ELECTRICAL AND ELECTRONICS ENGINEERING
SEMESTER PROJECT
Master Semester 2 - Fall 2019

Learning Based Image Quality Assessment Metrics Trained On Subjective Scores And Their Use In Image Compression

Student: Vivien DU BOIS DE DUNILAC

Supervised by: Pinar AKYAZI
Negin SAFAEI
Prof. Dr. Touradj EBRAHIMI

January 10, 2020

MULTIMEDIA SIGNAL PROCESSING GROUP
EPFL

EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

MULTIMEDIA SIGNAL PROCESSING GROUP
INSTITUTE OF ELECTRICAL ENGINEERING – IEL
SCHOOL OF ENGINEERING – STI

Prof. Dr. Touradj Ebrahimi
EPFL/STI/IEL/GR-EB
ELD 241
Station 11
CH-1015 Lausanne

Direct Phone : +4121 693 2606
Secretariat : +4121 693 2624
Fax : +4121 693 7600
E-mail : Touradj.Ebrahimi@epfl.ch
WWW : http://mmspl.epfl.ch

Lausanne, Winter 2019

Semester Project for
Vivien du Bois de Dunilac
Section of Computer Science (CS)
Semester Project Winter (PSH)

Deep Neural Network Based Image Quality Assessment Software Development

Image and video compression are essential tools for the storage and transmission of data for streaming and broadcasting. However, compression introduces artifacts and distortions in the original content, which may lead to a decrease in the quality of viewers' experience. It is therefore important for content providers and service providers to be able to compress the content efficiently and achieve high quality using low bitrates, and to evaluate the quality of images, using objective metrics and subjective quality assessment methodologies. Deep convolutional networks are increasingly used for vision tasks including image compression and quality evaluation. Current learning-based image compression algorithms present competitive results compared to state-of-the-art image compression methods such as JPEG. The performance of current learning-based image quality assessment algorithms is also competitive to that of state-of-the-art objective metrics in terms of correlation with subjective assessment results.

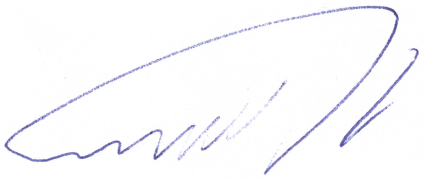
The goals of this project are two-fold. The first step is to implement a robust learning-based image quality assessment metric. This metric is then to be used as the distortion measure of a learning-based compression algorithm.

The following tasks should be performed during the project:

1. Study the state of the art on learning-based image quality evaluation.
2. Build a state-of-the-art learning-based image quality assessment model.
3. Train and test the model on selected image quality assessment databases.
4. Report the correlation of the model with subjective results and compare the performance of the model to state-of-the-art metrics. The comparison should include at least one learning-based metric.
5. Assess the strengths and weaknesses of the studied model.
6. Assess the performance of the model with respect to
    a. Loss function (batch correlation vs. mean absolute error)
    b. Patch size, number of patches
7. Use the learning-based metric in the loss function of a state-of-the-art learning-based image compression solution. Train and test the combined model and analyze the results.
8. Try to improve the precision of the metric by:
    a. Performing adversarial training
    b. Setting upper and lower bounds for predicted scores
    c. Using an objective metric during training
9. Conduct a thorough analysis on the results.

Deliverables (final report, final presentation, electronic package of all documents and software, etc.) must be submitted according to EPFL and MMSPG guidelines and schedule.

Responsible Assistants: **Pinar Akyazi, Negin Safaei**



Prof. Dr. Touradj Ebrahimi
Head of Multimedia Signal Processing Group

CONTENTS

Abstract

In the context of image compression image quality assessment (IQA) metrics can be used to help minimize the distortions perceived by humans in decoded images. This report studies different ways of optimizing deep neural network based full reference IQA metrics trained on subjective opinion scores and using them as loss functions for autoencoders aimed at image compression. Several ways of making the IQA metrics more resistant to the optimization procedure of the autoencoder that uses it in its loss function by exploiting characteristics of the full reference IQA problem are tested and the results are compared with the standard mean squared error loss function. Multiple techniques that aim to improve the generalization power of IQA neural networks to both new images and distortion types that have not been seen during training are also studied. The new models that are created by applying these techniques are evaluated on the LIVE, CSIQ, TID2013, JPEG_AI and JPEG_XL databases.

I. Introduction

Image quality assessment is a fundamental problem in image processing because of its wide range of applications. Objective image quality metrics which can be computed by machines can be used to automatically optimize any image transformation method without needing long, expensive and error prone human experiments. For instance when training an image compression neural network an objective metric can be used in the network's loss function to minimize the distortions introduced by the compression procedure. Objective metrics can also be used in many other applications such as benchmarking different image processing solutions that accomplish the same task or monitoring the quality of a broadcast in real time. Designing good metrics is challenging [1] and the many different use cases do not have the same requirements: training machine learning models with gradient descent requires the metric to be differentiable, image compression networks need to assess the notion of quality as perceived by humans and image processing pipelines need to judge how well suited the images that they produce at different stages are for the next to cite a few examples.

In this report we will focus on deep learning based image quality assessment metrics that aim to reproduce the judgment of a human being in the context of full reference image quality assessment, that is given an original image and a distorted version of this image measure the impact of the distortion on a scale of for instance one to ten. Deep learning based IQA metrics are used today because they provide good performance while being differentiable by definition. Some metrics are based on existing non-differentiable solutions and act as proxies for them [2] while others are trained on the results of subjective quality assessment experiments [3]. While neural networks are a great tool they are not without drawbacks and a major problem of using them as metrics to be optimized is their weak resistance to adversarial perturbations [4] which can result in worse image quality if they are used without caution.

This project aims to improve on existing deep learning based IQA metrics trained on subjective scores and study their use in the loss functions of image compression networks. The models presented in this report are based on the WaDIQaM-FR architecture [3] which consists of a siamese convolutional network to extract features from selected patches taken from the reference and distorted image and two fully connected networks that use the extracted features to produce a score and a weight for each patch. The final score of the distorted image is the weighted sum of the scores of the patches.

The report is structured as follows: In Section II a detailed description of the reference architecture and of the new models as well as performance evaluation of all the models are given. Section III details the use of the tested architectures in the loss functions of compression networks and compares them to a pure mean squared error baseline. The report is concluded in Section IV.

II. Deep learning based full reference image quality assessment

Because the goal of this project is to use the resulting metrics to optimize compression networks full reference metrics are much more interesting than their no reference counterparts as compression networks will create a distorted versions of the reference image they are being trained on at every step of training and full reference metrics correlate better with human judgment than their no reference counterparts. [3] In the rest of this report WaDIQaM will designate WaDIQAM-FR and any presented metric will be full reference unless specified otherwise.

It is interesting to note that the problem of full reference IQA based on subjective user scores has unique characteristics that can be useful. First the scores are bounded by the scale used in the experiments: it would make no sense for a network to give a score of 100 to an image if it has been trained on data from an experiment in which the participants were told to give a score between one and five. Then there exists a perfect solution for any image: if the distorted image is perfectly identical to the reference image the network can give it a perfect score. Finally since the target value is a single scalar it is probable that training several models on the same data yields a collection of metrics that give similar scores for the same image. If the models have a low bias [5] then standard variance reduction techniques such as averaging the results of different models should improve the performance of the models.

A. Methodology

To achieve the goal of improving the performance of deep learning based IQA metrics a reference metric was chosen to serve as a baseline architecture. Then several new network architectures were built by varying the topology of the network, the loss function and the regularization method used and their performances were compared against that of the baseline. The following subsections describe the databases used and the baseline architecture as well as the training and performance evaluation methods in details and explain the obtained results.

1) Databases: The performance of the models is measured on the TID2013 [6], LIVE [7], CSIQ [8], JPEG_XL and JPEG_AI image quality databases. TID2013 contains 3000 distorted images obtained by applying five different levels of 24 different distortions to 25 reference images. The distortions used in this database are based on random and structured noise as well as compression artifacts of JPEG and JPEG2000. LIVE contains 779 distorted images obtained by applying different levels of 5 distortions to 29 reference images. It shares approximately half of its reference images and four of its distortion types with TID2013. CSIQ contains 866 distorted images obtained by applying between 4 and 5 levels of 6 distortion types to 30 reference images. The images are completely different from the other databases but the distortion types are very similar to TID2013 and LIVE. The JPEG_XL database contains 305 distorted images obtained by compressing 7 reference images using different bitrates on the following codecs: aom, deepcoder, fuif, fvdo, hevc, jpeg, kakadu, pik, xavs, tat and webp. The JPEG_AI database contains 266 distorted images obtained by compressing 8 reference images using different bitrates on different image compression codecs including some deep learning based codecs.

