UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Generalizing texture transformers for super-resolution and inpainting

Author:
Teodor ROMANUS

Supervisor:
Roman RIAZANTSEV

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Teodor ROMANUS, declare that this thesis titled, "Generalizing texture transformers for super-resolution and inpainting" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Generalizing texture transformers for super-resolution and inpainting

by Teodor ROMANUS

Abstract

The new multi-camera smartphones and recent advancements in generalized Machine Learning models make it possible to bring new types of photo editing neural networks to the market. This thesis covers methods of image enhancement with texture transfer. The known high-resolution regions (reference) can be utilized to restore degraded areas of an image. The task of restoring partially degraded images can be defined "partial super-resolution." The task of restoring missing parts of images is called inpainting. We propose to use the novel Texture Transformer Network for Image Super-Resolution (TTSR) to solve the partial super-resolution and inpainting tasks.

The fully convolutional networks are unable to copy image patches. This inability forces the model to store textures using the train weights. The usage of the attention mechanism allows taking advantage of joint feature learning in low-resolution and high-resolution parts of images simultaneously, in which deep feature correspondences can be discovered by attention. This approach exhibits an accurate transfer of texture features.

The experiments confirm that the TTSR network can be used to solve the partial super-resolution and inpainting tasks simultaneously. Modifications of the network (different embedding sizes, soft-attention, trainable projections) study the architecture capacity to solve the specified tasks. The evaluation of results includes comparing the TTSR network with an inpainting network for the inpainting task.

Acknowledgements

I am incredibly grateful to Roman Riazantsev, my supervisor, whom this project would be impossible without. All the ideas, research questions, and suggestions gave me priceless experience in Deep Learning and Computer Vision.

Many thanks to Maksym Davydov, who helped identify the research direction and mentored the project along the way. Maksym also taught the Computer Vision course at the university, and I heard about Attention in Computer Vision first time from him.

Lastly, I would like to mention my family, especially my parents. Words cannot express my gratitude for the invaluable patience, help, and moral support at every moment of my study.

Contents

List of Figures

List of Tables

List of Abbreviations

SISR	**S**ingle **I**mage **S**uper-**R**esolution
RefSR	**Ref**erence-based **S**uper-**R**esolution
TTSR	**T**exture **T**ransformer Network for Image **S**uper-**R**esolution
SR-RAW	**S**uper-**R**esolution **Raw** dataset
QD-IMD	**Q**uick **D**raw **I**rregular **M**ask **D**ataset
MSE	**M**ean **S**quared **E**rror
PSNR	**P**eak **S**ignal-to-**N**oise **R**atio
SSIM	**S**tructural **Sim**ilarity index
FID	**F**réchet **I**nception **D**istance
GAN	**G**enerative **A**dversarial **N**etwork
LR	**L**ow **R**esolution
HR	**H**igh **R**esolution
GPU	**G**raphics **P**rocessing **U**nit

Chapter 1

Introduction

Image super-resolution aims to recover realistic high-resolution images from relevant low-resolution images. The recent advancements in this area helped greatly increase the quality of media content for better user experiences. The super-resolution spheres include but are not limited to entertainment industry, multimedia [Malczewski and Stasiński, 2009], medical imaging [Oktay et al., 2016] and satellite imaging [Yıldırım and Güngör, 2012]. For example, the NVIDIA Deep Learning Super Sampling technology increases graphics performance many times by rendering low-resolution images and applying super-resolution techniques to increase the image's resolution to very high values [NVIDIA, 2020].

The task of software-based image enhancement is very important nowadays. Every capturing device (an analog camera, a digital camera, a set of multiple cameras) inevitably has information loss and errors while measuring the amount of light received, which creates a task of processing raw data to acquire as much useful information as possible. After applying hardware image acquisition techniques, more advanced algorithms might be used to reconstruct image features lost by noise. The camera manufacturers employ their software algorithms which fuse the information of separate sequential captured images in order to produce a single high-resolution image [Wronski et al., 2019]. It is also a common practice to use a set of cameras to collect image information at different camera configurations (focus distances, apertures, etc.) [Carles, Downing, and Harvey, 2014].

The main direction in the super-resolution area is single-image super-resolution (SISR). The first SISR algorithms tended to lose the high-resolution features and resulted in blurry effects. The state-of-the-art SISR algorithms [Wang et al., 2021] use Generative Adversarial Networks to solve some of these issues. The alternative approach of **reference-based super-resolution (RefSR)** transfers high-resolution textures from a given Ref image to produce visually plausible results[Zheng et al., 2018; Zhang et al., 2019c]. However, many RefSR solutions use a straightforward way of textures transfer which may result in visually unacceptable images. The most recent works use high-level semantic features to solve this problem[Zhang et al., 2019c].

In this work, we study Reference-based image reconstruction problems. One of the tasks is **partial super-resolution**: by having the higher resolution data of the image part, we aim to transfer the high-resolution details to the whole image. Another task is **inpainting**: reconstructing the damaged, impaired, or missing parts. These two tasks (super-resolution and inpainting) can be used to improve the experience of current multi-camera setups; for example, in phone cameras: one can use the information from the wide-angle camera to increase the field of view of the photo captured with the regular camera, preserving the regular quality. By applying reconstruction techniques, one can reduce the damage of sun or lamp glares in the image. To solve these challenges, we use the Texture Transformer Network architecture, described in Section 3. TTSR is a RefSR architecture that leverages the

Attention mechanism to transfer the most relevant features from the existing parts of the image.

The work raises the following research questions:

Question 1. *Is it possible to make a neural network that does the super-resolution and inpainting tasks simultaneously?*

Question 2. *Is it possible to reduce the number of parameters of the network and preserve model accuracy?*

Question 3. *How does using soft-attention for texture transfer impact the network results?*

Question 4. *How does using hard-attention with trainable projections for texture transfer impact the network results?*

An essential part of the research is exploring networks, which can solve multiple tasks simultaneously. The most recent research in Deep Learning studies joint learning of different tasks for better individual task accuracy (for example, DeepMind in [Reed et al., 2022]). As a side benefit, multi-task networks run faster than sequential execution of the same tasks, which becomes essential in real-time applications.

The main part of the work consists of 5 chapters. Chapter 2 explores Related Work on the super-resolution and inpainting tasks. Chapter 3 (Background) introduces the network architecture, loss functions and metrics. Chapter 4 (Problem setting and Approach to Solution) sets the research hypothesis and outlines the research and evaluation plan. Chapter 5 introduces the Datasets used for network training. Chapter 6 describes conducted Experiments and analyses their results.

Chapter 2

Related work

2.1 Single-image super-resolution

With the recent advancements in the Deep Learning area, various Deep Learning methods were applied to the super-resolution problem which greatly overcome traditional non-learning based methods. The most successful architecture appears to be Convolutional Neural Networks (CNNs) with various loss functions, with one of the pioneering architectures of SRCNN - Super-Resolution CNN [Dong et al., 2014]. This paper was a foundation for a whole set of methods, most of them trying to optimize PSNR as well as the original paper did. As was shown later [Ledig et al., 2017], the PSNR metric highly disagrees with the human perception of resolution and cannot be used.

Opposing the SRCNN algorithms, state of the art (SOTA) super-resolution papers investigate Generative Adversarial Networks (GANs) [Ledig et al., 2017; Menon et al., 2020] and target to improve the perceptual quality of the images (ESRGAN - Enhanced Super-Resolution GAN [Wang et al., 2018], Real-ESRGAN is the improved version, trained with pure synthetic data [Wang et al., 2021]). The authors use the perceptual loss metric that optimizes the super-resolution model in a feature space instead of pixel space. The most recent papers in the area try to leverage the Attention mechanism for Convolutional Neural Networks, which uses Residual in Residual Blocks with Channel Attention to make the model focus on learning high-frequency information better and achieve better multi-channel (color) images super-resolution as well [Zhang et al., 2018; Zhang et al., 2019a].

2.2 Reference-based super-resolution

2.2.1 Classical Approaches

One of the first papers in the field [Gur and Zalevsky, 2007] described the research field as "Texture Reconstruction Guided by a High-Resolution Patch". At that time, the researchers were focused on fixing the artifacts which naturally occur by the imaging process: misfocus, compression and other forms of losing high-frequency information in images. They propose an iterative method of imposing the high-frequency information from the patch to the rest of the image in the Fourier frequency domain. The authors mention that this method may be applied only to improve image content slightly or as a preprocessing step for other image processing algorithms. The work does not provide any comparison to other methods.

Another classical algorithm was proposed in [Hidane et al., 2014; Hidane et al., 2016]. The authors refer to the task as *image zooming completion* problem. The method learns nonlocal interactions between low- and high-resolution images in patches and

then applies them to a low-resolution image. The results show much better similarity metrics, compared to bicubic interpolation of the missing part. The subsequent works of the same authors investigate different minimization objective functions [El Gheche et al., 2016] and reason them in terms of human perception of image resolution.

2.2.2 Generative Adversarial Networks

The successful deep learning algorithms for partial super-resolution started with Generative Adversarial Networks [Mao et al., 2016]. The task was to reconstruct pixelated/damaged parts of an image. The resulting method combines the adversarial autoencoder with two depixelate layers together with the SRCNN architecture from [Dong et al., 2014] and the deconvolution idea from [Haris, Shakhnarovich, and Ukita, 2018], both already described above. The results outperformed SISR approaches. On the other hand, the reconstruction quality depends on the size of the pixelated area or different kinds of blurrinesses.

