A novel perceptual metric in machine learning

Johanna Engman och Hanna Nilsson

Master's thesis
2020:E33

Lund Institute of Technology
Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

A Novel Perceptual Metric in Deep Learning

A comparison of Loss Functions for Image Denoising and Reconstruction

Johanna Engman
Hanna Nilsson

26 maj 2020

Department of Mathematics
Lund University
Academic supervisor: Magnus Oskarsson, LTH
Industrial supervisor: Jim Nilsson, NVIDIA
Examiner: Karl Åström

Abstract

Loss functions are a crucial part of image processing when using modern neural networks, trained with stochastic gradient descent. There exist multiple loss functions today, some claiming to be perceptual. Researchers at NVIDIA recently published a proposal of such a metric called FLIP. This led to our work on this thesis where we present a comparison between multiple loss functions, both established ones but also a completely new one. The loss functions were subjected to two problems, denoising RGB images and reconstructing MR images.

The central question is how does one evaluate the evaluation? If we train a network with the same architecture but with different loss functions, and then measure the performance with one of these loss functions, the result would most likely be biased. We therefore present a comparison between loss functions where we both visually and numerically evaluate the results as well as conducting user studies. We find that the weighted version of LPIPS + l_2 and MS-SSIM are especially good loss functions for mentioned problems, and the newly proposed metric FLIP performed well.

Keywords: Loss function, metric, deep learning, denoising, MRI reconstruction, U-net

Acknowledgement

A big thank you to everyone at the NVIDIA office in Lund, and a special thanks to Jim Nilsson, Jacob Munkberg, Pontus Andersson and Jon Hasselgren. Your expertises in deep learning and surrounding fields have been invaluable, as well as lending us computer hardware. Thanks is of course also due to our supervisor Magnus Oskarsson for discussions, feedback and always being updated on our work.

Lastly, we would like to thank the researchers at NVIDIA and our friends and family for participating in our user studies.

Johanna Engman and Hanna Nilsson
Lund, May 2020

Abbreviation

ANN Artificial Neural Network

CNN Convolutional Neural Network

ReLU Rectified Linear Unit

SGD Stochastic Gradient Descent

Adam Adaptive Moment

(s)RGB (Standard) Red Green Blue

SSIM Structural Similarity

MS-SSIM Multi Scale Structural Similarity

LPIPS Learned Perceptual Image Patch Similarity

MRI Magnetic Resonance Imaging

SNR Signal-to-noise Ratio

PSNR Peak Signal-to-noise Ratio

Contents

1 Introduction

During this thesis project we have been working closely together. We have divided the responsibility over different tasks equally and both of us are well agreed with the work of the other. This master's thesis was carried out at NVIDIA in Lund.

The report starts with a background to the problem of restoring images with deep learning, and continutes with a review of previous work within the same field. A section with theory on deep learning and especially different loss functions is stated, where theory on the investigated metric FLIP follows. Next in the report, the used datasets are presented. Further on we explain our method for our thesis, followed by the results we received. Finally, we discuss our findings and state a conclusion.

1.1 Background

The ability of restoring damaged images in an efficient and powerful manner has long been a goal within image analysis and machine learning. The development of the U-net [Ronneberger et al., 2015] was a breakthrough when it came to autoencoders' ability to successfully restore images.

A leading question is how to evaluate the performance of a network. Partly after the training is completed and partly, and maybe most importantly, during the training. The evaluation of the performance of the network when the training is complete is assigned to a chosen metric whereas the evaluation during training is a combination of a loss function and, for example, stochastic gradient descent. It is desirable to have a good way of calculating where in the image the model have succeeded with the recreating to conclude where the model needs to adjust and focus more.

How does one determine if two images are the same, with some threshold for dissimilarity? There are different approaches. Comparing pixel values in said images to obtain a good result is one, where only the individual pixels matter, and not the complete image. Another approach is trying to resemble the human visual system (HVS). The aim of the MRI reconstruction is that the restored images should resemble the original as much as possible according to a human, standing at a given distance from the images. Then there is no real need for the images to be precisely the same i.e., having the same pixel values. As long as the human experiences the images as equal, it could be good enough.

As mentioned above, the viewing distance could be taken into account when evaluating errors in an image. This is what the newly suggested perceptual metric FLIP [Anonymous, 2020] takes into consideration.

Another interesting application of image reconstruction is improving the quality of medical imaging. To reduce the time for a patient to be in a MR machine, several improvements have been made. In 2006 *compressed sensing* was introduced [Zbontar et al., 2018] to speed up the MR imaging process, although this led to undersampling and violation of the Nyquist theorem resulting in aliasing artifacts in the images. This speed-up and the consequences of it introduces a large need for MRI reconstruction.

Previously, medical images and data have been limited to hospitals and universities in collaboration with these hospitals. With the fastMRI release [Zbontar et al., 2018], however, a large dataset of MR images was

made available to researchers all over the world, triggering research in this field.

The new metric FLIP alongside with the new fast MRI dataset triggers curiosity. Both for how FLIP will perform compared to well known metrics and state-of-the-art metrics for deep learning but also for how different metrics perform in reconstruction of MR images. It is these curiosities that has triggered the ideas for this master thesis.

1.2 Previous Work

Previous work in the field of image analysis and deep learning connected to evaluating loss functions have focused both on evaluating existing metrics and developing new ones. Relevant research has also focused on the architecture of networks for image restoration.

A new architecture of neural networks was presented by [Ronneberger et al., 2015]. In the paper they propose their design, called U-net, which is further explained in section 2.2.1. The proposed network makes use of data augmentations, allowing the network to learn a larger capacity of invariance of the data. This makes the network well suited for when the task comes with a small dataset which is often the case when dealing with biomedical images.

Furthermore, there are multiple relevant research project about metrics. In [Wang et al., 2003], which is a continuation of the work of the Structural Similarity Index (SSIM) [Wang et al., 2004], the authors present a perceptual image quality assessment method named Multi-Scale Structural Similarity Index (MS-SSIM) which analyzes the statistical differences in an image. This metric is a more flexible variant of SSIM regarding variations in image resolution and viewing conditions. The authors state that ”the perceivability of image details depends on the sampling density of the image signal, the distance from the image plane to the observer, and the perceptual capability of the observer's visual system”. They point out that the commonly used metrics peak signal-to-noise ratio (PSNR) and mean squared error (MSE) cannot account for perceptuality well. The paper presents a comparison of different image quality assessment methods, including PSNR, MS-SSIM and its precursor SSIM. The metrics are measured against each other with different criteria, for example outlier ratio, the mean absolute error and the root mean squared error. The experiment showed that MS-SSIM exceeds the optimal SSIM model alongside with state-of-the-art image quality metrics. The results are based on that MS-SSIM is fed appropriate model parameters. The calibration of model parameters in image quality models is a rather challenging task, for which the authors propose an image synthesis approach.

A more recent metric, called LPIPS, is described in [Zhang et al., 2018]. It is based on an existing network as it uses internal activation of networks which are trained for image classification tasks. This amounts to the use of pretrained weights when training the network, to aim for perceptuality. The experiments are conducted on three network architectures, SqueezeNet, AlexNet and VGG [Zhang et al., 2018]. The authors propose the use of AlexNet as it is the fastest and gives the best performance. They mention that commonly used metrics such as PSNR and SSIM fail to account for several parts of human perception. They bring forward the aspiration of a ”perceptual distance, which measures how similar two images are in a way that coincides with human judgment”. The authors describe the struggles to establish such a metric, for example as there are different ”senses of similarity” with an example such as ”is a red circle more similar to a red square or to a blue circle?”. To evaluate their result they employ a two alternative forced choice (2AFC) study to confirm which of two distortions is more similar to the original image. To validate the experiment they also perform a just noticeable difference (JND) test, which presents the user with two patches, one distorted and one original, and asks if these patches are the same or not. They argue that a 2AFC test can present subjectivity as the users can ”consciously choose which respects of similarity they will choose to focus on". As there is a single correct answer for their JND test and therefore less subjective, the authors believe that this combination of two tests will result in meaningful results.

Barron [Barron, 2017] proposes a new loss function called General & Adaptive Robust Loss which is a generalization of several already existing loss functions such as the common l_1 and l_2. He motivates this by

advocating for the need of robustness, i.e., "that a model be less influenced by outliers than by inliers". He introduces a new robustness hyperparameter which is automatically determined by gradient-based optimization.

Zhao et al., [Zhao et al., 2017] describe how previous research in the area of neural network has focused on the architectures of the networks and the training convergence. They term the loss layer as the effective driver of the network's learning, but state that the loss "has attracted little attention within the image processing research community". In the article the well-known limitations of l_2 are described, the most important being that l_2 "does not correlate well with human's perception of image quality". A comparison of how the same architecture, trained with different metrics such as l_2, l_1, SSIM and MS-SSIM, performs for image restoration is made. To evaluate the results, the authors look at several image quality metrics and compare "the average values of these metrics on the testing dataset". What they found for their use case is that l_1 outperforms l_2 due to its convergence properties as l_2 easily gets stuck in local minima. l_1 also performs better than both SSIM and MS-SSIM, but the metric that they found to perform best was a mix of l_1 and MS-SSIM.

Facebook AI Research and NYU Langone Health have together established fastMRI [Zbontar et al., 2018], a research project trying to make MRI scans faster with the use of AI. The field of MR image reconstruction using machine learning is relatively new and rapidly expanding. A large dataset of different MR images was made available to the public, giving everyone with an interest in machine learning a chance to design the best model to reconstruct adequate MR images from under-sampled data. MRI reconstruction differs from many computer vision problems when deciding how accurate the reconstructed image has to be. As MRI is used to make diagnoses and deciding on treatment for patients it is essential that small texture changes are observed contrary to the gaming industry, for example.

The paper [Zbontar et al., 2018] have two approaches when solving the MRI reconstruction problem; one that is a classical non-machine learning approach and one with deep learning. The authors present results showing that the deep learning approach, using a U-net architecture for the network, outperforms the classical method. In the best circumstances for both methods, the U-net performs 40-50% better than the classical method, in terms of the normalized mean square error (NMSE). The authors further discuss the importance of determining the quality of a MRI reconstruction and mention that the most commonly used evaluation metrics are mean squared error (MSE) and signal-to-noise ratio (SNR) according to MRI reconstruction literature. They bring up the advantages of these metrics, being easy to understand and efficient computational-wise, but also the disadvantages, only evaluating pixels independently and not being able to capture structural distortion. They bring up metrics such as the Structural Similarity index (SSIM) and its successor, the Multiscale Structural Similarity index (MS-SSIM) which takes structural distortion into consideration. They also mention recent research that measures the perceptual quality of an image using pretrained deep neural networks. The authors wishes that the release of the dataset will trigger the research on MRI reconstruction metrics.

One can notice that the mentioned metrics in above work rank differently according to different authors. It is therefore worth mentioning that a metric can deliver varying results depending on how the problem it is set to solve is constructed. One metric could be designed to put more emphasis on color shifts, whereas another metric could be developed to focus on edge detection. Furthermore, the term "perceptuality" is widely used in mentioned papers and the opinions on which metrics that truly fulfills their claimed perceptuality are strong and seemingly diverging.

1.3 Problem Description

Given the new perceptual metric, FLIP [Anonymous, 2020], how will the resulting model perform in comparison to the established metrics, l_1, l_2, MS-SSIM and LPIPS, according to set criteria? When using these metrics as loss functions, with addition to General & Adaptive Robust Loss, which one is most suitable for the different tasks; denoising and reconstructing of MRI?

Furthermore, how do we perform a proper comparison between these loss functions? Usually when eval-

uating a network architecture one uses the metric to calculate an accuracy for the model. In this case we want to use a well known architecture, and change the loss function used in the training. But this setup makes the comparison and evaluation of the models more tricky. How do we evaluate the evaluation?

2 | Deep Learning

Deep learning is a broad expression that covers machine learning with multiple layers, hidden from the user, based on artificial neural networks. Here we will explain the relevant theory needed to understand our thesis. That is, the overall concept of deep learning using artificial neural networks based on convolutional neural networks.

2.1 Artificial Neural Network

The construction of artificial neural networks (ANNs) resembles, to some extent, the construction of the human brain. The network consists of artificial neurons, nodes, which pass information between each other through shared connections within the network. The neurons contain adjustable parameters, often referred to as weights, which determine the behaviour of the model. To train a neural network is to tune these weights to minimize a given loss function which is further discussed in section 2.4.

2.1.1 The perceptron

The simplest form of an ANN is the perceptron. The mathematical representation is

$$y = \psi\left(\sum_{i=1}^{n} x_i w_i + b\right), \tag{2.1}$$

where the input $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ is a n sized vector, which is weighted to a sum with the weights $\mathbf{w} = \{w_1, w_2, ..., w_n\}$ and put through an activation function ψ before the output y is obtained. The network also contains bias, b, which enables additionally adjustment of the model, as nodes may have different individual constant values [Goodfellow et al., 2016]. The outline of a perceptron is visualized in Figure 2.1.

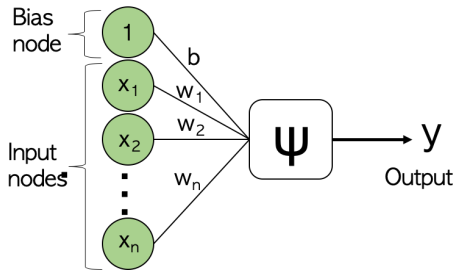


Figure 2.1: The outline of a single perceptron.

2.1.2 Activation function

The use of a non-linear activation function is essential for the network to be able to solve complex, high dimensional, and non-linear problems. The choice of the activation function depends on the problem, e.g., classification or regression. The Rectified Linear Unit function (ReLU) is a widely used activation function that preserves all values above zero and clamps all negative values to zero. The definition of ReLU is

$$\text{ReLU} = \max(0, x), \tag{2.2}$$

where x is the input to a node. The function offers a sparse activation, a better gradient propagation, and efficient computation [Goodfellow et al., 2016].

2.1.3 Multilayer perceptron

To say that an ANN is deep is to say that is has one or more hidden layers in its network. This is also called the multilayer perceptron, that is, connecting layers of perceptrons creating a larger network. In Figure 2.2 the layout of a multilayer perceptron is shown with two hidden layers, each with a bias node [Goodfellow et al., 2016].

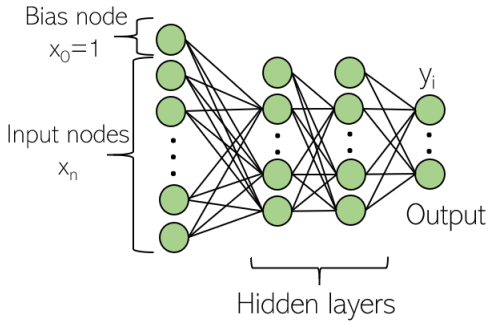


Figure 2.2: The outline of a multilayer perceptron.

