to 6.5 summarize the result of experimented class to show the factors affecting VSP by measuring the difference between the ground truth and the predicted one. These metrics and inception scores are computed by using 4,242 images from the dataset CAT2000, MIT1003, DUTOMRON, and PASCAL_S test sets. The higher values of AUC-J, SIM, AUC-B, sAUC, CC, NSS, and the lower values of EMD and KLD show that the prediction is closer to the ground truth.

Table 6.2 Comparison of the quantitative scores of several models on the CAT2000 dataset

No	***Experiment classes***	***Evaluation Metrics***					
		AUC-J	***AUC-B***	***sAUC***	***SIM***	***NSS***	***KLD***
1	SalGAN+ASPP	0.788	0.632	0.713	0.502	2.363	0.888
2	SalGAN+e	0.740	0.529	0.632	0.468	2.346	0.890
3	SalGAN+CHASPP	0.814	0.738	0.713	**0.615**	**2.395**	0.818
4	SalGAN+CHASPP+e	**0.826**	**0.762**	**0.728**	0.603	2.347	**0.815**

Table 6.3 Comparison of the quantitative scores of several models on the MIT1003 dataset

No	***Experiment classes***	***Evaluation Metrics***					
		AUC-J	***AUC-B***	***sAUC***	***SIM***	***NSS***	***KLD***
1	SalGAN+ASPP	0.789	0.672	0.653	0.602	2.563	0.839
2	SalGAN+e	0.775	0.681	0.632	0.582	2.546	0.867
3	SalGAN+CHASPP	0.806	0.730	0.680	0.615	**2.675**	0.818
4	SalGAN+CHASPP+e	**0.834**	**0.731**	**0.682**	**0.695**	2.617	**0.809**

Table 6.4 Comparison of the quantitative scores of several models on the DUTOMRON dataset

No	Experiment classes	Evaluation Metrics					
		AUC-J	AUC-B	sAUC	SIM	NSS	KLD
1	*SalGAN+ASPP*	*0.808*	*0.689*	*0.713*	*0.582*	*2.363*	*0.888*
2	*SalGAN+e*	*0.817*	*0.729*	*0.762*	*0.608*	*2.346*	*0.840*
3	*SalGAN+CHASPP*	*0.824*	*0.738*	*0.773*	*0.615*	***2.395***	*0.818*
4	*SalGAN+CHASPP+e*	***0.836***	***0.782***	***0.788***	***0.623***	*2.347*	***0.815***

Table 6.5 Comparison of the quantitative scores of several models on the PASCAL-S dataset

No	Experiment classes	Evaluation Metrics					
		AUC-J	AUC-B	sAUC	SIM	NSS	KLD
1	*SalGAN+ASPP*	*0.764*	*0.699*	*0.743*	*0.682*	*2.563*	*0.863*
2	*SalGAN+e*	*0.752*	*0.689*	*0.738*	*0.668*	*2.486*	*0.880*
3	*SalGAN+CHASPP*	*0.824*	*0.738*	*0.783*	*0.695*	*2.695*	***0.802***
4	*SalGAN+CHASPP+e*	***0.836***	***0.782***	***0.786***	***0.703***	***2.704***	*0.815*

Table 6.6 Inception Score (IS) on CAT2000 Dataset

No	Experimental class	Model	Inception Score (IS)
1	SalGAN+ASPP	VGG16	3.356 ± 0.04
2	SalGAN+e	EfficientNet-B7	3.521 ± 0.21
3	SalGAN+CHASPP	EfficientNet-B7	3.748 ± 0.03
4	**SalGAN+CHASPP+e**	EfficientNet-B7	**3.851 ± 0.01**

The bold values represent the best experimental results.

The quantitative experimental result shows SalGAN+CHASPP+e score best result as compared with another three experimental classes on the four dataset. Since many datasets and evaluation metrics are used in these experiments there may be variations in the obtained result on the different datasets as the above table 6.2 through table 6.6 described. In SalGAN +e, simply spatial pyramid pooling is substituted by edge loss and perceptual loss (represented by e in this study) function as one factor and it scores small results on three dataset. This is because of the sparse distribution of information in an image that is not discovered while searching for an attention area at the time of training and focuses only on low-level features. By increasing the number of sampling ranges which helped the model understood the pattern of an image. In SalGAN+CHASPP the result grows at a high rate than SalGAN+ASPP and SalGAN+e in the experiments. This magnifies the role of CHASPP in the process of identifying and predicting the attractive part of an image in VSP. Lastly, CHASPP, edge loss, and perceptual loss are combined with CHASPP and improved the performances as well as revealed the impacts of perceptual loss and edge loss in VSP evaluated through different evaluation metrics.