The state-of-the-art method [Zeng et al., 2021] questions the results of regular Generative Adversarial Networks, claiming they suffer from generating distorted structures and blurry textures in high-resolution images. To solve those challenges, the authors propose an enhanced GAN-based model, named Aggregated Contextual Transformation GAN (AOT-GAN), for high-resolution images inpainting.

2.2.3 Transformers

A relatively novel approach in image processing is Visual Transformer Networks [Wu et al., 2020]. This architecture drastically differs from all other approaches. While all other approaches (GANs, CNNs) typically operate on the pixel-level or (recently) features level, Transformers operate in semantic token space, using different image parts based on context. This is especially valuable for the considered application, where textures of the specific semantic category might be transferred to the requested locations instead of a random similar feature in the image. This modification only (the Attention mechanism) is considered to be the main justification to use Transformer architecture for the feature transfer and super-resolution tasks.

The architecture application in the super-resolution area is presented in [Yang et al., 2020]. The authors present Texture Transformer Network for Image Super-Resolution (TTSR), which shows how taking high-resolution images as references can help to transfer the high-resolution textures to low-resolution images (more details in the section 3).

2.3 Inpainting

Image inpainting aims to recover damaged or missing parts of images. This task has been in demand for a long time because of numerous practical applications in image editing (restoring photographs, removing objects in images, etc.).

The first inpainting algorithms tried to sample patches from an input image and paste them into the synthesized output. [Criminisi, Pérez, and Toyama, 2004] proposed an algorithm to remove large objects in an image by combining texture restoration and inpainting techniques to fill small image gaps. The most popular

algorithm which does not use neural networks is PatchMatch, which finds approximate nearest neighbor matches between image patches by using a randomized image sampling algorithm, outperforming all previous inpainting algorithms by orders of magnitude in the inference time [Barnes et al., 2009].

One of the first learning solutions used the encoder-decoder architecture, mapping the missing regions to the image in a low-dimensional feature space [Pathak et al., 2016]. The solutions struggled with visual artifacts and blurriness. Further work in this field fixed most of the drawbacks of the initial ideas though by significantly increasing the algorithm working time[Yu and Koltun, 2015].

The most recent Deep Learning solutions use Generative Adversarial Networks to achieve visually satisfying results with fast inference time. Various improvements are applied to increase the models accuracy: Contextual Attention mechanism [Yu et al., 2018], multi-pass algorithms [Nazeri et al., 2019], Aggregated Contextual Transformations [Zeng et al., 2022].

The state-of-the-art inpainting networks use Diffusion Models, which recently emerged with substantial results for image generation. The Palette Network [Saharia et al., 2021] can do image colorization, inpainting, uncropping, and JPEG restoration tasks by using conditional diffusion models.

2.4 Summary

This section reviews the previous work on single image super-resolution (SISR), reference-based super-resolution (RefSR), and inpainting, which are the most relevant to this work.

Chapter 3

Background

In this chapter we present a Texture Transformer Network architecture which is going to be a baseline of the whole research process. The in-detail explanations can be found in the original paper [Yang et al., 2020].

3.1 Texture Transformer

The network is designed to increase the resolution of a low-resolution image (LR) to the scale of the reference image (Ref).

The architecture of the Texture Transformer is presented in Figure 3.1. LR and Ref represent the input data. LR↑ corresponds to the bicubic up-sampled LR image to the resolution of the Ref image. Ref↓↑ corresponds to the bicubic down-sampled to the LR image resolution and up-sampled back image to be the domain-consistent with LR↑ . The transformer takes Ref , Ref↓↑ , LR↑ , and LR features produced by a backbone and outputs a feature map which is used later to generate an HR image.

The Texture Transformer consists of five modules: the learnable texture extractor (LTE), the relevance embedding module (RE), the hard attention module for feature transfer (HA), the soft-attention model for feature synthesis (SA), and the cross-scale feature integration module (CSFI) which is added at the top of the Texture Transformer. A short description of all modules is presented below.

3.1.1 Learnable Texture Extractor

The Learnable Texture Extractor is a feature extraction module which is trained together with the model. Such design results in joint feature learning between the LR and Ref images. The process can be explained as:

$$Q = LTE(LR\uparrow), \tag{3.1}$$

$$K = LTE(Ref\downarrow\uparrow), \tag{3.2}$$

$$V = LTE(Ref) \tag{3.3}$$

The extracted features **Q**uery, **K**ey, and **V**alue are the basic elements of the attention mechanism and used in the relevance embedding module.

3.1.2 Relevance Embedding

Relevance embedding finds the relevance between the LR and Ref images by calculating the similarity between Q and K. Both Q and K are unfolded into patches, and then a scalar product denotes the similarity measure between two patches in Q and K.

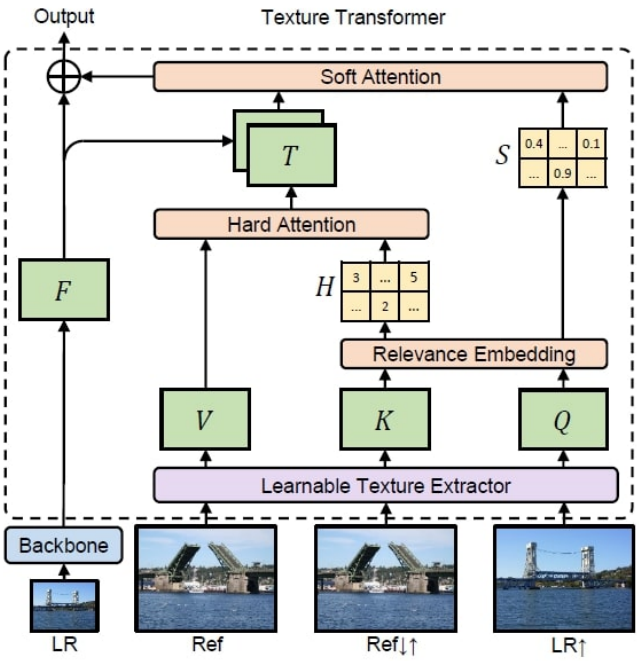


FIGURE 3.1: The Texture Transformer architecture (the image from [Yang et al., 2020]).

3.1.3 Attention

The Hard-Attention module transfers features V from the Ref image. The Attention mechanism introduced in [Vaswani et al., 2017] takes a weighted sum of all patches in Q. The authors claim this might cause an undesired blur effect in the image. To avoid it, only the patch with a maximum similarity is picked from the Q so we get 1-1 correspondences, which represents the most relevant position in the Ref image to transfer to a specific position in the LR image. We are going to question the selected approach and will experiment with the Soft-Attention layer in Section 3.

A Soft-Attention module is used to synthesize features from the transferred HR texture features T and features F of the LR image from a DNN backbone. To leverage only relevant features transfer, a soft-attention map is computed between the LR and Ref patches. For each patch in the LR image, a patch in the Ref image with the highest similarity score is selected. After that, the features F are fused with the patch with the weight of that similarity score.

As an output of this module (and the whole texture transformer) we get the synthesized output features.

3.1.4 Cross-scale feature integration

The original TTSR architecture mainly focuses on 4x super-resolution. The authors highlight the need to use features from the Ref image at different scales to facilitate the features of different domains to be transferred. The Texture Transformer (Soft-Attention as the last layer) returns texture features at different scales (1x, 2x, and 4x). The proposed CSFI module receives the exchanged features from other scales by up-sampling or down-sampling, followed by the channel-wise concatenation. The ending convolutions reduce the number of channels to three (for an RGB image) and produce the resulting SR image.

3.2 Loss Function

The Texture Transformer Network uses a combination of three loss functions:

$$\mathcal{L}_{overall} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv} + \lambda_{per}\mathcal{L}_{per}, \tag{3.4}$$

where $\lambda_{rec}, \lambda_{adv}, \lambda_{per}$ correspond to the weights of the selected loss function, specified at the training time. The loss functions are:

- $\mathcal{L}_{rec}$ is reconstruction loss: L1 norm between the expected and the actual image:

$$\mathcal{L}_{rec} = \frac{1}{CHW}\left\|I^{HR} - I^{SR}\right\|_1, \tag{3.5}$$

  where (C,H,W) denotes the size of the HR image, I^{SR} is a predicted SR image.

- $\mathcal{L}_{adv}$ is generative adversarial loss: generates clear and visually favorable images. The WGAN-GP[Gulrajani et al., 2017] approach is used:

$$\mathcal{L}_{adv} = -log(D(I^{SR}), \tag{3.6}$$

$$\mathcal{L}_D = -log(D(I^{HR})) - log(1 - D(I^{SR})), \tag{3.7}$$

  where $\mathcal{L}_D$ corresponds to the loss function of the discriminator network, $D(\cdot)$ is the output of the discriminator network.

- $\mathcal{L}_{pec}$ is perceptual loss: similarity in feature space between the expected and the actual image. Here it is an L2 norm between the corresponding VGG19 layers of two networks with expected and actual images passed. The second part is transferal perceptual loss, which compares features, extracted by LTE:

$$\mathcal{L}_{pec} = \frac{1}{C_i H_i W_i}\left\|\phi_i^{VGG}(I^{HR}) - \phi_i^{VGG}(I^{SR})\right\|_2^2 + \frac{1}{C_j H_j W_j}\left\|\phi_j^{LTE}(I^{HR}) - \phi_j^{LTE}(I^{SR})\right\|_2^2, \tag{3.8}$$

  where $\phi_i^{VGG}(\cdot)$ describes the features of VGG19 with the image specified, and $\phi_j^{LTE}(\cdot)$ describes the feature map of LTE.