2.1.4 Two kinds of learning

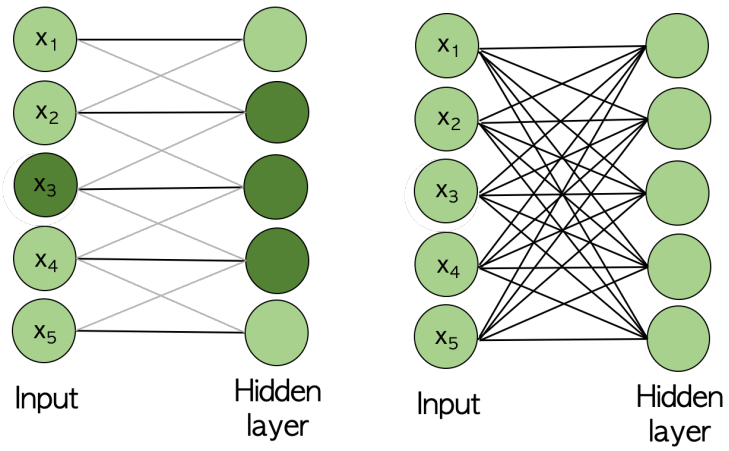
The two most common ways for a network to learn are through either supervised or unsupervised learning. Supervised learning can be compared to when we as humans learn new words. The first time we saw a giraffe, someone probably told us the word for this unknown animal. This approach is basically the same when applying supervised learning for an ANN. The network is faced with some input and its ground truth, i.e., labels that describe the input. The goal is to create a model that maps inputs to outputs. Unsupervised learning on the other hand is, which one can gather from the name, the opposite of supervised learning. Here the network is not presented with the ground truth of the inputs but is instead meant to itself find structures and features of the input [Goodfellow et al., 2016]. In our thesis supervised learning has been used.

2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) is a specific deep learning architecture which employs the linear operation called convolution, instead of general matrix multiplication which is used in standard neural networks. Using a CNN enables the model to capture spatial and temporal dependencies, making it the favorable choice for image analysis. The building blocks of a CNN could be divided into three parts; convolution, activation and pooling. The convolutional stage is when the image is filtered by a convolution operation using a kernel. For each filter, a feature map is generated, where extracted features are stored. The activation stage is when the activation function mentioned earlier is applied. The pooling stage is roughly a summation of neighbouring pixels which result in an image size reduction.

There are three main points which highlight the advantages of CNN; *sparse interactions*, *parameter sharing* and *equivariant representations* [Goodfellow et al., 2016]. The structure of the two first mentioned principles are shown in Figure 2.3. The idea of sparse interactions is that the kernel used is smaller than the input image, making it easier to detect edges and features that occupy only a fraction of the whole image. This can be compared to an ordinary neural network where all the nodes are instead fully connected i.e., every node

in one layer is connected to every node in the next layer. Parameter sharing is used to reduce the number of needed parameters in a network. Specific weights are shared between nodes in the network. In addition to making use of sparse interaction and parameter sharing, layers in a CNN also benefit from a property called equivariance to translation. This attribute entails that if a translation of the input occurs, the output translates equivalent [Goodfellow et al., 2016].



(a) Sparse interactions and parameter sharing between layers.

(b) Fully connected layers.

Figure 2.3: In a) an illustration is shown of the concepts *sparse interactions* and *parameter sharing*. The connections in bold represent that the connected nodes share the parameters (weights). Instead of x_3 being connected to all the hidden nodes, it is sparsely connected to three hidden nodes (darker colored). This can be compared to fully connected layers in b).

2.2.1 Autoencoder

One option for the architecture of a CNN is the autoencoder. An example of this architecture is shown in Figure 2.4. An autoencoder is a network built of two sections with the aim to copy the input to the output while having to significantly reduce the dimensionality of the data at one point along the way. The final reduction is executed in the *bottleneck*. The part on the left side of the bottleneck is the *encoder* that maps the input image to a more compact representation. On the right side of the bottleneck is the *decoder* which transform this downscaled representation back to an image.

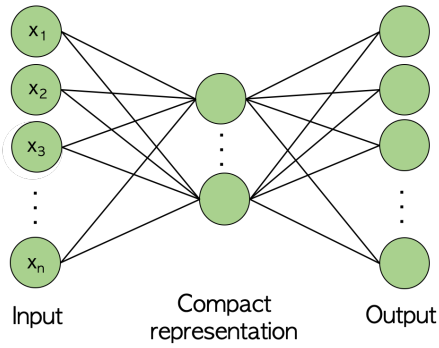


Figure 2.4: The outline of an autoencoder.

A special version of an autoencoder is the U-net, an autoencoder with modifications. In the paper "U-net:

Convolution Network for Biomedical Image Segmentation", Olaf Ronneberger et al., present the U-net architecture. They discovered that using an autoencoder with *skip connections*, a way of sending copies of the crop from layers in the encoder to layers in the decoder, significantly improved the performance of the model [Ronneberger et al., 2015].

In Figure 2.5 the outline of the U-net used in this thesis is presented. We have used ReLU as the activation function, with combination of convolution and max pooling stages. Max pooling is a variant of pooling where the highest value in a small neighbourhood is chosen as output. The kernel size, i.e the size of the filter in the convolution stage, was varied for the different layers.

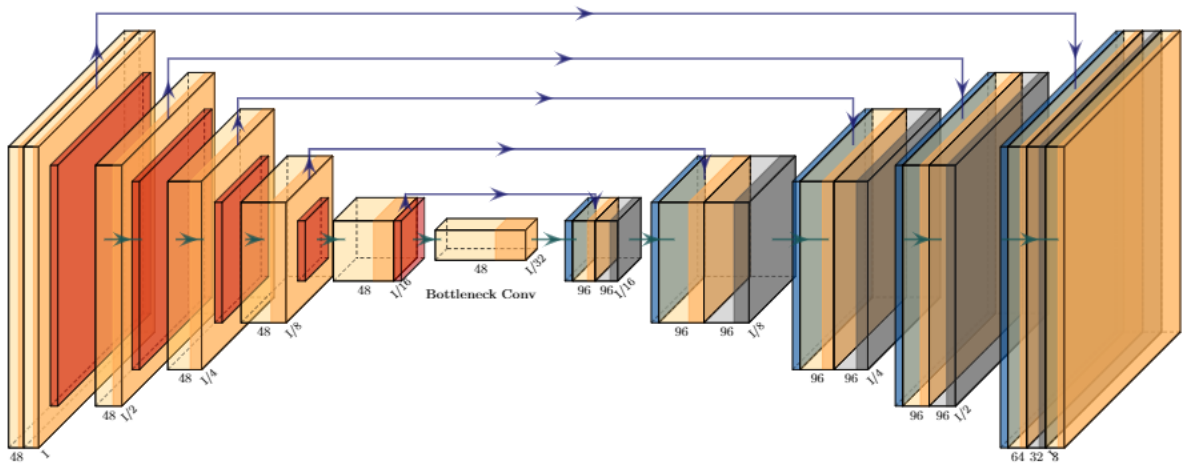


Figure 2.5: The design of the U-net used in this project on an image I. Here a yellow layer denotes the convolution and ReLU stages, a red layer denotes the pooling stage, and a blue layer is up sampling. The arrows indicate the skip connections.

2.3 Learning phase

The ANN that was previously described uses a method called back-propagation, sometimes only referred to as backprop, for calculating the gradient of the loss function. This is the first step for the network to be able to learn, and have some type of memory, being able to present feedback. Feedforward networks, a network structure where information only flows in one direction, from input to output, use this.

The next step for the network in the learning phase is optimization. One of the most commonly used methods is the Stochastic Gradient Descent (SGD). The finesse with SGD is that it optimizes over randomized small batches, a set of images in the dataset, of size P of the training data. This instead of the whole dataset, cutting down on the computational time. The main idea of the method is that it iteratively updates the weights of the network to minimize a given loss function. This is done for every batch. When this has been completed once for every batch, one epoch is fulfilled. The update of the weights is

$$\Delta\omega_k = \frac{1}{P}\sum_{p=1}^{P}\Delta\omega_{pk}, \qquad \text{where} \qquad \Delta\omega_{pk} = -\eta\frac{\partial E_p}{\partial\omega_k}, \tag{2.3}$$

and is based on the gradient of the loss function and in what direction it has its steepest descent thus targeting the weight update a small step in this direction. In said equation it is clear that the size of the step depends on the gradient of the loss function E and a set learning rate η, where the index k represents the iteration of the

update. The aim of the training of a neural network is accordingly to minimize the prediction error that arises.

An extension to the SGD is the Adam optimization algorithm [Goodfellow et al., 2016], also called Adaptive Moment Optimization. Unlike SGD, which maintains a single learning rate η, Adam uses an adaptive one, meaning it computes individual learning rates for different weights. It can be viewed as a combination of SGD with momentum and RMSProp. Momentum is a method to accelerate learning as it uses the gradient of the past gradients to determine the direction opposed to using only the gradient of the current step. For SGD, the size of the step depended on the gradient and the learning rate. For momentum, the size of the step instead depends on how large and how aligned a series of gradients are. Regarding the RMSProp algorithm, the learning rate is adapted based on the average of the recent magnitudes of the gradients for the weight. Adam differ from RMSProp here as it does not only take into account the average of the first moment but also makes use of the average of the second moments of the gradients. In this thesis Adam has been used for the optimization of the networks.

2.4 Loss functions

To find the weights that optimize a network, one often minimizes a given loss function. In the world of machine learning, said function can be called numerous things such as objective function, criterion, cost function, or error function. It is fairly common to separate the function to minimize and the function that measures a variation between a reference and a target.

2.4.1 l_1-norm

When using the l_1-norm, or Least Absolute Deviations as it is also called, as a loss function one minimizes the sum of all the absolute differences between the target value and the output value. This error E is presented as

$$E = \frac{1}{N}\sum_{n=1}^{N} |y_n - d_n|, \tag{2.4}$$

where y_n is the output value for pixel n, d_n the target value for pixel n and N is the number of pixels.

2.4.2 l_2-norm and PSNR

With the l_2-norm, or Least Square Error, one minimizes the sum of all the squared differences between the target value and the output value. This error E is presented as

$$E = \frac{1}{N}\sum_{n=1}^{N} (y_n - d_n)^2, \tag{2.5}$$

where y_n is the output value, d_n the target value and N is the number of pixels.

A loss that is closely related to the l_2-norm is the peak signal-to-noise ratio (PSNR) which is the ratio between the maximum value of a signal and the value of corrupting noise added to a signal. The PSNR value is often expressed with the logarithmic decibel scale to easier interpret the results so these do not vary in dynamic range too much. The loss is expressed as

$$E = 20 \cdot \log_{10}\left(\frac{\mathrm{MAX_I}}{\sqrt{l_2}}\right), \tag{2.6}$$

where MAX_I is the maximum pixel value of an image I and l_2 the calculated l_2-norm.

2.4.3 MS-SSIM

Using MS-SSIM as a loss function is a more complex strategy than previous functions as it aims to be perceptual. Its precursor is the SSIM given by

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^{\alpha} \cdot [c(\mathbf{x}, \mathbf{y})]^{\beta} \cdot [s(\mathbf{x}, \mathbf{y})]^{\gamma}. \tag{2.7}$$

The signals $\mathbf{x}$ and $\mathbf{y}$ are the elements one wants to compare the similarity between, where the signals in our case are images. The parameters $[\alpha, \beta, \gamma] > 0$ are used to adjust the weight of the three components l, c and s. These components are expanded to

$$\begin{aligned} l(\mathbf{x}, \mathbf{y}) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \\ c(\mathbf{x},\mathbf{y}) &= \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \\ s(\mathbf{x},\mathbf{y}) &= \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \end{aligned} \tag{2.8}$$

where l stands for luminance, c for contrast, and s for structure. μ_x and μ_y represent the mean value of $\mathbf{x}$ and $\mathbf{y}$, respectively, and σ_x and σ_y the variances of the signals. σ_{xy} is the covariance of $\mathbf{x}$ and $\mathbf{y}$. For the constants C_1, C_2, C_3, the relation $C_1 = (K_1L)^2$, $C_2 = (K_2L)^2$ and $C_3 = C_2/2$ stands. Here K_1, $K_2 << 1$ are scalar constants and L is the dynamic range of the pixel values. As proposed by the authors, the values for the constants were set to $K_1 = 0.01$ and $K_2 = 0.03$. A sliding-window approach is used when applying the algorithm, and the calculated SSIM indexes are stored in a quality map. When training an ANN, the loss needs to be assembled to one value. For this the mean of the SSIM values is used. The loss function MS-SSIM, i.e., the expansion of SSIM, introduces a scaling factor M in addition to declared equations. Here the original image is indexed as Scale 1 and the highest scale as Scale M. The definite MS-SSIM value is achieved by combining the measurement at different scales for

$$\text{MS-SSIM}(\mathbf{x},\mathbf{y}) = [l_M(\mathbf{x},\mathbf{y})]^{\alpha_M} \prod_{j=1}^{M} [c_j(\mathbf{x},\mathbf{y})]^{\beta_j} [s_j(\mathbf{x},\mathbf{y})]^{\gamma_j}. \tag{2.9}$$

The loss function for MS-SSIM finally becomes $E = 1-$MS-SSIM as MS-SSIM is a measure for similarity, and we want to minimize the dissimilarity [Wang et al., 2003].

2.4.4 LPIPS

The idea behind the LPIPS metric is using so called deep features when constructing a loss function. The metric comes with different possible variants, such as *lin*, *tune* and *scratch*. For *lin*, pre-trained network weights $\mathcal{F}$ are fixed, but linear weights w are learnt. For *tune*, a pre-trained classification model is used and all the weights for network $\mathcal{F}$ are tweaked. For the final variation, *scratch*, a network is initialized from random Gaussian weights and trained using judgment from linked studies [Zhang et al., 2018].

The approach consists of two parts. The first part is to calculate a distance and the second part is to predict perceptual judgment to finally end up with a loss function. The calculation of a distance between a reference patch x and a distorted patch x_0 with a given network $\mathcal{F}$ is calculated according to

$$d\left(x, x_0\right) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \left\| w_l \cdot \left(\hat{y}^l_{hw} - \hat{y}^l_{0hw}\right) \right\|_2^2, \tag{2.10}$$

where $\hat{y}^l$ and $\hat{y}^l_0$ are extracted deep embeddings from layer l, $\mathbf{w_l}$ is a scaling vector and H, W the spatial components [Zhang et al., 2018].

The used loss function culminates to

$$\mathcal{L}(x, x_0, x_1, h) = -h \log \mathcal{G}(d(x, x_0), d(x, x_1)) - (1 - h) \log(1 - \mathcal{G}(d(x, x_0), d(x, x_1))), \tag{2.11}$$

where d_0 and d_1 stand for two distances between patches x, x_0 and x, x_1 respectively, $\mathcal{G}$ is a small trained network and h the predicted perceptual judgment [Zhang et al., 2018].