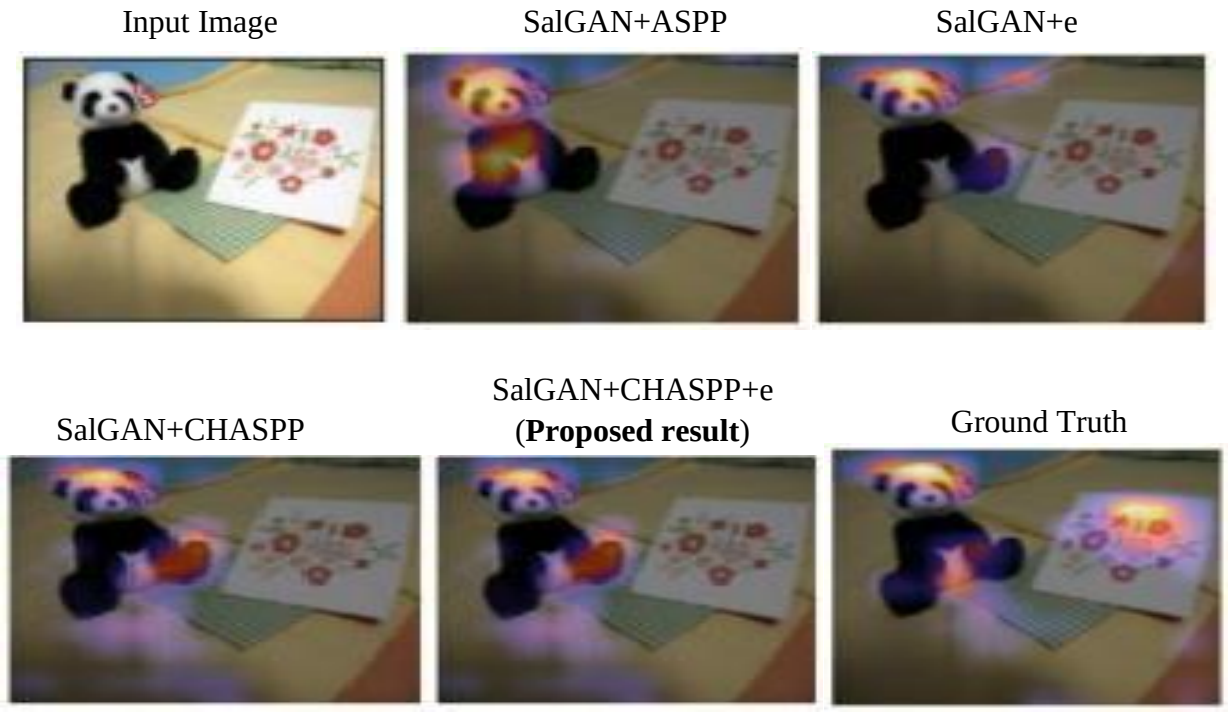


Figure 6.5 Model comparison MIT1003 dataset

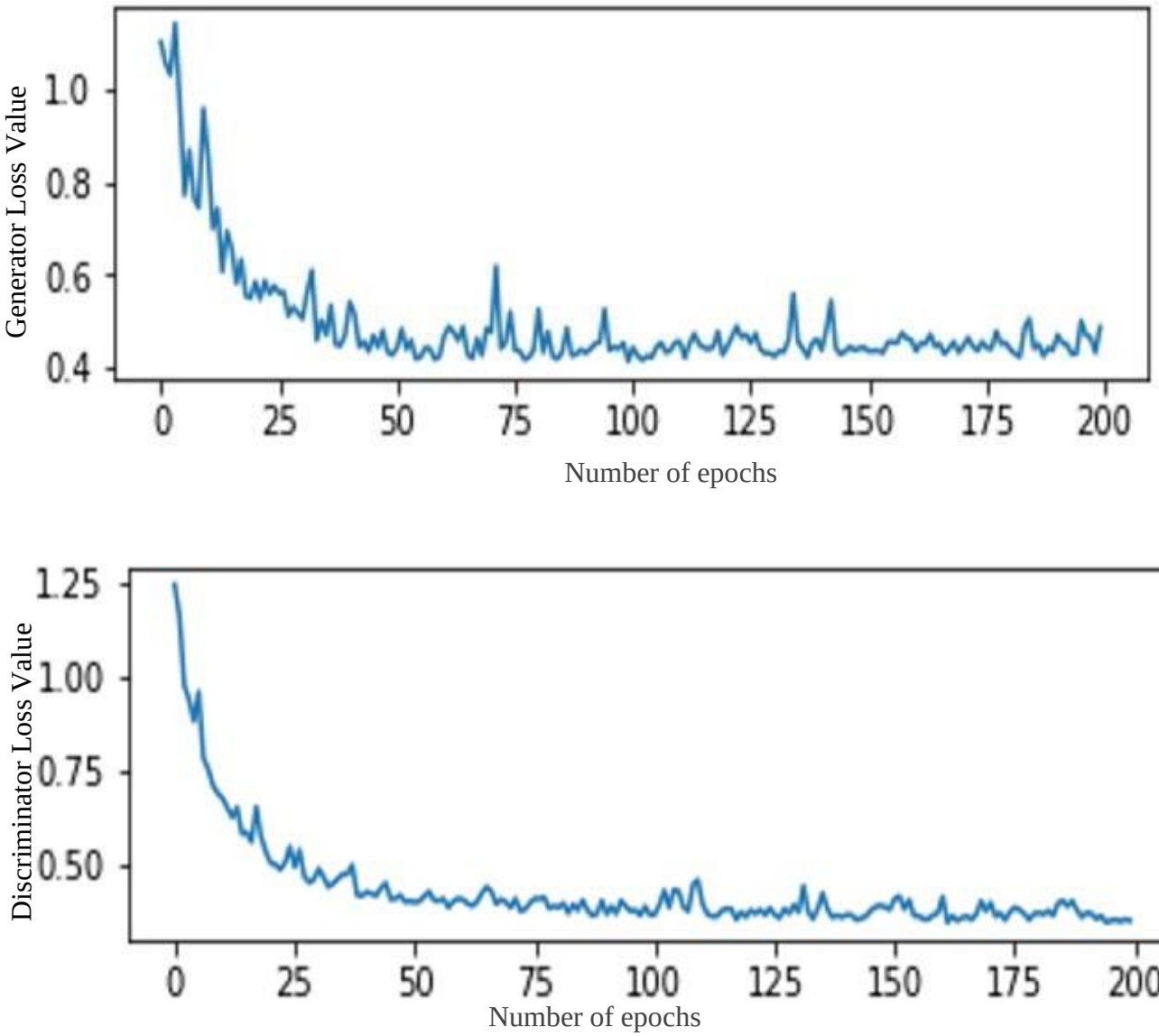


Figure 6.6 SalGAN+CHASPP+e training loss

6.5. Discussion

6.5.1. Discussion on Different Metrics Results

Different measures explained in section 3.6 were employed to assess the model's quality and performance, some of them were Kullback–Leibler divergence (KLD), histogram intersection (SIM), normalized scan path saliency (NSS), and three variants of the area under ROC curve (AUC-Judd, AUC-Borji, shuffled AUC). These metrics have been measured for all experiments as described in Tables 8, 9 and 10 above. The result of the proposed model solution is the best one as observed from the figure 6.5 on MIT1003 the common datasets used in VSP.

Therefore, the improvement on the metrics described in the above tables implied that making the atrous spatial pyramid pooling cascade and adding edge loss to the model is crucial. This implies the series form of convolution, edge, and perceptual loss are the most influential factors that are necessary for the improvement of the result and the inception score result scored on CAT2000 data are described as follow.