2) Baseline architecture: The metrics are evaluated using the WaDIQaM architecture which is shown in figure 1 as a baseline. It consists of a siamese convolutional network that extracts features from 32 by 32 pixels image patches. The feature vectors for the distorted and reference feature vectors are then merged by subtracting one from the other, concatenation or both. Finally the result of the merging operation is either sent to a single regression network that consists of two fully connected layers in which case the final image quality estimate is obtained by averaging the scores of the patches or to two regression networks where one computes a score for each patch while the other compute a weight, the final quality estimate is then obtained by computing the weighted average of the patch scores. In the first case the model is called Deep Image QuAlity Measure (DIQaM) and in the second case it is called Weighted Average Deep Image QuAlity Measure (WaDIQam). The original paper states that the different methods of merging the reference and distorted feature vectors has a minimal

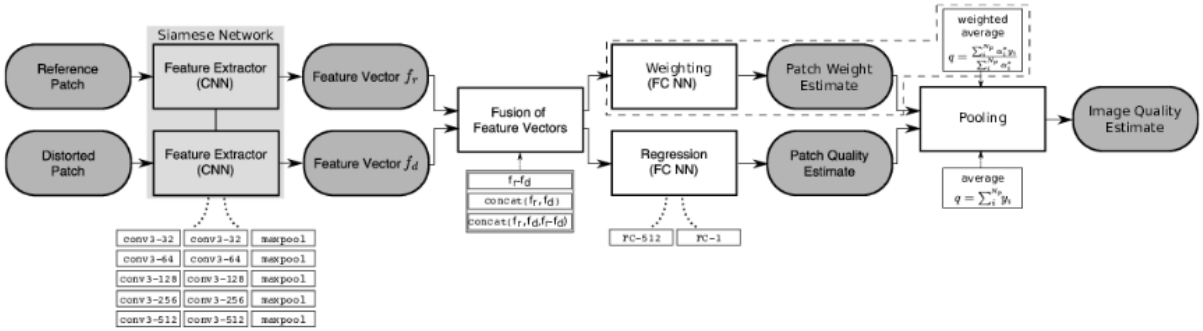


Figure 1: The DIQaM/WaDIQaM architecture taken from [3]

impact on the performance of the model, for this reason all of the models presented in this project use the simple subtraction of one vector from the other to combine them.

The WaDIQaM architecture as described in the original paper uses convolutional layers followed by max pooling layers to extract features from the image patches. It has been shown [9] that replacing max pooling layers by setting the stride of the existing convolutional layers to the desired downsampling factor produces networks with similar performances. This is done in the models trained in this project. The original paper does not mention any activation function for the layer that computes the weight given to each patch for the weighted average pooling of the final score. To ensure that the weights are always positive and avoid dividing by zero or having some negatively weighted patches canceling out other positively weighted patches a softplus activation function $f(x) = \log(1 + e^x)$ where log is the natural logarithm is added to the weight output layer.

The baseline architecture uses dropout [10] with a ratio of 0.5 as a regularization method on all the fully connected layers.

3) Training: The training method used in this project is similar but not identical to the one used in the WaDIQaM paper [3].

The models are trained with the ADAM optimizer [11] using the implementation of TensorFlow 1.14.0. The parameters of the optimizer are as follows: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\alpha = 5$ x 10^{-5}. The different models are all trained on the full TID2013 database. At every step of training ten images are randomly chosen from the distorted images of TID2013. From those images and the corresponding reference images 64 random patches of size 32 by 32 are extracted. The resulting batch is then fed to the optimizer. The random selection of patches is a way of augmenting the available data because it is extremely unlikely that two batches will ever be completely identical during training. This contributes to improving the robustness of the network and helps prevent overfitting. Since the selection of the images is completely random there is no practical way of knowing when all samples have been seen. For this reason an epoch is hereafter defined as 300 steps of training which corresponds to 3000 images, the size of the TID2013 database. After every epoch of training the model is validated on the JPEG_AI database. The model obtained after training until convergence and the one that obtained the best validation score are kept. Keeping the model that performed the best during validation is the same as early stopping and is a way of preventing overfitting by stopping the training before the validation loss starts increasing.

4) Performance evaluation: The final goal of an image quality assessment metric trained on human given mean opinion scores is to correlate well with scores given by humans. To measure how well the models perform at this task different correlation coefficients can be used. In this project the Pearson linear correlation coefficient is used to measure how well the predictions of the models correlate with human given scores and the Spearman rank order coefficient is used to measure the monotonicity of the predictions with respect to the mean opinion scores. The coefficients are computed using the stats module of scipy 1.2.1.

Table I: Results obtained for the baseline architecture

Database	DIQaM - Spearman	DIQaM - Pearson	WaDIQaM - Spearman	WaDIQaM - Pearson
TID2013	0.9295	0.9293	0.9349	0.9382
LIVE	0.8889	0.8804	0.8944	0.8878
CSIQ	0.9354	0.9238	0.9156	0.9043
JPEG_AI	0.7713	0.7954	0.8025	0.8253
JPEG_XL	0.8219	0.8064	0.8492	0.8226

Since the models are trained on TID2013 it is expected that they will perform better on images and distortions that are similar to those of TID2013. The LIVE database is quite similar to TID2013, about one half of LIVE's reference images also appear in TID2013 and most of the distortion types present in LIVE are also present in TID2013. This makes LIVE a poor choice to evaluate the generalization capabilities of the models but is still useful to test them on it because since the distorted images of LIVE are computed using distortions that are present in the training set, obtaining poor performance on the full LIVE database compared to TID2013 would indicate a severe lack of generalization power. The CSIQ database does not share any images with TID2013 but all of its distortion types are present in it. This makes it perfect to measure how well the metrics deal with new images. The JPEG_AI database does not share any images with TID2013 and because its distortions are all image codec based it only shares the JPEG and JPEG2000 distortion types with the training database which makes it very well suited to judge the generalization capabilities of the models. The fact that it is used to compute validation scores during training which are in turn used to keep the best model does mean that the scores measured on this database will probably be higher than those that would be measured on a database that has no link with the training process. Finally the JPEG_XL database shares no images with any other but does use some of the same codecs as JPEG_AI. Because of this no database is truly independent of all the others and measuring the generalization of the models completely fairly is not possible with this training setup. Ideally one would need three completely different databases in both distortions and reference images in order to measure the generalization power of the models with perfect accuracy but the problem then becomes the size of the databases.

To produce the quality estimates presented in the following sections 128 random patches are sampled for the small images of TID2013, LIVE and CSIQ and 256 patches are sampled for the larger images of JPEG_XL and JPEG_AI. The number of patches used were chosen to give similar results across different runs despite the randomness of the patch selection: the mean relative difference between two runs is 0.3% for small images and 0.5% for large images. Since two models are kept at the end of the training procedure the results will only show the ones that perform the best on the validation databases JPEG_AI and JPEG_XL. The results of this evaluation setup for the baseline architecture are given in table I and the graph of the training and validation losses per epoch can be seen in figure 2.

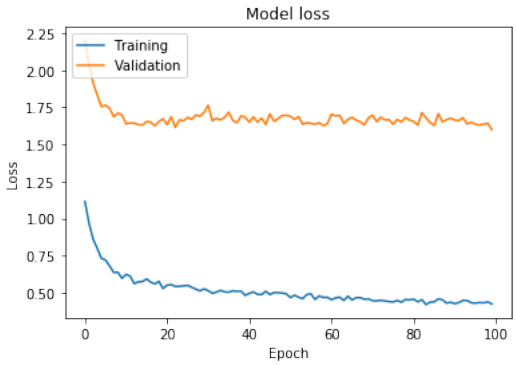


Figure 2: The training and validation loss of the baseline WaDIQaM model during training

Table II: Results obtained for a WaDIQaM model using MSE as its loss function

Database	WaDIQaM-MSE - Spearman	WaDIQaM-MSE - Pearson
TID2013	0.9368	0.9290
LIVE	0.8955	0.8854
CSIQ	0.9033	0.8796
JPEG_AI	0.7980	0.7900
JPEG_XL	0.8905	0.8690

Table III: Results obtained for a WaDIQaM model using the Pearson correlation coefficient as its loss function

Database	WaDIQaM-Corr - Spearman	WaDIQaM-Corr - Pearson
TID2013	0.9331	0.9365
LIVE	0.8924	0.8720
CSIQ	0.9114	0.8931
JPEG_AI	0.7629	0.8059
JPEG_XL	0.8527	0.8118

B. Loss functions

1) Mean squared error: Using mean squared error instead of mean absolute error can seem reasonable to prevent large errors on some samples but it also makes the training procedure more vulnerable to outliers. During the training of the model using MSE as the loss function the training loss behaved well but the validation loss was less stable than for MAE as shown in figure 3. The

Figure 3: The training and validation loss of the WaDIQaM model using MSE as its loss function during training

WaDIQaM-MSE model obtained the results shown in table II.

Mean squared error performed significantly worse than mean absolute error. A possible explanation for this is that human given scores are noisy and can induce the model in error if they are followed too closely. Mean absolute error is less sensitive to this type of problem.

