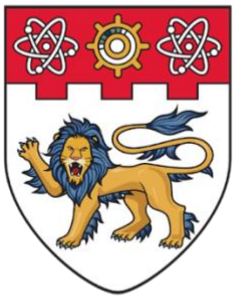

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Self-Supervised Deep Learning for Missing Image Predictions

Author	Yeoh Shun Bin
	U1922317C
Supervisor	A/P Yeo Chai Kiat
Examiner	Dr. Dmitrii Ustiugov
Project ID	SCSE22-0243

Submitted in fulfilment of the requirements for the Degree of Bachelor of Engineering (Computer Science) of Nanyang Technological University

AY 2022/23

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

Abstract

Deep learning has revolutionized the field of computer vision, enabling impressive advances in image classification, object detection, and segmentation. However, the success of supervised deep learning in computer vision is largely dependent on the availability of large annotated datasets, which can be time-consuming and expensive to acquire. Hence, self-supervised deep learning can be automated even with enormous volumes of unlabelled data, which reduces data labelling costs.

This project explores self-supervised learning techniques for computer vision-based tasks, with a specific focus on patch context prediction for missing image predictions. We explore the potential of self-supervised learning to develop a model that can predict missing parts of an image based on the surrounding context and evaluate the performance of state-of-the-art (SOTA) techniques in this area. The goal of this report is to demonstrate the potential of self-supervised learning for practical image restoration scenarios and to provide insights into the impact of hyperparameters on model performance.

Acknowledgements

The author expresses sincere appreciation to Associate Professor Yeo Chai Kiat for her unwavering support and guidance throughout the project. Despite the challenges faced during the project, the author found her advice to be of great assistance and it was a pleasure to work under her supervision.

In addition, the author acknowledges the support provided by the numerous contributors on Stack Overflow, GitHub and other sources who provided helpful pointers in overcoming programming obstacles encountered during the project.

List of Figures

List of Tables

Table of Contents

1. Introduction

1.1 Background and Motivation

The success of deep learning in computer vision tasks is largely attributed to the availability of large annotated datasets. However, manual annotation is a time-consuming and expensive process that can limit the scalability and accessibility of computer vision solutions [1]. In addition, models trained on these datasets may not generalize well to new or unseen examples.

Thus far, self-supervised learning schemes have shown promising performance in simpler tasks such as reassembly of jigsaw puzzles [2] and prediction of image rotation [3]. However, it has not been demonstrated how this approach may be used to extract usable information from real-life unlabelled image collections, which are in abundance and easily sourced on the internet.

This project presents a self-supervised solution for computer vision, illustrated in the use case of patch context prediction. An artificially generated image is obfuscated with a missing patch where the algorithm is used to reconstruct the full image based on the surrounding knowledge. By developing a self-supervised learning model that can predict missing parts of an image, there is potential of this technique for practical image restoration applications, such as medical imaging [4], where experts spend endless hours manually classifying and segmenting pictures. Therefore it will help to expand its application to complex real-world problems.

1.2 Problem Statement and Objectives

1.2.1 Problem Statement

Many computer vision applications have missing or partial images as a result of numerous problems such as occlusions, image corruption, or restricted camera perspectives. Conventional approaches for filling in missing picture data sometimes involve human interpretation or previous knowledge of the missing information, which may be time-consuming and error-prone. As a result, there is a need for a self-supervised deep learning technique to anticipate missing picture data effectively, efficiently and without the need for human interaction. This strategy should use the existing data to understand the underlying structure and patterns in the picture data and attain credible predictions for the missing information.

1.2.2 Objectives and Scope

The main objective of this project is to develop a self-supervised learning solution for computer vision, specifically in the context of patch context prediction, to complete missing parts of an image based on the surrounding context. The project will utilize state-of-the-art deep learning technique called autoencoder, to achieve this objective.

The scope of the project will include the following:

1. Data pre-processing which involves using CIFAR-10 unlabelled dataset as it comprises myriads of different images within the 60,000 colour images. The training data will be obfuscated with a patch being removed from the images and this constitutes the missing images dataset.

2. Building a model from scratch using a dataset of photos, some of which have been purposefully masked out.

3. Utilizing the remaining or surrounding pixels' context to forecast the missing portions of the images.

4. Evaluating the model's performance and quality of reconstructed images using metrics such as mean squared error (MSE), peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM).

5. Exploring potential computer vision applications of the developed model in industries such as medical or satellite imaging.

The project's primary focus will be on developing and evaluating the self-supervised learning model for patch context prediction. Further potential extensions and applications of the model in other image restoration tasks will also be explored. This project will not involve the development of any hardware or physical implementation of the model.

1.3 Research Questions

The following are the research questions for this project:

1. How can self-supervised learning techniques be used to effectively predict missing parts of images based on the surrounding context?

2. Which state-of-the-art self-supervised learning method is most suitable to produce the optimal missing image prediction model?

3. What hyperparameters have the greatest impact on model performance, and how can they be optimized for improved results?

4. How does the performance of self-supervised learning-based image restoration models compare to traditional supervised learning-based models?

5. In what practical scenarios, such as in computer vision or medical imaging, could self-supervised learning-based image restoration techniques be most useful?

6. How can the performance of self-supervised learning-based image restoration models be further improved, and what new research directions should be explored in this area?