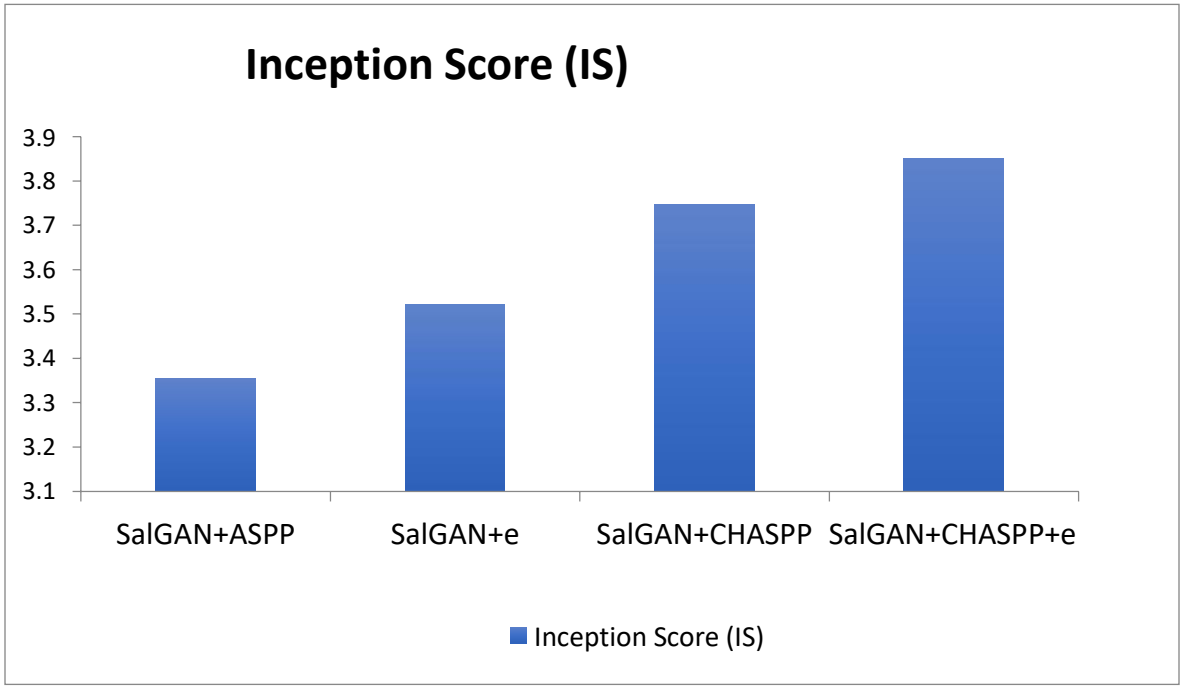


Figure 6.7 Inception Score (IS) on CAT2000 Dataset

7. CONCLUSION AND FUTURE WORKS

7.1. Conclusion

Visual saliency prediction is one of the uses of GAN. Visual attention refers to a collection of cognitive operations that select significant information from cluttered visual environments while filtering out irrelevant data. This can be done with the help of generative models specifically GAN which assists us to identify which part contains relevant information and/or not. Saliency prediction technology is widely used in various visual tasks, such as object detection, object tracking, image segmentation, and image retrieval. However, the original GAN yields do not have a good result in visual saliency prediction. To improve the result CHASPP module and edge loss function are added to SalGAN in this study and score the best result as described in section 6.3 and tables 6.2, 6.3 & 6.4.

This research introduces visual saliency prediction based on GAN incorporating CHASPP to improve the process of predicting part of the images correctly that attract human attention. To do this CHASPP module that covers the distribution of information in large scope and edge loss functions are added to the baseline work. The purpose of this study was to predict the accurate area of an image with as small computational cost and time as much as possible. This can be done with the help of the CHASPP module, perceptual loss, and edge loss function that plays a great role in the improvement of the prediction of attention area. This addition to SalGAN has indeed the modifications to make SalGAN+CHASPP+e predict the accurate place of an image.

The experiment demonstrated the architecture's logical approach, which included the use of a mathematical expression to improve the prediction. As shown in the result and appendices section, the VSP result employing the SalGAN+CHASPP+e section had improved visual saliency prediction and performance with several evaluation metrics (KLD, CC, EMD, SIM, NSS, and three variants of area under the ROC curve (AUC-Judd, AUC-Borji, shuffled AUC)).

In total, four separate experiments were conducted in this study for comparison. The first experiment looked at the effect of ASPP with GAN to improve the identifying part of image odd from its neighbor. The second experiment showed the effect of adding the perceptual loss and edge loss function to the model, which assisted us to get the correct location of the attractive position of an image than the first experiment because it adds stability to the generator's training.

The third experiment that substitutes ASPP with CHASPP, obtains more semantic information distributed in a sparse location in the prediction of attention area. The other experiment is the final proposed solution SalGAN+CHASPP+e compared with the second and third experiment and improve the problem of identifying the attention area of an image in the three experimental classes. Merging the CHASPP module and edge loss brings stability to the network throughout the entire process of finding AA.

With the inclusion of different restrictions to the learning function that it was trained on, the GAN-based VSP with CHASPP (proposed solution) has demonstrated significant gains. Finally, it's reasonable to conclude that adding CHASPP, perceptual loss, and edge loss to the predicting network would improve performance since they allow the network to focus on more critical futures, utilize the feature, and retain image information.

7.2. Future Work

The goal of this study was to develop a model for recognizing visual saliency prediction by detecting salient information from an image. However, the obtained result demonstrates infancy stage performance, which requires further investigation, particularly for a real-time procedure. Edge loss is critical component of the model that has a significant impact on network performance. The Prewitt edge detector was used for computational complexity because it is based on the first-order derivative. As a result, replacing the Prewitt edge detector with another edge detector can improve network performance.

Since only a static dataset is used in this study, we recommend using a dynamic dataset that makes visual saliency prediction more applicable, and video saliency prediction is not included in this study because of the tight schedule and limitation of resources. To expand the application of visual saliency prediction in a different sector, we recommend the next researcher add video saliency prediction and activity recognition.

***

Special Acknowledgement

This research was sponsored by Adama Science and Technology University with a grant number

ASTU/SM-R/560/22

Adama, Ethiopia

REFERENCES

Arjovsky, M., Chintala, S., & Bottou, L. (2017). *Wasserstein GAN*.

Assens, M., Giro-i-Nieto, X., McGuinness, K., & O'Connor, N. E. (2019). PathGAN: Visual scanpath prediction with generative adversarial networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11133 LNCS*, 406–422. https://doi.org/10.1007/978-3-030-11021-5_25

B, S. J., & Kamath, S. S. (2017). *Saliency Prediction for Visual Regions*. 48–60. https://doi.org/10.1007/978-3-319-56687-0

Barratt, S., & Sharma, R. (2018). *A Note on the Inception Score*. http://arxiv.org/abs/1801.01973

Borji, A. (2018). *Saliency Prediction in the Deep Learning Era: Successes, Limitations, and Future Challenges*. 1–44.

Borji, A., & Itti, L. (2015). *CAT2000: A Large Scale Fixation Dataset for Boosting Saliency Research*.