Kaplanyan et al., present [Kaplanyan et al., 2019] a weighted version of LPIPS with two other loss functions to try and cover as many properties as possible. The authors of LPIPS mention that the appearance of a grid is a common issue with their loss function, which motivated us to use the same approach as Kaplanyan et al., As l_2 is a well established loss function, this was chosen to be weighted with LPIPS. To clarify, we will use the weighted loss function LPIPS+l_2 for training a network, and the metric LPIPS when evalutating different models.

2.4.5 A General and Adaptive Robust Loss Function

The General & Adaptive Robust loss function [Barron, 2017] is a generalization of several already existing loss functions. A simple form of the loss function is

$$f(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha} \left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1 \right)^{\alpha/2} - 1 \right), \tag{2.12}$$

where x is the difference in pixel value, α is a shape parameter which controls the robustness of the loss and $c > 0$ is a scale parameter which enables adjustment of the loss's quadratic bowl when near $x = 0$. The final loss function, with the special cases $\alpha = 0, 2, -\infty$, becomes

$$\rho(x, \alpha, c) = \begin{cases} \frac{1}{2}(x/c)^2 & \text{if } \alpha = 2 \\ \log\left(\frac{1}{2}(x/c)^2 + 1\right) & \text{if } \alpha = 0 \\ 1 - \exp\left(-\frac{1}{2}(x/c)^2\right) & \text{if } \alpha = -\infty \\ \frac{|\alpha-2|}{\alpha}\left(\left(\frac{(x/c)^2}{|\alpha-2|} + 1\right)^{a/2} - 1\right) & \text{otherwise.} \end{cases} \tag{2.13}$$

When $\alpha = 2$ the loss function is similar to l_2 loss, and for $\alpha = 1$ it appears as a smoothed variant of l_1 loss. This loss is then used in constructing a probability density function which reads as

$$p(x|\mu, \alpha, c) = \frac{1}{cZ(\alpha)} \exp(-\rho(x - \mu, \alpha, c)) \qquad \text{where} \qquad Z(\alpha) = \int_{-\infty}^{\infty} \exp(-\rho(x, \alpha, 1)). \tag{2.14}$$

The loss function used for training an ANN is the negative log-likelihood of this probability density function [Barron, 2017]. In this thesis this loss function is referred to as Adaptive.

3 | FLIP: An Image Difference Evaluator

FLIP is a new image quality metric that researchers at NVIDIA are currently developing [Anonymous, 2020]. The goal of the metric is to achieve a per-pixel error map that is perceptually correct. One important part of FLIP is that it is based on an alternating comparison.

3.1 FLIP for RGB images

To calculate the FLIP error map, the first step is a preprocessing of both the reference and the test image. This is done taking the viewing distance and the display's properties into account to remove details too small to notice. The preprocessing consists of spatial filtering, and to do this the images must be in the correct color space. Therefore color transformations are used back and forth between the steps of calculating the error map.

After the preprocessing, the color difference, ΔE_c, is calculated, taking the Hunt effect into account. The Hunt effect means that brighter color differences are more noticeable than darker. This color error is the first part of the final error [Anonymous, 2020].

The second part consists of the feature error i.e., the edge and point dissimilarities. Also here the algorithm takes the viewing and display's conditions into consideration. To detect the edges and points, two convolutions are made with custom kernels. The feature difference, ΔE_f, is then calculated to be the maximum of the differences [Anonymous, 2020].

Finally the error map is computed by enhancing the color difference with the feature difference as

$$\mathrm{FLIP}_{\Delta \mathrm{E}} = (\Delta E_c)^{1-\Delta E_f}. \tag{3.1}$$

When using the FLIP error map as a loss function for machine learning, in this thesis, we calculated the mean of the error map. Since it is the same model evaluated by multiple people, we have not been able to customize the viewing and screen parameters. We trained the network with a viewing distance of 70 cm and with the properties of our displays.

3.2 FLIP for grayscale images

The FLIP metric was essentially developed for sRGB images [Anonymous, 2020]. Therefore, we have modified the metric to better suit grayscale images. First of all, the MR images were not in sRGB but in linear RGB so the color transformations between sRGB and linear RGB were no longer necessary. Furthermore, the color transformations used to account for the Hunt effect were also unnecessary, since these leave the luminance component untouched.

It is also worth mentioning that our grayscale version of FLIP is not based on the referenced FLIP version but on a slightly older one. Therefore our version lacks the point detection part and it is reasonable that

a newer version would result in slightly different results. But since we have not looked at rendered images, in which case it is more important with point comparison [Anonymous, 2020], it is viable to use the version that we did.

4 | Dataset

This thesis consists of two main tasks for the different models of ANN to solve. The first is denoising of color images and the second is reconstruction of MR images. For these tasks different datasets were used.

4.1 Denoising problem

A subset of 9000 images from the ImageNet dataset[1] with added Gaussian noise was used as training data. As validation set the Kodak image dataset[2] was used. Both these RGB datasets consist of various images of ordinary day to day life, eg., animals, boats, and buildings.

4.2 MRI Reconstruction problem

The fastMRI dataset is a large dataset containing MR images of both knees and brains[3]. For the knee dataset, data using both single- and multi-coils are available. For the brain dataset, only data using multi-coil is available.

The single-coil dataset consists of MR images when a single receiver coil has been used. This is an older method but single-coil machines are still in use in some hospitals. The multi-coil dataset, on the other hand, consists of images where multiple receiver coils has been used parallel, where each coil in highly sensitive in a specific area. To create one image out of this multi-coil-data the root-sum-of-squares is computed.

The images are predivided into training and validation sets. In this thesis the training dataset for the single-coil knee images was used for training and the validation set was used for the validation. For the multi-coil brain dataset the validation set was considered large enough and used for the training phase. Then a validation set was created with a subset from the initial training set available. These images all work as the ground truth.

When accelerating the MR process, artifacts are introduced and, to simulate this, a masking function that generates a random mask is provided along with the dataset. This masking is similar to artifacts that could arise in real life when using MRI. The acceleration factor can be set to either four or eight. In our experiments, the acceleration was set to eight. These random masks were used to create the input data. For every new pair of data a new random mask was generated [Zbontar et al., 2018].

[1] http://www.image-net.org/
[2] http://r0k.us/graphics/kodak/
[3] https://fastmri.org/dataset

5 | Method

Since deep learning is a broad field of research the first step was to navigate in this, choosing what metrics we wanted to test and compare the new FLIP metric to. The ones we chose, and are described in section 2.4, are either well known and commonly used metrics, or well performing and relevant metrics. The next step was to narrow down and decide which applications of deep learning our thesis should cover. Denoising was a relevant application since it is a well known area where a lot of research is made and it is a typical problem solved by deep learning. MRI reconstruction is also a well suited problem solved with deep learning and relevant research was available here as well.

The datasets we have used are big, easy to access, and have a given ground truth. The ImageNet and Kodak dataset are used frequently in research. The fastMRI dataset is within an interesting field that is not yet deeply explored in the deep learning research field. The combination of these applications and datasets made an interesting ground for our thesis.

5.1 Experiment setup

A U-net, such as the one described in Section 2.2.1, was used. The Python library for machine learning, PyTorch, was used throughout this project. The described loss functions were implemented to fit our architecture. Worth mentioning is that the network architecture AlexNet was used for LPIPS + l_2, with the variation *lin*, as this were the proposed combination by the authors. Further on, the code for the FLIP error map was translated from `MATLAB` to PyTorch and adjusted to work as a loss function and metric for machine learning. A version of FLIP for grayscale images was also implemented, see section 3.2. The decision of which learning rate to use for the different loss functions was based on evaluating the convergence rate of the training and validation loss. The rate was then adjusted for each loss function to achieve the best denoising or MRI reconstruction.

5.1.1 Denoising

For the denoising problem the training data was randomly cropped to 256×256 pixels. Augmentation was applied, a method where we, for example, rotate the images and/or flip the colors channels to achieve larger variations in dataset. Random Gaussian noise was then added to the input images, with a standard deviation $\sigma \in [0, 40]$. The network trained for 1000 epochs with all parameters constant except for the learning rate which was changed depending on which loss function was used.

5.1.2 MRI Reconstruction

For the MRI reconstruction problem, the training data was not randomly cropped, since the model should learn a whole physiological structure, but instead cropped to fit the network architecture. A random mask was applied on the input data [Zbontar et al., 2018]. The network trained for 1000 epochs for the knee images and for 500 epochs for the brain images. This was because the brain dataset is a lot larger and requires much computational power. We made sure that the network had converged in both cases. All parameters were constant except for the learning rate which was changed depending on which loss function that was used.

5.1.3 User Study

The observers were presented with pairwise comparison of images flipping between the denoised or reconstructed one and the reference image for the different models. The images were presented with the same resolution as the training had used, and there was no zooming option, since the question at issue is about perception at a given range. Which two loss functions that were used in each test scenario were randomized and not indicated to the user. The task was to chose which of the two image flips shows the highest similarity between the reference and test image. For the denoising problem, eight different images from the validation dataset were used. These are presented in Figure 5.1. The user study for the denoising problem thus consisted of 120 images in total. For the MRI reconstruction problem, four different images were used, resulting in 60 comparisons in total. These four images are shown in Figure 6.10, 6.11, 6.14 and 6.15.

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.1: The reference images that were used in the user study for denoising.

5.2 Evaluation

As mentioned in Section 1, an common approach when evaluating something in machine learning is to compare the accuracy based on the loss function.

The goal when training an ANN is to minimize the error function, and when we change how the error is computed, by switching loss function, the training will change accordingly. The choice of error function strongly correlate with the accuracy. Since the loss function now is a biased parameter we can no longer only use the accuracy values in the evaluation. This leaves the question of how to evaluate the models obtained by altering the loss functions somewhat difficult to answer.

To tackle this problem of evaluation we used three different approaches. First we calculated, for every model, for a set of metrics a mean over the validation datasets, similar to previous approaches [Zhao et al., 2017]. We also carefully observed the images, zooming in and out, flipping between different models and the reference images and documented our experiences. Lastly, we performed a user study comparing the results from the different loss functions. A discussion based on the measurements, our observations, and the user studies is provided in Section 7.

5.2.1 Measurements

When creating our measurements of the models performance, inspiration was taken from [Zhao et al., 2017], where they look at the metrics l_1, l_2, PSNR, SSIM, MS-SSIM, and a few other variations of SSIM. In our evaluation, we chose to use l_1, l_2 and PSNR since these are commonly used and familiar. In addition to these, we used MS-SSIM since this is a metric that tells more about the overall structure of the image. We also

used LPIPS as a metric out of curiosity for its behavior. Since a big task in this thesis was to compare FLIP to the other metrics it was a given choice to see how the models performed according to FLIP. Adaptive, on the other hand, was not considered to contribute with any further understanding of the models since both l_1 and l_2 already covers it.

5.2.2 Our own observations

To evaluate the resulting image quality, we first needed to establish some criteria for what we were looking for. The first criterion of quality should reasonably be that, in the case of denoising, the noise disappears. In the case of MRI reconstruction, the resulting image should be as similar to the original image as possible. The second criterion is that the colors and the structure of the components in the image should be preserved, e.g., a person's skin or the facade of a house. The third criterion is that the actual details should be preserved as good as possible, for example if an existing window is visible or not, and that no made up artifacts appear. These three criteria form the basis of our discussion on our own observations. To concretize this discussion, we gathered our findings with a score system reaching from 0 to 3.

5.2.3 User Study

To evaluate the results from the user studies, the methods of [Perez-Ortiz and Mantiuk, 2017] were used. In their git repository[1] they provide a set of `MATLAB` functions to analyze the results of a pairwise comparison experiments for images. To summarize the results, the probability of choosing one model over another was calculated in the same way as [Perez-Ortiz and Mantiuk, 2017] did.

The authors use a score system named JOD, just-objectionable-differences, where 1 JOD in difference relative to another condition indicates that 75% of observers selected one condition as better than the other. Using bootstrap methods they analyze the results to clarify which quality scores that are statistically significant and which are not. This method of using JODs and bootstrapping was used to interpret the results generating one plot of the scores and the uncertainties and one plot with a ranking triangle showing the relative distances in score between all neighbouring models.

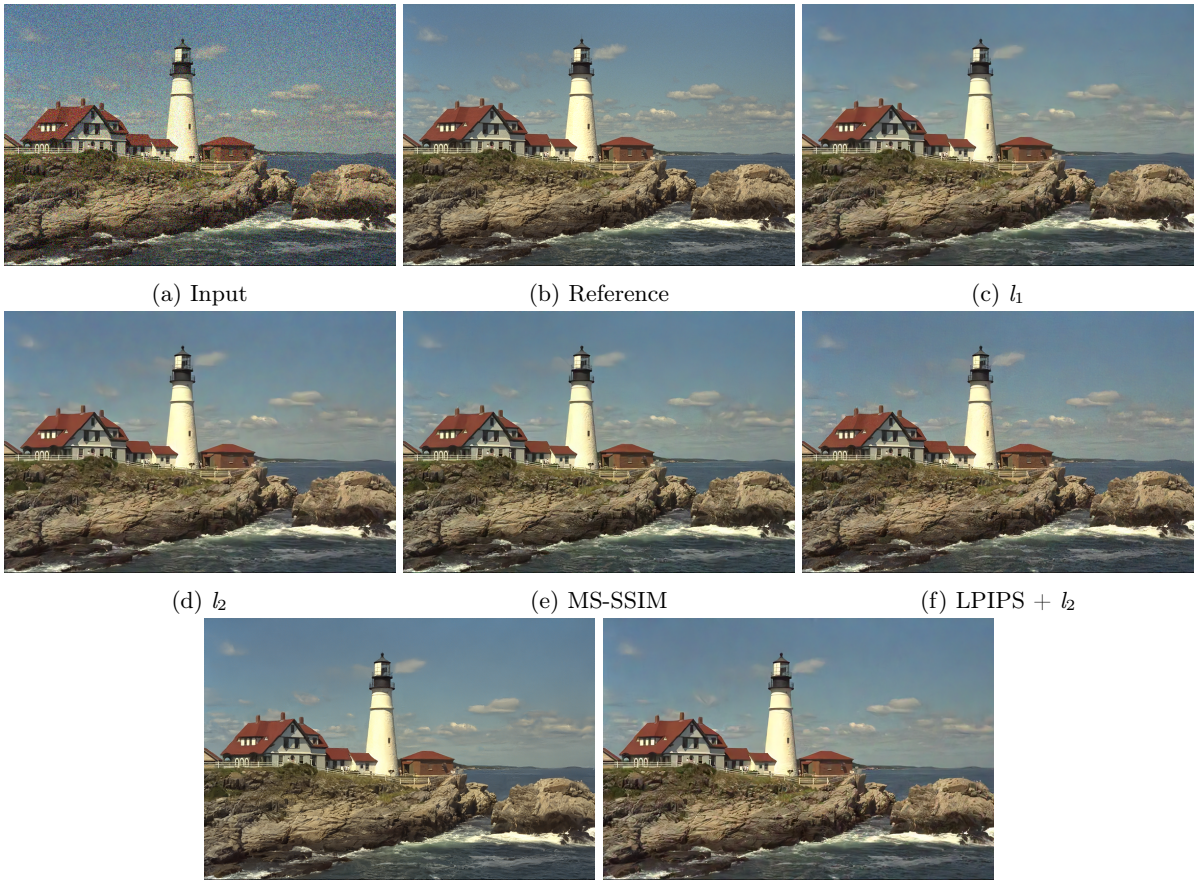
[1] https://github.com/mantiuk/pwcmp

6 Results

In this section all of our results are collected, divided into the two problems. Test images are presented from the denoising and MRI reconstruction problems, together with recorded values of each metric for each model. In addition to this, the results from the user study for the two problems are presented.