1.4 Contribution and Significance of the Study

The significance of this study lies in its exploration of self-supervised learning techniques for image completion tasks. By using self-supervised learning, which requires less labelled data, we can overcome some of the limitations of supervised learning, such as the need for large amounts of labelled data and the limited ability to generalize to new, unseen examples.

Additionally, image completion tasks have numerous practical applications, such as in the restoration of damaged or incomplete images in the fields of computer vision and medical imaging. By developing a model that can accurately predict missing portions of an image, we can improve image restoration processes and potentially reduce the cost and time required for expert human intervention.

Moreover, the use of deep learning models in image completion tasks is still an active area of research, and this study may contribute to the development of more accurate and efficient models. By exploring the impact of different self-supervised learning techniques and hyperparameters on model performance, this study may

provide insights into the optimal design of models for image completion tasks. Overall, the results of this study may contribute to the advancement of the field of deep learning and its practical applications in image completion tasks.

1.5 Project Schedule

Figure 1.1 shows the project schedule.

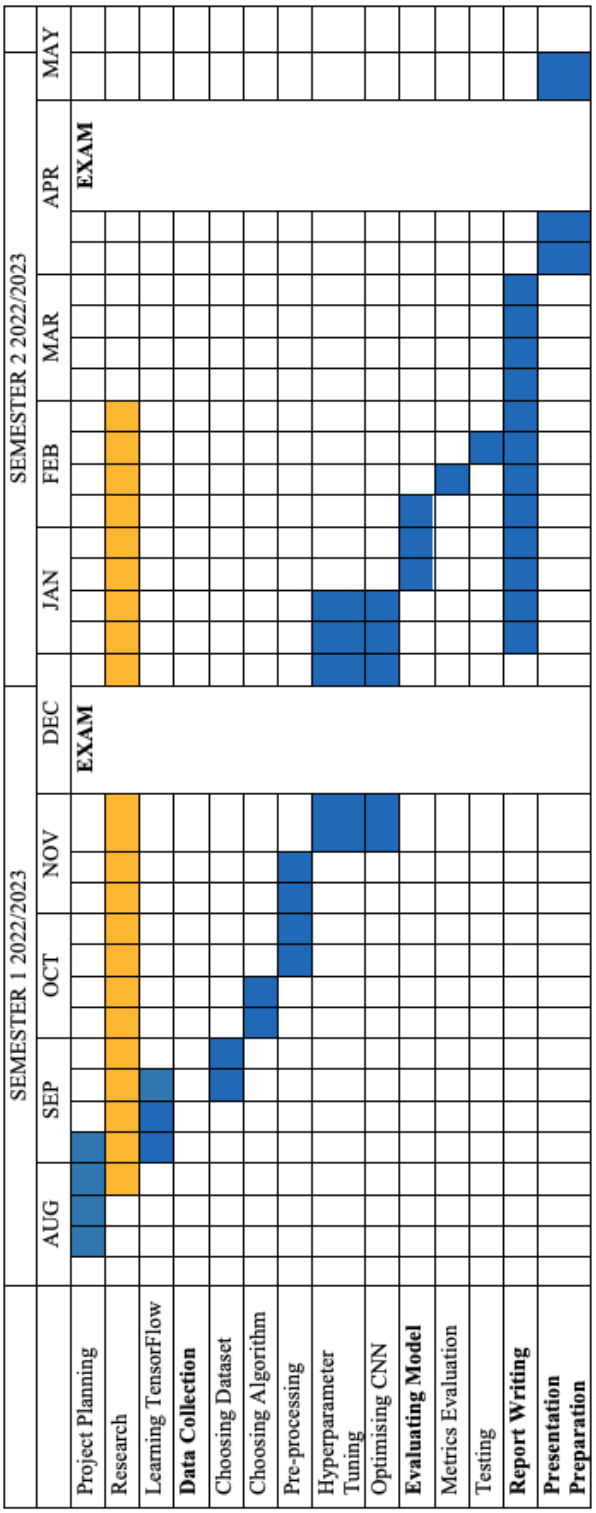

Figure 1.1 Project Schedule

1.6 Organisation of Report

The report is organised into several chapters to provide a clear and structured presentation of the research findings. Preceding the main body, readers will find an abstract, acknowledgements, list of figures and tables, and a comprehensive table of contents. The main body consists of five chapters that cover the essential components of the research, including an introduction, literature review, methodology, results and discussion, conclusion and future work. References and Appendix are at the end of this report. This organisational structure ensures that readers can easily navigate and comprehend the author's content.

2. Literature Review

2.1 Computer Vision and Deep Learning

Computer vision is a field of artificial intelligence concerned with enabling machines to interpret, analyse, and understand visual information from the spatial context around them. Deep learning, a subfield of machine learning, has revolutionized computer vision by enabling models to automatically learn from large amounts of data and perform complex visual tasks, such as object recognition, segmentation, and detection [5].

Deep learning models consist of multiple layers of interconnected neurons that process input data in a hierarchical manner, with each layer learning increasingly abstract and complex features [6]. Convolutional neural networks (CNNs), a specific type of deep learning model designed for image processing tasks, have become a cornerstone of modern computer vision [7]. CNNs use a series of convolutional and pooling layers to extract features from input images, followed by one or more fully connected layers for classification or regression tasks.

While CNNs have achieved impressive results on a wide range of computer vision tasks, they typically require large labelled datasets to train effectively. This limitation has spurred interest in self-supervised learning methods, which can learn from unlabelled data or generate their own labels through unsupervised or semi-supervised techniques.