Bruckert, A., Tavakoli, H. R., Liu, Z., Christie, M., & Le Meur, O. (2021). Deep saliency models: The quest for the loss function. *Neurocomputing*, *453*, 693–704. https://doi.org/10.1016/j.neucom.2020.06.131

Bylinskii, Z., Judd, T., Oliva, A., Torralba, A., & Durand, F. (2019). What Do Different Evaluation Metrics Tell Us about Saliency Models? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(3), 740–757. https://doi.org/10.1109/TPAMI.2018.2815601

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 1800–1807. https://doi.org/10.1109/CVPR.2017.195

Christiaan Klink, P., Jentgens, P., & Lorteije, J. A. M. (2014). Priority maps explain the roles of value, Attention, And salience in goal-oriented behavior. *Journal of Neuroscience*, *34*(42), 13867–13869. https://doi.org/10.1523/JNEUROSCI.3249-14.2014

Ghariba, B., Shehata, M. S., & McGuire, P. (2019). Visual saliency prediction based on deep learning. *Information (Switzerland)*, *10*(8). https://doi.org/10.3390/info10080257

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144. https://doi.org/10.1145/3422622

Hegadi, R. S. (2010). Image Processing: Research Opportunities and Challenges. *National Seminar on Research in Computers*, *March 2015*, 13–25. https://www.researchgate.net/publication/274247823

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

Kerkouri, M. A., Tliba, M., Chetouani, A., & Harba, R. (2021). Salypath: a Deep-Based Architecture for Visual Attention Prediction. *Proceedings - International Conference on Image Processing, ICIP*, *2021-Septe*, 1464–1468. https://doi.org/10.1109/ICIP42928.2021.9506295

Kroner, A., Senden, M., Driessens, K., & Goebel, R. (2020). Contextual encoder–decoder network for visual saliency prediction. *Neural Networks*, *129*, 261–270. https://doi.org/10.1016/j.neunet.2020.05.004

Le, A. (n.d.). *Predicting Visual Saliency: Where Do People Look?*

Lian, X., Pang, Y., Han, J., & Pan, J. (2021). Cascaded hierarchical atrous spatial pyramid pooling module for semantic segmentation. *Pattern Recognition*, *110*. https://doi.org/10.1016/j.patcog.2020.107622

Liu, N., Zhang, N., Wan, K., Shao, L., & Han, J. (2021). Visual Saliency Transformer. *Proceedings of the IEEE International Conference on Computer Vision*, 4702–4712. https://doi.org/10.1109/ICCV48922.2021.00468

Logan, D. (2005). *A review of literature on cicadas*. *8*(July), 1–31.

Mirza, M., & Osindero, S. (2014). *Conditional Generative Adversarial Nets*. 1–7.

Pan, J., Ferrer, C. C., McGuinness, K., O'Connor, N. E., Torres, J., Sayrol, E., & Giro-i-Nieto, X. (2017). *SalGAN: Visual Saliency Prediction with Generative Adversarial Networks*. 1–9.

Panda, B. (2020). *A survey on application of Population Based Algorithm on Hyperparameter Selection*. *October 2019*. https://doi.org/10.13140/RG.2.2.11820.21128

Phan, H., McLoughlin, I. V., Pham, L., Chen, O. Y., Koch, P., De Vos, M., & Mertins, A. (2020). Improving GANs for Speech Enhancement. *IEEE Signal Processing Letters*, *27*, 1700–1704. https://doi.org/10.1109/LSP.2020.3025020

Prior, N. (2020). North Dakota State University. *Graduate Study in Criminology and Criminal Justice*, *October*, 212–213. https://doi.org/10.4324/9781315721606-101

Riche, N., Duvinage, M., & Mancas, M. (2013). *A study of parameters affecting visual saliency assessment A study of parameters affecting visual saliency assessment*. *May 2014*.

Sun, Y., Zhao, M., Hu, K., & Fan, S. (2022). Visual saliency prediction using multi-scale attention gated network. *Multimedia Systems*, *28*(1), 131–139. https://doi.org/10.1007/s00530-021-00796-4

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 10691–10700.

Wang, W., & Shen, J. (2018). Deep Visual Attention Prediction. *IEEE Transactions on Image Processing*, *27*(5), 2368–2378. https://doi.org/10.1109/TIP.2017.2787612

Yan, F., Chen, C., Xiao, P., Qi, S., Wang, Z., & Xiao, R. (2022). Review of visual saliency prediction: Development process from neurobiological basis to deep models. *Applied Sciences (Switzerland)*, *12*(1). https://doi.org/10.3390/app12010309

Zhao, T., & Wu, X. (2019). Pyramid feature attention network for saliency detection.

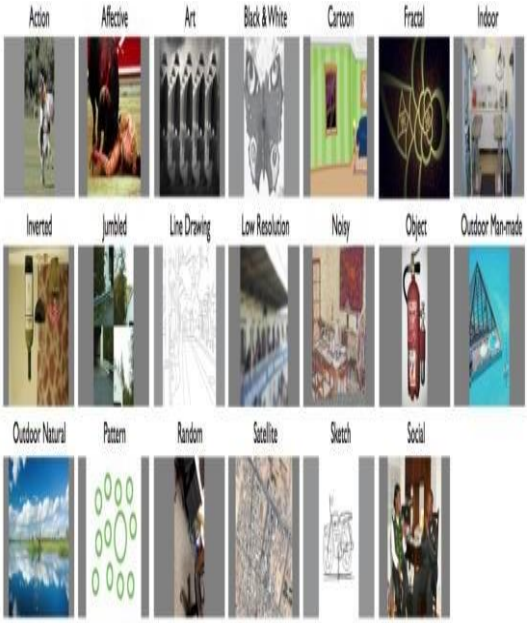
Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, *2019-June*, 3080–3089. https://doi.org/10.1109/CVPR.2019.00320

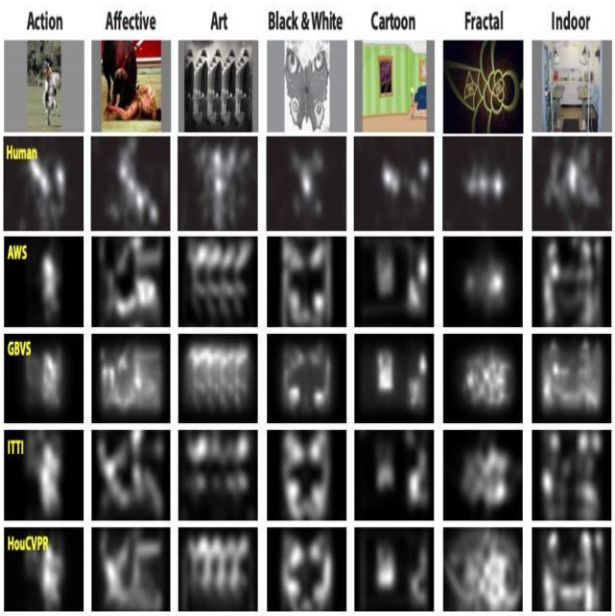
Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 2242–2251. https://doi.org/10.1109/ICCV.2017.244