2) Batch correlation: Since the goal of the metric is to correlate well with human scores it makes sense to use a correlation coefficient as the loss function of the IQA network. However this does complicate the training procedure as it is necessary to use large batch sizes to compute meaningful correlation coefficients and using only correlation coefficients in the loss functions means that the outputs of the network are not necessarily in the same range as the original scores which will cause additional problems if the metric is used in a compression network later as explained in section III. During the training of this model both the training and validation losses were less stable than for other models as shown in figure 4. The WaDIQaM-Corr model was trained using $1 - \text{PEARSON}^2$ as its loss function and obtained the results shown in table III.

The WaDIQaM-Corr model performed as well as the reference on the training set but did not generalize well. An explanation for this is that when using this loss function the network does not try to learn to predict the scores themselves but rather how to correlate them well inside a single batch of data. The range of the scores given by this model was also much smaller than for MAE. It is possible

Figure 4: The training and validation loss of the WaDIQaM model using batch correlation as its loss function during training

that numerical errors start to play more of a role when all of the predicted scores are closer together.

A regularized version of the loss function $L = \text{MAE} + (1 - \text{PEARSON}^2)$ was also tried but the training procedure became even more unstable and the results even worse. Many other possibilities of combining the correlation coefficients in the loss function can still be tried but none that were used in this project yielded competitive results.

C. Regularization

1) Dropout: In the dropout paper [10] the authors mention that when training a network consisting of several convolutional layers followed by dense layers adding dropout not only to the fully connected layers but also to the convolutional layers can help obtaining better performance. To verify if this technique was applicable to the WaDIQaM architecture several models were trained using different dropout values on the convolutional layers but all of them performed significantly worse than without using dropout even when using a drop rate of only 5%.

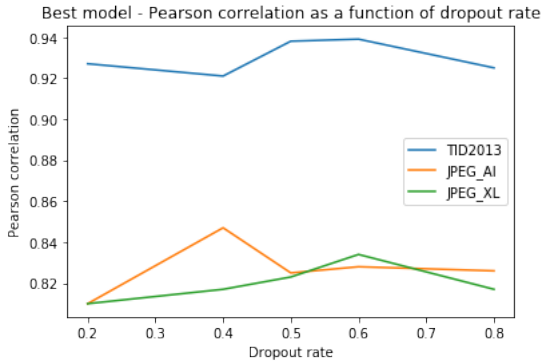
The original WaDIQaM paper does not mention any value for the dropout rate apart from 50%. To study the effect of varying the dropout rate applied to the dense layers models were trained for rates between 20% and 80%, the results achieved by these models are presented in figure 5. We can see that on fully trained models the increase in dropout rate reduces the correlation scores on the training database while the results on the validation databases are best for drop rates between 40% and 50%.

D. Architecture

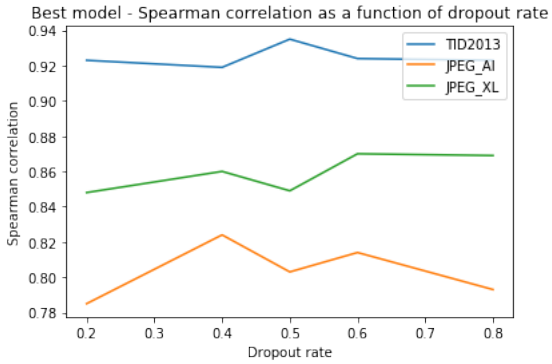
1) Residual connections and dense networks: One way of improving the performance of particularly deep neural network is using residual connections [12]. One can also use dense networks [13] to achieve the same goal. For the WaDIQaM base model these techniques do not yield very good results which is probably due to the fact that it is only twelve layers deep compared to the hundreds of layers of the models that typically benefit from them.

The models trained using these techniques scored slightly lower than the baseline and their results will not be detailed.

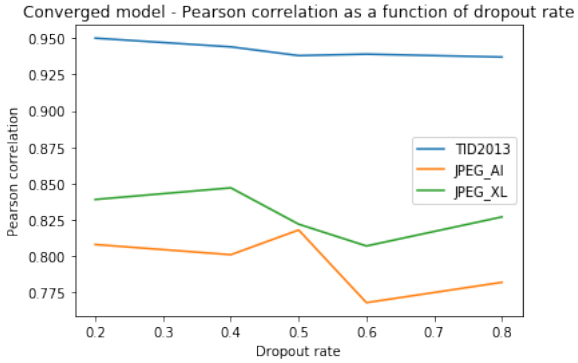
2) Patch sizes: The original WaDIQaM paper does not mention tests with other patch sizes than 32 by 32. To test the influence of this parameter on the baseline architecture two additional models have been: one uses 16 by 16 patches and the other 64 by 64 patches they will now be referred to as WaDIQaM-16 and WaDIQaM-64. The feature extraction part of the network consists of five groups of two convolutional layers in which the second layer downsamples by a factor of two. To adapt the models to their new patch sizes one group of convolutional layers is removed from the 16 by 16 patch size model and one is added to the 64 by 64 patch size model using layers with 1024 filters. The results achieved by these models are shown in tables IV and V.



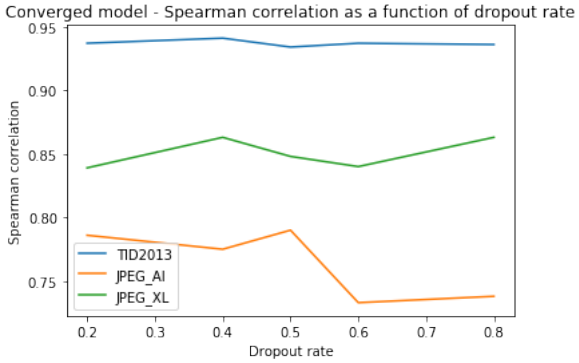
(a) The influence of dropout rate on Pearson correlation for models with the best validation loss during training



(b) The influence of dropout rate on Spearman correlation for models with the best validation loss during training



(c) The influence of dropout rate on Pearson correlation for fully trained models



(d) The influence of dropout rate on Spearman correlation for fully trained models

Figure 5: Results of varying the dropout rate of the fully connected layers of the baseline architecture

Table IV: Spearman correlation coefficients obtained for the baseline architecture and the models using different patch sizes

Database	WaDIQaM-32	WaDIQaM-16	WaDIQaM-64
TID2013	0.9349	0.9286	0.9015
LIVE	0.8944	0.9023	0.8744
CSIQ	0.9156	0.9179	0.8826
JPEG_AI	0.8025	0.7874	0.7492
JPEG_XL	0.8492	0.8521	0.8048

Table V: Pearson correlation coefficients obtained for the baseline architecture and the models using different patch sizes

Database	WaDIQaM-32	WaDIQaM-16	WaDIQaM-64
TID2013	0.9382	0.9309	0.9072
LIVE	0.8878	0.8953	0.8574
CSIQ	0.9043	0.9055	0.8713
JPEG_AI	0.8253	0.8072	0.7521
JPEG_XL	0.8226	0.8316	0.7933

The WaDIQaM-16 model achieves slightly worse performance than the baseline architecture this can be explained by the reduction of information present in the patches or by the removal of one layer in the feature extraction step. In the original WaDIQaM paper a smaller architecture that uses the same number of layer as WaDIQaM-16 is tested and the reduction in correlation is of the same magnitude which suggests that the depth of the feature extractor is more important than the size of the patches.

The performance of the WaDIQaM-64 is significantly worse. My hypothesis is that this is due to the lack of depth and small number of filters of the feature extractor. Unfortunately the model is already large with tens of millions of parameters and the NVIDIA GTX 970 that is available for training does not support more than the proposed WaDIQaM-64.

3) Multi-scale network: A multi-scale network uses information from inputs of multiple sizes that are then combined in some way to obtain the desired output. The inputs can be obtained using different methods: the simplest is to use different spatial scales [14] but transforms can also be applied to the data to obtain more information, one can also use different scales of the wavelet decomposition of the data to obtain good multi-scale inputs [15]. To investigate the benefits of using a multi-scale network a model that uses both 16 by 16 and 32 by 32 patches that will be called MS-WaDIQaM has been trained. The feature extractors of the baseline architecture is used for the bigger patches and the smaller patches use the same extractor as WaDIQaM-16. The feature vectors are then concatenated to obtain a single vector of size $256+512 = 768$ for each patch. Finally the feature vectors are fed to the networks of fully connected layers that compute the patch weight and score and aggregated through a weighted average of the scores much like in the baseline architecture with the only difference being that they now consist of two fully connected layers with 400 outputs and a dropout rate of 55% and an output layer instead of a single 512 outputs layer with a dropout rate of 50% before the output layer. A visualization of this new architecture is presented in figure 6.

