



**ISO/IEC JTC 1/SC29/WG1 N100514**  
**99<sup>th</sup> JPEG Meeting, Online, 24-28 April 2023**

**ISO/IEC JTC 1/SC 29/WG 1  
(& ITU-T SG16)**

## **Coding of Still Pictures**

### **JBIG JPEG**

Joint Bi-level Image  
Experts Group

Joint Photographic  
Experts Group

**TITLE:** JPEG AI Common Training and Test Conditions

**SOURCE:** WG1

**PROJECT:** JPEG AI (ISO/IEC 6048)

**STATUS:** Approved

**REQUESTED ACTION:** For distribution

**DISTRIBUTION:** Public

**Contact:**

ISO/IEC JTC 1/SC 29/WG 1 Convener – Prof. Touradj Ebrahimi  
EPFL/STI/IEL/GR-EB, Station 11, CH-1015 Lausanne, Switzerland  
Tel: +41 21 693 2606, Fax: +41 21 693 7600, E-mail: [Touradj.Ebrahimi@epfl.ch](mailto:Touradj.Ebrahimi@epfl.ch)



INTERNATIONAL ORGANISATION FOR STANDARDISATION

INTERNATIONAL ELECTROTECHNICAL COMMISSION

---

## **JPEG AI – LEARNING-BASED IMAGE CODING COMMON AND TRAINING TEST CONDITIONS**



# CONTENTS

<b>1</b>	<b>SCOPE .....</b>	<b>5</b>
<b><i>PART A – COMMON TRAINING AND TEST CONDITIONS FOR HUMAN VISUALIZATION .....</i></b>		<b><i>6</i></b>
<b>2</b>	<b>JPEG AI DATASET.....</b>	<b>7</b>
<b>3</b>	<b>EVALUATION PROCEDURE.....</b>	<b>8</b>
<b>4</b>	<b>TARGET RATES .....</b>	<b>8</b>
<b>5</b>	<b>OBJECTIVE QUALITY EVALUATION .....</b>	<b>9</b>
5.1	MS-SSIM DEFINITION AND COMPUTATION .....	9
5.2	IW-SSIM DEFINITION AND COMPUTATION .....	9
5.3	VMAF DEFINITION AND COMPUTATION .....	10
5.4	VIF DEFINITION AND COMPUTATION.....	10
5.5	PSNR-HVS-M DEFINITION AND COMPUTATION .....	10
5.6	NLPD DEFINITION AND COMPUTATION .....	10
5.7	FSIM DEFINITION AND COMPUTATION .....	10
<b>6</b>	<b>SUBJECTIVE QUALITY EVALUATION .....</b>	<b>11</b>
<b>7</b>	<b>COMPLEXITY EVALUATION .....</b>	<b>12</b>
<b>8</b>	<b>ANCHORS GENERATION .....</b>	<b>12</b>
8.1	JPEG ANCHOR .....	13
8.2	JPEG 2000 ANCHOR.....	14
8.3	HEVC INTRA ANCHOR.....	15
8.4	VVC INTRA ANCHOR.....	16
8.5	SOFTWARE FOR ANCHOR GENERATION .....	18
<b>9</b>	<b>NAMING CONVENTION FOR DECODED IMAGES AND BITSTREAMS .....</b>	<b>18</b>
<b>10</b>	<b>10 BIT IMAGES .....</b>	<b>19</b>
<b>11</b>	<b>EVALUATION FRAMEWORK AND RESULTS REPORTING TEMPLATE .....</b>	<b>19</b>
<b><i>PART B – COMMON TRAINING AND TEST CONDITIONS FOR COMPUTER VISION TASKS .....</i></b>		<b><i>20</i></b>
<b>12</b>	<b>COMPRESSED DOMAIN IMAGE CLASSIFICATION .....</b>	<b>21</b>
12.1	OBJECTIVE .....	21
12.2	TRAINING AND TEST DATASET .....	21
12.3	ANCHOR GENERATION.....	21
12.4	BITRATES .....	22
12.5	PERFORMANCE METRICS .....	23
12.6	EVALUATION FRAMEWORK AND TESTING PROCEDURE .....	23
12.7	JPEG AI NAMING CONVENTIONS OF IMAGE CLASSIFICATION .....	23
<b><i>PART C – COMMON TRAINING AND TEST CONDITIONS FOR IMAGE PROCESSING TASKS.....</i></b>		<b><i>25</i></b>
<b>13</b>	<b>COMPRESSED DOMAIN SUPER-RESOLUTION .....</b>	<b>26</b>
13.1	OBJECTIVE .....	26
13.2	TRAINING AND TEST DATASET .....	26
13.3	ANCHORS .....	26
13.4	BITRATES .....	27

13.5	PERFORMANCE METRICS .....	27
13.6	EVALUATION FRAMEWORK AND TESTING PROCEDURE .....	27
13.7	NAMING CONVENTIONS FOR SUPER-RESOLUTION .....	28
<b>14</b>	<b>COMPRESSED DOMAIN DENOISING .....</b>	<b>28</b>
14.1	OBJECTIVE .....	28
14.2	TRAINING AND TEST DATASET .....	28
14.3	ANCHORS .....	29
14.4	BITRATES .....	30
14.5	PERFORMANCE METRICS .....	30
14.6	EVALUATION FRAMEWORK AND TESTING PROCEDURE .....	31
14.7	NAMING CONVENTIONS FOR DENOISING.....	31
<b>REFERENCES .....</b>		<b>32</b>

# JPEG AI LEARNING-BASED IMAGE CODING COMMON TRAINING AND TEST CONDITIONS

## 1 Scope

The scope of the JPEG AI is the creation of a learning-based image coding standard offering a **single-stream, compact** compressed domain representation, targeting both **human visualization**, with significant compression efficiency improvement over image coding standards in common use at equivalent subjective quality, and effective performance for **image processing and computer vision tasks**, with the goal of supporting a **royalty-free baseline**.

This document describes the Common Training and Test Conditions (CTTC) for the JPEG AI image coding experiments. The main objectives of this document are:

- Define the common datasets that should be used in the evaluation of learning-based image coding solutions.
- Define the anchors that should be used to comparatively evaluate the performance of learning-based image coding solutions.
- Define the coding conditions, especially the target bitrates that an anchor or learning-based image coding solution should be able to achieve.
- Define the subjective evaluation procedure to perceptually evaluate all decoded images quality for standard reconstruction, namely the anchors and the learning-based image codecs.
- Define the quality, accuracy and complexity metrics for standard reconstruction, image processing and computer vision tasks that can be used to reliably evaluate the performance of a learning-based image codec.

These common training and test conditions should be used to evaluate different aspects of learning-based image codecs. The CTTC specification should be followed in all the experiments made by participants.

---

## **PART A – COMMON TRAINING AND TEST CONDITIONS FOR HUMAN VISUALIZATION**

## 2 JPEG AI Dataset

The JPEG AI dataset was created for the training, validation and performance evaluation of learning-based image coding solutions. The data set, described in this section, is used for image reconstruction task. Training and test conditions for image processing and computer vision tasks are described in later sections of this document, but data sets for all tasks are organized on a same way. This JPEG AI dataset is freely available with CC0 licensing to all JPEG AI proponents and must be used by all in the creation of learning-based image coding models; moreover, performance evaluation results should be reported for the model trained using this dataset for all contributions. The JPEG AI data set consists of two parts: natural and synthetic. More parts can be added in a future addressing JPEG AI standard requirements. Since training and testing procedure for “synthetic” content is under final verification, only “natural” part of dataset is used. The JPEG AI dataset is organized according to:

- **Training dataset:** The training dataset provides a set of images to create a model suitable for a learning-based image codec solution.
- **Validation dataset:** The validation dataset provides a set of images to be used during the training to validate the convergence of the training algorithm employed by some learning-based image codec solution.
- **Test dataset:** The test dataset cannot be used neither for training or for validation and will only be used to evaluate the final performance of learning-based image coding solutions. The test dataset includes 50 images to avoid overfitting, allows to track the performance improvements meeting to meeting, includes a wide variety of contents and resolutions and allows to cover a reasonably wide range of quality for the target bitrates specified here.

The diversity of the images contained in the JPEG AI training and validation dataset is high, namely in terms of their characteristics, such as content and spatial resolution. The JPEG AI dataset has the following characteristics:

- Format – PNG images (sRGB color space);
- Spatial resolution – from 256×256 to 8K (8 bit);
- Training/validation/test dataset: 5264/350/50 images.

The number of images allows for an efficient training/validation and is typically larger than the number of images used in previously available datasets. The training and validation dataset is available at <sftp://jpeg-ai@amalia.img.lx.it.pt>, password to be given by request (contact: [joao.ascenso@lx.it.pt](mailto:joao.ascenso@lx.it.pt)).

Detailed information about test set images can be found in ANNEX 1. It is recommended to verify md5 check sum after downloading JPEG AI test set to ensure no corruption happened during downloading.



### 3 Evaluation Procedure

Subjective quality evaluation of the proposals are expected to be done by at least two independent labs or/and by a sufficient amount of viewers using crowdsourcing platform, following well-established procedures and based on the decoded test images. Objective performance and complexity assessment reported to the group must also be cross-checked by at least one independent party. In Figure 1, the coding pipeline for learning-based image coding solutions, which is rather straightforward, is presented.

