Perception-Hacking with Uncompressed Image Transmission in Continuous-time Mixnets

Kuan Lon Vu

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2024

Abstract

Mixnets are privacy-enhancing networks that protect against strong adversaries. Their inherent high-latency design currently limits their possible use cases to those that can tolerate medium to high latencies. Data transmission nowadays consists of a large proportion of images, which often leads to increased latency due to network bottlenecks. Such problem is amplified in the context of mixnets due to the added delays to the packets as they traverse through the mixnet.

We present a novel approach to shorten the perceived latency of image transmission by leveraging an inpainting-based image reconstruction framework. The recipient could effectively receive a reconstructed image close to the original image with as low as only 38% of the total number of transmitted packets.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Kuan Lon Vu*)

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Tariq Elahi, for the insightful discussions, and for the access to the lab machine and the Raspberry Pi on which some experiments were conducted on. I also would like to thank my second marker, Dr. Marc Juarez, for the discussions and the feedback on my report.

Again, words cannot express my gratitude for the love and support from my family and friends. Thank you, Mum, for your encouragement and support. For Jacqueline, thank you for keeping me company as I worked on the project, and for reminding me of God's grace with your daily Psalm. You bring so much joy and comfort to me. For Ruth, thank you for being in the same city, for listening and for reminding me that I am wonderfully and fearfully made.

Thank you, Elisa, for taking the time to understand my project and for proofreading the report.

Thank you, Jingyuan and Mrinmoy for reading and giving feedback on the report.

Thank you to Ben for the discussions and suggestions.

Thank you to the people in my family and my small group who were praying for me throughout the year. For Karen and Jill, thank you for checking on me when I disappeared into my work.

Most importantly, to my Saviour, Jesus Christ, in whom I anchor my faith. Thank you for the incredible people He put into my path on this amazing journey.

And Ruth, that was fun, let's do that again! :)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Nowadays, web browsing is an integral part of our daily lives with the ubiquitous usages of the Internet. In the last decade, there has been a growing awareness of online anonymity. Even with the incorporation of state-of-the-art encryption schemes, it is well-acknowledged that the anonymity of the users can remain compromised when web browsing directly over the Internet; thus, there is an increasing interest in anonymous systems. The nature of The Onion Router (Tor) [15], a popular low-latency anonymising network, does not obfuscate the packet and network metadata such as the packet arrival times and packet sizes [14]. Current research demonstrates that low-latency anonymous networks such as Tor are vulnerable to timing attacks [22, 42]. Thus, the anonymity protection offered by these encryption schemes and synchronous networks falls short of protecting users against global passive adversaries (GPA) under relatively simple attacks such as packet counting in [40], where the strong adversary can observe all packet flows across the network to perform timing and traffic analysis [14]. This results in a growing interest in privacy-enhancing anonymous network infrastructures such as mix networks (mixnets) [14, 35].

Mixnet is a high-latency decentralised overlay network that provides anonymity protection against a strong global adversary deploying mixing and packet cryptographic transformation at each mix. The objective of mixing is to interfere with the timing metadata of participating components when observing a mix in the network, and also the network as a whole [9, 11, 14]. The mixing strategy is often based on some distribution, such as the exponential distribution, which determines the delays of the packets as they traverse through the mixnet. The utilisation of mixing incurs higher end-to-end latency for the packets and packets arriving out-of-order. Therefore, the performance of mixnet in terms of latency is intrinsically worse than that of other better-adapted networks in exchange for strong anonymity protection, with greater computational costs and resource usage.

In this work, we focus on continuous-time mixnets since this design provides optimal anonymity and bounded end-to-end latency [11]. Though the mixnet latency has reduced, its still-existent latency inevitably contributes to its slow adaptation to a

broader user base compared to other synchronous networks such as Tor. Hence, the usage for mixnet is conceptualised to use cases with medium to high latency tolerance such as instant messaging, cryptocurrency transactions and mail systems [9, 24, 35]. Image transmission is a fundamental element in web browsing. With more than five billion Internet users worldwide[1], web browsing is among the most common online activity. Web browsing requires low latency up to a certain threshold for acceptable user experience [1]. Tor is highly adapted for web browsing with its synchronous nature and dedicated free and open-source Tor browser to connect end users easily to the Tor overlay network[2]. While current literature considers mixnet for use cases where greater latency is tolerable, image transmission for web browsing is less forgiving towards latency. We work with images as images account for a large portion of data transmission [59]. Many websites engage their visitors through their vast usages of visual content. Image data tends to be larger in volume than other media, such as text; they often suffer from network bottlenecks during transmission as they require higher bandwidths. The issue of high latency worsens in the case of mixnets. In this work, we attempt to give an apparent reduced latency on image transmission, which we refer to as ”perception-hacking” - changing the users' perception to create an illusion of better performance. We believe that shortening the mixnet latency that is acceptable for web-browsing without compromising the anonymity guarantees of the mixnet could appeal to a greater audience and encourage a smoother transition to mixnet.

1.2 Previous Work in MInf Part 1

Of the many mixing strategies, MInf Part 1 project [45] is based on continuous-time mixnets where mixes delay packets individually. We investigated the overhead of cryptographic operations in packet processing at the mixes and the significance of the overhead on the anonymity guarantees of the continuous-time mixnet. It was demonstrated that the cryptographic packet unwrap operation takes a non-negligible period of time, and the duration of the operation is hardware-dependent.

The packet processing time breaks the anonymity when the average per-mix delay of the packets is shorter than the packet processing time. In such a case, the packet becomes over-delayed at the mix as the packet processing time is the overwhelming proportion of the actual per-mix delay experienced by the packet. Thus, these packets are vulnerable to packet-distinguishing attacks, which were also novelly presented. To mitigate against the effects of the processing overhead, the average per-mix delays should be sufficiently large to account for the vast range of possible machines used as mixes in order to maintain the anonymity property of the continuous-time mixnet. The greater average per-mix delay results in a greater probability of per-mix delays that are longer than the packet processing time of the mix, reducing the adversary attack surface.

One approach to enable mixnets for image transmission is to reduce the per-mix delay such that the mixnet end-to-end latency shortens to one that is suitable for low-latency use cases. Given these findings, it is not possible to reduce the average per-mix delay

[1] https://www.statista.com/statistics/617136/digital-population-worldwide/
[2] https://www.torproject.org/download/

directly as it could compromise the anonymity of the network. This warrants the work presented here which does not impact the anonymity guarantees of the mixnet.

1.3 Problem Statement

In image transmission, packets carrying the image information traverse the mixnet between the sender and the recipient. The recipient would wait for all the packets to obtain the complete image. To provide the illusion of quickness, we propose that the recipient stops waiting for the remaining packets once the first k packets arrive. It effectively reduces the network latency in image transmission to the time taken for the first k packets to traverse the mixnet. The packets that did not arrive are the slow packets. The information carried by those slow packets are reconstructed using image inpainting algorithm, as those slow packets can be regarded as *lost*. Thus, our reconstruction framework essentially becomes an information recovery task to conceal packet loss in image transmission.

A natural question arises at this point: *Why are we opting for an inpainting approach for loss concealment?* Modern-day networks employ sophisticated and well-tested approaches to mitigate the effects of packet loss, such as the automatic repeat request (ARQ) methods in the transmission control protocol (TCP) and error-correcting codes [13]. We argue that these approaches are not suitable in the context of mixnets. ARQ recovers the information loss through packet retransmission. In a mixnet, the retransmitted packets could have long per-mix delays, further extending the overall latency. Error-correcting codes depend on redundancy, resulting in more data being transmitted. The increase in data would directly increase the number of packets to transmit as the payload size per packet is fixed. Therefore, there are more packets to traverse and to be delayed in the high-latency network. Thus, the image recovery in mixnets must be performed solely with available information from the received packets on the recipient end, which image inpainting is well-suited for the task.

1.4 Project Objectives

The primary purpose of this project is to create an illusion of low latency in an inherently high-latency mixnet. While the approach of the presented work applies to mixnets with various mixing strategies, this project's scope focuses on continuous-time mixnets.

This project presents:

- An image reconstruction framework for image transmission in mixnets.
- The category of the inpainting algorithm is suited as an image reconstruction method in the mixnet setting.
- The design factors considered for the reconstruction framework, such as the type of inpainting algorithm.
- The proportion of packets out of the total number of packets the recipient waits for before beginning the image reconstruction process.

1.5 Contributions

To the best of our knowledge, this work is the first to expand the use cases of mixnet by creating the illusion of low latency. The main contributions of this project are as follows:

- Created an image reconstruction framework that is suitable for the mixnet context.
- Designed the components of the image reconstruction framework
- Trained and tested several inpainting models.
- Determined the appropriate components to the reconstruction framework, such as the type of inpainting algorithm.
- Verified the range of hardware configurations suitable for executing the reconstruction framework.
- Determined the proportion of packets out of the total number of packets to produce a close reconstruction of an image.

1.6 Terminology

This section clarifies some key terminologies used in this report that can be confused for a different meaning in a different context.

1. **Recipient**: The recipient in this report denotes the machine participating in the mixnet and receiving the image rather than the actual human operator.
2. **Packet**: In this report, a packet is defined as the specialised, layered-encrypted network packet in Sphinx format, specifically used in mixnets.
3. **End-to-end latency**: End-to-end latency is the time taken from the packet to traverse through the mixnet from the sender to the recipient. In this report, we use end-to-end latency and latency interchangeably.
4. **Slow packet**: A slow packet is a packet that does not arrive before the recipient begins the image reconstruction process.
5. **Image fidelity**: Image fidelity refers to the likeliness of an image to its original or reference image [33]. For historical purposes, image fidelity is also called image quality in the image inpainting context [32].

1.7 Report Organisation

The remaining of the report is structured as follows:

Chapter 2 gives an overview of the background of this project, including continuous-time mixnet, image inpainting and its algorithms related to this project. A discussion of the relevant work in image reconstruction methods is also provided.

Chapter 3 gives a high-level overview of the proposed image reconstruction framework and the design factors to consider when deciding on the suitable image inpainting algorithm for the framework.

Chapter 4 outlines the experimental framework, the experimental data and the experiment setup used to address the design factors for the image reconstruction framework.

Chapter 5 details the experiments carried out and the implications to the image reconstruction framework based on the summary of results from the experiment.

Chapter 6 evaluates and discusses some of the limitations in this work, and presents some possible directions for future work.

Chapter 2

Background and Related work

This chapter gives an overview of continuous-time mixnets and a brief introduction to the background of image inpainting and some of its techniques relevant to this project. It also presents the metric for image similarity evaluation and the image data source for the experiments.

2.1 Continuous-time Mixnets

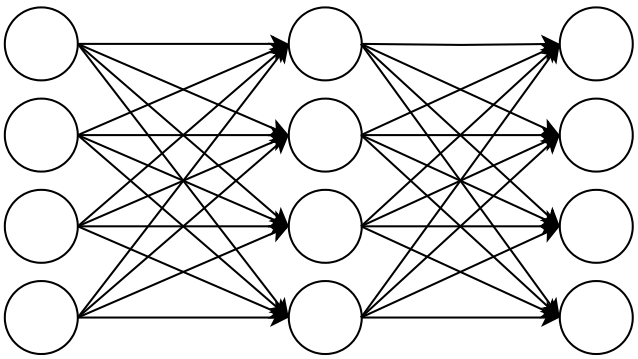


Figure 2.1: A 4 $\times$ 3 stratified topological mixnet, where the circles denote the mixes and the arrows represent the possible links between the mixes.

Mixnet is a decentralised network of mixes - machines that delay and then forward the packet along its route to its destination [9]. The mixnet design uses a packet mixing strategy to provide anonymity protection to its users against a global passive adversary (GPA) [14]. There are two capabilities of a GPA. Firstly, it can observe all incoming and outgoing traffic of end users and mixes in the network. Secondly, it can perform traffic and timing analysis attacks to correlate the sender and receiver of any identified packet by analysing the time of sending and arrival of a packet. A GPA cannot actively corrupt an honest participant in the network. Adding the per-mix latencies destroys the network metadata exploited by the adversary, thus achieving the objective of protecting the sender-recipient anonymity.

A packet passes through a sequence of mixes in a multi-hop fashion similar to onion-routing in Tor [15]. The multi-hop decentralised design distributes trust across the participants in the network, thus preventing a particular element of the network from being adversely targeted. It ensures that no nodes possess complete knowledge of both the sender and receiver for any packets.

A continuous-time mixnet is characterised by its mixing strategy whereby packets are delayed individually at each mix, and the rate at which end users send the packets is modelled stochastically by the Poisson process [11]. The exponential distribution that the sender randomly samples from when preparing the packets is parameterised by μ. In the context of mixnets, μ is a preset mixnet parameter that denotes the average per-mix delay in seconds. The variable μ is the mean of the exponential distribution.

A delay is selected for every mix on the packet's route through the mixnet. At each mix, the mix cryptographically transforms the incoming packet so that the packet being forwarded to its next destination is unrecognisable by the observing adversary [14]. The delay sampled contributes to the mixing at each mix - the packet is individually delayed and thus reordered to prevent the adversary from correlating the sender and receiver by measuring the message sent and arrival times. The anonymity of this mixing strategy is based on the memoryless property of the exponential distribution [10], where the probability of the packet being forwarded by the mix is independent of its arrival time. Consequently, any packet arriving at the mix has an equal likelihood of departing at the subsequent moment. Notice that the greater the value of μ, the more packets are being held in the mixes due to the increased average per-mix delay. This results in greater anonymity at the cost of longer end-to-end latency.

The packets in the mixnet are in the Sphinx packet format [12]. The Sphinx packet format consists of two parts - the header and the payload - each layer encrypted in reverse routing order. This means that the outermost encryption corresponds with the first mix of the packet's route and the innermost encryption is for the recipient. The header contains the metadata for the mixes which includes their respective per-mix delay for mixing and a single element of a cyclic group for its cryptographic operations [35]. In this work, the payload would be a partition of the image data. For the remaining of the report, we abbreviate Sphinx packets to simply packets, unless otherwise specified.