6.1 Denoising

For the denoising problem, the Kodak dataset was used as validation of the completed model trained with the different loss functions. From the Kodak dataset, we then chose some images showing characteristics of the models. These images are presented below in Figure 6.1, 6.3 and 6.5. A zoomed-in crop from each image is also presented to simplify for the reader in Figure 6.2, 6.4 and 6.6. In Table 6.1, measured values of each metric for each model is shown. Finally, the results from the user study are displayed in Table 6.2, Figure 6.7, and Figure 6.8.

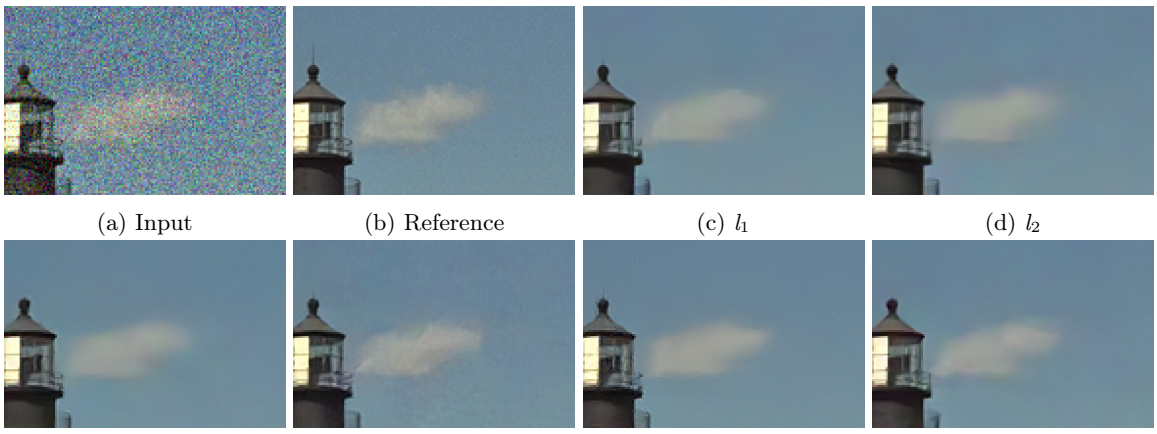


(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

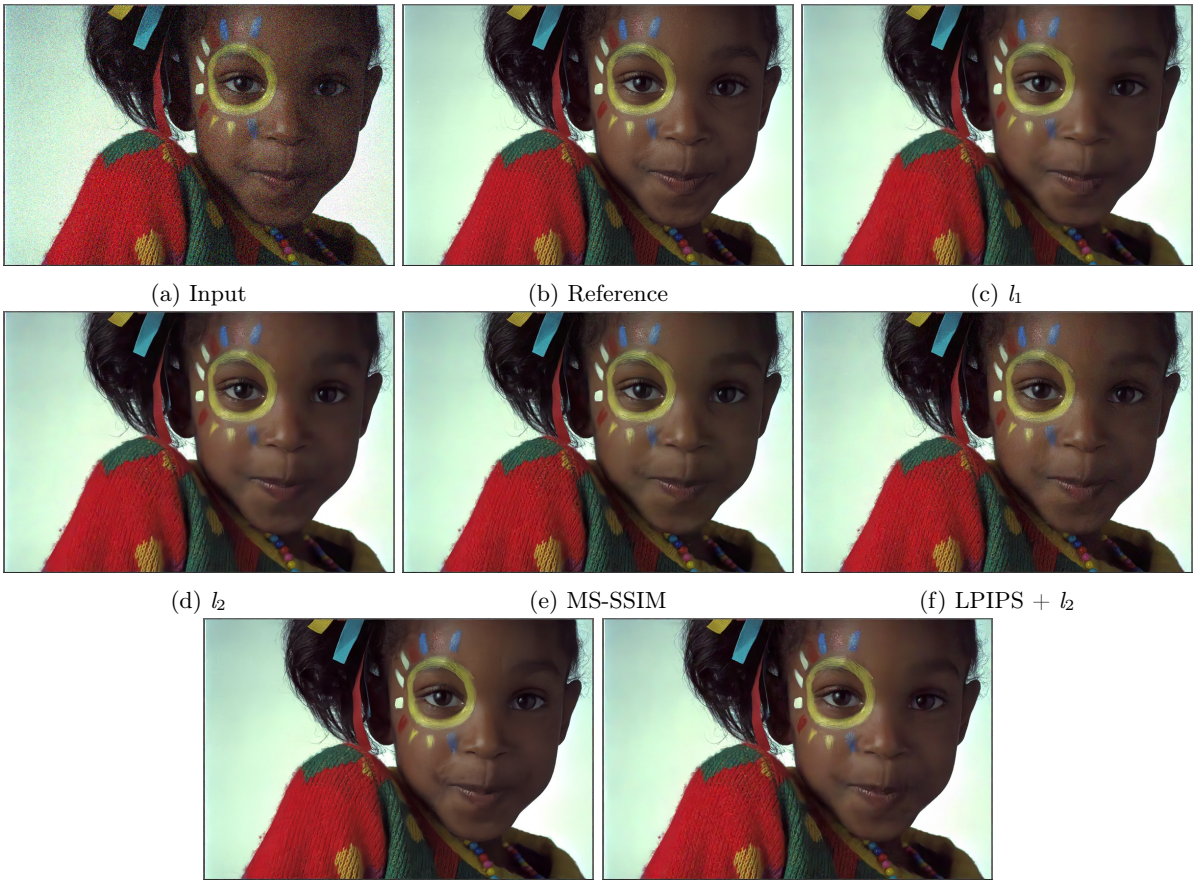
Figure 6.1: Input image, reference image and test images for the six loss functions stated in the captions when training on the ImageNet dataset.



(a) Input (b) Reference (c) l_1 (d) l_2

(e) MS-SSIM (f) LPIPS + l_2 (g) FLIP (h) Adaptive

Figure 6.2: Zoomed-in crop of Figure 6.1 of input image, reference image, and test images for the six loss functions stated in the captions when training on the ImageNet dataset.

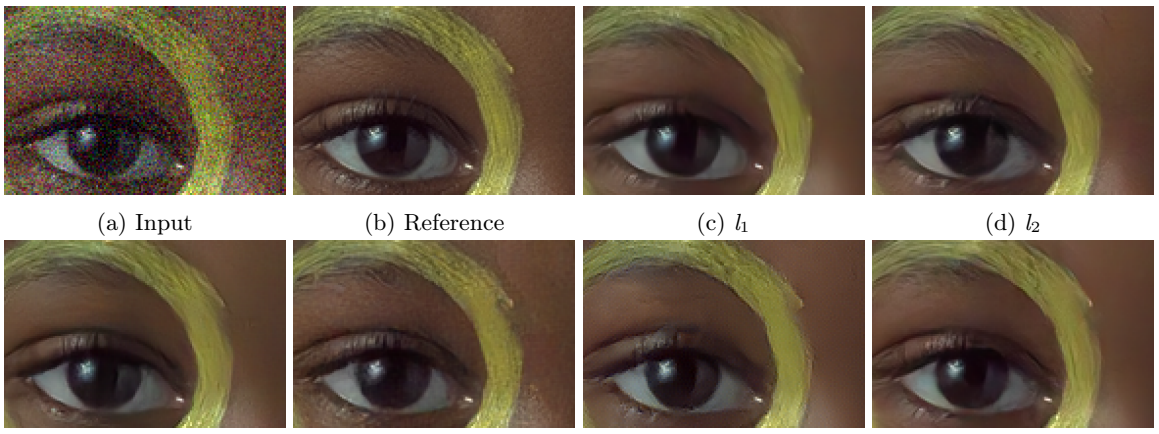


(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

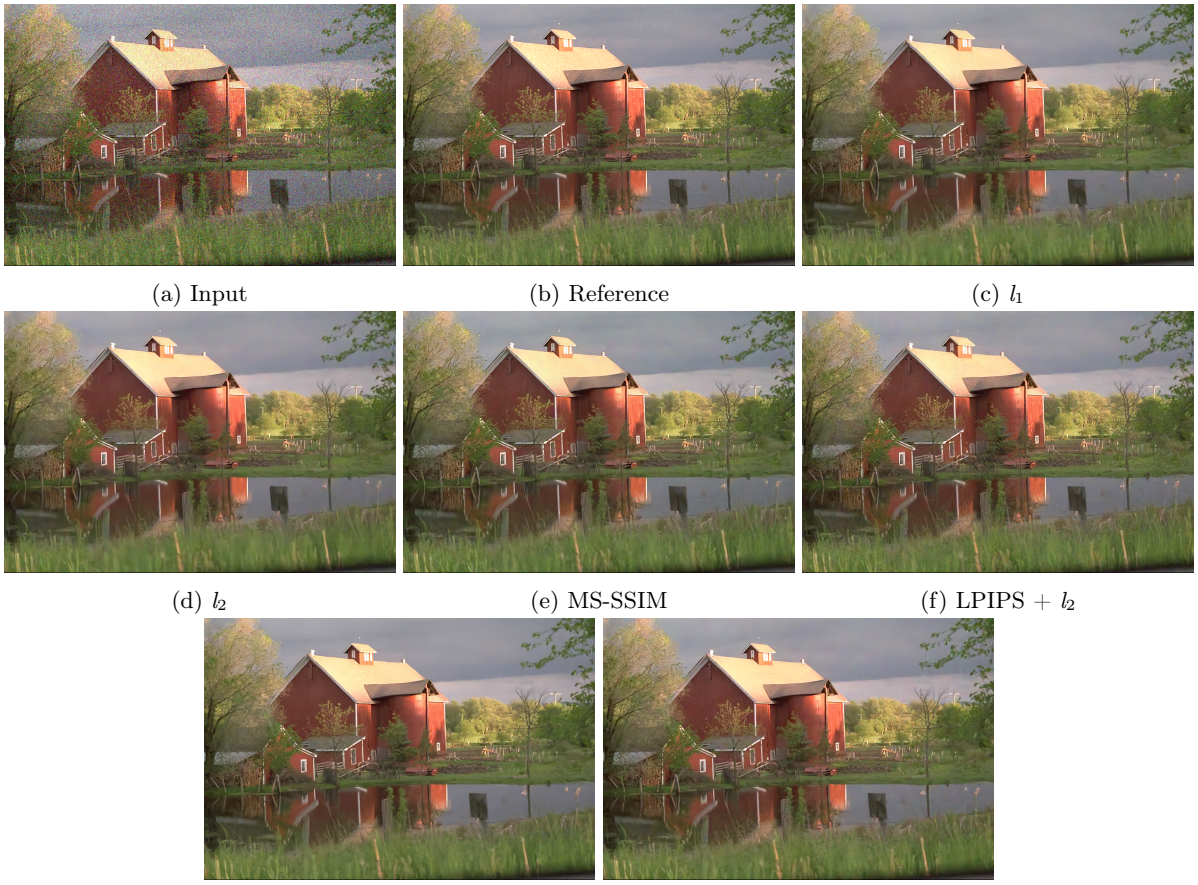
Figure 6.3: Input image, reference image, and test images for the six loss functions stated in the captions when training on the ImageNet dataset.



(a) Input (b) Reference (c) l_1 (d) l_2

(e) MS-SSIM (f) LPIPS + l_2 (g) FLIP (h) Adaptive

Figure 6.4: Zoomed-in crop of Figure 6.3 of input image, reference image, and test images for the six loss functions stated in the captions when training on the ImageNet dataset.

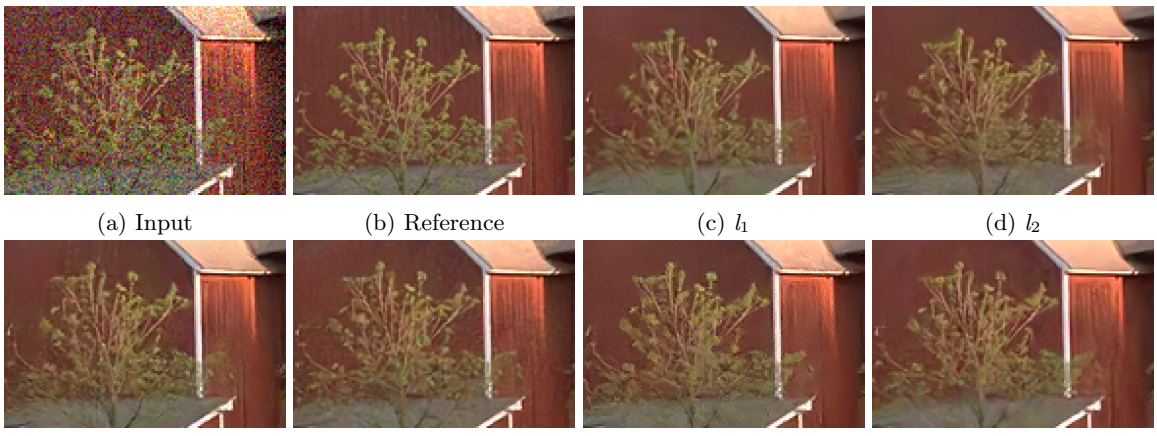


(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

Figure 6.5: Input image, reference image, and test images for the six loss functions stated in the captions when training on the ImageNet dataset.



(a) Input (b) Reference (c) l_1 (d) l_2

(e) MS-SSIM (f) LPIPS + l_2 (g) FLIP (h) Adaptive

Figure 6.6: Zoom-in crop of Figure 6.5 of input image, reference image, and test images for the six loss functions stated in the captions when training on the ImageNet dataset.

6.1.1 Measurements

In Table 6.1, the recorded measurements for our models for a set of metrics are shown.

Table 6.1: Recorded values when training with identical architecture on the ImageNet dataset, where the loss function differs between models. For each model, the calculated value of each metric was registered. For PSNR and MS-SSIM, a higher value represents a better reconstruction. For l_1, l_2, LPIPS and FLIP, a lower value represent a better reconstruction. For each model, the metric that received the best score is made bold.