The input of the encoder and the output of the decoder must be in the PNG (sRGB color space) format. Also, the learning-based image encoder and decoder should support the encoding and decoding of images with a bit depth of 8 or 10 bit (note: decoded image bit-depth must be the same of encoder input). Objective image quality is measured with luminance and color-based metrics and the RGB decoded images will be used for subjective quality evaluation. Regarding the objective quality metrics, they should operate at 10 bit bit-depth.

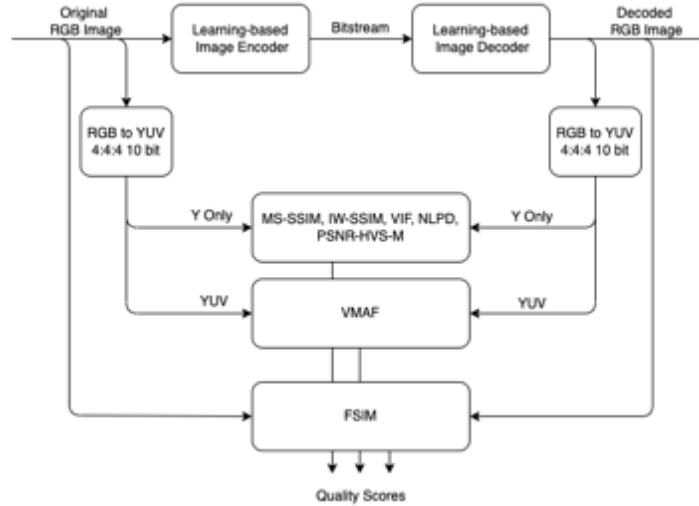


Figure 1. Encoding-decoding pipeline for learning-based image coding solutions.

### 4 Target Rates

Target bitrates for the objective evaluations include 0.03, **0.06**, **0.12**, **0.25**, **0.50**, **0.75**, 1.00, 1.50, and 2.00 bpp. The maximum bitrate deviation above the target bitrate should not exceed 10%. The 0.06, 0.12, 0.25, 0.50, 0.75 bpp bitrates in bold are mandatory and used for BD rate computation. The set of target bitrates for the subjective evaluations is a subset of the target bitrates for the objective evaluations and depends on the spatial complexity of the test images.

The bitrates specified should account for the total number of bits in the encoded file (or files) out of which the decoder can reconstruct a lossy version of the entire image. The main rate metric is the number of bits per pixel (bpp) defined as:

$$BPP = \frac{N\_TOT\_BITS}{N\_TOT\_PIXELS}$$

where  $N\_TOT\_BITS$  is the number of bits for the compressed representation of the image and  $N\_TOT\_PIXELS$  is the number of pixels in the reconstructed image.

## 5 Objective Quality Evaluation

Objective quality testing shall be done by computing several quality metrics, including MS-SSIM, IW-SSIM, VMAF, VIF, PSNR-HVS-M, NLPD and FSIM, between compressed and original image, at the target bitrates mentioned in the previous Section. This Section defines the objective image quality metrics that is used for the assessment of learning-based image coding solutions and techniques (networks and tools). The reference implementation of all objective quality assessment metrics is available at: <https://gitlab.com/wg1/jpeg-ai/jpeg-ai-qaf>. Set of metrics was selected based on their correlation with subjective quality score of natural, camera captures images (final selection of metrics for synthetic images is still under discussion).

### 5.1 MS-SSIM Definition and Computation

Multi-Scale Structural SIMilarity (MS-SSIM) [1] is one of the most well-known image quality evaluation algorithms and computes relative quality scores between the reference and distorted images by comparing details across resolutions, providing high performance for learning-based image codecs. The MS-SSIM [1] is more flexible than single-scale methods such as SSIM by including variations of image resolution and viewing conditions. Also, the MS-SSIM metric introduces an image synthesis-based approach to calibrate the parameters that weight the relative importance between different scales. A high score expresses better image quality.

### 5.2 IW-SSIM Definition and Computation

Information Content Weighted Structural Similarity Measure (IW-SSIM) [2] is an extension of the structural similarity index based on the idea of information content weighted pooling. This metric assumes that when natural images are viewed, pooling should be made using perceptual weights that are proportional to the local information content. Moreover, advanced statistical models of natural image are employed to derive the optimal weights which are combined with multiscale structural similarity measures to achieve the best correlation performance with subjective scores from well known databases.

### 5.3 VMAF Definition and Computation

The Video Multimethod Assessment Fusion (VMAF) metric [3] developed by Netflix is focused on artifacts created by compression and rescaling and estimates the quality score by computing scores from several quality assessment algorithms and fusing them with a support vector machine (SVM). The version 2.2.1 of VMAF metric is used. Even if this metric is specific for videos, it can also be used to evaluate the quality of single images and has been proved that performs reasonably well for learning-based image codecs. Since the metric takes as input raw images in the YUV color space format, the PNG (RGB color space) images are converted to the YUV 4:4:4 10 bits format using FFMPEG (BT.709 primaries). A higher score of this metric indicates better image quality.

### 5.4 VIF Definition and Computation

The Visual Information Fidelity (VIF) [4] measures the loss of human perceived information in some degradation process, e.g. image compression. VIF exploits the natural scene statistics to evaluate information fidelity and is related to the Shannon mutual information between the degraded and original pristine image. The VIF metric operates in the wavelet domain and many experiments found that the metric values agree well with the human response, which also occurs for learning-based image codecs. A high score expresses better image quality.

### 5.5 PSNR-HVS-M Definition and Computation

The PSNR-HVS-M [5] is a simple and effective quality model which uses DCT basis functions and is based on the human visual system (HVS). The model operates with  $8 \times 8$  pixel block of an image and calculates the maximum distortion that is not visible due to the between-coefficient masking. The proposed metric, PSNR-HVS-M, considers the proposed model and the contrast sensitivity function (CSF).

### 5.6 NLPD Definition and Computation

The Normalized Laplacian Pyramid (NLPD) is an image quality metric [6] based on two different aspects associated with the human visual system: local luminance subtraction and local contrast gain control. NLP exploits a Laplacian pyramid decomposition and a local normalization factor. The metric value is computed in the normalized Laplacian domain, this means that the quality of the distorted image relative to its reference is the root mean squared error in some weight-normalized Laplacian domain. A lower score expresses better image quality.

### 5.7 FSIM Definition and Computation

The feature similarity (FSIM) metric [7] is based on the computation of two low level features that play complementary roles in the characterization of the image quality and reflects different aspects of the human visual system: 1) the phase congruency (PC), which is a dimensionless feature that accounts

for the importance of the local structure and the image gradient magnitude (GM) feature to account for contrast information. The color version of the FSIM metric will be used. A high metric value express better image quality.

## 6 Subjective Quality Evaluation

To evaluate coding solutions, a subjective quality assessment methodology is used at different phases of the project (mandatory for CFP stage and other key phases of project). The subjective quality evaluation of the compressed images is performed on a smaller sub-set of the test dataset.

The Double Stimulus Continuous Quality Scale (DSCQS) methodology must be used, where subjects watch side by side the original image and the impaired decoded image and both are scored in a continuous scale. This scale is divided into five equal lengths which correspond to the normal ITU-R five-point quality scale, notably Excellent, Good, Fair, Poor and Bad. This method requires the assessment of both original and impaired versions of each test image. The observers are not told which one is the reference image and the position of the reference image is changed in pseudo-random order. The subjects assess the overall quality of the original and decoded images by inserting a mark on a vertical scale. The vertical scales are printed in pairs to accommodate the double presentation of each test picture.

The subjective test methodology will follow BT500.13 [8] and a randomized presentation order for the stimuli, as described in ITU-T P.910 [9] will be used; the same content is never displayed consecutively. There is no presentation or voting time limit. A training session should be organized before the experiment to familiarize participants with artefacts and distortions in the test images. At least, three training images must be used before actual scoring.

The images used for subjective evaluation are a subset of the test dataset images and its number must be selected depending on the number of codecs to be subjectively evaluated. A minimum of eight images of different characteristics representing JPEG AI use cases must be used. Moreover, four bitrate points covering a wide range of qualities must be used in the subjective evaluation and an expert viewing session may be organized to select bitrates, namely, to cover a significant range of qualities. The images to be used in the subjective evaluation must correspond to crops of the decoded images such that relevant coding artifacts are included.

To perform any subjective test, a semi-controlled crowdsourcing setup framework and/or a more controlled lab environment procedure can be used to show the images according to the DSCQS methodology. The semi-controlled crowdsourcing setup has been proven in the past its reliability, i.e. maintains a low variance of the scores [10]. The QualityCrowd2 [11] software and Amazon Mechanical Turk (or other similar platform) can be used for crowdsourcing. In exceptional cases (for example, due

to the COVID-19 pandemic) subjective evaluation may only be performed with a crowdsourcing approach. The number of subjects must be large enough to draw conclusions in a statistically meaning fashion.

## 7 Complexity Evaluation

The following complexity metrics must be computed:

- Number of parameters (weights) for the size of the largest model. Total number of parameters for all models, including models for all mandatory rate points.
- Model precision, that can assume floating-point, fixed-point or integer with  $N_a$ ,  $N_w$  bits for activation and weights. The  $N_a$ ,  $N_w$  value used must be reported.
- Running time with CPU only (mandatory) and with GPU enabled (recommended), for both encoder and decoder. Number of threads for CPU encoding and decoding must be limited to eight.
- MAC operations, number of Multiply Accumulate operations per sample (kilo), for encoder and decoder (the worst case) operations.
- Minimum GPU Memory Size for decoding. This value should be reported for 8K (7680×4320) images and will be used to assess the possibility of cross-check decoding (inference).
- Minimum GPU Memory Size for encoding. This value should be reported for 8K (7680×4320) images.

