AALBORG UNIVERSITY

STUDENT REPORT

Master's Thesis
Mathematical Engineering

Combining Algorithm Unrolling with Self-Supervised Learning

for Image Super-Resolution

Authors:
Andreas Kühne Larsen
Mads Arnløv Jørgensen
Magnus Jónhardsson

Supervisors:
Christophe Biscio
Zheng-Hua Tan

June 2nd 2023

Dept. of Mathematical Sciences
Skjernvej 4A
DK-9220 Aalborg Ø
http://math.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Combining Algorithm Unrolling with Self-Supervised Learning

Theme:
Image Super-Resolution

Project Period:
September 2022 to June 2023

Project Group:
MATTEK 4.105A

Authors:
Andreas Kühne Larsen
Mads Arnløv Jørgensen
Magnus Jónhardsson

Supervisors:
Christophe Biscio
Zheng-Hua Tan

Numbered Pages: 86

Date of Completion:
June 2nd 2023

Abstract:

Deep learning methods have shown to outperform model-based methods at image upscaling, and exhibit state-of-the-art performance. However, deep learning techniques suffer from drawbacks such as immense data requirements, computational costs, lack of interpretability/explainability, and overfitting. In an attempt to address these issues, the compatibility of algorithm unrolling and self-supervised learning is explored. First, the consequences of utilising the SSL framework data2vec to train a network inspired by ISTA-Net is examined. Then, replacing the linear projection in the encoder of vision transformers with LISTA and pre-training using the masked autoencoder framework, is investigated. Results for the first approach indicate that pre-training an ISTA-Net network using data2vec, might lead to increased generality in scarcely annotated scenarios, however strict attributions remain impossible. The results for the second approach indicate an increased performance in all the experiments. However, to solidify this result, an ablation study on how much can be attributed to the unrolled algorithm over an increased parameterisation, must be conducted.

Preface

This project is written in the period 01/09/2022 to 02/06/2023 by the group MATTEK 10 - 4.105A, attending the final semesters of the masters' programme in Mathematical Engineering at Aalborg University. For the algorithms attached, `Python 3.8` along with `PyTorch 1.13` are used - specifically, the PyTorch NGC Container V22.09. The figures shown throughout the project are created using `Tikz` and `matplotlib`.

The group would like to thank Christophe Biscio and Zheng-Hua Tan for supervision and guidance throughout the period of writing the project. The group would also like to extend our gratitude to Yonina C. Eldar for consultation and academic sparring, as well as CLAAUDIA for the access to and support with cloud HPC.

Andreas Kühne Larsen

Mads Arnløv Jørgensen

Magnus Jónhardsson

Contents

1 | Problem Analysis

1.1 Introduction

Continued pursuit of higher quality imaging is often hindered by the increasing costs of sensor-chips and optical components [Farsiu et al., 2004]. Additionally, certain imaging systems may be limited by past technological constraints, as it can be impractical or cost-prohibitive to upgrade their hardware components. In this context, super-resolution (SR) techniques offer a compelling solution to overcome these limitations by leveraging post-processing, and is thus widely used in the industry.

Consider, for example, satellite imaging. Upgrading lens components is extremely impractical because of their placement in orbit, thus if the requirements change during deployment, SR might be able to compensate. Similarly, consider security cameras, which are often acquired en mass and thus required to be low-cost. Because of their low cost, the camera hardware might not provide a resolution suitable for tasks such as person or vehicle identification, unless they are aided by SR. Thus, SR techniques can provide a viable alternative to costly hardware upgrades.

1.2 Image Super-Resolution

This section is mainly based on *Image Super-Resolution as Sparse Representation of Raw Image Patches* by Yang et al. [2008].

Formally, SR is the process of increasing the resolution of a signal through post-processing. Despite significant SR research, this still remains a challenging problem in some fields due to factors such as noise, blur, and limited data availability. In the context of images, SR pertains to the recovery of a high-resolution image given a low-resolution representation, and is often performed by assuming a prior on the relationship between the high- and low-resolution image. SR is theoretically possible by modelling the complete photographic process, from the initial reflection of light on a surface to the arrival of digital data on a computer. This process involves numerous distortions, noise processes, and non-linear transformations that are difficult to model accurately. Therefore, simplifications and assumptions can be made to create a tractable model that can be computed efficiently, often leading to a sufficiently good solution. To facilitate this, note that digital images can be stored as 3-dimensional tensors representing the light-intensities of the three primary colours red, green, and blue. Now assume that a high-resolution image $x \in \mathbb{R}^{h \times w \times c}$, and a low-resolution image $y \in \mathbb{R}^{h' \times w' \times c}$ are flattened such that they are represented by the vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ respectively. Then the photographic process can be

approximated by a linear model, represented by the over-determined set of equations:

$$\mathbf{y} = \Phi\mathbf{x}, \tag{1.1}$$

where $\Phi \in \mathbb{R}^{m\times n}$ is called the forward-projection matrix.

The general task of recovering $\mathbf{x}$ given its linear projection $\mathbf{y}$, is referred to as the linear inverse problem. The traditional solution to Equation (1.1) is given by the least squares problem

$$\arg\min_{\mathbf{x}} \|\Phi\mathbf{x} - \mathbf{y}\|_2^2. \tag{1.2}$$

However, since the forward-projection matrix Φ is wide, the problem is ill-conditioned, which makes a solution to the least squares problem, if it exists, meaningless. This can be overcome through regularisation to stabilise the solution. The following approach to regularisation is based on results from compressive sensing, which ensure that, under some assumptions, the high-resolution image can be precisely recovered from its low-resolution representation.

Given an over-complete dictionary matrix $\Psi \in \mathbb{R}^{n\times k}$, assume the image $\mathbf{x} \in \mathbb{R}^n$ can be written as a linear combination of atoms

$$\mathbf{x} = \Psi\boldsymbol{\alpha},$$

where $\boldsymbol{\alpha} \in \mathbb{R}^k$ is a vector with very few ($\ll k$) non-zero entries. Then an observation $\mathbf{y}$ of $\mathbf{x}$ can be modelled as

$$\mathbf{y} = \Phi\mathbf{x} = \Phi\Psi\boldsymbol{\alpha},$$

where $\Phi \in \mathbb{R}^{m\times n}$ is the projection matrix with $m < n$. The equation $\mathbf{x} = \Psi\boldsymbol{\alpha}$ is under-determined for the sparse coefficient vector $\boldsymbol{\alpha}$, and thus $\mathbf{y} = \Phi\Psi\boldsymbol{\alpha}$ is as well. However, the sparsest solution to the equation is unique if the dictionary satisfies the restricted isometry property [Foucart and Rauhut, 2013, p 133-135]. Consequently, it is possible to perfectly recover the high-resolution image $\mathbf{x}$ from a low-resolution image, assuming $\mathbf{x}$ is represented by a sufficiently sparse vector $\boldsymbol{\alpha}$. However, the restricted isometry property is hardly ever fulfilled, and though it can be relaxed if approximate solutions are sufficient, it is still very difficult to validate. Consequently, the coherence [Foucart and Rauhut, 2013, p. 111,114] is often used instead, as it is much easier to compute.

By utilising a sparsity prior for regularisation, the least squares problem in Equation (1.2) can be replaced with the pursuit problem

$$\hat{\boldsymbol{\alpha}}_P = \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \quad \text{s.t.} \quad \|\Phi\Psi\boldsymbol{\alpha} - \mathbf{y}\|_2^2 \leq \epsilon,$$

where $\|\cdot\|_0$ counts the number of non-zero elements. Having found the sparse representation the high-resolution image can be reconstructed as

$$\hat{\mathbf{x}} = \Psi\hat{\boldsymbol{\alpha}}_P.$$

Solving the pursuit problem is NP-hard, but the problem can be relaxed by substituting the ℓ_0 norm with the ℓ_1 norm. The problem then becomes

$$\hat{\boldsymbol{\alpha}}_{BPD} = \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_1 \quad \text{s.t.} \quad \|\Phi\Psi\boldsymbol{\alpha} - \mathbf{y}\|_2^2 \leq \epsilon.$$

The solutions to the ℓ_0 and ℓ_1 problems are known to agree for sufficiently sparse $\boldsymbol{\alpha}$ [Donoho, 2006]. The problem can alternatively be formulated using Lagrange multipliers:

$$\hat{\boldsymbol{\alpha}}_{BPD} = \arg\min_{\boldsymbol{\alpha}} \lambda\|\boldsymbol{\alpha}\|_1 + \frac{1}{2}\|\Phi\Psi\boldsymbol{\alpha} - \mathbf{y}\|_2^2, \tag{1.3}$$

where λ dictates the trade-off between sparsity and reconstruction discrepancy. This problem is called *basis pursuit denoising* (BPD). Thus, a solution to the SR task can be found by solving the BPD problem and reconstructing the high-resolution image with the corresponding dictionary.

1.3 Iterative Soft Thresholding Algorithm

The *Iterative Soft Thresholding Algorithm* (ISTA) is a simple, widely used optimisation method for solving the BPD problem in Equation (1.3). The core concept of ISTA involves updating the solution vector iteratively by performing an entrywise soft-thresholding operation. The iterative nature of ISTA allows for sequential refinement of the solution at each step, which, under certain conditions, leads to convergence towards the true sparse solution.

ISTA belongs to a popular class of optimisation algorithms known as proximal gradient methods, which are used to solve, possibly non-smooth, convex optimisation problems. Consider a general, non-smooth, solvable, convex optimisation problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x}), \tag{1.4}$$

where $f, g : \mathbb{R}^n \to \mathbb{R}$ are convex functions, and ∇f is L-Lipschitz (Appendix A.1). Since the function g is possibly non-smooth, gradient methods which assume differentiability might fail, however, it is possible to alleviate this issue by utilising proximal gradient methods which rely on the proximal operator (see Appendix A.2). The proximal gradient method arrives at a solution to Equation (1.4) from a starting point $\mathbf{x}^{(0)}$ by sequential application of the update step

$$\mathbf{x}^{(k)} = \text{prox}_{\lambda g}(\mathbf{x}^{(k-1)} - \lambda \nabla f(\mathbf{x}^{(k-1)})), \tag{1.5}$$

where $\lambda > 0$ can be interpreted as a step size.

When $f(\mathbf{x}) = \|A\mathbf{x} - \mathbf{y}\|_2^2$ and $g(\mathbf{x}) = \|\mathbf{x}\|_1$, i.e. the BPD problem, the update step in Equation (1.5) reduces to the ISTA iteration step

$$\boldsymbol{\alpha}^{(k)} = S_{\lambda/L}\left(\mathbf{x}^{(k-1)} - \frac{1}{L}A^T(A\mathbf{x}^{(k-1)} - \mathbf{y})\right),$$

where

$$S_\lambda(\mathbf{x})_i = \text{sign}(x_i)\max(|x_i| - \lambda, 0) \tag{1.6}$$

is the entry-wise soft-thresholding function and L is the Lipschitz constant of ∇f, which is an upper bound on the largest eigenvalue of $A^\intercal A$ [Beck and Teboulle, 2009]. The soft-thresholding operation promotes sparsity by reducing the magnitude of every entry.

1.4 Deep Learning

The BPD problem is traditionally solved using model-based optimisation algorithms, such as ISTA. Recent developments have shown deep learning to be a powerful alternative to model-based methods, relying on increased parameterisation to be applicable to a wider range of optimisation problems, including the BPD problem. The primary advantage of deep learning methods is their ability to approximate complex mappings by fitting

over-parametrised models to a dataset. This is achieved through the utilisation of deep neural networks (DNN), which permit learning abstract representations of the data. DNNs are capable of adapting to a multitude of different problems by fitting to data, making them incredibly versatile. For example, Dong et al. [2015] proposed a deep learning framework for SR, composed of a convolutional neural network (CNN) taking low-resolution images as input and producing corresponding high-resolution images, that, at the time, demonstrated state-of-the-art restoration quality.

DNNs have exhibited impressive performance in numerous fields, highlighting their great capacity to advance state-of-the-art. As a result, there has been considerable interest and investment in deep learning research, with many academic institutions establishing dedicated research groups focused on the development and implementation of deep learning methods[1][2]. However, deep learning methods also have a set of challenges and limitations which must be considered carefully when they are applied to real-world problems.

Some of the major issues DNNs face are:

- Overfitting: Because of over-parameterisation, DNNs are sometimes capable of remembering the entire dataset, acting more like a dictionary than a model describing the problem. Consequently, this results in decreased performance when presented with unseen data.
- Data and training: The most prominent method to increase the inference performance of DNNs is to increase their size and parametrisation, which in turn requires more data. Specifically, high-quality data is needed to obtain powerful models, which is expensive to acquire. Furthermore, the training of large DNNs is computationally costly and requires a lot of time.
- Explainability and interpretability: The inherent black-box nature of DNNs makes it difficult to interpret the causality of predictions and representations. Achieving high levels of accuracy is sometimes irrelevant without the ability to reason for the outcome, particularly in critical applications such as medical diagnosis.

Explainability & Interpretability

In the context of DNNs *explainability* refers to the knowledge that a specific parameter represents and how important is is to the performance of the model. Although similar, *interpretability* refers to the ability to determine cause and effect of the entire model.

[Johnson, 2020]

This project focuses on improving some of these limitations in the context of super-resolution. To facilitate this, some prevalent deep learning techniques will be examined.

1.5 Algorithm Unrolling

The progress and practical deployment of deep learning methods is heavily inhibited by the large data requirement and inherent black-box nature. In contrast to deep learning methods,

[1] `https://www.aicentre.dk/the-centre-p1`
[2] `https://research.google/teams/brain/`

model based methods rely on domain knowledge to design models. By hand-crafting models based on domain knowledge, model-based methods result in effective models, where each step of the procedure has an explicit purpose. Consequently, ensuring both explainability and interpretability.

An important emerging deep learning technique called *Algorithm Unrolling* (AU) attempts to alleviate the data requirements and black-box nature of DNNs by recasting iterative algorithms as neural networks. In the seminal work by Gregor and LeCun [2010], they propose a neural network architecture based on existing iterative BPD solvers to speed up sparse coding. One of the iterative solvers that they treat is ISTA, which is subsequently unrolled into *learned ISTA* (LISTA). LISTA is thus an AU method that can be used to solve the SR problem, as it is an iterative BPD solver.

AU inherits the inductive biases from the iterative algorithm which can decrease parameterisation, alleviating the data size requirements and increasing inference speed. Thus, AU enables a parsimonious DNN model which can represent complex mappings as a sequence of explainable operations, that can be optimised using back-propagation.

AU is however not without disadvantages. For example, unrolling can result in the loss of certain properties of the underlying iterative algorithm, such as convergence or stability guarantees, which are important in many applications. Additionally, effective training and initialisation schemes for AU networks are not thoroughly researched [Monga et al., 2019].

1.6 Self-Supervised Learning

Until recently, supervised learning (SL), which requires large annotated datasets, was the predominant approach for training DNNs. One of the major factors inhibiting the further improvement of deep learning models in many domains is the availability of annotated data. Annotations can be prohibitively expensive and/or time consuming to acquire, particularly in domains such as medical imaging, video analysis, and natural language processing (NLP). It is also difficult to create descriptive annotations that encompass all the important target features. Further, the performance of SL models is heavily dependent on the quality and resolution of annotations (think of an incorrect translation or unspecific dog breed) and the diversity of the dataset needs to be sufficient as to not over-represent any particular input-feature.

An emerging technique in deep learning called self-supervised learning (SSL) facilitates training using annotations derived programatically from unannotated data, these are referred to as pseudo-annotations. The pseudo-annotations are generated from some intrinsic structure in the data, which SSL then leverages to learn meaningful representations. These representations can subsequently be used in tasks such as image classification, object detection, and NLP.

An SSL framework typically involves a self-supervised pretext- and supervised downstream-task which are solved sequentially. The aim of the pretext-task is to learn useful representations of the data that can be beneficial for downstream-tasks. Successful design of pretext-tasks typically necessitates prior knowledge of the underlying structure of the data. Additionally, the design process must take into account the representation requirements of the downstream-task, as different pretext-tasks can induce different invariance properties within the learned representations. Note that the pretext-task does not need to be directly related to the downstream-task.

An example of a pretext-task is *masked prediction*. Masked prediction involves training models to predict missing information given a partial view of the input. In NLP, this can involve predicting hidden words in a sentence based on the surrounding text, motivating a representation that captures the contextual and semantic meaning of words. Such learned representations can be utilised in downstream-tasks like sentiment analysis and machine translation. Similarly, in computer vision, masked prediction may involve training a model to predict masked regions of an image from the surrounding pixels, thereby motivating the development of representations that capture the visual semantics of objects and scenes. Masked prediction methods rely on the assumption that missing information in the data can be inferred from the context. An illustration of the masked prediction process in NLP can be seen in Figure 1.1, where the masked word to be predicted is illustrated as a black box.

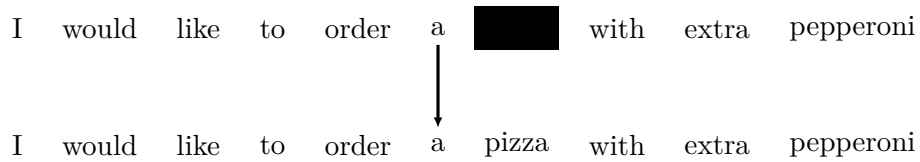

Figure 1.1: Masked prediction pretext-task for NLP, where the masked word to be predicted is illustrated as a black box.

Another explanatory example pretext-task is the jigsaw puzzle. The jigsaw task involves partitioning an image into equally sized tiles and shuffling them, and then training a model to arrange the tiles back into their original configuration. The goal is then for the learned representation to capture the spatial relationship between objects and scenes. The model trained on the jigsaw task can then be used for downstream-tasks such as object recognition, image segmentation, and classification. An illustration of the jigsaw process can be seen in Figure 1.2.

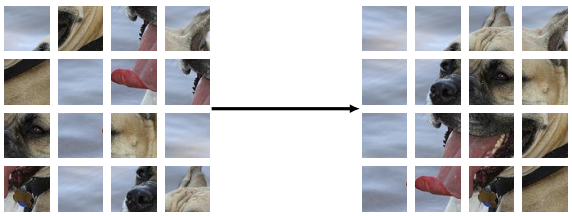

Figure 1.2: Illustration of the jigsaw pretext-task.

It is important to be aware of possible shortcomings of a pretext-task and how it relates to the downstream-task. Consider for example the jigsaw task: The model can solve this task by only checking the edges of each tile and then aligning the tiles such that the edges agree. In this case the model has not learned the underlying structure of the images or any semantics related to it, and the performance in a downstream-task may deteriorate.

For computer vision, a pretext-task referred to as contrastive learning has garnered significant attention. One of the most successful contrastive learning methods is SimCLR [Chen et al., 2020], which has produced state-of-the-art results on several benchmark datasets, including ImageNet.

Overall, the use of SSL enables transfer learning based on large amounts of unannotated data, reducing the reliance on annotations. By learning useful representations of the data by solving a pretext-task, the model can then more efficiently be applied to a wide range

of downstream-tasks, making SSL a versatile and valuable tool in the field of machine learning.

1.7 Problem Statement

In the context of super-resolution, the traditional approaches utilise model based solvers such as ISTA. However, deep learning approaches have demonstrated superior performance compared to model based methods. Despite their success, deep learning methods present certain notable drawbacks compared to model based methods, i.e. enormous data requirements, large computational cost, lack of explainability/interpretability, and a tendency towards overfitting. To further advance the field of super-resolution and address these issues, it might be beneficial to explore alternative approaches.

This report focuses on combining two prominent deep learning techniques, namely AU and SSL, as a novel approach to solving the SR problem. AU allows for construction of more parsimonious models which have a higher degree of explainability/interpretability. By training the models using self-supervision, it becomes more feasible to deploy them in domains with limited annotated data. Within this context, two distinct approaches for integrating AU and SSL emerge. The first approach involves using a prevalent SSL training framework to train an unrolled network. The second approach involves introducing an unrolled algorithm as a module to an existing DNN architecture that is traditionally used in SSL. By examining both approaches, this report aims to deepen our understanding, and explore the feasibility, of the combined utilisation of algorithm unrolling and self-supervised learning in the context of solving the super-resolution problem.

This leads to the following problem statement:

How can algorithm unrolling and self-supervised learning be combined to construct and train a deep neural network capable of solving the super-resolution task?

2 | Decision Making Algorithms

This chapter is based on *Model-Based Deep Learning: On the Intersection of Deep Learning and Optimization* by [Shlezinger et al., 2022].

The theory of algorithm unrolling is typically formulated as operating on the intersection between iterative algorithms and neural networks. Thus, to describe algorithm unrolling in a more rigorous framework, consider the unifying framework of decision making. Decision making is underpinned by the group of algorithms, broadly referred to as decision making algorithms. The design of such algorithms comprises two parts: a model of a problem, and the corresponding solver. Traditionally, a problem is described with a mathematically and computationally tractable model by assuming some prior. The corresponding solver then relies on domain knowledge to find the best solution. This approach to algorithm design is generally referred to as model-based methods. Alternatively, the design can take a data-centric approach, with the benefit that the solver can be learned end-to-end from data. State-of-the-art data-centric solvers mainly utilise deep neural networks (DNN) as their representation power allows them to be model agnostic [Abiodun et al., 2018].

The goal of decision making is to design a decision rule $f : \mathcal{X} \to \mathcal{S}$, with observations $x \in \mathcal{X}$ and decisions $s \in \mathcal{S}$. The design procedure comprises three steps:

1. Choose the decision rule family.
2. Tune the parameters of the family.
3. Evaluate the decision rule.

The design procedure is illustrated in Figure 2.1.

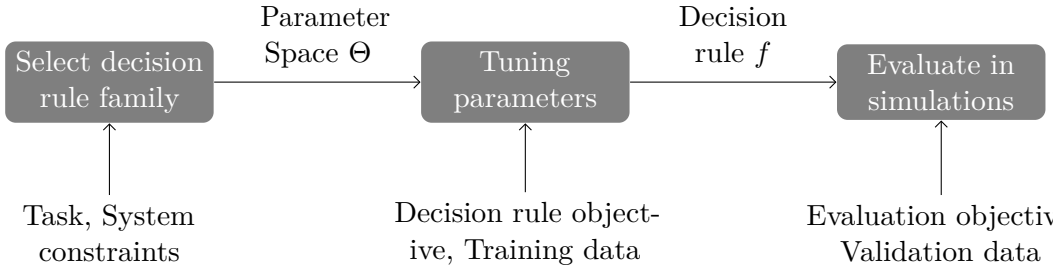

Figure 2.1: Illustration of the decision rule design procedure [Shlezinger et al., 2022].

Decision rules can be categorised into types depending on their formulation and assumptions. The focus of this chapter will be on the following two decision rule types:

- Iterative algorithms: An iterative algorithm computes a decision from a sequence of mappings $h^{(t)} : \mathcal{S} \times \mathcal{X} \to \mathcal{S}$, i.e. it iteratively computes decisions $s^{(t)} = h^{(t)}(s^{(t-1)}, x)$ until convergence, starting with an initial decision $s^{(0)}$. The algorithm is said to have converged when some error bound is reached. Alternatively, the algorithm is terminated after a fixed number of iterations $T \in \mathbb{N}$, thus the decision becomes $s = h^{(T)}(h^{(T-1)}(\cdots h^{(1)}(s^{(0)},x),x),x)$.
- Neural Networks: Traditional feed-forward neural networks (NN) are a special case of iterative algorithms utilising $t = 1, \ldots, T, T \in \mathbb{N}$ mappings $h^{(t)}(z) = \sigma(W^{(t)}z + b^{(t)})$, where σ is a non-linear function, and $\{W^{(t)},b^{(t)}\}$ are parameters of $h^{(t)}$. The decision s is then computed as $s = h^{(T)}(h^{(T-1)}(\cdots h^{(1)}(x)))$, i.e. the decision rule f comprises nested applications of $h^{(t+1)}$ on the intermediate decisions $s^{(t)} = h^{(t)}(s^{(t-1)})$. Each separate application of the mapping $h^{(t)}$ is referred to as a network *layer*, and the number of layers is always fixed.

Examples of other common decision rule types are affine transformations, decision trees, and optimisation-based decisions. The parameters of a decision rule are denoted θ, and the process of selecting the correct parameters is called *tuning*. Tuning can be performed by utilising domain knowledge related to the task, or based on some algorithm using synthetic or real data. Tuning the parameters of a decision rule using data is referred to as *training*, and is the approach for neural networks. The set $\mathcal{F} = \{f(\cdot;\theta)|\theta \in \Theta\}$ containing the decision rule for all choices of parameters $\theta \in \Theta$ is called the decision rule *family*. Typically, a larger parameter space Θ allows for more general and broad decision rule mappings.

After selecting a decision rule, its performance can be evaluated either from observation-decision pairs not used in the tuning process, or through simulations. Performance is subsequently determined by an evaluation objective $l : \mathcal{F} \times \mathcal{X} \times \mathcal{S} \to \mathbb{R}^+$. The objective l thus evaluates a decision rule $f(\cdot;\theta)$, from the observations x and corresponding desired decisions s^*, by assigning a score. However, for tuning the decision rule family a surrogate to l that admits optimisation may be required since the evaluation objective can be highly complex or abstract. The surrogate objective used for optimisation over the parameter space Θ is called the decision rule objective $\mathcal{L} : \mathcal{F} \to \mathbb{R}^+$. When the evaluation objective is well behaved, then it may be used as the decision rule objective. In the context of NNs the decision rule objective is typically referred to as the loss function. There are different approaches to the decision rule design procedure which depend on the assumptions and priors available to solve the underlying problem.

2.1 Model-Based Methods

Model-based methods are characterised by mathematical models that utilise domain knowledge to describe the problem. To ensure a tractable optimisation problem, it is often necessary to simplify the model, as accurate statistical models relating observations to decisions are in general difficult to obtain. To decrease model discrepancy, model-based methods also impose additional assumptions on the simplified model. Though model-based decision rules typically possess a low parameterisation, they can be applied to a broad range of observations as long as the underlying assumptions of the simplified model hold. Consequently, it is often possible to explicitly state when a model based method will work. This allows model-based methods to be both explainable and interpretable. Model-based methods have solvers with structures derived from the formulation of the model. Commonly, decision rule families are explicit solutions or iterative solvers. Tuning is performed by

optimising an analytic decision rule objective $\mathcal{L}$, which is derived from an understanding of the model and the evaluation rule objective l [Shlezinger et al., 2022].

Example 2.1 (Iterative Shrinkage Thresholding Algorithm)
Suppose a decision $s \in \mathcal{S}$ is related to the observation $x \in \mathcal{X}$ by $x = D(s)$, where $D : \mathcal{S} \to \mathcal{X}$. Determining the decision from the observation is in this case known as the inverse problem, which is ill-posed when D is not invertible. However, by assuming some prior, a solution can be found by minimising

$$E(x,s) = d(x,D(s)) + R(s),$$

where d is a metric and R is a regulariser to the solution founded in the prior. Given the linear inverse problem $\mathbf{x} = D\mathbf{s}$ with $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{s} \in \mathbb{R}^m$, and $D \in \mathbb{R}^{n\times m}$, the solution can be found by assuming sparsity as a prior and minimising

$$E(\mathbf{x},\mathbf{s}) = \|\mathbf{x} - D\mathbf{s}\|_2^2 + \lambda\|\mathbf{s}\|_1.$$

[Beck and Teboulle, 2009, p.184]

This can be solved using the iterative shrinkage thresholding algorithm (ISTA), which recursively determines the decision by applying

$$\mathbf{s}^{(t)} = h^{(t)}(\mathbf{s}^{(t-1)}) = S\Big(\frac{1}{L}D^\intercal\mathbf{x} + \Big(I - \frac{1}{L}D^\intercal D\Big)\mathbf{s}^{(t-1)}; \tau\Big), \tag{2.1}$$

where $S(v;\tau) = \text{sign}(v)\max(|v| - \tau, 0)$ is the soft-shrinkage function with parameter τ, and $L \in \mathbb{R}$ is the Lipschitz constant of $\nabla\|\mathbf{x} - D\mathbf{s}\|_2^2$ which is an upper bound on the eigenvalues of $D^\intercal D$ [Beck and Teboulle, 2009, p.191]. Thus, the decision rule family is defined over $\Theta = \{\tau \in \mathbb{R}, D \in \mathbb{R}^{n\times m}\}$. Depending on the modality of the observations, different domain knowledge can be exploited to tune the parameters.