One of the most promising applications of self-supervised learning in computer vision is image restoration, including tasks such as denoising, deblurring, and inpainting [8]. These tasks are challenging because they require models to infer missing or corrupted information from incomplete data, often with limited supervision. Self-supervised learning methods, which can leverage the structure and

context of images to generate high-quality restorations, have shown great promise for addressing these challenges [9].

2.2 Supervised Learning for Image Recognition

Supervised learning has been the dominant approach for image recognition tasks using deep learning algorithms. Convolutional Neural Networks (CNNs) have achieved state-of-the-art performance on a variety of image recognition datasets such as CIFAR-10, ImageNet and MNIST. Supervised learning involves the use of labelled data, where each image in the dataset is assigned a specific label or class. During the training phase, the CNN learns to identify the features that are most relevant for classifying each image correctly.

One of the most widely used CNN architectures for image recognition is the VGG network, which was introduced in 2014 by Simonyan and Zisserman. VGG network consists of 19 layers and is known for its deep architecture, which is capable of learning complex features from images. Another popular architecture for image recognition is the ResNet (Residual Network), introduced by He et al. in 2015 [10]. ResNet is a deeper architecture than VGG and can learn to identify complex features using residual blocks, which enable the network to learn the residual function between two layers.

Although supervised learning has been very successful for image recognition tasks, it requires a large amount of labelled data, which can be expensive and time-consuming to acquire. In addition, models trained on supervised learning tasks often struggle to generalize well to unseen examples. These challenges have motivated the development of self-supervised learning techniques for image recognition tasks.

2.3 Limitations of Supervised Learning

While supervised learning has proven to be a powerful technique for image recognition tasks, it has several limitations. One major limitation is the need for large amounts of labelled data to train models effectively. Labelling data can be time-consuming and expensive, particularly for tasks that require expert knowledge, such as medical image analysis. Additionally, labelled data may be biased or limited in scope, leading to models that do not generalize well to new, unseen examples.

Another limitation is that supervised learning requires the data to be representative of the problem being solved. For example, if the training data do not contain sufficient variation in lighting conditions or camera angles, the resulting model may perform poorly when presented with new, unseen examples with different lighting or angles.

Finally, supervised learning models may not be able to handle missing or incomplete data, which can frequently occur in real-world scenarios. In such cases, a model trained on complete data may not be able to provide accurate predictions or may require significant modifications to handle missing data. These limitations have led researchers to explore alternative approaches, such as self-supervised learning, which can address some of these challenges.

2.4 State-of-the-Art Self-Supervised Learning Techniques

Self-supervised learning has emerged as a promising alternative to supervised learning for computer vision tasks. It involves training models to predict certain properties of input without the need for explicit labels. This approach has been applied to various aforementioned computer vision tasks and has achieved remarkable results.

One popular self-supervised learning technique is contrastive learning, which trains models to maximize the similarity between different views of the same input while minimizing the similarity between views of different inputs. This technique has been used in recent works such as SimCLR (Simple Framework for Contrastive Learning of Visual Representations) and MoCo (Momentum Contrast) and has achieved state-of-the-art performance on several image classification benchmarks.

Another self-supervised learning technique is generative modelling, which involves training models to generate realistic samples of the input distribution. This approach has been applied to image completion tasks, where a model is trained to generate missing parts of an image based on the surrounding context. Recent works such as Deep Image Prior (DIP) and Context Encoders have shown promising results in image completion tasks.

The development of self-supervised learning techniques has the potential to overcome the limitations of supervised learning, such as the need for large amounts of labelled data. By training models to predict certain properties of input without explicit labels, self-supervised learning can make use of the vast amounts of unlabelled data that are available. Moreover, self-supervised learning can improve the generalization of models to new, unseen examples, as it encourages models to learn more meaningful representations of the input. Therefore, self-supervised

learning has the potential to significantly advance the field of computer vision and its practical applications.

2.5 Redirecting Self-Supervised Learning for Missing Image Predictions

In recent years, various self-supervised learning techniques have been proposed and have achieved state-of-the-art results in several tasks. One of the popular techniques is Contrastive Predictive Coding (CPC), which uses a predictive framework to learn representations from sequential data. In this method, the model is trained to predict future samples based on a subset of the current samples [11]. The learned representations can then be used for downstream tasks.

Another technique is SimCLR, which is a self-supervised learning approach that trains deep neural networks using contrastive learning without the need for manually labelled data. It teaches the model to distinguish between positive and negative instances by learning an image representation that is invariant to some types of transformations and then utilizing data augmentation to generate several representations of the same picture [12]. The model learns to map these various views to a common feature space in such a way that views of the same picture are near, while views of different images are far away from each other, which means contracting positive and contrasting negatives respectively [13].

In MoCo, the model is trained to predict image attributes stored in the memory bank by maximizing cosine similarity between the query image and a positive instance and reducing cosine similarity between the query image and a collection of negative instances. The memory bank is refreshed using a momentum update algorithm, which allows for improved model convergence [14]. Both SimCLR and MoCo have shown state-of-the-art performance on visual recognition tasks and enhanced downstream tasks that require labelled data. However, these

techniques are primarily designed to perform classification tasks, hence, they might not be effective approaches to be used in this project.