APPENDICES

Appendix A Sample Dataset

Appendix A.1: Sample Dataset from CAT2000

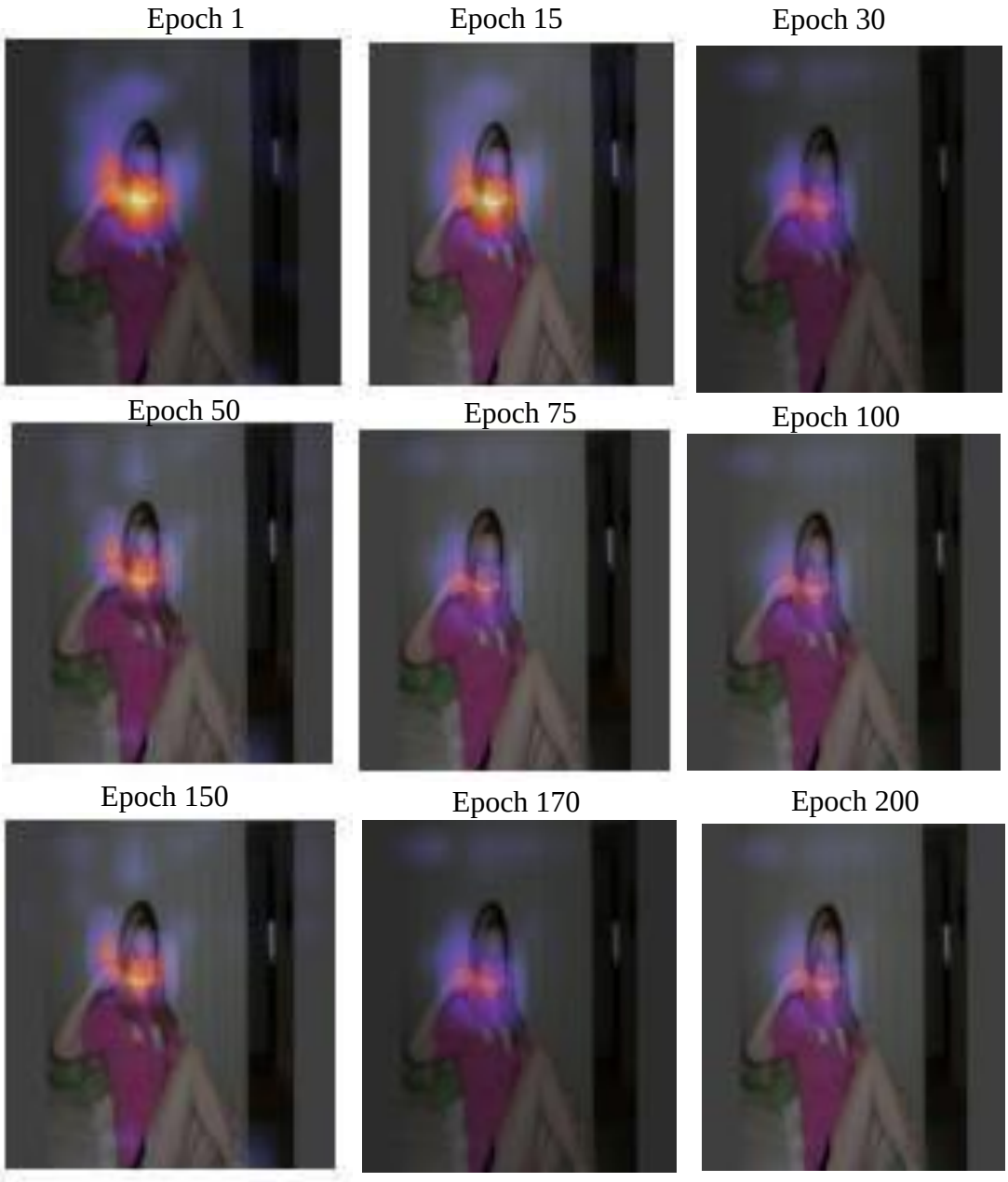




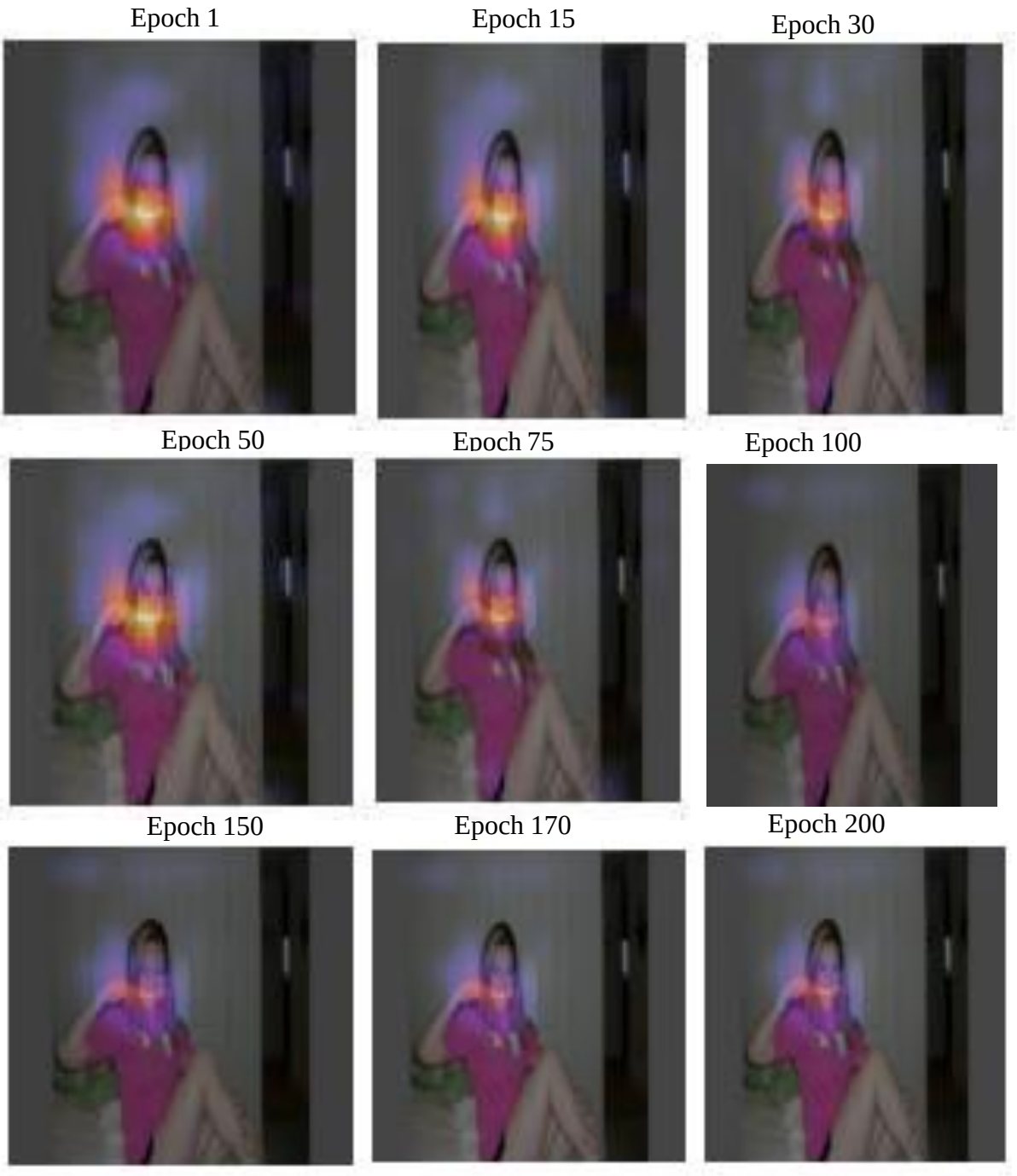
Appendix A.2: Sample Dataset from MIT1003

