|  |  |
| --- | --- |
|  | Moving Picture, Audio and Data Coding  by Artificial Intelligence  www.mpai.community |

|  |  |
| --- | --- |
| **N1682** | 2024/03/20 |
| **Source** | Video group |
| **Title** | MPAI-EVC Evidence Project report and plan |
| **Target** | MPAI Members |

The goal of the group is to enhance EVC (Essential Video Coding) using AI-tools to reach at least 25% improvement over the baseline profile. The group is currently working on three coding tools: Intra prediction, Super Resolution, and in-loop filtering. For each tool, in the following we describe the proposed approach and the steps of database building, learning phase and inference.

**MPAI dataset**

Th new dataset based on BVI-DVC Part 1 (University of Bristol)combined with

* Ultravideo dataset, containing 16 4K 10-bit raw sequences, available here <http://ultravideo.fi/>
* The opensource SVT datasets (7 new plus 5 old 4K sequences), available here <https://www.svt.se/opensource/content>
* The Tencent video dataset (85 4K sequences), available here <https://multimedia.tencent.com/resources/tvd>

for a total of 350 4K sequences.

To be used in the MPAI-EVC experiments, the sequences must be pre-processed. The actions taken by the group to prepare the sequences for the training are shown in Figure 1.



Figure 1 processing workflow

We finished the coding at fixed QP (as per the Common Test Conditions): 22, 27, 32, 37, 42, 47.

The coded video sequences are in YUV 4:2:0 10-bit format, and come either in the BT.709 or BT.2020 colour spaces. Since the super-resolution network operates on PNG images, the application of the correct colour space information in the conversion process from YUV to PNG is critical to avoid introducing unwanted image deterioration and, as a consequence, losing coding efficiency due to factors other than the coding scheme itself.

Thus, two different workflows are followed, accounting for the colour space difference. The ffmpeg command lines used for the conversion of BT.709 and BT.2020 content respectively, are

ffmpeg -pix\_fmt yuv420p10le -s videosize -i inpufile.yuv -vf scale=out\_color\_matrix=bt709:flags=full\_chroma\_int+accurate\_rnd+lanczos -c:v png outputdir/%3d.png

and

ffmpeg -pix\_fmt yuv420p10le -s videosize -i inpufile.yuv -vf scale=out\_color\_matrix=bt2020:flags=full\_chroma\_int+accurate\_rnd+lanczos -c:v png outputdir/%3d.png

Other datasets, such as the Youtube UGC dataset are being investigated.

The dataset is ready, and it is available on MPAI ftp server.

Also the test sequences are available on the ftp server.



Figure 2: MPAI-EVC datasets

We now have two datasets (Figure 2): a huge training dataset with 350 4K sequences that we use for training. Moreover, we have a test dataset with few sequences, only 24 sequences, which we use for testing. The overlap between these two datasets is zero, which means we do not use test sequences in the training phase.

In addition, all the results we present are based on the test sequences. We usually call them JVET sequences, because they are in common with the MPEG JEVET group.

**Intra prediction tool**

We address the challenge of predicting an intra-coded block given its context (Intra prediction) as an image inpainting problem, i.e. recovering pixels of an image that are unavailable due to, e.g. occlusions or information loss. Masked convolutional neural networks have been recently proposed for image inpainting exploiting the apriori information from the context to recover the missing image area. The proposed method relies on masked convolutions to generate the block predictor starting from a decoded context of 64 × 64 pixels (Figure 3). For example, for each 32x32 coding unit a 64x64 context is sent to the autoencoder. The autoencoder returns to the EVC encoder a 32x32 predictor that is considered as a 6th EVC Intra predictor mode that is put into competition with the other 5 predictors. The generated bitstream is fully decodable under the assumption that the autoencoder network is also available at the decoder side.



Figure 3: context on the left and the predictor on the right

The masked autoencoder (Figure 4) is trained in a supervised manner for 1000 Epochs over a set of randomly drawn patches from about 800 images representing various types of contents by minimising the absolute error (ABS) between the network output and the original patch.