2.2 Data-Based Methods

Data-based methods, such as neural networks, select the decision rule mapping by training, and are in general model agnostic. In the case of neural networks, the decision rule is highly parameterised, and thus belongs to a family applicable in a wide range of scenarios. Consequently, the resulting decision rule may be subject to overfitting. This generally occurs when the parameterisation is too large relative to the diversity of data used for training. Thus, data-based models often require large datasets and do not generalise well to out-of-distribution data. To mitigate issues such as overfitting, regularisers, restricting the parameter space, can be introduced to the decision rule objective working in tandem with the empirical measures. Note however, that backpropagation [Werbos, 1990] is the traditional training mechanism for neural networks, which necessitates that the decision rule objective has a defined gradient.

The complex and generic structure of neural networks results in lack of both explainability and interpretatbility, making it is very difficult to infer the rationale behind decisions, and identify decision rule limitations and potential failure cases. Consequently, it is difficult to determine which parts of the network are required, and which are largely redundant.

As interpretable systems allow for concise and efficient regularisation design, which can increase performance in inference tasks, it is desired to introduce this property to neural networks.

2.3 Algorithm Unrolling

Model-based and data-based methods are fundamentally different approaches and vary significantly in specificity and representation power. However, both are data-reliant parametric mappings, and can be described in the same decision-making framework. This motivates the idea of combining different aspects of both approaches to realise a new balanced decision rule type. One such type is called algorithm unrolling (AU), which recasts the iterations of an iterative algorithm as layers in a neural network. This allows parameters to be learned end-to-end from data, while exploiting the domain knowledge introduced by a model. The general process of unrolling an algorithm is illustrated in Figure 2.2

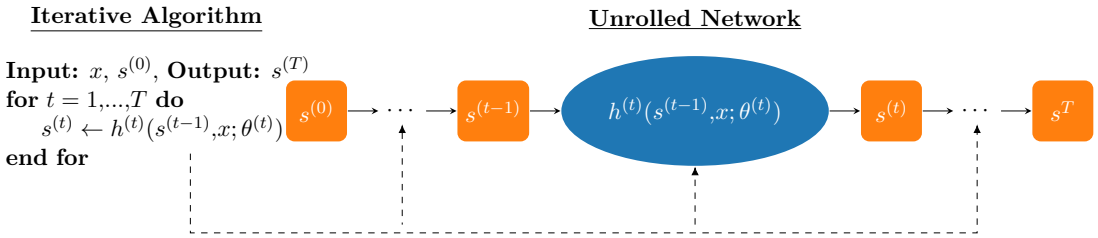

Figure 2.2: Illustration of the general unrolling process. Each iteration is mapped to a single layer, which is then stacked to form a DNN.

Iterative algorithms correspond to models with high bias and low variance, whereas NNs corresponds to models with high variance and low bias [Monga et al., 2019]. Unrolled networks are desired to be an intermediary between the two. Furthermore, by unrolling an iterative algorithm its representation power increases, since the parameterisation of the decision rule is increased. Typically, unrolled networks have less representation power than conventional NNs, but generalise better [Monga et al., 2019]. AU generally reaches similar performance to its iterative algorithm counterpart, while using significantly fewer iterations, thus AU can improve inference speed.

Contrary to iterative algorithms, which traditionally share parameters between iterations, unrolled networks do not necessarily share weights between layers, as each parameter is separately determined from training. Learning iteration-dependent parameters $\{\theta^{(t)}\}_{t=1}^{T}$, where T is the number of unrolled iterations, allows the network to admit accurate decision rules within a predefined number of iterations [Shlezinger et al., 2022]. Iteration-dependent parameters, increase the parameterisation and abstractness of the decision rule compared to sharing parameters. However, unrolled networks are less parametrised and more task specific than traditional NNs. This typically results in AU requiring less data for training. By training using data obtained from the true underlying system, the increased parameterisation allows unrolled networks to better overcome possible inaccuracies of the simplified model.

Some potential benefits of unrolled networks are thus, increased interpretability and lower parameterisation compared to traditional NNs, as well as faster inference compared to model-based methods. A comparison of all the presented decision rule types is seen in Table 2.1.

Approach	**Parameterisation**	**Interpretability**	**Generalisability**	**Inference**
Iterative Algorithm	Low	High	High	Slow
Generic Neural Network	High	Low	Low	Fast
Algorithm Unrolling	Medium	Medium	Medium	Fast

Table 2.1: Feature comparison between different decision rule types. [Monga et al., 2019]

Example 2.2
The model-based method ISTA, detailed in Example 2.1, can be unrolled into a DNN architecture called *Learned-ISTA* (LISTA) [Gregor and LeCun, 2010].

To motivate this, notice that iterations of ISTA, described in Equation 2.1, consist of linear operations followed by the non-linear soft-thresholding operation. This is similar to the architecture of a multi-layer perceptron with ReLU activations. Thus, the LISTA architecture can be constructed by recasting the linear operations in each ISTA iteration into a fully connected layer with ReLU activations, and subsequently stacking T layers to form a DNN. In other words, the decision rule computes a decision $\mathbf{s}$ from an observation $\mathbf{x}$ and initial decision $\mathbf{s}^{(0)}$ by nested applications of Equation (2.1):

$$\begin{aligned}\mathbf{s} &= f(\mathbf{x}) \\ &= S\Big(W_d^{(T)}\mathbf{x} + W_e^{(T)}\Big(\cdots S_{\tau^{(1)}}\Big(W_d^{(1)}\mathbf{x} + W_e^{(1)}\mathbf{s}^{(0)}; \tau^{(1)}\Big)\Big); \tau^{(T)}\Big),\end{aligned}$$

where the implicit substitutions of parameters

$$\begin{aligned}\frac{1}{L}D^\intercal &\mapsto W_d^{(t)}, \\ I - \frac{1}{L}D^\intercal D &\mapsto W_e^{(t)},\end{aligned}$$

has been performed to further expand the representation power. For LISTA the decision rule family $\mathcal{F}$ is defined for f over $\Theta = \{\tau^{(t)}, W_d^{(t)}, W_e^{(t)}\}_{t=1}^{T}$.

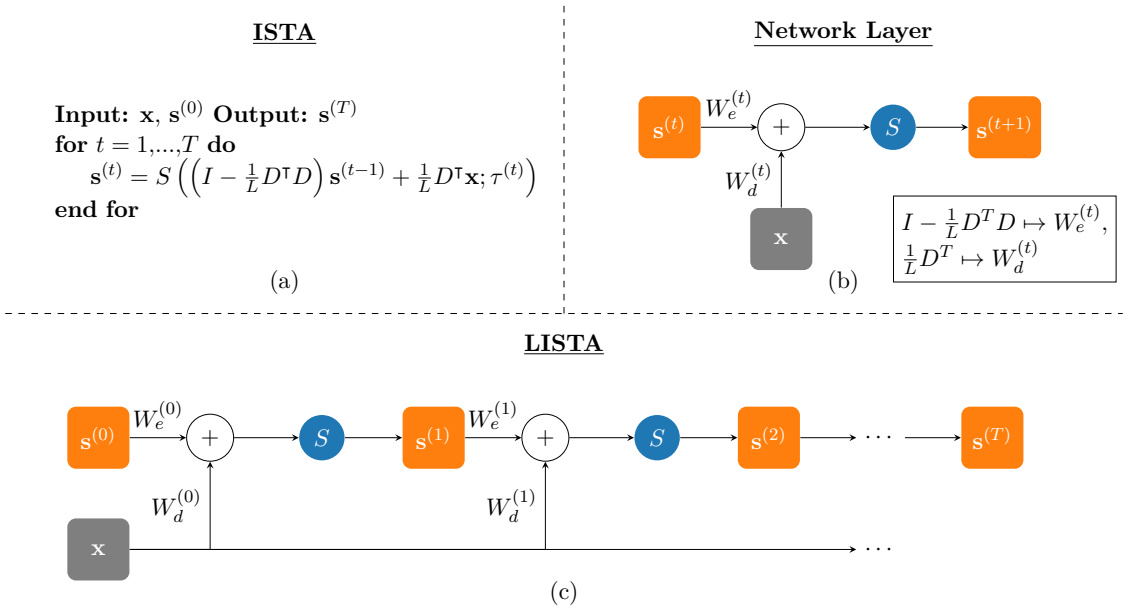

Figure 2.3: Illustration of unrolling procedure behind LISTA. A deep unrolled network is formed by stacking T ISTA iterations recast into network layers. (a) Iterative algorithm. (b) A single unrolled iteration. (c) Unrolled network.

3 | Self-supervised Learning

Self-supervised learning is a training paradigm for deep neural networks that utilises pseudo-labels as supervisory signals during training. These pseudo-labels are programmatically obtained from unlabelled data. There are various strategies for the pseudo-label generation procedure, some of which are outlined in this chapter. The purpose of self-supervised learning is to more effectively utilise the vast amount of unlabelled data available to train deep models. A fundamental engine of SSL is the data augmentation process which steers the learning of the model.

In this chapter all the neural networks are assumed to apply a function f on the form $f = t_\psi \circ h_\lambda$ where $h_\lambda : \mathcal{X} \to \mathcal{Z}$ is some feature extractor with parameters $\lambda \in \Lambda$ and $t_\psi : \mathcal{Z} \to \mathcal{S}$ is an output module with parameters $\psi \in \Psi$. The spaces $\mathcal{X}, \mathcal{Z}$, and $\mathcal{S}$ represent the observation, representation, and decision spaces respectively. The space of augmented data is denoted $\tilde{\mathcal{X}}$ and the space of the pseudo-labels is denoted $\tilde{\mathcal{S}}$.

3.1 Training Pipline

This section is based on *Self-Supervised Representation Learning: Introduction, advances, and challenges* by Ericsson et al. [2022].

SSL is typically not applied by itself but incorporated into a training pipeline consisting of a pretext- and subsequently downstream task. The goal of the pretext task is to design a model that produces meaningful representations by training on self-supervisory signals. The downstream model then utilises these representations, typically along with human-supervisory signals, to solve a specific task.

Definition 3.1 (Pretext and Downstream Task)
Let $\mathcal{X}$, and $\mathcal{S}$ be the space of observations and decisions respectively. Let $\mathcal{P} : \mathcal{X} \to \tilde{\mathcal{X}} \times \tilde{\mathcal{S}}$ be a process which generates pseudo-labelled data, and let

$$\mathcal{F}_p = \left\{ k_\gamma \circ h_\lambda : \tilde{\mathcal{X}} \to \tilde{\mathcal{S}} \mid \gamma \in \Gamma, \lambda \in \Lambda \right\},$$

be a family of decision rules, where $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{S}}$ are the space of augmented data and pseudo-labels respectively. Define a task

$$\arg\min_{\gamma,\lambda} \mathcal{L}_p(k_\gamma \circ h_\lambda), \tag{3.1}$$

for the objective $\mathcal{L}_p : \mathcal{F}_p \to \mathbb{R}^+$.

Similarly define a family of decision rules

$$\mathcal{F}_d = \{ g_\phi \circ h_\lambda : \mathcal{X} \to \mathcal{S} \mid \phi \in \Phi, \lambda \in \Lambda \},$$

with the objective $\mathcal{L}_d : \mathcal{F}_d \to \mathbb{R}^+$, and the task

$$\arg\min_{\phi,\lambda} \mathcal{L}_d(g_\phi \circ h_\lambda). \tag{3.2}$$

Equation (3.1) and (3.2) then defines a pretext- and downstream task respectively.

In a DNN with model $f = t_\psi \circ h_\lambda$, the feature extractor h_λ is referred to as the trunk, and the output module t_ψ the head. The head of a model is task-specific, whence the distinction between k_γ and g_ϕ in Definition 3.1. The training pipeline utilising SSL consists of the following steps:

1. With observations $x \in \mathcal{X}$ and decisions $s \in \mathcal{S}$, obtain annotated data $\mathcal{D}_a = \{(x_i,s_i)\}_{i=1}^M$ for the downstream task and unannotated data $\mathcal{D}_u = \{x_i\}_{i=1}^N$, with $N \gg M$, for the pretext task.
2. Define a pretext task with a pseudo-label process $\mathcal{P}$, objective $\mathcal{L}_p$, and corresponding head k_γ.
3. Construct the data set $\mathcal{D}_p = \mathcal{P}(\mathcal{D}_u)$.
4. Determine $\lambda^\star$ by solving the pretext task on $\mathcal{D}_p$.
5. Replace the head k_γ from the pretext model with a head g_ϕ for the downstream application.
6. Determine (ϕ^*,λ^*) by solving the downstream task on $\mathcal{D}_a$ with $h_{\lambda^\star}$.

The training pipeline is illustrated in Figure 3.1.

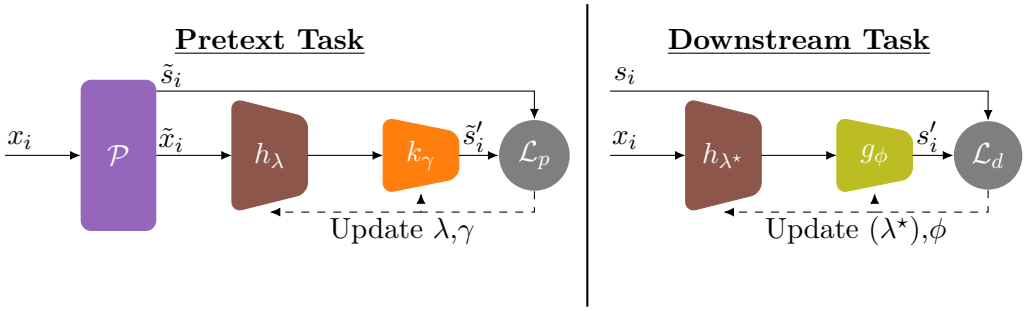

Figure 3.1: DNN training pipeline incorporating SSL. Notice that, λ does not necessarily get updated in the downstream task.

3.1.1 Linear Readout and Fine-tuning

There are two main approaches to using a pre-trained trunk to solve the downstream task: *linear readout* and *fine-tuning*.

Suppose $(\lambda^\star,\gamma^\star)$ are the weights of a pre-trained model $k_{\gamma^\star} \circ h_{\lambda^\star}$. In linear readout the trunk weights $\lambda^\star$ are frozen while the head's weights are optimised, thus the training

objective is

$$\arg\min_{\phi} \mathcal{L}_d(g_\phi \circ h_{\lambda^\star}),$$

where $\mathcal{L}_d$ is the downstream objective. It is dubbed linear readout as the output layer of the head g_ϕ is typically a linear function.

In fine-tuning both the trunk and head weights are optimised jointly, and the downstream training objective is

$$\arg\min_{\lambda,\phi} \mathcal{L}_d(g_\phi \circ h_\lambda)$$

where the trunk is initialised in $\lambda^\star$.

The choice between using linear readout or fine-tuning depends on the amount of available annotated data. Linear readout is generally preferable when annotations are few. However, the parameterisation of the head, i.e. the number of trainable weights, may be advantageously increased for better performance, given more annotated data. Fine-tuning is the preferable approach to fit the downstream model parameters in the case of mismatched pretext and downstream tasks, unsuitable pretext training data, and/or sufficient annotated data. Both methods can effectively transfer knowledge from the pretext task to the downstream task. [Ericsson et al., 2022]

3.2 Pretext Tasks

To derive self-supervisory signals, SSL relies heavily on data augmentations such as masking, adding noise, or modality-specific augmentations like rotations and crops. In Definition 3.1, $\tilde{x} \in \tilde{\mathcal{X}}$ represents an augmented view of the observation $x \in \mathcal{X}$. Thus, the pseudo-label process $\mathcal{P}$ associated with the pretext task also defines the augmentation process. In order for the pretext model to learn meaningful representations, it must be able to recognise and exploit the intrinsic structure in the data. The augmentation process must therefore be designed to facilitate this, and consequently some degree of domain knowledge is required. As alluded to in Section 3.1.1, the pretext task and the downstream task are related by more than just sharing a trunk. Consider the downstream task of optical character recognition. If the augmented view $\tilde{x}$ is a rotation of the image x, then the pretext task facilitates representations agnostic to rotations. Consequently, the downstream task is significantly harder, since characters such as "6" and "9", or "d" and "p" become indistinguishable. Furthermore, it is crucial that the augmentation process induces a sufficiently complex task, otherwise the pretext model may fail to learn meaningful representations.

3.2.1 Masked Prediction

This augmentation strategy relies on the assumption that the missing information from a partial view of an observation can be inferred from context, given that the held-out information is intrinsically related to the partial view. Given a data set $\mathcal{D}_u = \{x_i\}_{i=1}^N$, the pseudo-label process $\mathcal{P}$ returns the data set $\mathcal{D}_p = \{(\tilde{x}_i, \tilde{s}_i)\}_{i=1}^N$, where $\tilde{x}_i$ is a partial view of x_i and the pseudo-label $\tilde{s}_i$ is the remaining components of x_i [Ericsson et al., 2022]. This process is illustrated in Figure 3.2. Since the dimensions of $\tilde{x}_i$ and x_i are different, it can cause problems in the downstream task if the trunk does not process input as sequences. This can be mitigated by changing the pseudo-label process to produce augmented views $\tilde{x}_i$ of the same dimension as x_i, where the masked components are represented with some

mask token. The design of the mask token is non-trivial and various considerations must be made. A mask token can, for example, be the value 0, indicating lack of information. There are however issues with this token, such as skewing the distribution of the input. The strategy for selecting which values to mask can vary between implementations, such as masking related or random information. Furthermore, the proportion of the signal that is masked can also be changed. The pretext model is trained to minimise a suitable objective $\mathcal{L}_p$, such as the empirical mean squared error:

$$\lambda^\star,\gamma^\star = \arg\min_{\lambda,\gamma} \frac{1}{N} \sum_{(\tilde{x}_i,\tilde{s}_i)\in\mathcal{D}_p} (k_\gamma(h_\lambda(\tilde{x}_i)) - \tilde{s}_i)^2.$$

Masking is an effective data augmentation strategy for natural language processing, computer vision, and speech Baevski et al. [2022]. The difficulty of solving the pretext task is intimately related to the masking strategy and ratio, which heavily depend on the modality of the data. Some results have shown that for images, it is important to mask a large proportion of the input in order to learn meaningful representations, since images are highly information redundant. Conversely, natural language is a highly information dense modality, and therefore a high masking ratio can result in the inference task being too complex. He et al. [2021]

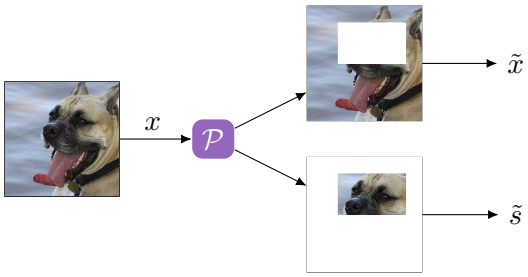

Figure 3.2: Pretext task pseudo-label process for masked prediction.

3.2.2 Transformation Prediction

By assuming that inputs have a canonical view, transformation prediction learns meaningful representation by perturbing the input [Ericsson et al., 2022]. For natural images, the canonical view might be dictated by gravity creating a notion of direction, thus rotations can be an obvious perturbation method.

Given an unlabelled data set $\mathcal{D}_u$ containing data points in canonical view, the pseudo-label process $\mathcal{P}$ applies a transformation T_ω, with parameter $\omega \in \Omega$, to the data points. The resulting pseudo-labelled dataset is thus $\mathcal{D}_p = \{(\tilde{x}_i,\tilde{s}_i)\}_{i=1}^N$, where $\tilde{x}_i = T_{\omega_i}(x_i)$ and $\tilde{s}_i = \omega_i$. This process is illustrated in Figure 3.3. The pretext model is then trained to predict the transformation parameter by minimising an objective such as the categorical cross entropy loss:

$$\lambda^\star,\gamma^\star = \arg\min_{\lambda,\gamma} \frac{1}{N} \sum_{(\tilde{x}_i,\tilde{s}_i)\in\mathcal{D}_p} - \sum_{\omega\in\Omega} \tilde{s}_i(\omega) \log(k_\gamma(h_\lambda(\tilde{x}_i))(\omega)),$$

when the pseudo-labels $\tilde{s}_i$ and predictions $k_\gamma(h_\lambda(\tilde{x}_i))$ are considered PMFs. Note, that the categorical cross entropy loss is only valid given a discrete set of transformations Ω.

In this case Ω comprises distinct classes, where each class represents a specific transformation parameter. The DNN is then tasked with predicting which class was used for the transformation, rather than the specific transformation parameter.

There are some considerations regarding the utilisation of transformation prediction. Specifically, in the absence of a canonical view of the input with respect to the applied transformations, the pretext model will fail to learn meaningful representations of the data. Furthermore, by utilising transformation prediction the pretext model learns to produce representations that maintain the notion of the applied transformation in latent space. Hence, the representations of the model become equivariant to the applied transformation, i.e., for a function $f : \mathcal{X} \to \mathcal{S}$, and two transformations $T_\omega : \mathcal{X} \to \mathcal{X}$ and $T'_{\omega'} : \mathcal{S} \to \mathcal{S}$ the equivariance property is

$$f(T_\omega(x)) = T'_{\omega'}(f(x)), \quad x \in \mathcal{X}$$

which may not be desireable for the downstream task.

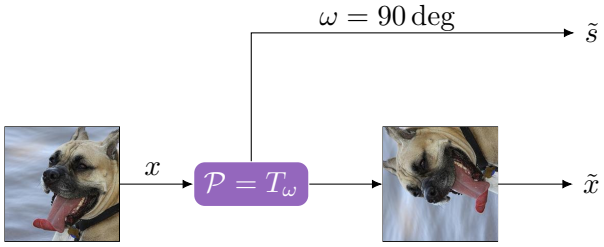

Figure 3.3: Pretext task pseudo-label process for transformation prediction.

3.2.3 Instance Discrimination

Instance discrimination is a pretext task in which each data point in an unlabelled dataset $\mathcal{D}_u = \{x_i\}_{i=1}^N$ is treated as its own class, i.e. the model is trained to discriminate between N distinct classes.

The classical way of employing instance discrimination is by assigning each data point a label as $\mathcal{P}(x_i) = \tilde{s}_i$ for $i = 1,2,\ldots,N$, where $\tilde{s}_i$ is a vector with zero-valued entries except at index i, where it is one. Assigning each data point x_i a one-hot vector admits instance discrimination using the categorical cross-entropy. This approach to instance discrimination has issues, such as scalability to large data sets, since the number of classes directly depends on the number of data points. In extreme cases this results in millions or even billions of classes. Another issue is a lack of intraclass variation, as every class is only represented by a single training example.

A different approach to instance discrimination is contrastive learning. Contrastive learning seeks to solve both of the issues related to classical instance discrimination: scaleability and poor intraclass variation. It achieves this by generating different views of each class through data augmentation and the task is then simply to predict if pairs of inputs belong to the same class. This implies that the target label is binary; 1 if the pair belongs to the same class and 0 otherwise.

The difficulty of contrastive learning lies in obtaining different views of the unlabelled data. Given a data point x_i, two different views are generated by a process Φ as $x^a \sim \Phi(x_i)$ and $x^+ \sim \Phi(x_i)$, where x^a is referred to as the anchor and x^+ as a positive sample. The anchor is then contrasted to its positive sample and to k negative samples,

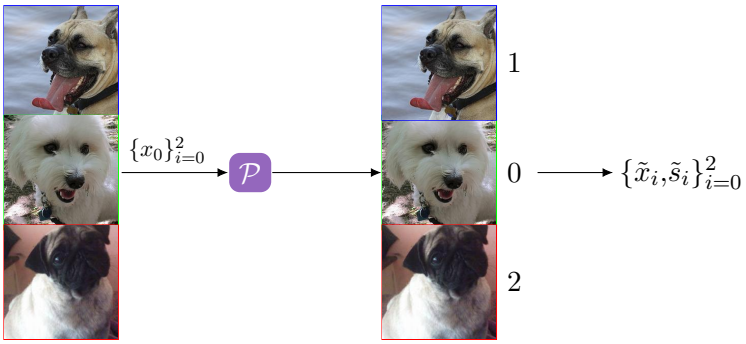

Figure 3.4: Pretext task pseudo-label process for instance discrimination.

obtained by applying the process to other data points than x_i as $x_j^- \sim \Phi(x_j)$ for $j \in \mathcal{I}_k = \{j_1,\ldots,j_k\}$, $k < N$ where $\mathcal{I}_k \cap \{i\} = \varnothing$. Thus, for each input x_i create the positive pair (x^a,x^+) and k negative pairs $\{(x^a,x_j^-)\}_{j\in\mathcal{I}_k}$ and concatenate these as $\tilde{x}_i = [(x^a,x^+),(x^a,x_{j_1}^-),\ldots,(x^a,x_{j_k}^-)]$ with pseudo label $\tilde{s}_i = [1,0,\ldots,0]$ containing k zeroes for the negative pairs. The pretext pseudo-label process repeats this for all inputs, i.e. $\mathcal{P}(\mathcal{D}_u) = \mathcal{D}_p = \{(\tilde{x}_i,\tilde{s}_i)|i = 1,\ldots,N\}$.

The model weights are determined as

$$\lambda^\star,\gamma^\star = \arg\min_{\lambda,\gamma} \sum_{(\tilde{x}_i,\tilde{s}_i)\in\mathcal{D}_p} -\mathbb{E}\left[\log\left(\frac{\sum_{l=1}^k \Psi(k_\gamma(h_\lambda(\tilde{x}_{i,l})))\tilde{s}_{i,l}}{\sum_{l=1}^k \Psi(k_\gamma(h_\lambda(\tilde{x}_{i,l})))}\right)\right]$$

where the operations of the trunk and head are applied separately to each element of the 2-tuple, and Ψ is some appropriate similarity measure [Ericsson et al., 2022].

Contrastive learning is most efficient when there is a large number of negative pairs, however increasing the number of negative pairs also increases the computational load during training. Note, that if too few negative pairs are used, then the model may not learn the details that distinguish instances. Furthermore, the most effective choice of k, the ratio between positive to negative samples, and how exactly to sample negative values, is not obvious [Ericsson et al., 2022].

The data augmentation and use of positive and negative pairs is what separates contrastive learning from classical instance discrimination. However, the data augmentation process needs to be designed carefully, as the process Φ can cause the model to be invariant to such processes. Contrastive learning further builds upon the base assumption of instance discrimination, that all instances represent semantically different objects. In many datasets this assumption does not hold since there will be multiple examples of similar objects, thereby creating false-negatives during training. Despite this, contrastive learning has proven to be an effective task.