As mentioned previously, both Deep Image Prior and Context Encoder are self-supervised learning techniques that are commonly used for image inpainting because of its ability to predict the representations of an augmented view of an image based on its surroundings [15]. However, DIP is mainly used when there is a limited amount or poor quality of training data to enhance images. As such, context encoder or autoencoder will be better suited as the model will require less iterations to converge thus achieving the objective of this project more efficiently [16]. To do so, when the model is trained on a dataset of complete images, the autoencoder is able to generate the missing regions by producing the most likely values in the latent space and decode them to eventually produce the whole, reconstructed image from the missing image [17].

The aforementioned state-of-the-art self-supervised learning techniques have the potential to significantly improve the performance of computer vision tasks while reducing the need for manual annotations. All in all, the technique that has stood out, aligned with the scope and decided to be used is autoencoder.

3. Methodology

3.1 Data Collection and Preparation

The CIFAR-10 dataset is used to train and evaluate our self-supervised learning model. The CIFAR-10 dataset consists of 100 classes of 32x32 colour images, with 600 images per class. The dataset is split into training and testing sets, with 50,000 images in the training set and 10,000 images in the test set. The classes are grouped into 20 super-classes, each containing five fine-grained classes.

For self-supervised learning image inpainting on CIFAR-10, the dataset has to be pre-processed to be used as the input for the model. The images were obfuscated by masking with a square size of 10x10 at random position of every image. By doing so, these pre-processed images will emulate missing images for the prediction task through self-supervised deep learning approach. Furthermore, by randomly masking out different regions of the input images during training, the model can learn to predict missing pixels in a robust and generalizable manner [18]. Figure 3.1 shows an example of a masked versus the original images. Figure 3.2 shows the code snippet for the image masking.

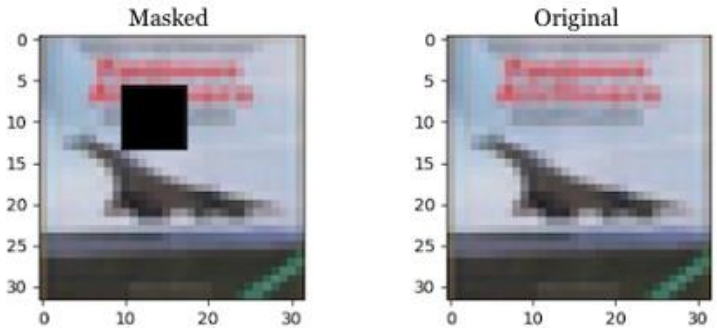

Figure 3.1 Masked image with 10x10 square at random regions and original image

```python
# function to randomly create masks of mask_size x mask_size in images
def random_mask(imgs, mask_size):
    # make a copy of images
    masked_imgs = np.copy(imgs)
    # max possible values for starting / ending points of masks
    x_max, y_max = imgs.shape[1:3]
    for i in range(len(masked_imgs)):
        x = np.random.randint(x_max - mask_size)
        y = np.random.randint(y_max - mask_size)
        masked_imgs[i, x:x + mask_size, y:y + mask_size, :] = 0
    return masked_imgs
```

Figure 3.2 Code snippet of image masking for training data

3.2 Architecture Design and Implementation

The model architecture used for self-supervised image inpainting on CIFAR-10 is based on convolutional autoencoder with skip connections. The input image has a size of (32, 32, 3) and undergoes a sequence of convolutional and pooling layers to encode it into a smaller representation, followed by deconvolutional and up-sampling layers to decode it back to its original size.

The encoder comprises three convolutional layers with 32, 64, and 128 filters, respectively, and each layer is followed by a max pooling layer with a stride of 2 to reduce the spatial size of the feature maps by half. The decoder includes corresponding deconvolutional layers with an up-sampling factor of 2, which are concatenated with the output of the corresponding encoder layer to form a skip connection.

Finally, the decoder includes a final convolutional layer with sigmoid activation function to generate the reconstructed image. The skip connections facilitate access to the lower-level features learned by the encoder, which enables the model to preserve more detailed information and improve the reconstruction quality. The Adam optimizer and the specified loss function are used to train the

model. Figure 3.3 shows the implemented model architecture. Figure 3.4 shows the code snippet for generating the self-supervised model.