  As was shown in [Johnson, Alahi, and Fei-Fei, 2016], adding perception loss to the model training process provides more human-plausible results, comparing to using the reconstruction loss only.

3.3 Quality Metrics

The most common quality metrics (similarity functions) to be used in image processing are:

1. **MSE**, the mean squared error. Given a $m \times n$ target image I and its noisy approximation K, MSE is defined as

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{3.9}$$

2. **PSNR**, the peak-signal-to-noise ratio: penalizes for the high contrast differences:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right), \tag{3.10}$$

where MAX_I is the maximum pixel value in the image.

3. **SSIM**, the structural similarity index measure: penalizes for the loss of the structural information. For the image patches I and K.

$$SSIM = \frac{(2\mu_I\mu_K + c_1)(2\sigma_{IK} + c_2)}{(\mu_I^2 + \mu_K^2 + c_1)(\sigma_I^2 + \sigma_K^2 + c_2)}, \tag{3.11}$$

where:

- μ_I, μ_K are average of images I and K,
- $\sigma_I, \sigma_K, \sigma_{IK}$ are the respective variances and the covariance of images I and K,
- $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ stabilize the division with the weak denominator,
- $L = 2^{bits per pixel} - 1$ is the dynamic range of the pixel-values,
- $k_1 = 0.01, k_2 = 0.03$.

The structural similarity of an image is computed as an average of scores of each pixel.

4. **FID**, the Fréchet inception distance: allows to assess the quality of images created by a generative model. As introduced in [Heusel et al., 2017], it can be computed from the mean and the covariance of the activations when the synthesized and real images are fed into the Inception network as:

$$FID = \|\mu_I - \mu_K\|_2^2 + \text{tr} \left(\Sigma_I + \Sigma_K - 2(\Sigma_K^{\frac{1}{2}} \cdot \Sigma_I \cdot \Sigma_K^{\frac{1}{2}})^{\frac{1}{2}} \right), \tag{3.12}$$

where $\mathcal{N}(\mu_I, \Sigma_I)$ and $\mathcal{N}(\mu_K, \Sigma_K)$ are the Gaussian distributions of the Inception network activations of the target images I and the generated images K respectively. As the internal network, Inception-v3 trained on ImageNet is used [Heusel et al., 2017].

The authors of the Texture Transformer Network use PSNR and SSIM to compare their solution with other networks. For SSIM, 5x5 sliding windows are used. As discussed in [Ledig et al., 2017], when applied in super-resolution tasks, these similarity functions often disagree with the human perception of resolution. For example, the inferred images with high PSNR are often blurred. This raises the need for careful examination of the obtained results. The authors partially overcome this issue by using the perception loss in training, which takes into account image features extracted from the VGG network, as described in the section 3.2.

3.4 Summary

This chapter introduces the Texture Transformer Network, all the network modules, the loss functions, and quality metrics.

Chapter 4

Problem setting and Approach to Solution

This section discusses the research questions, specifies the problems to solve, and outlines the research plan. There are four main research questions introduced in Chapter 1. They are discussed in detail below.

4.1 Question 1. Training a multi-task network

Problem formulation.

Partial super-resolution. Given two images with the overlapping fields of view: a low-resolution input image and a high-resolution reference image, generate an image with the resolution of the high-resolution image and the scene of the low-resolution image.

An example of the partial super-resolution task is image zoom completion: given a high-resolution image and a wide field-of-view image , generate an image with the resolution of the high-resolution image and the field of view of the wide image. The output image is a wide high-resolution image (Figure 4.1).

Inpainting. Given an image with the damaged part, generate an image that is identical outside of the damaged part, with this part to look visually plausible and unidentifiable as a damaged part by humans (Figure 4.2).

A "damaged part" can be defined differently. One of the proposed research directions is the sun or a lamp producing the artifacts (lens flares) onto the camera sensor. In another definition, it is the user-supplied parts of the image, provided as a binary mask, which should be inpainted.

FIGURE 4.1: Example of the partial super-resolution task. Left to right: low-resolution wide-angle image, high-resolution image, and resulting wide high-resolution image (images from [Hidane et al., 2016]).

FIGURE 4.2: Example of the image damage by inpainting by the NVIDIA Inpainting Network [Liu et al., 2018] (free-to-use image from www.pexels.com)

Hypothesis 1 (Multi-task network)**.** The Texture Transformer Network can solve the partial super-resolution and inpainting tasks at the same time because the reference image contains missing information.

4.2 Question 2. Reducing the number of parameters

In an attempt to explore the possibilities and limitations of the chosen architecture, we study the differences between the Attention mechanism for feature transfer in the original TTSR paper [Yang et al., 2020] and the baseline Attention mechanism, introduced in [Vaswani et al., 2017].

One of the significant differences of the TTSR architecture is that the Relevance Embedding module (Section 3.1.2) encodes all pixels with embeddings of size 2304 (9×256). Since the authors of the original papers did not justify the need for such embedding size and other papers use a lot smaller embedding size (600 in [Vaswani et al., 2017]), the optimal embedding size selection was made a part of the research plan.

Hypothesis 2 (Reducing number of parameters)**.** It is possible to reduce the embedding size of the TTSR network because other methods work just fine with smaller embeddings.

4.3 Question 3. Soft-Attention

Another notable difference between the two approaches is TTSR using Hard Attention (introduced in Section 3.1.3). In contrast, the classic approach suggests Soft Attention (so-called "Scaled Dot-Product Attention") when calculating correspondences between the Query and Key matrices, represented by embeddings of input and reference images in our architecture.

As mentioned in Section 3.1.3, using TTSR for the texture transfer task results in blurred images. We will test this hypothesis, too.

Hypothesis 3 (Soft-Attention)**.** The accuracy of super-resolution and inpainting tasks can be increased when using the Soft-Attention layer for the feature transfer inside the TTSR network because Soft-Attention works efficiently in many transformer models.

4.4 Question 4. Trainable Projections

Another difference between the classic Attention mechanism in [Vaswani et al., 2017] and the Texture Transfer Network in [Yang et al., 2020] is using Trainable Attention projection weights, which "linearly project the queries, keys, and value into different subspaces to jointly attend to information from different representation subspaces at different positions."

As a part of this work, we explore if there exists any subspace projection of Query and Key matrices where the super-resolution task shows better results. Adding these projections potentially introduces the application of Multi-Head attention for the super-resolution and inpainting tasks.

Hypothesis 4 (Trainable Projections)**.** The accuracy of the super-resolution and inpainting tasks can be increased when using trainable projections Q and K for the feature transfer inside the TTSR network because an input image and a reference image can have different features due to the inpainting mask.

4.5 Approach

The research questions and hypotheses above outline the need to have a well-structured plan to validate them.

The selected research methodology inherits the scientific method. It contains the next steps:

1. **Problem formulation**: Ask questions and formulate hypotheses.
2. **Data analysis**: Ask questions and formulate hypotheses.
3. **Prediction**: Describe the proposed solution.
4. **Testing**: Implement the proposed solution and run experiments.
5. **Evaluation**: Evaluate the solution. Estimate the impact.
6. **Refinement**.

4.6 Hypotheses verification

While answering the questions in Section 1 and testing hypotheses in the sections 4.1-4.4, one needs a way to evaluate the outcomes of the proposed solution. In all hypotheses described above, the outputs are the proposed neural network architectures together with the corresponding dataset to train it.

In order to check of the network can be used to the applied problems, are:

1. Training time evaluation, based on loss functions. During the training process the loss function should generally decrease. The loss functions described in the section 3.2 penalize both image similarity in the contrast space as well as the feature space.

2. Post-training time evaluation. The validation of the networks which output images is performed by comparing the expected and actual generated images by computing a similarity score between them.

During the hypothesis validation the inspection of the loss functions as well all aggregated similarity metrics will be used. If all loss functions converge, the hypothesis is very likely to be accepted.

4.7 Comparing to other solutions

We plan to compare our solution for the tasks of super-resolution and inpainting. To the best of our knowledge, there are no solutions that can solve both tasks simultaneously.

The main novelty of the thesis is exploring the inpainting task by using the network designed for a different task. Thus, we will compare TTSR with an inpainting network for the inpainting task. The comparison will allow us to analyze the pros and cons of the selected architecture for the main challenging task. The original Texture Transformer Network is not a good candidate for comparison because it does reference-based super-resolution, as was shown by the original paper authors, and it cannot challenge the inpainting part of the solution.

Palette is an state-of-the-art inpainting network. This is a multi-task diffusion model which uses a class-conditional U-Net architecture with Self-Attention [Saharia et al., 2021]. The network can solve image colorization, inpainting, uncropping, and JPEG restoration tasks. The authors show that the network performs as well or better than task-specific specialist counterparts. The Palette network shows the best results on the inpainting task at the Papers With Code website [1].

In the comparizon, the PSNR, SSIM and FID metrics will be examined as described in Section 3.3.

4.8 Summary

This chapter discusses the research questions, sets the research hypothesis, describes the methodology of comparing the TTSR networks with different networks. Four main hypothesis are formulated which are planned to be evaluated by the experiments in Chapter 6. The Palette network is selected as a candidate for evaluating the inpainting capabilities of the TTSR network.