Loopix anonymity system is a realisation of the stratified continuous-time mixnet structure, introduced by Piotrowska *et al.* [35]. Figure 2.1 illustrates an example of the network structure of a 4×3 mixnet in stratified topology. While our image reconstruction framework is integrated with a Loopix-styled mixnet, the techniques presented are generalisable to any mixnet structure.

2.2 Image Inpainting

In the context of computer vision, image inpainting is a common digital restoration technique applied to still images and videos. It is extended from the analogous ancient practice of artwork restoration used to reconstruct the image to its original state [6, 23]. In digital image inpainting, inpainting is the process of reconstructing lost or deteriorated regions of the images or videos [23]. Existing algorithms often employ the underlying

assumption that the geometric structure and statistical information are common between the visible and missing areas of the image [19]. The unknown regions can be filled by interpolating from neighbouring known pixels during the restoration process [23]. The objective of the image inpainting algorithms is to create an inpainted image such that the result is visually pleasing and close to the original or reference image [19]. Due to its simplicity, the applications of image inpainting range widely from removing overlaying objects for image enhancement to concealing information loss due to compression or packet loss in transmission [16, 19].

2.2.1 Digital Image Restoration

Digital image restoration through image inpainting has been an extensively researched topic since the 1950s, which began from recovering information loss for images taken in sub-optimal conditions in astronomical imaging [4]. Since then, image restoration techniques have been widely applied in image processing. In this section, we focus on the significant works conducted on image inpainting that are related to the methods in Section 2.2.2.

Digital image inpainting is considered to have been pioneered by Bertalmio *et al.* in [6], of which the NS and TELEA algorithms improve. Their framework emulates the restoration techniques used by experts on ancient artworks through non-linear partial differential equations [39]. However, the results were sub-optimal as the inpainted regions tended to be blurry with some unnatural artefacts.

In deep learning-based inpainting technique, the U-net introduced by Ronneberger [37] is another popular network structure besides the FCN. The U-net has a greater computational cost compared to the FCN whilst having the ability to retain a larger amount of spatial information. Since the inpainting process in the mixnet context is only meaningful when the inference is completed quickly. Thus, deep learning methods that utilise the U-net architecture are unsuitable for this project.

2.2.2 Image Inpainting Methods

This section presents some traditional and deep learning-based image inpainting algorithms. These algorithms use inpaint masks to indicate the defective regions to be inpainted in the image, and those regions are interpolated in the pixel domain using information in the unmasked areas. A inpaint mask of an image indicates whether the pixels in the image are valid or unknown; it is visualised as a monochromatic image with the exact dimensions as the image.

2.2.2.1 Traditional Methods

In literature, traditional methods conventionally refer to the category of non-deep learning-based inpainting techniques [23, 49]. We highlight briefly the technical details of two well-acknowledged algorithms implemented in the publicly available OpenCV library[1].

[1] `https://docs.opencv.org/4.x/d7/d8b/group__photo__inpaint.html`

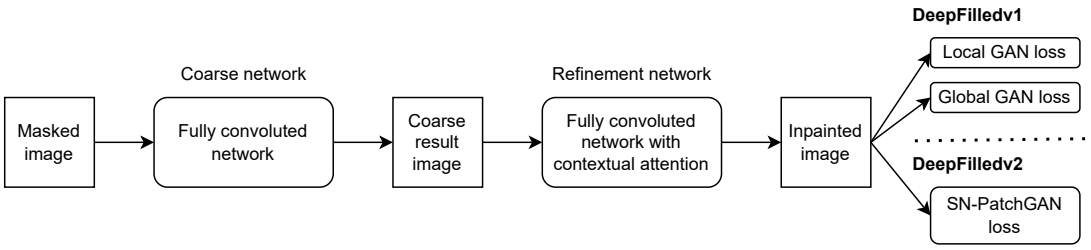


Figure 2.2: A simplified overview of network structure of DeepFillv1 and DeepFillv2.

2.2.2.1.1 Navier-Stokes Inpainting Algorithm Introduced by Bertalmio *et al.* [5], the Navier-Stokes Inpainting algorithm (henceforth, NS) is a diffusion-based approach based on principles from fluid dynamics. The algorithm directly applies the solutions of the well-established Navier-Stokes equations; it draws an analogy between image intensity and stream function for an incompressible Newtonian flow. The inpainting process extends the information from known regions determined using partial differential equations to the neighbouring unknown regions [19].

2.2.2.1.2 Fast Marching Method Inpainting Algorithm Established by Telea [43], the inpaint technique is based on a numerical technique called the Fast Marching Method (henceforth, TELEA). The inpaint mask marks the unknown regions on the image, and the algorithm begins inpainting with the unknown pixels on the boundary to the known pixels. The process repeats iteratively until all unknown pixels are interpolated. The pixel to be inpainted at each iteration is selected according to the Fast Marching Method, and the inpainting propagates to the centre of the unknown regions [34].

2.2.3 Deep Learning-based Methods

2.2.3.1 Motivation for Deep Learning Inpaint Models

There has been much advancement with deep learning approaches in digital image inpainting algorithms in the study of computer vision and image processing [36]. Deep learning models build on simpler concepts to learn more complex ones hierarchically [18]. Variants of the convolution neural networks (CNNs), often in combined usage with Generative adversarial networks (GANs), adapted for the inpainting task are the two predominant network architectures in the field of deep learning-based inpainting methods [56]. Fully convolutional networks (FCNs) are popular CNN variants frequently used in literature [56]. Researchers utilise these deep learning networks to capture high-level semantics effectively to produce complex inpainting results [49].

2.2.3.2 DeepFillv1 and DeepFillv2 Models

We present two state-of-the-art deep learning-based inpainting methods relevant to this project: DeepFillv1 [54] and its enhanced version, DeepFillv2 [53].

2.2.3.2.1 DeepFillv1 Proposed by Yu *et al.* in the paper ”Generative Inpainting with Contextual Attention” [54], the well-cited generative inpainting framework uses the

typical two-stage coarse-to-fine network architecture with a novel contextual attention layer, depicted in Figure 2.2. Classic CNN architectures capture the entire image context at the trade-off of losing spatial information in deeper layers in the network [54]. This is because the spatial size of features reduces as the layers increase. The contextual attention overcomes the shortcomings of classic CNN-based frameworks. It enables the refinement generator to use feature information from distant spatial locations to reconstruct the local unknown regions [54].

In coarse-to-fine network architecture, the model performs a coarse prediction at the first stage and produces a refined result based on the coarse prediction at the second stage [49]. Both stages were trained jointly under one training process for efficiency. The network architecture consists of two generator networks and two discriminator networks. The two generators perform the coarse-to-fine inpainting, and the result is passed onto the two discriminators with GAN losses for global and local inpainting evaluation. The global discriminator assesses the images as a whole, while the local discriminator assesses solely on the inpainted regions. The generators have similar FCN structures with dilated convolutions [52] at both stages, except that the refinement generator at the second stage has an additional contextual attention module.

2.2.3.2.2 DeepFillv2 This framework is also proposed by the same team of researchers in DeepFillv1 [53]. Likewise to the network structure of its predecessor illustrated in Figure 2.2, the framework leverages a two-stage coarse-to-fine network in DeepFillv2. However, to support irregular inpainting masks, gated convolution is used in DeepFillv2 instead of the standard convolution in DeepFillv1. DeepFillv2 improves the use of partial convolution on inpainting with irregular masks. Irregular masks represent a more realistic inpainting task, as the unknown regions are not always guaranteed to be regular. Partial convolution was introduced by Liu *et al.* [30] to enable smooth inpainting with irregularly shaped inpaint masks. Missing pixels are usually filled with meaningless placeholder information in the image at the recipient. Previous models, including DeepFillv1, utilise the standard convolution mode, producing blurry results. It is because the standard convolution is conditioned equally on the valid pixels and the missing pixels without distinction, when the missing pixels are usually filled with meaningless placeholder information [56]. Partial convolution ensures that the convolution results depend only on the valid pixels in the current mask. However, partial convolution is limited by its restrictive mask updating rule. If the convolution result depends on at least one valid pixel, the corresponding location is regarded as valid in the subsequent convolution [30]. DeepFillv2 improves on partial convolution by introducing gated convolution. Rather than the hard updating rules, the mask update is learned from the data in gated convolution [53]. Thus, gated convolution can be regarded as a more general form of the partial convolution with greater flexibility for mask updating [49]. SN-PatchGAN is an efficient variant of GAN loss novelly introduced in the same work. It replaces the use of local and global GAN loss in DeepFillv1 as it has been found to be more suited for the inpainting task with irregular masks.

2.2.4 Evaluation Metric

2.2.4.1 Structure Similarity Index

The human visual system (HVS) views the images transmitted through the mixnet, hence it is reasonable to base our evaluation on a perceptual image quality metric consistent with human visual perception. In this work, analogous to loss concealment in transmission, the ground truth, i.e. the reference image, is known since it is simply the pre-transmitted image, objective full-reference metrics are applicable. Thus, we use the structural similarity (SSIM) index, a popular perceptual-based metric for the evaluation.

Presented by Wang and Bovik in [47], Structural Similarity (SSIM) index is a full-reference metric that quantifies comprehensively the likeliness of two images, i.e. the reference/original image and the processed image. SSIM combines the three distinctive comparisons of luminance, contrast and structure into an overall score. Equation 2.1 denotes the per-pixel SSIM for images A and B with image dimension of $M \times N$.

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y,)]^{\gamma}, \tag{2.1}$$

where l is the function for luminance, c is the function for contrast and s is the function for structure, the parameters $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ adjust the relative significance of their respective components [47].

The authors rewrote Equation 2.1 with $\alpha = \beta = \gamma = 1$ in terms of the mean intensities μ_x and μ_y and their standard deviations σ_x and σ_y of pixels x and y respectively to give Equation 2.2 [47]:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.2}$$

where $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$, $\sigma_x = (\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2)^{\frac{1}{2}}$, and C_1 and C_2 are constants for stability when the sums in the denominator are close to 0. Note that μ_y and σ_y are defined similarly but are omitted due to limited space. The standard deviations are indicators of the signal contrasts.

In practice, Wang *et al.* advise applying SSIM locally rather than globally to more accurately reflect how the HVS is only able to perceive certain regions of the image with high resolution at a time, among other reasons [47]. The mean SSIM (MSSIM) index of the local SSIM scores across the entire image between images A and B is given by:

$$MSSIM(A,B) = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} SSIM(x,y). \tag{2.3}$$

Since MSSIM is a full-reference metric, one of the images A and B is the reference image. The MSSIM satisfies the boundedness property [47]. The scores range from -1 to 1, where -1 indicates complete dissimilarity between the images, and a value closer

to 1 denotes better image fidelity. The value of 1 is achieved when images A and B are identical.

MSSIM considers structural information in its design to overcome the limitation of metrics, such as mean square error (MSE) and its logarithmic-based variant (PNSR) [29]. MSE-based metrics are well known for their simplicity and computational efficiency but also for their drawback of providing a weak indication of perceived visual quality in human [29, 32, 33, 48]. This limitation is caused by the fact that these metrics fundamentally perform pixel-by-pixel comparisons, overlooking the overall structural changes - evident when the image is viewed in its entirety - which the human visual system is commonly assumed to be.

2.2.4.2 Correlation of MSSIM and Human Quality Perception

While assessing image fidelity is seemingly an effortless task for human assessors, the underlying mechanism is complex and remains an active area of research. Over the years, much research has been conducted to build perceptual image fidelity metrics that emulate the characteristics of the HVS [29]. MSSIM aims to measure image distortion that reflects the high sensitivity of HVS to structural changes in order to simulate the perceived image fidelity [32].

Prior research presents the SSIM metric as an effective indicator of degradation or improvement in the structure of denoised images and shows some correlation with the subjective human perception of image fidelity [32]. The mean opinion score (MOS), a subject quality assessment metric, is representative of the human quality perception since it is defined as the average rating of a given still or video image on a particular scale from a group of human observers through subject viewing assessments [31]. Wharton *et al.* [48] conducted a study comparing objective perceptual metrics with human evaluation. Among the investigated metrics, they found that the MSSIM achieved the highest correlation coefficient with the human perception measured by the subjective Difference Mean Opinion Score when evaluating various types of image distortion.

2.3 Existing Work on Packet Loss Concealment in Data Transmission

The work undertaken here draws a parallel with the loss concealment in the packet-based transmission of media data. In data transmission, a packet is the small medium in which data is transmitted. Packetisation is the process where the data is segmented into packets that traverse the network when the data is larger than the packet payload size of a single packet. Packet loss is a common issue in data transmission. Media data such as audio, images and videos are susceptible to packet loss as they yield undesirable artefacts that affect the quality of the transmitted data. Packet loss concealment has been addressed with various approaches in the literature, such as error-correcting codes at the physical layer, and retransmission at the data link layer [44]. However, the work of our particular interest is those leveraging inpainting approaches to provide alternative methods for concealing packet loss during data transmission.

In audio data transmission, Lee and Chang [27] proposed using deep learning to conceal digital speech packet loss. Their approach involves a regression-based deep neural network to reconstruct the features of the lost speech signals. The results showed that the signals reconstructed using DNN significantly improved downstream speech recognition tasks. However, their framework operates on the feature level of the audio signal, which requires feature selection whilst our work is at the pixel level.

Bajić [2] proposed using partial differentiable equation-based inpainting methods to recover missing features in latent space to maintain object detection accuracy. His work is applicable in the scenario of collaborative intelligence - where a deep learning model is divided into the edge-based and cloud-based sections [3]. The relatively lightweight feature extraction into feature tensors is performed locally on the edge device, and the resource-intensive inference is completed in the cloud, as there are often more processing resources in cloud computing than in local hardware. With the lossy nature of the network/transport layer, parts of the feature tensors could be missing due to packet loss. While the results showed that latent-space inpainting improves on existing solutions for feature recovery, the proposed recovery framework is demonstrated on latent features rather than natural images.