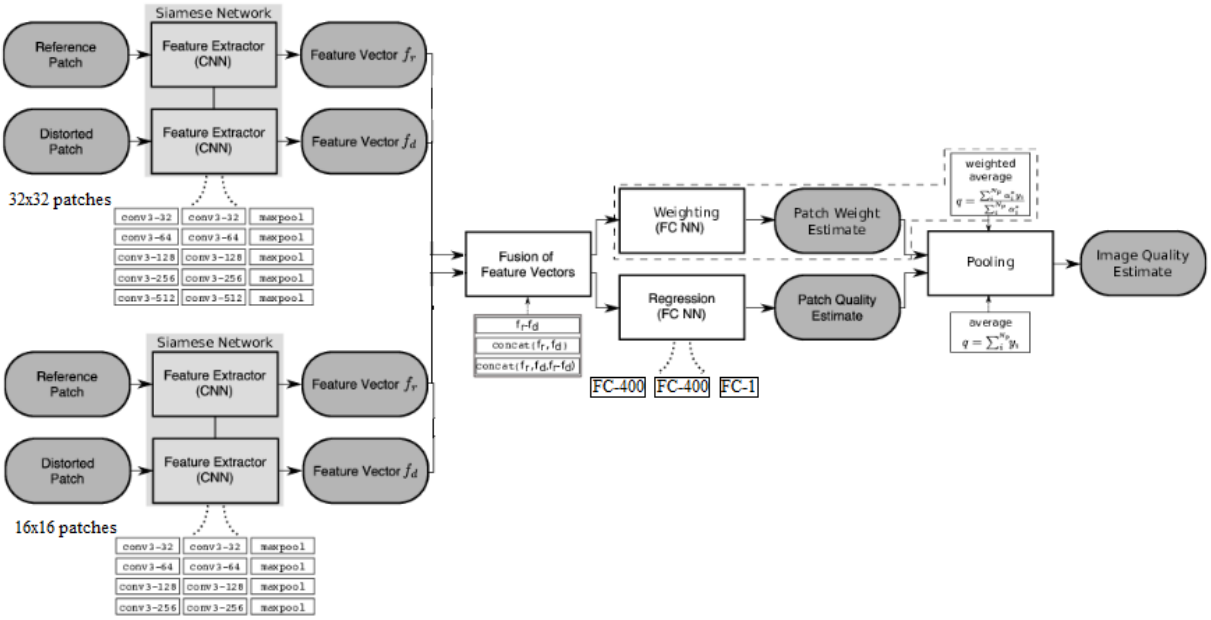


Figure 6: The MS-WaDIQaM architecture

The MS-WaDIQaM model performs better on the training database and on LIVE but is slightly worse than the baseline architecture for the other databases. Since LIVE is the most similar to TID2013 these results indicate that the model slightly suffers from overfitting compared to the reference method. The performance differences are not very large and it is difficult to establish definitively if one of the two architectures is really superior to the other.

Table VI: Results obtained for the MS-WaDIQaM model

Database	MS-WaDIQaM - Spearman	MS-WaDIQaM - Pearson
TID2013	0.9406	0.9463
LIVE	0.9014	0.8929
CSIQ	0.8953	0.8813
JPEG_AI	0.8042	0.8173
JPEG_XL	0.8365	0.8104

Table VII: Spearman correlation coefficients obtained for the models aggregated with bagging. The models used are MS-WaDIQaM, WaDIQaM and WaDIQaM with dropout rates 40% and 60%, added to the aggregation in this order.

Database	MS-WaDIQaM	Two models	Three models	Four models
TID2013	0.9406	0.9366	0.9388	0.9407
LIVE	0.9014	0.8998	0.9028	0.8996
CSIQ	0.8953	0.9202	0.9249	0.9239
JPEG_AI	0.8042	0.8062	0.8129	0.822
JPEG_XL	0.8365	0.8778	0.8799	0.8795

4) Bagging: Bagging is the practice of training multiple predictors to perform the same task then combining their outputs by averaging in the case of regression or performing a majority vote in the case of classification [16]. Bagging works best when training different predictors gives different results: different decision boundaries in the case of classification and different error locations for regression. In the specific case where the results of the regression can be modeled as y_{true} plus some zero mean noise where y_{true} is the correct value, that is when the model has low bias and some variance then averaging the results of several predictors will reduce the variance and improve the predictions. [5] One of the advantages of bagging is that it is very quick and easy to test if it improves the performance of a model for a given task as long as more than one model have been trained. To see if bagging can improve the results of the models their results are averaged then correlation coefficients are recomputed on the aggregated scores. The models used in this test are MS-WaDIQaM, WaDIQaM and WaDIQaM with dropout rates 40% and 60%. They are added to the aggregation in this order. The evaluation results for the aggregated models are shown in tables VII and VIII.

The Pearson coefficients become larger as more models are added to the aggregation for every database except LIVE where they remain almost unchanged. The Spearman coefficients remain the same for TID2013 and LIVE but become larger for the other databases. The overall performance of the aggregated IQA metric is greatly improved by the bagging process. This result confirms that the IQA networks considered in this project have a low bias and produce different results on different training runs even with similar training data and network architectures and shows that bagging is an effective way of improving the performance of IQA metrics based on machine learning.

E. Results

After all of the experiments detailed in the previous subsections two new architectures achieve results that are comparable to the baseline architecture: MS-WaDIQaM and the aggregated metric

Table VIII: Pearson correlation coefficients obtained for the models aggregated with bagging. The models used are MS-WaDIQaM, WaDIQaM and WaDIQaM with dropout rates 40% and 60%, added to the aggregation in this order.

Database	MS-WaDIQaM	Two models	Three models	Four models
TID2013	0.9463	0.9422	0.9447	0.9462
LIVE	0.8929	0.8855	0.8909	0.8872
CSIQ	0.8813	0.9202	0.9117	0.9118
JPEG_AI	0.8173	0.8287	0.8319	0.8397
JPEG_XL	0.8104	0.8438	0.8453	0.8416

Table IX: Spearman correlation coefficients obtained for MS-WaDIQaM, the models aggregated with bagging, baseline WaDIQaM and the reference paper's WaDIQaM model

Database	MS-WaDIQaM	Aggregated	WaDIQaM	Reference WaDIQaM
TID2013	0.9406	0.9407	0.9349	0.946
LIVE	0.9014	0.8996	0.8944	0.9360
CSIQ	0.8953	0.9239	0.9156	0.9310
JPEG_AI	0.8042	0.822	0.8025	0.7313
JPEG_XL	0.8365	0.8795	0.8492	0.8428

Table X: Pearson correlation coefficients obtained for MS-WaDIQaM, the models aggregated with bagging, baseline WaDIQaM and the reference paper's WaDIQaM model

Database	MS-WaDIQaM	Aggregated	WaDIQaM	Reference WaDIQaM
TID2013	0.9463	0.9462	0.9382	0.9400
LIVE	0.8929	0.8872	0.8878	0.9390
CSIQ	0.8813	0.9118	0.9043	0.9320
JPEG_AI	0.8173	0.8397	0.8253	0.8102
JPEG_XL	0.8104	0.8416	0.8226	0.8388

produced by bagging. The scores of these two architectures as well as the baseline and the scores produced by the WaDIQaM paper's model [17] trained on TID2013 are shown in tables IX and X. The implementation provided by the authors of the paper was slightly modified to use the same number of patches as this project's implementation to compute the quality estimates of the images.

Let us first leave the aggregated metric aside and focus on the results obtained by a single network at a time. We can see that the reference paper's model performs better on TID2013, LIVE and CSIQ but worse on JPEG_AI and comparably on JPEG_XL. Since it performs better on the databases that are the most similar to the training set and worse on those that are different we can conclude that the reference paper's model overfits the data slightly. The models trained during this project used the JPEG_AI database to compute the validation loss and determine when to stop training whereas the reference paper's model uses a subset of TID2013 that is not used for training. This can explain the difference in performance of the models and indicates that TID2013 does not contain enough data to train metrics that generalize to unseen distortions. The good performance obtained by all the models on CSIQ and LIVE which contain different images than TID2013 but similar distortion types indicates that generalizing to unseen images is less of a problem.

The aggregated metric clearly outperforms the others on JPEG_AI and JPEG_XL and beats this project's models on CSIQ but still performs worse than the reference paper's model on the training set and the databases that are the most similar to it. From these results it appears that aggregating the results of several models trained on the same data improves the generalization power compared to each individual model but does not improve the results obtained on the training set.

The apparent lack of generalization power of all the models when it comes to unseen image distortion types indicates that without using more data than TID2013 any model will have trouble when being used as the loss function of an image compression network since this process is bound to create new distortion types as it progresses. To properly assess the severity of this problem more training data is required in order to create at least three reasonably sized image databases that differ in both reference images and distortion types that could be used as training, validation and performance evaluation datasets. Because of this it seems that proxy IQA metrics based on non differentiable but proven to be effective objective metrics such as VMAF [18] would be the better choice to train image compression networks since the number of training examples available depends only on the processing power and storage capacity of the machine that trains the networks.

(a) Original image

(b) Image produced by the autoencoder

Figure 7: Results of training the autoencoder using MSE as the loss function

III. Image compression using trained IQA metrics

A well known problem of neural networks is their poor resistance to adversarial attacks. When using an IQA metric as its loss function an image compression network is constructing adversarial examples to try and maximize the score given by the metric. This can result in low quality images that are given a high score by the IQA metric. When using a proxy metric it is easier to deal with this issue because the noisy images can be added to the training dataset at any point during training with the correct score. [2] This is obviously not the case for MOS based metrics where many humans need to give their opinions on an image in order to obtain a correct score. This section explores different ways of using the IQA metrics in the loss function of a compression network to find one that effectively mitigates this problem.