[1] https://paperswithcode.com/task/image-inpainting

Chapter 5

Datasets

The CUFED dataset, used by the TTSR authors, can be used for the inpainting task. However, it cannot be used for the partial super-resolution task since the input and reference images do not capture the same scene, and we do not have texture information at different spatial resolutions.

An exhaustive search has been made to tackle the partial super-resolution datasets. Many datasets contain the low-resolution images constructed from the corresponding high-resolution images by using some degradation process (blurring, downsampling, etc.). Using such datasets creates a risk of training the model to overcome that degradation process. As the primary dataset for the research, the SR-RAW dataset[Zhang et al., 2019b] was selected. It contains ultra-high-resolution images taken with the same camera at different focus distances, thus achieving different scales (Figure 5.1).

5.1 SR-RAW dataset

The SR-RAW dataset consists of 500 scenes. Each scene has seven images taken from the camera in the same spot but with different focal lengths. Each resolution is 4240x2832.

Let's consider two subsequent image pairs in each scene (for example, 240mm and 150mm in Figure 5.1). The right image, taken with a smaller focal length, can be considered a "zoomed out" version of the left image, containing the same scene as in the left image (yellow bounding box) in smaller resolution and additional resolution information at a wider field of view. The zoomed-out image also captures the scene with a wider field of view (outside the yellow bounding box), highlighting our main task: increase the resolution of an image outside the yellow bounding box, given the high resolution inside of it.

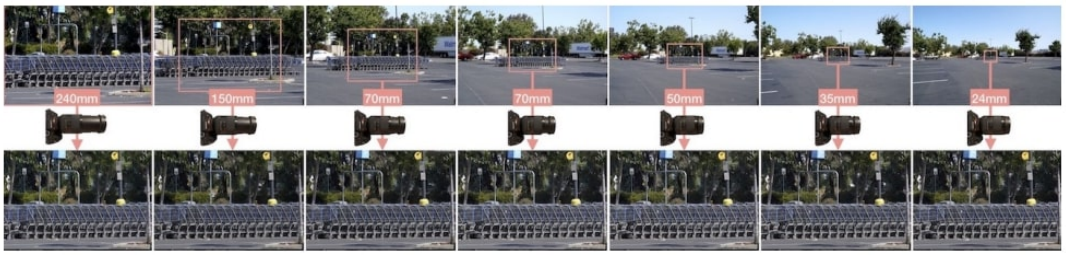


FIGURE 5.1: The SR-RAW dataset sample (the image from [Zhang et al., 2019b]).

FIGURE 5.2: The QD-IMD dataset sample (the image from [Liu et al., 2018]).

The chosen dataset is a good candidate for super-resolution tasks. The main benefit of the SR-RAW dataset compared to all others is that high- and low-resolution images have the same degradation models (noise sources, lens distortions) as in the real-word multi-lens systems (stereo cameras, smartphone cameras, etc.), so the neural network can learn to handle natural image features, not simply "unblur" the image. As an additional feature, the low- and high-resolution pairs of the images are independent of each other (one was not created from another by applying a degradation model).

Due to the lens distortion effects (radial and tangential distortion), it is not straightforward how each pixel of the high-resolution image can be mapped to the low-resolution image. Fortunately, this is one of the concerns addressed by the authors of the original SR-RAW dataset [Zhang et al., 2019b]. The authors provide a script to undistort images and align them to get a simple 1-to-1 mapping which is helpful for the training. Removing the distortion step is vital for the network not to learn about some specific distortion types. In all the experiments below, the SR-RAW images are undistorted at the preprocessing step.

5.2 Quick Draw Irregular Mask Dataset

In order to solve the inpainting task, we should obtain damaged images with the corresponding reconstruction images. The amount and nature of such damage are also subject to discussion.

To get realistic damage configurations, a Quick Draw Irregular Mask Dataset (QD-IMD) was chosen. The dataset contains binary (black-and-white) image masks, which are actual human drawings. The authors' hypothesis is that the combination of strokes drawn by the human hand is a good source of patterns for irregular masks [Liu et al., 2018].

The dataset consists of 50 million images, each in 512x512 resolution. An example of masks is provided in Figure 5.2.

5.3 FFHQ Dataset

We want to ensure that the Texture Transformer Network can successfully solve the super-resolution and inpainting tasks in multiple scenarios. This includes the need to test the network with multiple datasets. So, we will train the network on the SR-Raw dataset and test it on the FFHQ dataset.

The Flickr-Faces-HQ (FFHQ) dataset consists of 70,000 high-quality images at 1024×1024 resolution and contains considerable variation in terms of age, ethnicity and image background. The images were crawled from Flickr, thus inheriting all the

FIGURE 5.3: The FFHQ dataset sample (the image from [Karras, Laine, and Aila, 2019]).

biases of that website, and automatically aligned and cropped [Karras, Laine, and Aila, 2019]. An example of the dataset is presented in the Figure 5.3.

The FFHQ dataset differs from the SR-Raw dataset in a way that it does not contain pairs of images that can be used for the reference-based super-resolution task. So the pairs will be constructed in the next way: each image in the dataset will be considered a reference image, a low-resolution image will be generated by taking the central part of the reference image and upscaling it by 1.4 times by using the bicubic interpolation to ensure similar zoom with the SR-Raw dataset. This way, we will obtain a low-resolution input image and a high-resolution reference image, which have similar properties to the SR-Raw dataset.

It is worth to mention that the degradation model of the low-resolution image is not the same as in the SR-Raw dataset, so the trained network might be unable to perform resolution with the same quality in the FFHQ dataset, but in theory the model should learn different noise sources from the real images in the FFHQ dataset. The assumption of the noise invariability is going to be tested in the experiments.

5.4 Applying the inpainting mask

Applying the inpainting mask to an image is a challenging task. We cannot simply put zero pixels in the input image because it might already contain zero pixels in the dark areas of the image, which cannot be considered damaged. A typical implementation of passing masks is adding a separate channel with the mask to an input image. This way, the model can learn based on data from a separate image channel.

In our case, the decision was made to preserve the structural inputs of the TTSR architecture and pass 3-channel images as an input, the same as the original model. The input image pixel values are mapped from the range [0-255] to [1-255] by assigning all zero pixels a value of 1. This change has a minimal impact on the structural image information, including features and edges; it is also indistinguishable to the human eye. This way, zero pixels in the input image can be exclusively reserved by the mask and recognized by the model accordingly. The experimental evaluation of such design is made in the section 1.

5.5 Experimental configuration

The train dataset chosen for the following experiments is a combination of the SR-RAW dataset and the QD-IMD dataset. The train dataset samples are provided in Table 5.1. The test dataset is a similar combination of the FFHQ dataset and the QD-IMD dataset.

An algorithm for combining images in the train dataset is the following:

TABLE 5.1: Examples of the train dataset: Input and Reference images as an input and Ground Truth as an output

1. Pick a pair of subsequent images from the SR-RAW dataset from the same scene (the high-resolution and the low-resolution images). Run the preprocessing step (undistorting and aligning).

2. Crop a sub-image in both high- and low-resolution images at a random location. This is going to be **the reference image** (both low- and high-resolution parts of it)

3. Crop a sub-image in the low-resolution image close to the reference image. This is going to be **the input image**.

4. Apply the damaging mask to the input image: zero pixels in the locations specified by the mask.

For the test dataset, the algorithm is the same, however the pair of images is selected by manually degrading the FFHQ dataset. This process is explained in details in Section 5.3.

In the following experiments, the image size is 256x256. The input and reference images partially overlap (the overlap area is 20-60%). The overlap simulates the partial super-resolution task, where the high-resolution outcome of the super-resolution task is partially known. Also, the inpainting mask is only applied to the dataset part, in our case, to 60% of input images.

A side effect of overlapping input and reference images is that one part of the inpainting mask lies inside the reference image, while another part is outside the reference image. This means that the model should identify if the missing pixels of the input image are present in the output image and take the textures from there. In case there are no such pixels, the actual inpainting task appears. This facilitates the feature transfer even inside the inpainting task.

The train/validation/test split is performed as follows:

- The training dataset consists of 450 scenes (90%) of the SR-RAW dataset, resulting in 3150 train images.

- The validation dataset consists of 50 scenes (10%) of the SR-RAW dataset, resulting in 350 test images.

- The test dataset consists of 350 images of the FFHQ dataset.

All the metrics in the Experiments chapter are computed for the test dataset.

An essential extension of the training processing is the randomized runtime generation of the training dataset. Since we take a 256x256 sub-image from a 4240x2832 raw image, at each epoch, we select a random part of the image to be cropped, giving us unique training images at every epoch with minimal data duplication.

5.6 Summary

This chapter introduces the Super-Resolution RAW (SR-RAW) and FFHQ datasets for the super-resolution task and the QD-IMD dataset for inpainting, discusses the pros and cons of the datasets and explains the dataset generation for the experiments in Chapter 6.

Chapter 6

Experiments

The Experiments chapter includes validation of the selected architecture, hypothesis verification, and evaluation of results.

6.1 Preparation

Evaluation of the original paper is an important part of the research. It allows to reason about the results reported in the paper, as well as with the hypothesis tested there.

In an attempt to confirm the original paper results, we obtained the source code[1] implementation of the original TTSR paper as well as the original dataset, CUFED, the **CU**ration of **F**lickr **E**vents **D**ataset. The dataset consists of 11'871 train image pairs (image + reference image) and 756 test images. All images have resolution 160x160 pixels.