An example on how to measure these complexity parameters is available at [12]. Moreover, the specifications of the CPU and GPU (and their model) used to obtain the complexity results according to the aforementioned metrics must be reported. **These complexity metrics should be accounted during testing (encoding and decoding processes) in the same machine for both anchors and for the evaluated learning-based image coding solution.**

The complexity of the training process is less relevant for the purpose of evaluating the learning-based image coding solution and may be reported optionally.

## 8 Anchors Generation

This Section describes the anchor generation process. As anchors, JPEG, JPEG 2000 and HEVC will be used. The list of anchors may be reduced.

- JPEG (ISO/IEC 10918-1 | ITU-T Rec. T.81)
- JPEG 2000 (ISO/IEC 15444-1 | ITU-T Rec. T.800)
- HEVC Intra (ISO/IEC 23008-2 | ITU-T Rec. H.265)

- VVC Intra (ISO/IEC 23090-3 | ITU-T Rec. H.266)

Information on available software and configurations to be used for these anchors is described next. The target bitrates for the *objective* evaluations are the same as Section 4. For format and color conversion the following program packages are used:

- FFmpeg version 3.4.8 (<https://git.ffmpeg.org/gitweb/ffmpeg.git/tag/refs/tags/n3.4.8>)
- ImageMagick 6.9.7-4 (<https://github.com/ImageMagick/ImageMagick6/tree/6.9.7-4>)

## 8.1 JPEG Anchor

JPEG does not specify a rate allocation mechanism allowing to target a specific bitrate. Hence, an external rate control loop is required to achieve the targeted bitrate. The following conditions apply:

- Available software: JPEG XT reference software, v1.62
  - Available at <http://jpeg.org/jpegxt/software.html>.
  - License: GPLv3
- Command-line specification (to use within the rate-control loop):
- Command-line examples:
  - ImageMagick will be used to convert file format from PNG to PNM:

```
convert [INPUTFILE_PNG].png -strip [INPUTFILE_PNM].pnm
```

- Encoder command line:

```
jpeg -q [QUALITY_PARAMETER] -h -qt 3 -s 1x1,2x2,2x2 [INPUTFILE_PNM].pnm  
[FILE_BITS].bits
```

where the h is to optimize Huffman tables -qt 3 to select visually improved quantization tables, -s 1x1,2x2,2x2 to use 420 subsampling and -oz to use trellis quantization.

- Decoder command line:

```
jpeg [FILE_BITS].bits [OUTPUTFILE_PNM].pnm
```

- ImageMagick will be used to convert file format from PNM to PNG:

```
convert [OUTPUTFILE_PNM].pnm [OUTPUTFILE_PNG].png
```

- Quality assessment [12] should be conducted between [INPUTFILE\_PNG].png and [OUTPUTFILE\_PNG].png, using size of [FILE\_BITS].bits for bitrate calculation.

## 8.2 JPEG 2000 Anchor

The JPEG 2000 anchor generation should support two configurations: 1) PSNR optimized which is used for objective assessment; and 2) Visually optimized which is used for subjective assessment. A target rate can be specified using the `-rate [bpp]` parameter. The following conditions apply:

- Available software: Kakadu, v8.0.5
  - Available at <http://www.kakadusoftware.com>.
  - License: demo binaries freely available for non-commercial use
- Command-line specification:
  - ImageMagick will be used to convert file format from PNG to PPM:

```
convert [INPUTFILE_PNG].png -strip [INPUTFILE_PPM].ppm
```

- Encoder command line for the visual quality optimized configuration:

```
kdu_compress -i [INPUTFILE_PPM].ppm -o [FILE_BITS].bits -rate <TARGET_BPP>  
Qstep=0.001 -tolerance 0 -full -precise -no_weights -num_threads 1
```

- Encoder command line for the MSE weighted configuration:

```
kdu_compress -i [INPUTFILE_PPM].ppm -o [FILE_BITS].bits -rate <TARGET_BPP>  
Qstep=0.001 -tolerance 0 -full -precise -no_weights -num_threads 1
```

- Decoder command line:

```
kdu_expand -i [FILE_BITS].bits -o [OUTPUTFILE_PPM].ppm -precise
```

- ImageMagick will be used to convert file format from PPM to PNG:

```
convert [OUTPUTFILE_PPM].ppm [OUTPUTFILE_PNG].png
```

- Quality assessment [12] should be conducted between [INPUTFILE\_PNG].png and [OUTPUTFILE\_PNG].png, using size of [FILE\_BITS].bits for bitrate calculation.

### 8.3 HEVC Intra Anchor

For HEVC Intra, an external rate control loop is required to achieve targeted bitrate. The HEVC RD performance for the target bitrates are obtained with the following conditions:

- Available software: HEVC Test Model (HM 16.20)
  - Available at [https://hevc.hhi.fraunhofer.de/svn/svn\\_HEVCSoftware/tags/HM-16.20+SCM-8.8/](https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.20+SCM-8.8/)
  - License: BSD
- FFMPEG will be used to convert the PNG (RGB) to YUV following the BT.709 primaries according to:

```
ffmpeg -hide_banner -i [INPUTFILE_PNG].png -pix_fmt yuv444p10le -vf  
scale=in_range=full:in_color_matrix=bt709:out_range=full:out_color_matrix=bt709 -  
color_primaries bt709 -color_trc bt709 -colorspace bt709 -y [INPUTFILE_YUV].yuv
```

Losses introduced by this conversion can be checked using the procedure described in



ANNEX 2 and appears to be negligible.

- HEVC Configuration files to be used are available here:
  - [https://gitlab.com/wg1/jpeg-ai/jpeg-ai-anchors/-/blob/main/Compression/HEVC/encoder\\_intra\\_main\\_scc\\_10.cfg](https://gitlab.com/wg1/jpeg-ai/jpeg-ai-anchors/-/blob/main/Compression/HEVC/encoder_intra_main_scc_10.cfg)
- Encoder command line:

```
TAppEncoderStatic -c encoder_intra_main_scc_10.cfg -i [INPUTFILE_YUV].yuv -wdt <WIDTH>
-hgt <HEIGHT> -b [FILE_BITS].bits -f 1 -fr 25 -q <QP> --FrameSkip=0 --InputBitDepth=10
--InputChromaFormat=444 --ChromaFormatIDC=444 --Level=6.2
```

where <WIDTH> and <HEIGHT> are width and height of the input YUV file, <QP> is a quality parameter from the list.

- Decoder command line:

```
TAppDecoderStatic -d 10 -b [FILE_BITS].bits -r [OUTPUTFILE_YUV].yuv
```

- FFMPEG will be used to convert the decompressed YUV to reconstructed PNG (RGB) following the BT.709 primaries according to:

```
ffmpeg -f rawvideo -vcodec rawvideo -s <WIDTH>x<HEIGHT> -r 25 -pix_fmt yuv444p10le -i
[OUTPUTFILE_YUV].yuv -pix_fmt rgb24 -vf
scale=in_range=full:in_color_matrix=bt709:out_range=full:out_color_matrix=bt709 -
color_primaries bt709 -color_trc bt709 -colorspace bt709 -y [OUTPUTFILE_PNG].png
```

- Quality assessment [12] conducted between [INPUTFILE\_PNG].png and [OUTPUTFILE\_PNG].png, using size of [FILE\_BITS].bits for bitrate calculation.

## 8.4 VVC Intra Anchor

For VVC Intra, an external rate control loop is also required to achieve targeted bitrate. The VVC RD performance for the target bitrates are obtained with the following conditions:

- Available software: VVC Test Model (VTM 11.1)
  - Available at [https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware\\_VTM](https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM)
  - License: BSD

- FFMPEG will be used to convert the PNG (RGB) to YUV following the BT.709 primaries according to:

```
ffmpeg -hide_banner -i [INPUTFILE_PNG].png -pix_fmt yuv444p10le -vf  
scale=in_range=full:in_color_matrix=bt709:out_range=full:out_color_matrix=bt709 -  
color_primaries bt709 -color_trc bt709 -colorspace bt709 -y [INPUTFILE_YUV].yuv
```

Losses introduced by this conversion can be checked using the procedure described in

ANNEX 2 and appear to be negligible.

- Encoder command line:

```
EncoderAppStatic -c cfg/encoder_intra_vtm.cfg -c cfg/classSCC.cfg -i  
[INPUTFILE_YUV].yuv -wdt <WIDTH> -hgt <HEIGHT> -b [FILE_BITS].bits -f 1 -fr 10 -q <QP>  
--InputBitDepth=10 --InputChromaFormat=444 --ChromaFormatIDC=444 --  
TemporalSubsampleRatio=1 --Level=6.2
```

where <WIDTH> and <HEIGHT> are width and height of the input YUV file, <QP> is a quality parameter from the list.

- Decoder command line:

```
DecoderAppStatic -d 10 -b [FILE_BITS].bits -r [OUTPUTFILE_YUV].yuv
```

- FFMPEG will be used to convert the decompressed YUV to reconstructed PNG (RGB) following the BT.709 primaries according to:

```
ffmpeg -f rawvideo -vcodec rawvideo -s <WIDTH>x<HEIGHT> -r 25 -pix_fmt yuv444p10le -i  
[OUTPUTFILE_YUV].yuv -pix_fmt rgb24 -vf  
scale=in_range=full:in_color_matrix=bt709:out_range=full:out_color_matrix=bt709 -  
color_primaries bt709 -color_trc bt709 -colorspace bt709 -y [OUTPUTFILE_PNG].png
```

- Quality assessment [12] should be conducted between [INPUTFILE\_PNG].png and [OUTPUTFILE\_PNG].png, using size of [FILE\_BITS].bits for bitrate calculation.