Figure 4: Procedure for training the convolutional autoencoder used to generate the Intra predictor.

We worked on the architecture of the encoder, switching to a VGG-inspired topology where the encoder is made of 5 pairs of stacked convolutional layers with 3x3 filters interleaved by subsampling for a total of 10 convolutional layers (results in Table 1).



Table 1: results with VGG-like architecture over the standard JVET test sequences for the 22-37 and 22-42 QP ranges and for 5 modes

Table 1 shows the improvement compared to the previous architecture: in green the delta wrt reference, 5 modes. The BD rate increased by an average of 1 percent from the previous architecture.



Table 2: results with the new architecture VGG-like and for 6 modes

Table 2 shows that at the moment there is no gain in adding a sixth mode (values in red), rather there is a generalized loss (small gains only for class A, more pronounced losses for class E). We are trying to explain these numbers by calculating some statistics on the predictor modes (Table 3).



Table 3: percentage of modes use: left: 5 modes, reference; middle: 5 modes, new architecture; right: 6 modes new architecture on JVET Class A

Table 3 shows the percentage of use of a particular mode on the Class A JVET sequences. With the new architecture and 5 modes, the percentage of neural predictor use increases from 51% to 62%. Unfortunately, by adding the sixth mode, the percentage drops to 56%, because with 5 modes the DC mode is replaced, while with 6 modes the DC mode reaches 25% utilisation.



Figure 5: distribution of modes use

Figure 5 shows that Class A has about 81% of the predictions in modes 0 and 1, whereas this number drops to 76% for the Class E.

Adding a 6th new intra yield no gains currently. From our analysis, it seems that NN predictor is more selected than DC but still we have no gains. This is probably due to the cost of signalling the 6th predictor. We are investigating the bit-cost for signalling and residuals.

**Super-resolution tool**

The super-resolution step is added as a post-processing tool. The picture before encoding with EVC baseline profile is downscaled and then the super-resolution network is applied to the decoded picture to get the native resolution.

We have carried out extensive preliminary experiments, to help us in understanding the different options that will make sense to carry in the final experiments using the standard training and test datasets, which will be used in all the experiments carried out in this activity. We have trained the selected deep-learning approach for super-resolution on 4 QPs (15,30,37 and 45). We have tested its performances on 8 test sequences for the case of SD to HD, and on 3 test sequences for the case of HD to 4K.

The group has worked on the computation of the BD-rate SD to HD, Table 2, showing an improvement of -3.14% when compared with the ground truth EVC.

|  |  |  |
| --- | --- | --- |
| **Sequence** | **Class** | **BD-Rate** |
| Crowd Run | Class B 1920x1080 60/50 fps, 8 bpp | -1.24% |
| Ducks Take Off | Class B 1920x1080 60/50 fps, 8 bpp | 2.12% |
| Park Joy | Class B 1920x1080 60/50 fps, 8 bpp | 1.40% |
| Diego and Owl | Class B 1920x1080 60/50 fps, 8 bpp | 8.11% |
| Rome 1 | Class B 1920x1080 60/50 fps, 8 bpp | 0.19% |
| Rome 2 | Class B 1920x1080 60/50 fps, 8 bpp | -18.81% |
| Rush Hour | Class B 1920x1080 60/50 fps, 8 bpp | 4.90% |
| Talk Show | Class B 1920x1080 60/50 fps, 8 bpp | -21.75% |
| **Average: -3.14%** | | |

Table 2: BD-rate performances on all the 8 test sequences

The HD to 4K testing phase has been finalised on all QPs (15,30,37 and 45).

Figure 6 shows the BD-rate curves for each sequence, where the network trained on QP 15 dataset has been used over the test set for all QPs. The reasoning behind it, is to learn the features information, which is intrinsic within the compressed sequences, at the highest quality possible (QP 15). This will avoid any shortcoming in the generalisation capabilities that we have faced doing separated training for each QP.