3.2.4 Self-Distillation

Self-distillation relies solely on the model itself to generate pseudo-labels, and can be seen as a special case of knowledge-distillation. Knowledge-distillation is typically utilised to perform model compression, alleviating the problem of parameter space inflation when scaling up the size of DNNs. Knowledge-distillation works by employing two models, one serving as a teacher, and the other as a student. While the teacher model is trained only on

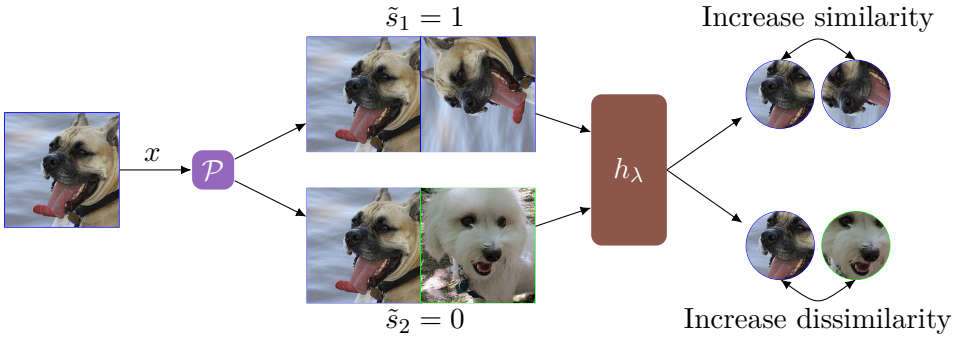

Figure 3.5: Pretext task for contrastive learning with pseudo-label process

ground truth labels, the student model is trained on both ground truth and the teacher's predictions [Pham et al., 2022]. Thereby facilitating the transfer of knowledge between the two models, which for model compression, necessitates that the student's parameterisation is smaller than the teacher's. Self-distillation however, refers to the case where both models utilise the exact same architecture.

In a self-supervised setting, the concept of self-distillation can be introduced using a single model to represent teacher and student. Let $h_{\lambda^{(0)}} : \mathcal{X} \to \mathcal{S}$ be a model parameterised by an initial state $\lambda^{(0)}$. Given the dataset $\mathcal{D}_u = \{x_i\}_{i=1}^N$, let the pseudo-label process at step $k \in \mathbb{N}$ be defined as

$$\mathcal{P}^{(k)}(\mathcal{D}_u) = \{A(x_i), h_{\lambda^{(k-1)}}(x_i)\}_{i=1}^N = \left\{\left(\tilde{x}_i, \tilde{s}_i^{(k)}\right)\right\}_{i=1}^N,$$

where A denotes some data augmentation. Self-distillation finds the parameterisation $\lambda^{(k)}$ greedily based on the predictions from the previous model parameterisation:

$$\lambda^{(k)} = \arg\min_{\lambda} \sum_{x_i \in \mathcal{D}_u} \mathcal{L}_p(h_\lambda(\tilde{x}_i)), \tilde{s}^{(k)}).$$

Suitable objectives $\mathcal{L}_p$ can range from the empirical mean squared error to the categorical cross entropy [Balestriero et al., 2023]. The single-model self-distillation concept is illustrated in Figure 3.6.

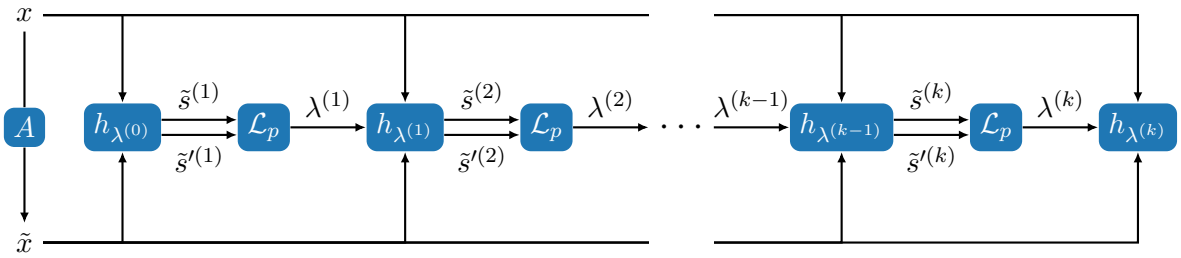

Figure 3.6: The single-model interpretation of the self-distillation process.

In a practical setting, self-distillation is typically implemented using the dual model teacher/student setup. The student and teacher are then shown different augmentations of the input to ensure that they are not presented with the same information, and the student subsequently attempts to predict the teacher's response. The augmentation strategy often

involves either a mask or transformation [Balestriero et al., 2023]. The teacher is then updated based on the parameterisation of the student, after which the process repeats. The dual model allows the teacher and student to update using separate schedules, which helps alleviate model collapse. This can be seen in models such as BYOL [Grill et al., 2020] and data2vec [Baevski et al., 2022] which utilise an exponential moving average of the student's parameterisation to update the teacher instead of a drop-in replacement.

4 | The Vision Transformer

Trying to extract information encoded in images using a computer poses an interesting problem. Though humans instinctively know how to decode this information, it is hard to formalise this process. Until recently, convolutional neural networks were the dominant approach. CNNs exhibit certain characteristics such as *locality*, *weight sharing*, and *translation invariance* for small perturbations [Goodfellow et al., 2016]:

- Locality ensures that neighbouring pixels are related as they all contribute to the same weighted sum.
- Weight sharing refers to the convolution reusing the same weights for different parts of the image, imparting the idea that certain patterns repeat.
- Translation invariance is achieved by combining the *translation equivariance* property of convolutions with *pooling* layers, allowing small translations of the input to correspond to the same output.
  - Translation equivariance indicates that a shift of the input will result in a corresponding shift to the output.
  - Pooling works by applying spatially local summary statistics, such as the sample mean or max operations.

These inductive biases are consistent with how vision is expected to work in cats [Goodfellow et al., 2016], LeCun et al. [1989], [Hubel and Wiesel, 1962].

In computer vision an alternative approach to CNNs is the *vision transformer* (ViT). ViT does not exhibit the same inductive biases as a CNN. However, contrary to a CNN, the ViT allows every layer of the DNN to incorporate information from everywhere in an image at the same time. This is only possible for a CNN with extreme kernel sizes. By interpreting an image as a sequence, the ViT can attend globally to an image in each layer by relying on the *transformer* architecture.

This chapter is mainly based on *Attention Is All You Need* by Vaswani et al. [2017], and *An Image is Worth 16x16 Words - Transformers for Image Recognition at Scale* by Dosovitskiy et al. [2021], with inspiration from the blog-posts of Bloem [2019], and Kazemnejad [2019].

4.1 Transformers

Traditional feed-forward NNs excel at extracting information and structure from data, however, they fail to consider sequential information. For a modality such as text, where

the relation between subsequent words in a sentence is extremely important, the lack of sequential information can be detrimental when trying to understand context, e.g. think of words with multiple meanings such as "set". This problem can be alleviated by using recurrent neural networks (RNN), which incorporate previous network states when processing new input, enabling the network to consider the temporal information of an input sequence. However, as there is a dependency in time, RNNs are inherently restricted to sequential computations, akin to iterative algorithms. Consequently, RNNs do not scale well to long sequences as computing long-term dependencies is slow and introduces the *vanishing gradient problem* [Salem, 2022, p. 60-61]. There exists gated RNNs, such as *Long Short Term Memory*, which mitigate these problems, though they are still fundamentally RNNs and thus sequential [Salem, 2022, p. 71-72].

In the seminal paper *Attention is All You Need* [Vaswani et al., 2017], the *transformer* model is proposed for sequence transduction tasks. The transformer abandons recursion in favour of a feed-forward structure, relying on attention mechanisms and positional encodings. This allows entire sequences to be processed simultaneously, while modelling sequence dependencies, decreasing computational complexity compared to RNNs, and providing state-of-the-art performance [OpenAI, 2023]. Typically, the transformer is pre-trained on a large corpus and subsequently fine-tuned to a specific task. The transformer architecture can be seen in Figure 4.1.

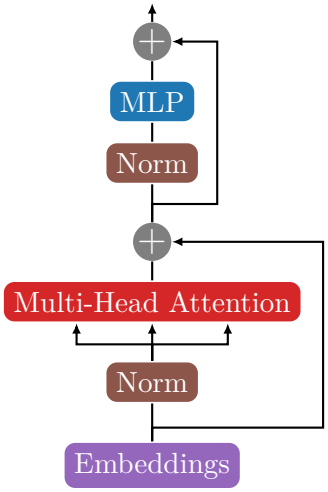

Figure 4.1: Transformer architecture.

4.2 Attention

To motivate the attention mechanism, consider a simple recommendation system. The essence of such a system is calculating similarities between vector-embeddings. These embeddings are simply vector representations of information, such as sounds, words, or images.

Let $\mathbf{x}$,$\mathbf{y} \in \mathbb{R}^n$ be embeddings representing the same modality. A similarity score can then be used to evaluate how closely related the two embeddings are perceived. The inner product of embeddings can be utilised as an intuitive similarity measure. With this approach, each entry in $\mathbf{x}$ and $\mathbf{y}$ contributes to a weighted sum giving the final score, thereby imparting a notion of agreement between the two embeddings.

Example 4.1 (Music Recommendation)
Let $\mathbf{x} \in \mathbb{R}^n$ be an embedding of a piece of music describing the prevalence of different genres. Similarly, let $\mathbf{y} \in \mathbb{R}^n$ encode the listeners affinity for the corresponding genres. Implicitly,

$$\mathbf{x} = \begin{bmatrix} \text{Contains Jazz} \\ \text{Contains Rock} \\ \text{Contains Country} \\ \vdots \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \text{Likes Jazz} \\ \text{Likes Rock} \\ \text{Likes Country} \\ \vdots \end{bmatrix},$$

where each entry corresponds to a numerical value. A recommendation can then be made based on the score $\mathbf{x}^\intercal\mathbf{y}$: The more the listener likes jazz, the higher that attribute is weighed in the score. Note, that negative values are allowed, and no normalisation is applied. This allows for a listeners' fondness for a particular genre to outweigh their dislike for another. Consequently, if a piece of music is represented as containing a negative amount of jazz, and the listener has a dislike for jazz, then the similarity is impacted positively.

Building from Example 4.1, it becomes impractical to manually assign traits when crafting embeddings for a recommendation system. However, from a mathematical perspective, there is no need to have tangible traits. Consequently, it is possible to simply learn the features given enough data, and the specific learning task defines how features are related. This is the underlying principle of attention.

The most basic attention mechanism calculates a sequence of attention vectors $\mathbf{a}_i \in \mathbb{R}^n$ from a sequence of input vectors $\mathbf{x}_i \in \mathbb{R}^n$. Let $X = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_t\end{bmatrix}^\intercal$ and $A = \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_t\end{bmatrix}^\intercal$ be input embeddings and attention vectors respectively, with the relation $A = WX$ where

$$W_{ij} = \sigma(\mathbf{w}_i)_j = \frac{e^{w_{ij}}}{\sum_{k=1}^{t} e^{w_{ik}}}, \quad w_{ij} = \mathbf{x}_i^\intercal \mathbf{x}_j, \quad i,j = 1,2,...,t.$$

The *softmax*[1] operation σ essentially creates a probability distribution. The weight matrix W can be interpreted as the required attention between elements of the sequence X, i.e. determining which elements of the sequence X are contextually related. An illustrative example of the basic attention calculation can be seen in Figure 4.2.

4.2.1 Scaled Dot-Product Attention

The attention mechanism is often separated into three states: *queries*, *keys*, and *values*.

Query: The element $\mathbf{x}_i$ of the input sequence X which is compared to every other element to compute the weights for the attention $\mathbf{a}_i$.

Key: Every element $\mathbf{x}_j$ of the input sequence X which is compared to the element $\mathbf{x}_i$ to compute the weights for the attention $\mathbf{a}_i$.

Value: Every element of the input sequence X which is used in a linear combination to form the attention $\mathbf{a}_i$.

[1] `https://pytorch.org/docs/stable/generated/torch.nn.Softmax.html`

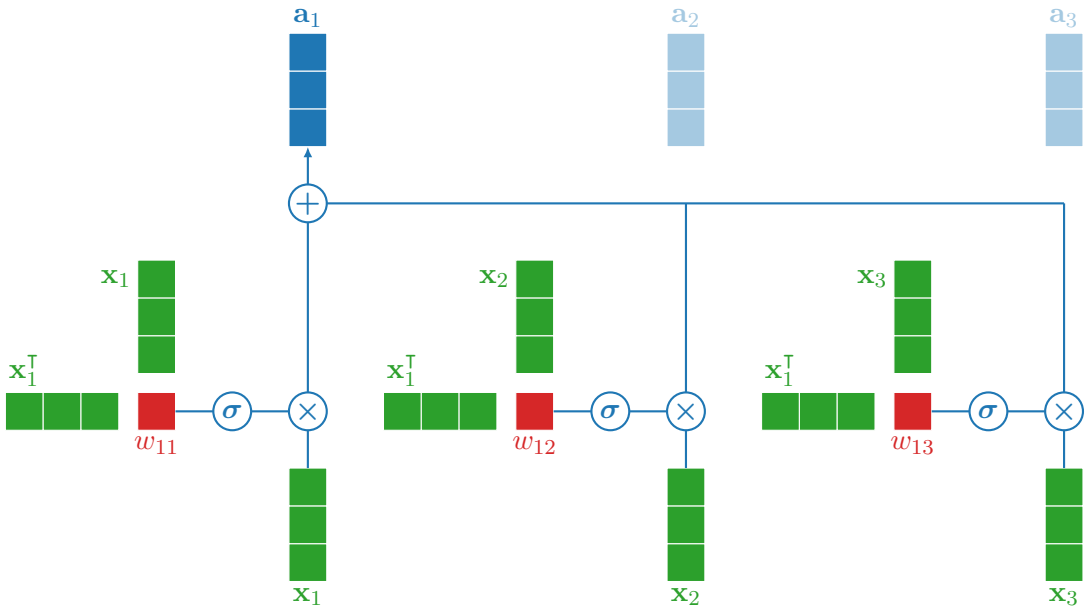

Figure 4.2: An illustration of the calculations involved in the basic attention mechanism. Note the slight abuse of the softmax operation σ.

This lends itself to the interpretation of performing a dictionary lookup. Each input (query) is compared to the index (keys) to see which entries (values) match. As described, the input $X \in \mathbb{R}^{t \times n}$ is pulling triple duty as query, key, and value. To increase the representation power, new vectors are derived using linear transformations to represent the three states. Let, $T^q, T^k \in \mathbb{R}^{n \times d}$, and $T^v \in \mathbb{R}^{n \times m}$ be the linear transformations utilised to generate the queries $\mathbf{q}_i \in \mathbb{R}^d$, keys $\mathbf{k}_i \in \mathbb{R}^d$, and values $\mathbf{v}_i \in \mathbb{R}^m$ respectively, as

$$\begin{aligned} Q &= XT^q, \quad Q \in \mathbb{R}^{t \times d} \\ K &= XT^k, \quad K \in \mathbb{R}^{t \times d} \\ V &= XT^v, \quad V \in \mathbb{R}^{t \times m}. \end{aligned}$$

This extension of the attention mechanism can be seen in Figure 4.3.

Note that, in general, as the dimensionality of the queries and keys increases, so does the magnitude of each $w_{ij} = \mathbf{q}_i^\intercal \mathbf{k}_j$. As the softmax operation produces very small gradients at the limits of its domain, a scaling factor corresponding to the dimensionality is applied before the softmax.

Definition 4.2 (Scaled Dot-Product Attention)
Let $Q \in \mathbb{R}^{t \times d}$, $K \in \mathbb{R}^{t \times d}$, and $V \in \mathbb{R}^{t \times m}$ have rows comprised of query, key, and value vectors respectively. The *scaled dot-product* attention mechanism $\mathcal{A}$ is then

$$\mathcal{A}(Q,K,V) = \sigma\left(\frac{QK^\intercal}{\sqrt{d}}\right)V.$$

[Vaswani et al., 2017]

In most practical cases, the dimension of the values m is equal to the embedding dimension n to allow residual connections.

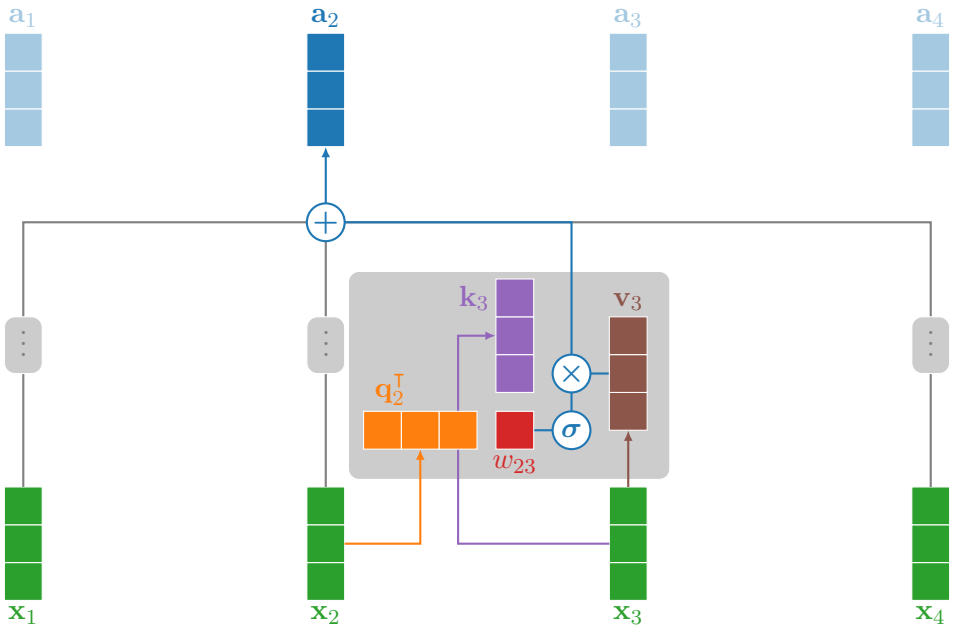

Figure 4.3: Illustration of the extended attention mechanism. Note the slight abuse of the softmax operation σ.

The computational complexity of the attention operation is $\mathcal{O}(nt^2)$, compared to $\mathcal{O}(n^2t)$ a recurrent layer [Vaswani et al., 2017]. Consequently, when the sequence length t is smaller than the embedding dimension n, which is often the case, the attention calculation is faster than an RNN.

4.2.2 Multi-Head Attention

It is possible to run multiple attention mechanisms in parallel, by creating additional queries, keys, and values using h learned projections. To keep the computational cost in line with a single attention operation, the attention vectors are mapped to a smaller dimension. After computing each of the h attention operations their outputs are concatenated and a final linear projection T^O is applied to return to the input dimension n.

Definition 4.3 (Multi-Head Attention)
Let $Q \in \mathbb{R}^{t\times d}$, $K \in \mathbb{R}^{t\times d}$, and $V \in \mathbb{R}^{t\times m}$ represent queries, keys, and values respectively. Let $i \in \{1,2,\ldots,h\}$, where h is a common factor of d and m, and let

$$T^{Q,i} \in \mathbb{R}^{d\times d/h}, \quad T^{K,i} \in \mathbb{R}^{d\times d/h}, \quad T^{V,i} \in \mathbb{R}^{m\times m/h}, \quad T^{O} \in \mathbb{R}^{m\times n}.$$

Define

$$A_i = \mathcal{A}(QT^{Q,i}, KT^{K,i}, VT^{V,i}), \quad A_i \in \mathbb{R}^{t\times m/h}.$$

The *multi-head* attention mechanism with h heads is then

$$\mathcal{A}_{\text{Multi}}(Q,K,V) = \begin{bmatrix} A_1 & A_2 & \cdots & A_h \end{bmatrix} T^O.$$

[Vaswani et al., 2017]

A block diagram depicting the different approaches to attention can be seen in Figure 4.4.

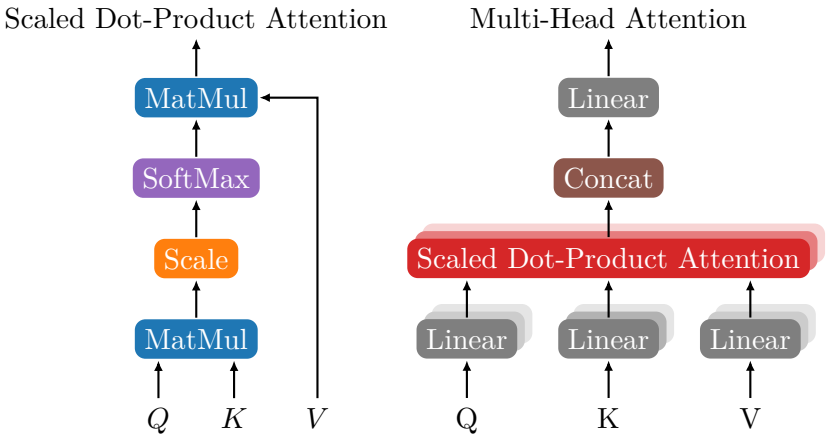

Figure 4.4: Block diagrams of single- and multi-headed attention.

Multi-head attention allows the transformer to attend to information from different subspace-representations at different positions jointly Vaswani et al. [2017]. This means that in a sentence like "Andreas buys potatoes from Mads" there is a possibility of one head that encodes for who receives the potatoes, and another that encodes for who sells the potatoes. Another example would be encoding grammatical gender (masculine, feminine, neutral) or quantity (singular, plural). This is contrary to a single scaled dot-product attention where all of this information is necessarily summed, decreasing its ability to discriminate.

An illustration of the attention behaviour can be seen in Figure 4.5

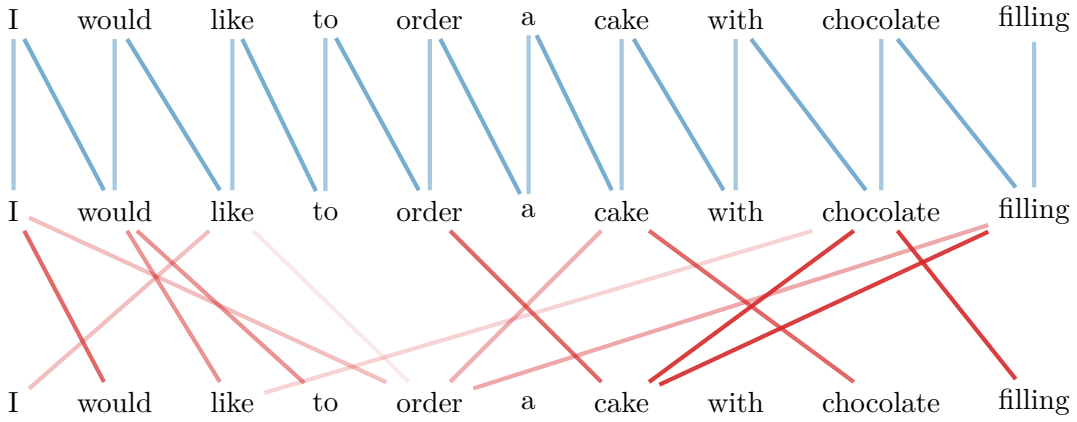

Figure 4.5: Constructed example of how different words in a sentence might attend to each other for two different attention heads. The colour opacity corresponds to a higher attention.

4.3 Positional Encoding

Though utilising multiple heads enables attending to different associations of a sequence, it does not incorporate any information regarding the position of the embedding in the sequence. The transformer handles this problem by introducing positional embeddings directly to the input embeddings, allowing the information to be cascaded through the network.

The naive approach to positional encoding is to represent the embedding $\mathbf{x}_i$ with a number τ_i which scales linearly with the index i. This approach does not generalise well however, as every sequence length then should be represented during training. Furthermore, for long sequences, this might have the problem of introducing energy to the model and potentially inflating the importance of input embeddings positioned later in a sequence. This can be alleviated by normalising the range such that $\tau \in [0,1]$. However, the normalisation also removes information pertaining to the length of the sequence, which results in an inconsistent interpretation of the positional embedding when presented with sequences of varying lengths. To circumvent all of these problems the transformer represents positions using sinusoids.

Definition 4.4 (Sinusoidal Positional Embedding)
Let $\mathbf{x}_i \in \mathbb{R}^n$ be the i'th vector in a sequence $X = \Big[\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_t\Big]$, and let

$$\omega_j = \rho^{-\frac{2j}{n}}, \quad \rho > 1, j \in \left\{1,2,\dots,\frac{n}{2}\right\}.$$

The sinusoidal positional embedding $\mathbf{p}_i$ corresponding to $\mathbf{x}_i$ is then

$$\mathbf{p}_i = \Big[\sin(i\omega_1) \quad \cos(i\omega_1) \quad \sin(i\omega_2) \quad \cos(i\omega_2) \quad \cdots \quad \sin\Big(i\omega_{n/2}\Big) \quad \cos\Big(i\omega_{n/2}\Big)\Big]^\intercal.$$

[Vaswani et al., 2017]

Definition 4.4 can alternatively be expressed in matrix form:

$$P_{ik} = \begin{cases} \sin\Big(i\omega_{(k+1)/2}\Big), & k \text{ is odd} \\ \cos\Big(i\omega_{k/2}\Big), & k \text{ is even,} \end{cases} \quad \text{for} \quad P = \Big[\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_t\Big]^\intercal. \tag{4.1}$$

Notice that $i \in \{1, 2, \dots, t\}$ denotes the position of an element in the sequence, and $k \in \{1, 2, \dots, n\}$ indexes the embedding dimension. Thus, for each entry k in the input embedding a different phase-frequency pair is assigned, creating a geometric progression of wavelengths from 2π to $2\pi\rho$. Examples of the positional embeddings created from Equation (4.1) can be seen in Figure 4.6, and Figure 4.7.

The encoding scheme presented in Equation (4.1) satisfies the following:

- Each sequence position has a unique embedding.
- Increasing the sequence length to $t + 1$ does not change the first t positional embeddings.

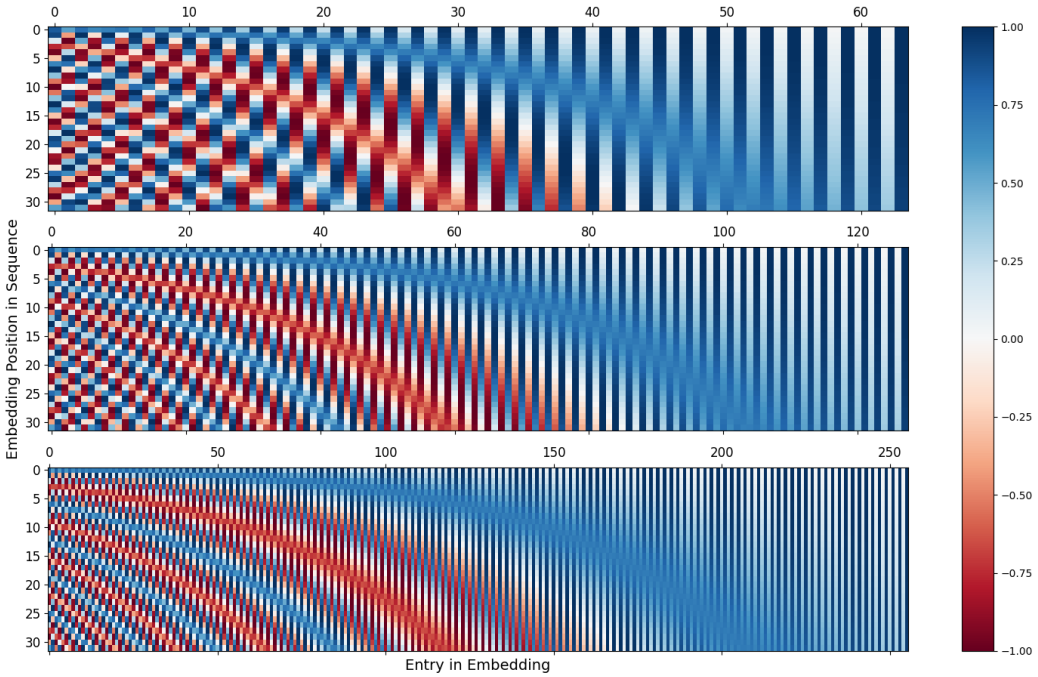

Figure 4.6: Sinusoidal positional embeddings with varying dimensionality.