Appendix B: Results on Different Epochs

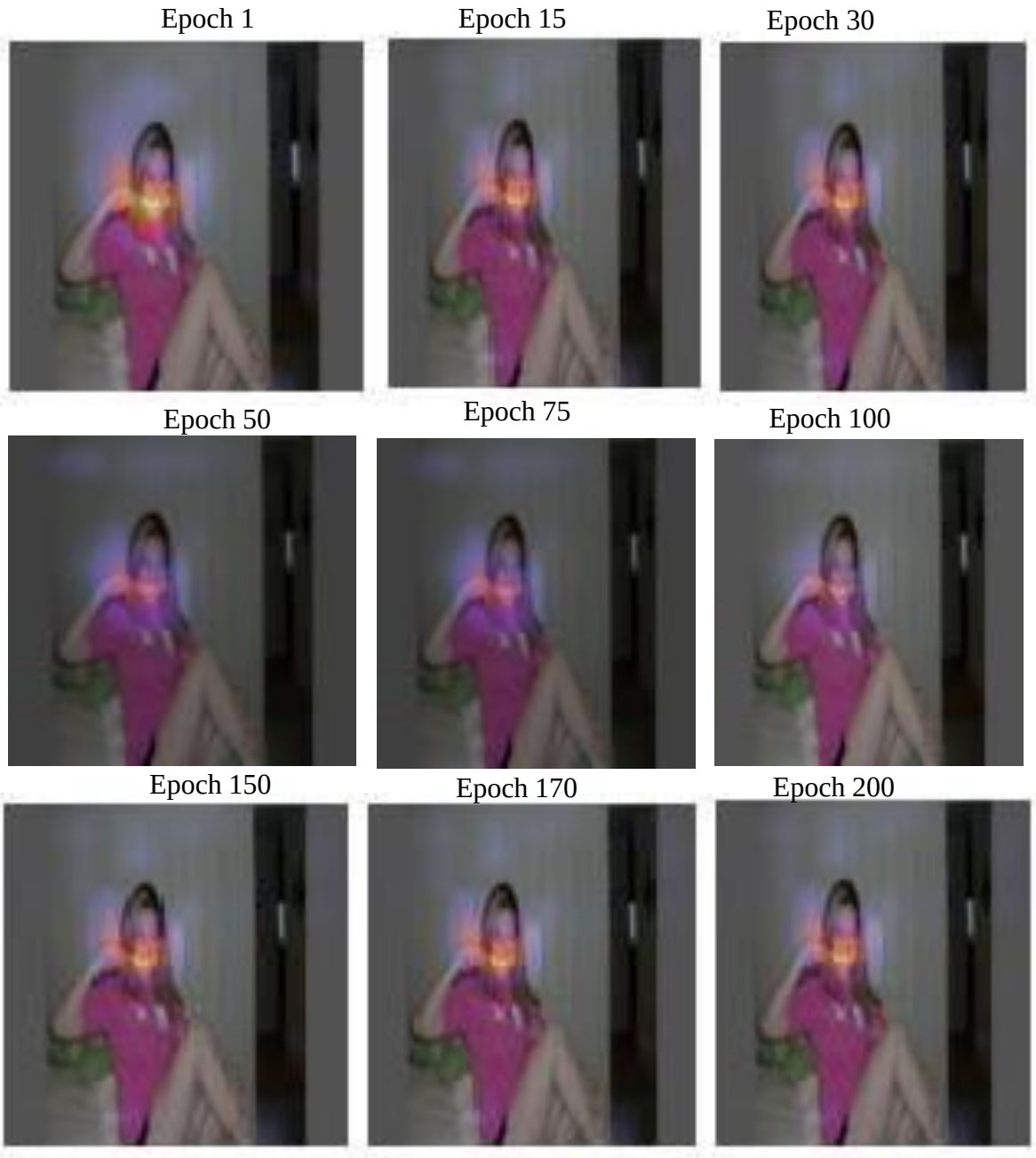
Appendix B.1: Result of SalGAN+ASPP



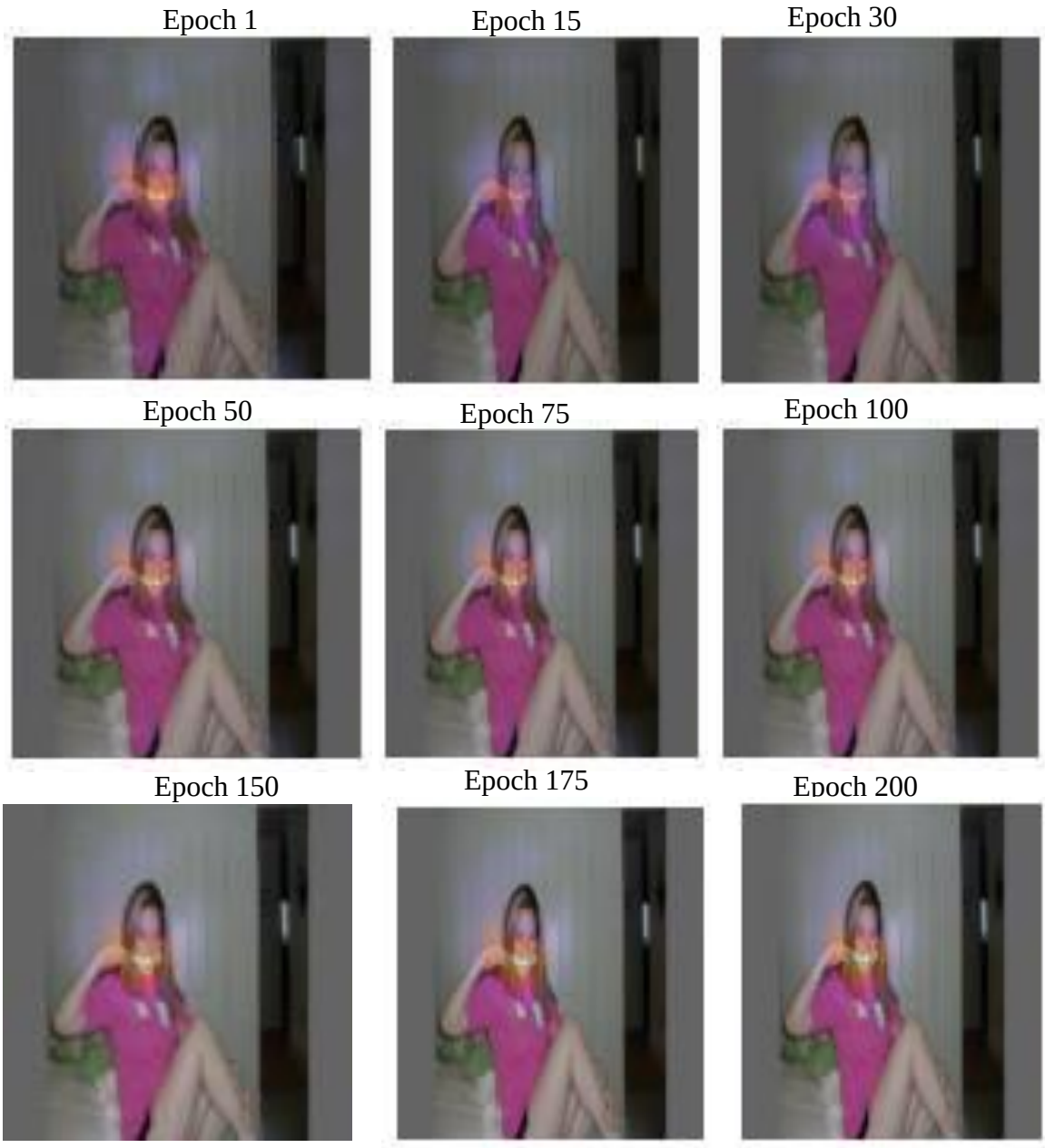
Appendix B.2: Result of SalGAN+e



Appendix B.3: Result of SalGAN+CHASPP



Appendix B.4: Result of SalGAN+CHASPP+e



Appendix C: Sample Code

Sample Data for Download data set and preprocessing

```
class SALICON:
    n_train = 10000
    n_valid = 5000

    def __init__(self, data_path):
        self._target_size = config.DIMS["image_size_salicon"]

        self._dir_stimuli_train = data_path + "stimuli/train"
        self._dir_stimuli_valid = data_path + "stimuli/val"

        self._dir_saliency_train = data_path + "saliency/train"
        self._dir_saliency_valid = data_path + "saliency/val"

        if not os.path.exists(data_path):
            parent_path = os.path.dirname(data_path[:-1])
            parent_path = os.path.join(parent_path, "")

            download.download_salicon(parent_path)

    def load_data(self):
        train_list_x = _get_file_list(self._dir_stimuli_train)
        train_list_y = _get_file_list(self._dir_saliency_train)

        _check_consistency(zip(train_list_x, train_list_y), 10000)

        train_set = _fetch_dataset((train_list_x, train_list_y),
                                   self._target_size, True)

        valid_list_x = _get_file_list(self._dir_stimuli_valid)
        valid_list_y = _get_file_list(self._dir_saliency_valid)

        _check_consistency(zip(valid_list_x, valid_list_y), 5000)

        valid_set = _fetch_dataset((valid_list_x, valid_list_y),
                                   self._target_size, False)

        return (train_set, valid_set)


class MIT1003:
```

```
    n_train = 803
    n_valid = 200

    def __init__(self, data_path):
        self._target_size = config.DIMS["image_size_mit1003"]

        self._dir_stimuli = data_path + "stimuli"
        self._dir_saliency = data_path + "saliency"

        if not os.path.exists(data_path):
            parent_path = os.path.dirname(data_path[:-1])
            parent_path = os.path.join(parent_path, "")

            download.download_mit1003(parent_path)

    def load_data(self):
        list_x = _get_file_list(self._dir_stimuli)
        list_y = _get_file_list(self._dir_saliency)

        _check_consistency(zip(list_x, list_y), 1003)

        indices = _get_random_indices(1003)
        excerpt = indices[:self.n_train]

        train_list_x = [list_x[idx] for idx in excerpt]
        train_list_y = [list_y[idx] for idx in excerpt]

        train_set = _fetch_dataset((train_list_x, train_list_y),
                                   self._target_size, True)

        excerpt = indices[self.n_train:]

        valid_list_x = [list_x[idx] for idx in excerpt]
        valid_list_y = [list_y[idx] for idx in excerpt]

        valid_set = _fetch_dataset((valid_list_x, valid_list_y),
                                   self._target_size, False)

        return (train_set, valid_set)


class CAT2000:

    n_train = 1600
```

```
n_valid = 400

def __init__(self, data_path):
    self._target_size = config.DIMS["image_size_cat2000"]

    self._dir_stimuli = data_path + "stimuli"
    self._dir_saliency = data_path + "saliency"

    if not os.path.exists(data_path):
        parent_path = os.path.dirname(data_path[:-1])
        parent_path = os.path.join(parent_path, "")

        download.download_cat2000(parent_path)

def load_data(self):
    list_x = _get_file_list(self._dir_stimuli)
    list_y = _get_file_list(self._dir_saliency)

    _check_consistency(zip(list_x, list_y), 2000)

    indices = _get_random_indices(100)

    # sample uniformly from all 20 categories
    ratio = self.n_train * 100 // 2000