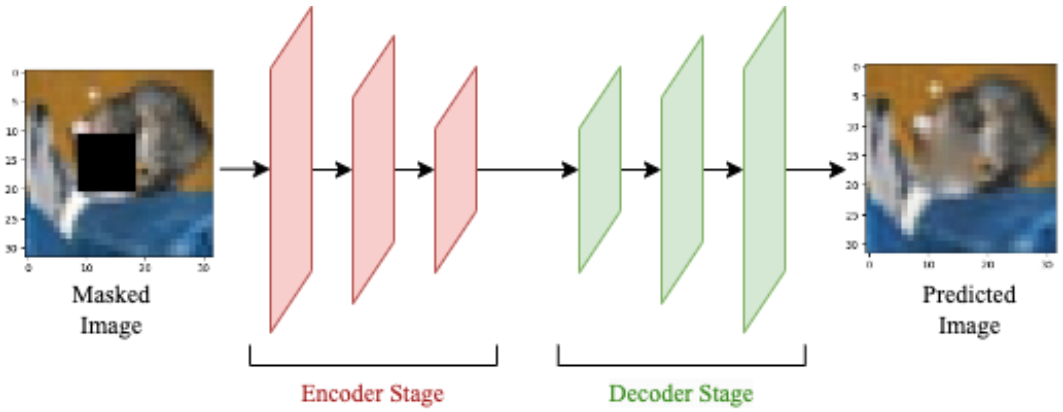

Figure 3.3 Model Architecture

```python
# model definition - output = pixel value between 0-1 using sigmoid
# The model will take a batch of masked images and predict the missing pixels
def get_model(loss, learning_rate):
    # Define the convolutional autoencoder with skip connections
    input_img = tf.keras.layers.Input(shape=(32, 32, 3))

    # Encoder
    x1 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
    x1 = tf.keras.layers.MaxPooling2D((2, 2), padding='same')(x1)  # 16
    x2 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(x1)
    x2 = tf.keras.layers.MaxPooling2D((2, 2), padding='same')(x2)  # 8
    x3 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(x2)
    x3 = tf.keras.layers.MaxPooling2D((2, 2), padding='same')(x3)  # 4

    # Decoder
    x4 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(x3)
    x4 = tf.keras.layers.UpSampling2D((2, 2))(x4)
    x5 = tf.keras.layers.Concatenate()([x4, x2])
    x6 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(x5)
    x6 = tf.keras.layers.UpSampling2D((2, 2))(x6)
    x7 = tf.keras.layers.Concatenate()([x6, x1])
    x8 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same')(x7)
    x8 = tf.keras.layers.UpSampling2D((2, 2))(x8)

    decoded = tf.keras.layers.Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x8)

    # Autoencoder
    model = tf.keras.models.Model(input_img, decoded)
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate), loss=loss)

    return model
```

Figure 3.4 Code snippet for generating self-supervised model

3.3 Training

This model is then trained using the self-supervised learning technique. During training, the model is fed partially masked images and is trained to predict the missing part. The loss function used for training is the MSE loss.

The model is trained on randomly masked images for 100 iterations with one epoch per iteration as it increases the variety of training data and leads to faster convergence. In addition, randomly masking parts of the training images can act as a form of regularization, forcing the model to learn more robust features that are less dependent on specific pixels or regions of the image.

The model performance is evaluated on a separate set of validation images after each iteration, during hyperparameter tuning and post hyperparameter tuning using the test set images.

3.4 Evaluation

The performance of the model is evaluated using various metrics such as Mean squared error (MSE), Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). PSNR and SSIM are image quality metrics that measure the similarity between the predicted and ground truth images.

4. Results and Discussion

4.1 Experimental Setup

The experimental setup comprises hardware and software. For hardware, a machine with a high-performance graphics processing unit (GPU) is required, with at least 8GB of memory and a CPU with at least 16GB of RAM is also required to handle the large dataset, which NTU GPU cluster has positively met the aforementioned requirements.

For software, the virtual environment with Python programming language of Version 3.8 was used, alongside several packages to be installed such as, Tensorflow deep learning framework, Hyperopt library for hyperparameter tuning, NumPy for numerical computing, Matplotlib library for data visualization.

4.2 Performance Evaluation Metrics

MSE measures the average squared difference between the predicted and actual pixel values in an image. A lower MSE indicates a better model that is able to reconstruct the missing regions of the image. However, it does not evidently measure the quality of the image as perceptual quality of reconstructed image is not considered [19].

PSNR is crucial as it measures the ratio of the maximum signal power to corrupting noise power that affects the fidelity of the image. This means that the difference in the predicted and original images; pixels are measured [20]. As such, a higher PSNR value also indicates a better-quality reconstruction or compression, as it means that the reconstructed or compressed signal is more similar to the original or reference signal.

Lastly, SSIM is a metric that considers structural similarity between the predicted to the ground truth image in terms of overall structure and texture, such as luminance and contrast differences. It is a more accurate measure of image quality compared to MSE as it provides a perceptual measure of the similarity between the images [21]. SSIM values range from 0 to 1, with 0 being no match and 1 being a perfect match between reconstructed and actual image.

4.3 Impact of Hyperparameters on Model Performance

Hyperparameter tuning was done in an automatedly using the validation set using the Hyperopt library in Python. The goal was set to minimize the loss (MSE) calculated over the validation set. Hyperopt uses Bayesian optimization for hyperparameter tuning thus requires much smaller number of iterations as compared to a grid search. Bayesian optimization finds the best set of hyperparameters by constructing a probabilistic model of the objective function and selects the next set of hyperparameters to be evaluated, determined by the expected improvement of the model's performance [22].

Table 4.1 Tested parameters for training

status	batch_size	learning_rate	loss
ok	32	0.0007	0.003718843
ok	48	0.0009	0.00404658
ok	64	0.0005	0.00502059
ok	32	0.0005	0.00426306
ok	48	0.0006	0.004821263
ok	**64**	**0.001**	**0.003714701**
ok	48	0.0006	0.004434858
ok	64	0.0006	0.004866138
ok	64	0.0009	0.003926978
ok	64	0.0007	0.004729392
ok	48	0.0009	0.003949971
ok	48	0.0005	0.004737281
ok	48	0.0005	0.004294298

ok	64	0.0006	0.004657257
ok	64	0.0007	0.004524256
ok	32	0.0003	0.004693863
ok	48	0.0008	0.004054436
ok	64	0.0006	0.004651783
ok	48	0.0006	0.00438882
ok	32	0.0004	0.004268168

Table 4.1 shows the tried parameters and the associated validation loss that was calculated for the same set of masked validation images, which was also used as a metric to select the best set of parameters for training. It can be observed that the changes in hyperparameters can increase or decrease its validation loss by approximately 35% (validation loss from 0.005021 to 0.003715). With 0.003714701 being the least validation loss, the parameters of 64 batch size and 0.001 rate were selected.