## 8.5 Software for Anchor Generation

The anchor generation for standard reconstruction task is available at: <https://gitlab.com/wg1/jpeg-ai/jpeg-ai-anchors>. Instructions for users can be found in README.md. This repository includes the software for anchor generation, namely the codecs and scripts used to generate all decoded data for the test images.

## 9 Naming Convention for Decoded Images and Bitstreams

The PNG decoded files should adhere to the following naming convention:

VM\_<IMGID>\_TE\_<RES>\_<ORIGINAL BIT DEPTH>bit\_sRGB\_<BR>.png

The bitstream files should adhere to the following naming convention:

VM\_<IMGID>\_TE\_<BR>.bits

with:

- IMGID is an identification of the image with 5 digits (from 00001 to 00050)
- TE is a fixed value which represents it is a test image
- RES is the spatial resolution (<WIDTH> and <HEIGHT> are width and height )
- Bit depth (which can be 8 or 10 bit)
- Color space (which must be sRGB)
- BR target bitrate for decoded images: YXX (e.g. 1.25 bpp would be ‘125’ and 0.05 would be 005)

## 10 10 bit images

PNG format with 16 bit-depth used for storing 10 bit data. The data put to high 10 bit starting from the most significant bit. All other bits set to 1. The table below shows an example of storing 10-bit data in 16-bit register.

Bit #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	a <sub>0</sub>	a <sub>1</sub>	a <sub>2</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>	a <sub>6</sub>	a <sub>7</sub>	a <sub>8</sub>	a <sub>9</sub>	1	1	1	1	1	1

where a<sub>N</sub> is N<sup>th</sup> bit of the data.

## 11 Evaluation Framework and Results Reporting Template

Evaluation framework for standard image reconstruction task is publicly available at [12]. Instructions for users can be found in README.md. Results reporting template, which includes all information mandatory to be reported (according to Sections 5 and 7) with **example** of anchor data is available in the same git repository. The results reporting template must be used.

---

## **PART B – COMMON TRAINING AND TEST CONDITIONS FOR COMPUTER VISION TASKS**

## **12 Compressed Domain Image Classification**

### **12.1 Objective**

The objective of this Section is the evaluation of compressed domain image classifiers. These image classifiers receive as input a quantized latent representation (and not a decoded image) and should achieve competitive image classification accuracy with respect to full decoding followed by image classification (especially at low rates) as well as lower complexity.

The quantized latent code of an image from a pretrained end-to-end (E2E) image codec is the input to the compressed domain image classification network. Naturally, suitable training is also needed to derive the weights of the compressed domain image classification network. The compressed-domain image processing task has been investigated in JPEG AI Exploration Studies which have showed great potential (WG1N92049 ES3.1). A description of a possible compressed domain network is available at WG1N100105.

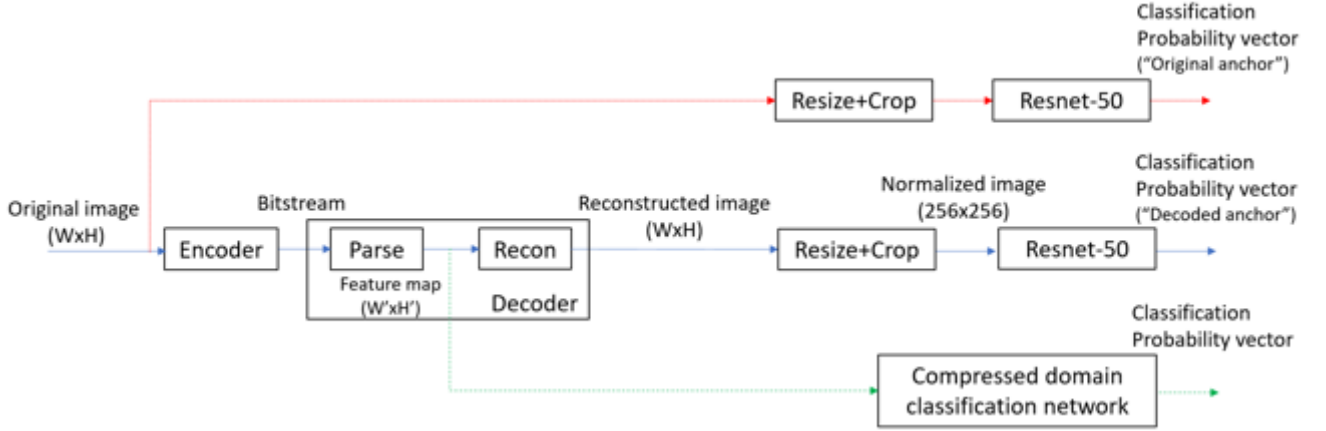
### **12.2 Training and Test Dataset**

The ImageNet dataset from the Large-Scale Visual Recognition Challenge 2012 (ILSVRC2012) [13] is used for training, validation and test. This dataset includes images of various sizes, from around 100x100 to 6Kx5K, and the most frequent sizes are around 500x300 or 300x500. The class label is the folder name in ImageNet database. There are in total 1000 classes, all 1000 classes will be used for assessment proposals in this category. This dataset can be briefly characterized as:

- Training dataset is the ImageNet training set, which includes around 1300K images.
- Testing dataset is the ImageNet validation set, which includes 50K images.

### **12.3 Anchor Generation**

The anchors are based on a state-of-the-art Resnet-50 image-domain classification network [14]. The pre-trained Resnet-50 model was obtained from Torchvision (model file name: resnet50-19c8e357.pth) [15]. The anchor generation process is illustrated in Figure 2.



**Figure 2. Illustration of the procedure to generate the original anchor, the decoded anchor and the compressed domain image classification test results.**

The anchors for the image classification are defined next:

- **Original Anchor:** Image classification is applied to the original images, before any compression, to assess the performance without any compression artifacts (pre-trained Resnet-50 [15] is used as the classification network). Before being fed to the Resnet-50, the images of various sizes in the testing dataset are normalized to 256x256 by two steps. First, an image is resized to make the shorter side 256 while keeping the aspect ratio of the image, e.g., by using the `resize()` function in Pytorch (with default bilinear interpolation). This step generates 256xN or Nx256 images ( $N \geq 256$ ). Then, center cropping is used to generate a square image of 256x256. The normalized 256x256 images are classified by the Resnet-50 model, and the 1000-class probability vector of the input image is the output of the Resnet-50 model. The top-1 and top-5 accuracy are measured based on the probability vectors and the ground-truth class labels of the test images. Scripts for performing images normalization and classification with ResNet-50 are described in the Section 11.8.
- **Decoded Anchor:** Image classification is applied to fully decoded images, i.e., from the decoded pixel-wise representation (using pre-trained Resnet-50 as in the original anchor as the classification network). Procedure is identical to original anchor generation (as shown in Fig. 2) except that the decoded images are the input of the Resnet-50 image domain classification network. For the decoder anchor case, any codec could potentially be used, but for the purpose of Call for Proposals evaluation each proponent should use the decoder that was submitted for the standard reconstruction track.

## 12.4 Bitrates

Four target bitrates 0.12, 0.25, 0.50, 0.75 bpp (bit per pixel) should be reported. The rate is measured as the total number of bits of the bitstreams of the testing dataset divided by the total number of pixels of original images the testing dataset.

## 12.5 Performance Metrics

There are two performance metrics, to measure the classification accuracy: Top-1 accuracy which is mandatory and Top-5 accuracy which may be optionally reported. They are described next:

- Top-1 accuracy: the class with the highest probability in the probability vector is the same as the class label of the image.
- Top-5 accuracy: one of the five classes with the highest probability in the probability vector is the same as the class label of the image.

To measure the complexity, the following metrics are used:

- Total size of the compressed domain classification network model(s) (for all mandatory rate points), measured by the product of the number of network parameters and the precision of the parameters (in bytes).
- kMAC/px for performing the image-domain classification and compressed-domain classification over the entire testing set, including those for image decoding, image-domain classification, latent decoding and latent-domain classification, respectively. Note that the average is over all pixels of the original images in the testing dataset.
- Processing run time for the entire testing set, including both decoding (if needed) and classification operations

## 12.6 Evaluation Framework and Testing Procedure

The anchor generation software and supporting material for compressed domain image classification is available at: <https://gitlab.com/wgl/jpeg-ai/jpeg-ai-anchors/-/tree/main/Classification>. Instruction for the usage of the code can be found in the README.md file. Pre-trained models used by the testing script are also linked in the package. To test the ResNet-50 model using the original images (i.e., for the original anchor):

```
python -m Classification.process --Classification.data_dir /path/to/dataset --output  
/path/to/output_dir
```

where data\_url /path/to\_images points 1) to the location of uncompressed images for the original anchor, or 2) to the location of decompressed images for the decoded anchor (different location of images compressed at different quality level).

## 12.7 JPEG AI Naming Conventions of Image Classification

The following is mandatory for bitstreams and should be honored.