    excerpt = np.tile(indices[:ratio], 20)

    for idx, _ in enumerate(excerpt):
        excerpt[idx] = excerpt[idx] + idx // ratio * 100

    train_list_x = [list_x[idx] for idx in excerpt]
    train_list_y = [list_y[idx] for idx in excerpt]

    train_set = _fetch_dataset((train_list_x, train_list_y),
                               self._target_size, True)

    # sample uniformly from all 20 categories
    ratio = self.n_valid * 100 // 2000
    excerpt = np.tile(indices[-ratio:], 20)

    for idx, _ in enumerate(excerpt):
        excerpt[idx] = excerpt[idx] + idx // ratio * 100

    valid_list_x = [list_x[idx] for idx in excerpt]
    valid_list_y = [list_y[idx] for idx in excerpt]
```

```
class PASCALS:

    n_train = 650
    n_valid = 200

    def __init__(self, data_path):
        self._target_size = config.DIMS["image_size_pascals"]

        self._dir_stimuli = data_path + "stimuli"
        self._dir_saliency = data_path + "saliency"

        if not os.path.exists(data_path):
            parent_path = os.path.dirname(data_path[:-1])
            parent_path = os.path.join(parent_path, "")

            download.download_pascals(parent_path)

    def load_data(self):
        list_x = _get_file_list(self._dir_stimuli)
        list_y = _get_file_list(self._dir_saliency)

        _check_consistency(zip(list_x, list_y), 850)

        indices = _get_random_indices(850)
        excerpt = indices[:self.n_train]

        train_list_x = [list_x[idx] for idx in excerpt]
        train_list_y = [list_y[idx] for idx in excerpt]

        train_set = _fetch_dataset((train_list_x, train_list_y),
                                   self._target_size, True)

        excerpt = indices[self.n_train:]

        valid_list_x = [list_x[idx] for idx in excerpt]
        valid_list_y = [list_y[idx] for idx in excerpt]

        valid_set = _fetch_dataset((valid_list_x, valid_list_y),
                                   self._target_size, False)

        return (train_set, valid_set)


    n_train = 99
    n_valid = 50
```

```
  def __init__(self, data_path):
    self._target_size = config.DIMS["image_size_fiwi"]

    self._dir_stimuli = data_path + "stimuli"
    self._dir_saliency = data_path + "saliency"

    if not os.path.exists(data_path):
      parent_path = os.path.dirname(data_path[:-1])
      parent_path = os.path.join(parent_path, "")

      download.download_fiwi(parent_path)

  def load_data(self):
    list_x = _get_file_list(self._dir_stimuli)
    list_y = _get_file_list(self._dir_saliency)

    _check_consistency(zip(list_x, list_y), 149)

    indices = _get_random_indices(149)
    excerpt = indices[:self.n_train]

    train_list_x = [list_x[idx] for idx in excerpt]
    train_list_y = [list_y[idx] for idx in excerpt]

    train_set = _fetch_dataset((train_list_x, train_list_y),
                    self._target_size, True)

    excerpt = indices[self.n_train:]

    valid_list_x = [list_x[idx] for idx in excerpt]
    valid_list_y = [list_y[idx] for idx in excerpt]

    valid_set = _fetch_dataset((valid_list_x, valid_list_y),
                    self._target_size, False)

    return (train_set, valid_set)


class TEST:


  def __init__(self, dataset, data_path):
    self._target_size = config.DIMS["image_size_%s" % dataset]
```

```
        self._dir_stimuli_test = data_path

    def load_data(self):
        test_list_x = _get_file_list(self._dir_stimuli_test)

        test_set = _fetch_dataset(test_list_x, self._target_size,
                                  False, online=True)
        return test_set


def get_dataset_iterator(phase, dataset, data_path):

    if phase == "train":
        current_module = sys.modules[__name__]
        class_name = "%s" % dataset.upper()