In terms of image transmission over the Internet, there is limited work on inpainting approaches for packet loss concealment. This is due to present packet recovery methods such as ARQ and error-correcting codes can be suitably applied in those networks. Hemami and Meng [21] and Wang and Zhu [46] proposed reconstruction schemes on transform-coded images. The recovery target was the transform coefficient in block-transform coded images. The data packetisation in which those schemes are applied involves partitioning images into 8×8 blocks and performing a 2D transformation to obtain their corresponding transform coefficients. These blocks of transform coefficients are the payloads of the transmitted packets. Thus, a packet loss indicates a missing block — the proposed schemes reconstructed those lost blocks by extracting information from adjacent blocks.

2.4 Birds 525 Species - Image Classification Dataset

The ”Birds 525 Species Image Classification Dataset” (henceforth, Birds525) is a dataset that consists of 89,885 JPG images of 525 species of birds, dedicated to the public domain under CC0. Of the total number of images, 84,635 images are the training set, while the validation and testing sets consist of 5 images per species, resulting in 2,625 images each. The structure of the images is consistent: they are all 224×224 resolution 3-channelled images where the three channels denote the RGB colour encoding scheme; each image consists of a bird that takes up at least half of the pixels in the image [17]. We select this dataset primarily for its fixed-sized images as an image data source for our experiment, hence the imbalanced species in the training set is not a factor of concern. The image dimensions are also sufficiently large to justify the occurrence of slow packets. Since the dataset was curated for image classification tasks, the dataset includes species labels for each image, which are not relevant to our task and, therefore, have been omitted.

Chapter 3

Proposed Image Reconstruction Framework

This chapter outlines the integration of the image reconstruction framework into the mixnet infrastructure and discusses factors to consider in the framework design.

3.1 Image Packetisation Model

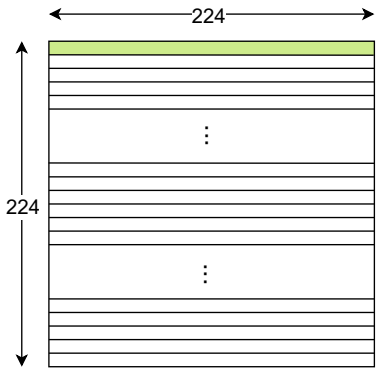


Figure 3.1: Image segmentation for data packetisation where the partition unit is a row of the image. Each row (green) of the image has a dimension of 1×224 pixels.

In this section, we establish the packetisation model on which the theoretical assumptions of our image inpainting framework will be built. The images in Birds525 all have dimensions of $224 \times 224 \times 3$ pixels. For the purposes of the data packetisation in our experiment simulation, the images are represented by a Numpy array with the shape of (224, 224, 3). The packet payload is assumed to be a multiple of the size of a *row* of dimension 1×224 pixels. Similar to data packetisation in standard network protocols, image data is segmented into smaller partitions by the size of the packet payload as illustrated in Figure 3.1. We place a constraint on the correlation between slow packet and missing image information, such that the loss of a packet results in the loss of rows of pixels in the image. The image is also assumed to be packetised

into a stream of packets such that the existence of a slow packet does not affect the decoding of subsequent rows of pixels in the image at the recipient. These constraints are analogous to the ones also made on the transform coefficient blocks in the work by Hemami and Meng [21], and Wang [46]. Thus, the effect of the slow packets in terms of content loss in the image is limited to the loss of information carried by those slow packets; the transmitted image would display missing regions in the respective rows of the slow packets. Based on this packet loss model, we focus on the pixel-wise image reconstruction techniques in our reconstruction framework.

3.2 But What is a *Slow Packet*?

Thus far in this report, we have defined slow packets conceptually using the idea of a cut-off time. In this section, we give a more concrete definition of both slow packets and the cut-off time.

In order to define slow packets, we first need to define the cut-off time as the arrival time of k-th packet after the recipient receives the first packet of the stream of packets for the image. The value k denotes the number of packets required to achieve a close reconstruction of the pre-transmitted image. We quantity this *closeness* based on the MSSIM score because of MSSIM's correlation with human perception as detailed in Section 2.2.4.2. The inpainted image P' and the original image P are close if:

$$MSSIM(P,P') \geq 0.70 \tag{3.1}$$

While there is a previous study by Zinner *et al.* [59] that presented a mapping from the objective MSSIM scores to the subjective MOS scores. However, it is not suitable for this work on still images, as the mapping was primarily based on videos, which have other factors to take into consideration, such as frame rates.

Thus we decided to define a suited closeness threshold based on the existing literature on human tolerance on image degradation. It is reasonable to draw an analogy to image degradation because the quality of the images should reduce with a small k as there is less information available for inpainting when k is small. Tadros *et al.* demonstrate that human observers are robust towards large degrees in many types of image manipulation. The work by Wang *et al.* in [47] presents a range of images with various degrees of distortion and their MSSIM scores. Combining this information, we propose a value of 0.70 as an appropriate closeness threshold for this work, as it reflects humans' relatively high tolerance to image distortion.

It is necessary to have this closeness threshold when creating the performance illusion in image transmission - the end-to-end latency could be perceived as shorter when the recipient renders an image close to the original image without waiting for all the image information to arrive. We define slow packets in terms of the value k. Slow packets are *any remaining packets that are yet to arrive after the recipient receives the k-th packet*. The value k is to be determined empirically in our experiments in Chapter 5.

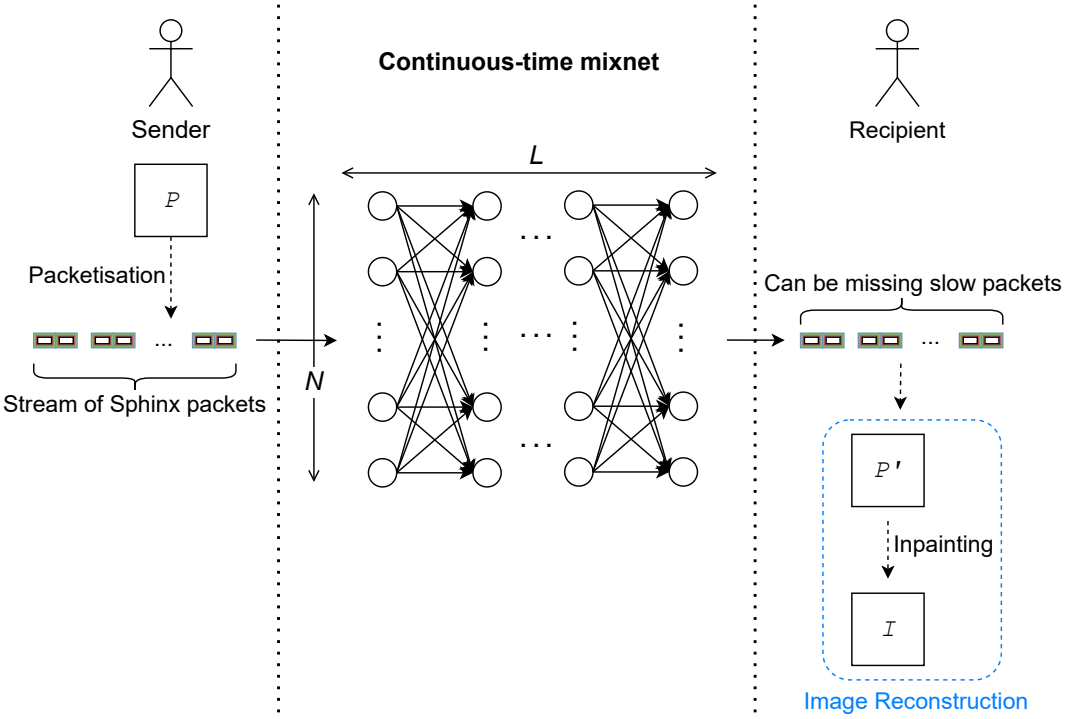


Figure 3.2: An overview of the integration of the framework into the mixnet infrastructure with a $N \times L$ mixnet, and the workflow of transmitting an image P from the sender to the recipient through the stratified mixnet. For simplicity, the cryptographic transformations on the packets are omitted as the overheads they incur are accounted in per-mix delays.

3.3 Framework Design

3.3.1 Framework Integration

Figure 3.2 depicts the mixnet infrastructure with the image reconstruction framework integrated at the recipient. As the integration is beyond the actual mixnet, this additional framework does not alter the functionality of the existing mixnet infrastructure. We now give a brief description of the workflow of image transmission with this framework. Since this project focuses on the image reconstruction component, the implementation details of aspects other than image reconstruction is considered out-of-scope and are assumed. To transmit an image P between sender and recipient, the sender would prepare the (Sphinx) packets to transmit across the mixnet. Packet preparation begins with image data segmentation. Likewise to standard networking protocols, data packetisation segments image data into a stream of packets according to the packet payload size. For each packet, the sender individually samples every per-mix delay and cryptographically encrypts the headers and the payloads in layers for the mixes on the packets' routes. The stream of packets is forwarded into the mixnet. Once the first packet of the stream is received, the recipient waits for a certain period of time based on the value of μ for more packets to arrive. Once the waiting period elapses, the recipient ceases waiting for further packets. In the case of missing image information due to slow packets, the recipient begins the image reconstruction process.

3.3.2 Design Factors of the Image Reconstruction Framework

The appropriate image reconstruction framework should be fast and lightweight, whilst achieving high image fidelity. Thus the framework design should consider the subsequent factors to enhance the adaptability of the framework and the performance of the mixnet in terms of reduced latency.

1. The image fidelity of the inpainted images with their original pre-transmitted images is achieved by the inpainting methods. The inpainted image should be *close* to the pre-transitted image in terms of the MSSIM score according to Equation 3.1.
2. The timing overhead incurred when performing the inpainting process
3. The hardware requirement to execute the image reconstruction framework.
4. The conditions under which leveraging the framework is beneficial to reduce the overall latency. Specifically, the number of packets out of the total number of packets the recipient waits for before starting the inpainting process.
5. The packet preparation process that aids the inpainting process. In particular, the way the rows of packets are placed into different packets determines the structure of the effective pixels during the inpainting process and, thus, the outcome of the process.
6. The nature of the inpainting algorithm. Inpainting algorithms can be categorised into two general categories: traditional and deep learning-based, each with its strengths and shortcomings.

3.3.3 Possible Inpainting Algorithms

As mentioned in Section 3.3.2, there are generally two categories of inpainting algorithms, traditional and deep learning-based in the field of computer vision. We investigated three inpainting algorithms: (1) NS, (2) TELEA, and (3) DeepFillv2. The former two are traditional inpainting methods while DeepFillv2 is deep learning-based. We selected these algorithms primarily for their availability. The NS and TELEA algorithms are natively supported in the OpenCV library, the largest open-source computer vision library[1]. Therefore both algorithms can be easily integrated into the image reconstruction framework pipeline. DeepFillv2 is a state-of-the-art image inpainting model proposed by Yu *et al.* [54].

As part of our preliminary studies, we explored several published frameworks such as the Masked Autoencoder [20] and SimMIM with the Swin transformer [50], which are approaches used for self-supervised pretraining of computer vision models. These frameworks are used in the pretraining stage to learn the features in the images. The models pretrained on these two frameworks have achieved promising results in numerous benchmarks for downstream image-related tasks such as image classification [58]. This is because the self-supervised pretraining objective consists of masking parts of the image which Masked Autoencoder or SimMIM restores during training. While these

[1] `https://opencv.org`

frameworks have general-purpose functionality which can be finetuned for downstream tasks, it is evident that the stated self-supervised pretraining objective is essentially equivalent to the objective of our image reconstruction framework. Hence we also explored these possibilities in the earlier stages of the project.

3.3.4 Why DeepFillv2?

After some experimentation, we decided not to further pursue the Masked Autoencoder and SimMIM frameworks for the following reasons.

Masked Autoencoder employs a typical pretrain-finetune architecture. For the inpainting task, we are interested in the pretraining stage where masked images are reconstructed to learn image features for future downstream tasks. Thus, we would need to perform the pretraining stage on Birds525. However, the pretraining stage requires large volumes of data, as the pretraining was conducted on more than 1,200,000 images in the original paper [20]. Due to the smaller dataset size of Birds525, the pretraining would be prone to overfitting, which would affect the image reconstruction results. Thus, we deemed the Masked Autoencoder unsuitable for this project.

For SimMIM, we attempted to integrate the open source codebase of SimMIM[2] into our experiment framework pipeline for the image reconstruction framework. However, we encountered some technical issues in the process, and there was only limited support available through the repository's *Issue* on the GitHub page. With the tight constraint of this project timeline, we opted to explore an alternative model in order to continue progressing through the project.

In terms of DeepFillv2, we selected it as the deep learning method to investigate based on two reasons: the computer vision task it was designed to solve and the integrability of the existing codebase into our own frameworks. The DeepFillv2 framework was designed specifically for the image inpainting task, Therefore it is already recognised as an advanced image inpainting technique, contrary to the two aforementioned frameworks. Furthermore, the authors maintained and provided much technical support in the "GitHub Issues" page of their codebase[3]. Despite the reduced online presence of the authors in the past year, most technical issues we encountered during integration were addressed through the extensive additional information provided by previous interactions on the *Issues* page. Thus we were able to successfully incorporate the deep learning module for further exploration.

[2] `https://github.com/microsoft/SimMIM`

[3] `https://github.com/JiahuiYu/generative_inpainting`

Chapter 4

Experiment Framework Methodology

This chapter outlines the experiment setup, including the simulation and environment assumptions, and the methodology for determining the suitable inpainting algorithm for the image reconstruction framework.