A. Methodology

In order to test the different methods of using the IQA models an autoencoder architecture was trained first on MSE to establish a baseline then on different IQA metrics it is referred to as the reference solution. The resulting images are then compared visually and with three objective metrics: MS-SSIM, PSNR and MSE when the quality of the images is at least comparable to those produced by the baseline.

The autoencoder which is shown in figure 8 consists of three convolutional layers with 32 filters each. The first layer has a stride of 4 and a kernel size of 9 by 9 while the two other layers have a stride of 2 and a kernel size of 5 by 5. It is trained on the reference images of the TID2013 database. The objective metrics are then computed on the training images.

Once the IQA metrics are proven to work on a simple autoencoder more sophisticated compression networks can be trained with the same loss function.

B. Results

In this subsection the results of different ways of adding the IQA metrics to the loss function of the autoencoder are presented in the order in which they were tried during the project. At first the quality estimates used for training the model were computed on a regular grid of patches of size 32 by 32. Using this method an image of size 384 by 512 such as the reference images of the TID2013 database is transformed into a 12 by 16 array of 32 by 32 patches. Unless specified otherwise this is the method used to train the autoencoder.

The autoencoder architecture used in this section is very simple and is not intended to be an actual image compression solution. Because of its simplicity it does not try to minimize the entropy

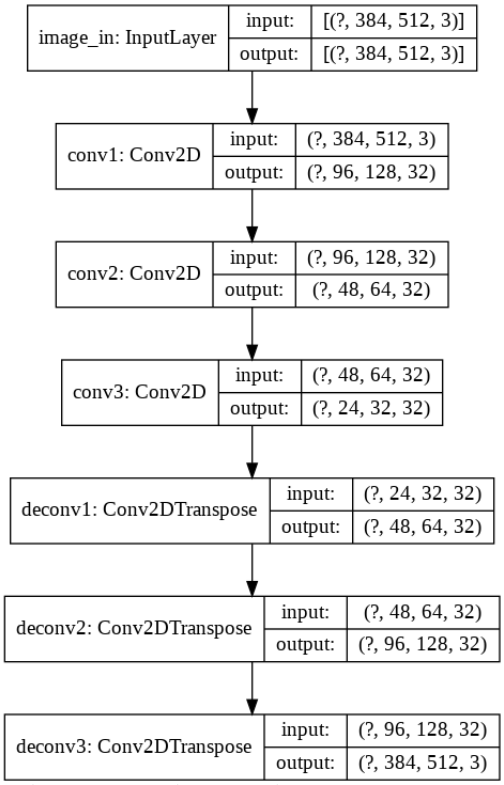


Figure 8: The autoencoder architecture used to test the performance of WaDIQaM based loss functions

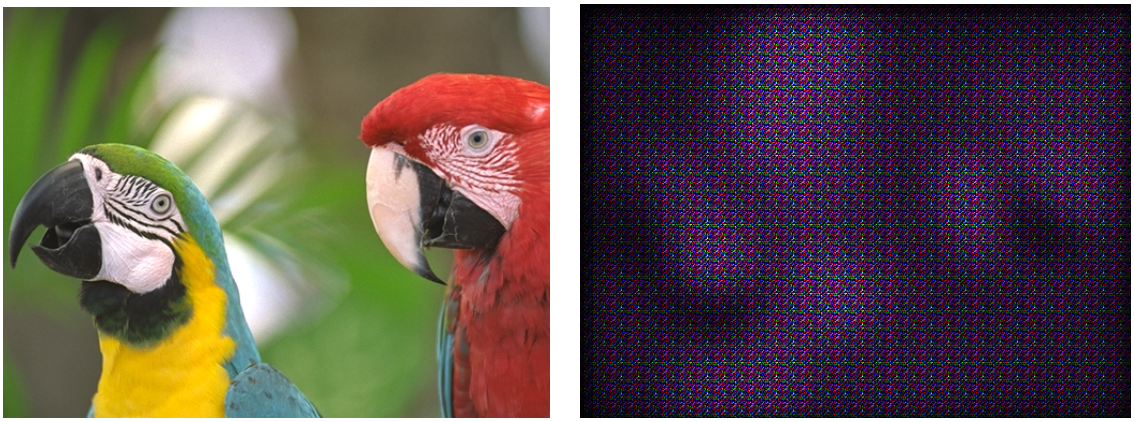
of the latent representation and any measure that depends on it such as bits per pixel would not be meaningful. Its only goal is to prove that using a neural network trained on subjective scores to produce quality estimates used in its loss function can be better than using only MSE. The deciding factor in establishing the usefulness of the IQA metric is the quality of the images produced by the autoencoder as defined by the objective metrics that are computed on them.

1) IQA metric only: The first way of adding the metric to the loss function is to simply use it as the loss function. The network is then trained to minimize $-$IQA where IQA is the quality assessment estimate produced by the metric. Unfortunately the results of using this simple method are not good. An example image is shown in figure 9.

In this example we can see how easily the IQA metric can be fooled by adversarial examples. This effect is particularly strong in networks that deal with images because the manifold of natural images is an extremely small fraction of the total image space. To get an idea of just how enormous the image space is one can try to generate completely random 384 by 512 grayscale images ,which assuming a pixel range of 256 values gives $256^{384\ x\ 512} = 256^{196,608}$ different possibilities, and see how long it takes to generate an image that looks like a photograph. Therefore a vast amount of the image space will never be sampled in training sets which means that the maximum value that any network trained on images can give is probably on a point in image space that has nothing to do with any of the training examples.

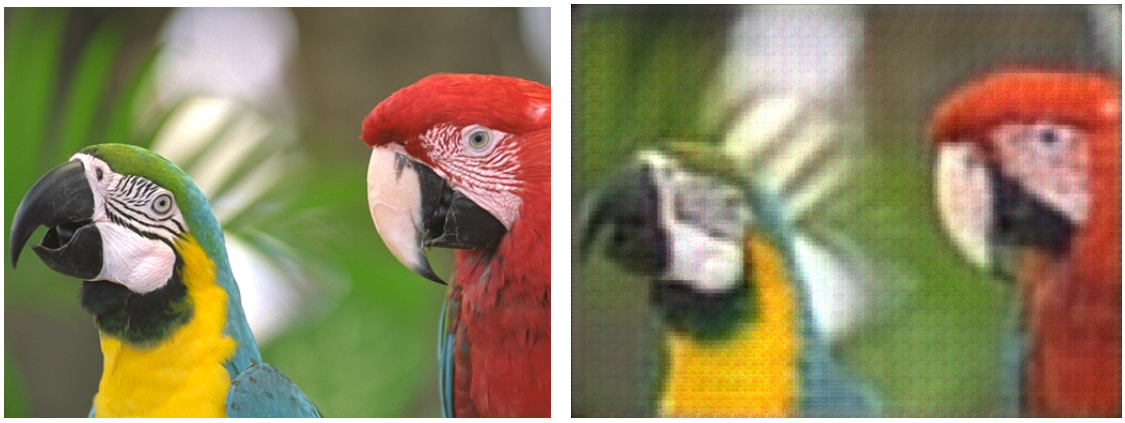
2) IQA and MSE: To improve the results of the autoencoder MSE is combined with the quality estimates and the network is trained to minimize λMSE $-$ IQA. This is a way of constraining the network to explore only regions of the image space that are close to the original image. As shown in figure 12 the resulting image is recognizable but a noise pattern similar to the one that is seen in the previous result is still visible.

3) Constraining the output range of the metric: A common problem in both using only the IQA score as the loss and combining it with MSE is that the quality estimates produced by the model are not bounded. For instance the image shown in figure 9 receives a score of several thousands while



(a) Original image *(b) Image produced by the autoencoder*

Figure 9: Results of training the autoencoder on the IQA metric only



(a) Original image *(b) Image produced by the autoencoder*

Figure 10: Results of training the autoencoder on the IQA metric combined with MSE

the network was trained on scores ranging from 0 to 9. A way of improving this situation is to add a function on the output of the network which prevents the scores from going outside the range of the training scores. To this end the baseline architecture of the IQA metric was modified and a sigmoid function $S(x) = \frac{9}{1+e^{-x}}$ was added at the end of the network as shown in figure 11. The modified metric is trained in the same way as the baseline and achieves the same performance. Similarly to the previous method the network is trained to minimize $\lambda\text{MSE} - \text{IQA}$ With bounded scores the metrics can no longer dominate the MSE completely and the images produced by the autoencoder shown in