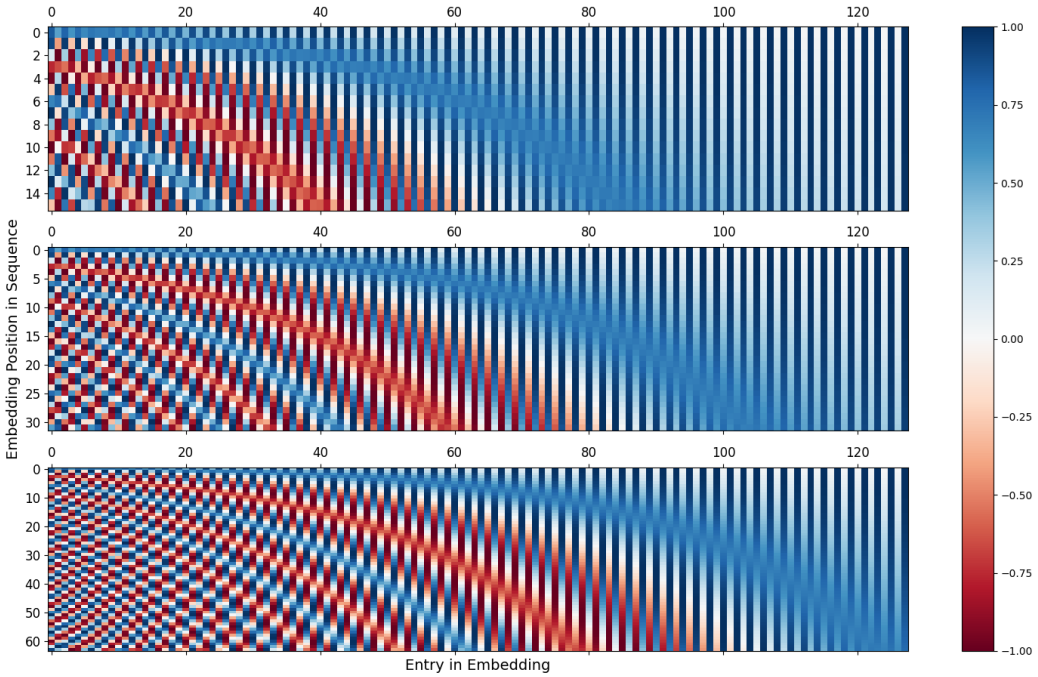

Figure 4.7: Sinusoidal positional embeddings with varying sequence lengths.

- The values are bounded between -1 and 1.

[Kazemnejad, 2019]

Furthermore, the sinusoidal embeddings allow the transformer to calculate attention based on relative positions since $\mathbf{p}_{i+k} = T\mathbf{p}_i$, i.e. every positional embedding can be represented by a linear projection of any other (Appendix C). The attention mechanism essentially assigns a score to such a projection, and thus the sinusoidal positional encoding behaves nicely in this context.

Lastly, as can be seen in Figure 4.8, the sinusoidal positional encoding also ensures that the distance between neighbouring embeddings $\mathbf{p}_i$ and $\mathbf{p}_{i+l}$ is symmetric and changes smoothly with l. This helps promote input embeddings placed closely together in a sequence,

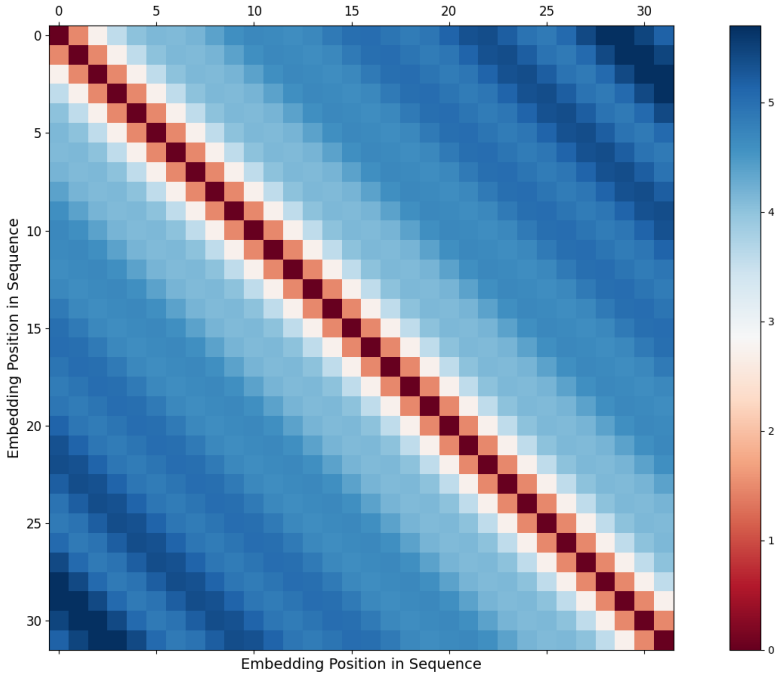

Figure 4.8: ℓ_2-distance between neighbouring sinusoidal positional embeddings.

while de-emphasising distant embeddings in the attention mechanism.

4.4 An Image as a Sequence

The transformer was originally envisioned to work on sequences of text, but a slight reframing of how images are represented, allows the transformer to operate on these as well. Building on the transformer architecture, the vision transformer (ViT) employs the same attention mechanism and a nearly identical structure to generate meaningful image embeddings.

Recall that digital images are often represented by a matrix encoding light intensities. If the image $x \in \mathbb{R}^{h \times w \times c}$ is flattened to form a vector $\mathbf{x} \in \mathbb{R}^{hwc}$, it is possible to consider it as a hwc-length sequence with scalar embeddings, or as a 1-length sequence with a hwc-dimensional embedding. However, these either do not scale well with the attention mechanism, or circumvent it entirely. Consequently, a balance must be struck by partitioning

the image into t patches, each considered an element in a sequence. In [Dosovitskiy et al., 2021] a single linear projection is applied to each flattened patch to create the input embeddings.

More explicitly, let h,w denote the height and width in pixels of an image, c the number of colour-channels, h_p, w_p denote the height and width in pixels of an image patch with h_p,w_p factors of h and w respectively. Given an image $x \in \mathbb{R}^{h\times w\times c}$, first separate it into $t = \frac{hw}{h_\mathrm{p} w_\mathrm{p}}$ patches $x_{\mathrm{p},i} \in \mathbb{R}^{h_\mathrm{p}\times w_\mathrm{p}\times c}$, then flatten to $\mathbf{x}_i \in \mathbb{R}^{h_\mathrm{p} w_\mathrm{p} c}$ for $i = 1,2,\ldots,t$, and concatenate these to form $X \in \mathbb{R}^{t\times h_\mathrm{p} w_\mathrm{p} c}$. A linear projection $L \in \mathbb{R}^{h_\mathrm{p} w_\mathrm{p} c\times n}$ maps the patches to the embedding dimension and a learnable `[class]` token is prepended. Finally, the positional embedding $P \in \mathbb{R}^{(t+1)\times n}$ is added to yield the final image embedding $\mathcal{E} \in \mathbb{R}^{(t+1)\times n}$:

$$\begin{aligned}\mathcal{E} &= \begin{bmatrix} \texttt{[class]} \\ \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_t\end{bmatrix}^\intercal L \end{bmatrix} + P \\ &= \begin{bmatrix} \texttt{[class]} \\ XL \end{bmatrix} + P.\end{aligned} \tag{4.2}$$

This embedding is then processed by the transformer. An overview of the vision transformer (ViT) architecture can be seen in Figure 4.9.

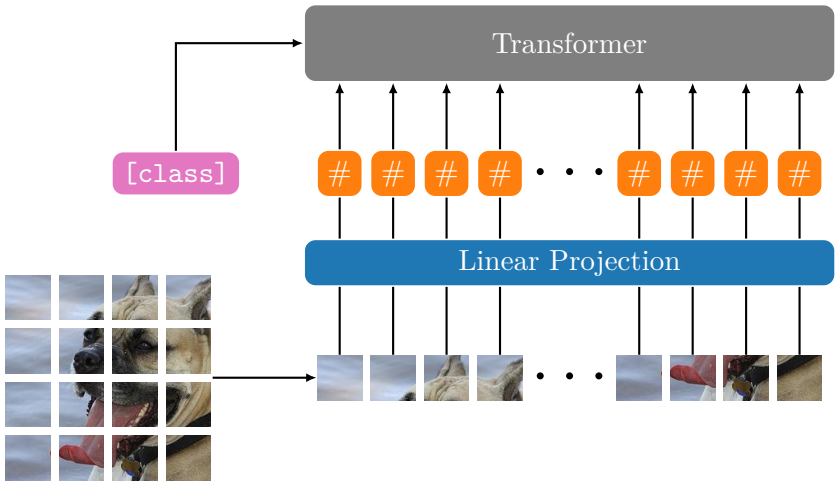

Figure 4.9: Vision transformer architecture. An image is segmented into patches, a linear transformation is applied to each patch, and a positional embedding is added to each of the input embeddings before entering the transformer. An additional learnable `[class]` token is included after the linear projection [Dosovitskiy et al., 2021].

When the attention mechanism is applied to image patches, the transformer is allowed to attend between image regions. This might result in patches containing eyes attending strongly with patches containing other facial features, such as a mouth or nose. Similarly, the background patches might attend weakly with the main subject of the image. An illustration of the basic attention mechanism presented in Section 4.2 applied to an image can be seen in Figure 4.10. Note that the basic attention mechanism applied to image patches will have a tendency to favour regions of high colour-intensity, as shown by the figure. Furthermore, when multiple attention mechanisms are applied simultaneously, a heatmap representing the receptive field of the transformer can be calculated using, e.g., *attention flow* [Abnar and Zuidema, 2020], however this is beyond the scope of this project.

Figure 4.10: The basic attention mechanism applied to a 224x224 image using 4x4 patches.

5 | Models

This chapter is designed to define and explain the models which are utilised to explore the intersection between algorithm unrolling and self-supervised learning. This chapter formulates models encompassing the two distinct approaches: Training a traditional unrolled network using self-supervision, and incorporating an unrolled network into a proven SSL model. However, first the reference unrolled algorithms are introduced.

5.1 LISTA

As mentioned in Example 2.2, the LISTA architecture was originally proposed by Gregor and LeCun [2010] to approximate solutions to the sparse coding problem in a fixed number of iterations.

The LISTA architecture is derived by unrolling ISTA, and as such, the network is specifically designed to solve the BPD problem

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|D\boldsymbol{\alpha} - \mathbf{x}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|, \tag{5.1}$$

where $\boldsymbol{\alpha}$ is the sparse code corresponding to the dictionary D and input vector $\mathbf{x}$. ISTA solves the BPD problem by iterating

$$\boldsymbol{\alpha}^{(k)} = S_\lambda\Big(\frac{1}{L}D^\intercal\mathbf{x} + (I - \frac{1}{L}D^\intercal D)\boldsymbol{\alpha}^{(k-1)}\Big), \quad \boldsymbol{\alpha}^{(0)} = \mathbf{0}, \tag{5.2}$$

where S_λ is the entry-wise soft-thresholding operator, described in Equation (1.6), with parameter λ. Using the unrolling process, each iteration is recast as

$$\boldsymbol{\alpha}^{(k)} = S_{\lambda^{(k)}}\Big(W_d^{(k)}\mathbf{x} + W_e^{(k)}\boldsymbol{\alpha}^{(k-1)}\Big),$$

where the predetermined dictionary D and shrinkage parameter λ are replaced by the learnable parameters $\{W_d^{(k)}\}_{k=1}^T$, $\{W_e^{(k)}\}_{k=1}^T$, and $\{\lambda^{(k)}\}_{k=1}^T$. Thus, contrary to ISTA, each iteration is allowed a unique parameterisation, increasing the representation power but removing the domain knowledge associated with the dictionary construction. However, it is possible to show that the optimal parameters tend to the weight coupling scheme asymptotically [Chen et al., 2018]. Notice that the code $\boldsymbol{\alpha}$ is essentially a high-dimensional (sparse) embedding of the input $\mathbf{x}$, which is relevant for Section 5.3.2.

Given a training-set of sparse codes $\{\boldsymbol{\alpha}_i\}_{i=1}^N$, $\boldsymbol{\alpha}_i \in \mathbb{R}^n$, where N denotes the batch size, the loss function $\mathcal{L}$ is the empirical mean squared error:

$$\mathcal{L} = \frac{1}{nN}\sum_{i=1}^{N}\Big\|\boldsymbol{\alpha}_i^{(T)} - \boldsymbol{\alpha}_i\Big\|_2^2.$$

Consequently, LISTA is trained supervised and requires knowledge of the sparse codes beforehand. For images, sparse codes can be found using algorithms such as OMP [Elad, 2010, p.37] or ISTA, provided a suitable dictionary. Alternatively, both the sparse codes and dictionary can be found greedily by applying K-SVD [Elad, 2010, p.234]. The LISTA architecture can be seen in Figure 2.3.

Directly utilising the architecture presented in [Gregor and LeCun, 2010] for image super-resolution, would require knowledge of both the sampling process Φ and the domain Ψ, as presented in Section 1.2. With the BPD formulation in Equation (5.1), this information is contained in D, meaning that the upscaling can only be performed after the LISTA layers by multiplying Ψ and $\boldsymbol{\alpha}$. Consequently, each layer of the network is not utilised directly for the super-resolution task, and extensive prior knowledge, might be required to accurately reconstruct the high-resolution image. However, by reframing the BPD problem a more suitable architecture can be extrapolated.

5.2 ISTA-Net: LISTA for Image Super-Resolution

To create a baseline for the experimental results, a LISTA inspired architecture called ISTA-net [Zhang and Ghanem, 2018] will be utilised.

The ISTA-Net architecture is motivated with a slight reformulation of the basis-pursuit denoising problem in Equation (5.1):

$$\min_{\mathbf{x}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\tilde{\Psi}\mathbf{x}\|_1. \tag{5.3}$$

Here, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ correspond to flattened high- and low-resolution images respectively, while $\Phi \in \mathbb{R}^{m\times n}$ is a downsampling matrix, and $\tilde{\Psi}\mathbf{x} \in \mathbb{R}^N$ sparse codes. The reformulation thus assumes $\Phi = D\tilde{\Psi}$ and $\boldsymbol{\alpha} = \tilde{\Psi}\mathbf{x}$. Note that, Equation (5.3), contrary to (5.1), explicitly states that the sparse codes $\boldsymbol{\alpha}$ can be synthesised from $\mathbf{x}$. Furthermore, this formulation also directly solves the super-resolution task, rather than relying on a separate reconstruction dictionary, creating a more efficient use of the unrolling mechanism for solving the super-resolution task.

Because of $\tilde{\Psi}$, the solution to Equation (5.3) is not immediately analog to Equation (5.2). Thus, consider the proximal step from Equation (1.5) in the context of Equation (5.3):

$$\begin{aligned}\mathbf{r}^{(k)} &= \rho\Phi^\intercal\mathbf{y} + (I - \rho\Phi^\intercal\Phi)\mathbf{x}^{(k-1)} \\ \mathbf{x}^{(k)} &= \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{x} - \mathbf{r}^{(k)}\right\|_2^2 + \lambda\left\|\tilde{\Psi}\mathbf{x}\right\|_1,\end{aligned} \tag{5.4}$$

yielding the super-resolution image $\mathbf{x}^{(k)} \in \mathbb{R}^n$. It can be shown that, if $\tilde{\Psi}$ is an orthogonal matrix, then $\mathbf{x}^{(k)} = \tilde{\Psi}^\intercal S_\lambda(\tilde{\Psi}\mathbf{r}^{(k)})$ (Appendix B.1). The assumption of a linear bijection might however be too restrictive. Motivated by the high representation power and the universal approximation property of CNNs, ISTA-Net replaces the linear map $\tilde{\Psi}$ with the trainable non-linear maps $\{\mathcal{F}_{\tau^{(k)}}\}_{k=1}^T$, where

$$\mathcal{F}_{\tau^{(k)}} = \text{Conv2D} \circ \text{ReLU} \circ \text{Conv2D},$$

with learnable parameters $\{\tau^{(k)}\}_{k=1}^T$. The Conv2D operation is performed according to the PyTorch implementation[1], while ensuring sufficient padding as to keep the filtered input the same length.

[1] `https://pytorch.org/docs/1.13/generated/torch.nn.Conv2d.html`

When utilising a non-linear operator, the solution to Equation (5.4) is harder to find. Consequently, the ISTA-Net architecture assumes that $\mathbf{x}$ and $\mathcal{F}_{\tau^{(k)}}(\mathbf{x})$ are normally distributed with mean $\mathbf{r}^{(k)}$ and $\mathcal{F}_{\tau^{(k)}}(\mathbf{r}^{(k)})$ respectively. Under these assumptions the approximation

$$\left\|\mathcal{F}_{\tau^{(k)}}(\mathbf{x}) - \mathcal{F}_{\tau^{(k)}}(\mathbf{r}^{(k)})\right\|_2^2 \approx \kappa^{(k)}\left\|\mathbf{x} - \mathbf{r}^{(k)}\right\|_2^2 \tag{5.5}$$

holds. Examination of the validity of this assumption is outside the scope of this project, but can be found in the original ISTA-Net paper by Zhang and Ghanem [2018]. Incorporating the approximation (5.5) into Equation (5.4) yields

$$\mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathcal{F}_{\tau^{(k)}}(\mathbf{x}) - \mathcal{F}_{\tau^{(k)}}(\mathbf{r}^{(k)})\right\|_2^2 + \underbrace{\lambda^{(k)}\kappa^{(k)}}_{\theta^{(k)}} \left\|\mathcal{F}_{\tau^{(k)}}(\mathbf{x})\right\|_1,$$

and thus

$$\mathcal{F}_{\tau^{(k)}}(\mathbf{x}^{(k)}) = S_{\theta^{(k)}}(\mathcal{F}_{\tau^{(k)}}(\mathbf{r}^{(k)})).$$

By also introducing $\tilde{\mathcal{F}}_{\tilde{\tau}^{(k)}}$ and enforcing $\tilde{\mathcal{F}}_{\tilde{\tau}^{(k)}} \circ \mathcal{F}_{\tau^{(k)}} = \mathrm{Id}$, the update step becomes

$$\mathbf{x}^{(k)} = \tilde{F}_{\tilde{\tau}^{(k)}}(S_{\theta^{(k)}}(\mathcal{F}_{\tau^{(k)}}(\mathbf{r}^{k}))).$$

See Appendix B.2 for the derivation. To facilitate that $\tilde{\mathcal{F}}_{\tilde{\tau}^{(k)}}$ is the inverse of $\mathcal{F}_{\tau^{(k)}}$, it is designed with operational symmetry to $\mathcal{F}_{\tau^{(k)}}$ such that,

$$\tilde{\mathcal{F}}_{\tilde{\tau}^{(k)}} = \mathrm{Conv2D} \circ \mathrm{ReLU} \circ \mathrm{Conv2D}$$

with no bias. In other words, the number of kernels expands and contracts symmetrically around the soft-shrink operation such that $\mathcal{F}_{\tau^{(k)}}$ produces multiple filtered versions of the input signal, and $\tilde{\mathcal{F}}_{\tilde{\tau}^{(k)}}$ recombines these to a single output signal. The learnable parameters for ISTA-Net are therefore $\{\theta^{(k)}\}_{k=1}^{T}$, $\{\tau^{(k)}\}_{k=1}^{T}$, and $\{\rho^{(k)}\}_{i=1}^{T}$.

Given the high-resolution training images $\{\mathbf{x}_i\}_{i=1}^{B}$, where B denotes the batch size, the loss function $\mathcal{L}$ comprises two terms:

$$\mathcal{L}_{\mathrm{D}} = \frac{1}{nB}\sum_{i=1}^{B}\left\|\mathbf{x}_i^{(T)} - \mathbf{x}_i\right\|_2^2 \tag{5.6}$$

$$\mathcal{L}_{\mathrm{IM}} = \frac{1}{nB}\sum_{i=1}^{B}\sum_{k=1}^{T}\left\|\tilde{\mathcal{F}}_{\tilde{\tau}}^{(k)}(\mathcal{F}_{\tau}^{(k)}(\mathbf{x}_i^{(k)})) - \mathbf{x}_i^{(k)}\right\|_2^2 \tag{5.7}$$

Equation (5.6) is designed to lower the discrepancy between the original high-resolution image $\mathbf{x}_i$ and the estimated super-resolution image $\mathbf{x}_i^{(T)}$, while Equation (5.7) incentivises the inverse morphism constraint $\tilde{\mathcal{F}}_{\tilde{\tau}^{(k)}} \circ \mathcal{F}_{\tau^{(k)}} = \mathrm{Id}$. Thus,

$$\mathcal{L} = \mathcal{L}_{\mathrm{D}} + \gamma\mathcal{L}_{\mathrm{IM}},$$

where $\gamma \in \mathbb{R}$ is a regularisation parameter.

The initialisation $\mathbf{x}^{(0)}$ is found as $\mathbf{x}^{(0)} = Q_{\mathrm{init}}\mathbf{y}$ where

$$Q_{\mathrm{init}} = \arg\min_{Q} \|QY - X\|_F^2 = XY^{\intercal}(YY^{\intercal})^{-1}, \quad Q_{\mathrm{init}} \in \mathbb{R}^{n\times m}, \tag{5.8}$$

with

$$X = \begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_q\end{bmatrix}, \quad Y = \begin{bmatrix}\Phi\mathbf{x}_1 & \cdots & \Phi\mathbf{x}_q\end{bmatrix},$$

given q high-resolution training images $\{\mathbf{x}_i\}_{i=1}^{q}$, and downsampling matrix Φ.

The ISTA-Net architecture can be seen in Figure 5.1.

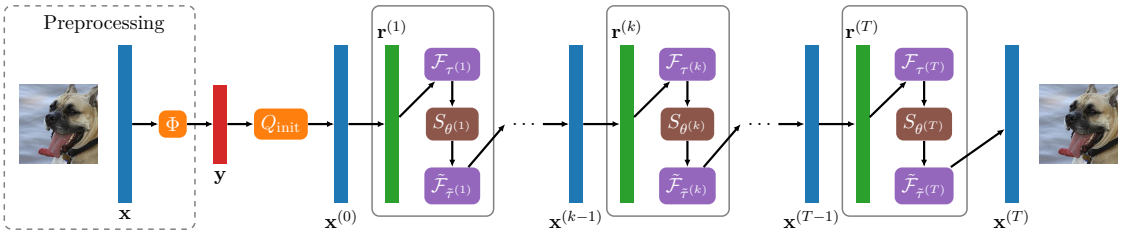

Figure 5.1: ISTA-Net for image super-resolution. The high-resolution image $\mathbf{x}$ is downsampled to the low-resolution image $\mathbf{y}$. The low-resolution image $\mathbf{y}$ is then projected using Q_{init} for the initial guess $\mathbf{x}^{(0)}$. The initialised super-resolution image $\mathbf{x}^{(0)}$ is then introduced to the first layer, producing a refined image $\mathbf{x}^{(1)}$. This is repeated for each of the T layers producing the final super-resolution image $\mathbf{x}^{(T)}$.

5.3 Masked Autoencoder

With the rise of self-supervised learning some traditionally unsupervised frameworks have been revised or reformulated to include self-supervision. One such revision, is to utilise masked prediction (Section 3.2.1) and task an autoencoder with inferring the held-out information. Such autoencoders are called masked autoencoders (MAE). MAEs have been successfully deployed for computer vision problems by using vision transformers (Chapter 4), an asymmetric design of the encoder and decoder, and a masking ratio of around 75% [He et al., 2021]. However, the concept of a masked autoencoder is not limited to a specific architecture and may work for others as well.

5.3.1 Incorporating Vision Transformers

The MAE partitions an input image into patches and then randomly masks them according to a given masking ratio. The encoder of the MAE only encodes non-masked patches, which, when the masking ratio is high, yields a significant computational gain. The missing elements of the encoded sequence are introduced to the decoder as shared and learned mask tokens, to which positional embeddings are added. These tokens represent the missing patches and together with the positional encoding, allows the decoder to better interpret their spatial relationship. Thus, the decoder processes both the mask tokens and the encoded patches. Typically, the decoder is designed to be significantly smaller than the encoder. By processing the mask tokens only in the decoder, the encoder only sees real patches. This translates well to downstream tasks, where it is unnatural to see mask tokens. Furthermore, if the mask tokens are included in the encoder the performance deteriorates, likely due to seeing inputs comprised of both unnatural mask tokens as well as real patches He et al. [2021]. The MAE model is illustrated in Figure 5.2.

The masking procedure $\mathcal{M}_r$ in MAE takes an image embedding $\mathcal{E} \in \mathbb{R}^{(t+1)\times n}$ (Equation (4.2)), randomly selects $q = \lfloor rt \rfloor$ out of the t patch embeddings with $r \in [0,1)$, and generates a binary vector $\mathbf{y} \in \mathbb{B}^t$ which indicates whether a patch is masked. The masking procedure is described by

$$\mathcal{M}_r(\mathcal{E}) = (\mathcal{E}^M \in \mathbb{R}^{(q+1)\times n}, \mathbf{y} \in \mathbb{B}^t).$$

Subsequently, K_1 transformer blocks are used to encode $\mathcal{E}^M$, and after normalising, the decoder linearly projects the code before inserting $t - q$ learnable mask tokens. Then $\mathbf{y}$ is used to organise the positions of the coded patches and K_2 transformer blocks are used to

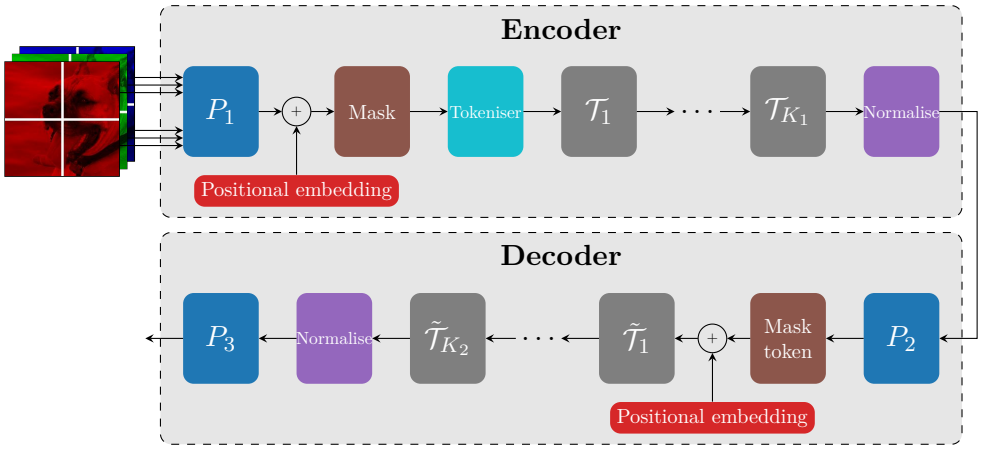

Figure 5.2: Masked autoencoder model description.

decode the encoded sequence. Finally a linear projection maps the signal to the prediction $X' \in \mathbb{R}^{t \times h_p w_p c}$.

The entire model is defined as $k_\gamma \circ h_\lambda$ where $h_\lambda : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^{(q+1) \times n}$ comprises the encoder and $k_\gamma : \mathbb{R}^{(q+1) \times n} \to \mathbb{R}^{t \times h_p w_p c}$ the decoder. Thus, for an image $x \in \mathbb{R}^{h \times w \times c}$ the model yields a prediction $X' \in \mathbb{R}^{t \times h_p w_p c}$ as

$$X' = (k_\gamma \circ h_\lambda)(x).$$

The MAE model predicts the full image given a partial view, but since the difficulty of the task arises from predicting the missing patches of the image, the loss function needs to reflect this. The loss function is the empirical mean squared error restricted to the pixels associated with the missing patches. Thus for an unlabelled dataset of images $\{x_i \in \mathbb{R}^{h \times w \times c}\}_{i=1}^N$ the objective is

$$\mathcal{L}_p = \frac{1}{Nq} \sum_{i=1}^{N} \left(\left((k_\gamma \circ h_\lambda)(x_i) - X_i \right)^\intercal \mathbf{y}_i \right)^2,$$

where $X_i \in \mathbb{R}^{t \times h_p w_p c}$ denotes the partitioned x_i and $\mathbf{y}_i$ is a binary vector that selects patches.