4.1 Experiment Assumptions

For the purposes of the simulation, we consider the case of one sender and one recipient, participating honestly in the mixnet. The mixnet through which the packets traverse before reaching the recipient has a Loopix-style mixnet structure with balanced layers. That is the mixes are arranged in stratified layers with an equal number of mixes per every layer. The mixes are physical machines with measurable computational power which ensures bounded unwrap time. Since this work focuses on information reconstruction on the recipient end, the mixnet and all its participants are assumed to be honest and without corruption. This assumption is reasonable as there is prior literature on detecting and excluding corrupted mixes from the mixnet [28]. Therefore, any adversarial activities in the mixnet are beyond the scope of this work. The existence of slow packets is solely due to the longer per-mix delays selected during packet preparation at the sender. The rows of pixels carried by the slow packets would be replaced by placeholder values. The inpainting process would replace those values with more meaningful values. We speculate that our reconstruction approach can be applied to the situation where the slow packets are caused by adversarial activities in the mixnet. However, the exact investigation is left as future work.

In terms of the images in the experiment, we assume that the images are transmitted without compression to ensure all images in the experiment have the same file size. This is because the image file size influences the number of packets required to transmit the image from the sender to the recipient. Having the assumption of identical image file size for the experiments justifies using the same number of packets to transmit them through the mixnet in our experimental setup. In reality, it is true that images are exported into image formats such as JPEG or PNG before transmission, which results in varying file sizes, thus the number of packets required for image transmission could differ by each image. The aforementioned assumption for uncompressed images

(a) Block-based (b) Interleave-based

Figure 4.1: Visualisation of (a) block-based and (b) interleave-based inpainting masks simulating 25% of packets as slow packets.

is necessary to simplify the experiment conditions in order to focus on investigating the inpainting process. We leave extending the framework to support image format-compressed images as future work.

4.2 Packetisation Schemes and Image Information Loss Models

Information loss models depict the manner in which missing information is displayed using inpaint masks. The selected three inpainting algorithms (outlined in Section 3.3.3) require a binary inpaint mask to indicate the masked region. In our case, we draw an analogy between the masked regions and the missing regions carried by the slow packets. The regions marked as unknown in the inpainting mask are the pixels to be reconstructed. The proportion of masked regions of the image is directly related to the number of slow packets. Thus, this is the foundation of our information loss model: the missing pixels are represented by using a binary mask to overlay onto the experiment images; with more slow packets, the areas of masking on the image increase.

As mentioned in Section 3.3.2, packet payload could be arranged such that the inpainting algorithm can perform well from the available information. The structure of an unknown region displayed at the recipient is heavily dependent on the arrangement of image information during the packet preparation process at the sender. We propose two information loss models, arising from the two different packetisation schemes: blocked-based and interleave-based. Block-based packetisation schemes lead to block-based information loss, and interleave-based packetisation schemes lead to interleave-based information loss. During packetisation, we operate in units of rows of pixels as detailed in Section 3.1. The block-based packetisation means that we partition such that contiguous rows of pixels are put into the same packet. The interleave-based packetisation, as the name implies, spreads out the rows of pixels in an alternating manner across the packets. Specifically for N number of packets, we label the packets with packet number n where $n = 0, ..., N-1$. The packet p_i of which the i-th row is allocated to is determined by: $p_i = i \bmod N$, where $\bmod$ denotes the modulo operation. Thus, for a given percentage of slow packets, the loss models have distinct displays of missing pixels from their respective masks. Figure 4.1 visualises the inpaint masks of the missing pixels with 25% of slow packets.

4.3 Experiment Data

We decided to sample a selection of images with similar file sizes from the training set of Birds525, which are referred to henceforward as *experiment images*. The experiment images are a set of non-overlapping images with the remaining images in Birds525. We created this separate set of images for the following reasons:

- We can set a number for the total number of packets for the simulation, as an experiment framework parameter.
- Limited storage on the test machine - for every image, we will make several duplicates with different masking and inpainting techniques applied, increasing the total number of images to store for recording purposes.
- For the deep learning inpainting method, finetuning the model would need a large image dataset. We can regard the experiments as additional testing for the deep learning model, so the model cannot be trained on the images used in the experiments. For the experiments, we would need a separate set of images besides the test subset of images in Birds525 as good machine learning practice.

Considering the storage limitation on the test machines and the need for a large dataset for finetuning the deep learning mode, we selected 107 images to set aside as the experiment images. The images are selected randomly as they all have the same uncompressed file sizes and are curated under the same settings specified in Section 2.4. The experiment image set is 107 based on our estimates of storage requirements on the Laptop for a large number of images over several iterations of the experiments. Notice that for the 107 experimental images and 7 possible numbers of slow packets (from one to at most seven out of eight total packets) with two masking schemes, there are $107 \times 7 \times 2 = 1498$ images plus their respective masks and masked images. This number of images is suitable as it is sufficiently large for future statistical analysis with the normality assumption under the central limit theorem.

4.4 Test Machines

The experiments are conducted on the following machines, they are selected because of their range of high and low computational power which represent the possible machine configurations that could be used as a recipient. We run the same experiment configuration on all the machines for comparable results. The machines are:

1. AMD EPYC 7302 16-core processor @ 3.29 GHz with 512 GB of system memory.
2. AMD EPYC 7302 16-core processor @ 3 GHz with 192 GB of system memory, and a NVIDIA GeForce RTX 3080 graphics card with 10 GB of memory.
3. Apple M2 8-core processor @ 3.29 GHz with 16 GB of system memory.
4. Raspberry Pi 3 Model B @ 1.2 GHz with 1 GB of system memory.

For the remaining of the report, we refer to test machine 1 as *DICE (SC)*, test machine

2 as *Lab machine*, test machine 3 as Laptop and test machine 4 as *Raspberry Pi*. DICE (SC) and Lab machine are categorised as powerful compute-intensive machines, while the Laptop is a standard daily usage machine, and the Raspberry Pi is a low-compute intensity microcomputer. Note that the Laptop is the primary test machine, except for experiments for DeepFillv2 (see Section 4.5.5 for detailed reasoning) and the runtime experiments, which are conducted on several test machines. It is reasonable to compute the inpainted images on a single machine since the inpainted outcome of the algorithms are not hardware dependent.

4.5 Experimental Setup

4.5.1 Uncompressed image file size

With the assumption of uncompressed images of identical file size, it is reasonable to use the same number of packets for all experiment images. The file size of uncompressed images depends on its image dimensions and bit depth. Bit depth b is defined as the number of bits required to represent a pixel. Thus file size S of an uncompressed $M \times N$ image in bytes is given by[1]: $S = \frac{M \times N \times b}{8}$. To calculate the file size of the experiment images, we first need to confirm their bit depths. We extracted the bit depth per colour channel for each image by inspecting the image file metadata using the `python-magic` library[2]. We confirmed that all images are 3-channelled with 8 bits of bit depth per channel, thus the overall bit depth b is $3 \times 8 = 24$ bits. Recall that all images in Birds525 have pixel dimensions of 224×224. The uncompressed file size is $S = \frac{224 \times 224 \times 24}{8} = 150,528$ bytes.

4.5.2 Number of packets and payload size

The payload size determines the number of packets required to transmit an image. In this work, The number of rows in an image is directly derived from the image's pixel dimension in height. Since our information loss models are related to the rows of pixels, the value for the number of packets, and hence the packet payload size, needs to be based on the number of pixel rows. In order to achieve representative results from the experiments, we need to use a sufficient number of packets to simulate the transmission. Having a stream of packets for an image, we can investigate the impact between various proportions of slow packets on the inpainted result.

Jee *et al.* [24] proposed that a sphinx packet payload size of 4096 bytes provides a good balance between anonymity and the performance of the network. Anonymity is defined as the indistinguishability of a packet among all the packets traversing through the network. The performance of the network is measured by the goodput which is defined as the rate of useful information traversing the network excluding the network overhead [24]. However, image file size easily reaches beyond 4 KB for a small monochrome from our preliminary exploration, we argue that this suggested payload size is not applicable to this work as we are working with uncompressed images. Uncompressed

[1] `http://preservationtutorial.library.cornell.edu/intro/intro-06.html`
[2] `https://pypi.org/project/python-magic/`

images are known to have larger file sizes, which require more packets to transmit from the sender to the recipient for a payload size of 4 KB. Furthermore, the payload size suggested by Jee *et al.* is not based on the number of pixel rows, thus it is not well-suited for the work here.

Therefore, we propose a packet number (and thus payload size) based on the pixel dimension in height that is suitable for this work. After some exploration with different packet numbers, we decided that 8 is a suitable number of packets for this work. It results in a payload size of $\frac{150{,}528}{8} = 18{,}816$ bytes. We give the following reasons for such a decision for packet number:

- 8 is a factor of 224 which is the height of the image in pixels. Thus, the image data would partition evenly across all the packets, and every packet contains the same number of rows. Given one slow packet in the sequence of packets, the number of valid pixels lost is invariant regardless of which packet in the stream is actually the slow packet.
- Eight packets gives an adequate range of possible number of slow packets to investigate in the experiments.

While using 8 packets to transmit an uncompressed image is sufficient in this work, we acknowledge that different media are likely to require different numbers of packets for efficient transmission. The implemented experimental framework is directly reusable with some adjustments for the media type and modification to the total number of packets during experiment initialisation. However, since the project's focus was uncompressed images, we did not investigate other packet numbers and it is left as future work.

4.5.3 Structure of Experiment Framework

For our experiments, it is not required to build the experiment framework on a mixnet network simulator. Recall that while the slow packets are incurred by the mixnet, the image inpainting process takes place at the recipient rather than within the mixnet infrastructure. Thus we are interested in image recovery with the received packets after the mixnet network at the recipient.

The experiment framework is a pipeline with components implemented in Python, the experiment pipeline is repeated for different numbers of slow packets, which can in turn determine a specific value of k (as defined in Section 3.2) that gives reconstructed images close to their original. Partitioning every experimental image into 8 packets, there are at most seven slow packets, since the recipient cannot inpaint on no data at all - they must wait for at least one packet.

Subsequently, for all possible number of slow packets from 1 to 7, the experiment is outlined as such:

1. We create monochromatic block-based and interleave-based inpaint masks and write them to file locally for future reference.
2. The created masks are overlayed onto the experimental images using bitwise AND operation. We refer to these images as the *masked images*. Thus, it results

(a) Original (b) Block-masked (c) Interleave-masked

Figure 4.2: A selected image with filename '485.jpg' and its block- and interleave-based masked images when the number of slow packets is 3.

in two sets of seven masked images for every experiment image for blocked-based and interleave-based masking respectively; each image corresponds to a certain number of slow packets. Figure 4.2 illustrates an example of the masked images.

3. We apply the three inpainting methods onto each masked image and the resultant images are referred to as *inpainted images*.

The simulation for the traditional inpainting algorithms is run by executing `image_comparison_v3.py`, specifying the path to the experimental images, the output directories for the masked images, their masks and the inpainted images. The Python script first performs steps 1 and 2 to create the masks and the masked images. Then for step 3, the script executes the traditional inpainting algorithms. For the deep learning-based method, we use a separate script for the simulation as the deep learning model is implemented separately. The simulation of deep learning-based inpainting on the image is run by executing `./multiple_images_test.sh` in the terminal, specifying the paths to the masked images and their masks, and the output directory of the inpainted images. Here, we are not creating the masked images and their corresponding masks again. This is because recall that the masked images and masks are created during the simulation for the traditional inpainting algorithms. We pass the created masked images and masks to the deep learning simulation, in order to ensure that the deep learning-based algorithm inpaints on the same set of masked images and masks.

4.5.4 Design of Experiment Components

This section outlines the design of the key components of the experiment framework. The exact implementation and any relevant modifications to the provided codebase can be found in the submitted code. For block-based masks, we computed the contiguous area of rows that needs to be masked, based on the proportion of slow packets out of the total number of packets. One masked region denotes the information of one slow packet, thus Figure **??** contains two masked regions as there are two slow packets. Recall that the per-mix delays are sample randomly, thus the masked regions are also selected randomly because there is a uniform likelihood of any packets in being a slow packet. For interleave-based masks, to ensure that adjacent pixel rows are put into different packets, the interleaving is achieved according to the description in Section 4.2. We selected pixel rows belonging to the received packets (`total_packets - slow_packets`) and

set the remaining pixel rows as masked regions in order to create the mask.

In terms of traditional inpainting methods, for the NS and TELEA algorithms, they are directly invoked through the `cv2.INPAINT_NS` and `cv2.INPAINT_TELEA` flags in the `inpaint()` function of the OpenCV for Python library. The provided inpaint function requires both the masked image and its corresponding monochromatic mask as parameters to operate. For the deep learning-based inpainting method, the original Tensorflow implementation for DeepFillv2 is available in the authors' Github repository. It includes the script (`test.py`) to apply the model to inpaint a masked image. We incorporated the script into a bash script (`multiple_images_test.sh`) to perform inpainting for all experiment images. The full bash script is available in Section A.1.

4.5.5 Practical Issues during Setup

There were some technical issues to overcome when preparing the experimental framework. In particular, we encountered package availability issues when setting up the part of the environment framework for DeepFillv2. This is because the provided codebase for DeepFillv2 has been frozen; it is only compatible with Tensorflow version 1, which is supported until Python version 3.7. Moreover, it was advised to run the given codebase on Python 3.6[3]. This gave rise to the first issue to overcome: the system requirement for DeepFillv2. Python 3.6 is only available natively on x86 architecture systems. As listed in Section 4.4, the test machine Laptop has the Apple silicon M2 processor which uses the ARM64 architecture. From our investigations, there is no native build of Python below version 3.8 available for Apple silicon processors. The system requirement issue was resolved by the use of Rosetta 2 to emulate the x86 architecture on the Laptop, thus earlier Python versions could be installed. However, we then faced the second issue: disabled Python packages. Since the end-of-life phase of Python 3.6 was reached at the end of 2021, package managers such as Homebrew have disabled the related Python packages. The installation of Python 3.6 is no longer available through official channels as we encountered error messages of deprecated upstream. We also encountered system requirement issues when setting up DeepFillv2 on the Raspberry Pi as the deep learning model required resource-intensive packages that are not available for the Raspberry Pi. Therefore, we decided to perform the experiments related to DeepFillv2 on the Lab machine as it is an x86 system where we were able to set up the provided DeepFillv2 implementation without errors. For the runtime experiment, we were also able to set up DeepFillv2 on DICE (SC), as it has a similar system configuration to the Lab machine.