As already mentioned in the section 3.2, the model can be trained with multiple loss functions. The authors of the original paper use two loss functions combinations and report their training results accordingly:

- train the model with the Reconstruction Loss only;
- train the model with all 3 loss functions (Reconstruction Loss, Adversarial Loss, Perceptual Loss).

6.1.1 Train Texture Transfer Network with a single loss function

Experiment 0.1. Train TTSR using a single loss function (reconstruction loss).

Configuration. batch size: 8, epoch count: 30, train time: 11 hours, $rec_w = 1$, all other = 0

Analysis. We can clearly see that the model can be trained further. The paper's authors say the model needs 200 epochs to achieve a PSNR of 27.09 (we are at 26.69). This seems feasible.

[1] https://github.com/researchmm/TTSR

	Loss coefficients			PSNR*		SSIM*	
Experiment	Rec	Adv	Per	Reported	Ours	Reported	Ours
Experiment 0.1	1	0	0	26.24	26.7	0.784	0.785
Experiment 0.2	1	0.001	0.01	25.6	25.2	0.764	0.755

TABLE 6.1: Comparing the results which were reported by the TTSR authors (Reported) and our results (Ours). * Higher is better.

	Architecture	PSNR*	SSIM*	FID†
Experiment 1	TTSR	**34.89**	0.9014	47.64
Experiment 2	TTSR with reduced feature space	34.23	**0.9031**	**43.90**
Experiment 3	TTSR with soft-attention	34.49	0.8962	47.63
Experiment 4	TTSR with trainable projections	34.15	0.8890	59.36
	No super-resolution	31.33	0.8758	140.2

TABLE 6.2: Comparing the results of different architectures of the super-resolution and inpainting tasks. * Higher is better. † Lower is better.

Loss function, metrics, and samples are attached in Appendix B.

6.1.2 Train Texture Transfer Network with all loss functions

Experiment 0.2. Train TTSR using all loss functions (Reconstruction Loss, Adversarial Loss, Perceptual Loss).

Configuration. batch size: 6, epoch count: 30 + 2 (first two epochs only rec loss, then all losses), train time: 13 hours, $w_{rec} = 1, w_{per} = 0.01, w_{adv} = 0.01$

Analysis. The model performed a lot better than Experiment 0.1. It also seems it can be trained further (because loss functions did not achieve saturation yet).

The PSNR/SSIM metrics of this model drastically differ from the previous (the best PSNR is 25.2 compared to 26.69 in Experiment 0.1); however, the visuals are a lot more plausible. This confirms the sentence of the paper where the authors said that high PSNR does not mean high-quality perception for humans [Yang et al., 2020].

What is interesting is that adversarial loss became negative and decreased.

Loss function, metrics, and samples are attaches as Appendix B.

6.1.3 Summary

By following the train instructions, we confirmed the paper results. Additionally, the authors provide pre-trained network weights that confirm the paper results for the specified task.

6.2 Hypotheses check

In this section the Hypotheses, defined in Sections 4.1 - 4.4, are checked.

6.2.1 Training TTSR for super-resolution and inpainting

Experiment 1. Train the TTSR network for the super-resolution and inpainting tasks by using the SR-RAW and QD-IMD datasets.

The dataset SR-RAW + QD-IMD, introduced in the section 5, was explicitly designed to tackle the hypothesis 4.1 which states that the TTSR architecture can be used to solve the super-resolution and inpainting tasks together. The dataset contains low-resolution images, which need to be enhanced both at the boundaries (the input and reference images do not overlap completely) and inside (inpainting mask applied in the image).

Hardware. 2 x NVIDIA GeForce RTX 3090 24Gb, AMD Ryzen Threadripper 3990X, 128 Gb RAM.

TABLE 6.3: Examples of TTSR solving the super-resolution task (above) and the inpainting task (below)

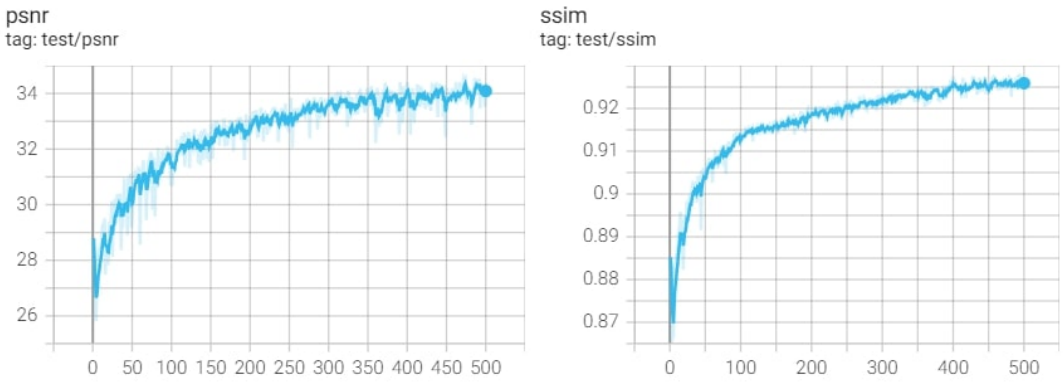


FIGURE 6.1: PSNR and SSIM metrics at each epoch, Experiment 1 (blue)

Configuration: Batch size: 20, epoch count: 500+10, train time: 24 hours, $w_{rec} = 1, w_{per} = 0.01, w_{adv} = 0.01$

Before the experiment, multiple preliminary training runs were performed to obtain optimal epoch count and batch size. The number of epochs is such that there is no visible metrics improvement for the last 100 epochs.

Each model is initially trained with reconstruction loss only for the first ten epochs. This preprocessing step is needed to stabilize the training process and introduce good initial weights for the perceptual and adversarial losses. 500+10 means running ten initial epochs with reconstruction loss and 500 epochs with all losses after that.

Additionally, all the metrics are computed without super-resolution by comparing the low-resolution input image and the corresponding high-resolution ground truth.

Results. The validation dataset metrics PSNR and SSIM are provided in Figure 6.1. On the test dataset, PSNR is 34.89, SSIM is 0.9014, and FID is 47.64. The loss functions are available in Appendix C.

Analysis. The PSNR metric of the input image without any super-resolution is 31.33, while for the output image it is 34.89, which corresponds to a much better similarity with the high-resolution image, indicating that the image resolution has increased. SSIM and FID metrics show huge improvement, too. These results **confirm the hypothesis #1**: the TTSR architecture can be used to solve the super-resolution and inpainting tasks together.

The image quality metrics achieve very high values. For example, the SSIM metric for the texture transfer task in Experiment 0.2 is 0.755 (out of 1), while for the selected problem, it is 0.9252. Additionally, the visual inspection of the model outputs indicates the high quality of super-resolution and inpainting.

Inspecting the model outputs (Table 6.3) shows that TTSR can solve both super-resolution and inpainting tasks successfully.

This model will be called a baseline model. All the subsequent modifications will be compared to it.

6.2.2 Training TTSR with reduced feature space

Experiment 2. Investigate the TTSR network accuracy when using smaller embedding sizes.

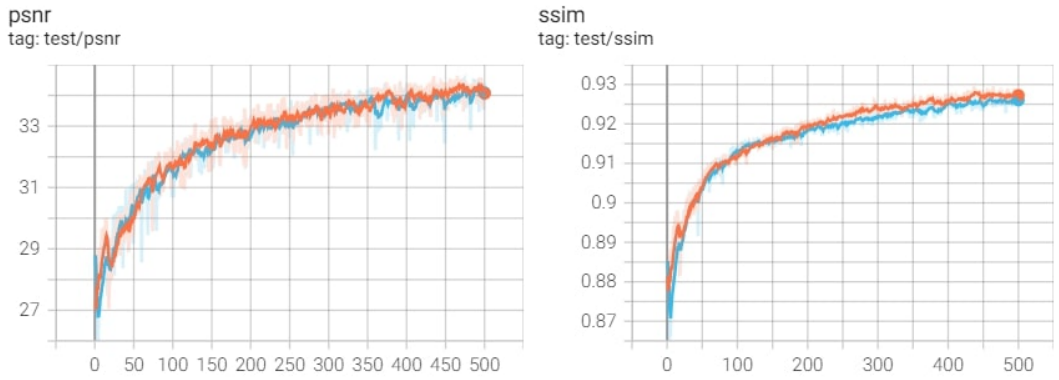


FIGURE 6.2: PSNR and SSIM metrics at each epoch, Experiment 2: embedding size 2304 (blue) and 576 (orange)

Embedding size	Batch size	Training time, hours	PSNR*	SSIM*	FID†
2304 (9×256)	20	24	**34.89**	0.9014	47.64
1152 (9×128)	22	22	34.44	0.9012	46.23
576 (9×64)	24	20	34.49	**0.9031**	**43.90**
288 (9×32)	28	16	34.33	0.8971	48.16

TABLE 6.4: Comparing the results of TTSR, trained with different embedding sizes, Experiment 2. * Higher is better. † Lower is better

Configuration: Embedding size: 2304/1152/576/288, batch size: maximum possible that still fits in the GPU memory, epoch count: 500+10, $w_{rec} = 1, w_{per} = 0.01, w_{adv} = 0.01$

The embedding size reduction is implemented by applying an additional channel-reducing convolutional layer at the output of the Learnable Feature Extraction module. The schematic implementation is provided in Appendix A.

Results. The comparison table with the metrics obtained at different embedding sizes is available in Table 6.4. The comparison of the training performance at different embedding sizes is available in Figure 6.2.