The MAE is used as the pretext task to train the model $k_\gamma \circ h_\lambda$. After pre-training, the weights $\lambda^\star$,$\gamma^\star$ are transferred to the downstream model $g_{\gamma^\star} \circ h_{\lambda^\star}$, after which the parameterisation of the final projection P_3 of g is increased by a factor m in order to perform super-resolution. The downstream task is then to perform super-resolution on low resolution images by utilising the training loss

$$\mathcal{L}_d = \frac{1}{M} \sum_{i=1}^{M} \Big((g_\varphi \circ h_\lambda)(D_m(x_i)) - X_i \Big)^2$$

for the dataset $\{x_i \in \mathbb{R}^{mh \times mw \times c}\}_{i=1}^M$ of high resolution images and downscaling operator D_m that downscales with factor m.

5.3.2 Incorporating LISTA and Vision Transformers

Traditional ViT uses a linear projection to create embeddings (Equation (4.2)), that are then encoded using self-attention and MLPs. In this section, the linear projection is replaced

by a non-linear projection F, specifically a T layer LISTA architecture (Section 5.1). This model will be referred to as ISTA-MAE. For an image $x \in \mathbb{R}^{h \times w \times c}$ partitioned into patches $X \in \mathbb{R}^{t \times h_p w_p c}$, the embedding $\mathcal{E} \in \mathbb{R}^{(t+1) \times n}$ is given as

$$\mathcal{E} = \begin{bmatrix} \texttt{[class]} \\ F(X) \end{bmatrix} + P.$$

Here F sequentially applies T LISTA layers to X with a zero-vector as initialisation for the sparse code. Since the embeddings should be sparse they are penalised using the ℓ_1 norm. Thus, the loss function is

$$\mathcal{L}_p = \frac{1}{N} \sum_{i=1}^{N} \left(\left((k_\gamma \circ h_\lambda)(x_i) - X_i \right)^\intercal \mathbf{y}_i \right)^2 + \lambda \| F(X) \|_1,$$

with weighing parameter $\lambda = 0.0001$. The number of parameters for F is $L(cnhw + n^2)$ compared to nhw parameters from a traditional ViT embedding. In relation to Figure 5.2 the proposed approach replaces the projection P_1 with F.

In the proposed approach the attention mechanism attends to sparse representations of the patches. Sparse codes are hypothesised to be an effective choice of embedding, since they dictate which dictionary atoms are used to synthesise the inputs. By extension the sparse codes must contain inherent semantic similarities between inputs. For example, consider the case where the dictionary is a Fourier basis. In this feature space, images which contain many of the same frequencies will tend to cluster together. For the basic attention mechanism, sparse codes with disjoint support would result in zero attention. That is, images which do not have any semantic relation with respect to the dictionary, will not attend to each other. This relation is generally not sustained for the scaled dot-product attention. However, utilising linear transformations on embeddings in a space where the embeddings are already separable, is regarded to be an acceptable starting point for the network. As such, the sparse embeddings are well suited for the attention mechanisms that ViTs utilise.

5.4 data2vec

In an attempt to create a general framework for SSL, Baevski et al. [2022] propose data2vec and show successful deployment for speech, language, and computer vision. data2vec incorporates self-distillation (Section 3.2.4) along with a transformer architecture (Chapter 4) to construct a unified learning framework. Thus, data2vec presents a model relying on predicting latent representations from masked inputs.

As data2vec utilises self-distillation, the pseudo-label generation process comprises both data augmentation and feature extraction. Consequently, two identical neural networks are constructed, employing a masking strategy where the teacher is presented the full input while the information presented to the student network is masked. Similarly to MAE (Section 5.3), data2vec utilises the ViT, interpreting an image as a sequence of image patches, and masks by replacing portions of the image sequence with a learned mask embedding. Prediction is also only performed on the masked portion, and thus the difficulty of the task depends on the masking ratio. Typical data2vec masking ratios for computer vision are around 60% [Baevski et al., 2022].

Formally, let h_{λ_s} and h_{λ_t} denote the student and teacher networks. The parameters of the teacher is updated as an exponentially moving average of the student's parameters to

avoid model collapse. The weight update rule for the teacher network is thus

$$\lambda_t \leftarrow \tau\lambda_t + (1-\tau)\lambda_s,$$

where τ determines how much the parameters are updated. A schedule is used for τ, such that it linearly increases from an initial value of τ_0 to a target value τ_e over j updates, after which it is fixed. The student's parameters are updated using backpropagation and stochastic gradient descent.

For a teacher and student network with T layers, let $\tilde{s}^{(t)}$ denote the latent representation of $h_{\lambda_t}(x_i)$ at the t'th layer. The task of the student network h_{λ_s} is then to predict

$$\tilde{s} = \frac{1}{K}\sum_{k=1}^{K}\tilde{s}^{(T-k)},$$

the average latent representation of the last K layers. Thus, given the unlabelled dataset $\mathcal{D}_u = \{x_i\}_{i=1}^N$, the pseudo-label process is given as $\mathcal{P}(\mathcal{D}_u) = \{(\mathcal{M}_r(x_i),\tilde{s}_i)\}_{i=1}^N$.

An illustration of the data2vec process can be seen in Figure 5.3.

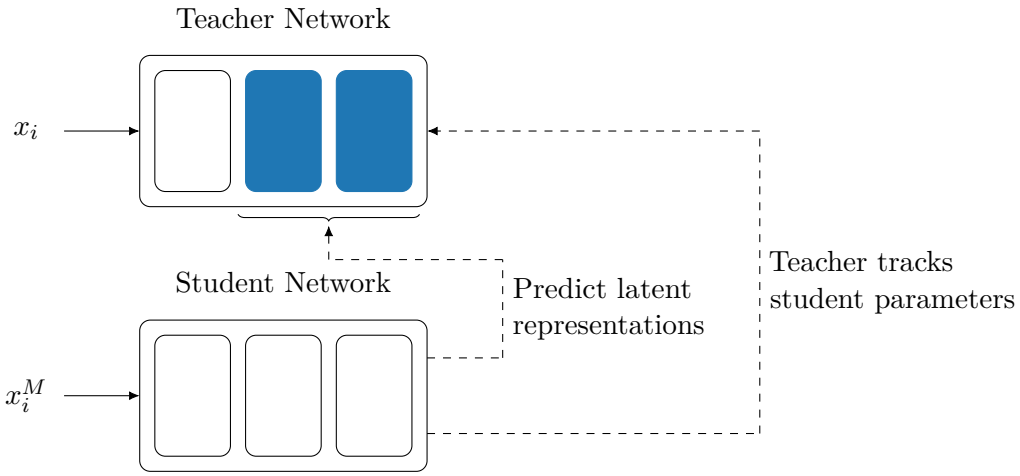

Figure 5.3: Illustration of the data2vec framework.

The objective of the pretext task is minimising the average smooth-L1 loss restricted to the masked patches by $\mathbf{y}_i$:

$$\mathcal{L}_p = \frac{1}{Nq}\sum_{i=1}^{N}\mathcal{L}_{\ell_1}(h_\lambda(x_i),\tilde{s}_i)^\intercal\mathbf{y}_i,$$

where q is the number of masked patches and the smooth-L1 loss is given by

$$\mathcal{L}_{\ell_1}(h_\lambda(x),\tilde{s}) = \begin{cases}\frac{1}{2}(\tilde{s}-h_\lambda(x))^2\beta^{-1}, & |\tilde{s}-h_\lambda(x)| \le \beta,\\ |\tilde{s}-h_\lambda(x)| - \frac{1}{2}\beta, & \text{otherwise.}\end{cases}$$

β is a hyperparameter that determines the transition from a squared to an L1 loss.

5.4.1 Incorporating ISTA-Net

Though data2vec was originally formulated for computer vision using ViT architectures, the authors specify that "[...] alternative architectures may be equally applicable" [Baevski et al., 2022]. Thus, both teacher and student networks can be created as identical ISTA-Net models (Section 5.2). As ISTA-Net does not consider the input as a sequence, the masking procedure needs to change. Consequently, for the image $x_i \in \mathbb{R}^{h \times w}$, the masking procedure is

$$\mathcal{M}_r(x_i) = (x_i \odot M, \mathbf{y} \in \mathbb{B}^{hw}),$$

where $M \in \mathbb{B}^{h \times w}$ is a matrix of ones where a random submatrix of size $h_m \times w_m$ is replaced by zeros. The shape of the submatrix is constrained by the masking ratio r. This corresponds to setting a rectangular region of each image to black. The masking procedure is applied to the initialisation $x^{(0)}$. An example of the masking procedure can be found in Figure 5.4. Thus, the ISTA-Net student is presented with both masked and non-masked

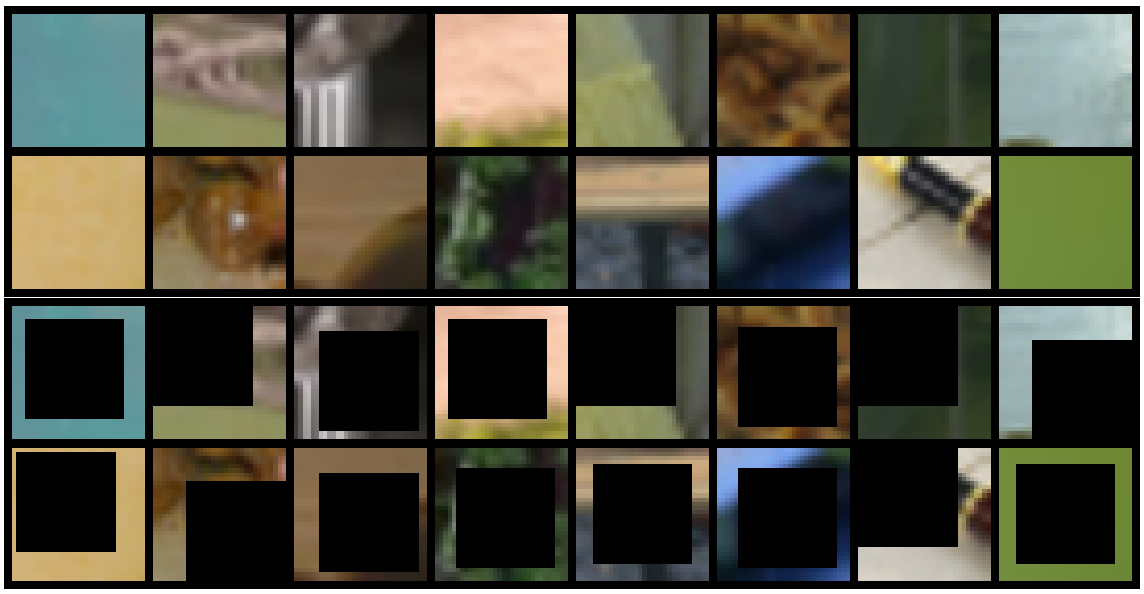

Figure 5.4: Random image crops from the ImageNet dataset. Top: Images presented to the teacher. Bottom: Images presented to the student.

information. Pre-training is then performed using a combination of the smooth-L1 loss and the inverse morphism constraint (Equation (5.7)):

$$\mathcal{L} = \mathcal{L}_p + \gamma \mathcal{L}_{\text{IM}}.$$

For the downstream task, only the pre-trained student model $h_{\lambda^\star}$ is transferred, and both the teacher model and the masking procedure are discarded. The student is then fine-tuned according to the original formulation of ISTA-Net. This corresponds to training the ISTA-Net model h_λ as presented in Section 5.2, but with the initial parameterisation $\lambda = \lambda^\star$. This model will be referred to as ista2vec.

6 | Experiments

With a basis in the models presented in Chapter 5, this chapter outlines model parameters, hyperparameters, evaluation metrics, and datasets for reproducibility. The goal is to design experiments that test the performance of the models in different scenarios and allow for fair comparisons between them. The code for training the models outlined in this chapter can be found at `https://github.com/LarsenAndreas/SSL_ISTA`

6.1 General Setup

The models are trained on three distinct datasets: the *102 Category Flower Dataset* [Nilsback and Zisserman, 2008], the *Oxford-IIIT Pet Dataset* [Parkhi et al., 2012], and the *2017 ILSVRC dataset* [Russakovsky et al., 2015]. To aid with readability, these datasets will be referred to as *Flowers*, *Pets*, and *ImageNet* respectively. ImageNet is utilised exclusively for self-supervised pre-training and is, as such, not needed for evaluation. Conversely, both Flowers and Pets have the last 20% reserved for evaluation. Note that, the datasets are pruned to only include RGB images.

Every model is fine-tuned/trained on either the first 80% or 1% of Pets or Flowers. Subsequently, the models are denoted

1. M_{F} trained on $\approx 80\%$ (6551) of Flowers.
2. M_{P} trained on $\approx 80\%$ (5896) of Pets.
3. $M_{\mathrm{F,R}}$ trained on $\approx 1\%$ (81) of Flowers.
4. $M_{\mathrm{P,R}}$ trained on $\approx 1\%$ (73) of Pets.

A superscript will be used to denote the specific architecture.

The evaluation data is prepared by extracting a 128×128 center crop from the remaining 20% of the respective dataset, and subsequently downscaled to obtain 64×64 low-resolution images. The low-resolution images are then preprocessed and upscaled, according to the model specifications, to infer the super-resolution image. All experiments compare the super-resolution image with the ground truth image by their Peak Signal-to-Noise Ratio (PSNR):

$$\mathrm{PSNR}(u, v) = 10 \log_{10} \left(\frac{I_{\max}^2}{\mathrm{MSE}(u, v)} \right),$$

where $I_{\max} = 1$ for images with values in the range [0,1], and thus reduces to

$$\mathrm{PSNR}(u, v) = -10 \log_{10}(\mathrm{MSE}(u,v)).$$

The experiments are divided into the three following categories:

- **Baseline:** M_{F}, and M_{P} are evaluated on Flowers and Pets respectively.
- **Restriction:** $M_{\mathrm{F,R}}$ and $M_{\mathrm{P,R}}$ are evaluated on Flowers and Pets respectively.
- **Generality:** M_{P}, $M_{\mathrm{P,R}}$ are evaluated on Flowers, and M_{F}, $M_{\mathrm{F,R}}$ are evaluated on Pets.

The baseline experiments are designed to provide a reference for the model performance when presented with in-distribution data. The restriction experiments are designed to provide a similar reference, while mimicking a scenario with data-scarcity, as SSL has been shown to increase the performance in such a scenario. The generality experiments should provide insight into how well these model generalises to out-of-distribution data.

6.2 ISTA-Net

To establish a general baseline, four ISTA-Net models (Section 5.2) are trained to upscale 16×16 image patches to 32×32. As the images from both datasets posses both a horisontal and vertical resolution much larger than required ($\gg 32$), each minibatch uniformly samples 32×32 crops of the full images as targets. Thus, for every minibatch, the images are preprocessed as follows:

1. A random patch $x_{\mathrm{p}} \in \mathbb{R}^{32\times32\times3}$ is sampled.
2. The patch is flattened to the target/label $x \in \mathbb{R}^{1024\times3}$.
3. The low-resolution input patch $y \in \mathbb{R}^{256\times3}$ is calculated as $y = \Phi x$.

The downsampling matrix Φ is designed such that it corresponds to dividing a 32×32 image patch into a regular grid of 2×2 squares, calculating the average pixel value of each square, and using the averages as the pixels in the 16×16 low-resolution image patch.

6.2.1 Model Parameters

The initilisation matrix Q_{Init}, is calculated for M_{F}, M_{P}, $M_{\mathrm{F,R}}$, and $M_{\mathrm{P,R}}$ by uniformly sampling the color channels of 6551, 5896, 81, and 73 high-resolution image patches respectively to their training datasets. All models are trained using an Adam optimiser [Kingma and Ba, 2017] with default PyTorch initialisation[1].

The ISTA-Net models use 9 layers and take an input image of dimension 16×16 and upscale with a factor of 2 to obtain 32×32 high-dimensional images. The non-linear operation $\mathcal{F}$ uses 32 and 32^2 3×3 kernels for the first and second convolution respectively in each layer. To have operational symmetry, $\tilde{\mathcal{F}}$ uses 32^2 followed by 32 3×3 kernels. The parameters are initialised according to the Xavier-normal scheme [Glorot and Bengio, 2010]. The models have in total 181,458 trainable parameters.

The hyperparameters are identical across the models and described in Table 6.1.

[1] `https://pytorch.org/docs/1.13/generated/torch.optim.Adam.html`

Epochs	Batch Size	Learning Rate	γ	Decay Rates	Stability Constant
500	64	10^{-4}	0.01	$(0.9, 0.999)$	10^{-8}

Table 6.1: Hyperparameters used for training the baseline ISTA-Net models.

6.3 MAE

To establish a self-supervised baseline, four MAE models (Section 5.3.1) are trained to upscale 64×64 images to 128×128. The training of all four models comprises two stages: 1) The models are identically pre-trained using the MAE framework (Section 5.3), and 2) fine-tuning is conducted following the general setup described in Section 6.1.

6.3.1 Model Parameters

The parameters are identical across the models and can be found in Table 6.2. The MLP ratio for all transformers in both the encoder and decoder is 4. The models have a total of 5,317,632 trainable parameters.

Miscellaneous		
Image size	Patch size	Masking ratio
64	8	0.75
ViT Encoder		
Depth	Number of heads	Embedding size
6	8	256
ViT Decoder		
Depth	Number of heads	Embedding size
2	8	128

Table 6.2: Model parameters of the MAE models.

6.3.2 Pre-Training

The pre-training is performed on ImageNet. Every image x in ImageNet is pre-processed by sequential application of the following transformations:

1. A random patch $x_\mathrm{p} \in \mathbb{R}^{h \times w \times 3}$ of the original images is extracted. The height h and width w are randomly selected between 20% and 100% of the original images' vertical and horisontal resolution.
2. The patch is resized to $x_\mathrm{p} \in \mathbb{R}^{64 \times 64 \times 3}$ using bicubic interpolation.
3. The patch is flipped horisontally with a probability of 50%.
4. The patch is normalised with respect to the pixel mean and standard deviation of ImageNet.

The MAE models are trained using an AdamW [Loshchilov and Hutter, 2019] optimiser with weight-decay and learning rate warm-up. All fully connected layers of the model are

initialised following a Xavier-uniform [Glorot and Bengio, 2010] distribution, and the class and mask token are initialised following a zero-mean Gaussian distribution with standard deviation 0.02.

The hyperparameters used for pretext training are described in Table 6.3.

Epochs	Batch Size	LR	Weight Decay	Warm-up Epochs	Warm-up Target
18	512	0.0001	0.05	10	0.002

Table 6.3: Hyperparameters used for pre-training the MAE models.

6.3.3 Fine-Tuning

After pre-training, the model weights are transferred to a downstream model and the final fully connected layer is initialised following $\mathcal{U}(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}})$. Finally, the MAE model is fine-tuned to perform super-resolution on data comprising pairs of high and low resolution images, where the model predicts the high resolution image from the low resolution image as described in Section 6.1.

The hyperparameters used for fine-tuning of MAE are described in Table 6.4.

Epochs	Batch Size	LR	Weight Decay	Warm-up Epochs	Warm-up Target
1000	16	0.001	0.05	50	0.01

Table 6.4: Hyperparameters used for downstream training of MAE with ViT architecture.

6.4 ISTA-MAE

This section describes the training procedure for the ISTA-MAE model described in Section 5.3.2 which utilises both ViT and LISTA. The training is split into two subsequent phases: 1) self-supervised pretext training, and 2) supervised fine-tuning, utilising the setup described in Section 6.1. The training setup is identical to the one used for the MAE models, but the architecture differs in the initial encoding.

6.4.1 Model Parameters

The ISTA-MAE models utilise 5 LISTA layers to obtain the encoder embedding, and similarly to the baseline MAE model it utilises an MLP ratio of 4. The remaining model parameters are identical to the baseline MAE model seen in Table 6.2. The models have a total of 5,778,437 trainable parameters.

6.4.2 Pre-Training

The hyperparameters used for self-supervised pre-training are identical to the ones used for the MAE models, and these can be seen in Table 6.3.

6.4.3 Fine-Tuning

The hyperparameters used for fine-tuning the ISTA-MAE models are identical to the ones utilised for the MAE models, and these can be seen in Table 6.4.

6.5 ista2vec

This section describes the training procedure for the ista2vec model described in Section 5.4, which utilises the data2vec training paradigm for ISTA-Net. Four ISTA-Net models (Section 5.2) are trained to predict 32×32 representations of 16×16 image patches. The training is split into two subsequent phases: 1) self-supervised pre-training, based on the data2vec framework, and 2) supervised fine-tuning, utilising the setup described in Section 6.1.

6.5.1 Model Parameters

The parameters of all four models are identical to the ISTA-Net models (Section 6.2.1, and they are initialised following a Xavier-normal distribution [Glorot and Bengio, 2010]. Since the model size is the same as the ISTA-Net models, these models each contain 181,458 trainable parameters.

6.5.2 Pre-Training

The pre-training phase adopts the data2vec student/teacher setup: Two ISTA-Net architectures are initialised with identical parameterisations, and trained as described in Section 5.4 on ImageNet with preprocesing identical to Section 6.2. The pre-training hyperparameters can be seen in Table 6.5. A cosine schedule is used to anneal the learning rate. Note that,

Epochs	Batch Size	Learning Rate	K	β
42	512	10^{-5}	1	2
EMA Schedule	**Masking Ratio**	**Decay Rates**	**Stability Constant**	γ
(0.9995, 0.9998)	0.60	(0.9, 0.999)	10^{-8}	0.01

Table 6.5: Hyperparameters used for pre-training ISTA-Net in the data2vec framework.

only the final representation from the teacher is predicted, i.e. $K = 1$. This is chosen based on the original paper [Baevski et al., 2022] presenting a relatively small performance increase for $K > 1$ on images, compared to text and sound. Furthermore, because of unrolling, each sequential ISTA-Net layer generates a more refined representation [Zhang and Ghanem, 2018]. Consequently, averaging these would correspond to a worse representation than simply using the final one.

To help increase the generality of the representations, the initialisation matrix Q_{Init} (Equation (5.8)), is replaced with TorchVisions `Resize` function[2] with bilinear interpolation. Thus, Q_{Init} is not derived from ImageNet samples, hopefully increasing the effectiveness of fine-tuning.

[2] `https://pytorch.org/vision/0.15/generated/torchvision.transforms.Resize.html`

6.5.3 Fine-Tuning

After pre-training, the teacher models and input masking are dropped, and the student models are fine-tuned in a supervised setting. Training is performed identically to Section 6.2, thus replacing the `Resize` function with the initialisation matrix Q_{Init}. The fine-tuning hyperparameters are shown in Table 6.1.

7 | Results

This chapter presents the findings of the experiments described in Chapter 6. The observations in this chapter are based purely on the PSNR which is not necessarily indicative of the perceptual quality of the reconstructions. Note that all the figures only depict performance on the evaluation set, i.e. the last 20% of the respective datasets.

The performance of each model will be illustrated as bar plots of the average PSNR over the entire evaluation dataset, as this allows for easy overall comparisons between models on both Flowers and Pets. To contextualise the observations from the bar plots, the individual model performance on a per-image basis is examined. This study is visualised in terms of *win rate* plots, each depicting the absolute difference in PSNR between two models. The plots always present the win rate for either ista2vec or ISTA-MAE against their respective reference models. Thus, the ISTA-Net based models are never compared with the MAE based models. The win rate plots serve as a high-resolution depiction of the otherwise aggregated results of Figure 7.5-7.16, as averaging might grant an unfair advantage to models which perform extremely well on very specific imagery.

Examples of recovered super-resolution images from every model can be seen in Figure 7.1-7.4.

Figure 7.1: Low-resolution, high-resolution, and super-resolution images for the baseline ISTA-Net and ista2vec models.

7.1 Baseline

This section will present the results from the *baseline* experiments. This includes the bar plot in Figure 7.5 as well as the win rate Figure 7.6-7.9.

Figure 7.2: Low-resolution, high-resolution, and super-resolution images for the baseline MAE and ISTA-MAE models.

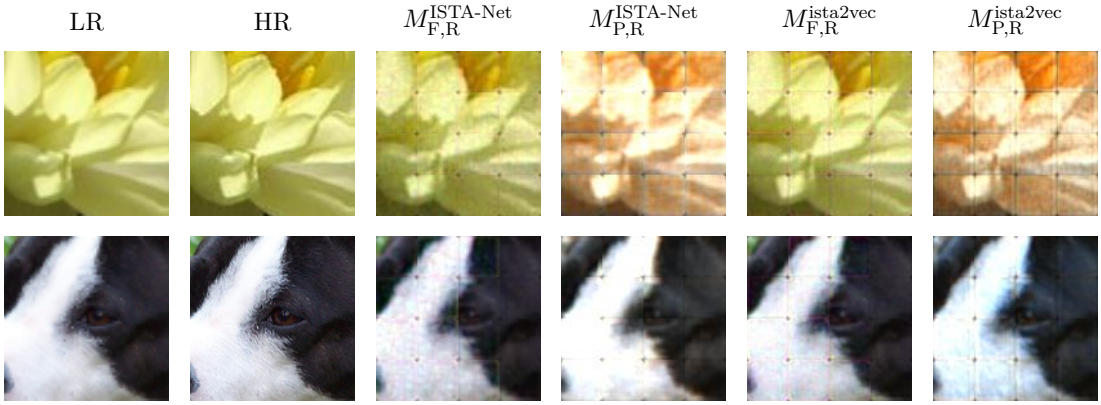

Figure 7.3: Low-resolution, high-resolution, and super-resolution images for the restriction ISTA-Net and ista2vec models.

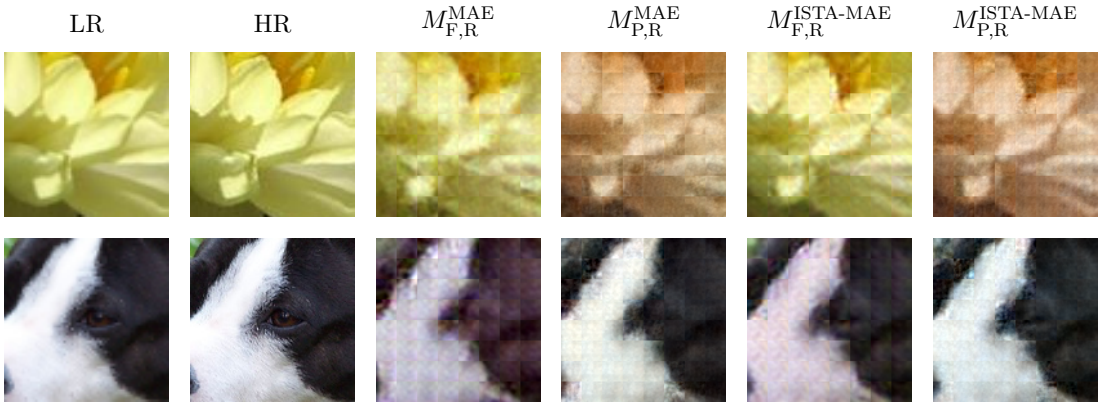

Figure 7.4: Low-resolution, high-resolution, and super-resolution images for the restriction MAE and ISTA-MAE models.