**Figure 6** BD-Rate curve for all sequences and QPs, showing the BD-rate variation (Bjontegaard) averaged over all the QPs.

The above results in the SD2HD and HD24K case were calculated by training the network on a Kaggle dataset (8-bit sources) and testing the sequences listed in the tables (8-bit sequences).

**New training Dataset set**

We decided to build a new training dataset consisting of high-quality 4K sequences (350 sequences) to improve on previous results by exploiting 10-bit sources.

We are following two paths:

* Feeding the neural network directly with the Y component from the patch extraction phase (without converting into the specific image format, avoiding any additional introduction of errors) and feeding the neural network with the Y component replicated 3 times (to follow the data input standard of the neural network)

This will be done only for the lower QP because in the preliminary experiment, as described above, we are able to guarantee the best performances while avoiding any generalisation issue.

The training datasets, patches extraction, for all the types of training listed in ‘Training Strategies’, have been initiated. However, for the SD2HD case, some issues with few sequences have been found and are under investigation. We have also started to analyse the test sequences to see if there are any potential problems so that we will be ready after training to run the test.

**New training Strategies**

We have identified different types of training strategies as follows:

1. Using the original network and its weights without any additional training
2. Starting from the configuration of point 1., we suggest performing refinement training (fine tuning) allowing us to compensate for the limited number of sequences we may have in the standard training set. The training will be based on patch extraction from the MPAI training dataset, using two approaches: random selection and importance sampling approach.

Based on the preliminary experiments this will be performed only on the lower QP (22). This will help in reducing the overall required training time. This approach has been so far found as the best solution providing good results for all QPs at the inference stage and limiting generalisation issues as shown in the previous sections.

1. Training the DRLN from scratch using patches selection approach that will give the best performances as shown in 2.
2. Training the DRLN from scratch, avoiding the patches selection approach. This means training on the full resolution input frame, helping to see if the patches approach introduces some degradation in the SR.
3. Training only on the SD2HD transformation and use it also for the HD24K. This will allow us to reduce the overall required training time.
4. Training only on uncompressed data. This will be done, to see if the obtained quality can be further improved.

|  |  |
| --- | --- |
| DRLN pretrained | Authors' original DRLN trained on RGB |
| DRLN refined | Fine tuning of the authors' original network |
|  |  |
| TFZ | Trained from zero, from scratch |
| EVC original | MPEG5-EVC |
| EVC + NN intra | MPEG5-EVC modified with NN for intra enhancement |

**Nomenclature/Acronyms used through all the document**

**Patches extraction for the SR block**

We have used two approaches to extract the patches, from the encoded frame sequences, for the creation of the training set to be used in both training approaches, DRL refined and TFZ.

Here the idea is to create a large training and validation datasets where the knowledge of the upsampling transformation to be learned is well represented despite the reduced number of patches when compared to the all possible patches in a frame, i.e. less information redundancies etc.

**Random sampling patches extraction**

For each input frame a patch is extracted by taking the patch using random origin coordinates keeping in consideration the borders. The patch is consequently cropped, to match the input and output resolutions requested by the super-resolution transform.

Due to the low complexity of the random sampling technique, it has been decided to perform it online at training time for help to increase the generalisation capabilities of the used deep-learning approach.

This consists in randomly selecting a possible patch for each input frame of the input sequences. We would like to stress that for each epoch we generate a different training set, with the goal to increase the generalisation capabilities of the used deep-learning approach.

Concerning the validation data set, it has been generated at the 1st training epoch, following the 20 % split rule, and it has been kept the same for the whole training process. This allows a comparable evaluation at the different epochs during the whole training process.

**Non-uniform sampling patches extraction based on entropy content**

Typically, the patches data set generated through uniform random sampling may contain large redundancy information, which will not contribute to the improvement of learning the specific task, while reducing the generalisation capabilities, and at the same time large data set will increase the time required for the training phase.