- for bit-streams (in `bit` folder)

`<TEAMID>_<IMGID>_TE_<BR>.bits`

Same streams must be decodable by decoder in standard reconstruction task submitted by same team, in order to produce reconstructed image for further decoded anchor computation.

Here `BR` takes values 012, 025, 050, 075. Top-1 and Top-5 accuracy for each bit-rate should be computed by the sub-task decoder. Examples for Top-1 and Top-5 accuracy computation can be found [here](#).

In this task `IMGID` is image name in of [ILSVRC 2012](#).

For example, names of input images are

[DATASET DIRECTORY](#)

```
n01440764
  ILSVRC2012_val_00000293.JPEG
  ILSVRC2012_val_00002138.JPEG
  ...
```

```
n01443537
  ILSVRC2012_val_00000236.JPEG
  ILSVRC2012_val_00000262.JPEG
  ...
```

...

Corresponding names for bitstreams are

`_Classification/bit`

```
n01440764
  <TEAMID>_00000293_TE_<BR>.bits
  <TEAMID>_00002138_TE_<BR>.bits
  ...
```

```
n01443537
  <TEAMID>_00000236_TE_<BR>.bits
  <TEAMID>_00000262_TE_<BR>.bits
```

---

## **PART C – COMMON TRAINING AND TEST CONDITIONS FOR IMAGE PROCESSING TASKS**

## 13 Compressed Domain Super-Resolution

### 13.1 Objective

Compressed-domain Super Resolution (SR) is an image processing task that consists in performing learning-based super resolution directly on the latent-space representation of a JPEG AI learning-based image codec [16]. Compressed-domain SR aims at reducing the computational cost of potentially required up-scaling of a compressed image. Moreover, super resolution may contribute to reducing bandwidth costs, as well as lowering the required storage capacity for local and cloud-based systems. The compressed-domain super resolution task has been investigated in JPEG AI Exploration Studies. An example of a compressed-domain super resolution task can be found in WG1N92049 ES3.3. A description of a possible compressed domain super-resolution network is available at WG1N100105.

### 13.2 Training and Test Dataset

The JPEG AI dataset should be used for training. The hidden test images from the JPEG AI dataset will be down-sampled and used as test input images for this task. The down-sampling of original high-resolution JPEG AI test images will be performed with a factor of 4 using one the following methods:

- Bilinear interpolation
- Bicubic interpolation
- Spline interpolation
- Lanczos3 interpolation

It will not be disclosed which down-sampling method will be used for each image.

### 13.3 Anchors

The anchors are based on two methods: 1) a classical up-sampling method and 2) a DNN-based super resolution method, both with an up-sampling of factor of 4, namely:

- 1) Classical up-sampling based on Lanczos interpolation filter with a window size of 3 and 8;
- 2) WDSR [17] network with the pretrained WDSRx4 [18] model.

The anchors defined next only differ in the point of the codec pipeline where the super-resolution methods described above are applied (as shown in Figure 3):

- **Original Anchor** (same for all): The super resolution methods (as described above) are applied to the images from the test dataset that have been down-sampled from original high-resolution images. All up-sampling is applied before any compression, thus avoiding any compression artifacts.

- **Decoded Anchor** (varies between proponents): Super resolution is applied in pixel domain to fully decoded images. These images have been obtained by down-sampling the original high-resolution images using the methods specified above, which are then encoded and decoded using the learning-based image codec that was submitted for the standard reconstruction track.

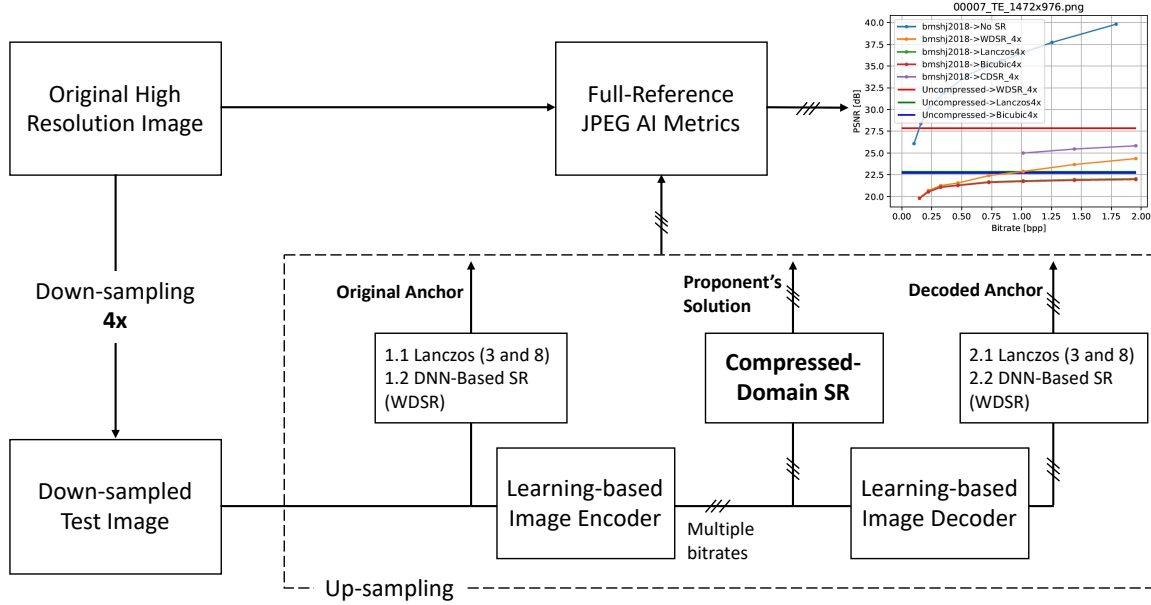


Figure 3. Illustration of the procedure to generate the original anchor, the decoded anchor and the compressed domain super resolution test results (proponent contribution).

### 13.4 Bitrates

The following bitrates should be covered: 0.03, **0.06**, **0.12**, **0.25**, **0.50**, **0.75**, 1.00, 1.50, 2.00 bpp. The bold typeface indicates mandatory bitrates.

### 13.5 Performance Metrics

Objective quality evaluation should be performed using the JPEG AI quality metrics listed in the Section 5 of this document, specifically, using the implementations that are available in the JPEG AI quality assessment framework. Computation complexity evaluation should also be performed according to Section 7 and the results must be reported per pixel, counting all pixels of low-resolution images.

### 13.6 Evaluation Framework and Testing Procedure

The anchor generation software and supporting material for compressed domain super-resolution is available at: <https://gitlab.com/wg1/jpeg-ai/jpeg-ai-anchors/-/tree/main/SuperResolution>. Instruction for the usage of the code can be found in the README.md file.

## 13.7 Naming Conventions for Super-Resolution

The following is mandatory for bitstreams and reconstructed images and should be honored.

- for bit-streams (in `bit` folder)

`<TEAMID>_<IMGID>_TE_<BR>.bits`

- for bit-reconstructed images (in `rec` folder)

`<TEAMID>_<IMGID>_TE_<RES>_<ORIGINAL BIT DEPTH>bit_sRGB_<BR>.png`

Here is `RES` resolution of up-sampled to full size ground truth image (not resolution of image encoded); `BR` takes values 006, 012, 025, 050, 075.

Same streams must be decodable by decoder in standard reconstruction task submitted by same team, to produce reconstructed images for further decoded anchor computation.

## 14 Compressed Domain Denoising

### 14.1 Objective

Compressed-domain Image Denoising is an image processing task that aims at removing the noise directly from the latent representation of learning-based coding solution while decoding. For this purpose, a compressed-domain decoder that integrates denoising operations at the decoder side of learning-based image compression methods should be proposed. The pipeline should be able to compress and denoise images simultaneously, being the information of the noise distribution and standard deviation known. Integrating the denoising operations in the decoder has the advantage of reducing the computational complexity and, potentially, improving the performance of the pipeline when compared to the decoding and denoising in cascade. A description of a possible compressed domain denoising network is available at WG1N100105.

### 14.2 Training and Test Dataset

In image denoising research, a common practice is to assume a noise model (usually Gaussian) and develop methods for removing such noise from noisy images. The proof-of-concept is achieved by starting with clean images, adding noise to them, and then assessing how well the proposed denoising method can remove added noise, all the while knowing the reference clean image. However, the noise in practical applications is more complex than a simple independent and identically distributed (iid) Gaussian noise. Therefore, a practical noise generator has been designed, by estimating the parameters

of a Poissonian-Gaussian noise model [21] from the noisy images in the Smartphone Image Denoising Dataset (SIDD) [22]. The noise generator is shared online and is available at [23]. The provided noise generator can be used to add noise to the clean images of the JPEG AI training dataset to obtain training noisy images.

During the training phase, the noisy images can be obtained by adding noise obtained from a noise generator to the original images in the JPEG AI training dataset. The provided noise generator returns a noisy image for the given input image. For the testing phase, images from the hidden JPEG AI test dataset will be contaminated with the noise obtained from the provided noise generator. The noisy test dataset will be shared with the proponents.