Analysis. The results show that the embedding size of 576 (9×64) results in the same loss function training speed per epoch and the **x1.2 larger batch size**. The reduced embedding size effectively increases the training speed by x1.2 times without sacrificing model accuracy. The model accuracy is very similar to the original embedding size, with PSNR being slightly lower and SSIM slightly higher. However, the FID metric became significantly better, showing that the model focused slightly less on the Attention and texture transfer and focused more on the generative part of the TTSR architecture. On the contrary, embeddings of smaller sizes (9×32) result in worse training accuracy. This is probably because such small embedding is not enough to explain features in an image. This statement needs additional verification outside the scope of this work.

Additional experiments at a different computer with the 6GB GPU show that the 576 embedding size allows processing 6 images in a batch instead of 4 earlier, resulting in x1.5 train speed.

6.2.3 Training TTSR with Soft-Attention

Experiment 3. Investigate the TTSR network accuracy when using the Soft-Attention mechanism for feature transfer.

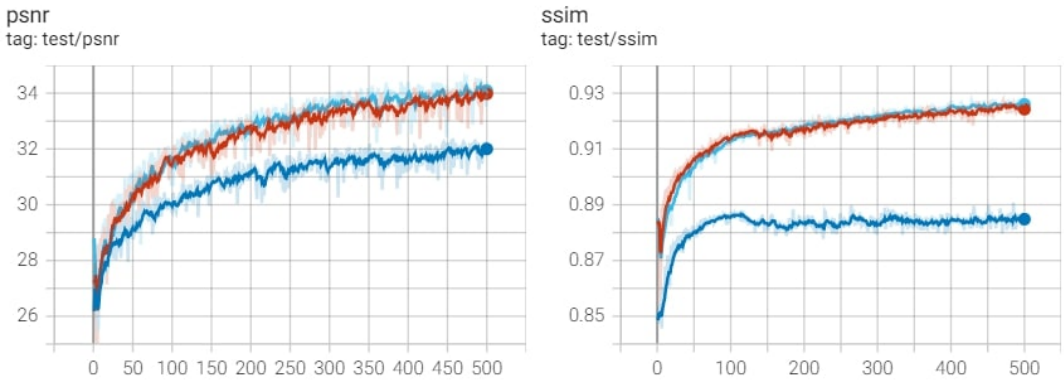


FIGURE 6.3: PSNR and SSIM metrics at each epoch, Experiments 3 and 4: Baseline TTSR (blue), TTSR with Soft-Attention (red) and TTSR with trainable Attention weights (dark blue)

The additional motivation of this experiment is to explore the performance of inference by using Soft-Attention and Hard-Attention approaches. Since Soft-Attention is simple matrix multiplication, it should work faster than Hard-Attention (complex permutations of the matrices).

Configuration. Batch size: 20, epoch count: 500+10, train time: 28 hours, $w_{rec} = 1, w_{per} = 0.01, w_{adv} = 0.01$

The Soft-Attention is applied inside the Learnable Feature Extractor module as explained in [Vaswani et al., 2017]:

$$\text{SoftAttention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{6.1}$$

where Q is an input $LR \uparrow$ image features ("query"), K is the degraded reference $Ref \downarrow\uparrow$ image features ("query") and V is the reference Ref image features ("value"), d_k is the embedding size. The schematic implementation is provided in Appendix A.

Results. The validation metrics PSNR and SSIM are provided in Figure 6.3. The test dataset PSNR is 34.49; SSIM is 0.8962; FID is 47.63.

Analysis. The results show that Soft-Attention resulted in similar model accuracy as the default Hard-Attention. This means that the Soft-Attention mechanism can be used to solve the super-resolution and inpainting tasks. Since the model accuracy is similar, both Attentions can be used interchangeably, and the choice of the specific mechanism might be task-specific. Comparing output images of TTSR and TTSR with Soft-Attention demonstrates that the models work similarly, too (Table 6.3).

The results **partially prove the hypothesis #3**: Soft-Attention indeed can be used for the super-resolution and inpainting tasks. However, it does not make the model observably more accurate. The images are not blurred.

The Soft-Attention modification also highlights the difference between the Texture Transfer and the Partial Super-Resolution and Inpainting tasks. The authors of the Texture Transfer Network state that Soft-Attention cannot be used for their task because it "may cause blur effect which lacks the ability to transfer HR texture features" [Yang et al., 2020]. There seem not to be such artifacts in both Partial Super-Resolution and Inpainting tasks.

From the performance point of view, Soft-Attention and Hard-Attention show different results: the train time of 500 epochs is 24 hours for Hard-Attention and 28

hours for Soft-Attention. This can be explained by the Hard-Attention backpropagation having fewer gradients in the Learnable Texture Extractor module, while in Soft-Attention, all pixels of the Query and Key matrices are connected. Thus, the calculation of the gradients takes a longer time. Additionally, the inference time for Hard-Attention is 190ms per image, while for Soft-Attention, it is 200ms per image, contradicting the initial expectations. In summary, the performance of forward and backward propagation of the fully-connected matrix multiplication is a lot slower that the propagation of indexes shuffle.

6.2.4 Training TTSR with Trainable Projections

Experiment 4. Train the Partial Super-Resolution and Inpainting tasks by leveraging the projective transformations inside the Attention mechanism.

Configuration. Batch size: 20, epoch count: 500+10, train time: 24 hours, $w_{rec} = 1, w_{per} = 0.01, w_{adv} = 0.01$

The linear layers are added to the Learnable Feature Extraction module and applied to the Query and Key matrices before computing the attention map. There are three different Value matrices for different scales inside the LTE module; applying transformations to them is omitted. The schematic implementation is provided in Appendix A.

Results. The evaluation metrics PSNR and SSIM are provided in Figure 6.3. On the test dataset, PSNR is 34.15; SSIM is 0.8890; FID is 59.36.

Analysis. The results show much worse model accuracy than all other architectures: FID is 47.64 for the baseline model and 59.36 for the modification with trainable projections. The same goes with SSIM: the difference is larger than comparing to all other experiments. The evaluation metrics show that PSNR still shows the potential to grow while SSIM stopped growing after the epoch 100. This means that the inferred images become more and more blurred while not adding any additional structural information to the image. This is confirmed by comparing the TTSR and TTSR with Trainable Projections network outputs (Table 6.3).

The potential reason of the model degradation is that additional linear transformations for query and key possibly moved features into separate spaces and made dot products less meaningful. Projections could improve performance if the input vectors represented different things. The provided results indicate that the usage of trainable projections for Attention for the partial super-resolution and inpainting tasks is questionable. More experiments in this area are needed. **The hypothesis #4 cannot be proved**.

6.3 Evaluation of results

6.3.1 Evaluation of TTSR for inpainting

Experiment 5. Evaluate and compare the inpainting task accuracy of the TTSR network with the Palette network.

Dataset. To facilitate fair terms for inpainting for both networks, we will modify the test dataset so that the input and reference images do not overlap. Such dataset change will make both TTSR and Palette inpaint image parts they have never seen before. This also means that the reference image will not always contain high-resolution textures, which can be transferred to the low-resolution input image.

	HR			LR		
Architecture	PSNR*	SSIM*	FID†	PSNR*	SSIM*	FID†
TTSR	24.29	0.8357	61.20	25.95	**0.8851**	80.08
Palette	**28.96**	**0.8485**	**26.76**	**32.00**	0.8849	**40.28**

TABLE 6.5: Comparing the results of different architectures of the inpainting task for HR (High-res ground truth) and LR (Low-res ground truth). * Higher is better. † Lower is better.

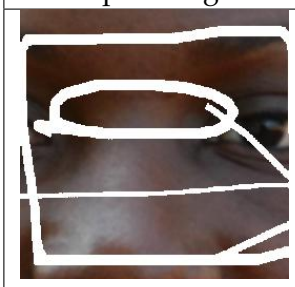
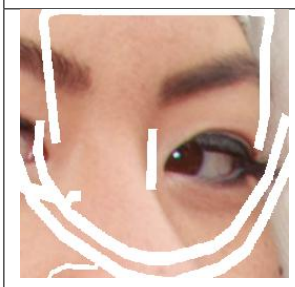
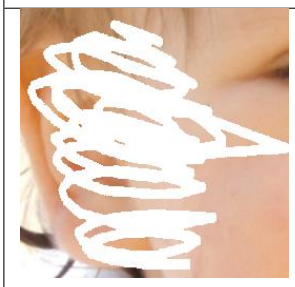
Input image	Reference image	TTSR	Palette
			
			
			
			

TABLE 6.6: Comparizon of the inpainting task results: Input and Reference images as an input and outputs of the TTSR and Palette networks

Metrics. Since TTSR does both super-resolution and inpainting, and Palette does inpainting only, ground truths for these networks are different. Understanding the difficulties of this comparison, we compared both networks with two ground truth images: A high-resolution output image (an expected output for TTSR) and a low-resolution input image with mask pixels inpainted (an expected output of Palette). We are going to compute metrics for both ground truths and discuss them. We are going to compute PSNR and SSIM metrics **only for the inpainted pixels**. Due to the complicated implementation of FID, we are going to compute the FID score for the full images.

Configuration. The weights of the TTSR network are taken from the Experiment 1. The weights for the Palette network are available at the project website[2]. Weights from the Places2 dataset were used.

Results. The train metrics PSNR and SSIM are provided in Figure 6.5. TTSR and Palette inputs are available in Table 6.6.