Metric \ Model	Noisy	l_1	l_2	MS-SSIM	LPIPS + l_2	FLIP	Adaptive
$1000 \cdot l_1$	78.22	24.52	**18.50**	19.30	19.02	20.87	18.60
$1000 \cdot l_2$	9.61	1.11	**0.68**	0.74	0.70	0.86	0.69
PSNR	20.17	29.81	**32.05**	31.69	31.89	31.01	31.97
MS-SSIM	0.8930	0.9614	0.9805	**0.9814**	0.9801	0.9808	0.9810
LPIPS	0.4593	0.3182	0.2274	0.2096	**0.0500**	0.2083	0.2235
FLIP	0.4593	0.1611	0.1207	0.1246	0.1247	**0.1121**	0.1221

6.1.2 User study

Here follows the results of the user study where 14 observers participated. The participants consisted of researchers at NVIDIA, other students from LTH, and our families.

Table 6.2: The probability to chose one loss function over another. For each comparison, the winner is made bold. If a comparison results in a tie, no one is made bold.

Loser \ Winner	l_1	l_2	MS-SSIM	LPIPS + l_2	FLIP	Adaptive
l_1	0	0.3214	**0.5179**	**0.7232**	**0.5893**	**0.5089**
l_2	**0.6786**	0	**0.7054**	**0.7768**	**0.6875**	**0.6161**
MS-SSIM	0.4821	0.2946	0	**0.6696**	0.4643	0.4554
LPIPS + l_2	0.2768	0.2232	0.3304	0	0.3393	0.2500
FLIP	0.4107	0.3125	**0.5357**	**0.6607**	0	0.4554
Adaptive	0.4911	0.3839	**0.5446**	**0.7500**	**0.5446**	0

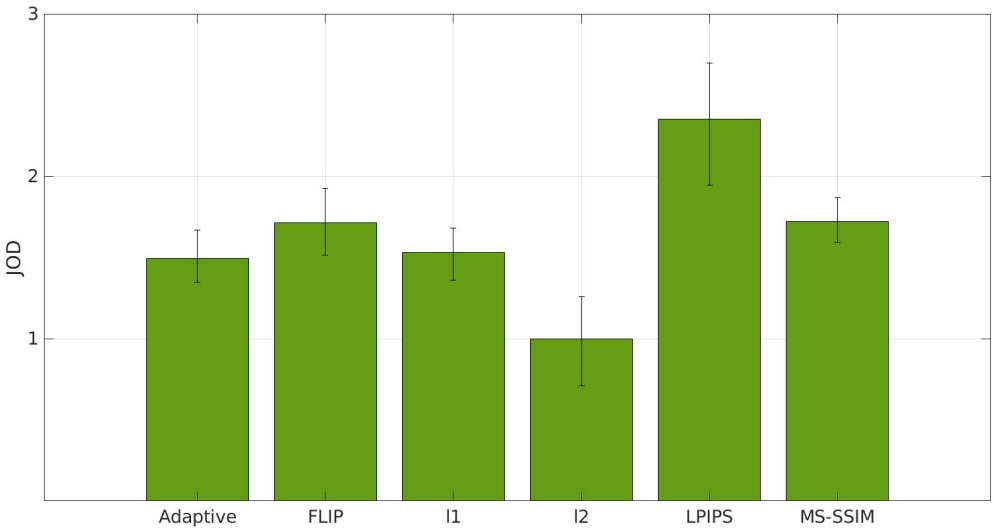


Figure 6.7: Visualization of the scaling results and 95% confidence intervals for the six loss function on the ImageNet dataset. The difference of 1 JOD indicates that 75% of observers selected one loss function as better than the other. The model with the lowest score is scaled to 1. The weighted loss LPIPS + l_2 has been used where it says "LPIPS".

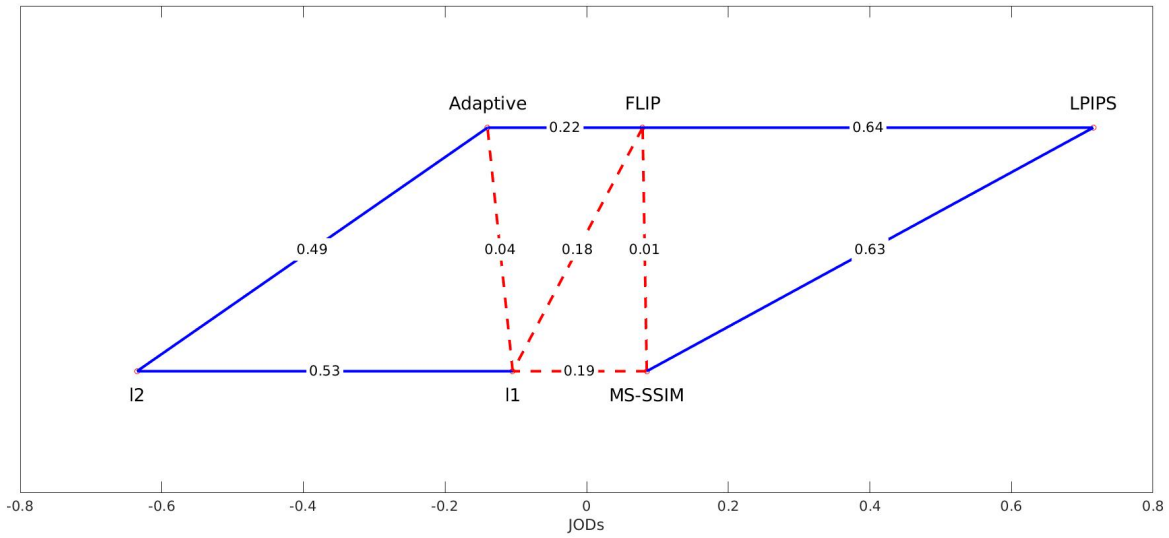
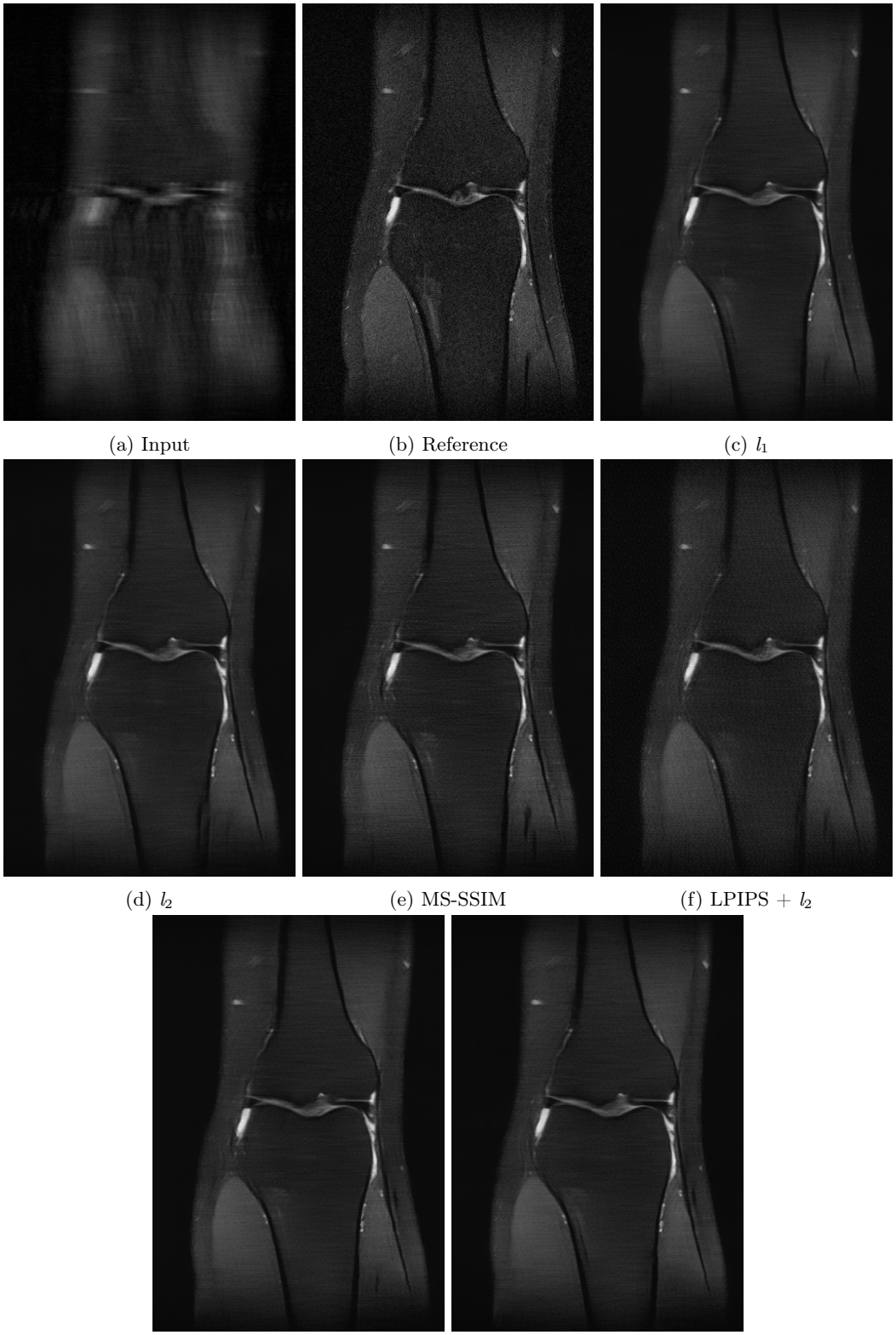


Figure 6.8: Graphical representation of the scaling for the six loss function on the ImageNet dataset. Each loss function is only connected to its close neighbours as this is what we are most interested in. Blue solid lines represent statistically significant differences and red dashed lines represent statistically insignificant differences. The x-axis shows the scaling in JODs. The weighted loss LPIPS + l_2 has been used where it says "LPIPS".

6.2 MRI Reconstruction

For the MRI reconstruction of the knee, the validation dataset from fastMRI for single-coil knee was used as validation of the completed model trained with the different loss functions. From this output, some images were chosen showing interesting aspects of the models. These images are presented below in Figures 6.9, 6.10, and 6.11. A zoomed-in crop from Figure 6.11 is also presented to simplify for the reader in Figure 6.12. A similar approach was made for the MRI reconstruction of the brain, apart from validating on the validation dataset from fastMRI for multi-coil brain. The chosen images for the output from this MRI reconstruction are presented in Figures 6.13, 6.14, and 6.15. A zoomed-in crop from Figure 6.15 is also included in Figure 6.16. In Table 6.3 for knee and 6.4 for brain, the measured values of each metric and for each model are shown. Finally, the results from the user study are displayed in Table 6.5, Figure 6.17 and Figure 6.18.

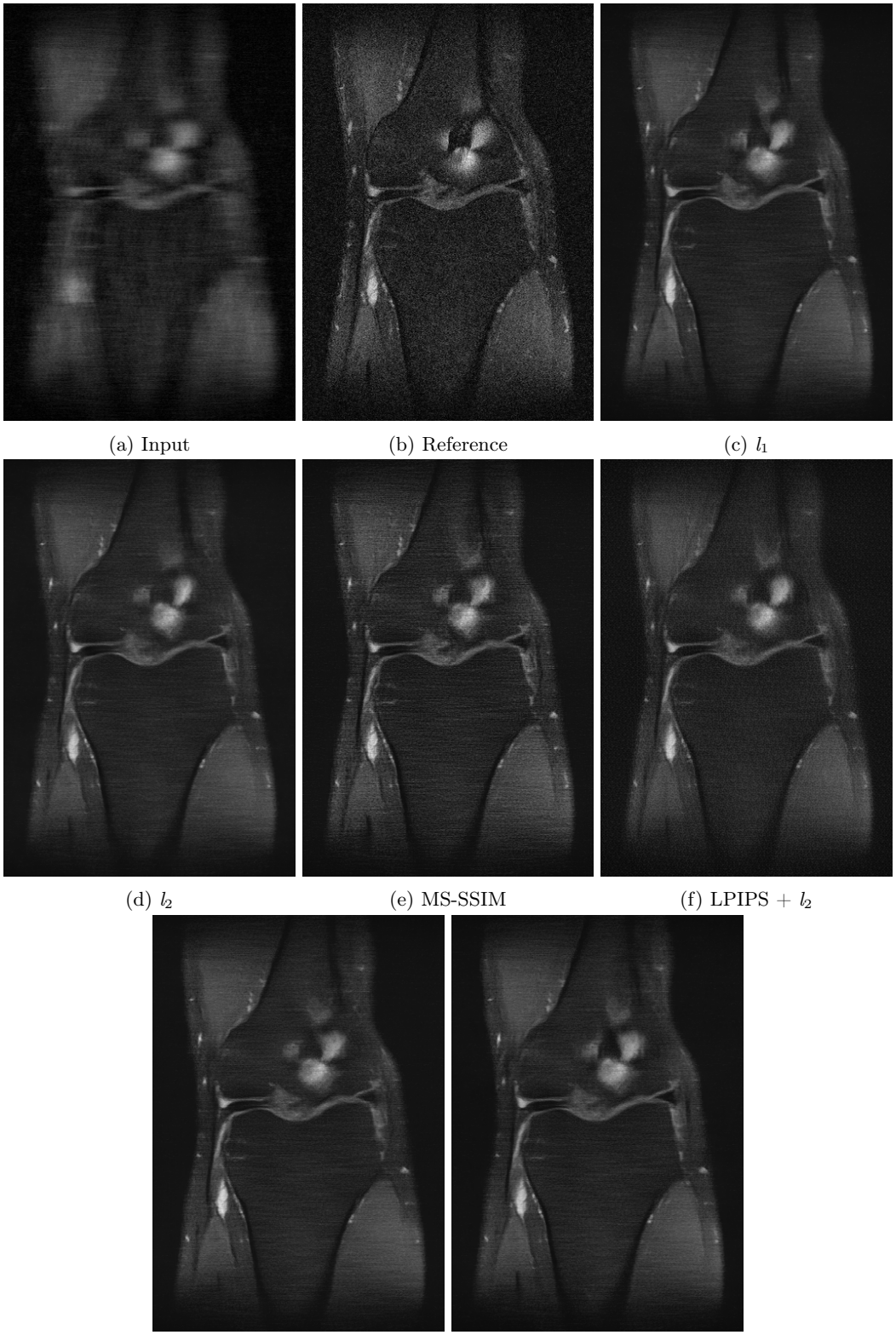


(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

Figure 6.9: Input image, reference image, and test images for the six loss functions stated in the captions when training on the fastMRI dataset for knee.