4.6 Evaluation Metric

As the evaluation metric for image fidelity between the inpainted image and the pre-transmitted image, we are using the mean Structure Similarity Index (MSSIM) because of its consistency with various distortion types [29]. MSSIM is a full-reference objective metric that shows a correlation with human visual perception. In the following section, we give reasonings for using an objective metric over subjective human assessors.

[3] `https://github.com/JiahuiYu/generative_inpainting/issues/105`

4.6.1 Motivation for Objective Quality Metric

Historically both objective and subjective quality assessment metrics are used to measure the performance of various inpainting methods [8, 29, 51]. Subjective viewing tests are a type of human evaluation known for their effectiveness; however, they come with additional challenges since a number of conditions need to be controlled. These include test conditions such as lighting, image size, viewing angle, viewing distance and duration of viewing [55]. Additionally, variation in participant conditions needs to be factored in such as physical health, psychological differences, and the background of the participant [55]. Maintaining these conditions is expensive, a cost which is added to by the need for multiple human participants and repeated viewing sessions of various image distortions with specialised equipment [29]. It is important to include some post-assessment processing to aggregate the results and evaluate the inter-participant agreement for every test condition to enable further analysis [29, 55]. The aim of objective metrics is to reflect the image or video quality such that the scores align with human perception of quality [41]. Using an objective metric overcomes the technical challenges inherent in subjective methods: it intrinsically enables repeatability and allows consistent statistical analysis without further processing [29].

4.6.2 Use of MSSIM in the Experiment

We used the MSSIM formulation where the per-pixel SSIM has default parameters of $\alpha = \beta = \gamma = 1$, as it gives a good balance between the luminance, contrast and structure functions. It results in Equation 2.2 in Section 2.2.4.1. For every experiment image, we compute the MSSIM score in the following scenarios at the recipient.

1. **Pre-inpainting MSSIM score**: The MSSIM score between the original pre-transmitted image and the pre-inpainted image before the inpainting process.
2. **Post-inpainting MSSIM score**: The MSSIM score between the original pre-transmitted image and the inpainted image obtained by the inpainting process.

Choice of MSSIM implementation There are several open-source Python packages that implement the MSSIM, namely `sewar`[4], a package specifically for image quality metrics, and `scikit-image`[5] (also known as `skimage`), a package for a wide range of image processing functionalities. In the early stages of the experiments, we opted to use the implementation provided in `scikit-image` as the library is popular among its large user base as well as peer-reviewed code so that the code base is high quality and consistent. However, we encountered dependencies issues when setting up the experimental experiment on the Raspberry Pi due to hardware limitations. Since we are only using the MSSIM functionality, we do not need all the functionalities offered by `scikit-image`. To maintain a consistent experiment environment, we switched to using `sewar` package, which is more lightweight as it only contains image quality metrics. The package is actively being maintained and has at least one million installations, thus it is also a popular package and suitable for this project.

[4] `https://pypi.org/project/sewar/`

[5] `https://scikit-image.org/docs/stable/api/skimage.html`

Chapter 5

Experiments

This chapter details the experiments conducted to address the design factors in Section 3.3.2.

5.1 Image Fidelity Experiments

5.1.1 DeepFillv2 Method

This section details the experiments conducted to optimise the DeepFillv2 model in order to produce high-fidelity inpainted images.

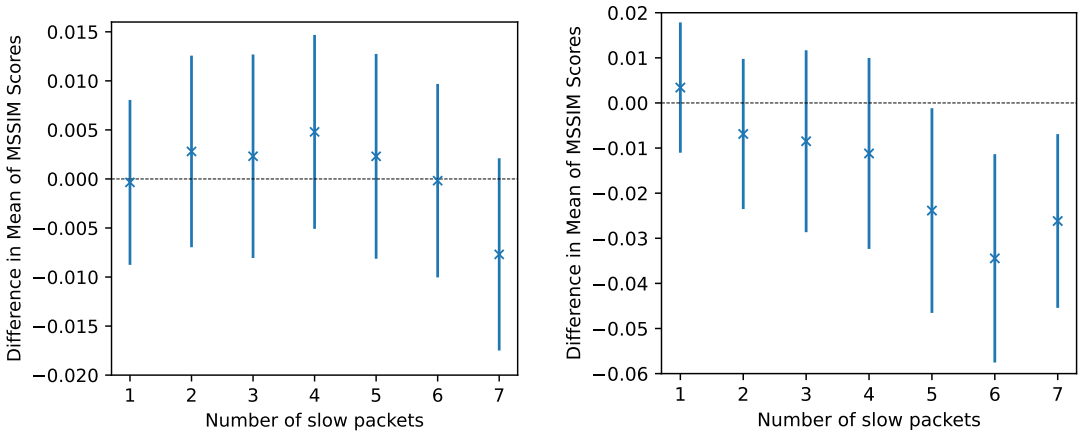
5.1.1.1 Exploring Pretrained Models

This experiment compares the inpainted results on the experimental images using the provided pretrained models of DeepFillv2. The authors of DeepFillv2 released two sets of model weights pretrained on two commonly image datasets Places2 [57] and CelebA-HQ [25] on their GitHub repository. Places2 consists of more than ten million images of different categories of natural scenery and man-made constructions, and CelebA-HQ is a dataset of 300,000 human faces.

5.1.1.1.1 Experiment Workflow To run the inpainting process with the two pretrained models, we specified the checkpoint directory of the command line argument for `multiple_image_test.sh` to the path to the respective model weight directory. The inpaint model was required to inpaint on two masking schemes for every image, blocked- and interleave-based (Section 4.5.3, Step 2). Thus, there were two sets of inpainted images for each pretrained model. For the sets of inpainted images, the post-inpainting MSSIM scores are computed and are averaged to give the overall mean for each number of slow packets. In order to assess whether a model significantly outperforms the other model in terms of the average MISSIM scores, we computed the difference in the mean scores obtained by the models and their 95% confidence intervals (CIs). The CIs of the differences between the means give more robust statistical interpretations; they give lower false negative rates than the CIs of the mean MSSIM obtained by each model.

Number of slow packets	Block-based		Interleave-based	
	CelebA-HQ	Places2	CelebA-HQ	Places2
1	0.836	0.836	0.894	0.891
2	0.734	0.731	0.837	0.844
3	0.625	0.623	0.767	0.776
4	0.517	0.512	0.700	0.712
5	0.401	0.399	0.610	0.633
6	0.283	0.283	0.506	0.540
7	0.161	0.169	0.314	0.341

Table 5.1: The mean of the post-inpainting MSSIM scores of block- (left) and interleave-based (right) masks achieved by the CelebA-HQ and Places2 pretrained models, for different numbers of slow packets.



(a) Block-based

(b) interleave-based

Figure 5.1: The difference in mean post-inpainting MSSIM scores between CelebA-HQ and Places2 models on the inpainted images (denoted with '×') for a various number of slow packets under block-based (5.1a) and interleave-based masking (5.1b). Error bars represent the corresponding 95% confidence intervals. The horizontal dashed line denotes no difference between the mean scores obtained by the two models.

5.1.1.1.2 Results Table 5.1 shows the average post-inpainting MSSIM scores. Figure 5.1 visualises the differences in the means of the MSSIM scores between the models with their CIs. The pretrained models were employed on the masked example images in Figure 4.2 in Chapter 4. The results are displayed in the Section A.2.3.

5.1.1.1.3 Discussion For both Places2 and CelebA-HQ models, the mean post-inpainting MSSIM scores under the 'Block-based' column in Table 5.1 decreased in even intervals as the number of slow packets increased. This indicated that the reconstructed images of the block-based masking scheme had a steady decrease in fidelity despite the application of inpainting. Both models achieved similar mean post-inpainting MSSIM scores for block-based masked images. This was further confirmed

in Figure 5.1a as the range of CIs all included 0. The findings in Figure 5.1a also agreed with the 2-sampled t-test at 5% significance level, where the p-values were mostly hugely greater than 0.05 (See Table A.1). Thus, there was strong evidence at 95% confidence level that neither model on average produced significantly better inpainted results when inpainting with block-masked images. Meanwhile, the Places2 model mostly outperformed the CelebA-HQ model on average post-inpainting MSSIM score of the interleave-masked images. In the case of inpainting interleave-masked images, we found that the average post-inpainting MSSIM scores for interleave-masked images were higher than the ones for block-masked images. The increase in the average scores was particularly noticeable as the number of slow packets reached beyond 3. The mean MSSIM scores increased by 34% and 37% from the scores for block-based masking for CelebA-HQ and Places2 models respectively, when there were 4 slow packets. The percentage increase of the mean MSSIM scores continued and eventually doubled when the slow packet number was 7. The full breakdown of the percentage increases is in Table A.2. Regarding the closeness of the inpainted images and their original images, an inpainted image was considered to be close to its original when the post-inpainting MSSIM score was at least 0.70 (Equation 3.1). Therefore, the pretrained models only reached the closeness threshold when the number of slow packets was 1 (Table 5.1). Thus, the latency was shortened by the time taken for 86% of packets to arrive, given there were 8 packets in total.

The limitations in using the Places2 and CelebA-HQ models on the experimental images are as follows. The visual inpainting outcomes on Figure 4.2a displayed some large patches of incorrect colours and undesirable artefacts. In terms of the data distribution, the two pretrained models were trained on various scenes and human faces, while the model was tested on experimental images which were images of birds. The domains of the images used for training and testing the models were not the same. Drawing both training and testing data from the same distribution would give more representative results. This gave the motivation for finetuning. In the case of testing on the different domains of images, it is common to finetune the pretrained model weights to the specific image domain. In the next sections, we attempted to finetune a set of DeepFillv1 model weights using the Birds525 dataset.

5.1.1.2 Finetuning ImageNet-Pretrained Model on Birds525

Finetuning a model to a specific task is a fundamental deep learning technique that could optimise the model performance during testing. The datasets Places2 and CelebA-HQ were of very different domains compared to Birds525. The authors of DeepFillv1 released the pretrained model weights on the ImageNet [38] dataset. Thus, the purpose of this experiment was to obtain an adapted set of model weights based on the Birds525 dataset.

Whilst the authors had demonstrated some realistic inpainting on images of birds using the ImageNet DeepFillv1 model in [54], the ImageNet model weights were not directly usable on the DeepFillv2 model. This was because the shape of the model weights of DeepFillv1 was not equal to the shape required for DeepFillv2[1], so we could not

[1] https://github.com/JiahuiYu/generative_inpainting/issues/345

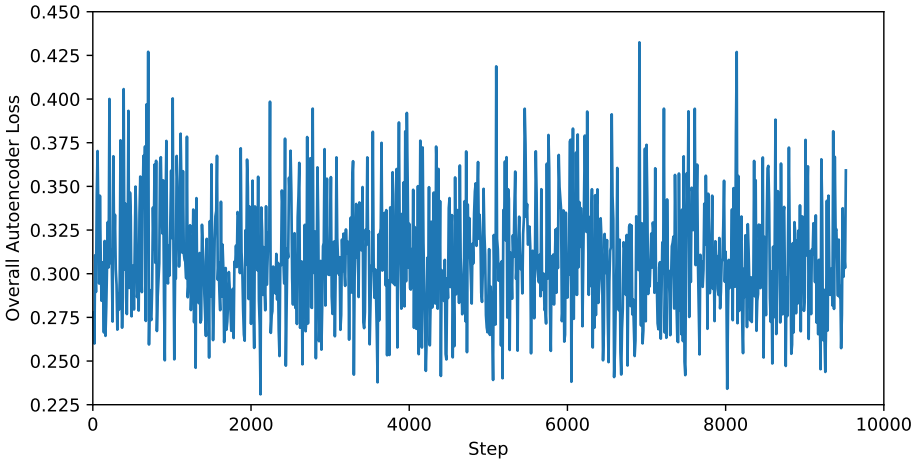


Figure 5.2: The overall autoencoder loss for finetuning the ImageNet model weights with loss ratio 1:1 using the Birds525 dataset.

independently verify the inpainting outcome of ImageNet model weights on DeepFillv2 before finetuning. However, We chose to work with the ImageNet pretrained model weights as there are selections of bird images in the ImageNet dataset, so ImageNet images are of a closer domain to Birds525 than Places2 and CelebA-HQ. We deemed that the ImageNet model weight was a good starting point for this experiment. Thus, the ImageNet model weights were used for bootstrapping the model initialisation for finetuning, where we supplemented the learnings from a well-performing model to bird-specific images of Birds525.

5.1.1.2.1 Experiment Workflow The Birds525 data used for the finetuning process was the set of training images (non-overlapping with the experimental images) and the validation images. Thus we could assert that the model had been trained with data of the same distribution when testing the inpainting process with the experimental images.

As part of the provided codebase, there is a Python script `train.py` which provides both train and finetune functionalities (differentiated by the values in the configuration). In order to finetune the ImageNet model, there were some modifications to the original DeepFillv2 configuration. The parameter configurations for pretraining (also referred to as training in the DeepFillv2 code) and finetuning, and the model hyperparameters were specified in the provided `inpaint.yml` file. For this experiment, we used the default configurations as detailed in the `inpaint.yml` file since previous pretrained models under those configurations provided high-quality inpainted results. The full configuration file was given in Section A.3.1. The path to the model to finetune from was modified to the ImageNet model directory as specified in the instructions on the GitHub repository. To begin the finetuning process, we ran the provided `train.py` which initialised the model and training parameters as specified in the configuration file. The finetuning process took approximately 157 hours on the Lab machine.

From the information provided for DeepFillv2 [53, 54] and its supplementary Github repository, the implementation of DeepFillv2 does not include an explicit metric to measure model convergence. It is advised to stop the finetuning process when the losses

used in the model converge. The loss function relevant to our finetuning process was the overall autoencoder loss of the entire image. It denoted the overall reconstruction loss and was defined as the sum of the pixel-wise l_1 loss in the masked regions and the SN-PatchGAN loss.