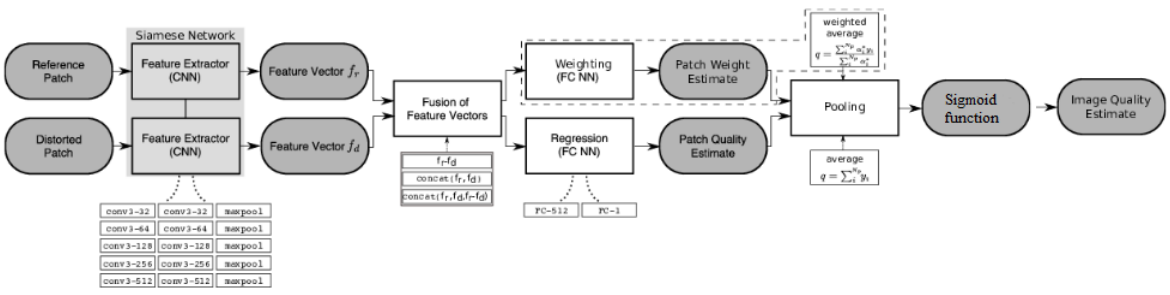
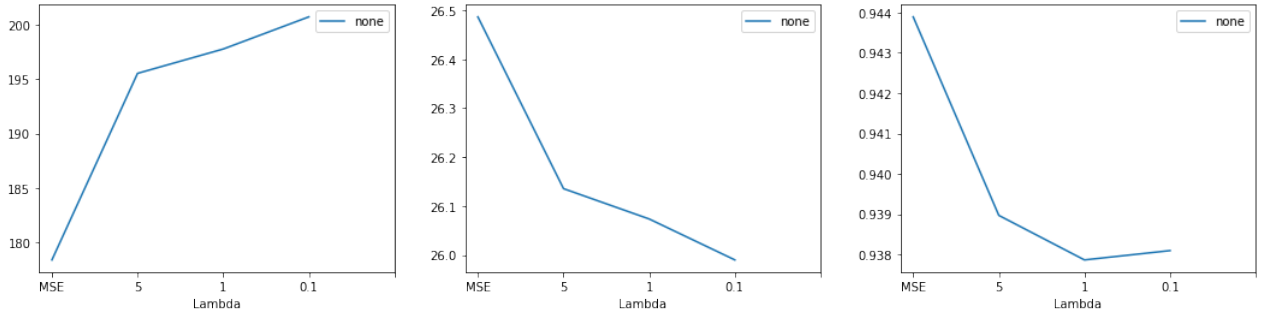


Figure 11: The baseline architecture with a sigmoid function added to constrain the scores to a given range

(a) Original image *(b) Image produced by the autoencoder*

Figure 12: Results of training the autoencoder on the IQA metric with constrained scores combined with MSE



(a) MSE as a function of lambda *(b) PSNR as a function of lambda* *(c) MS-SSIM as a function of lambda*

Figure 13: Objective metrics computed on the images obtained by training the autoencoder with the loss $\lambda MSE - IQA$. A value of MSE on the x axis means that only MSE was used as the loss function.

figure 12 start to look as good as those produced by the reference solution shown in figure 7.

With images of this quality is starts to become interesting to look at objective metrics to see how the use of the IQA metrics affects the quality of the images produced by the autoencoder. They are presented in figure 13.

From this graph it becomes apparent that the IQA metric has a negative impact on the performance of the autoencoder. A probable explanation is that the IQA network is too sensitive to adversarial perturbations and because of this the only contribution of the metric to the training process is adding an adversarial noise pattern similar to the one that can be seen in figures 9 and 12 to the image because it thinks that it improves its quality. A quick verification by looking at the difference between an image produced by the reference solution and one produced by the network trained using the IQA metric does in fact reveal those noise patterns as shown in figure 14. They are particularly present in the low frequency parts of the image, for instance the top left corner, which indicate that it is harder to fool the IQA metric in high frequency regions where the differences between the two images do not seem to follow the same structured pattern.

4) Integrating mean squared error in the IQA metric: When using the solution presented in III-B3 one needs to set the lambda parameter to train the autoencoder which represents extra work. If the MSE that is used to train the autoencoder could be integrated to the IQA metric the need for the lambda parameter would disappear and the autoencoder could simply be trained using the IQA metric only as its loss function instead of $\lambda MSE - IQA$ while keeping the benefits of MSE. To test this solution two new IQA architectures based on the baseline WaDIQaM architecture were built: the first

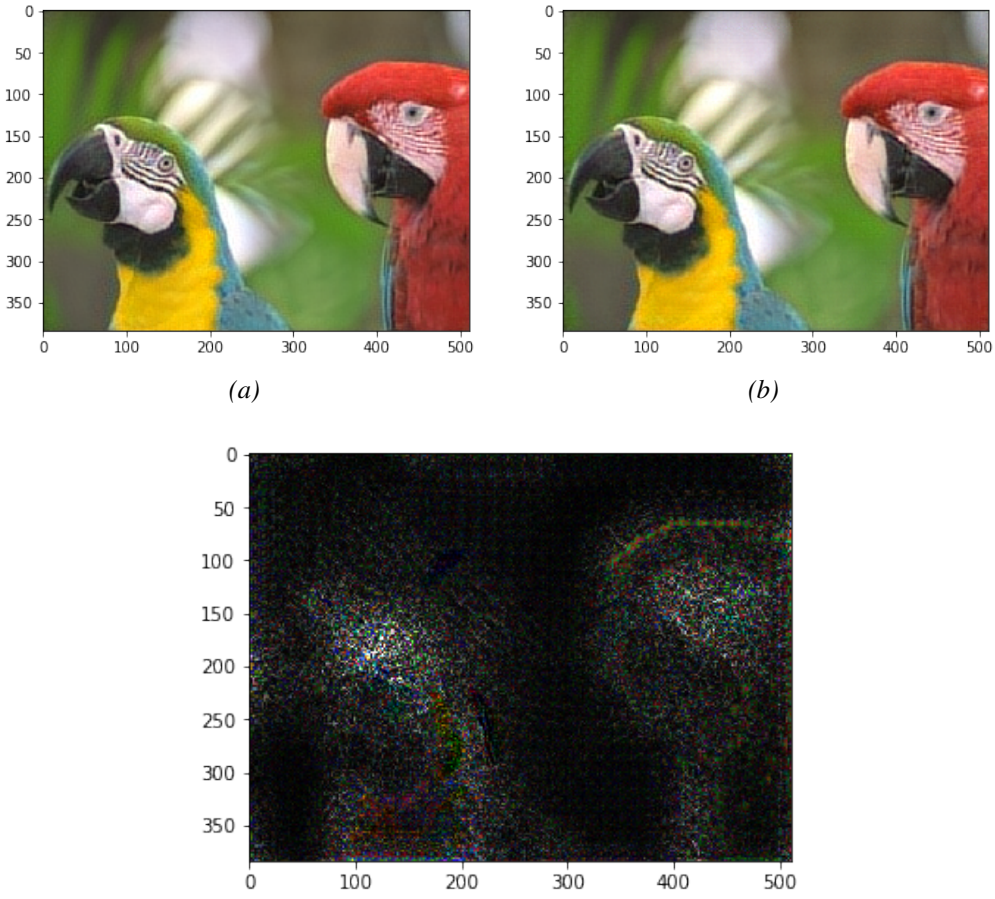


Figure 14: 14a shows the parrots image generated by the reference solution. 14b shows the same image generated by the autoencoder described in subsubsection III-B3. 14c shows the difference between the two images multiplied by ten to be more visible.

one appends the mean squared error between each patch of the distorted image and the corresponding patch of the reference image to the feature vector created by the feature extraction part of the network and is shown in figure 15 while the second one adds two fully connected layers at the end of the network that take as input the final score given to the distorted image and the mean squared error between the full distorted and reference images to output the image quality estimate and is shown in figure 17.

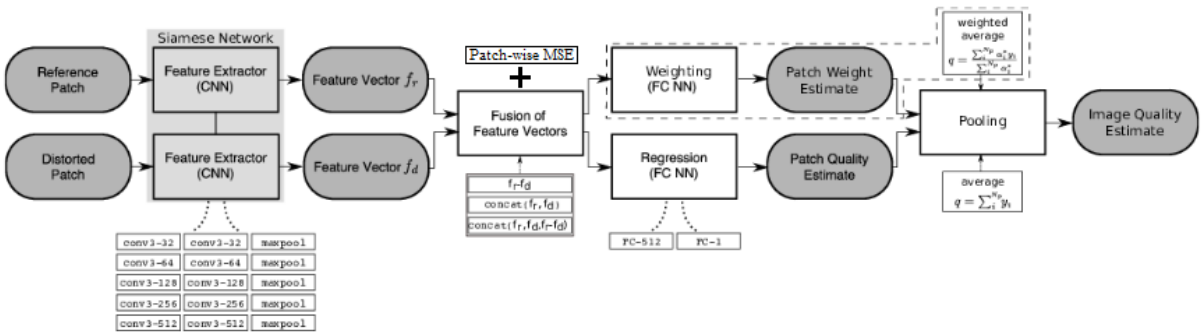


Figure 15: WaDIQaM with patch wise MSE added to the feature vectors of each patch

Figures 16 and 18 show the results of training the autoencoder using these new architectures as

(a) Original image *(b) Image produced by the autoencoder*

Figure 16: Results of training the autoencoder using the IQA metric with patch wise MSE added to the feature vectors of each patch as the loss function