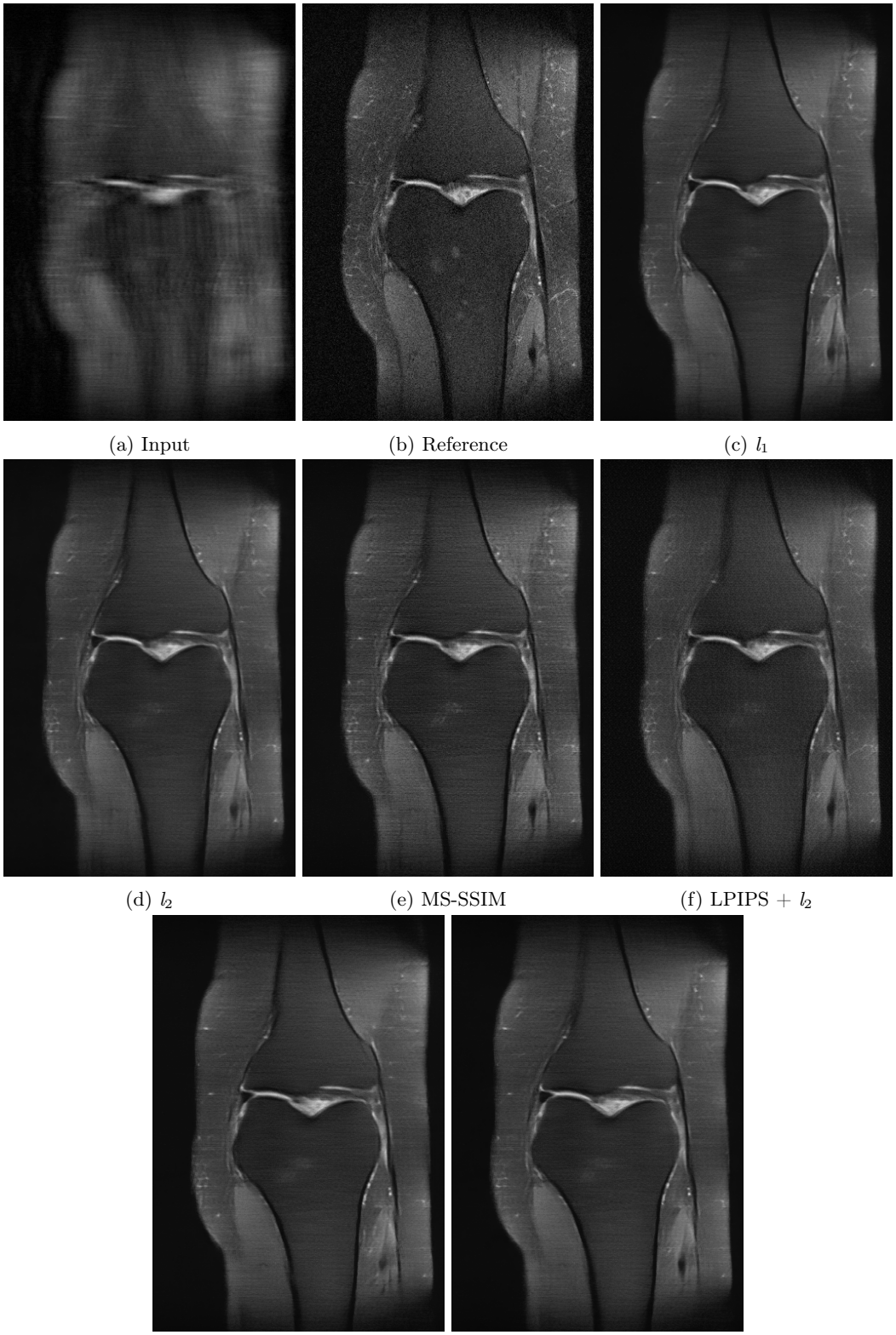


(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

Figure 6.10: Input image, reference image, and test images for the six loss functions stated in the captions when training on the fastMRI dataset for knee.

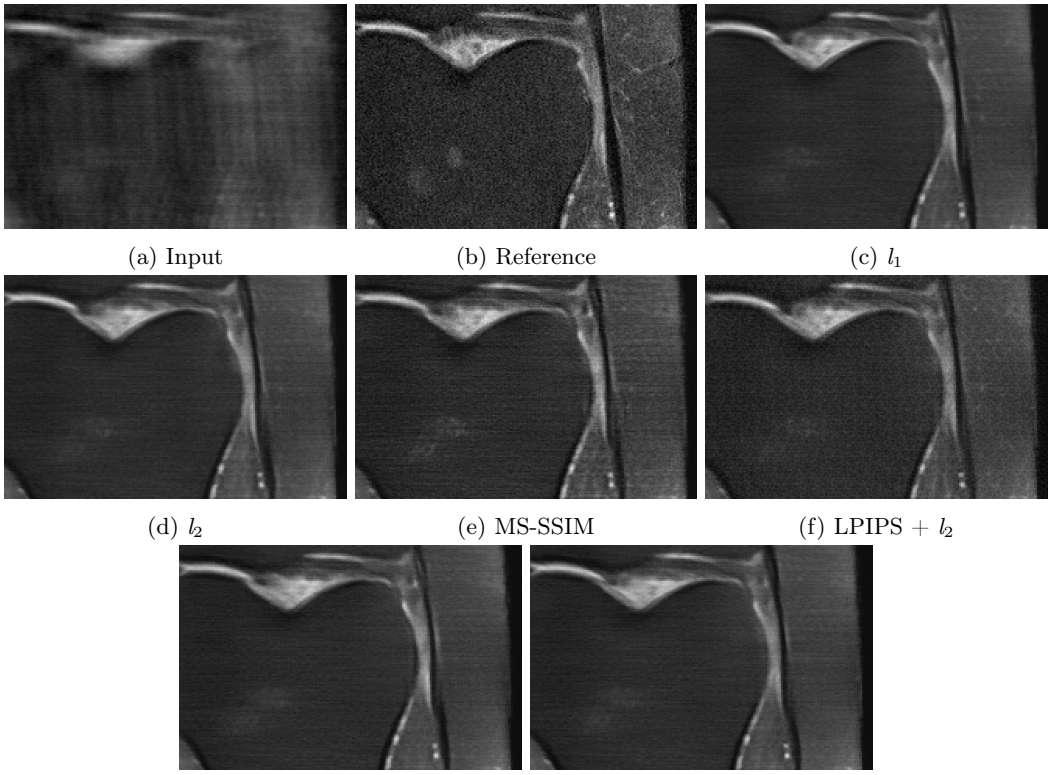


(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

Figure 6.11: Input image, reference image, and test images for the six loss functions stated in the captions when training on the fastMRI dataset for knee.

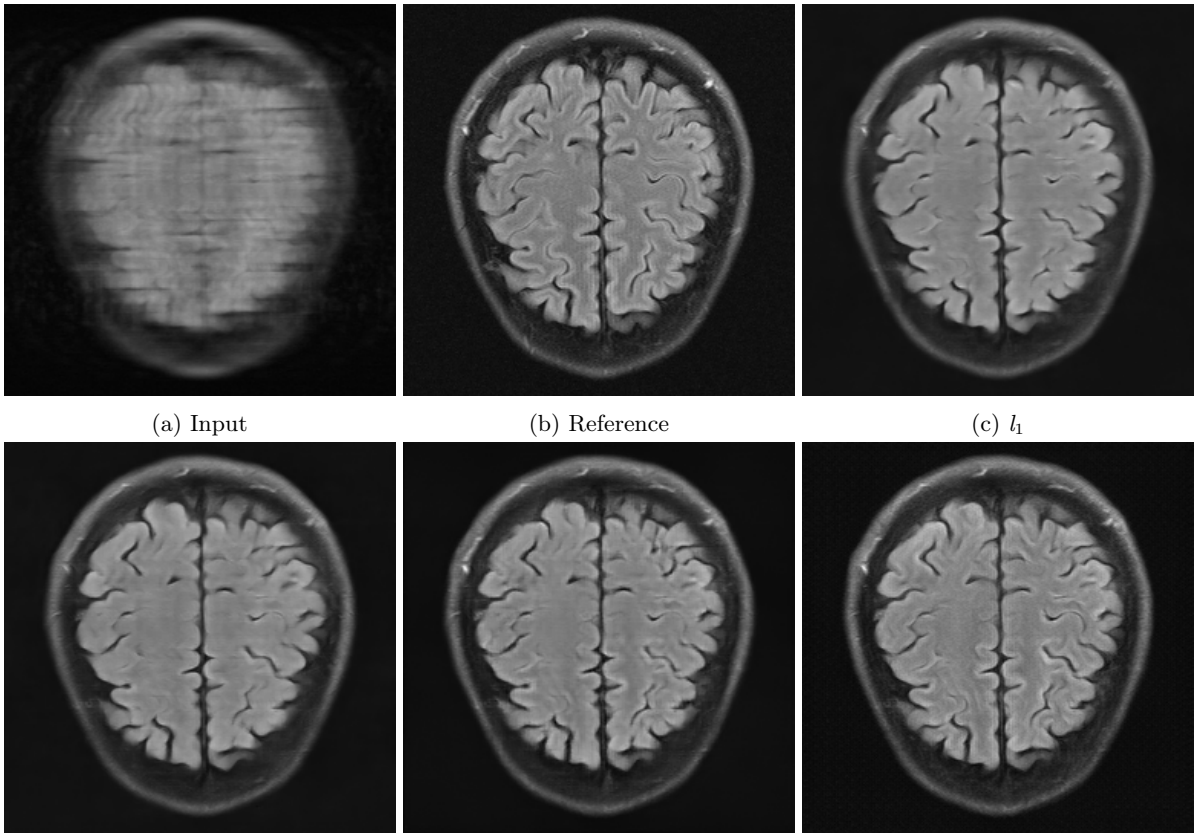


(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

Figure 6.12: Zoomed-in crop of Figure 6.11 of input image, reference image, and test images for the six loss functions stated in the captions when training on the fastMRI dataset for knee.

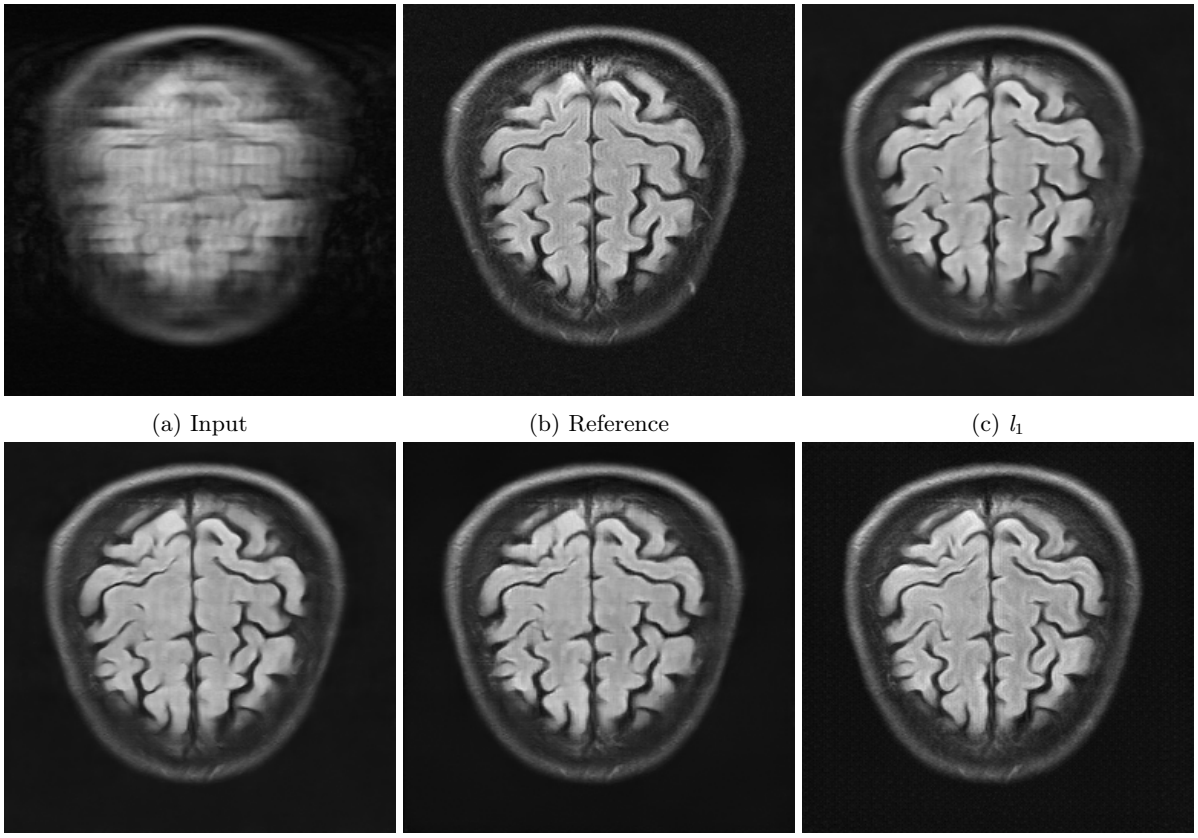


(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

Figure 6.13: Input image, reference image, and test images for the six loss functions stated in the captions when training on the fastMRI dataset for brain.

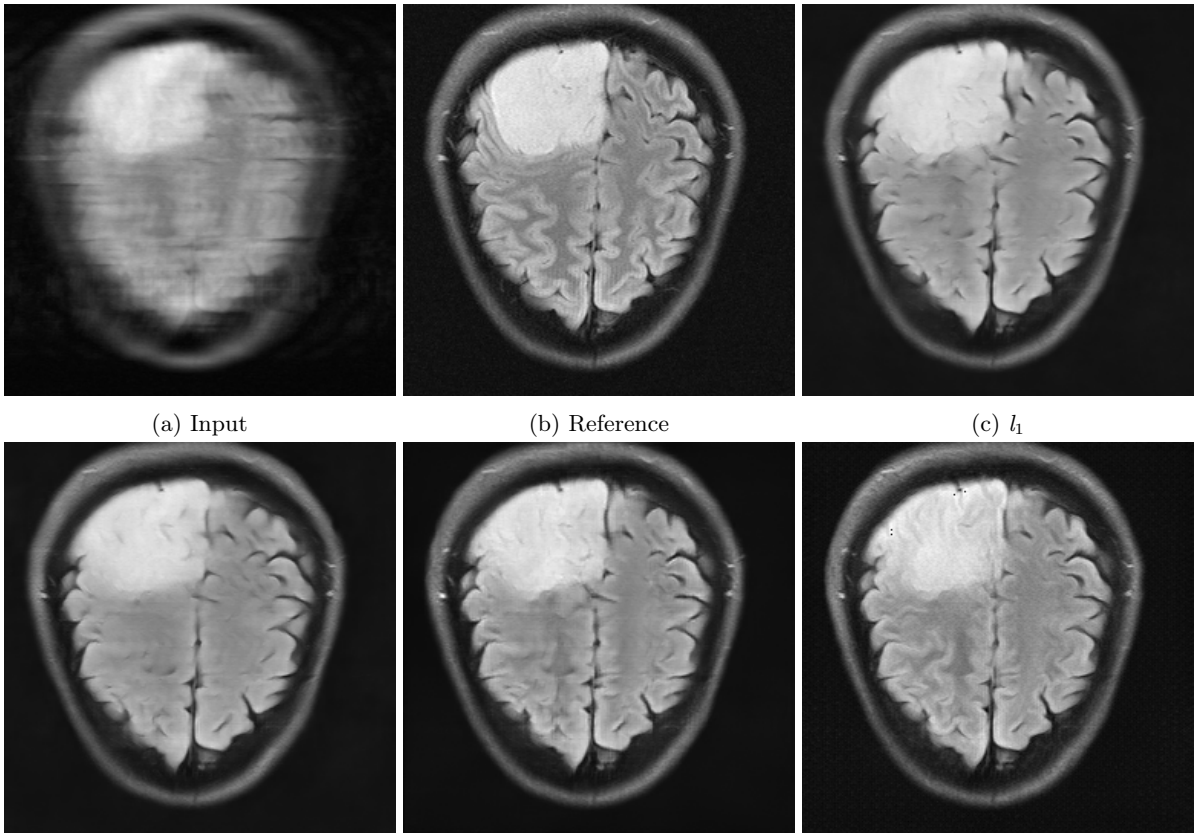


(a) Input (b) Reference (c) l_1

(d) Ll_2 (e) MS-SSIM (f) LPIPS + l_2

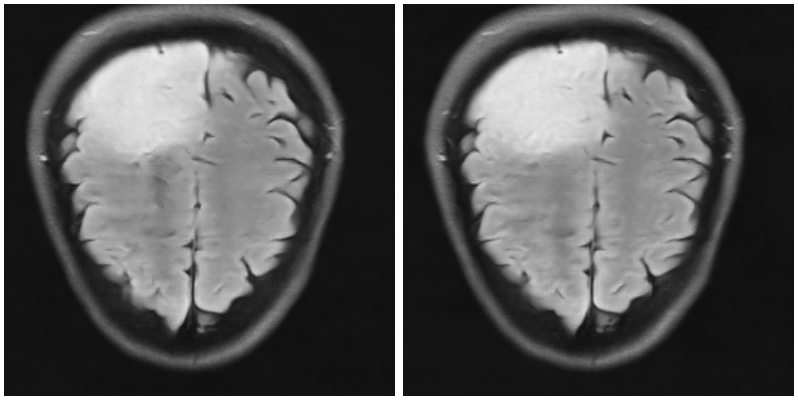
(g) FLIP (h) Adaptive

Figure 6.14: Input image, reference image, and test images for the six loss functions stated in the captions when training on the fastMRI dataset for brain.