### 14.3 Anchors

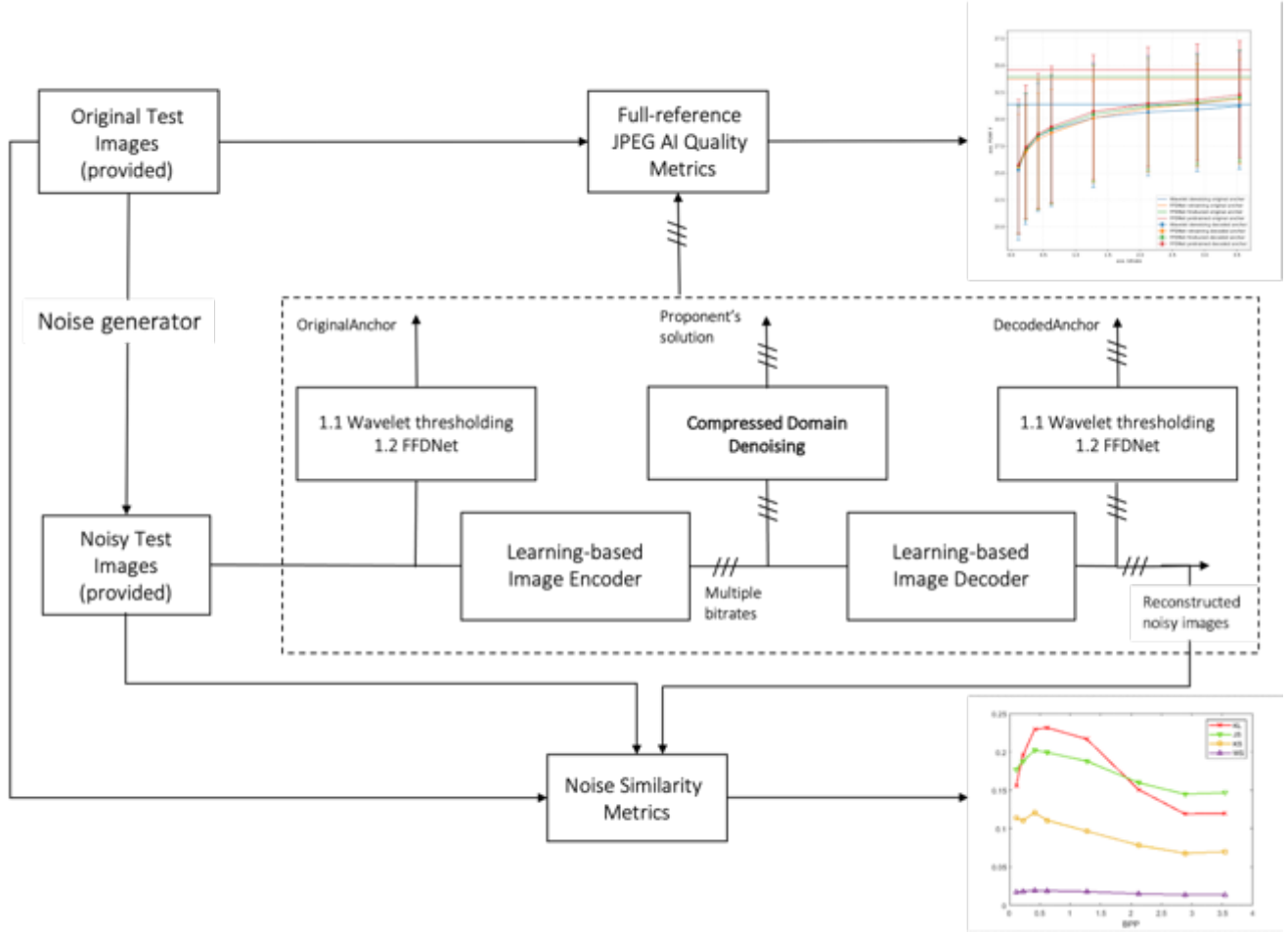
Two anchors based on two different denoising methods were examined: 1) a learning-based denoising method and 2) a conventional denoising method have been selected and used to denoise the noisy dataset images. More precisely, the selected denoising methods are:

- FFDNet [24], using the provided pretrained model and available at [23].
- Classical Wavelet thresholding denoising [25]. The denoising is implemented in Python using the scikit-learn library. The script to denoise using this method is available at [23].

Two anchors should be generated and compared with the proponents' solution:

- **Original Anchor** (same for all proponents): the image denoising methods (i.e. FFDNet and Classical Wavelet thresholding as described above) are applied to the images of the given noisy test dataset. The denoising is applied before any compression, thus avoiding any compression artifact.
- **Decoded Anchor** (varies between proponents): denoising is applied in the pixel domain to fully decoded images. These images should be created by encoding and decoding the given noisy test dataset using the learning-based image codec that was submitted for the standard reconstruction track.

The anchor generation pipeline is illustrated in Figure 4.



**Figure 4** - Illustration of the procedure to generate the original anchor, the decoded anchor and the compressed-domain denoising test results (proponent contribution).

#### 14.4 Bitrates

The following bitrates should be covered: 0.03, 0.06, **0.12**, **0.25**, **0.50**, **0.75**, 1.00, 1.50, 2.00 bpp. The bold typeface indicates mandatory bitrates. However, it is recommended that proponents provide results for all possible bitrates.

#### 14.5 Performance Metrics

The performance of the proposed compression and denoising pipeline should be evaluated using two different assessment methodologies:

- The visual quality of the denoised images should be evaluated based on the full-reference objective quality metrics listed in Section 5 of this document, namely through the implementations provided in the JPEG AI quality assessment framework. Computational complexity evaluation should also be performed according to Section 7.

- For the reconstructed noisy images, however, the image quality metrics in Section 5 do not correlate with the quality of the reconstructed noise. Hence, specific metrics for evaluating the goodness of fit between the input noise (i.e., noise in the input image) and the reconstructed noise (i.e., noise in the reconstructed image) are needed. The similarity between the reconstructed noise and the input noise can be evaluated using the following widely-known metrics that measure the similarity between probability distributions. The proposed metrics are:

- 1) Kullback-Leibler (KL) divergence,
- 2) Jensen–Shannon (JS) divergence,
- 3) Kolmogorov-Smirnov (KS) statistic,
- 4) Wasserstein distance.

The aforementioned metrics are useful in evaluating certain use cases where noise reconstruction is desirable, e.g. noise is added as an artistic feature. The scripts provided in [23] should be used to calculate the noise similarity metrics on the reconstructed test images.

## 14.6 Evaluation Framework and Testing Procedure

The anchor generation software and supporting material for compressed domain denoising is available at: <https://gitlab.com/wg1/jpeg-ai/jpeg-ai-anchors/-/tree/main/Denoising>. Instruction for the usage of the code can be found in the README.md file.

## 14.7 Naming Conventions for Denoising

The following is mandatory for bitstreams and reconstructed images and should be honored.

- for bit-streams (in `bit` folder)

`<TEAMID>_<IMGID>_<NOISE LEVEL>_TE_<BR>.bits`

- for bit-reconstructed images (in `rec` folder)

`<TEAMID>_<IMGID>_<NOISE LEVEL>_TE_<RES>_<ORIGINAL BIT`

`DEPTH>bit_sRGB_<BR>.png`

Here `NOISE LEVEL` indicates noise level of encoded image; `BR` takes values 012, 025, 050, 075.

Same streams must be decodable by decoder in standard reconstruction task submitted by same team, to produce reconstructed images for further decoded anchor computation.



## References







- [1] Z. Wang, E. P. Simoncelli, and A.C. Bovik, "Multiscale structural similarity for image quality assessment," Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, Pacific Grove, CA, USA, November 2003.
- [2] Z. Wang, Q. Li, "Information Content Weighting for Perceptual Image Quality Assessment", IEEE Transactions on Image Processing, Vol. 20, No. 5, May 2011.
- [3] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy and M. Manohara, "Toward A Practical Perceptual Video Quality Metric", Available [here](#).
- [4] H.R. Sheikh and A. C. Bovik, "Image Information and Visual Quality," IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal, Canada, August 2004.
- [5] N. Ponomarenko, F. Silvestri, K. Egiazarian, M. Carli, J. Astola, V. Lukin, "On between-coefficient contrast masking of DCT basis functions", Proceedings of the third international workshop on video processing and quality metrics (Vol. 4), 2007.
- [6] V. Laparra, J. Ballé, A. Berardino, and E. P. Simoncelli, "Perceptual Image Quality Assessment using a Normalized Laplacian Pyramid", S&T Symposium on Electronic Imaging: Conf. on Human Vision and Electronic Imaging, San Francisco, CA, USA, February 2016.
- [7] L. Zhang, L. Zhang, X. Mou, D. Zhang, "FSIM: a Feature Similarity Index for Image Quality Assessment," IEEE Transactions on Image Processing, vol. 20, no. 8, pp. 2378-2386, August 2011.
- [8] ITU-R Recommendation BT.500-13, "Methodology for the subjective assessment of the quality of television pictures," International Telecommunications Union, Geneva, Switzerland, 2012.
- [9] ITU-T Recommendation P. 910, "Subjective video quality assessment methods for multimedia applications," International Telecommunication Union, Geneva, 2008.
- [10] D. Saupe, F. Hahn, V. Hosu, I. Zingman, M. Rana, and S. Li, "Crowdworkers proven useful: a comparative study of subjective video quality assessment," International Conference on Quality of Multimedia Experience (QoMEX), Lisbon, Portugal, June 2016.
- [11] C. Keimel, J. Habigt, C. Horch, and K. Diepold, "Qualitycrowd: a framework for crowd-based quality evaluation," Picture Coding Symposium, Krakow, Poland, May 2012.
- [12] "JPEG AI quality assessment framework", <https://gitlab.com/wg1/jpeg-ai/jpeg-ai-qaf>
- [13] <http://image-net.org/challenges/LSVRC/2012/>
- [14] Resnet-50 Pytorch implementation, <https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py>
- [15] Pretrained Resnet-50 model from Torchvision, <https://download.pytorch.org/models/resnet50-19c8e357.pth>
- [16] E. Upenik, M. Testolina and T. Ebrahimi, "Towards super resolution in the compressed domain of learning-based image codecs," in SPIE Applications of Digital Image Processing XLIV, 2021.
- [17] J. Yu, Y. Fan, J. Yang, N. Xu, Z. Wang, X. Wang and T. Huang, "Wide Activation for Efficient and Accurate Image Super-Resolution," in arXiv:1808.08718, 2018.
- [18] [Online]. Available: <https://github.com/ychfan/wdsr>.
- [19] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: residual learning of deep CNN for image denoising," IEEE transactions on Image Processing, vol. 26, no. 7, pp. 3142–3155, 2017.
- [20] S.G. Chang, B. Yu, and M. Vetterli. "Adaptive wavelet thresholding for image denoising and compression." IEEE Transactions on Image Processing, vol. 9, no. 9, pp. 1532-1546, September 2000.
- [21] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical poissonian-gaussian noise modeling and fitting for single-image raw-data," IEEE Trans. Image Processing, vol. 17, no. 10, pp. 1737–1754, Oct. 2008
- [22] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in Proc. IEEE CVPR'18, pp. 1692-1700, June 2018.
- [23] <https://gitlab.com/wg1/jpeg-ai/jpeg-ai-anchors>.
- [24] K. Zhang, W. Zuo, and L. Zhang. "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," IEEE Transactions on Image Processing, vol. 27, no. 9, pp. 4608-4622, Sept. 2018.
- [25] S. G. Chang, B. Yu, and M. Vetterli. "Adaptive wavelet thresholding for image denoising and compression." IEEE Transactions on Image Processing, vol. 9, no. 9, pp. 1532-1546, Sept. 2000.