4.4 Discussion of Results

Our model achieved an SSIM of 0.97 and a PSNR of 27.12 on the CIFAR-10 dataset which indicates that the model can reconstruct the missing or corrupted parts of the images with a high level of accuracy and fidelity. The results prior to hyperparameter tuning are as follow, SSIM of 0.79 and PSNR of 17.85. Table 4.2 shows the improved results before and after hyperparameter tuning.

Table 4.2 Comparison of results over CIFAR-10 before and after fine-tuning. +Higher is better.

Hyperparameter tuning	**SSIM+**	**PSNR+**
Before	0.79	17.85
After	**0.97**	**27.12**

Figures 4.1 and 4.2 show the instances of the generated images alongside the metrics calculated.

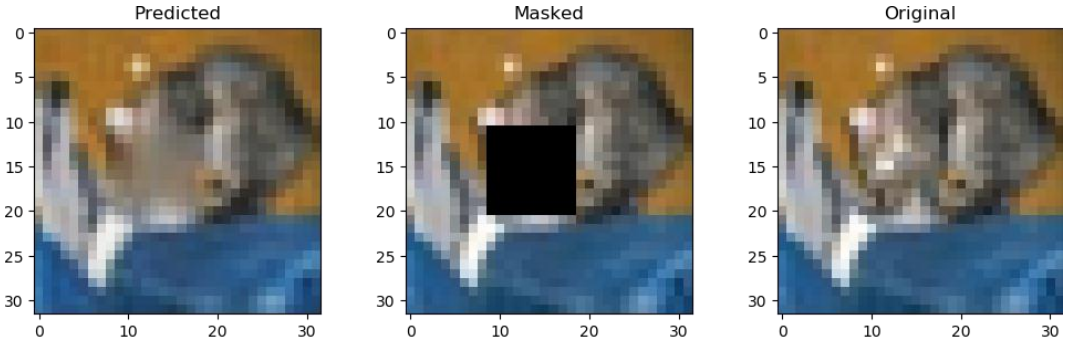

Figure 4.1 Instance 1 with MSE: 0.0024, SSIM: 0.91, PSNR: 26.21

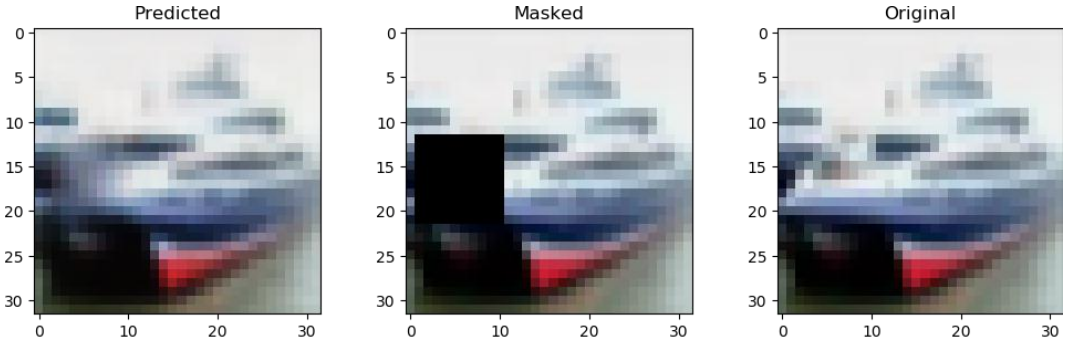

Figure 4.2 Instance 2 with MSE: 0.0051, SSIM: 0.94, PSNR: 22.90

With a result of SSIM value of 0.97, the reconstructed images depicted are almost identical to the original images, with a high degree of structural similarity. This indicates that the model has successfully learned to fill in the missing patch of the images based on the remaining visible information, while preserving the underlying structure and features of the original images. Similarly, the PSNR indicates that the reconstructed images have a low average error compared to the original images, which means that the model has successfully reconstructed the missing or corrupted parts of the images with a high degree of accuracy.

However, it was observed that even though the metrics show that the generated image is close to a good match for the original image, the output images are slightly blurred compared to the original images.

Table 4.3 shows the comparison of our model against other different models. It shows that our model performs better than the other models.

Table 4.3 Comparison of results with different models. +Higher is better.

Method	**Datasets**	**SSIM+**	**PSNR+**
Artist-Net [23]	Paris StreetView ImageNet	0.59	19.94
Hsu et. al. [24]	ImageNet	0.69	25.43
Xiang et al. [25]	Internet images	0.95	27.42
Our method	**CIFAR-10**	**0.97**	**27.12**

4.5 Problems Encountered

During the research for performance evaluation metrics in Chapter 4.2, binary cross entropy (BCE) loss was used instead of MSE loss but after more in-depth understanding of the cross entropy, it was found that BCE is more suitable for problems where the missing pixel values are binary, in black and white, as the pixel values are typically represented in binary 0 or 1. As such, the BCE loss metric was scraped and MSE loss became one of the metrics to be analysed since missing pixel values are continuous which suits the goal of colored image inpainting at high accuracy.