5.1.1.2.2 Results Figure 5.2 gives the curve of the overall autoencoder loss throughout the finetuning process.

The loss curve did not move to converge throughout the duration of the finetuning process. The inpainted results also contained many undesirable artefacts (see Figure A.3 for an example). We suspected that the finetuning had failed as the curve did not plateau towards convergence. Thus, this motivated the hyperparameter tuning in the next section.

5.1.1.3 Tuning Loss Ratio Hyperparameter

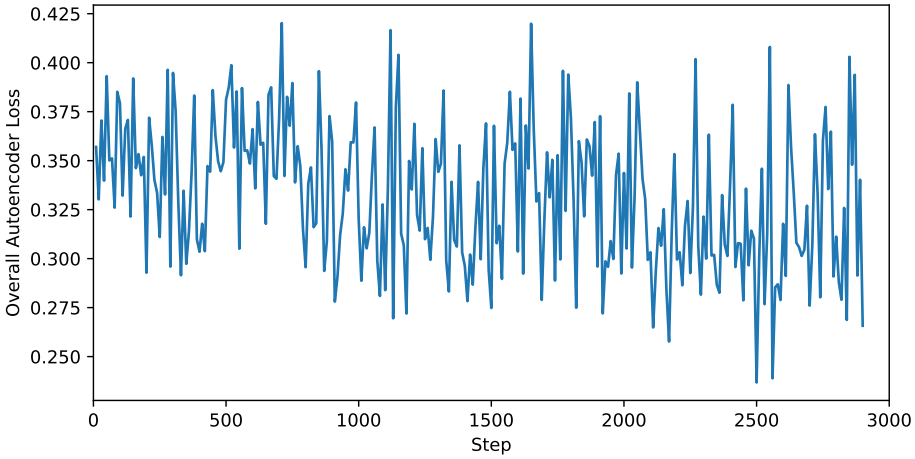
In this experiment, we adjusted the ratio that combined the pixel-wise l_1 loss and the SN-PatchGAN loss. The default was set to 1:1 in the configuration file. We found several discussions posted by other researchers on the DeepFillv2 Github repository on tuning such ratio by reducing the parameter for the SN-PatchGAN on a trial-and-error basis to produce results with fewer artefacts.[2,3]

5.1.1.3.1 Experiment Workflow We experimented with two different ratios on the finetuning process by reducing the parameter for the SN-PatchGAN; the ratios were 0.5:1 and 0.9:1. We also tested the ratio 1:1 again in order to verify whether the previous finetuning failed was due to the randomness of the finetuned model which was particularly sensitive to the experimental images. The finetuning process took approximately 28 hours, 107 hours and 107 hours for ratios 0.5:1, 0.9:1 and 1:1 respectively.

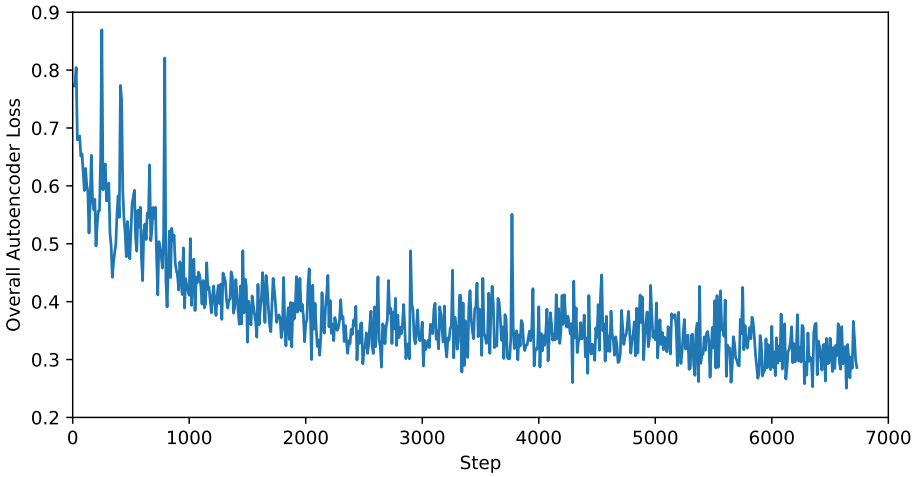
5.1.1.3.2 Results Figure 5.3 illustrates the overall autoencoder loss for the aforementioned loss ratios. The finetuning process for ratio 0.5:1 was stopped at around 3000 steps as it was clear that the model had failed from the non-converging autoencoder loss curve in Figure 5.3a. Figures 5.3b and 5.3c show that the model loss for ratios 0.9:1 and 1:1 had converged as the curves plateaued. The loss curves indicate that the models had been successfully finetuned.

We tested the inpainting process with the experimental images using those two models. The mean of the MSSIM scores achieved by the models are given in Table 5.2. Comparing with scores achieved by the CelebA-HQ and Places2 model in Table 5.1, the two pretrained models mostly outperformed the Birds525-finetuned ImageNet model. While the finetuned ImageNet model achieved higher average MSSIM scores when there were 6 and 7 slow packets with block-based masking, the scores were lower than the closeness threshold of 0.70. Thus, we concluded that finetuning an existing model to the Birds525 specific task was not more suitable as the inpainting algorithm for the image reconstruction framework.

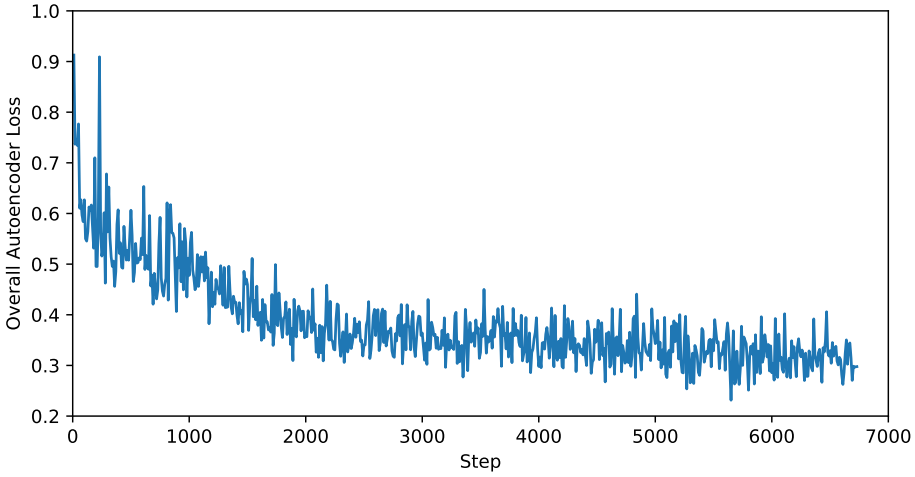
[2] `https://github.com/JiahuiYu/generative_inpainting/issues/421`
[3] `https://github.com/JiahuiYu/generative_inpainting/issues/267`



(a) 0.5:1



(b) 0.9:1



(c) 1:1

Figure 5.3: The overall autoencoder loss for finetuning the ImageNet model weights using the Birds525 dataset, for loss ratios 0.5:1, 0.9:1, and 1:1.

Number of Slow Packets	Block-based		Interleave-based	
	0.9:1	1:1	0.9:1	1:1
1	0.834	0.833	0.871	0.872
2	0.730	0.726	0.768	0.761
3	0.621	0.617	0.682	0.652
4	0.513	0.509	0.597	0.552
5	0.401	0.397	0.500	0.467
6	0.289	0.286	0.398	0.365
7	0.181	0.178	0.243	0.221

Table 5.2: The mean of the post-inpainting MSSIM scores of block-based (left) and interleave-based (right) masks achieved when the loss ratios are 0.9:1 and 1:1, for different numbers of slow packets.

We suspected that the lack of improved performance was caused by out-of-distribution finetuning - that is, the mismatch between the distributions of the data used in the pretraining and finetuning processes. Kumar *et al.* [26] demonstrated the significance of in-distribution when finetuning, as the finetuned models often underperform, particularly in the case of a large difference in the data distributions and high-quality pretrained features. In our situation, ImageNet contains a much wider range of categories such that it consists of only a small subset of bird images. Thus, there are possible dissimilarities between the pretrain and finetune data distribution. In the following section, we attempted to overcome the out-of-distribution issue by training the model from scratch using the Birds525 dataset.

It is also interesting to note that the model train with a loss ratio of 0.9:1 generally produced higher MSSIM scores than the one with a ratio of 1:1. It is possible that other loss ratios would result in MSSIM scores closer to the closeness threshold. However, it was not investigated further as there was a fundamental issue of out-of-distribution finetuning which would not yield good results.

5.1.1.4 Training on Birds525 from Scratch

This experiment trains the DeepFillv2 model from scratch and assesses whether the average MSSIM scores improve. Training the model from scratch had the advantage of learning features and representation for inpainting solely on the Birds525 dataset.

5.1.1.4.1 Experiment Workflow In the configuration file, we set the training data path to the training set of Birds525. The set of training images of Birds525 used did not overlap with the experimental images as it is good machine learning practice to test on unseen data. Despite the findings in the previous section, it was difficult to determine whether there were other factors in the finetuning that contributed holistically to the improved MSSIM values. Hence, we decided to leave the loss ratio as the default 1:1 since the model was being trained from scratch. To start the training process, we ran the `train.py` script. The training process took approximately 230 hours.

(a) DeepFillv2 on block-masked

(b) DeepFillv2 on interleave-masked

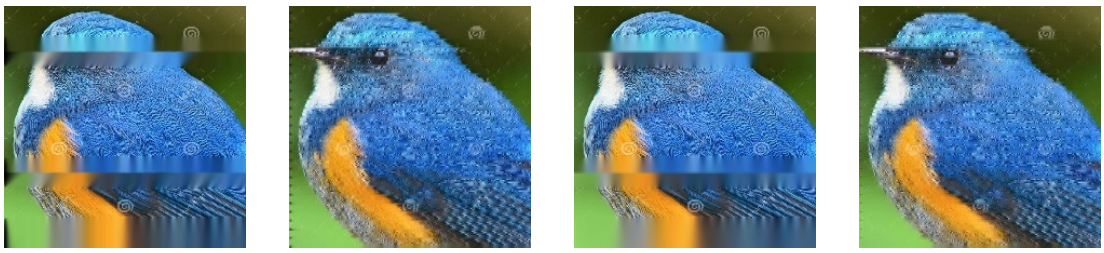
Figure 5.4: Various inpainted images of Figure 4.2 using the DeepFillv2 algorithm for the block-masked image (Figure 4.2b) on the left, and the interleave-masked image (Figure 4.2c) on the right.

Number of Slow Packets	Block-based	Interleave-based
1	0.835	0.882
2	0.731	0.802
3	0.623	0.720
4	0.517	0.638
5	0.406	0.526
6	0.290	0.401
7	0.174	0.263

Table 5.3: The mean of the post-inpainting MSSIM scores of block-based and interleave-based masks per number of slow packets, achieved by the DeepFillv2 model trained from scratch on Birds525.

5.1.1.4.2 Results The converged autoencoder curve is omitted here due to space limitation as it has a similar shape to Figure 5.3b; the actual plot can be found in Figure A.4. Figure 5.4 shows the inpainted results of the example image in Figure 4.2 of Chapter 4. Table 5.3 gives the average post-inpainting MSSIM score per number of slow packets for the two masking schemes.

In Figure 5.4a, DeepFillv2 successfully inpainted the scripts of masked regions whilst preserving the contours of the bird, thoigh there were some colour inconsistencies and missing features such as the beak. It was also apparent that there were clear markings in Figure 5.4b at the masked areas. Inspecting the figure closely, we found that while the model inpainted the correct type of colours such as a yellow colour for the masked regions around the valid yellow regions, the markings were caused by the wrong shades of the colours being inpainted. We hypothesised that the model produced these inconsistencies due to the difference in the masks used in the original training and in the testing here. The masks in the original training had a rectangular patch with



(a) NS on block-masked (b) NS on interleave-masked (c) TELEA on block-masked (d) TELEA on interleave-masked

Figure 5.5: Various inpainted images of Figure 4.2 using NS, TELEA, and DeepFillv2 algorithms. Figures 5.5a and 5.5c are for the block-masked image (Figure 4.2b), and Figures 5.5b to 5.5d are for the interleave-masked image (Figure 4.2c).

several other irregular patches, while the masks that were applied to the experimental images have rectangular patches spanning across the images. The mean post-inpainting scores reflected the visual outcomes. The inpainting on the interleave-based masking scheme reached the closeness threshold defined in Section 3.2. The features of the bird remained evident, albeit the blurriness. The inpainting on the block-based masking had lower MSSIM scores as the image no longer contained the beak nor the eye of the bird.

5.1.1.5 Conclusion on the DeepFillv2 Experiments

Based on the experiments performed on the DeepFillv2 model, we concluded the following findings. Firstly the interleave-based masking (hence the interleave-based packetisation scheme) produced consistently higher post-inpaint MSSIM scores for all possible numbers of slow packets. Secondly, the provided Places2 and CelebA pretrained models overall achieved the highest MSSIM scores, followed by the Birds525 model that was trained from scratch. We suspected it was because the pretrained models had the optimal model hyperparameters. However, we decided to use the Birds525 model for DeepFillv2 in subsequent discussions as it produced better inpainting visually as there was less colour inconsistency in the inpainted images.

5.1.2 NS and TELEA Methods

This experiment compares the image fidelity achieved by the traditional-based inpainting methods, in order to establish an inpainting method that was acceptable for the image reconstruction framework.

5.1.2.1 Experiment Workflow

The traditional methods are used directly as they, by design, do not require any additional processes for optimisation as DeepFillv2 in the previous sections. The average MSSIM scores per number of slow packets were computed as part of the execution of `image_comparison_v3.py`.

Number of Slow Packets	Block-based		Interleave-based	
	NS	TELEA	NS	TELEA
1	0.900	0.901	0.972	0.969
2	0.797	0.800	0.893	0.886
3	0.687	0.691	0.832	0.820
4	0.575	0.578	0.795	0.798
5	0.455	0.460	0.711	0.714
6	0.335	0.340	0.614	0.616
7	0.207	0.213	0.506	0.517

Table 5.4: The mean of the post-inpainting MSSIM scores of block-based (left) and interleave-based (right) masks achieved by the NS and TELEA algorithms, for different numbers of slow packets.