        dataset_class = getattr(current_module, class_name)(data_path)
        train_set, valid_set = dataset_class.load_data()

        iterator = tf.data.Iterator.from_structure(train_set.output_types,
                                                   train_set.output_shapes)
        next_element = iterator.get_next()

        train_init_op = iterator.make_initializer(train_set)
        valid_init_op = iterator.make_initializer(valid_set)

        return next_element, train_init_op, valid_init_op

    if phase == "test":
        test_class = TEST(dataset, data_path)
        test_set = test_class.load_data()

        iterator = tf.data.Iterator.from_structure(test_set.output_types,
                                                   test_set.output_shapes)
        next_element = iterator.get_next()

        init_op = iterator.make_initializer(test_set)

        return next_element, init_op


def postprocess_saliency_map(saliency_map, target_size):

    saliency_map *= 255.0
```

```
    saliency_map = _resize_image(saliency_map, target_size, True)
    saliency_map = _crop_image(saliency_map, target_size)

    saliency_map = tf.round(saliency_map)
    saliency_map = tf.cast(saliency_map, tf.uint8)

    saliency_map_jpeg = tf.image.encode_jpeg(saliency_map, "grayscale", 100)

    return saliency_map_jpeg

        image_list.append(image)

    image_list.append(original_size)
    image_list.append(files)

    return image_list


def _resize_image(image, target_size, overfull=False):

    current_size = tf.shape(image)[:2]

    height_ratio = target_size[0] / current_size[0]
    width_ratio = target_size[1] / current_size[1]

    if overfull:
        target_ratio = tf.maximum(height_ratio, width_ratio)
    else:
        target_ratio = tf.minimum(height_ratio, width_ratio)

    target_size = tf.cast(current_size, tf.float64) * target_ratio
    target_size = tf.cast(tf.round(target_size), tf.int32)

    shrinking = tf.cond(tf.logical_or(current_size[0] > target_size[0],
                                      current_size[1] > target_size[1]),
                        lambda: tf.constant(True),
                        lambda: tf.constant(False))

    image = tf.expand_dims(image, 0)

    image = tf.cond(shrinking,
                    lambda: tf.image.resize_area(image, target_size,
                                                 align_corners=True),
                    lambda: tf.image.resize_bicubic(image, target_size,
                                                    align_corners=True))
```

```
def _get_file_list(data_path):

    data_list = []

    if os.path.isfile(data_path):
        data_list.append(data_path)
    else:
        for subdir, dirs, files in os.walk(data_path):
            for file in files:
                if file.lower().endswith((".png", ".jpg", ".jpeg")):
                    data_list.append(os.path.join(subdir, file))

    data_list.sort()

    if not data_list:
        raise FileNotFoundError("No data was found")

    return data_list


def _get_random_indices(list_length):


    indices = np.arange(list_length)
    prng = np.random.RandomState(42)
    prng.shuffle(indices)

    return indices


def _check_consistency(zipped_file_lists, n_total_files):


    assert len(list(zipped_file_lists)) == n_total_files, "Files are missing"

    for file_tuple in zipped_file_lists:
        file_names = [os.path.basename(entry) for entry in list(file_tuple)]
        file_names = [os.path.splitext(entry)[0] for entry in file_names]
        file_names = [entry.replace("_fixMap", "") for entry in file_names]
        file_names = [entry.replace("_fixPts", "") for entry in file_names]

        assert len(set(file_names)) == 1, "File name mismatch"
```