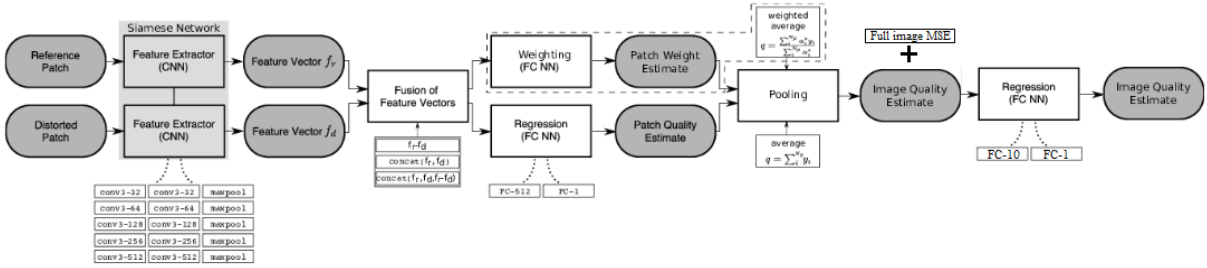
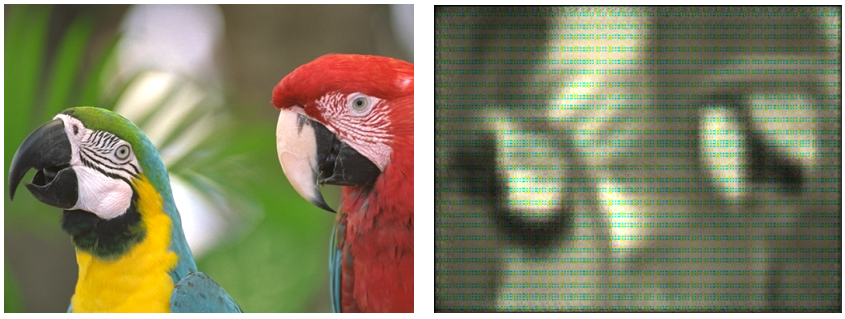


Figure 17: WaDIQaM with the full image MSE added at the end of the network to change the prediction results

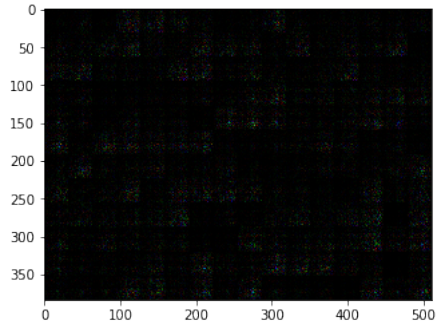
the loss function: LOSS = -IQA. The results obtained this way are strictly worse than the ones of subsubsection III-B3 this idea was therefore deemed inconclusive and abandoned.

5) Adversarial training: To make the IQA network more robust to adversarial perturbations a method similar to DeepFool [19] is used. First an IQA metric is trained as usual. Then every distorted image of the TID2013 database is slowly transformed by gradient ascent on the IQA network until it receives a score that is close to maximal. Finally the metric is trained again for five epochs with a learning rate reduced twenty times on both the original database and the adversarial images produced in the previous step. This process is called a round of adversarial training and its results can be seen in figure 19. The adversarial training process makes the IQA metric harder to fool as the magnitude of the perturbation needed to trick it into giving an excellent score grows.

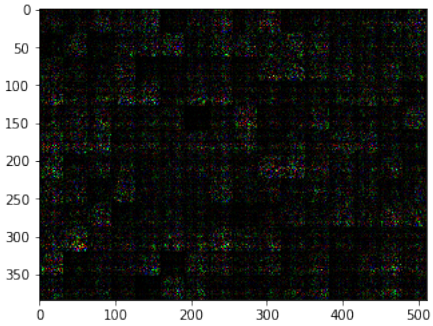


(a) Original image *(b) Image produced by the autoencoder*

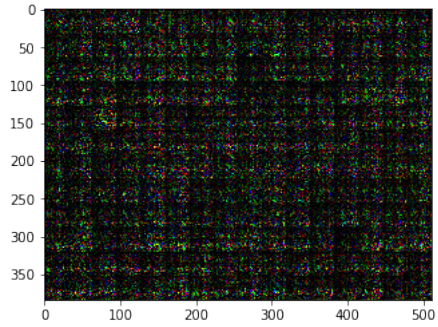
Figure 18: Results of training the autoencoder using the IQA metric with MSE between the reference and distorted image added at the end of the network as the loss function



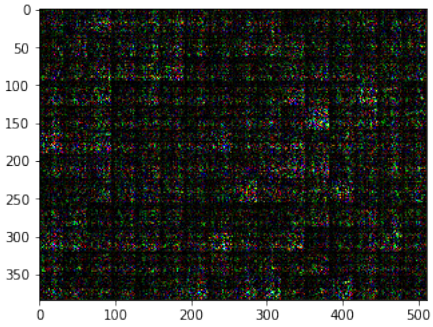
(a) First adversarial round mean absolute noise: 0.02



(b) Second adversarial round mean absolute noise: 0.2



(c) Third adversarial round mean absolute noise: 0.4



(d) Sixth adversarial round mean absolute noise: 0.46

Figure 19: A visualization of the magnitude of the perturbation needed to make the network output a score of 8.75 from an image of score 5. The noise is multiplied by 100 to be visible.

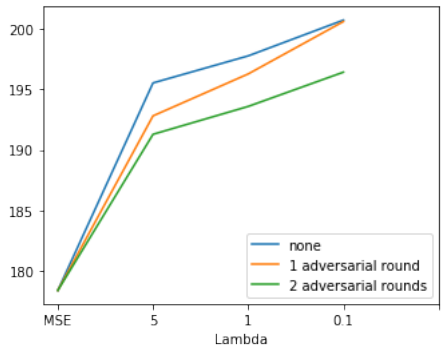
The metrics obtained by performing this procedure are then used to train the autoencoder in the same way as in the previous subsubsection. The objective metrics computed on the images generated by those new models can be seen in figure 20.

The new metrics perform slightly better than the one that did not receive adversarial training but they are still harming the performance of the autoencoder compared to simply using MSE. This is clearly not sufficient and a new approach is needed. We can see in the noise patterns of figure 19 that extracting the patches in a regular grid affects the way in which the adversarial perturbation is computed: the grid of patches is clearly visible in the noise itself.

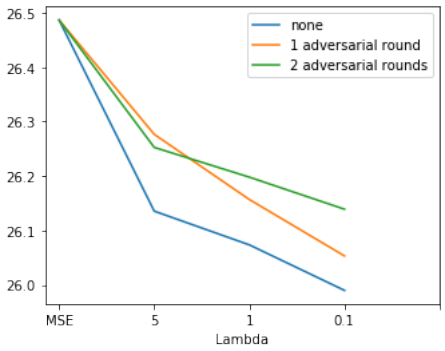
6) Random patch selection: To verify the influence of the grid patch selection on the results of the autoencoder a new version is trained by selecting random patches to compute the image quality estimate at every step of training. Adversarial training is also performed on the metric again since the noise patterns are different. Using this method the adversarial rounds were harder to compute and most images could not obtain scores higher than 7.5 which is why the target score for adversarial images was set to this value. This also affects the adversarial noise itself which as can be seen in figure 21 is no longer made of quasi identical squares but resembles the input image. The mean absolute magnitude of the noise also increased dramatically from only 0.02 to 3.1 when no adversarial training is performed.

Once again the autoencoder is trained using the new metrics and selecting random patches at each step to compute the image quality estimates. The results of this new training round are shown in figure 22.

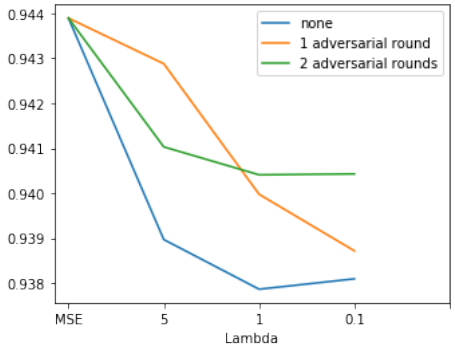
Using the IQA metric in this way improves both PSNR and MS-SSIM for the images generated by the autoencoder for high values of lambda. As can be expected lowering the value of lambda cause



(a) MSE as a function of lambda



(b) PSNR as a function of lambda

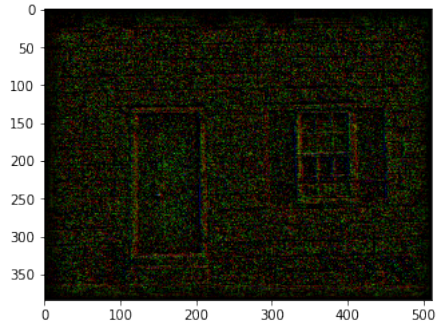


(c) MS-SSIM as a function of lambda

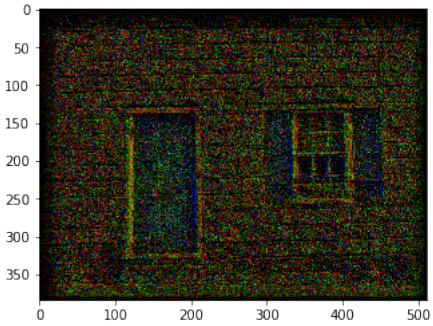
Figure 20: Objective metrics computed on the images obtained by training the autoencoder with the loss $\lambda MSE - IQA$ after adversarial training. A value of MSE on the x axis means that only MSE was used as the loss function.

the MSE to grow since more importance is given to IQA metric compared to MSE. Performing more adversarial rounds seems to make the network reach better PSNR and MS-SSIM values while reducing the impact of using the IQA metric on the MSE. The optimal value of lambda seems to change with the number of adversarial rounds from close to one when no adversarial training has been performed to close to five after the two adversarial training rounds. This might mean that increasing the amount of noise needed to fool the network has in part the same effect as reducing the value of lambda and giving more importance to the IQA metric over MSE. When lambda is set to a value smaller than one the training procedure becomes unstable and produces unpredictable results. The cause of this phenomenon is probably that giving too much importance to a metric that acts on random parts of the image at every step causes it to undo the changes that it made at the previous step and hinders the convergence of the model.