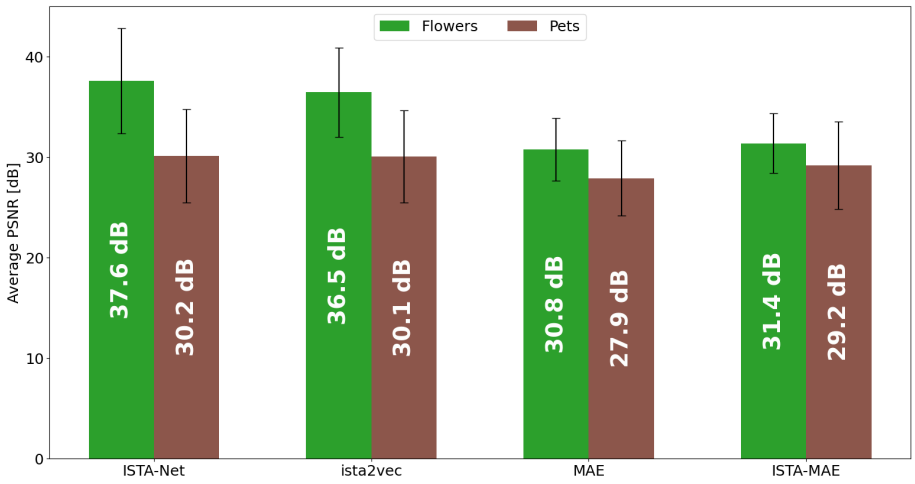

Figure 7.5: Baseline experiment. Average PSNR for the M_{F} and M_{P} models. The black line denotes the standard deviation.

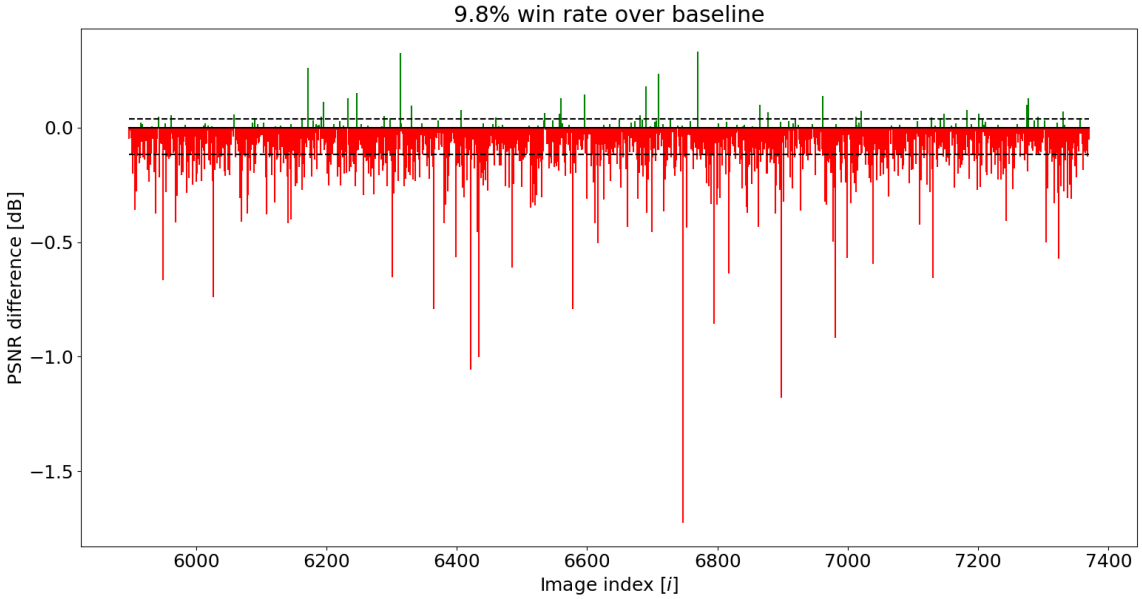

Figure 7.6: ista2vec win rate against ISTA-Net, both trained on 80% of Pets and tested on the remaining 20% of Pets. Green indicates that ista2vec wins and red indicates that ISTA-Net wins.

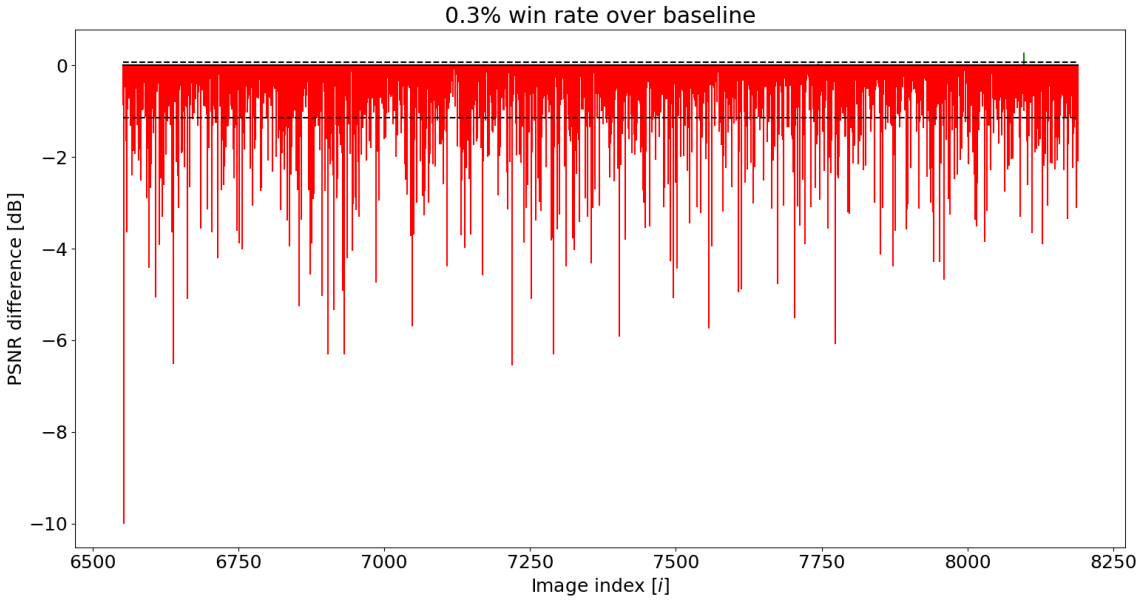

Figure 7.7: ista2vec win rate against ISTA-Net, both trained on 80% of Flowers and tested on the remaining 20% of Flowers. Green indicates that ista2vec wins and red indicates that ISTA-Net wins.

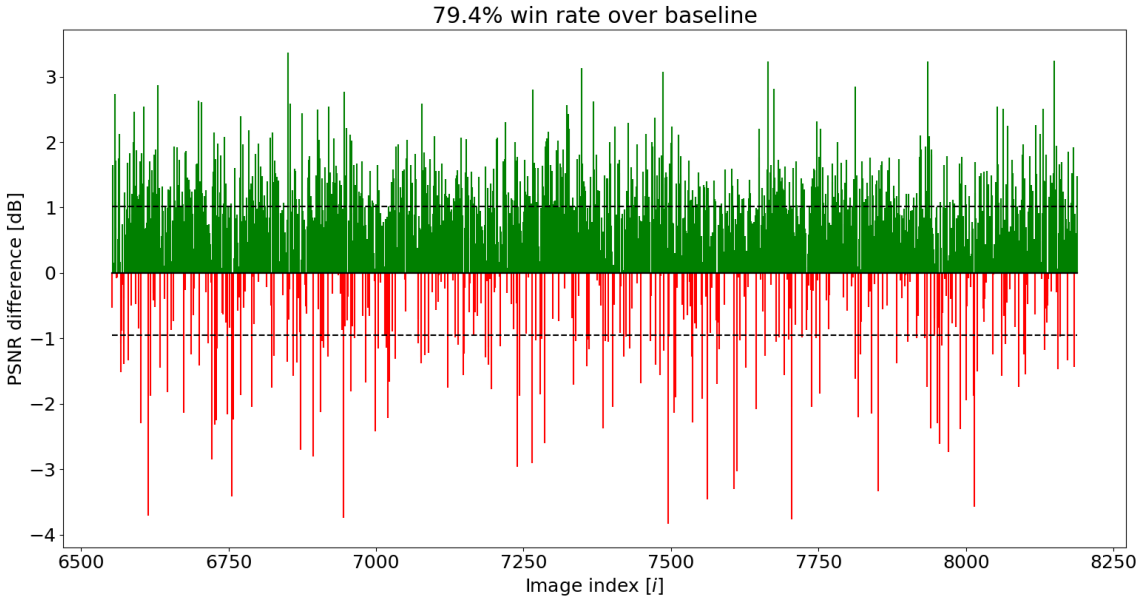

Figure 7.8: ISTA-MAE win rate against regular MAE, both trained on 80% of Flowers and tested on the remaining 20% of Flowers. Green indicates that ISTA-MAE wins and red indicates that regular MAE wins.

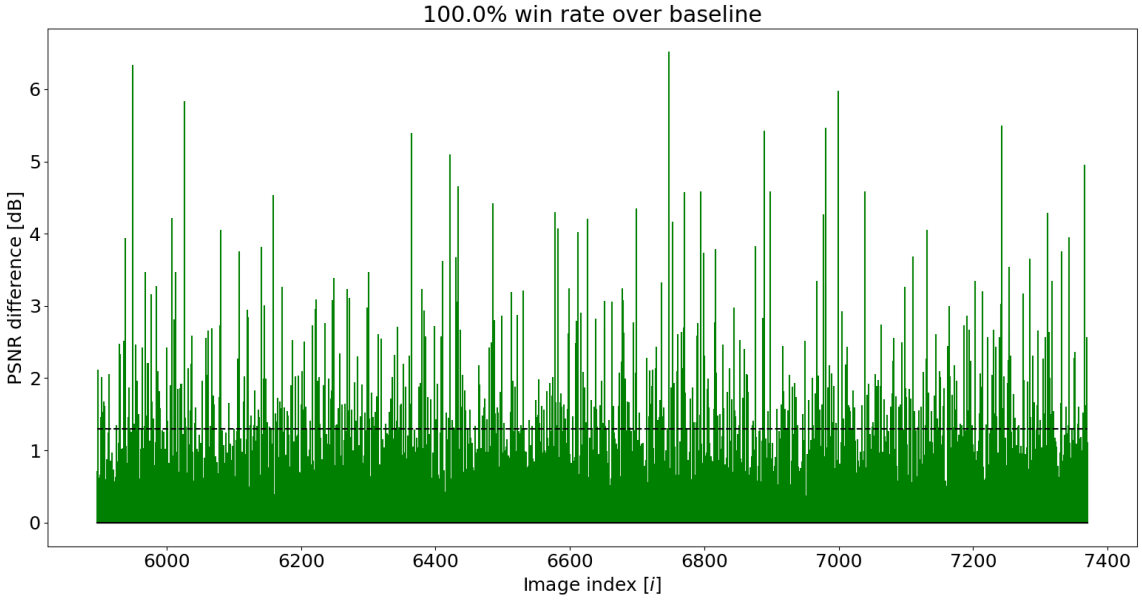

Figure 7.9: ISTA-MAE win rate against regular MAE, both trained on 80% of Pets and tested on the remaining 20% of Pets. Green indicates that ISTA-MAE wins and red indicates that regular MAE wins.

The performance of $M_{\mathrm{F}}^{\text{ISTA-Net}}$ compared with $M_{\mathrm{F}}^{\text{ista2vec}}$, and $M_{\mathrm{P}}^{\text{ISTA-Net}}$ compared with $M_{\mathrm{P}}^{\text{ista2vec}}$ is indistinguishable when looking solely at Figure 7.5, as the average PSNR is easily within the standard deviation. However, Figure 7.6 and Figure 7.7 seem to indicate that ISTA-Net is the better performing model, having a difference as high as 10 dB and average loss margin of 1 dB when comparing $M_{\mathrm{F}}^{\text{ISTA-Net}}$ to $M_{\mathrm{F}}^{\text{ista2vec}}$.

Based on Figure 7.5 the performance of $M_{\mathrm{F}}^{\text{MAE}}$ and $M_{\mathrm{F}}^{\text{ISTA-MAE}}$ are similar. The figure further shows that $M_{\mathrm{P}}^{\text{MAE}}$ and $M_{\mathrm{P}}^{\text{ISTA-MAE}}$ perform similarly. Noticably, Figure 7.8 and 7.9 show that ISTA-MAE has a higher PSNR on 79.4% and 100% of Flowers and Pets test images respectively.

Lastly, Figure 7.5 also shows $M_{\mathrm{F}}^{\text{ISTA-Net}}, M_{\mathrm{F}}^{\text{ista2vec}}$ outperforming $M_{\mathrm{F}}^{\text{MAE}}, M_{\mathrm{F}}^{\text{ISTA-MAE}}$ with an average PSNR approximately 5-7 dB higher. Note that this statement does not hold for M_{P}.

7.2 Restriction

This section will present the results from the *restriction* experiments. This includes the bar plot seen in Figure 7.10 as well as the win rate plots in Figure 7.11-7.14.

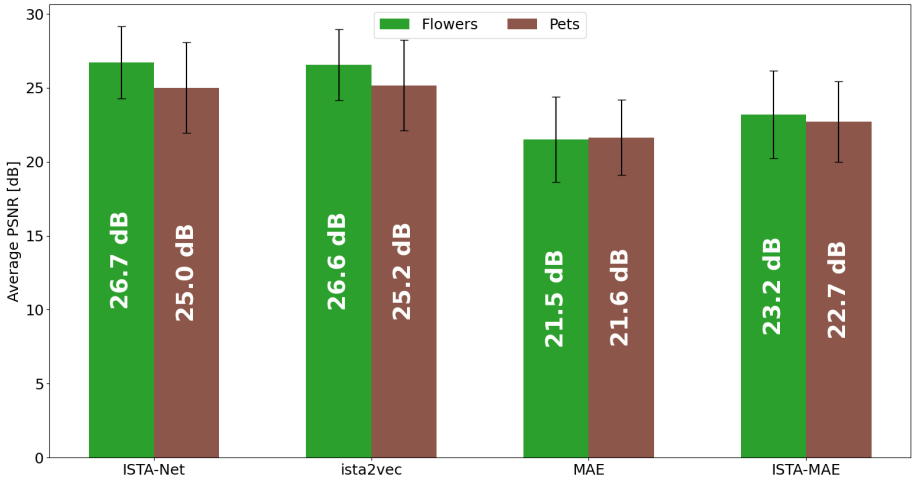

Figure 7.10: Restriction experiment. Average PSNR for the models $M_{\mathrm{F,R}}$ and $M_{\mathrm{P,R}}$. The black line denotes the standard deviation.

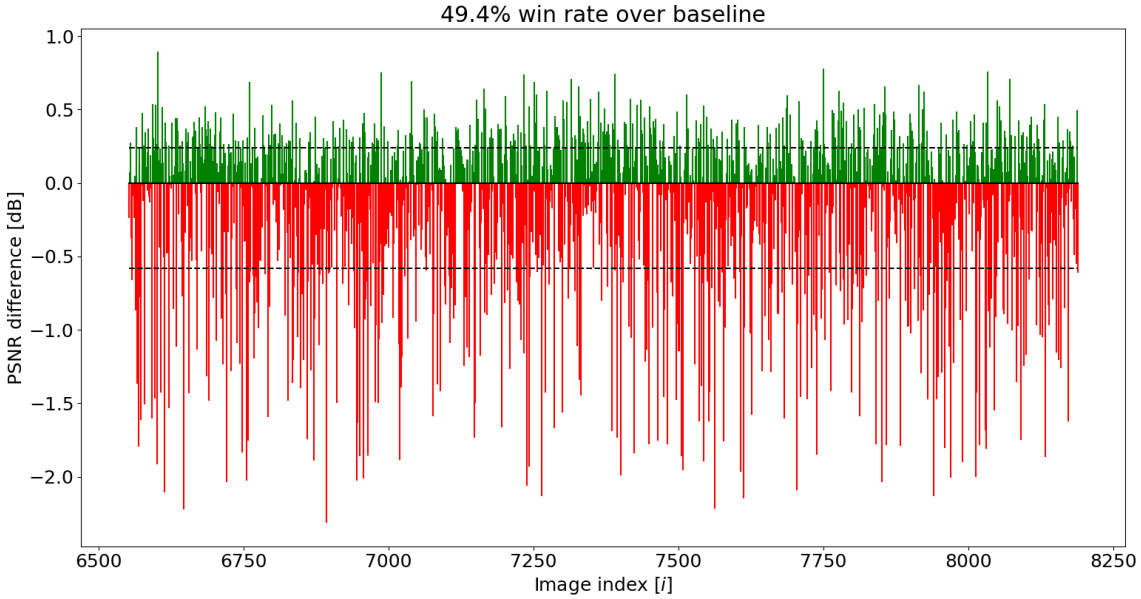

Figure 7.11: ista2vec win rate against ISTA-Net, both trained on 1% of Flowers and tested on the remaining 20% of Flowers. Green indicates that ista2vec wins and red indicates that ISTA-Net wins.

The performance of $M_{\mathrm{F,R}}^{\text{ISTA-Net}}$ compared with $M_{\mathrm{F,R}}^{\text{ista2vec}}$, and $M_{\mathrm{P,R}}^{\text{ISTA-Net}}$ compared with $M_{\mathrm{P,R}}^{\text{ista2vec}}$ is indistinguishable when looking solely at Figure 7.10. However, by inspecting Figure 7.11, a slight edge in performance on a per-image basis can be attributed to $M_{\mathrm{F,R}}^{\text{ISTA-Net}}$, as it has a higher average PSNR lead of 0.5 dB compared to 0.25 dB. $M_{\mathrm{P,R}}^{\text{ista2vec}}$ has a higher win rate of 77.3% than $M_{\mathrm{P,R}}^{\text{ISTA-Net}}$, as can be seen in Figure 7.12.

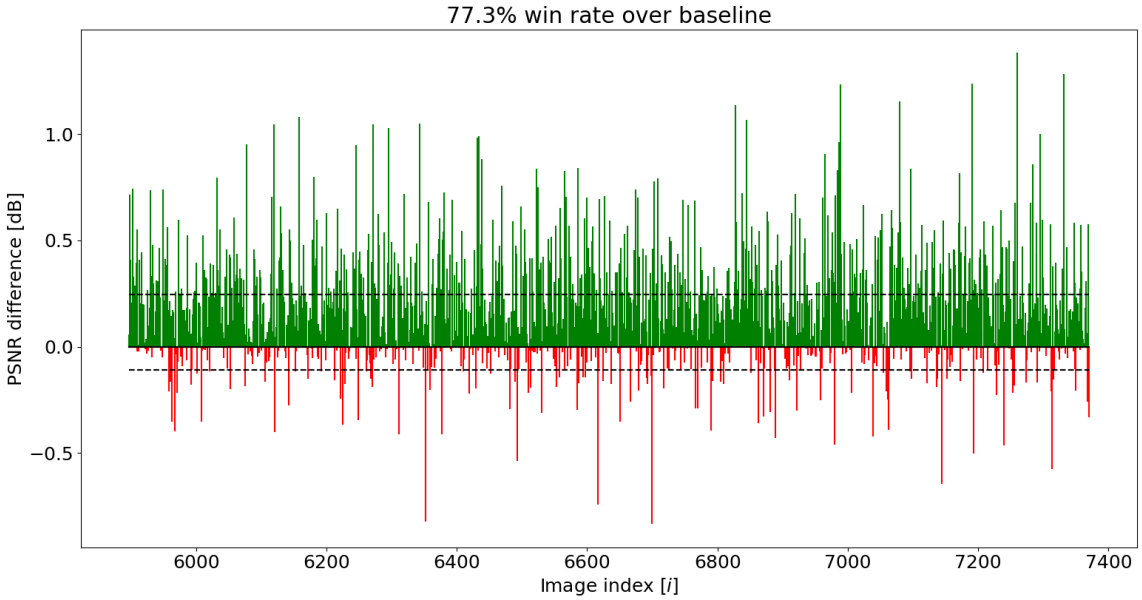

Figure 7.12: ista2vec win rate against ISTA-Net, both trained on 1% of Pets and tested on the remaining 20% of Pets. Green indicates that ista2vec wins and red indicates that ISTA-Net wins.

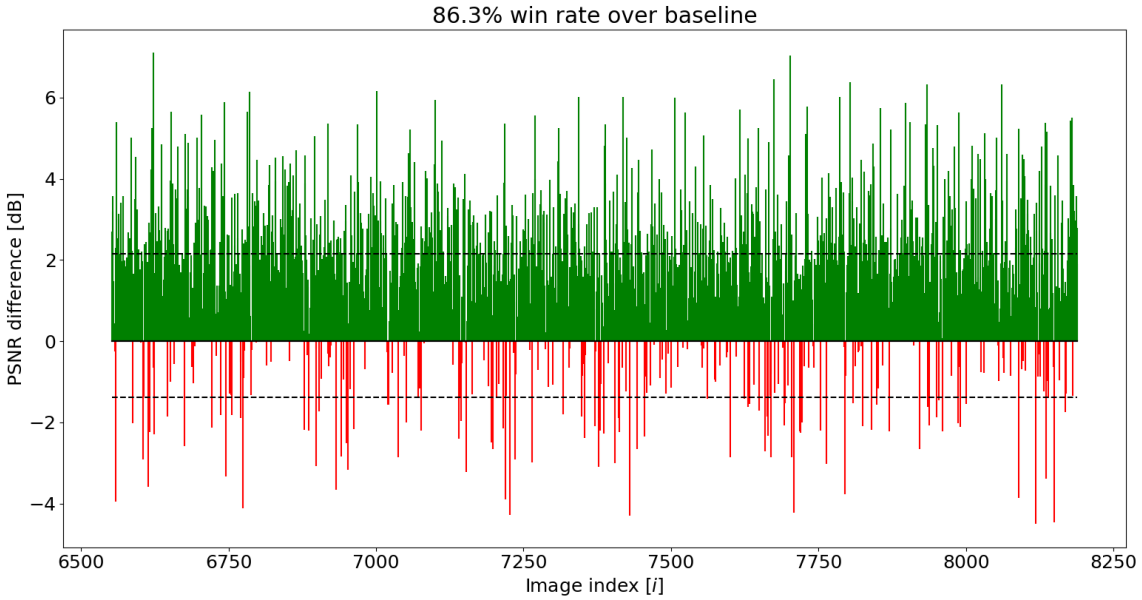

Figure 7.13: ISTA-MAE win rate against regular MAE, both trained on 1% of Flowers and tested on the remaining 20% of Flowers. Green indicates that ISTA-MAE wins and red indicates that regular MAE wins.

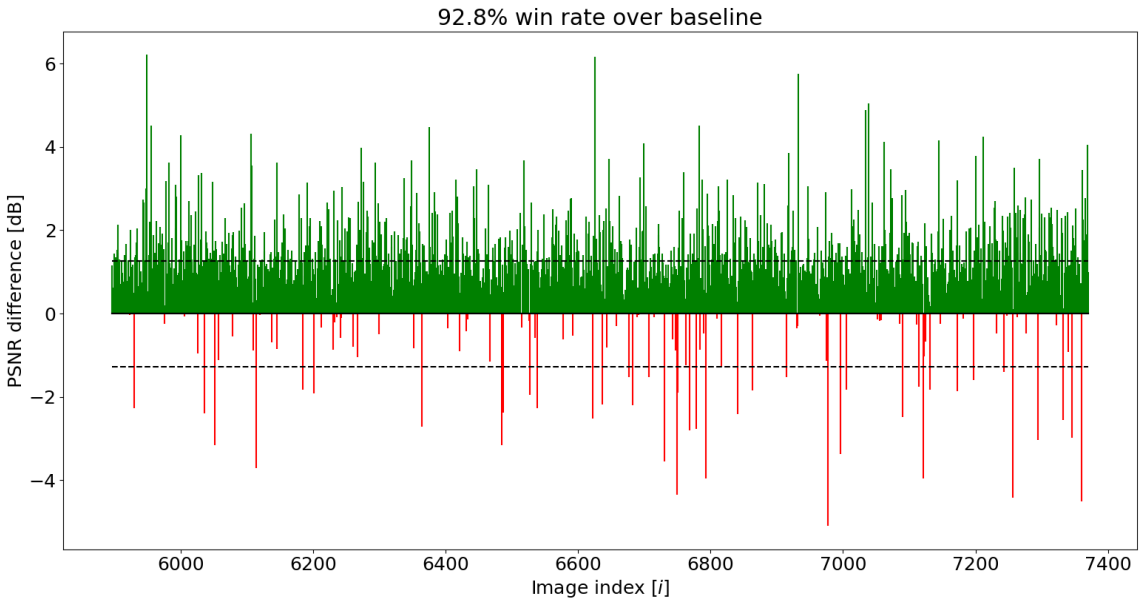

Figure 7.14: ISTA-MAE win rate against regular MAE, both trained on 1% of Pets and tested on the remaining 20% of Pets. Green indicates that ISTA-MAE wins and red indicates that regular MAE wins.

Figure 7.10 shows that the performance is similar for $M_{\text{F,R}}^{\text{MAE}}$ and $M_{\text{F,R}}^{\text{ISTA-MAE}}$, as well as for $M_{\text{P,R}}^{\text{MAE}}$ and $M_{\text{P,R}}^{\text{ISTA-MAE}}$. The win rate plots in Figure 7.13 and 7.14 show that ISTA-MAE has a higher PSNR for 86.3% and 92.8% of the Flowers and Pets test images respectively. Note that for $M_{\text{F,R}}^{\text{ISTA-MAE}}$ has an average lead of approximately 2 dB over $M_{\text{F,R}}^{\text{MAE}}$.

As a final note on the restriction experiments, $M_{\text{F,R}}^{\text{ISTA-Net}}$ and $M_{\text{F,R}}^{\text{ista2vec}}$ outperform $M_{\text{F,R}}^{\text{MAE}}$ and $M_{\text{F,R}}^{\text{ISTA-MAE}}$ as can be seen in Figure 7.10.

7.3 Generality

This section will present the results from the *generality* experiments. This includes the bar plots seen in Figure 7.15 and 7.16, and the win rate plots in Figures 7.17-7.24.

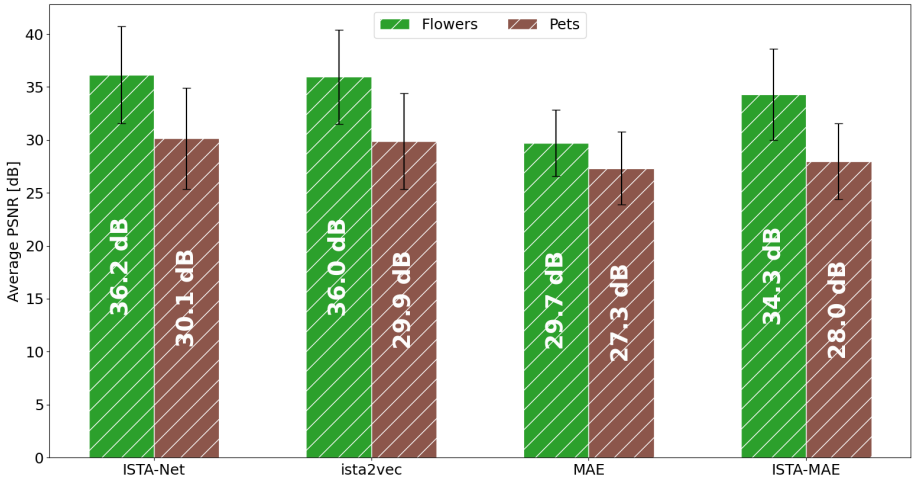

Figure 7.15: Generality experiment. Average PSNR for the M_{F} and M_{P} models. The black line denotes the standard deviation.