On the other hand, excessive random-access memory (RAM) issue was encountered during hyperparameter tuning because the dataset was loaded to memory. This issue was resolved by decreasing the resource requirements with the GPU memory being cleared in each iteration as shown in Figure 4.3.

```
# Hyperparameter tuning using Hyperopt - scoring cnn function returns the loss value using the parameters
# passed by optimize cnn function
def scoring_cnn(params):

    tf.keras.backend.clear_session()
    model = get_model(params['loss'], params['learning_rate'])

    model.fit(masked_imgs, x_train, batch_size=params['batch_size'], epochs=1, verbose=0)
    preds = model.predict(masked_valid, verbose=0)
    loss = mean_squared_error(x_valid, preds).numpy().mean()
    gc.collect()
    return loss
```

Figure 4.3 Use of tf.keras.backend.clear_session() to free memory space

5. Conclusion and Future Work

5.1 Summary of Findings

Overall, the model has achieved a high PSNR (27.12) and SSIM (0.97) values which indicates that it is effective at reconstructing missing or corrupted parts of images in other similar datasets or real-world scenarios. In addition, considering it takes around 20 mins to complete entire training, it is computationally very efficient. Compared to other models, the proposed model achieves better performance in terms of PSNR and SSIM.

However, the reconstructed images appear slightly blurred or visually unsatisfactory, which could indicate that the model is prioritizing fidelity to the visible information over visual quality. SSIM and PSNR are objective metrics that primarily measure the similarity between the original and reconstructed images in terms of their structure and pixel values, but do not necessarily capture subjective aspects of visual quality such as sharpness, texture, and detail.

When it comes to deep learning, the goal is to achieve the best possible performance on a given task. While there may be other models or approaches that have outperformed the model derived from this project, it is important to note that this model is still considered successful as there are numerous factors that can influence the results, such as the size and quality of the dataset, choice of hyperparameters and architecture of the model used. Therefore, this model is considered valuable as it has outperformed other approaches that have been previously used.

5.2 Implications and Contributions

While high SSIM and PSNR values are important indicators of fidelity to the visible information, these metrics do not necessarily capture the subjective aspects of visual quality such as sharpness, texture, and detail. By addressing this issue and improving the visual quality of the reconstructed images, the continued development of self-supervised learning models has the potential to improve the quality and efficiency of image restoration tasks in a wide range of applications such as medical or satellite imaging.

5.3 Limitations and Future Research Directions

There are several ways to address the implication and improve the visual quality of the reconstructed images. The usage of perceptual loss functions, as it utilises pre-trained deep neural networks to measure the perceptual similarity between the original and reconstructed images in terms of high-level features such as texture and color. By using this, the model can learn to optimize for visual quality as well as fidelity.

Furthermore, adversarial training can be used as it involves training the model to generate images that not only match the visible information but also fool a discriminator network into thinking that the images are real. This can help the model learn to generate more visually plausible images with sharpness, texture, and detail.

Lastly, the complexity of the model can be increased. With a more complex model, such as a deep convolutional neural network with multiple layers and skip connections, the intricate details of the image can be captured and in turn produce visually satisfying results.

References

[1] C. B. Rasmussen, K. Kirk, and T. B. Moeslund, “The challenge of data annotation in Deep Learning-A case study on whole plant corn silage,” *MDPI*, 18-Feb-2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/4/1596. [Accessed: 13-Jan-2023].

[2] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” arXiv, 22-Aug-2017. [Online]. Available: https://arxiv.org/abs/1603.09246. [Accessed: 19-Sep-2022].

[3] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” arXiv, 21-Mar-2018. [Online]. Available: https://arxiv.org/abs/1803.07728. [Accessed: 19-Sep-2022].

[4] M. J. Willemink and W. A. Koszek, “Preparing medical imaging data for machine learning,” *Radiology*, Apr-2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7104701/. [Accessed: 02-Jan-2023].

[5] LeCun, Y., Bengio, Y., & Hinton, G., “(PDF) Deep Learning - Researchgate.” [Online]. Available: https://www.researchgate.net/publication/277411157_Deep_Learning. [Accessed: 16-Feb-2023].

[6] Goodfellow, I., Bengio, Y., & Courville, A., Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7780459. [Accessed: 07-Jan-2023].

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. “ImageNet classification with deep convolutional Neural Networks,” 2012. [Online]. Available: https://www.researchgate.net/publication/267960550_ImageNet_Classification_with_Deep_Convolutional_Neural_Networks. [Accessed: 08-Jan-2023].

[8] X. Chu, L. Chen, C. Chen, and X. Lu, “Improving image restoration by revisiting global information aggregation,” *arXiv.org*, 02-Aug-2022. [Online]. Available: https://arxiv.org/abs/2112.04491. [Accessed: 26-Jan-2023].

[9] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “Sphereface: Deep Hypersphere embedding for face recognition,” *arXiv.org*, 29-Jan-2018. [Online]. Available: https://arxiv.org/abs/1704.08063. [Accessed: 27-Jan-2023].

[10] M. Shafiq and Z. Gu, “Deep residual learning for image recognition: A survey,” *MDPI*, 07-Sep-2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/18/8972. [Accessed: 14-Jan-2023].

[11] A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv.org*, 22-Jan-2019. [Online]. Available: https://arxiv.org/abs/1807.03748. [Accessed: 16-Jan-2023].

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” *arXiv.org*, 01-Jul-2020. [Online]. Available: https://arxiv.org/abs/2002.05709. [Accessed: 14-Jan-2023].