To mitigate the above issues, a patch extraction approach based on the patch entropy content has been adopted. In the context of image processing entropy is the measure of information content within the image area, i.e. "low entropy areas" are areas with low information content, "high entropy areas" are areas with high information content.

Based on the above concept, we have used a standard approach for extracting the patches with the goal of extracting the patches with the highest information content. This approach can be summarised as follow:

1. All possible patches for each frame of the sequences are extracted.
2. For each patch its entropy is calculated.
3. We want to know the frequency a specific entropy value occurs. This gives us a good understanding on how the entropy values are distributed. To derive this information, the histogram based on the patches entropy is calculated.
4. The histogram is normalised so the frequency sum is equal to 1, normalised count.
5. From the histogram calculated in point 4), the classical cumulative distribution function (CDF) is computed. This describes the distribution of a random variable and its utility is that it can be used for any type of random variable.
6. Then the CDF is uniformly sampled on the Y axes for identifying the entropy values on axes X, which correspond to the sampled values on Y.
7. Finally, given the entropy value in X, we identify its corresponding bin on the entropy histogram and then we select a random patch among the patches in the selected bin.

**Intra & super resolution combined**

This section will describe the activities so far done using approaches 1 and 2, as training strategy for the SR tool. For now approach 2 is limited to the random patches selection.

The group tested the pipeline with the two combined tools Intra and Super resolution (Figure 11). So far, gains for the NNIntra the DLRN SR tools have been evaluated independently, however the underlying combined gains are expected to be strongly not linear. Therefore, we performed coding experiments where the two NN Intra and SR encoding tools were jointly activated. In the combined tests, a 4K picture is bicubic-downscaled to HD before being EVC-compressed with the NNIntra tool enabled. At the output of the decoder, the SR tool is upsampling the frame at the native 4K resolution. Experiments were performed only on the first 9 frames (IBBBBBBBB GOP scheme) of the Class A 4K sequences due to complexity constraints; the QPs under consideration are 22, 27, 32, 37, 42.