5.1.2.2 Results

Figure 5.5 gives the inpainted results by NS and TELEA algorithms on Figure 4.2a, for comparison purposes with the results of DeepFillv2.

In Figures 5.5a and 5.5c, the two algorithms performed worse visually than the DeepFillv2 model as the outline of the bird was heavily misaligned at the boundaries and regions of masked regions. Examining the MSSIM scores reported in Table 5.4, the traditional methods had higher average scores at smaller numbers of slow packets than DeepFillv2. This could be because the height of the masked regions was small which was advantageous for interpolating from nearby known pixels. We decided that the traditional algorithms were well-suited for up to 5 slow packets according to the closeness threshold and the visual outcomes, as the general features of the birds in the images remained distinguishable. A two-sampled t-test was performed between the scores for NS and TELEA under interleave-based masking; there was no significant difference between the algorithms at 5% significance level [(t(1496)=-0.04480, p=0.9643].

5.2 Runtime Experiments

This experiment compares the time taken for the NS, TELEA and DeepFillv2 models to complete the inpainting process, on the test machines listed in Section 4.4.

5.2.1 Experiment Workflow

We assumed that the recipient always loaded the inpainting algorithm prior to receiving image packets. Thus the model loading time was not part of the experiment. We defined the inpainting time as the elapsed time in seconds between the start and end times to perform the inpainting on a single experiment image through calling of the inpainting function of the inpainted algorithm. Specifically, the time at which the function to inpaint the image was called, is denoted as the `start_time`. Once the inpainting function finished, the time was recorded, marking the `end_time`. This timing mechanism was added to the experiment framework and was implemented with the

Test Machine	NS	TELEA	DeepFillv2
Laptop	11.5 ± 6.33	11.3 ± 5.75	N/A
Lab machine	22.1 ± 11.9	18.5 ± 9.59	488 ± 23.3
DICE (SC)	19.2 ± 5.86	21.0 ± 7.10	315 ± 25.9
Raspberry Pi	313 ± 172	320 ± 170	N/A

Table 5.5: Mean inpaint times of an image and their standard deviation in milliseconds given various inpainting methods on interleave-based masking on different test machines.

`time` module in Python before and after the invocation of the inpainting algorithm to mark the relevant start and end times. Based on the previous experiments on the two masking schemes, we decided to test interleave-based masking only because of its consistently better performance. Thus for each test machine, we collected the sets of inpaint times for interleave-masked images using the NS, TELEA, and DeepFillv2 models. The inpaint times for DeepFillv2 were collected on the Lab machine and DICE (SC) as those were the only machines that met the system requirements for DeepFillv2.

5.2.2 Results

Table 5.5 summarises the average inpaint times on the experiment images under the interleave-based masking scheme for the test machines.

As expected, the average inpaint times of the traditional models were much shorter than the ones of the DeepFillv2 model. There was at least a 22-fold and 15-fold increase when compared to the traditional methods on the Lab machine and DICE (SC) respectively. The Raspberry Pi was only able to run the traditional methods. When comparing its inpainting times against those of the other test machines, the Raspberry Pi exhibited at least 14-fold increase for both the NS and TELEA algorithms.

In terms of the specific traditional methods, two-sampled t-tests were performed to compare the difference between the average inpainting times of NS and TELEA. There was a significant difference in the average inpainting time between the two traditional algorithms on the Lab machine [(t(1496)=8.161, p=6.976e-16] and DICE (SC) [t(1496)=5.440, p=6.220e-08]. The differences in inpainting times for the Laptop and the Raspberry Pi were insignificant at the 5% significance level. We suspect there were other factors that contributed to the record runtime, such as system load during the experiment, since neither traditional algorithm was significantly different across the test machines consistently.

5.3 Implications on the Mixnet Image Reconstruction Framework

This section summarises the experiment results to address the design factors detailed in Section 3.3.2. From the image fidelity experiments, leveraging the interleave-based packetisation model was beneficial. Given an inpainting model and a certain number of

slow packets, the model would achieve higher MSSIM scores compared to block-based packetisation. Thus in the mixnet context, the sender should prepare the stream of packets such that the image pixel rows follow the interleave-based packetisation to maximise the reconstructing performance of the inpainting model at the recipient.

The image fidelity experiments also addressed the minimum number of packets the recipient should wait for before starting the image reconstruction process, i.e. the value of k. The value of k is dependent on the ability of the chosen inpainting algorithm. We should select a model that maintains high MSSIM scores w.r.t. the closeness threshold even as the number of slow packets is large. Given 8 total packets, we investigated the three inpainting models under all possible numbers of slow packets. We found that CelebA-HQ and Places2 pretrained DeepFillv2 model could tolerate at most 50% slow packets under interleave-based packetisation while the traditional methods permitted 5 slow packets. Thus the recipient could begin the image reconstruction process once three (thus 38%) packets arrive using a traditional inpainting method in the reconstruction framework. Note that since the packets arrive out-of-order, the packet preparation process needs to be modified to include the total number of packets in the headers of every packet so that the recipient can set the value of k.

It is crucial to consider the timing overhead incurred by the inpainting algorithm as part of the design of the image reconstruction framework. This is because a key objective of this work was to reduce the perceived latency by leveraging the reconstruction framework. Hence the timing overhead must not dominate the mixnet network latency for the usage of the image reconstruction framework to be meaningful. Conducting the runtime experiment could identify the inpainting algorithms with the most efficient implementation. While DeepFillv2 was designed to be lightweight compared to other deep learning models, it remains the most complex model among the models explored in this work. Its complexity in model architecture reflected its longer inference time during inpainting. Combining the good inpainting performance and the fast runtime, the traditional methods are most suited as the inpainting algorithm for the image reconstruction framework, with no significance preference over NS or TELEA.

In terms of hardware for the inpainting task, by using the traditional algorithms, the image reconstruction framework is executable from all the test machines of various hardware configurations. However, we note that the runtime experiments showed that the Raspberry Pi was not a suitable candidate as a recipient in practice as it displayed a much slower average inpainting time compared to the other test machines on the fast inpainting models.

Chapter 6

Project Evaluation

This chapter evaluates the overall work done in this project by discussing some of its limitations and some interesting future work that could be built on the work here.

6.1 Evaluation and Limitations

We acknowledge that the evaluation in this work has taken a restrictive approach to measuring human perception of image fidelity quantitatively since our primary evaluation metric is the MSSIM score, an objective metric. We discussed repeatability as an important advantage of objective metrics. Repeatability is crucial to this project especially when training the DeepFillv2 models, as the MSSIM scores were used to guide hyperparameter selections. In the future, it would be interesting to build a mapping of our MSSIM-based findings to a subjective metric such as the MOS, to further enhance the experiment results.

Improving the MSSIM performance of the different DeepFillv2 models explored in Section 5.1.1 was challenging. Notably a significant period of the experiment stage of the project timeline was dedicated to setting up and training the DeepFillv2 models with various sets of parameters. The models, such as the model trained from scratch on Birds525, eventually showed promising results and could be improved to match the performance of the traditional methods. However, we decided not to pursue the deep learning model further as its high runtime is impractical for actual usage in the image reconstruction framework. For the time taken to inpaint the missing pixels using the deep learning models, the recipient could simply wait for more or even the remaining packets to arrive to render the complete image.

The traditional inpainting methods explored in this work are diffusion-based, which means that the inpainting is limited to the information provided by the known pixels nearby to the unknown pixels. Thus, these models cannot leverage a holistic, high-level understanding of the image semantics. Hence, diffusion-based models are often more suited for inpainting missing patches in the image background. However, in this work, we have demonstrated that such restriction could be compensated by using the appropriate packetisation schemes that are suitable for the inpainting algorithms,

namely, interleaved-based packetisation.

While we have shown that inpainting algorithms can effectively fill in realistically the masked areas in images, it comes with an important implication. Utilising an inpainting algorithm to reconstruct missing information could include additions to the image that might influence the true semantics of the original images. From our experiments, none of the inpainting models reached an MSSIM score of 1, indicating a perfect replica of the original image. Thus, there is no guarantee that the reconstructed image is semantically identical to the original image. In some use cases such as on the Birds525 dataset, it was acceptable that there were some alterations compared to the original images. However, there are other use cases, such as in legal documents, where modifications would not be acceptable. It is the responsibility of the human operator at the recipient to decide whether the image reconstruction scheme is applicable.

6.2 Future Work

To the best of our knowledge, our application of an image reconstruction scheme is novel in the context of continuous-time mixnet. We have demonstrated that on the Birds525 dataset, the recipient could obtain a closely reconstructed image using only 38% of the total number of packets transmitted. Thus, the recipient only needs to wait for the shortened duration of time for the first 38% packets to arrive instead of the entire duration for all the packets to arrive. This approach to shortening the perceived latency of information transmission is also applicable to many other media, such as videos, text, and audio signals.

We give a brief discussion on some of the considerations when applying the reconstruction approach to text. There are different granularities to tokenise text data. Text data is also subject to complex linguistic processing for semantics, such as long-range dependencies and linguistic ambiguity to perform prediction, adding another layer of complexity. It would be interesting to explore the model for the reconstruction task. Most state-of-the-art text models are large language models (LLMs) such as the GPT-3 [7]. The LLM architectures are more complex than the models explored in this project. In addition to different media transmission, there could be more exploration of other types of packetisation schemes which further improve the inpainting results.

These are all interesting aspects to explore. Extending the work presented here to other low-latency media transmission can further widen the use cases of mixnet.

Bibliography

[1] Xiao Bai, Ioannis Arapakis, B. Barla Cambazoglu, and Ana Freire. Understanding and Leveraging the Impact of Response Latency on User Behaviour in Web Search. *ACM Transactions on Information Systems*, 36(2):21:1–21:42, August 2017.

[2] Ivan V. Bajić. Latent-Space Inpainting for Packet Loss Concealment in Collaborative Object Detection, January 2021. arXiv:2102.00142 [cs].

[3] Ivan V. Bajić, Weisi Lin, and Yonghong Tian. Collaborative Intelligence: Challenges and Opportunities. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8493–8497, June 2021. ISSN: 2379-190X.

[4] M.R. Banham and A.K. Katsaggelos. Digital image restoration. *IEEE Signal Processing Magazine*, 14(2):24–41, March 1997. Conference Name: IEEE Signal Processing Magazine.

[5] M. Bertalmio, A.L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, December 2001. ISSN: 1063-6919.

[6] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, July 2020. arXiv:2005.14165 [cs].

[8] RIR BT. Methodology for the subjective assessment of the quality of television pictures. *International Telecommunication Union*, 4, 2002.

[9] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[10] George Danezis. Better Anonymous Communications. 2004.

[11] George Danezis. The traffic analysis of continuous-time mixes. In *Proceedings of the 4th international conference on Privacy Enhancing Technologies*, PET'04, pages 35–50, Berlin, Heidelberg, May 2004. Springer-Verlag.

[12] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *2009 30th IEEE Symposium on Security and Privacy*, pages 269–282, 2009.

[13] David J. Wetherall and Andrew S. Tanenbaum. Computer Networks, Fifth Edition [Book], 2010. ISBN: 9780133485936.

[14] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym Network. pages 1–38, February 2021.

[15] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[16] Omar Elharrouss, Noor Almaadeed, Somaya Al-Maadeed, and Younes Akbari. Image inpainting: A review. *Neural Processing Letters*, 51:2007–2028, 2020.

[17] Gerald Piosenka. BIRDS 525 SPECIES- IMAGE CLASSIFICATION, 2023.

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

[19] Christine Guillemot and Olivier Le Meur. Image inpainting: Overview and recent advances. *IEEE signal processing magazine*, 31(1):127–144, 2013.

[20] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners, December 2021. arXiv:2111.06377 [cs].

[21] S.S. Hemami and T.H.-Y. Meng. Transform coded image reconstruction exploiting interblock correlation. *IEEE Transactions on Image Processing*, 4(7):1023–1027, July 1995. Conference Name: IEEE Transactions on Image Processing.

[22] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-TIN. How much anonymity does network latency leak? *ACM Transactions on Information and System Security*, 13(2):13:1–13:28, March 2010.

[23] Jireh Jam, Connah Kendrick, Kevin Walker, Vincent Drouard, Jison Gee-Sern Hsu, and Moi Hoon Yap. A comprehensive review of past and present image inpainting methods. *Computer vision and image understanding*, 203:103147, 2021.

[24] Mathieu Jee, Ania M. Piotrowska, Harry Halpin, and Ninoslav Marina. Optimizing anonymity and performance in a mix network. In Esma Aïmeur, Maryline Laurent, Reda Yaich, Benoît Dupont, and Joaquin Garcia-Alfaro, editors, *Foundations and Practice of Security*, pages 53–62, Cham, 2022. Springer International Publishing.

[25] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation, February 2018. arXiv:1710.10196 [cs, stat].

[26] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution, February 2022. arXiv:2202.10054 [cs].

[27] Bong-Ki Lee and Joon-Hyuk Chang. Packet Loss Concealment Based on Deep Neural Networks for Digital Speech Transmission. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(2):378–387, February 2016. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

[28] Hemi Leibowitz, Ania M Piotrowska, George Danezis, and Amir Herzberg. No right to remain silent: isolating malicious mixes. In *PROCEEDINGS OF THE 28TH USENIX SECURITY SYMPOSIUM*, volume 28, pages 1841–1858. USENIX Association, 2019.

[29] Weisi Lin and C. C. Jay Kuo. Perceptual visual quality metrics: A survey. *Journal of Visual Communication and Image Representation*, 22(4):297–312, May 2011.

[30] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image Inpainting for Irregular Holes Using Partial Convolutions, December 2018. arXiv:1804.07723 [cs].