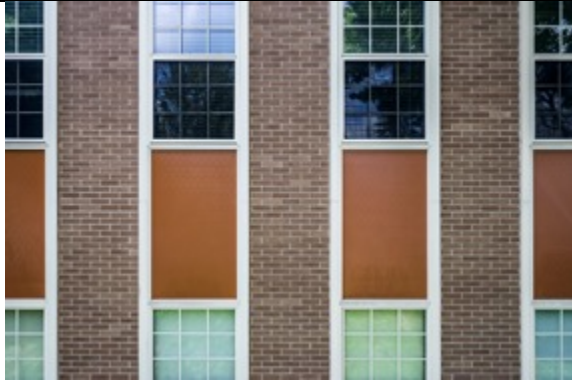
# ANNEX 1 Test set for standard reconstruction task

The test images are shown in Table 1.

**Table 1 50 images in standard reconstruction task test set**

 <p>00001 TE 2096x1400 8bit sRGB.png</p>	 <p>00002 TE 2144x1424 8bit sRGB.png</p>
 <p>00003 TE 1944x1296 8bit sRGB.png</p>	 <p>00004 TE 1808x1352 8bit sRGB.png</p>
 <p>00005 TE 1336x872 8bit sRGB.png</p>	 <p>00006 TE 1544x1120 8bit sRGB.png</p>