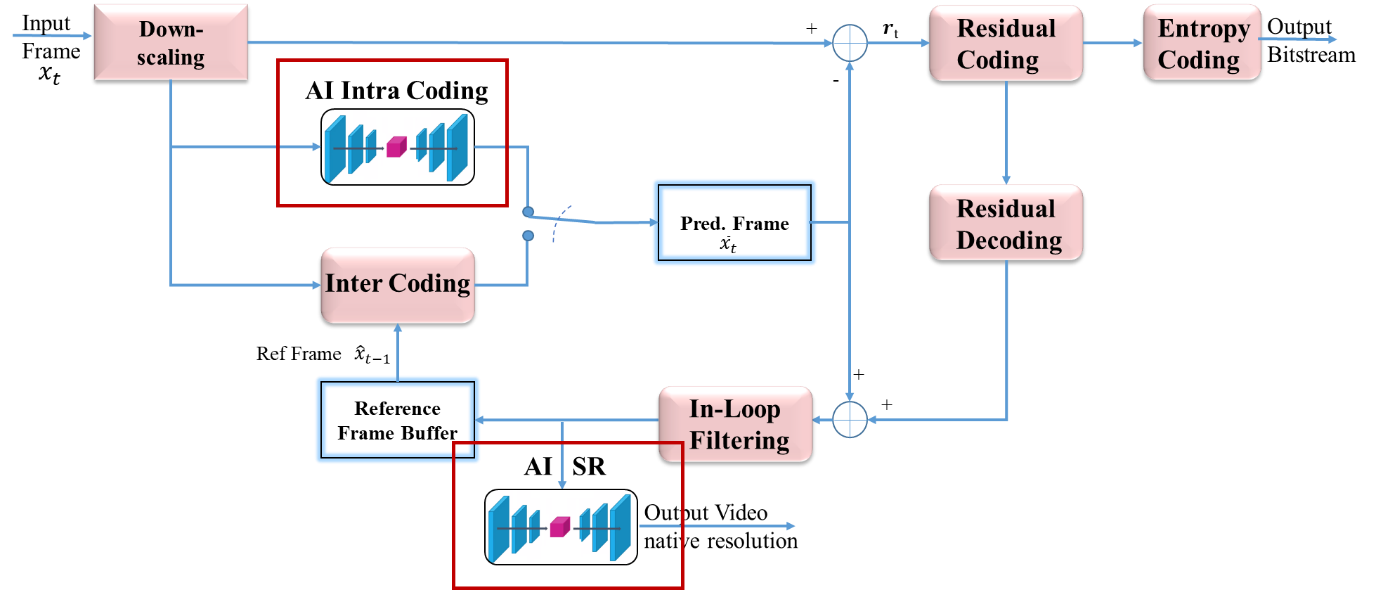


Figure 11: Setup for the combined NNIntra and Super Resolution (SR) tools

For the SR tool, we considered two approaches. The former, was the pretrained DLRN network, as it appears in the original work of its authors. The latter was the fine tuned version with the MPAI reference training dataset described in the MPAI datasets section. The results, of this activity, have been compared to the classical bicubic interpolation algorithm used as baseline (Figure 13).

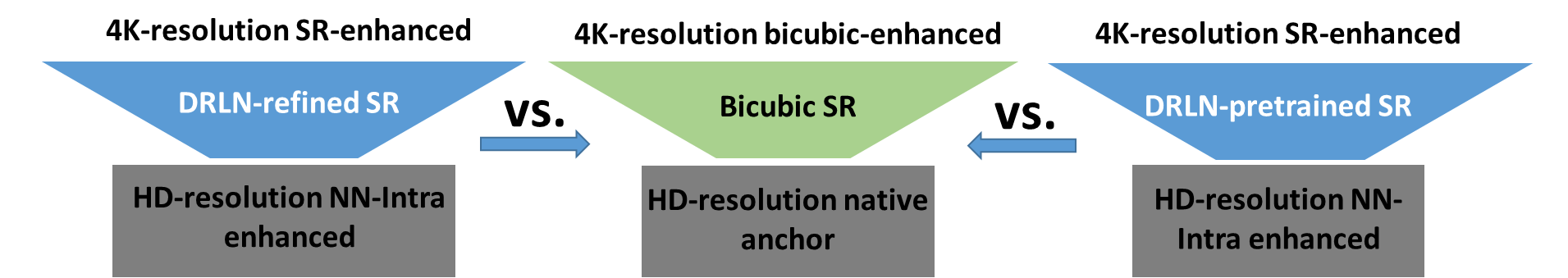
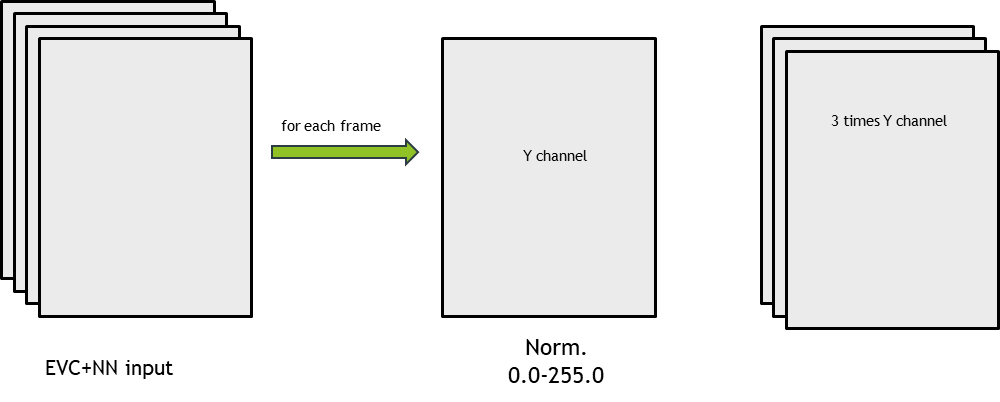
****

Figure 13: Joint NNIntra +SR comparison setups.