[31] Anush K Moorthy, Zhou Wang, and Alan C Bovik. Visual perception and quality assessment. *Optical and Digital Image Processing, Wiley-VCH Verlag GmbH & Co. KGaA*, pages 419–439, 2011.

[32] Peter Ndajah, Hisakazu Kikuchi, Masahiro Yukawa, Hidenori Watanabe, and Shogo Muramatsu. *SSIM image quality metric for denoised images*. International Conference on Visualization, Imaging and Simulation - Proceedings, November 2010.

[33] Thrasyvoulos N. Pappas, Robert J. Safranek, and Junqing Chen. Perceptual Criteria for Image Quality Evaluation. In *Handbook of Image and Video Processing*, pages 939–959. Elsevier, 2005.

[34] Soureesh Patil, Amit Joshi, and Suraj Sawant. Recovering Images Using Image Inpainting Techniques. In Hariharan Muthusamy, János Botzheim, and Richi Nayak, editors, *Robotics, Control and Computer Vision*, pages 27–38, Singapore, 2023. Springer Nature.

[35] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In *Proceedings of the 26th USENIX Security Symposium*, pages 1199–1216. USENIX Association, August 2017. 26th USENIX Security Symposium : USENIX 2017 ; Conference date: 16-08-2017 Through 18-08-2017.

[36] Zhen Qin, Qingliang Zeng, Yixin Zong, and Fan Xu. Image inpainting based on deep learning: A review. *Displays*, 69:102028, September 2021.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs].

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015.

[39] Nermin M. Salem. A Survey on Various Image Inpainting Techniques. | Future Engineering Journal | EBSCOhost, July 2021. ISSN: 2314-7229 Issue: 2 Pages: 1 Volume: 2.

[40] Andrei Serjantov and Peter Sewell. Passive-attack analysis for connection-based anonymity systems. *International Journal of Information Security*, 4(3):172–180, June 2005.

[41] H.R. Sheikh, M.F. Sabir, and A.C. Bovik. A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, November 2006. Conference Name: IEEE Transactions on Image Processing.

[42] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: attacks and defenses. In *Proceedings of the 11th European conference on Research in Computer Security*, ESORICS'06, pages 18–33, Berlin, Heidelberg, September 2006. Springer-Verlag.

[43] Alexandru Telea. An Image Inpainting Technique Based on the Fast Marching Method. *Journal of Graphics Tools*, 9(1):23–34, January 2004.

[44] Subir Varma. *Internet congestion control*. Morgan Kaufmann, 2015.

[45] Kuan Lon Vu. Exposed amongst the hidden: An empirical analysis of the effects of the sphinx packet unwrap operation on the anonymity of the continuous-time mixnet. MInf Part 1 Project, The University of Edinburgh, April 2023.

[46] Yao Wang and Qin-Fan Zhu. Signal loss recovery in DCT-based image and video codecs. In *Visual Communications and Image Processing '91: Visual Communication*, volume 1605, pages 667–678. SPIE, November 1991.

[47] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. Conference Name: IEEE Transactions on Image Processing.

[48] Eric Wharton, Karen Panetta, and Sos Agaian. Human visual system based similarity metrics. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 685–690, October 2008. ISSN: 1062-922X.

[49] Hanyu Xiang, Qin Zou, Muhammad Ali Nawaz, Xianfeng Huang, Fan Zhang, and Hongkai Yu. Deep learning for image inpainting: A survey. *Pattern Recognition*, 134:109046, February 2023.

[50] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9653–9663, 2022.

[51] Huan Yang, Yuming Fang, Weisi Lin, and Zhou Wang. Subjective quality assessment of screen content images. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 257–262. IEEE, 2014.

[52] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions, April 2016. arXiv:1511.07122 [cs].

[53] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. Free-Form Image Inpainting with Gated Convolution, October 2019. arXiv:1806.03589 [cs].

[54] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative Image Inpainting with Contextual Attention, March 2018. arXiv:1801.07892 [cs].

[55] Fan Zhang and David R. Bull. Chapter 7 - measuring video quality. In Sergios Theodoridis and Rama Chellappa, editors, *Academic Press Library in signal Processing*, volume 5 of *Academic Press Library in Signal Processing*, pages 227–249. Elsevier, 2014.

[56] Xiaobo Zhang, Donghai Zhai, Tianrui Li, Yuxin Zhou, and Yang Lin. Image inpainting based on deep learning: A review. *Information Fusion*, 90:74–94, February 2023.

[57] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 Million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, June 2018. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[58] Zexian Zhou and Xiaojing Liu. Masked Autoencoders in Computer Vision: A Comprehensive Survey. *IEEE Access*, 11:113560–113579, 2023. Conference Name: IEEE Access.

[59] Thomas Zinner, Oliver Hohlfeld, Osama Abboud, and Tobias Hossfeld. Impact of frame rate and resolution on objective QoE metrics. In *2010 Second International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 29–34, June 2010.

Appendix A

DeepFillv2 Supplementary Materials

A.1 Bash Script for DeepFillv2

```
#!/bin/bash

# This script is used to test the multiple images inpainting model.
# Parameters: <path_to_masked_images_folder> <path_to_masks_folder> <path_to_outputs_folder>

# Check if the number of arguments is correct
if [ "$#" -ne 4 ]; then
    echo "Usage: $0 <masked_images_folder> <masks_folder> <outputs_folder> <checkpoint dir>"
    exit 1
fi

# Extract folder paths from command-line arguments
masked_images_folder="$1"
masks_folder="$2"
outputs_folder="$3"
checkpoint_dir="$4"

# Check if the folder exists
if [ ! -d "$masked_images_folder" ]; then
    echo "Error: Folder '$masked_images_folder' does not exist."
    exit 1
fi

# Make the outputs folder if it does not exist
if [ ! -d "$outputs_folder" ]; then
    mkdir -p "$outputs_folder"
```

```
    fi

    # Loop through each file in the folder
    for filename in "$masked_images_folder"/*; do
        # Check if the file is a regular file
        if [ -f "$filename" ]; then
            # Perform an action here, for example, print the filename
            echo "Processing file: $filename"

            # filename: chunk_mask_dropnum1_total8_filename15.jpg
            # Extract file_num
            file_num=$(echo $filename | sed -n 's/.*filename\([0-9]*\).*/\1/p')

            # Extract mask_type
            mask_type=$(basename "$filename" | cut -d'_' -f1)

            # Extract dropnum
            dropnum=$(basename "$filename" | grep -oP '(?<=_dropnum)\d+(?=_total)')

            # Extract total_packets, it is the number after 'total'
            total_packets=$(echo $filename | sed -n 's/.*total\([0-9]*\).*/\1/p')

            # Print the extracted variables
            echo "file_num: $file_num"
            echo "mask_type: $mask_type"
            echo "dropnum: $dropnum"
            echo "total_packets: $total_packets"

            # mask_name="${masks_folder}/${mask_type}_mask_dropnum${dropnum}_total${total_packets}_filename${file_num}.png"
            # if mask_name is 'chunk', then use mask_name="${masks_folder}/${mask_type}_mask_dropnum${dropnum}_total${total_packets}_filename${file_num}.png"
            # else if mask_name is 'strip', then use mask_name="${masks_folder}/${mask_type}_mask_dropnum${dropnum}_total${total_packets}.png"
            if [ "$mask_type" == "chunk" ]; then
                mask_name="${masks_folder}/${mask_type}_mask_dropnum${dropnum}_total${total_packets}_filename${file_num}.png"
            elif [ "$mask_type" == "strip" ]; then
```

```
            mask_name="${masks_folder}/${mask_type}
↪ _mask_dropnum${dropnum}_total${total_packets}.png"
        else
            echo "Error: Invalid mask type. Exiting."
            exit 1
        fi

        inpaint_name="${outputs_folder}/${file_num}_${
↪ mask_type}_dropnum${dropnum}_total${total_packets}
↪ _generative.jpg"

        echo "mask_name: $mask_name"
        echo "inpaint_name: $inpaint_name"

        python test.py --image "$filename" --mask "$mask_name"
↪ --output "$inpaint_name" --checkpoint_dir "$checkpoint_dir"

        # Catch error
        # Check the exit status of the Python script
        if [ $? -ne 0 ]; then
            echo "Error: Python script failed. Exiting."
            exit 1
        fi
    fi
done
```

A.2 Places2 and CelebA-HQ Pretrained Model Results

A.2.1 Significance Test Results

	Block-based		Interleave-based	
Number of slow packets	t-statistic	p-value	t-statistic	p-value
1	-0.1606	0.8726	0.9069	0.3655
2	1.1030	0.2713	-1.5871	0.1140
3	0.8569	0.3925	-1.6166	0.1075
4	1.8652	0.0635	-2.0317	0.0434
5	0.8487	0.3970	-4.0398	0.0001
6	-0.0681	0.9458	-5.7306	0.0000
7	-3.0139	0.0029	-5.2194	0.0000

Table A.1: Significant test results for the difference in mean of post-inpainting MSSIM scores between Places2 and CelebA-HQ models.

A.2.2 Percentage Increase from Block- to Interleave-based Masking

	Percentage Increase (%)	
Number of Slow Packets	CelebA-HQ	Places2
1	6.5	6.1
2	12.3	13.3
3	18.5	19.7
4	26.2	28.1
5	34.1	37.0
6	44.0	47.6
7	48.8	50.5

Table A.2: The percentage increase of mean post-inpainting MSSIM scores from block- to interleave-based masking achieved by the pretrained models.

A.2.3 Selected Inpainted Outcome for CelebA-HQ and Places2 Pretrained Models

A.2.3.1 Places2 Pretrained Model

(a) DeepFillv2 on block-masked

(b) Places2 on interleave-masked

Figure A.1: Various inpainted images of Figure 4.2 using the Places2 pretrained model for the block-masked image (Figure 4.2b) on the left, and the interleave-masked image (Figure 4.2c) on the right.

A.2.3.2 CelebA-HQ Pretrained Model

(a) DeepFillv2 on block-masked

(b) Places2 on interleave-masked

Figure A.2: Various inpainted images of Figure 4.2 using the CelebA-HQ pretrained model for the block-masked image (Figure 4.2b) on the left, and the interleave-masked image (Figure 4.2c) on the right.

A.3 Finetuning DeepFillv2

A.3.1 DeepFillv2 Configuration File

The default configuration of the `inpaint.yml` file, provided by the authors:

```
# ========================== Basic Settings ==========================
# machine info
num_gpus_per_job: 1  # number of gpus each job need
num_cpus_per_job: 4  # number of gpus each job need
num_hosts_per_job: 1
memory_per_job: 32  # number of gpus each job need
gpu_type: 'nvidia-tesla-p100'

# parameters
name: places2_gated_conv_v100  # any name
model_restore: ''  # logs/places2_gated_conv
dataset: 'celebahq'  # 'tmnist', 'dtd', 'places2', 'celeba', 'imagenet', 'cityscapes'
random_crop: False  # Set to false when dataset is 'celebahq', meaning only resize the images to img_shapes, instead of crop img_shapes from a larger raw image. This is useful when you train on images with different resolutions like places2. In these cases, please set random_crop to true.
val: False  # true if you want to view validation results in tensorboard
```

```
log_dir: logs/full_model_celeba_hq_256

gan: 'sngan'
gan_loss_alpha: 1
gan_with_mask: True
discounted_mask: True
random_seed: False
padding: 'SAME'

# training
train_spe: 4000
max_iters: 100000000
viz_max_out: 10
val_psteps: 2000

# data
data_flist:
  # https://github.com/jiahuiyu/progressive_growing_of_gans_tf
  celebahq: [
    'data/celeba_hq/train_shuffled.flist',
    'data/celeba_hq/validation_static_view.flist'
  ]
  # http://mmlab.ie.cuhk.edu.hk/projects/celeba.html, please to use
   ↪ random_crop: True
  celeba: [
    'data/celeba/train_shuffled.flist',
    'data/celeba/validation_static_view.flist'
  ]
  # http://places2.csail.mit.edu/, please download the high-
   ↪ resolution dataset and use random_crop: True
  places2: [
    'data/places2/train_shuffled.flist',
    'data/places2/validation_static_view.flist'
  ]
  # http://www.image-net.org/, please use random_crop: True
  imagenet: [
    'data/imagenet/train_shuffled.flist',
    'data/imagenet/validation_static_view.flist',
  ]

static_view_size: 30
img_shapes: [256, 256, 3]
height: 128
width: 128
max_delta_height: 32
max_delta_width: 32
```

```
batch_size: 16
vertical_margin: 0
horizontal_margin: 0

# loss
ae_loss: True
l1_loss: True
l1_loss_alpha: 1.

# to tune
guided: False
edge_threshold: 0.6
```

A.3.2 Inpainted Result from Finetuning with Default Loss Ratio

Figure A.3: A example of inpainting an image from the experimental images with 3 slow packets, using the ImageNet model finetuned on Birds525.

A.4 Training DeepFillv2 from Scratch

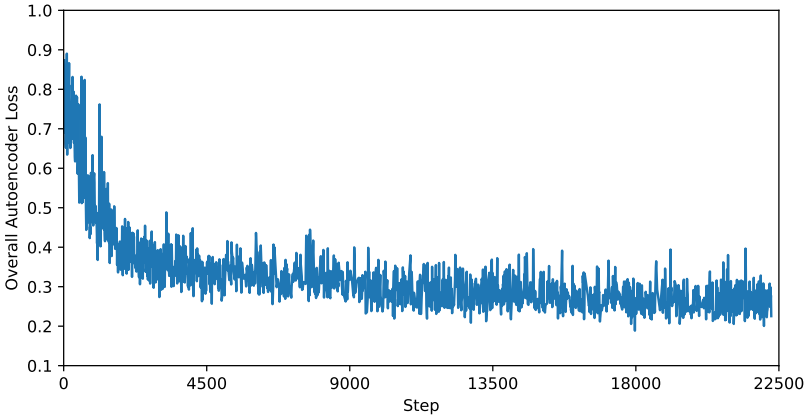


Figure A.4: The overall autoencoder loss for training the DeepFillv2 model from scratch on the Birds525 dataset.