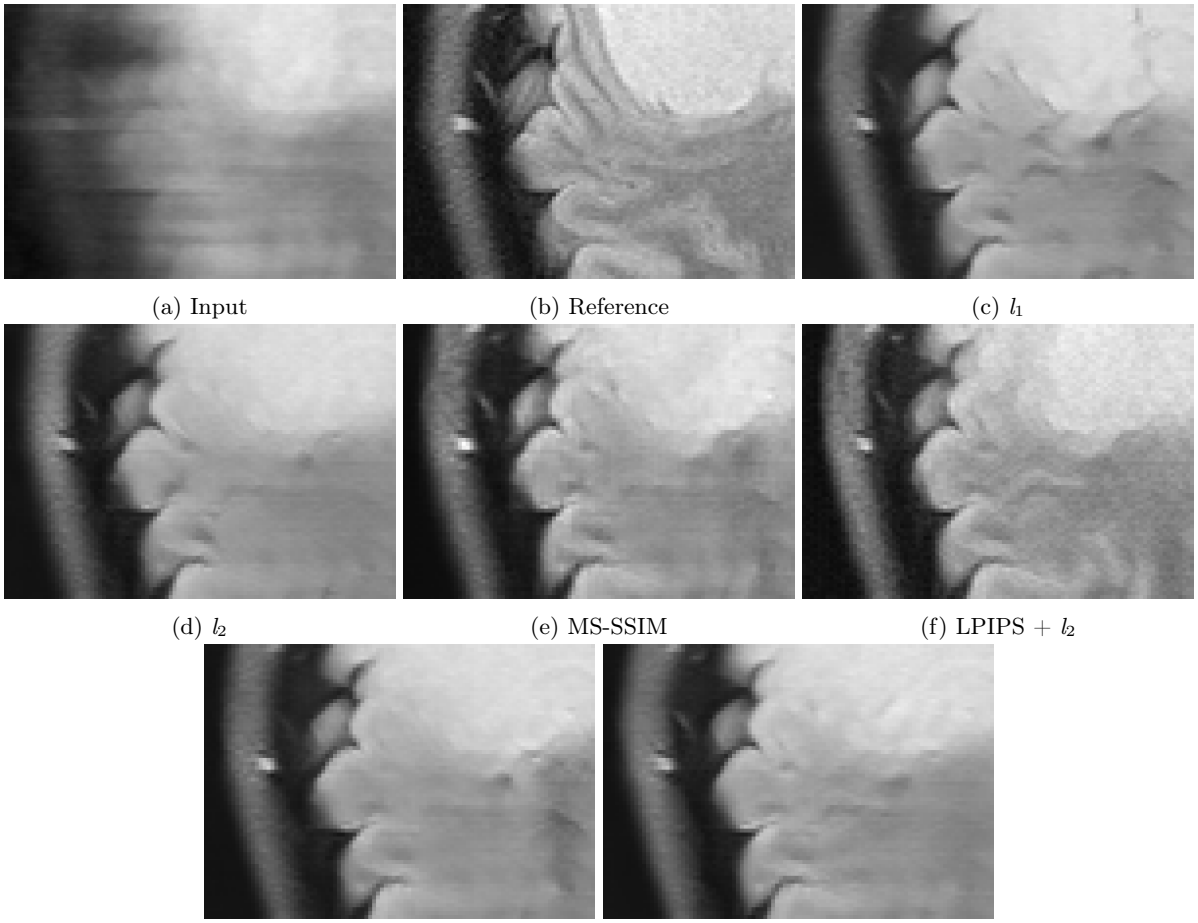
(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2



(g) FLIP (h) Adaptive

Figure 6.15: Input image, reference image, and test images for the six loss functions stated in the captions when training on the fastMRI dataset for brain.



(a) Input (b) Reference (c) l_1

(d) l_2 (e) MS-SSIM (f) LPIPS + l_2

(g) FLIP (h) Adaptive

Figure 6.16: Zoomed-in crop of Figure 6.15 of input image, reference image, and test images for the six loss functions stated in the captions when training on the fastMRI dataset for brain.

6.2.1 Measurements

In Table 6.3 and 6.4, the recorded measurements for our models for a set of metrics are shown.

Knee

Table 6.3: Recorded values when training with identical architecture on the fastMRI dataset of a knee, where the loss function differs between models. For each model, the calculated value of each metric was registered. For PSNR and MS-SSIM, a higher value represents a better MRI reconstruction. For l_1, l_2, LPIPS and FLIP, a lower value represent a better MRI reconstruction. For each model, the metric that received the best score is made bold.

Metric \ Model	Masked	l_1	l_2	MS-SSIM	LPIPS + l_2	FLIP	Adaptive
$1000 \cdot l_1$	28.91	**20.35**	20.53	20.96	21.89	20.46	20.36
$1000 \cdot l_2$	2.39	**1.16**	1.17	1.24	1.28	1.17	1.17
PSNR	26.71	**30.29**	30.25	30.01	29.81	30.23	30.27
MS-SSIM	0.9111	0.9591	0.9585	**0.9595**	0.9559	0.9590	0.9590
LPIPS	0.3517	0.3597	0.3618	0.3352	**0.1563**	0.3644	0.3589
FLIP	0.1972	0.1241	0.1238	0.1255	0.1309	**0.1222**	0.1245

Brain

Table 6.4: Recorded values when training with identical architecture on the fastMRI dataset of a knee, where the loss function differs between models. For each model, the calculated value of each metric was registered. For PSNR and MS-SSIM, a higher value represents a better MRI reconstruction. For l_1, l_2, LPIPS and FLIP, a lower value represent a better MRI reconstruction. For each model, the metric that received the best score is made bold.

Metric \ Model	Masked	l_1	l_2	MS-SSIM	LPIPS + l_2	FLIP	Adaptive
$1000 \cdot l_1$	54.66	28.17	28.32	30.06	32.59	**27.57**	27.64
$1000 \cdot l_2$	5.96	1.87	1.85	1.95	2.21	1.85	**1.83**
PSNR	22.61	27.53	27.55	27.37	26.83	27.56	**27.62**
MS-SSIM	0.8652	0.9541	0.9531	**0.9587**	0.9509	0.9544	0.9555
LPIPS	0.4274	0.2973	0.2971	0.2906	**0.1496**	0.3011	0.3002
FLIP	0.3053	0.1890	0.1918	0.2097	0.2280	**0.1842**	0.1862

6.2.2 User study

Here follows the results of the user study where 12 observers participated. The participants consisted of researchers at NVIDIA, other students from LTH, and our families.

Table 6.5: The probability to chose one loss function over another. For each comparison, the winner is made bold. If a comparison results in a tie, no one is made bold.

Loser \ Winner	l_1	l_2	MS-SSIM	LPIPS + l_2	FLIP	Adaptive
l_1	0	**0.5417**	**0.5625**	0.4375	0.2708	0.4583
l_2	0.4583	0	**0.6458**	**0.5833**	0.5000	0.4167
MS-SSIM	0.4375	0.3542	0	0.4375	0.2917	0.2917
LPIPS + l_2	**0.5625**	0.4167	**0.5625**	0	0.3750	0.3276
FLIP	**0.7292**	0.5000	**0.7083**	**0.6250**	0	0.5000
Adaptive	**0.5417**	**0.5833**	**0.7083**	**0.6724**	0.5000	0

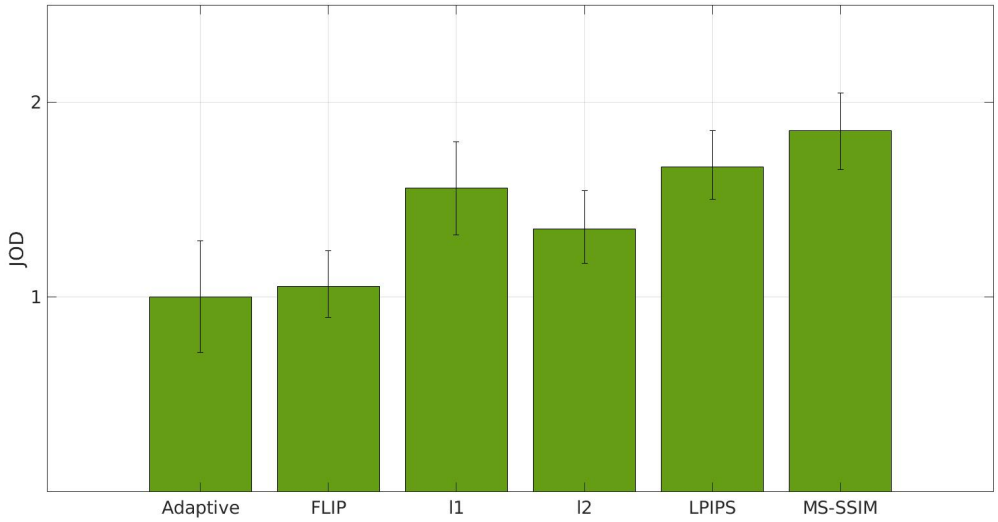


Figure 6.17: Visualization of the scaling results and 95% confidence intervals for the six loss function on the fastMRI dataset. The difference of 1 JOD indicates that 75% of observers selected one loss function as better than the other. The model with the lowest score is scaled to 1. The weighted loss LPIPS + l_2 has been used where it says "LPIPS".

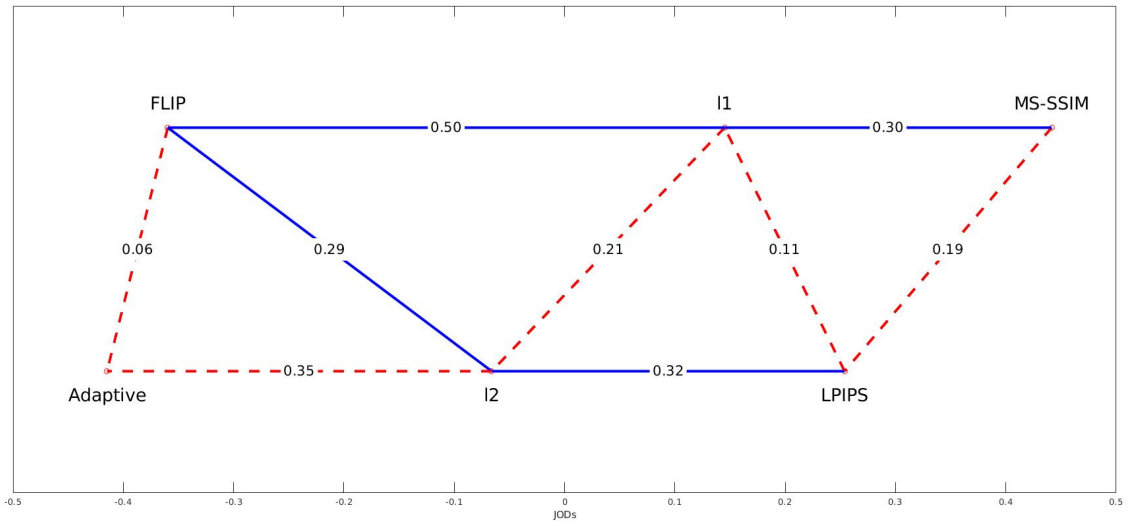


Figure 6.18: Graphical representation of the scaling for the six loss function on the fastMRI dataset. Each loss function is only connected to its close neighbours as this is what we are most interested in. Blue solid lines represent statistically significant differences and red dashed lines represent statistically insignificant differences. The x-axis shows the scaling in JODs. The weighted loss LPIPS + l_2 has been used where it says "LPIPS".

7 | Discussion

Our discussion is divided into three parts for each problem; measurements, our own observations when working with the metrics and the result from the user study.

7.1 Denoising

In this section for the denoising problem, the Figures 6.1, 6.2, 6.3, 6.4, 6.5, 6.6 are discussed.

7.1.1 Measurements

In Table 6.1, it is clear that the metric the model is optimized for is a biased parameter since most of the models perform best according to their own metric. What is interesting is that it still tells us something about the models. We see that l_1 is outperformed by l_2 for all metrics. The model optimized using Adaptive performs just slightly poorer than l_2 and this is reasonable since the Adaptive metric behaves as a mix between l_1 and l_2. The model using l_2 has the lowest FLIP values except for FLIP itself. Assuming that FLIP actually is a perceptual metric, then l_2 should be the second best model in terms of perceptuality. But maybe this just shows that there are some mathematical similarity between l_2 and FLIP. What is interesting is that the other two metrics that claim to be perceptual are MS-SSIM and LPIPS + l_2 and they obtain a poorer value of FLIP. The metric LPIPS, on the other hand, rates the weighted version of itself, LPIPS + l_2, as the best, followed by FLIP and then MS-SSIM.

7.1.2 Our own observations

When observing the outputs of the models, some differences are noticed. Below follows a discussion of how we perceive the images and an evaluation based on our subjective view. In Table 7.1, a summary of our findings are shown where we have given the models scores in the three criteria stated in Section 5.2.2.