00007 TE\_1472x976\_8bit\_sRGB.png



00008 TE\_1912x1272\_8bit\_sRGB.png



00009 TE\_1976x1312\_8bit\_sRGB.png



00010 TE\_1744x1160\_8bit\_sRGB.png



00011 TE\_1512x2016\_8bit\_sRGB.png



00012 TE\_1920x1280\_8bit\_sRGB.png

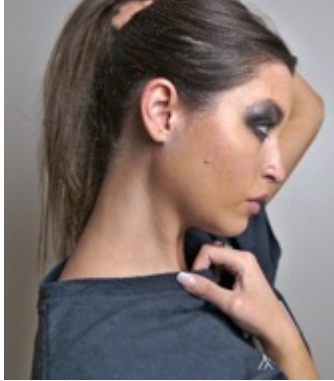


00013 TE\_3680x2456\_8bit\_sRGB.png



00014 TE\_3680x2456\_8bit\_sRGB.png





00015 TE 1744x2000 8bit sRGB.png



00016 TE 1192x832 8bit sRGB.png



00017 TE 1280x848 8bit sRGB.png



00018 TE 3032x1856 8bit sRGB.png



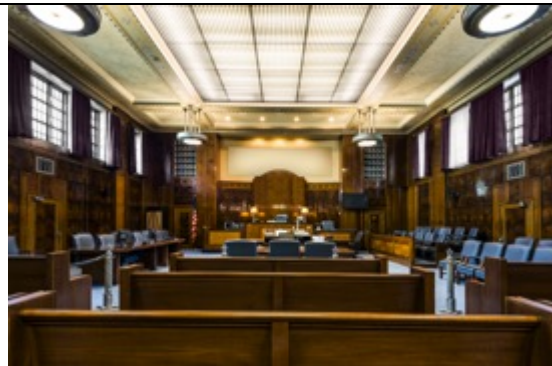
00019 TE 1920x1080 8bit sRGB.png



00020 TE 3680x2456 8bit sRGB.png



00021 TE 2192x1520 8bit sRGB.png



00022 TE 1248x832 8bit sRGB.png





00023 TE 2464x1640 8bit sRGB.png



00024 TE 1536x1024 8bit sRGB.png



00025 TE 1984x1320 8bit sRGB.png



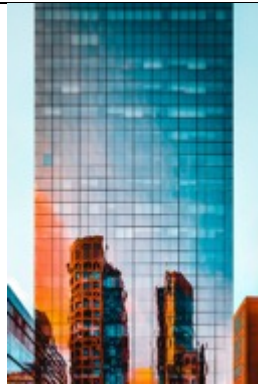
00026 TE 1784x1296 8bit sRGB.png



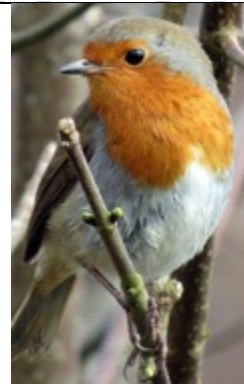
00027 TE 3680x2456 8bit sRGB.png



00028 TE 800x1200 8bit sRGB.png



00029 TE 976x1472 8bit sRGB.png



00030 TE 560x888 8bit sRGB.png



00031 TE 1752x1856 8bit sRGB.png



00032\_TE\_7680x5120\_8bit\_sRGB.png



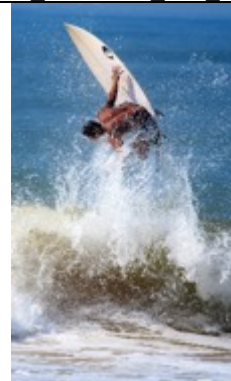
00033 TE 2120x1608 8bit sRGB.png



00034 TE 1072x928 8bit sRGB.png



00035 TE 877x1658 8bit sRGB.png



00036 TE 998x1675 8bit sRGB.png

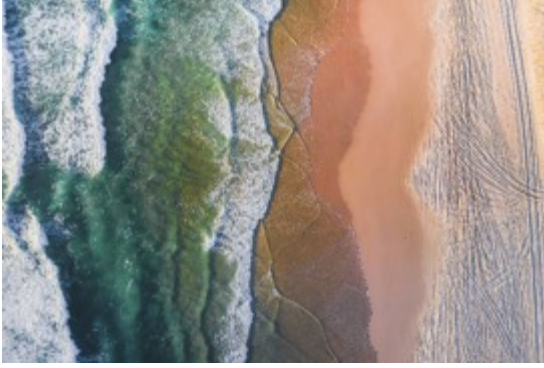


00037 TE 5616x3744 8bit sRGB.png



00038 TE 8160x6120 8bit sRGB.png





00039 TE 5464x3640 8bit sRGB.png



00040 TE 7394x4932 8bit sRGB.png



00041 TE 3374x5055 8bit sRGB.png



00042 TE 2787x4004 8bit sRGB.png



00043 TE 945x840 8bit sRGB.png



00044 TE 1430x1834 8bit sRGB.png

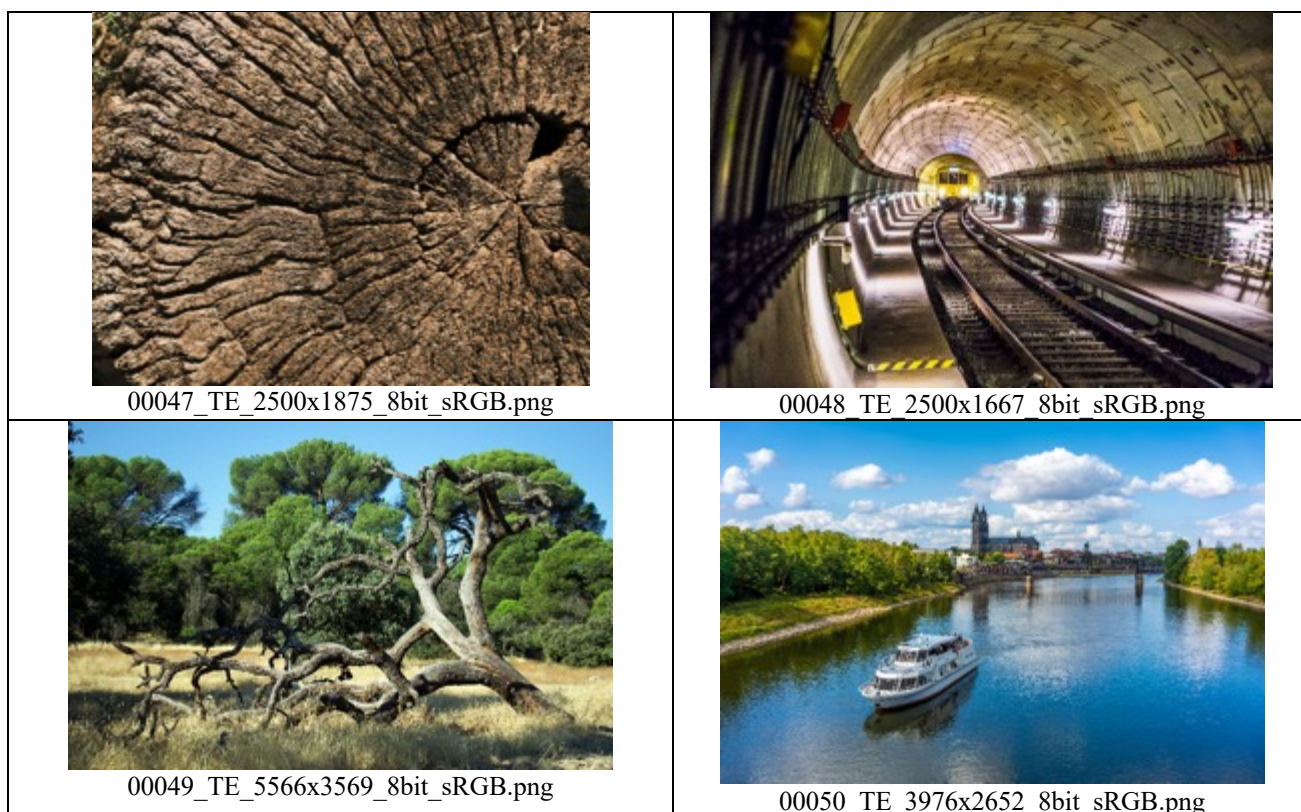


00045 TE 2533x1897 8bit sRGB.png



00046 TE 2816x1878 8bit sRGB.png





The list of MD5 check sums is presented next:

0d89b343312dbb9ed05f3a94b40364a8 00001\_TE\_2096x1400\_8bit\_sRGB.png  
acc74bd138c122f74136cf795635f37c 00002\_TE\_2144x1424\_8bit\_sRGB.png  
48257df4164f878cda6aa04e728916e8 00003\_TE\_1944x1296\_8bit\_sRGB.png  
7b3109b6f7bb7a22477f9e5b327f3a59 00004\_TE\_1808x1352\_8bit\_sRGB.png  
c4774fa32ca1dd77a28aa13c75b5a5d3 00005\_TE\_1336x872\_8bit\_sRGB.png  
8934866c4b1c891c91a3ba96fac4a733 00006\_TE\_1544x1120\_8bit\_sRGB.png  
cc103f0b5d7d92c608c159a5deace439 00007\_TE\_1472x976\_8bit\_sRGB.png  
8924c4659dc7393c096db8126f6642da 00008\_TE\_1912x1272\_8bit\_sRGB.png  
a49db044c5645b47fb4197f7adbecfa4 00009\_TE\_1976x1312\_8bit\_sRGB.png  
08f51e6bcb2f92b4448549d669c9f7dc 00010\_TE\_1744x1160\_8bit\_sRGB.png  
e5dab1ce7fbb66eef876884f55de5d2f 00011\_TE\_1512x2016\_8bit\_sRGB.png  
5c51b7bd3b394001fd8ccb9d9cac1f57 00012\_TE\_1920x1280\_8bit\_sRGB.png  
5fa5db1a07e2cb19ef847f9d823b598f 00013\_TE\_3680x2456\_8bit\_sRGB.png  
1dda3e5d5ccdb40c5f4cf6eb02774e53 00014\_TE\_3680x2456\_8bit\_sRGB.png  
140ca18a8576dfdf0c1583110becc0e9 00015\_TE\_1744x2000\_8bit\_sRGB.png  
e934809e740399d82526dd90871fd333 00016\_TE\_1192x832\_8bit\_sRGB.png  
460f8ccc3250d1855c3afa4bc7ddcf38 00017\_TE\_1280x848\_8bit\_sRGB.png  
fbb4ab162a9ff915b2f5f40a395ea3d5 00018\_TE\_3032x1856\_8bit\_sRGB.png  
657e0082654a76c50cadef06ff2aadf4 00019\_TE\_1920x1080\_8bit\_sRGB.png  
fcc03e82b6ed235044daa70bd5370641 00020\_TE\_3680x2456\_8bit\_sRGB.png  
b2988da7a85da0ff1bb5a15ff68131dc 00021\_TE\_2192x1520\_8bit\_sRGB.png  
c15564c9fc2ff51decae549de8754053 00022\_TE\_1248x832\_8bit\_sRGB.png

b4fc0ff9e5e9cf34bb1775b1e6c0ff59 00023\_TE\_2464x1640\_8bit\_sRGB.png  
acb69a2c27e9830e45d996249b83a6e6 00024\_TE\_1536x1024\_8bit\_sRGB.png  
4fbe1b030ed1d7e80a371e36845e18d7 00025\_TE\_1984x1320\_8bit\_sRGB.png  
024756224193fbf4abf8158749a2fa5c 00026\_TE\_1784x1296\_8bit\_sRGB.png  
ad533d527533022d473a256dd2ea6c10 00027\_TE\_3680x2456\_8bit\_sRGB.png  
5ac1dd91f41966c6a676d8a97c03e22d 00028\_TE\_800x1200\_8bit\_sRGB.png  
65f3b7111950973fe61ffcf0d3ff9089 00029\_TE\_976x1472\_8bit\_sRGB.png  
fea8098efec8d29575d3e221a1730d47 00030\_TE\_560x888\_8bit\_sRGB.png  
31d6f23b9af8bff57988c6f6f219f243 00031\_TE\_1752x1856\_8bit\_sRGB.png  
0663ad808d4345183b071bec3c234fbc 00032\_TE\_7680x5120\_8bit\_sRGB.png  
87be33141221251c23a3691288a79e44 00033\_TE\_2120x1608\_8bit\_sRGB.png  
7e5fec8489ee768339830f4840086f15 00034\_TE\_1072x928\_8bit\_sRGB.png  
683084c0010ff170d47a4738f6965ab0 00035\_TE\_877x1658\_8bit\_sRGB.png  
219ae70c8bffcf08fb7370d9d0250f61a 00036\_TE\_998x1675\_8bit\_sRGB.png  
d3e60fbf1d78634ab9a10e3d6610ea03 00037\_TE\_5616x3744\_8bit\_sRGB.png  
23c1b443d5048cf9fd2af830ebbc6e23 00038\_TE\_8160x6120\_8bit\_sRGB.png  
a06b4427adeb732fe55a3ba02ae342b7 00039\_TE\_5464x3640\_8bit\_sRGB.png  
1633f3fe0b8e425c95a7eabc2d18d7e2 00040\_TE\_7394x4932\_8bit\_sRGB.png  
69efd468370fabd6dea61ea836a4475d 00041\_TE\_3374x5055\_8bit\_sRGB.png  
36aadd68f2edb9f66767017fb5c919a4 00042\_TE\_2787x4004\_8bit\_sRGB.png  
891d0350a59366125d7e877ab49247b4 00043\_TE\_945x840\_8bit\_sRGB.png  
ffc219708813a278368a11da1c31d5c3 00044\_TE\_1430x1834\_8bit\_sRGB.png  
2862b96b64ca49ef4f12e3f817fe714e 00045\_TE\_2533x1897\_8bit\_sRGB.png  
484ecce7ee80aa2e92003761c29f572f 00046\_TE\_2816x1878\_8bit\_sRGB.png  
78c967afe4d1e841f7099d5c418c7c28 00047\_TE\_2500x1875\_8bit\_sRGB.png  
457db5e5d93f19bfccdbaa54bf99796a 00048\_TE\_2500x1667\_8bit\_sRGB.png  
fd8d461fc08bdf8f8f5955e24895319b 00049\_TE\_5566x3569\_8bit\_sRGB.png  
58d7c95f16c5e2ab62d89de4380a34b9 00050\_TE\_3976x2652\_8bit\_sRGB.png

## ANNEX 2 - Color space conversion validity check

An example of a color space conversion validity check script is presented below.

```
#!/usr/bin/env bash

ffmpeg -hide_banner \
-i IMAGE_WxH.png \
-pix_fmt yuv444p10le \
-vf scale=in_range=full:in_color_matrix=bt709:out_range=full:out_color_matrix=bt709 \
-color_primaries bt709 -color_trc bt709 -colorspace bt709 \
-y IMAGE_WxH.png.yuv

ffmpeg -hide_banner \
-f rawvideo -vcodec rawvideo -s [W]x[H] -r 25 -pix_fmt yuv444p10le \
-i IMAGE_WxH.png.yuv \
-pix_fmt rgb24 \
-vf scale=in_range=full:in_color_matrix=bt709:out_range=full:out_color_matrix=bt709 \
-color_primaries bt709 -color_trc bt709 -colorspace bt709 \
-y IMAGE_WxH.png.yuv.png

compare 00005_TE_1336x872.png 00005_TE_1336x872.png.yuv.png 00005_TE_diference.png
```