Analysis. The results show that both TTSR and Palette networks can successfully solve the inpainting task with the specified dataset.

When comparing the metrics of the high-resolution image as ground truth (Figure 6.5), both networks show average accuracy, with Palette being significantly better. High PSNR of the Palette network shows it produces colors which describe the expected image better; the SSIM values are similar, indicating the networks reconstruct similar features. The reason why the color are different might be TTSR overfitting for the SR-Raw dataset: with the same exposure time, different focal distances mean different amount of light reaching the camera sensor, which might cause color bias to be remembered by the model.

The FID value is a lot better in Palette, which is probably caused by the imperfection of the FID metric in this experiment. TTSR changes pixels outside the inpainting mask, resulting in score impact of those pixels to be non-zero while its always zero for Palette which does not change these pixels. Nevertheless, the FID value of 60 shows a good result for TTSR, this is at the same level as Trainable Projections experiment before, but with a much harder dataset: there are no high-resolution textures to transfer features from in most cases.

When comparing the metrics of the low-resolution image as ground truth, both TTSR and Palette show better results. The SSIM of both networks got significantly higher, indicating that the structural information in the inpainted parts of an image is closer to the low-resolution image than the reference image. Additionally, Palette now generates correct color intensities of the inpainted parts, which fall into the color gamma of the input image, while TTSR colors are still far off. This is illustrated by the increased PSNR for Palette while only model increase for TTSR.

Table 6.6 shows the results of inpainting of both TTSR and Palette networks. First row of the table illustrates that both networks deal well with the inpainting task: both model can identify semantic features of an image (eyes, eyebrows, nose) and inpaint them properly.

The second row of the table illustrates a systematic Palette issue: when an inpainting mask consists of thin lines, the network produces artifacts. On the other side, the same case is a favorable scenario for TTSR, which can transfer textures on multiple scale levels and inpaint really nice features.

The two last rows in the table demonstrate the issues of TTSR. The first issue is already mentioned above: the input and reference images of the SR-Raw dataset

[2] https://github.com/Janspiry/Palette-Image-to-Image-Diffusion-Models

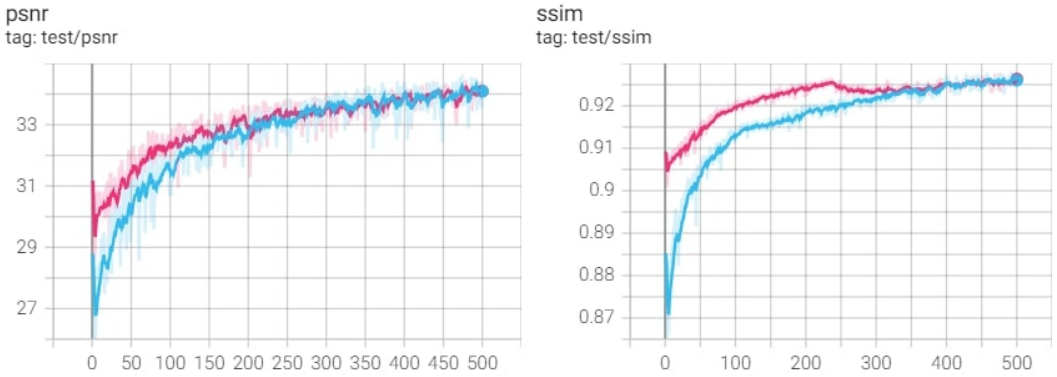


FIGURE 6.4: PSNR and SSIM metrics at each epoch, Experiment 6: Default-initialized weights (blue) and Texture-Transfer initialized weights (pink)

have different color intensities, and in case the reference image does not have relevant high-resolution textures, a constant color bias is added to the input image. This is exactly the case in the third row in the table: there is no skin in the reference image, so in the TTSR output the whole skin becomes more red. The last row shows that TTSR was not trained well on the large inpainting masks: the model cannot identify the image semantic features properly and decide where to transfer high-resolution textures from. The resulting features become pixelated and contain artifacts.

It is worth mentioning that the state-of-the-art inpainting networks, including Palette, can reconstruct huge areas. In this work, we apply relatively small inpainting masks as proof that the proposed architecture can solve the inpainting task in principle. After the baseline hypothesis is proved, it makes sense to explore inpainting capabilities of the TTSR architecture at the larger mask sizes.

6.3.2 Training TTSR with pre-trained data

Motivation. One of the features of the TTSR model is that it is already used to solve a Texture Transfer task. This domain is related to the Partial Super-Resolution and Inpainting tasks. Thus, we can conclude that all the mentioned tasks might have similar feature extraction module weights.

There is not much scientific novelty in this experiment. However, by using weights of one task to train a model for a similar task, we can achieve faster training speed and potentially better accuracy due to the Transfer learning mechanism.

Experiment 6. Take a TTSR model weights from the result of the Texture Transfer training and use them as the starting point of the Partial Super-Resolution and Inpainting training process.

Configuration: Batch size: 20, epoch count: 500+10, train time: 24 hours, $w_{rec} = 1, w_{per} = 0.01, w_{adv} = 0.01$

Results. The train metrics PSNR and SSIM are provided in Figure 6.4.

Analysis. The results confirm the assumption of having a Transfer learning effect here: the model showed higher accuracy than the default-initialized weights. This shows that initializing training weights properly is an important part of the model architecture.

A notable improvement happens with the training accuracy: the model shows high PSNR and SSIM values from the beginning of the training process, achieving

the highest accuracy at the 200-th epoch, compared to the 400-500 epochs needed for the baseline model to achieve the same accuracy. This means that by selecting appropriate initial weights **the training time can be reduced by a factor of 2**.

6.4 Summary

This chapter describes conducted experiments and analyses their results. The results confirm the hypothesis 1-3 and show that the TTSR network can successfully solve the super-resolution and inpainting tasks. Comparing the TTSR and Palette networks for the inpainting task shows that TTSR is capable of solving the task. The pros and cons of the network decisions are discussed.

Chapter 7

Conclusions

The thesis studies image transformer networks for the tasks of super-resolution and inpainting. The main research task is to generalize the Texture Transformer Network (TTSR) for solving the super-resolution and inpainting tasks simultaneously.

The experiments confirm that the TTSR network can be used to solve the super-resolution and inpainting tasks. Additional research shows that the number of parameters in the model's encoder is too large and can be reduced, preserving the model's accuracy. Using Soft-Attention for feature transfer produces similar results as the original Hard-Attention approach. Adding trainable projections does not improve the model accuracy in the experimental setup.

Exploring the inpainting capabilities of the TTSR network shows that the model produces visually plausible results and, in combination with high-resolution textures, can produce better results than specialized inpainting networks. When comparing the isolated inpainting capabilities of the TTSR network, the competitor inpainting network shows better results.

The thesis results add more understanding to the domain of multi-task transformer super-resolution networks. The obtained results might be used by other researchers to conduct studies in the related domains.

Appendix A

Code listings

The full implementation of all Attention layers is available at Github[1]. The simplified implementation of the Attention forward passes:

```
import torch
import torch.nn as nn
import torch.nn.functional as F

# default implementation
def HardAttention(in_feats_unfold, ref_feats_unfold):
    ...

# reduced embedding size
def HardAttentionEmbedding(in_image, ref_image, embedding_dim=64):
    # in_image          -> (*, H, W, 3)
    # in_feats          -> (*, H, W, 256)
    # in_feats_reduced  -> (*, H, W, embedding_dim)
    # in_feats_unfold   -> (*, H*W, embedding_dim*9)

    in_feats = LearnableTextureExtractor(in_image)
    ref_feats = LearnableTextureExtractor(ref_image)

    # in __init__:
    # self.reduce_channels =
    #       nn.Conv2d(in_channels=256, out_channels=embedding)

    in_feats_reduced = self.reduce_channels(in_feats)
    ref_feats_reduced = self.reduce_channels(ref_feats)

    in_feats_unfold = F.unfold(in_feats_reduced, kernel_size=(3, 3))
    ref_feats_unfold = F.unfold(ref_feats_reduced, kernel_size=(3, 3))

    return HardAttention(in_feats_unfold, ref_feats_unfold)

def SoftAttention(in_image, ref_image):
    # in_image          -> (*, H, W, 3)
    # in_feats          -> (*, H, W, 64)
    # in_feats_unfold   -> (*, H*W, 64*9)
    # attn              -> (*, H*W, H*W)
```

[1] https://github.com/romanus/TTSR/blob/master/model/SearchTransfer.py

```
    in_feats = LearnableTextureExtractor(in_image)
    ref_feats = LearnableTextureExtractor(ref_image)

    in_feats_unfold = F.unfold(in_feats_reduced, kernel_size=(3, 3))
    ref_feats_unfold = F.unfold(ref_feats_reduced, kernel_size=(3, 3))

    # Q*K^T
    attn = torch.matmul(ref_feats_unfold.transpose(1, 2), in_feats_unfol

    # (Q*K^T)/sqrt(d_k)
    embedding_dim = in_feats_unfold.size(1)
    attn *= 1/math.sqrt(embedding_dim)

    # softmax((Q*K^T)/sqrt(d_k))
    attn = F.softmax(attn, dim=1)

    return attn

# trainable projections
def HardAttentionProjections(in_image, ref_image):
    # in_feats          -> (*, H, W, 64)
    # in_feats_unfold   -> (*, H*W, 64*3*3)

    in_feats = LearnableTextureExtractor(in_image)
    ref_feats = LearnableTextureExtractor(ref_image)

    in_feats_unfold = F.unfold(in_feats_reduced, kernel_size=(3, 3))
    ref_feats_unfold = F.unfold(ref_feats_reduced, kernel_size=(3, 3))

    # in __init__:
    # self.w_qs = nn.Linear(64*9, 64*9, bias=False)
    # self.w_ks = nn.Linear(64*9, 64*9, bias=False)

    in_feats_unfold = self.w_qs(in_feats_unfold.transpose(1, 2))
    ref_feats_unfold = self.w_ks(ref_feats_unfold.transpose(1, 2))

    return HardAttention(in_feats_unfold, ref_feats_unfold)
```