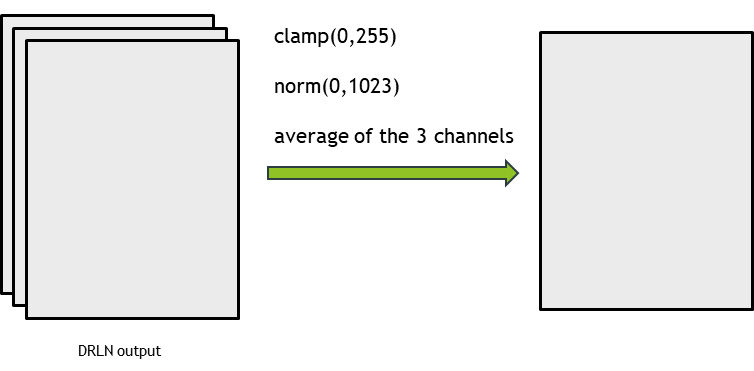
Since in this experiment, we are using the original pre-trained DRLN and its fine tuned version as described above, which in both cases the pre-training was done with a 8-bit content, we need to normalize its input between 0 and 255, for both cases the test and the fine tuning. For all sequences and for all QPs we took the output of the EVC + NN intra and for each frame we considered the Y channel.

Since the DLRN network was pre-trained over RGB data, so we decided to input into the network the Y channel repeated 3 times, for both the test and the fine tuning (training approach 2.) (Figure 14).



**Figure 14:** input to the DLRN network

The output of the Network is clipped between 0 and 255 and normalized between 0 and 1023 to restore the original 10 bit depth. Finally, we take the mean of the 3 channels of the DLRN as the upsampled Y channel upon which we measure video quality (Figure 15).