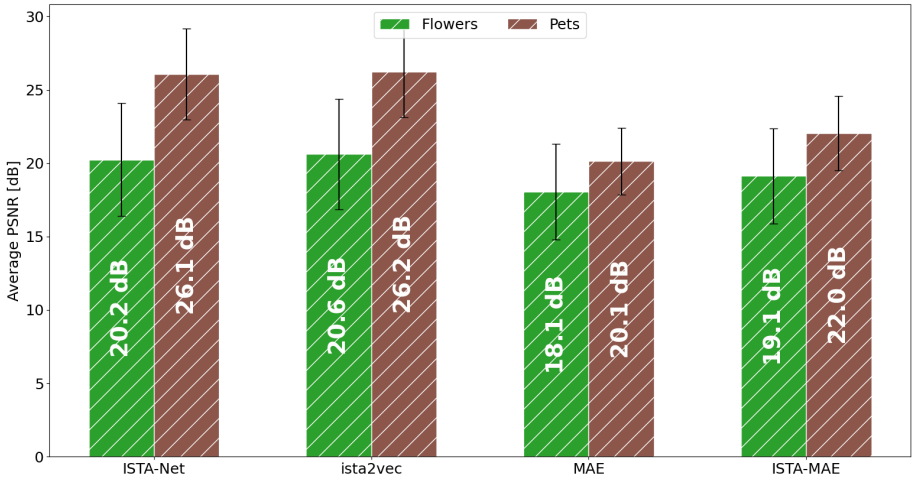

Figure 7.16: Generality experiment. Average PSNR for the $M_{\mathrm{F,R}}$ and $M_{\mathrm{P,R}}$ models. The black line denotes the standard deviation.

The performance of $M_{\mathrm{P}}^{\text{ISTA-Net}}$ compared with $M_{\mathrm{P}}^{\text{ista2vec}}$, and $M_{\mathrm{F}}^{\text{ISTA-Net}}$ compared with $M_{\mathrm{F}}^{\text{ista2vec}}$ is indistinguishable when looking solely at Figure 7.15. However, Figure 7.17 and 7.18, seem to indicate that ISTA-Net is the better performing model, having a difference as high as 5 dB when comparing $M_{\mathrm{F}}^{\text{ISTA-Net}}$ to $M_{\mathrm{F}}^{\text{ista2vec}}$. Note that the average loss margin is smaller than 0.5 dB when comparing ISTA-Net to ista2vec.

When comparing $M_{\mathrm{P}}^{\text{MAE}}$ and $M_{\mathrm{P}}^{\text{ISTA-MAE}}$ on Flowers, Figure 7.15 show that ISTA-MAE outperforms MAE. For $M_{\mathrm{F}}^{\text{MAE}}$ and $M_{\mathrm{F}}^{\text{ISTA-MAE}}$ on Pets however, the figure shows that the models have similar performance. Figure 7.21 and 7.22 show that ISTA-MAE wins for 96.5% and 100% of the test images on Pets and Flowers respectively.

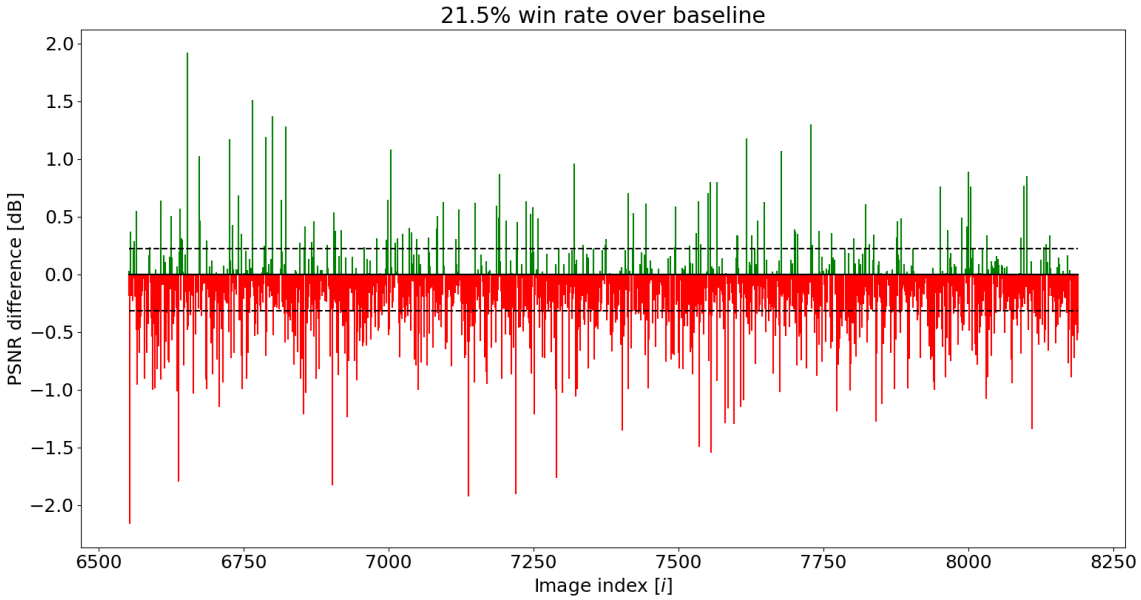

Figure 7.17: ista2vec win rate against ISTA-Net, both trained on 80% of Pets and tested on the remaining 20% of Flowers. Green indicates that ista2vec wins and red indicates that ISTA-Net wins.

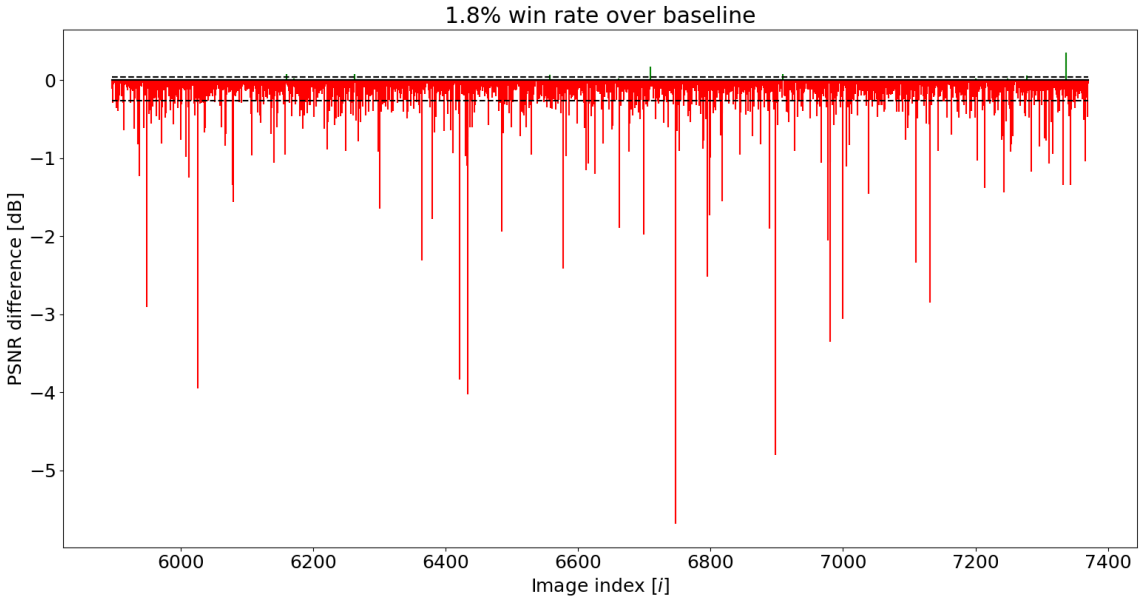

Figure 7.18: ista2vec win rate against ISTA-Net, both trained on 80% of Flowers and tested on the remaining 20% of Pets. Green indicates that ista2vec wins and red indicates that ISTA-Net wins.

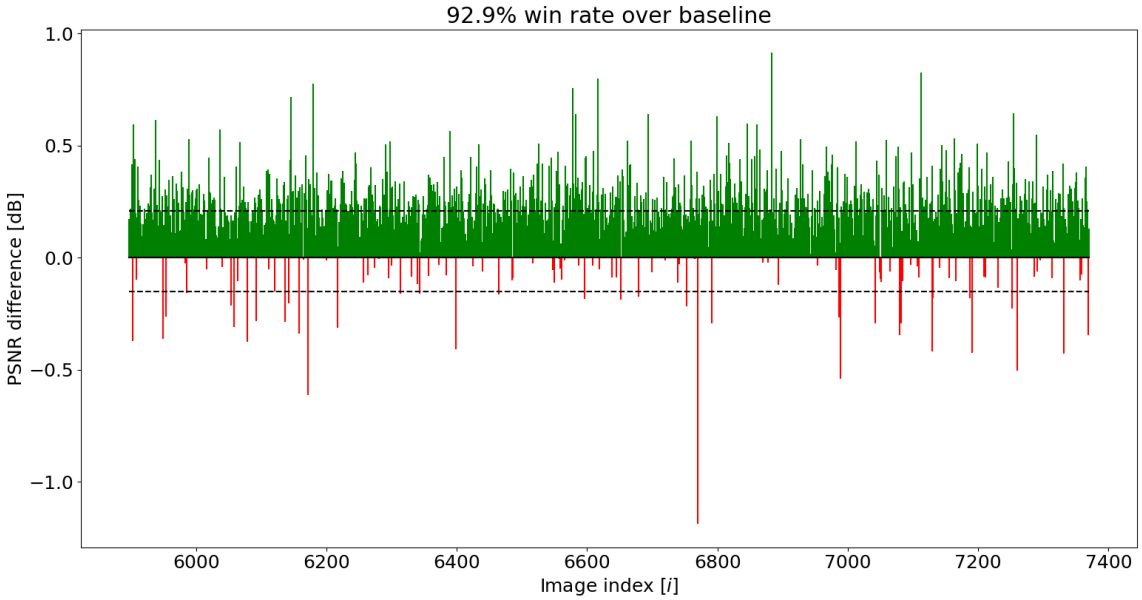

Figure 7.19: ista2vec win rate against ISTA-Net, both trained on 1% of Flowers and tested on the remaining 20% of Pets. Green indicates that ista2vec wins and red indicates that ISTA-Net wins.

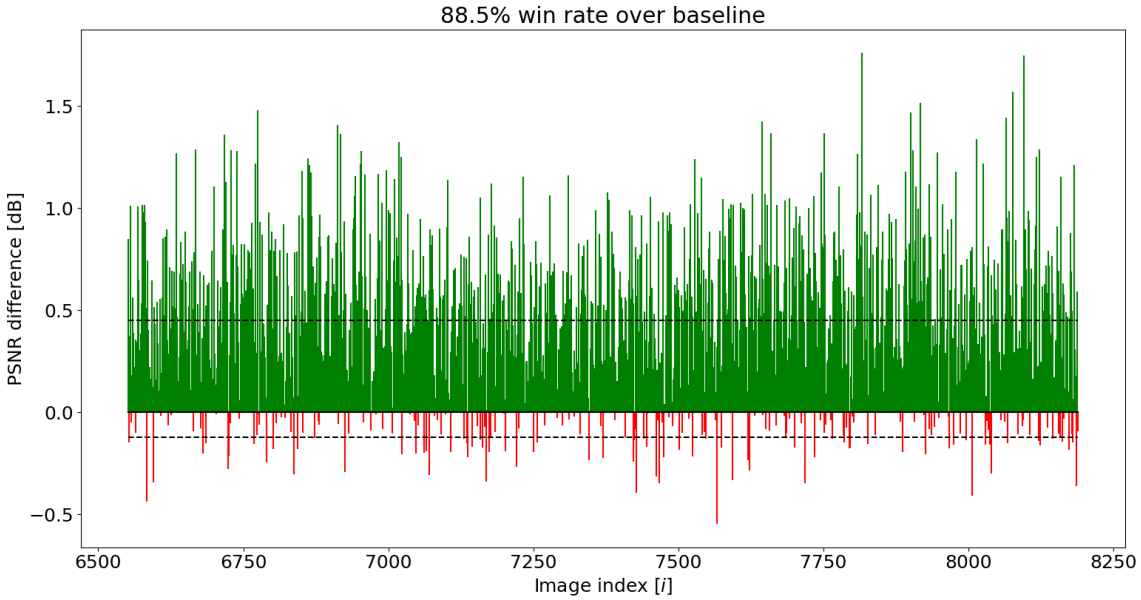

Figure 7.20: ista2vec win rate against ISTA-Net, both trained on 1% of Pets and tested on the remaining 20% of Flowers. Green indicates that ista2vec wins and red indicates that ISTA-Net wins.

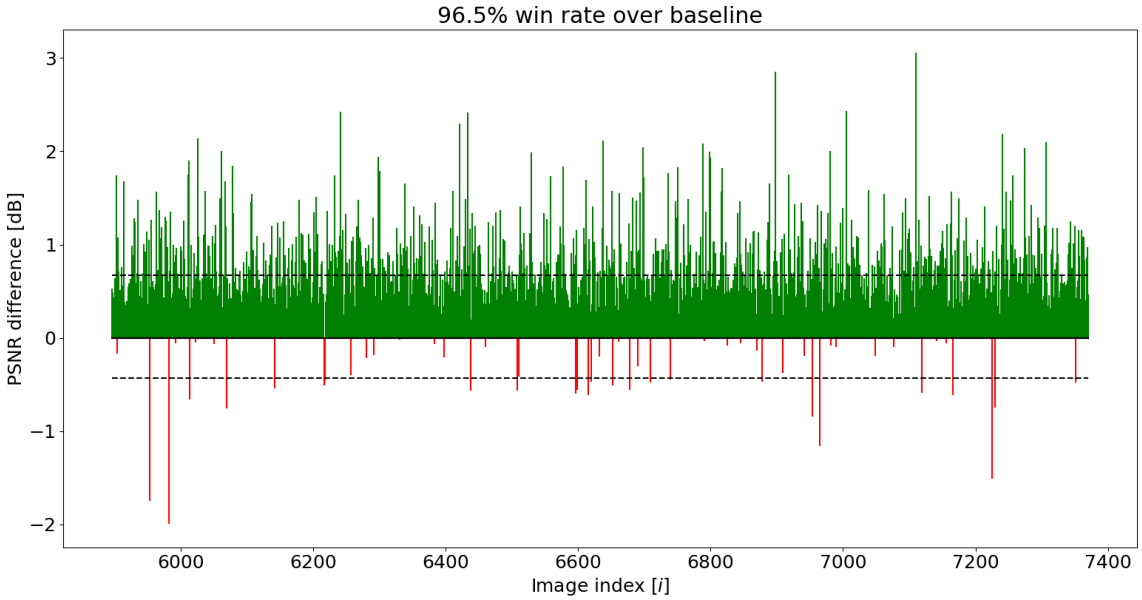

Figure 7.21: ISTA-MAE win rate against regular MAE, both trained on 80% of Flowers and tested on the remaining 20% of Pets. Green indicates that ISTA-MAE wins and red indicates that regular MAE wins.

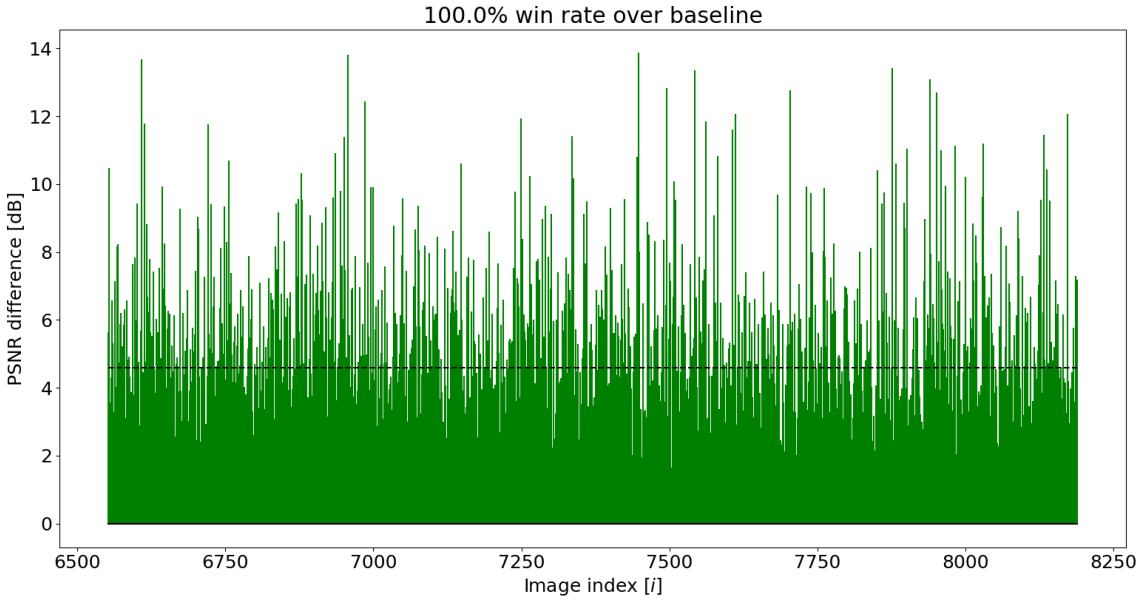

Figure 7.22: ISTA-MAE win rate against regular MAE, both trained on 80% of Pets and tested on the remaining 20% of Flowers. Green indicates that ISTA-MAE wins and red indicates that regular MAE wins.

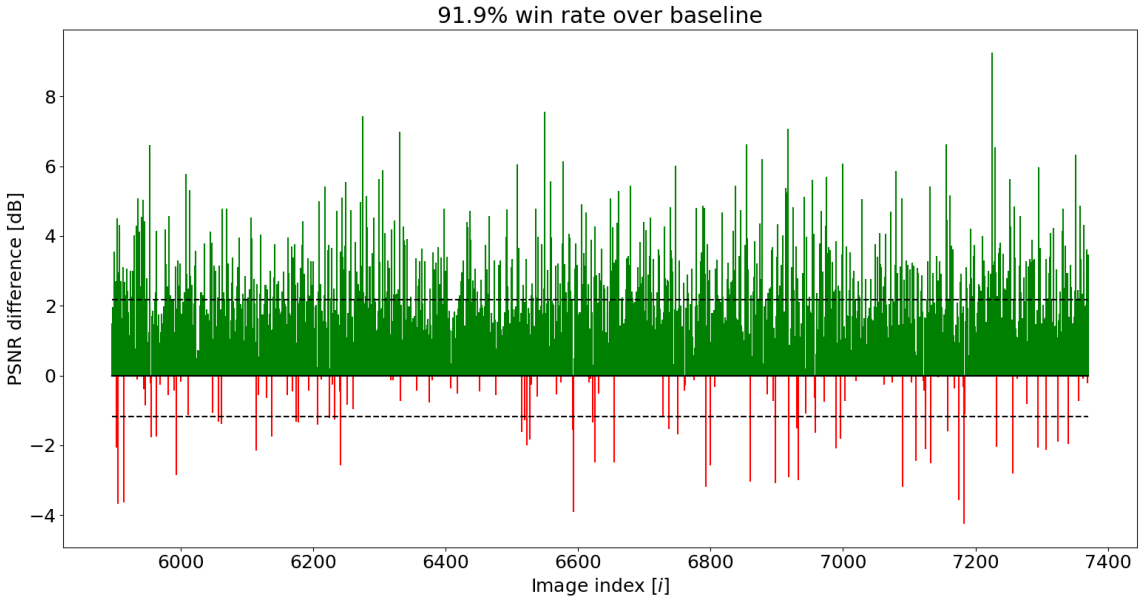

Figure 7.23: ISTA-MAE win rate against regular MAE, both trained on 1% of Flowers and tested on the remaining 20% of Pets. Green indicates that ISTA-MAE wins and red indicates that regular MAE wins.

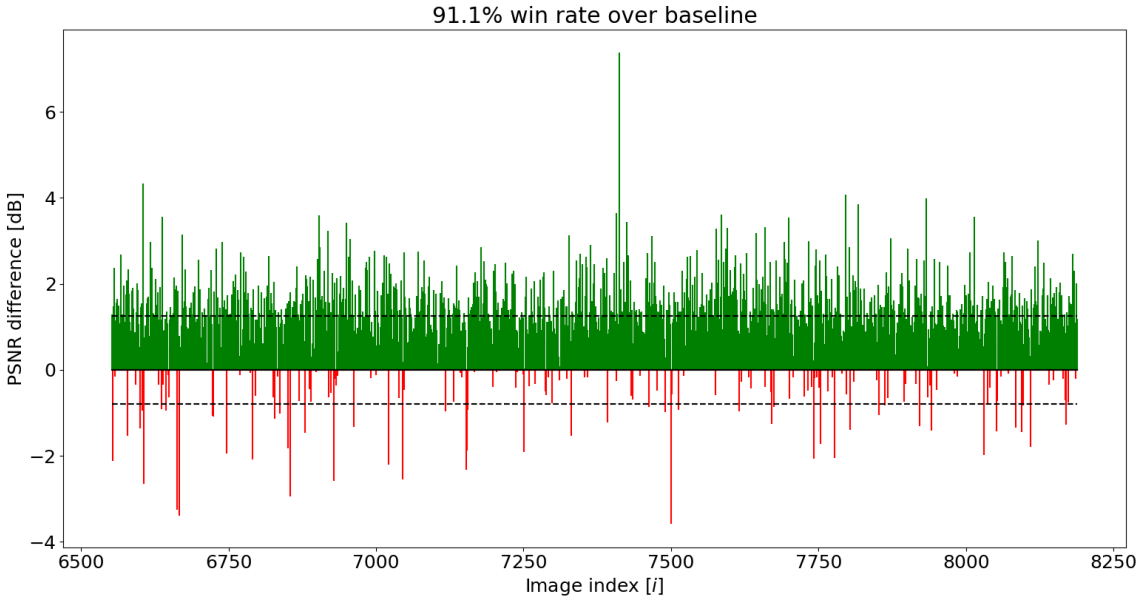

Figure 7.24: ISTA-MAE win rate against regular MAE, both trained on 1% of Pets and tested on the remaining 20% of Flowers. Green indicates that ISTA-MAE wins and red indicates that regular MAE wins.

When only inspecting the average PSNR shown in Figure 7.16, the performance of $M_{\mathrm{P,R}}^{\text{ISTA-Net}}$ compared with $M_{\mathrm{P,R}}^{\text{ista2vec}}$, and $M_{\mathrm{F,R}}^{\text{ISTA-Net}}$ compared with $M_{\mathrm{F,R}}^{\text{ista2vec}}$ is indistinguishable. However, from Figure 7.19 and 7.20 it is seen that $M_{\mathrm{F,R}}^{\text{ista2vec}}$ and $M_{\mathrm{P,R}}^{\text{ista2vec}}$ have much higher win rates than their respective baseline models. Note that the PSNR lead is only around 0.25 dB and 0.5 dB for $M_{\mathrm{F,R}}^{\text{ista2vec}}$ and $M_{\mathrm{P,R}}^{\text{ista2vec}}$ respectively. Further, based on the results shown in Figure 7.16, $M_{\mathrm{F,R}}^{\text{ista2vec}}$ can be said to outperform $M_{\mathrm{F,R}}^{\text{MAE}}$ and $M_{\mathrm{F,R}}^{\text{ISTA-MAE}}$ on Pets.

When comparing $M_{\mathrm{P,R}}^{\text{MAE}}$ and $M_{\mathrm{P,R}}^{\text{ISTA-MAE}}$ on Flowers, Figure 7.16 show that they obtain similar average PSNR. This also applies to $M_{\mathrm{F,R}}^{\text{MAE}}$ and $M_{\mathrm{F,R}}^{\text{ISTA-MAE}}$ on Pets. Figure 7.23 and 7.24 show that ISTA-MAE wins for 91.9% and 91.1% of the test images from Pets and Flowers respectively.

8 | Discussion

The aim of this report is to study novel approaches to synergistically combining algorithm unrolling and self-supervised learning, specifically in the context of solving the super-resolution task. This leads to two proposed models, ista2vec and ISTA-MAE, which are extensions of ISTA-Net and MAE respectively. The experiments outlined in Chapter 6 were conducted to analyse the performance of the models, and the results thereof were presented in Chapter 7. This chapter encompasses a discussion of the results, the validity of the proposed approaches, and general considerations.

8.1 Model Performance

The general results seem to indicate that ISTA-Net and ista2vec perform better than MAE and ISTA-MAE for image super-resolution. This might be attributed to the transformer-based models featuring a huge parameterisation, thus requiring a very large and diverse dataset to train effectively, leading to long training times. Though the models are trained on ImageNet, which contain approximately 1.3 million different images, because of hardware constraints, MAE and ISTA-MAE might not have had enough time to train. Conversely, the unrolled networks feature a much smaller parameterisation, thus alleviating the dataset requirements. Furthermore, ISTA-Net and ista2vec utilise random crops during fine-tuning compared to MAE and ISTA-MAE, which utilise center crops, thus enabling ISTA-Net and ista2vec to better learn the underlying target distribution. However, ISTA-Net and ista2vec are not definitively the best, as Figure 7.5 shows that the PSNR of all Pets based models lie within the standard deviation of each other. Is is also worth noting that there only exists one instance of every model, and thus bad performance can be attributed a bad initialisation.

SSL should allow for more general models, which can be trained to successfully solve a downstream task, utilising a smaller dataset than a purely supervised approach would allow. Consequently, when comparing the performance of ISTA-Net and ista2vec, it is expected that ista2vec is superior in the restriction and generality experiment. The generality results presented in Section 7.3 seem to meet this expectation for the restricted case, while ISTA-Net achieved higher performance for the unrestricted case. This is based on the win rate over ISTA-Net, however, by examining the average win and loss margin, it can be seen that these are smaller than 0.5 dB. Thus, it is difficult to conclude which model is better. In the purely restriction experiment of Section 7.2, the results are very similar to the generality experiment, though with slightly different attributions. However, as the win and loss margins are still relatively small and their average PSNR is similar, it remains difficult to favour either model. The restricted generality experiment poses the most difficult task, as the models are trying to predict out-of-distribution targets, while being restricted to

only 1% of the datasets during training. Thus, it is hypothesised that for the pre-training performed on ista2vec to be advantageous, the downstream task needs to be sufficiently difficult, as the default parameter initialisation otherwise provides a sufficiently efficient starting point.

As described in the results, ISTA-MAE outperforms MAE in all experiments when taking both the average PSNR and the win rate into account. Furthermore, the win margin is generally higher for ISTA-MAE than it is for ista2vec while also achieving 100% win rates in some experiments. Thus there is more evidence suggesting ISTA-MAE is the better model. The increased performance may be attributed to algorithm unrolling allowing the transformer to better utilise the features and generalise them to both datasets, as per the consideration presented in Section 5.3.2.

Note that, in the restriction experiments (Section 7.2) neither ISTA-MAE nor MAE perform well, despite being pre-trained on ImageNet. This may be due to the transfer learning method: The models are trained downstream using fine-tuning instead of linear readout. Thus all ≈5,000,000 parameters need to be fine-tuned on only 1% of the datasets.

Lastly, the performance differences between MAE, ISTA-MAE and ISTA-Net, ista2vec may be explained by the super-resolution task being too difficult for the transformers to perform utilising a single linear projection, and thus a more complex output module such as an MLP may be more suitable.

8.2 Shortcomings and Sources of Error

Though there exists more sophisticated approximations[1] to human perception, the PSNR is a widely used empirical quality measure when reference imagery is available. Furthermore, the PSNR is calculated using the MSE, which is consistent with the training objective across all models. However, as the super-resolution imagery is all based on patching, and PSNR is a per-pixel measure, the PSNR will therefore fail to capture discrepancies between adjacent patches. Thus, for subsequent research, image quality metrics such as PIQE[2] might yield more indicative results.