Appendix B

Reproducing the original paper

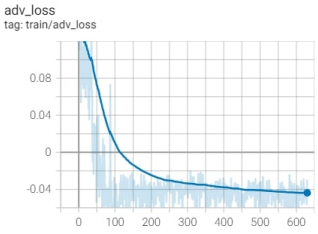
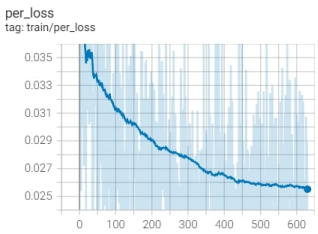
Reconstruction loss	Adversarial loss	Perceptual loss
		

TABLE B.1: Loss functions of Exp. 0.1 (single loss function, first line) and Exp. 0.2 (all loss functions, second line): 20 ticks/epoch

Input image	Reference image	Inferred result
		
		

TABLE B.2: Example images of Exp. 0.1 (single loss function, first line) and Exp. 0.2 (all loss functions, second line)

Appendix C

Training TTSR

The loss functions for experiments 1-4 are presented. During each epoch, the cumulative loss function was collected and cleared up at the beginning of the next epoch.

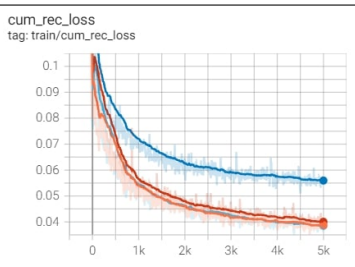
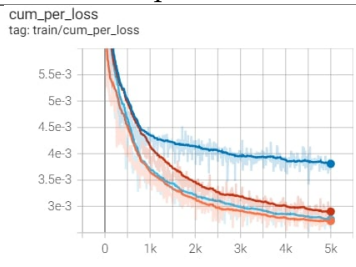
Reconstruction loss	Perception loss	Adversarial loss
		

TABLE C.1: Combined loss functions for experiments 1-4: TTSR (light blue), TTSR with reduced feature domain (orange), TTSR with Soft-Attention (red), TTSR with Trainable Projections (dark blue)

We observed the strange behavior of the adversarial loss being unstable. The training parameters and coefficients were the same as the original Texture Transformer network, but the original network does not show such behavior. This means the model hyperparameters may be tuned to observe better accuracy and a more stable training process.

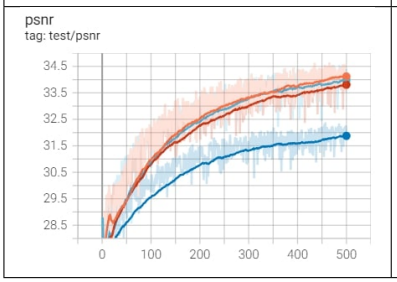
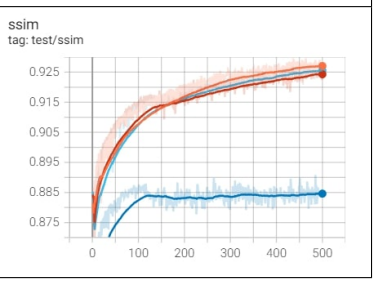
PSNR	SSIM
	

TABLE C.2: Combined quality metrics for experiments 1-4: TTSR (light blue), TTSR with reduced feature domain (orange), TTSR with Soft-Attention (red), TTSR with Trainable Projections (dark blue)

Bibliography

Barnes, Connelly et al. (2009). "PatchMatch: A randomized correspondence algorithm for structural image editing". In: *ACM Trans. Graph.* 28.3, p. 24.

Carles, Guillem, James Downing, and Andrew R Harvey (2014). "Super-resolution imaging using a camera array". In: *Optics letters* 39.7, pp. 1889–1892.

Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama (2004). "Region filling and object removal by exemplar-based image inpainting". In: *IEEE Transactions on image processing* 13.9, pp. 1200–1212.

Dong, Chao et al. (2014). "Learning a deep convolutional network for image super-resolution". In: *European conference on computer vision*. Springer, pp. 184–199.

El Gheche, Mireille et al. (2016). "Texture reconstruction guided by a high-resolution patch". In: *IEEE Transactions on Image Processing* 26.2, pp. 549–560.

Gulrajani, Ishaan et al. (2017). "Improved training of wasserstein gans". In: *arXiv preprint arXiv:1704.00028*.

Gur, Eran and Zeev Zalevsky (2007). "Iterative Single-Image Digital Super-Resolution Using Partial High-Resolution Data." In: *World Congress on Engineering*, pp. 630–634.

Haris, Muhammad, Gregory Shakhnarovich, and Norimichi Ukita (2018). "Deep back-projection networks for super-resolution". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1664–1673.

Heusel, Martin et al. (2017). "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30.

Hidane, Moncef et al. (2014). "Super-resolution from a low-and partial high-resolution image pair". In: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 2145–2149.

Hidane, Moncef et al. (2016). "Image zoom completion". In: *IEEE Transactions on Image Processing* 25.8, pp. 3505–3517.

Johnson, Justin, Alexandre Alahi, and Li Fei-Fei (2016). "Perceptual losses for real-time style transfer and super-resolution". In: *European conference on computer vision*. Springer, pp. 694–711.

Karras, Tero, Samuli Laine, and Timo Aila (2019). "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410.

Ledig, Christian et al. (2017). "Photo-realistic single image super-resolution using a generative adversarial network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690.

Liu, Guilin et al. (2018). "Image inpainting for irregular holes using partial convolutions". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 85–100.

Malczewski, K and R Stasiński (2009). "Super resolution for multimedia, image, and video processing applications". In: *Recent Advances in Multimedia Signal Processing and Communications*. Springer, pp. 171–208.

Mao, Haiyi et al. (2016). "Super resolution of the partial pixelated images with deep convolutional neural network". In: *Proceedings of the 24th ACM international conference on Multimedia*, pp. 322–326.

Menon, Sachit et al. (2020). "Pulse: Self-supervised photo upsampling via latent space exploration of generative models". In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 2437–2445.

Nazeri, Kamyar et al. (2019). "Edgeconnect: Generative image inpainting with adversarial edge learning". In: *arXiv preprint arXiv:1901.00212*.

NVIDIA (2020). *NVIDIA DLSS 2.0: A Big Leap in AI Rendering*. https://www.nvidia.com/en-gb/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering/. [Online; accessed 31-May-2022].

Oktay, Ozan et al. (2016). "Multi-input cardiac image super-resolution using convolutional neural networks". In: *International conference on medical image computing and computer-assisted intervention*. Springer, pp. 246–254.

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

Reed, Scott et al. (2022). "A Generalist Agent". In: *arXiv preprint arXiv:2205.06175*.

Saharia, Chitwan et al. (2021). "Palette: Image-to-image diffusion models". In: *arXiv preprint arXiv:2111.05826*.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Wang, Xintao et al. (2018). "Esrgan: Enhanced super-resolution generative adversarial networks". In: *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0.

Wang, Xintao et al. (2021). "Real-esrgan: Training real-world blind super-resolution with pure synthetic data". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1905–1914.

Wronski, Bartlomiej et al. (2019). "Handheld multi-frame super-resolution". In: *ACM Transactions on Graphics (TOG)* 38.4, pp. 1–18.

Wu, Bichen et al. (2020). "Visual transformers: Token-based image representation and processing for computer vision". In: *arXiv preprint arXiv:2006.03677*.

Yang, Fuzhi et al. (2020). "Learning texture transformer network for image super-resolution". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5791–5800.

Yıldırım, Deniz and Oğuz Güngör (2012). "A novel image fusion method using IKONOS satellite images". In: *Journal of Geodesy and Geoinformation* 1.1, pp. 75–83.

Yu, Fisher and Vladlen Koltun (2015). "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122*.

Yu, Jiahui et al. (2018). "Generative image inpainting with contextual attention". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5505–5514.

Zeng, Yanhong et al. (2021). "Aggregated Contextual Transformations for High-Resolution Image Inpainting". In: *arXiv preprint arXiv:2104.01431*.

— (2022). "Aggregated contextual transformations for high-resolution image inpainting". In: *IEEE Transactions on Visualization and Computer Graphics*.

Zhang, Han et al. (2019a). "Self-attention generative adversarial networks". In: *International conference on machine learning*. PMLR, pp. 7354–7363.

Zhang, Xuaner et al. (2019b). "Zoom to learn, learn to zoom". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3762–3770.

Zhang, Yulun et al. (2018). "Image super-resolution using very deep residual channel attention networks". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 286–301.

Zhang, Zhifei et al. (2019c). "Image super-resolution by neural texture transfer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7982–7991.

Zheng, Haitian et al. (2018). "Crossnet: An end-to-end reference-based super resolution network using cross-scale warping". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 88–104.