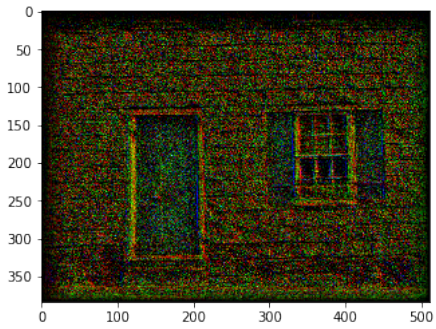
A visual comparison of the results obtained by the reference solution and the final model obtained after adversarial training and using random patches with lambda equal to five is shown in figure 23. The images produced by the two autoencoders are very similar but the proposed solution preserves some high frequency details better while the MSE autoencoder handles colors better.



(a) First adversarial round mean absolute noise: 3.1

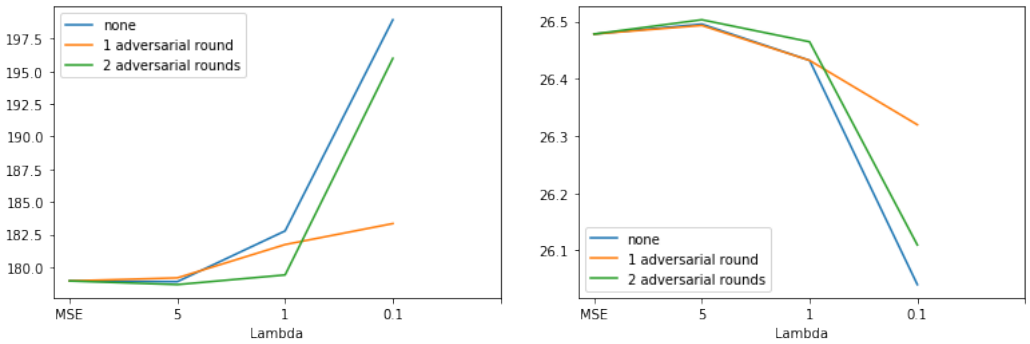


(b) Second adversarial round mean absolute noise: 5.1

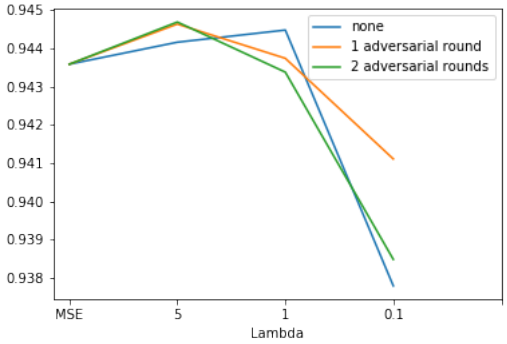


(c) Third adversarial round mean absolute noise: 6.1

Figure 21: A visualization of the magnitude of the perturbation needed to make the network output a score of 7.5 from an image of score 5 using random patches to compute it. The noise is multiplied by 10 to be visible.



(a) MSE as a function of lambda *(b) PSNR as a function of lambda*



(c) MS-SSIM as a function of lambda

Figure 22: Objective metrics computed on the images obtained by training the autoencoder with the loss $\lambda MSE - IQA$ after adversarial training using random patches instead of a fixes grid. A value of MSE on the x axis means that only MSE was used as the loss function.

(a) Proposed autoencoder *(b) Original image* *(c) MSE autoencoder*

(d) Proposed autoencoder *(e) Original image* *(f) MSE autoencoder*

Figure 23: A comparison of the images produced by the proposed autoencoder as described in III-B6 using lambda equal to five and those produced by the reference solution trained with MSE. The most notable differences are the top left of the necklace in the first comparison and the face of the child in the bottom right in the second. The color of the safety vests is also brighter in the MSE image.

IV. Conclusion

We have studied the use of different techniques to train deep neural networks for full reference image quality assessment including using batch correlation as the loss function, multi-scale networks and averaging the results of several networks. We discovered that averaging the results of different models trained on the same can lead to an increase of the generalization power of the metrics by a significant amount, increasing the Pearson correlation coefficient by 2.7% and the Spearman correlation coefficient by 2.2% on the JPEG_AI database which consists of both new images and distortion types that were not seen during training. On JPEG_XL the increase was of 3.7% Pearson and 4.9% Spearman. The multi-scale architecture that we tried did not outperform the baseline on validation data but it did manage to obtain the same performances while being better on the training set which makes it an interesting candidate for training with more data in the future.

We have used the IQA metrics built using the techniques mentioned above inside the loss functions of autoencoders to show that they can lead to an increase in quality over using MSE only. We discovered that randomness is a crucial part of building metrics that are robust enough to avoid being fooled into giving high scores by the optimization procedure using adversarial perturbations and that separating an image into a regular grid of patches works very well when computing quality estimates for natural images and does not work at all when optimizing an autoencoder. We also learned that constraining the IQA metric to force its predictions to stay in the range that was used in the training data helps in stabilizing the training procedure and that adversarial training is a good way of improving the resistance of a neural network to adversarial perturbations.

Using all the techniques that we discovered during the course of this project we managed to show that it is possible to obtain better results by using an IQA metric trained on subjective opinion scores than by using only mean squared error when training an autoencoder on images. Nonetheless we have also noticed that the amount of data available will always be a problem for metrics train on mean opinion scores whereas metrics that are trained to approximate non differentiable objective IQA metrics do not suffer from this because new scores can always be computed by the machine that is training the networks.

Further work should focus on applying these techniques on an image compression network that is trained to minimize the size of the compressed images it produces to see if the use of IQA metrics can also be beneficial in this context.

REFERENCES

[1] Z. Wang, A. C. Bovik, and L. Lu, "Why is image quality assessment so difficult?" in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, May 2002, pp. IV–3313–IV–3316.

[2] L.-H. Chen, C. G. Bampis, Z. Li, A. Norkin, and A. C. Bovik, "Proxiqa: A proxy approach to perceptual optimization of learned image compression," 2019.

[3] S. Bosse, D. Maniry, K. Müller, T. Wiegand, and W. Samek, "Deep neural networks for no-reference and full-reference image quality assessment," *CoRR*, vol. abs/1612.01697, 2016. [Online]. Available: http://arxiv.org/abs/1612.01697

[4] A. Fawzi, S. Moosavi-Dezfooli, and P. Frossard, "The robustness of deep networks: A geometrical perspective," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 50–62, Nov 2017.

[5] T. G. Dietterich and E. B. Kong, "Machine learning bias, statistical bias, and statistical variance of decision tree algorithms," Technical report, Department of Computer Science, Oregon State University, Tech. Rep., 1995.

[6] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, and C.-C. J. Kuo, "Image database tid2013: Peculiarities, results and perspectives," *Signal Processing: Image Communication*, vol. 30, pp. 57 – 77, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596514001490

[7] L. C. H.R. Sheikh, Z. Wang and A. Bovik, "Live image quality assessment database release 2." [Online]. Available: http://live.ece.utexas.edu/research/quality

[8] E. C. Larson and D. M. Chandler, "Most apparent distortion: Full-reference image quality assessment and the role of strategy," *Journal of Electronic Imaging*, vol. 19 (1), 2010.

[9] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[13] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[14] V. Ganapathy and K. L. Liew, "Handwritten character recognition using multiscale neural network training technique," 2008.

[15] A. B. Geva, "Scalenet-multiscale neural-network architecture for time series prediction," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1471–1482, Nov 1998.

[16] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996. [Online]. Available: https://doi.org/10.1007/BF00058655

[17] S. Bosse, D. Maniry, K. Müller, T. Wiegand, and W. Samek, "Deepiqa implementation," 2016. [Online]. Available: https://github.com/dmaniry/deepIQA

[18] R. Rassool, "Vmaf reproducibility: Validating a perceptual practical video quality metric," in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, June 2017, pp. 1–2.

[19] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *CoRR*, vol. abs/1511.04599, 2015. [Online]. Available: http://arxiv.org/abs/1511.04599