The datasets themselves constitute a source of error. Both Flowers and Pets are relatively small datasets comprising less than 10,000 images. Pets is especially problematic, as it does not have strongly defined subjects. Thus, the potential features present in Pets might not be limited to features inherent to cats or dogs. In contrast, Flowers consists mostly of centered subjects with a clear distinction between foreground and background. The hypothesis of a difference in features is strengthened when comparing Figure 7.5 and 7.15 to Figure 7.10 and 7.16. These show that the average PSNR is similar for Flowers and Pets with M_{F} and M_{P} except for specifically $M_{\mathrm{P}}^{\text{ISTA-MAE}}$, which has a 3 dB higher average PSNR for Flowers than $M_{\mathrm{F}}^{\text{ISTA-MAE}}$. At the same time, the restriction experiments show that the change in PSNR for Flowers ranges between -3 dB to -6 dB, while it remains mostly the same for Pets, which ranges between -1.5 dB to +1.2 dB. Thus, the features learned from Flowers seem to be more general, and it is expected that Flowers contains a balance between low- and high-frequency features whereas Pets favour high-frequency features. Additional examination of the datasets can be seen in Appendix D.

The extension of MAE to ISTA-MAE results in an increased parametrisation, as the

[1] `https://www.mathworks.com/help/images/image-quality-metrics.html`
[2] `https://mathworks.com/help/images/ref/piqe.html`

linear projection is replaced by a 5 layer LISTA architecture. Thus, an ablation study is warranted, as it is unclear whether the increased performance is a consequence of the introduction of the unrolled network, or simply because of the increased parametrisation. Furthermore, as only fine-tuning is utilised when transferring weights to the downstream task, the ablation study should be extended to also include performance testing of linear readout. This is especially interesting for the restriction experiments as per Section 3.1.1. Note also, that while the ISTA-MAE model utilises an L1 penalty term on the embedding during pre-training to incite more sparse representations and thereby better utilise the domain knowledge behind LISTA, it is uncertain if this regularisation is required. Preliminary experiments showed that weighting the L1 term using $\lambda = 0.0001$ yielded the best performance. As this choice of λ was also the smallest tested, this indicates that it may not be necessary to include the L1 penalty. The ablation study should therefore also include a more thorough examination of the applied L1 regularisation. In the same vein, in its original formulation data2vec solely relies on the smooth-L1 loss. However, the ista2vec model adds the inverse morphism constraint from the ISTA-Net formulation during pre-training. This constraint is added to incentivise a better initialisation for the downstream model, though it is unclear if the constraint is required for effective pre-training. The ablation study should thus include pre-training and fine-tuning with and without the inverse morphism constraint.

8.3 General Considerations

The aim of the project is to examine the intersection of algorithm unrolling and self-supervised learning, specifically how they can be utilised in combination to construct and train DNNs. Thus, the study is limited to establishing strategies for combining both methods, and investigating the consequences thereof. Therefore, comparisons are limited to the proposed models and their baseline counterparts, i.e. any comparison to state-of-the-art methods is deemed beyond the scope. Since the study is unconcerned with state-of-the-art performance, and due to time constraints, limited work has been allocated towards hyperparameter tuning. Consequently, performance results should not be considered as maximal potential the proposed methods. In order for transformer networks to reach their maximal potential, they require a lot of training on copious amounts of data. Dosovitskiy et al. [2021] claimed that the ViT was sufficiently pre-trained to achieve excellent results when trained on the ImageNet21k dataset, which consists of 14,197,122 (14 million) images. Due to computational constraints the ViTs utilised in this project were limited to train on ImageNet and were limited to have ≈5,000,000 parameters, which might have led to subpar performance, compared to the architectures potential.

Because of its direct relation to an iterative algorithm, the latent representations of unrolled networks are tangible, and represent a refined version of the previous decision. Conversely, network architectures typically utilised for self-supervised learning provide abstract latent representations that are not easily related to the input. Thus, the typical SSL architecture does not impose restrictions on the latent representations and allows them to freely mutate to fit the training task. This proves to be a challenge when framing AU in a self-supervised setting, as the benefits of AU are generally attributed to the tangibility of its representations. However, by not sharing parameters across layers, as was otherwise originally proposed [Gregor and LeCun, 2010], the generality increases, which alleviates some of the restrictions initially forced on the latent representation.

Replacing the ViT architecture in data2vec with the ISTA-Net architecture provides a

simple approach to incorporating SSL. However, the original implementation of ISTA-Net exhibits the same behaviour as an iterative algorithm, specifically that each latent representation is a refined version of the last, starting with the input [Zhang and Ghanem, 2018]. Thus, the latent representation that the student predicts is interpretable as a refinement of the input image. Without a supervised objective, the refined representation created by the teacher simply corresponds to an unspecified filtering of the input. Consequently, even if the student learns to represent information in the masked area, it still would not necessarily serve as a good initialisation, as iterative algorithms are advantageously initialised in a state corresponding to a refinement of the original input. Similarly, this is also why utilising a pre-trained network as a trunk to an unrolled network is not presented in this work. As benefits of utilising algorithm unrolling arise from its priors, serving an unrolled network with an abstract latent representation defeats the purpose of AU, unless the latent representation can be guaranteed to fit the priors.

Utilising AU to create input embeddings represents the other proposed approach to combining SSL and AU. This might allow more explicit utilisation of the benefits that the unrolled architecture provides, while also producing abstract representations. This approach does however, not inject interpretability into the gestalt of the model, as the final representation is a heavily processed version of the original more tangible input embedding provided by the unrolled algorithm. Even though the absolute PSNR is not nearly as high as the ISTA-Net based models, the ISTA-MAE model consistently outperforms its MAE counterpart, thus suggesting that the embedding strategy works.

9 | Conclusion

Algorithm unrolling and self-supervised learning can be combined under the unifying mathematical framework of decision making algorithms. From this, two distinct approaches were presented: ista2vec which focuses on utilising self-supervision to train the fully unrolled network ISTA-Net, and ISTA-MAE which incorporates LISTA as a feature extractor in the vision transformer. Though ista2vec is based on the established ISTA-Net architecture, it is cast in the learning framework of data2vec. This allowed ista2vec to be pre-trained, using self-supervision to determine its initialisation. ISTA-MAE extends the ViT architecture by replacing the initial encoding step with LISTA, and pre-trains it using MAE. The idea being to create embeddings based on domain knowledge. Both models were then trained and compared to ISTA-Net and a traditional MAE for two datasets: 102 Category Flower Dataset, and the Oxford-IIIT Pet Dataset. The results showed similar performance between ISTA-Net and ista2vec, while ISTA-MAE outperformed MAE. Because of the limited number of tests, it was impossible to directly attribute any increase in performance to the inclusion of LISTA. The experiments and subsequent discussion does, however, seem to indicate that the ISTA-MAE approach to combining the self-supervised learning and algorithm unrolling is the most synergistic: By replacing task-specific modules of existing general architectures, such as the transformer, with an unrolled algorithm, both the performance and generality may increase. Conversely, it is not evident that self-supervised pre-training of a fully unrolled network exploits the advantages of self-supervised learning, and thus the combination is not properly motivated.

10 | Bibliography

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938.

Abnar, S. and Zuidema, W. (2020). Quantifying attention flow in transformers.

Baevski, A., Hsu, W.-N., Xu, Q., Babu, A., Gu, J., and Auli, M. (2022). data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language.

Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., Schwarzschild, A., Wilson, A. G., Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsiavash, H., LeCun, Y., and Goldblum, M. (2023). A Cookbook of Self-Supervised Learning.

Beck, A. and Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Bloem, P. (2019). Transformers from scratch. `https://peterbloem.nl/blog/transformers`. Last Accessed: 18/04/2023.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations.

Chen, X., Liu, J., Wang, Z., and Yin, W. (2018). Theoretical Linear Convergence of Unfolded ISTA and Its Practical Weights and Thresholds. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 9079–9089. Curran Associates Inc.

Denk, T. (2019). Linear relationships in the transformer's positional encoding. `https://timodenk.com/blog/linear-relationships-in-the-transformers-positional-encoding/`. Last Accessed: 18/04/2023.

Dong, C., Loy, C. C., He, K., and Tang, X. (2015). Image Super-Resolution Using Deep Convolutional Networks.

Donoho, D. L. (2006). For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

Elad, M. (2010). *Sparse and Redundant Representations*. Springer New York, NY, first edition. edition.

Ericsson, L., Gouk, H., Loy, C. C., and Hospedales, T. M. (2022). Self-Supervised Representation Learning: Introduction, Advances, and Challenges. *IEEE Signal Processing Magazine*, 39(3):42–62.

Farsiu, S., Robinson, D., Elad, M., and Milanfar, P. (2004). Advances and Challenges in Super-Resolution. *International Journal of Imaging Systems and Technology*, 14(2):47–57.

Foucart, S. and Rauhut, H. (2013). *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Gregor, K. and LeCun, Y. (2010). Learning Fast Approximations of Sparse Coding. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 399–406.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent: A new approach to Self-Supervised Learning.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2021). Masked autoencoders are scalable vision learners.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

Johnson, J. (2020). Interpretability vs explainability: The black box of machine learning. Last acessed: 26-04-2023.

Kazemnejad, A. (2019). Transformer architecture: The positional encoding. `https://kazemnejad.com/blog/transformer_architecture_positional_encoding/`. Last Accessed: 18/04/2023.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization.

Monga, V., Li, Y., and Eldar, Y. C. (2019). Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing.

Nilsback, M.-E. and Zisserman, A. (2008). Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*.

OpenAI (2023). Gpt-4 technical report.

O'Searcoid, M. (2007). *Metric Spaces*. Springer.

Parikh, N. and Boyd, S. (2014). Proximal Algorithms. *Foundations and Trends in Optimization*. `https://web.stanford.edu/~boyd/papers/pdf/prox_algs.pdf`.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. (2012). Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Pham, M., Cho, M., Joshi, A., and Hegde, C. (2022). Revisiting self-distillation.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Salem, F. M. (2022). *Recurrent neural networks : from simple to gated architectures*. Springer, Cham, Switzerland.

Shlezinger, N., Eldar, Y. C., and Boyd, S. P. (2022). Model-Based Deep Learning: On the Intersection of Deep Learning and Optimization.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Yang, J., Wright, J., Huang, T., and Ma, Y. (2008). Image super-resolution as sparse representation of raw image patches. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Zhang, J. and Ghanem, B. (2018). Ista-net: Interpretable optimization-inspired deep network for image compressive sensing.

Appendices

A | ISTA Preliminaries

The content of this Appendix is mainly based on [Parikh and Boyd, 2014].

Lipschitz Functions and Contraction

Definition A.1 (Lipschitz Function)
Given metric spaces (X,e) and (Y,d) and a function $f : X \to Y$, then f is called L-Lipschitz on X with Lipschitz constant $L \in \mathbb{R}^+$, if there exists a k such that

$$d(f(a),f(b)) \leq Le(a,b), \quad \forall a,b \in X.$$

[O'Searcoid, 2007, p.154]

Definition A.2 (Proximal Operator)
Given a closed proper convex function f, the *Proximal operator* $\mathbf{prox}_f(v) : \mathbb{R}^n \to \mathbb{R}^n$ of f is defined as

$$\mathbf{prox}_f(v) = \arg\min_x \left(f(x) + \frac{1}{2}||x - v||_2^2 \right).$$

The function which is minimised on the r.h.s above is strongly convex and not infinite everywhere – it has a unique minimiser for every v. For a scaled function λf, where $\lambda > 0$, the proximal operator can be expressed as

$$\mathbf{prox}_{\lambda f}(v) = \arg\min_x \left(f(x) + \frac{1}{2\lambda}||x - v||_2^2 \right),$$

which is referred to as the proximal operator of f with parameter λ.

[Parikh and Boyd, 2014, p.124]

The proximal operator, $\mathbf{prox}_{\lambda f}(v)$, attempts to reduce the value of f without moving too far away from the point v. The proximal operator $\mathbf{prox}_f$ can be interpreted as a sort of gradient step for the convex function f. The parameter λ dictates how large the penalty is for straying away from v, in effect determining how far the points are mapped. Upon application the points originally inside the domain of the function f remain inside, and the points outside the domain move towards the boundary and the minimum. Since explicit formulas of proximal operators are available for many simple penalty functions

f, proximal operators can often be computed efficiently. In addition, proximal operators do not require f to be differentiable, making proximal operators useful for non-smooth optimisation problems. [Parikh and Boyd, 2014, p. 124-125]

B | Proximal Operator Derivations

B.1 With L1 Regularisation in Transform Domain

Consider the optimisation problem

$$\mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{x} - \mathbf{r}^{(k)}\right\|_2^2 + \lambda\|\Psi\mathbf{x}\|_1$$

with solution

$$\text{prox}_{\lambda f}(\mathbf{r}) = \arg\min_{\mathbf{x}}(\|\Psi\mathbf{x}\|_1 + \frac{1}{2\lambda}\|\mathbf{x} - \mathbf{r}\|_2^2) = \arg\min_{\mathbf{x}} \varphi(\mathbf{x},\mathbf{r}).$$

An analytical solution to the proximal operator is found by finding a critical point of $\varphi(\mathbf{x},\mathbf{r})$. The derivative of $\varphi(\mathbf{x},\mathbf{r})$ w.r.t. $\mathbf{x}$ is

$$\begin{aligned}\frac{\partial\varphi(\mathbf{x},\mathbf{r})}{\partial\mathbf{x}} &= \frac{1}{2\lambda}\left(\frac{\partial}{\partial\mathbf{x}}\mathbf{x}^\intercal\mathbf{x} - \frac{\partial}{\partial\mathbf{x}}\mathbf{x}^\intercal\mathbf{r} - \frac{\partial}{\partial\mathbf{x}}\mathbf{r}^\intercal\mathbf{x}\right) + \frac{\partial}{\partial\mathbf{x}}\|\Psi\mathbf{x}\|_1 \\ &= \frac{1}{2\lambda}(2\mathbf{x}^\intercal - 2\mathbf{r}^\intercal) + \text{sign}(\Psi\mathbf{x})^\intercal\Psi \\ &= \frac{1}{\lambda}(\mathbf{x}^\intercal - \mathbf{r}^\intercal) + \text{sign}(\Psi\mathbf{x})^\intercal\Psi.\end{aligned}$$

To see that $\frac{\partial}{\partial\mathbf{x}}\|\Psi\mathbf{x}\|_1 = \text{sign}(\Psi\mathbf{x})^\intercal\Psi$ notice that for $\Psi \in \mathbb{R}^{m\times n}$ and $\mathbf{x} \in \mathbb{R}^n$

$$\begin{aligned}\frac{\partial}{\partial\mathbf{x}}\|\Psi\mathbf{x}\|_1 &= \frac{\partial}{\partial\mathbf{x}}(|\Psi_1^\intercal\mathbf{x}| + |\Psi_2^\intercal\mathbf{x}| + \cdots + |\Psi_m^\intercal\mathbf{x}|) \\ &= \text{sign}(\Psi_1^\intercal\mathbf{x})\Psi_1^\intercal + \text{sign}(\Psi_2^\intercal\mathbf{x})\Psi_2^\intercal + \cdots + \text{sign}(\Psi_m^\intercal\mathbf{x})\Psi_m^\intercal \\ &= \text{sign}(\Psi\mathbf{x})^\intercal\Psi.\end{aligned}$$

By setting the derivative to zero, the critical point can be found as

$$\begin{aligned}\frac{\partial\varphi(\mathbf{x},\mathbf{r})}{\partial\mathbf{x}} = \frac{1}{\lambda}(\mathbf{x}^\intercal - \mathbf{r}^\intercal) + \text{sign}(\Psi\mathbf{x})\Psi^\intercal &= 0 \\ \implies \mathbf{r}^\intercal = \mathbf{x}^\intercal + \lambda\,\text{sign}(\Psi\mathbf{x})\Psi^\intercal \\ \mathbf{r} = \mathbf{x} + \lambda\Psi^\intercal\,\text{sign}(\Psi\mathbf{x}).\end{aligned}$$

Now assume that Ψ is orthogonal, then

$$\begin{aligned}\Psi\mathbf{r} &= \Psi\mathbf{x} + \lambda\Psi\Psi^\intercal\,\text{sign}(\Psi\mathbf{x}) \\ &= \Psi\mathbf{x} + \lambda\,\text{sign}(\Psi\mathbf{x}).\end{aligned}$$

By examining the entries of $\Psi\mathbf{r}$ it is found that

$$(\Psi\mathbf{r})_i = \begin{cases} (\Psi\mathbf{x})_i - \lambda, & (\Psi\mathbf{x})_i < 0 \\ (\Psi\mathbf{x})_i + \lambda, & (\Psi\mathbf{x})_i > 0. \end{cases}$$

It follows that

$$(\Psi\mathbf{x})_i = \begin{cases} (\Psi\mathbf{r})_i + \lambda, & (\Psi\mathbf{r})_i < -\lambda \\ 0, & |(\Psi\mathbf{r})_i| \leq \lambda \\ (\Psi\mathbf{r})_i - \lambda, & (\Psi\mathbf{r})_i > \lambda. \end{cases}$$

This can be expressed using the soft-shrinkage function S_ν as

$$\Psi\mathbf{x} = S_\lambda(\Psi\mathbf{r}).$$

Finally, by orthogonality

$$\mathbf{x} = \Psi^\intercal S_\lambda(\Psi\mathbf{r}).$$

B.2 With L1 Regularisation in Nonlinear Domain

Consider the optimisation problem

$$\mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{x} - \mathbf{r}^{(k)}\right\|_2^2 + \lambda\|\mathcal{F}(\mathbf{x})\|_1$$

for nonlinear function $\mathcal{F}$. By Zhang and Ghanem [2018]

$$\left\|\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{r}^{(k)})\right\| \approx \alpha\left\|\mathbf{x} - \mathbf{r}^{(k)}\right\|_2^2$$

and thus the optimisation problem is approximated as

$$\mathbf{x}^{(k)} \approx \arg\min_{\mathbf{x}} \frac{1}{2\alpha}\left\|\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{r}^{(k)})\right\|_2^2 + \lambda\|\mathcal{F}(\mathbf{x})\|_1.$$

The solution is given by the proximal operator

$$\text{prox}_{\lambda f}(\mathbf{r}) = \arg\min_{\mathbf{x}}(\|\mathcal{F}(\mathbf{x})\|_1 + \frac{1}{2\alpha\lambda}\|\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{r})\|_2^2) = \arg\min_{\mathbf{x}} \varphi(\mathbf{x},\mathbf{r}).$$

An analytical solution to the proximal operator is found by finding a critical point of $\varphi(\mathbf{x},\mathbf{r})$. To find the derivative of $\varphi(\mathbf{x},\mathbf{r})$ w.r.t. $\mathbf{x}$, the substitution $\mathbf{z} = \mathcal{F}(\mathbf{x})$ and $\mathbf{y} = \mathcal{F}(\mathbf{r})$ is used, and it is assumed that $\mathcal{F}'$ exists. Thus,

$$\frac{\partial\varphi(\mathbf{x},\mathbf{r})}{\partial\mathbf{x}} = \frac{1}{2\alpha\lambda}\left(\frac{\partial\mathbf{z}^\intercal\mathbf{z}}{\partial\mathbf{z}}\frac{\partial\mathbf{z}}{\partial\mathbf{x}} - \frac{\partial\mathbf{z}^\intercal\mathbf{y}}{\partial\mathbf{z}}\frac{\partial\mathbf{z}}{\partial\mathbf{x}} - \frac{\partial\mathbf{y}^\intercal\mathbf{z}}{\partial\mathbf{z}}\frac{\partial\mathbf{z}}{\partial\mathbf{x}}\right) + \frac{\partial}{\partial\mathbf{z}}\|\mathbf{z}\|_1$$

with $\partial\mathbf{z}/\partial\mathbf{x} = \mathcal{F}'(\mathbf{x})$. Simplifying yields the expression

$$\begin{aligned}\frac{\partial\varphi(\mathbf{x},\mathbf{r})}{\partial\mathbf{x}} &= \frac{1}{2\alpha\lambda}(2\mathbf{z}^\intercal - 2\mathbf{y}^\intercal)\mathcal{F}'(\mathbf{x}) + \text{sign}(\mathbf{z})^\intercal\mathcal{F}'(\mathbf{x}) \\ &= \left(\left(\frac{1}{\alpha\lambda}\mathcal{F}(\mathbf{x})^\intercal - \frac{1}{\alpha\lambda}\mathcal{F}(\mathbf{r})^\intercal\right) + \text{sign}(\mathcal{F}(\mathbf{x}))^\intercal\right)\mathcal{F}'(\mathbf{x}).\end{aligned}$$

Due to convexity of norms there exists only one minimum which can be found by solving

$$\begin{aligned} 0 &= \frac{1}{\alpha\lambda}(\mathcal{F}(\mathbf{x})^\intercal - \mathcal{F}(\mathbf{r})^\intercal) + \operatorname{sign}(\mathcal{F}(\mathbf{x}))^\intercal \\ \mathcal{F}(\mathbf{r}) &= \mathcal{F}(\mathbf{x}) + \alpha\lambda \operatorname{sign}(\mathcal{F}(\mathbf{x})). \end{aligned}$$

Using similar arguments as in Section B.1, the solution is found as

$$\mathcal{F}(\mathbf{x}) = S_{\alpha\lambda}(\mathcal{F}(\mathbf{r})).$$

Assume that there exists $\tilde{\mathcal{F}}$ satisfying that $\tilde{\mathcal{F}} \circ \mathcal{F} = \zeta \operatorname{id}$, then

$$\mathbf{x} = \frac{1}{\zeta}\tilde{\mathcal{F}}(\mathcal{F}(\mathbf{x})) = \frac{1}{\zeta}\tilde{\mathcal{F}}(S_{\alpha\lambda}(\mathcal{F}(\mathbf{r}))) \propto \tilde{\mathcal{F}}(S_{\alpha\lambda}(\mathcal{F}(\mathbf{r}))).$$

C | Positional Encoding

Theorem C.1 (Linear Relationships in Sinusoidal Positional Encodings) Let

$$\mathbf{p}_i = \begin{bmatrix} \sin(i\omega_1) \\ \cos(i\omega_1) \\ \sin(i\omega_2) \\ \cos(i\omega_2) \\ \vdots \\ \sin\Big(i\omega_{n/2}\Big) \\ \sin\Big(i\omega_{n/2}\Big) \end{bmatrix}, \quad \omega_i \triangleq \rho^{-\frac{2i}{n}},$$

where $\rho \in \mathbb{R}^+$, $i \in \mathbb{N}$, and $\mathbf{p}_i \in \mathbb{R}^n$.

Then there exists a linear transformation $T^k \in \mathbb{R}^{n \times n}$ such that

$$T^k \mathbf{p}_i = \mathbf{p}_{\mathbf{i+k}},$$

for any $k \in \mathbb{N}$.

Denk [2019]

Proof: Consider the following rules of arithmetic for sinusoids:

$$\begin{aligned} \sin(\alpha + \beta) &= \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta) \\ \cos(\alpha + \beta) &= \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta). \end{aligned}$$

This corresponds to

$$\begin{bmatrix} \sin((i+k)\omega_j) \\ \cos((i+k)\omega_j) \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(k\omega_j) & \sin(k\omega_j) \\ -\sin(k\omega_j) & \cos(k\omega_j) \end{bmatrix}}_{\Psi_j} \begin{bmatrix} \sin(i\omega_j) \\ \cos(i\omega_j) \end{bmatrix}.$$

Consequently, T^k can be designed as

$$T^k = \begin{bmatrix} \Psi_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Psi_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Psi_{n/2} \end{bmatrix},$$

where

$$\mathbf{0} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

■

D | Dataset Deficiencies

The performance of every model is affected by the quality of the dataset. This section aims to explore the datasets to highlight potential problems, as well as to describe how these are expected to affect the trained models.

D.0.1 Monotone Background Seperation

As both datasets contain images taken in the real world with a camera, the contrast between the background and subject varies from image to image. Both datasets feature images in which the subject is emphasized by stark contrast to the background. Examples of these can be seen in Figure D.1 As can be seen, the separation between subject and background is achived using a (mostly) monochrome background. Consequently, the model might be inclined to learn low-pass filters in order to adequately represent these images. As these images contrast with the larger part of the dataset, in which the background includes some sort of texture, emphasizing low-frequency information might leads to a degradation in performance.

D.0.2 Unclear Subject

The datasets are supposed to feature flowers or pets as the subject in the images. However, both datasets include images in which it is not necessarily clear that these are the subjects. Examples of these can be seen in Figure D.2. Consequently, the datasets might contain a much more diverse set of images than initially expected. Though this can be considered a positive for some tasks, this might significantly increase the complexity of the image super-resolution task.

D.0.3 Unnatural Imagery

When doing image compressive sensing, there is often an assumption of working with "natural" images. This generally refers to images which are not artificially constructed, such as in a studio or using graphic design. Thus, as both datasets include images which contains lettering either beside of across the main subject, and aspects of graphical design, reconstruction performance might suffer. Examples of these can be seen in Figure D.3

Figure D.1: Example of images with an increased contrast between subject and background.

Figure D.2: Example of images where there exists multiple subjects which are not flowers or pets.

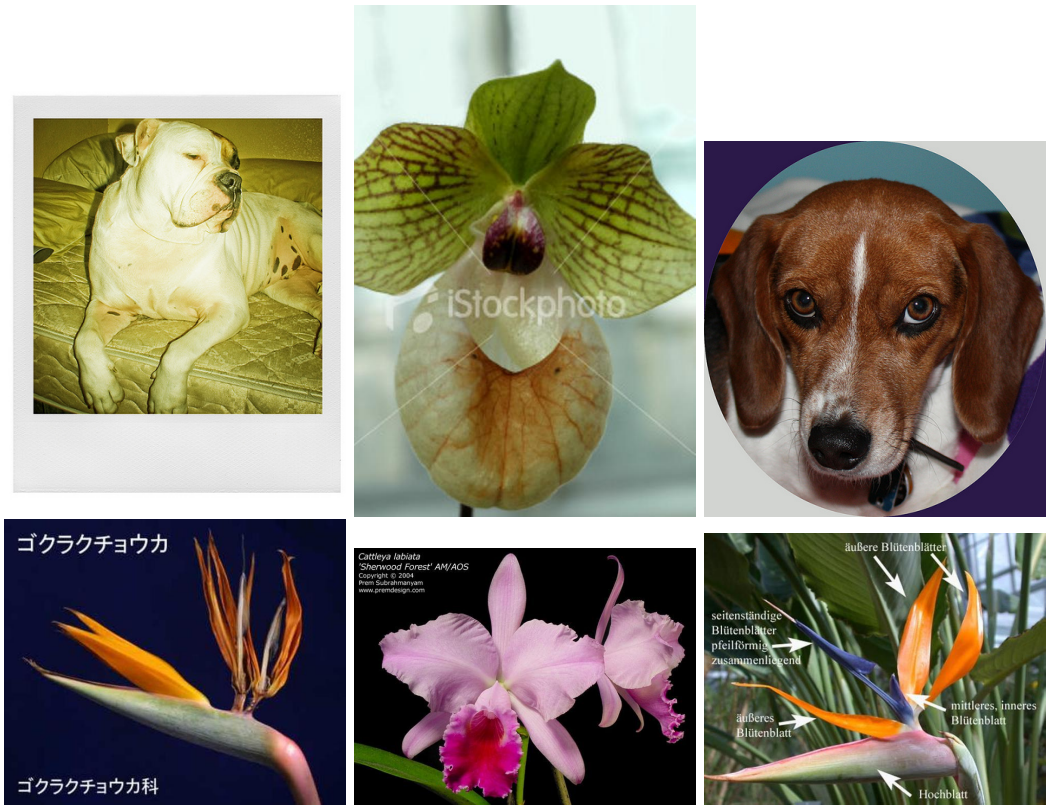

Figure D.3: Example of images with lettering or graphic design.