Looking at the amount of denoising in Figure 6.2, a crop of Figure 6.1, LPIPS + l_2 seems to miss some noise, while all the other models denoised well. Considering the cloud in the image, LPIPS + l_2 has preserved the structure and details of it best and both MS-SSIM and l_2 smooths the cloud. When considering the tip of the roof, the stick is best preserved in FLIP, but a part still remains in the restoration of LPIPS + l_2 and l_1. The other models fail to keep the stick. Looking at the overall structure, although LPIPS + l_2 fails slightly in the denoising, the model keeps the image's structure.When looking at the windows of the house, the model using l_2 seems to be best at preserving straight lines.

In Figure 6.4, a crop of Figure 6.3, LPIPS + l_2 once again succeeds with restoring the structure of the image, but at the cost of the denoising part, which can be seen at the top of the girl's nose. The reflection in the eye is not as sharp with LPIPS + l_2 as the reference and the color is not correctly recreated. The other models seem to restore this reflection much better. Looking at the eyelashes, l_2 and Adaptive seem to keep more details than the other models. FLIP keeps the structure of the images fairly well. Looking at the paint strokes, all models but MS-SSIM and FLIP over-smooths the image. Here FLIP succeeds with keeping both the structure and details. On the other hand FLIP fails to restore the eyebrow under the colored circle.

In Figure 6.6, a crop of Figure 6.5, it is not as clear as before whether LPIPS + l_2 is poorer in its denoising than the other models. For the right facade with the wooden panel structure, FLIP fails the most to keep some of the most visible lines. Some of the branches are better preserved with MS-SSIM than l_1, l_2 and Adaptive. LPIPS + l_2 seems to cope as well. FLIP doesn't smooth the tree as much as l_1, l_2 and Adaptive. The rightmost branch is best preserved with FLIP, but at the same time a branch to the left is a bit too filled.

Table 7.1: Our own observations on the performance of the loss functions for the ImageNet dataset graded on a scale from 0 to 3.

Model	Denoising	Structure	Details
l_1	●●●	●○○	◐○
l_2	●●●	●○○	◐○
MS-SSIM	●●●	●●○	●○
LPIPS + l_2	●●◐	●●◐	●○
FLIP	●●●	●●○	●○
Adaptive	●●●	●○○	◐○

If we now sum up the scores from table 7.1 MS-SSIM, FLIP and LPIPS + l_2 are tied for first place.

7.1.3 User study

Looking at Table 6.2, the probability of a model, using a specific loss function, being perceived as better than another model is presented. It shows that the probability of LPIPS + l_2 being chosen over any of the other models is relatively high. Another thing that is clear is that l_2 is rarely preferred. It is also interesting that the probability of LPIPS + l_2 loosing against FLIP is larger the probability of LPIPS + l_2 loosing to MS-SSIM. This is probably because LPIPS + l_2 and MS-SSIM manage to capture the same appearances and when failing to do so FLIP manage better since that model uses a different approach. Comparing with our own observations of LPIPS + l_2 in Table 7.1, we state that LPIPS + l_2 is not as good as the other models at the actual denoising part. The good results of LPIPS + l_2 can partially be explained as some level of noise makes us perceive the image as sharper.

In Figure 6.7, the relative quality points are shown for each model with associated error bar illustrating the uncertainty. It is easy to crown LPIPS + l_2 the winner. MS-SSIM comes at second place and FLIP is close behind on third place. The model based on l_2 is an evident loser, obtaining the lowest score. It is also interesting that l_1 obtains a better score than Adaptive, although Adaptive is a new and suggested good loss function.

The triangle plot in Figure 6.8 shows whether the differences between the models are statistically significant or not, the blue lines show significance and the red-dashed indicate non-significance. Here it once again shows that LPIPS + l_2 is the most preferred choice, but it also shows that the difference between MS-SSIM and FLIP is not significant and therefore it is hard to tell which one of these that is the better choice. Nor is it clear what ranking l_1 should have. It is significant that l_2 obtains a lower score than any other model, and FLIP is significantly better then Adaptive.

When evaluating the results from this user study it is worth mentioning that users could have different perceptions on where the largest error occurs. One user can constantly focus on color shift, and another on details. Even if some results from the user study seem clear, a recurrent feedback was that it was sometimes difficult to choose which image was the better. This reasoning could contribute to understanding the inconclusiveness of some of the results.

7.2 MRI Reconstruction

In this section for the MRI reconstruction, the Figures 6.9, 6.10, 6.11, and 6.12 for the knee reconstruction and Figures 6.13, 6.14, 6.15, and 6.16 for the brain reconstruction, are discussed.

7.2.1 Measurements

In Table 7.2, it is l_1 that now outperforms l_2, and also Adaptive as opposed to the results discussed in the previous section. The model using l_1 is also the one with the second best FLIP value, second to FLIP itself. The same goes for the MS-SSIM value, confirming that l_1 this time seems to have found a good minimum. Another interesting result is that MS-SSIM is the second best model next to LPIPS + l_2 according to LPIPS and maybe this tells us something about the perceptuality in the MRI reconstruction using MS-SSIM. Still, the LPIPS value is somewhat difficult to interpret, noticing that the masked version obtains a lower LPIPS value than the reconstructed one using l_2, FLIP, and Adaptive, something we do not agree with. This result tells us a lot about the quality of LPIPS as a metric, the values does not represent the perceived quality of the images well.

In Table 6.4, Adaptive proves its right to be considered. In the case of the brain MR images, Adaptive performed better than both l_1 and l_2. Somewhat surprising is that FLIP obtains a lower l_2 value than any of the other models, something that is quite unexpected considering the previous results. Once again MS-SSIM obtains the second lowest LPIPS value, proving that there is some correlation between the two of them.

For both Table 6.3 and 6.4, LPIPS + l_2 performs very poorly according to l_1 and l_2. This verifies that the models are in two different categories, one purely mathematical (l_1 and l_2) and one more complex with properties hard to concretize (LPIPS + l_2).

7.2.2 Our own observations

Looking at Figure 6.9, both LPIPS + l_2 and MS-SSIM are perceived as sharper but they have some made up artifacts, like double lines in the lower right part. The model using l_1, as well as the one using FLIP, is a lot better at preserving the original line. For l_2, the line is still discernible but for Adaptive the line is gone. MS-SSIM and l_1 seem pretty good at preserving the overall details and colors. There are some lines between the fat and muscle tissue that l_2 and Adaptive preserve better than the other models.

In Figure 6.10, MS-SSIM resembles the reference most but LPIPS + l_2 is still a good competitor. Here FLIP, l_1, and MS-SSIM seem to preserve the details best, especially the ”dots” in the upper bone.

In Figure 6.11, FLIP fails to preserve the contrasts in the image. The cut in the lower right muscle is far best reconstructed by LPIPS + l_2, followed by MS-SSIM, while other models make up details in this area. In the cropped images in Figure 6.12, l_1 preserves the contrasts and the lines very well. Looking at the fat tissue, LPIPS + l_2 keeps the structures the best and MS-SSIM performs well in this area too.

Looking at the images in Figure 6.13, l_1 and l_2 are the most smoothed around the edges of the skull while LPIPS + l_2 and MS-SSIM reconstruct this part the best. LPIPS + l_2 also preserves the brightness the best, where MS-SSIM also succeeds to maintain some shift better than the other models. Adaptive, FLIP, l_1, and l_2 blend the colors more to a grayish one, instead of having the actual shifts.

Regarding Figure 6.14, LPIPS + l_2 tends to fill in some of the details, where the other metrics keep the truth, especially l_2, Adaptive, and FLIP. These three are more blurry than the others, though.

Looking at the images in Figure 6.15, and especially the tumor, LPIPS + l_2 fails to preserve the boundaries, but l_1 succeeds and is much more exact. Both LPIPS + l_2 and MS-SSIM make up brain structure in the tumor, that should not be there. FLIP is the best model for keeping the tumor clean at the cost of the whole image being smoothed. LPIPS + l_2 is still the best at preserving the brain's structure. Both LPIPS + l_2

and MS-SSIM are experienced as better at demasking the images.

In Figure 6.16, a crop of Figure 6.15, it shows that FLIP and l_1 are better at preserving the outlines of the tumor, FLIP is better regarding the horizontal line and l_1 reconstruct the edge and vertical line very well. Here it clearly shows that LPIPS + l_2 is no good at reproducing the tumor but the folds in the brain are well preserved, something that FLIP completely fails at. In the crop it shows that l_1, Adaptive, and LPIPS + l_2 preserves the color shifts well.

Table 7.2: Our own observations on the performance of the loss functions for the fastMRI dataset graded on a scale from 0 to 3.

Model	Demasking	Structure	Details
l_1	●●○	●◐○	●●○
l_2	●◐○	●○○	●○○
MS-SSIM	●●●	●●○	●●○
LPIPS + l_2	●●●	●●◐	●●○
FLIP	●●○	●○○	●◐○
Adaptive	●●○	●○○	●○○

When working with MR images, one may be much more concerned with small and exact details, not the overall perceptual experience. If the question is which model that resembles the reference the most, LPIPS + l_2 seems to be the best choice followed by MS-SSIM. But if the question is whether the tumor is clearly marked or not, then LPIPS + l_2 is the worst and l_1 and FLIP performs well. It seems wrong to neglect the fact that these are medical images, with a different purpose than the noisy images.

7.2.3 User study

For the user study regarding the MR images, the resulting probability of a model being chosen over another is shown in Table 6.5. The differences are not as large in this case as they were for the denoising problem in Table 6.2. One sees, though, that MS-SSIM is the winner closely followed by LPIPS + l_2 and, interestingly enough, l_1. Unlike the results from the user study regarding the denoising where FLIP was significant better than Adaptive, there is now a 50% probability to chose either of them. This also holds for the comparison between FLIP and l_2.

In Figure 6.17, the JOD value for the models is shown and here the differences are not as clear as for the denoising problem but we see that MS-SSIM is the big winner. FLIP and Adaptive obtain the lower scores, although the uncertainty of Adaptive's score are a lot larger than for FLIP.

Figure 6.18, shows the differences in quality and, once again, the blue lines indicate statistical significance and the red-dashed lines insignificance. Here we see that the difference between MS-SSIM and LPIPS + l_2 is not significant, nor is the difference between l_1 and LPIPS + l_2. We also see that l_2 performs alot better in the MRI reconstruction case than for the denoising problem, being better than both Adaptive and FLIP. For this problem l_1 is performing significantly better than FLIP.

Although MS-SSIM is the best performing model according to the user study, followed by LPIPS + l_2, they both have a tendency to create false artifacts in the MRI reconstruction case, and this is something important to be aware of.

An explanation of why l_1 and l_2 performed better for grayscale is that they only look at the individual pixel value. MS-SSIM also performs very well. This could be due to the metric being developed for grayscale images. LPIPS + l_2 on the other hand is a loss function using pretrained weights and has not been trained

on grayscale images, which could explain why it is not the clear winner in the case of MRI reconstruction. A similar explanation goes for the new metric FLIP which is developed for RGB images, taking color shifts and suchlike in consideration. This explains why FLIP performed considerably better for the denoising problem.

8 | Conclusion

In this thesis, a comparison between the newly developed metric FLIP and the established metrics l_1, l_2, MS-SSIM and LPIPS has been carried out. These metrics, alongside General & Adaptive Robust Loss, have also been used as loss functions to train a neural network as part of the comparison. The comparison consisted of outputs from two different problems, denoising and MRI reconstruction, where the metrics have been used as loss functions when training the same architecture of an U-net. The evaluation part consisted of measurements, our own observations, and user studies.

To answer the problem description given in Section 1.3 and to conclude our results we find that the new metric, FLIP, holds up well in comparison to the established metrics. Especially for denoising, FLIP is a good option, much better performing than l_2 which is maybe the most common metric today. There seems to be a lot of potential for this metric to obtain perceptually good results. For the MR images, FLIP is not a suitable loss function today.

A lot of work was put into evaluating the loss functions to try to sort out the quality of them. We found that a mix of evaluation methods gave the most holistic perspective. The user study showed that LPIPS + l_2 and MS-SSIM are the best loss functions for obtaining a perceptually good result. For using LPIPS and MS-SSIM as metrics, they fail to tell how good the other models are.

There is no metric that fully correlates with the results of the user study i.e., there is no mathematical measurements that we have made which tell us how people will perceive the resulting images.

It has become clear that the purpose of the loss function matters a lot. What data one uses should affect the choice of loss function for a network; whether the loss function is developed for RGB images or for grayscale images.

A part of this thesis was to make the novel metric FLIP applicable for deep learning. As this is a first attempt, additional work could be done for tweaking the metric to be even more suitable for deep learning. As the metric was developed for RGB images, further effort could be made for making FLIP more suitable for grayscale images. Furthermore, deciding which learning rate is the optimal for a specific metric is a trial-and-error task. Therefore it is not definitive that featured learning rates in this thesis are ideal, and future work could be done on this matter.

To further broaden the analysis of the loss functions, more training sessions could be performed to better capture the variation within the obtained models with the same loss function. It could be interesting to compare these possible variations with the variations between models using different loss functions.

References

[Anonymous, 2020] Anonymous (2020). FLIP: An Image Difference Evaluator for Alternating Images. *HPG2020 (in submission)*.

[Barron, 2017] Barron, J. T. (2017). A General and Adaptive Robust Loss Function. *CoRR*, abs/1701.03077.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press, first edition.

[Kaplanyan et al., 2019] Kaplanyan, A. S., Sochenov, A., Leimkühler, T., Okunev, M., Goodall, T., and Rufo, G. (2019). DeepFovea: Neural Reconstruction for Foveated Rendering and Video Compression using Learned Statistics of Natural Videos. *ACM Trans. Graph*, 38(4).

[Perez-Ortiz and Mantiuk, 2017] Perez-Ortiz, M. and Mantiuk, R. K. (2017). A Practical Guide and Software for Analysing Pairwise Comparison Experiments.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597.

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheokh, H. R., and Simoncelli, E. P. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4).

[Wang et al., 2003] Wang, Z., Simoncelli, E. P., and Bovik, A. C. (2003). Multi-Scale Structural Similarity for Image Quality Assessment. *IEEE Asilomar Conference on Signals, Systems and Computers*.

[Zbontar et al., 2018] Zbontar, J., Knoll, F., Sriram, A., Muckley, M. J., Bruno, M., Defazio, A., Parente, M., Geras, K. J., Katsnelson, J., Chandarana, H., Zhang, Z., Drozdzal, M., Romero, A., Rabbat, M., Vincent, P., Pinkerton, J., Wang, D., Yakubova, N., Owens, E., Zitnick, C. L., Recht, M. P., Sodickson, D. K., and Lui, Y. W. (2018). fastMRI: An Open Dataset and Benchmarks for Accelerated MRI.

[Zhang et al., 2018] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *CVPR 2018*.

[Zhao et al., 2017] Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2017). Loss Functions for Image Restoration with Neural Networks. *IEEE Transactions on computational imaging*.

Master's Theses in Mathematical Sciences 2020:E33
ISSN 1404-6342

LUTFMA-3413-2020

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/