[13] A. Béres, “Keras documentation: Semi-supervised image classification using Contrastive Pretraining with simclr,” *Keras*, 24-Apr-2021. [Online]. Available: https://keras.io/examples/vision/semisupervised_simclr/. [Accessed: 18-Jan-2023].

[14] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” *arXiv.org*, 23-Mar-2020. [Online]. Available: https://arxiv.org/abs/1911.05722. [Accessed: 14-Jan-2023].

[15] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Deep image prior,” *arXiv.org*, 17-May-2020. [Online]. Available: https://arxiv.org/abs/1711.10925. [Accessed: 16-Jan-2023].

[16] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by Inpainting,” *arXiv.org*, 21-Nov-2016. [Online]. Available: https://arxiv.org/pdf/1604.07379.pdf. [Accessed: 16-Jan-2023].

[17] M. H. Givkashi, M. Hadipour, A. PariZanganeh, Z. Nabizadeh, N. Karimi, and S. Samavi, “Image inpainting using AutoEncoder and guided selection of predicted pixels,” *arXiv.org*, 17-Dec-2021. [Online]. Available: https://arxiv.org/abs/2112.09262. [Accessed: 01-Mar-2023].

[18] C. O'Sullivan, “Augmenting images for Deep Learning,” *Medium*, 17-Nov-2022. [Online]. Available: https://towardsdatascience.com/augmenting-images-for-deep-learning-3f1ea92a891c. [Accessed: 03-Feb-2023].

[19] C. Thomas, “Deep Learning Image Enhancement Insights on loss function engineering,” *Medium*, 21-Feb-2020. [Online]. Available: https://towardsdatascience.com/deep-learning-image-enhancement-insights-on-loss-function-engineering-f57ccbb585d7. [Accessed: 08-Feb-2023].

[20] K. Sangeetha, P. Sengottuvelan, and Balamurugan, “A comparative analysis of exemplar based image inpainting algorithms,” *A Comparative Analysis of Exemplar Based Image Inpainting Algorithms*, Dec-2011. [Online]. Available: https://www.researchgate.net/publication/257951855_A_Comparative_Analysis_of_Exemplar_Based_Image_Inpainting_Algorithms. [Accessed: 10-Feb-2023].

[21] Z. Wang, C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “University of Waterloo,” *Image Quality Assessment: From Error Visibility to Structural Similarity*, 04-Apr-2004. [Online]. Available: https://ece.uwaterloo.ca/~z70wang/publications/ssim.pdf. [Accessed: 10-Feb-2023].

[22] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, “Hyperparameter optimization for machine learning models based on Bayesian optimization,” Journal of Electronic Science and Technology, 01-Mar-2019. [Online]. Available: http://www.journal.uestc.edu.cn/en/article/doi/10.11989/JEST.1674-862X.80904120. [Accessed: 13-Feb-2023].

[23] L. Liao, “Artist-Net: Decorating the Inferred Content With Unified Style for Image Inpainting,” IEEE Xplore Full-text PDF: 14-Feb-2019. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8669751. [Accessed: 01-Mar-2023].

[24] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari, “Image Inpainting: A Review,” arXiv.org, 13-Sep-2019. [Online]. Available: https://arxiv.org/abs/1909.06399. [Accessed: 25-Feb-2023].

[25] P. Xiang, L. Wang, J. Cheng, B. Zhang, J. Wu, “A deep network architecture for image inpainting - researchgate,” A deep network architecture for image inpainting. In:2017 3rd IEEE international conference on computer and communications (ICCC), Dec-2017. [Online]. Available: https://www.researchgate.net/publication/324464355_A_deep_network_architecture_for_image_inpainting. [Accessed: 01-Mar-2023].

Appendix

A Details of Datasets Used

The diverse dataset of CIFAR-10 contains 60,000 images across 10 classes, with 6000 images per class which makes this challenging dataset suitable for training and evaluating deep learning models. The small images of 32 by 32 pixels allow efficient performance on computational resources for a manageable training time. This benchmark dataset is widely used for computer vision tasks, such as object detection, segmentation and image classification. Hence, this renders compatibility of similar studies to utilise these images and also allows ease of project results reproducibility and comparability.

B Implementation Details

Firstly, the virtual environment of Python version 3.8 has to be installed with the necessary packages such as Tensorflow, Hyperopt, NumPy and Matplotlib. CIFAR-10 is the chosen dataset to obfuscate some images with missing parts. This artificially generated dataset will be the pretext task to predict the missing parts of an image given the remaining parts. To pre-process the data, images were cropped out and saved as separate images which will be used to train the self-supervised model.

Next, to train the model, the pairs obtained from pre-processed dataset are used where in the architecture of the network, the convolutional layers will extract features from the images whilst fully connected layers predict the missing parts of the image.

Furthermore, to evaluate the model, the test set is used on a trained model by computing metrics such as MSE, SSIM and PSNR score. Visualisation of the predicted missing parts were plotted using Matplotlib to showcase the reconstructed image, missing image and original image side by side. The model that is parsed to achieve the results will be a trained self-supervised model that is able to predict missing parts of an image.

Lastly, it is crucial to fine-tune the trained model to improve performance where key parameters such as batch size, learning rate, number of training epochs and loss function, were used. However, depending on the particular task, different hyperparameters may have different optimum values. In order to determine the ideal hyperparameter settings, it is crucial to experiment with various values and track the model's performance. The trained and fine-tuned model can eventually be used in image inpainting scenarios for a variety of applications including image restoration for medical and satellite imagery.