**Figure 15:** output of the DLRN network

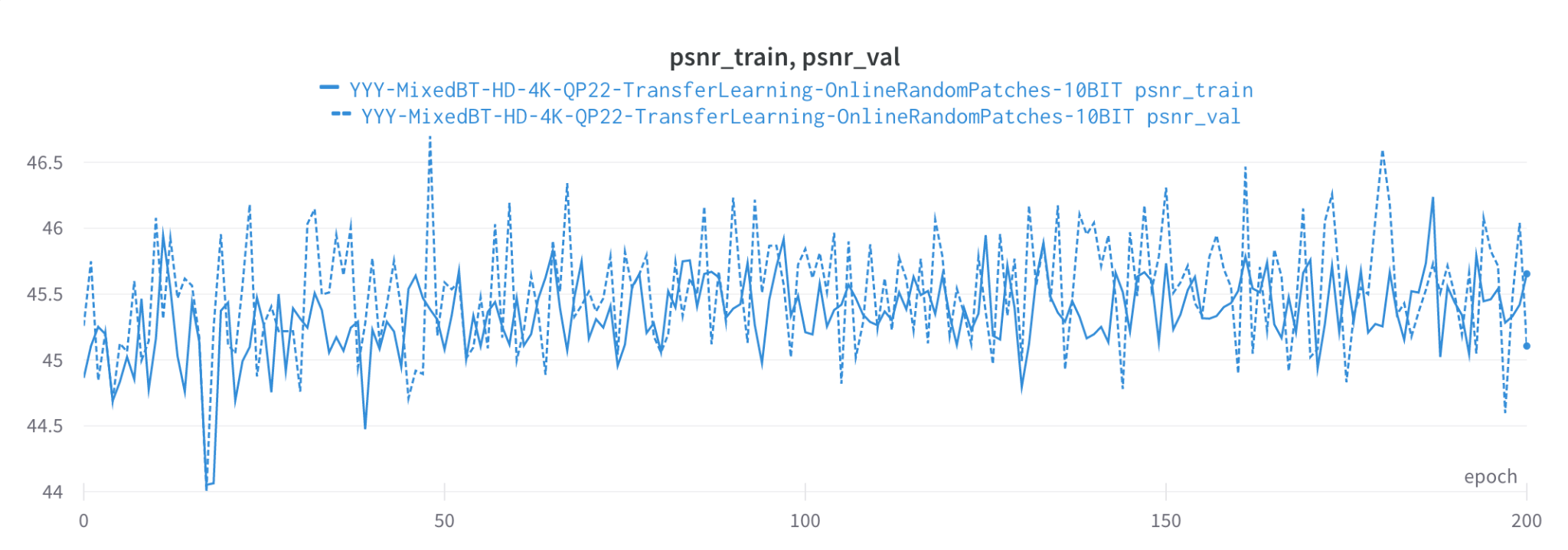
**DRLN Refined**

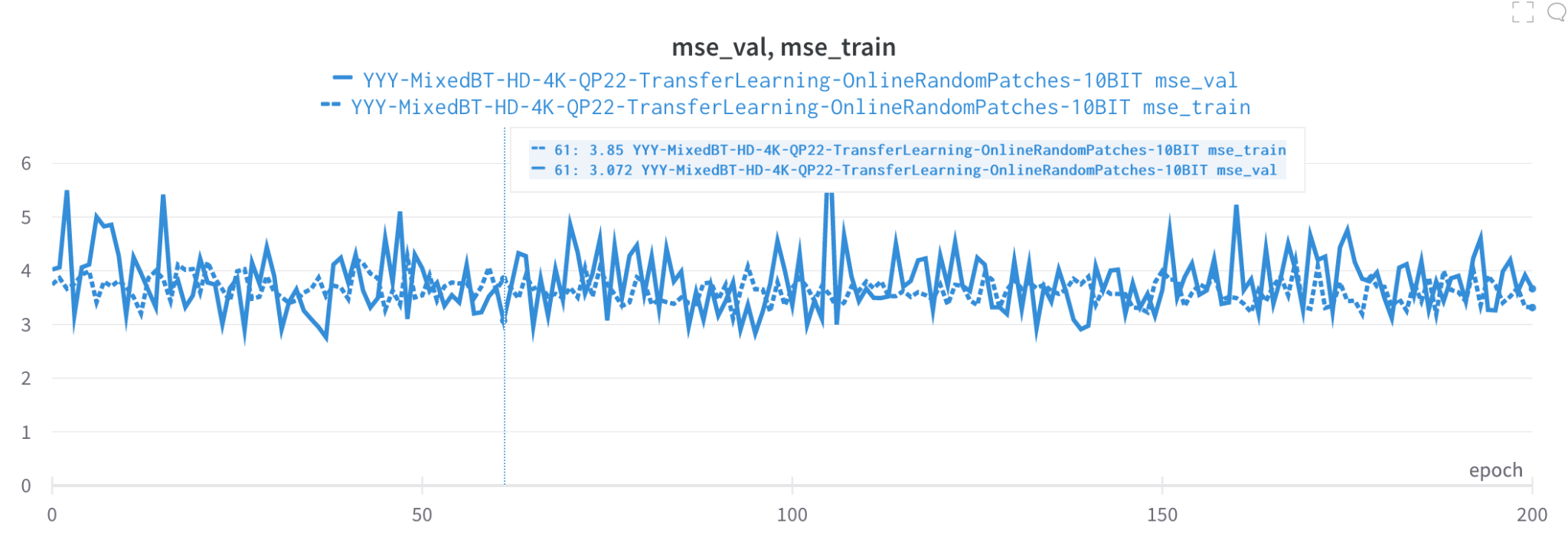
This section is showing the training performances of the refined DRLN, using as training dataset the MPAI dataset of 350 sequences pair HD-4K, specified in section MPAI datasets. Fine tuning is a well known tuning approach, which is used to infer knowledge referred to a different dataset domain where the approach will be used, while keeping the original features learned during a training phase for a different dataset domain.

To achieve it the layers related to the features extraction of the deep-learning approach are frozen and the training is performed with the new dataset only on the final layers of the network.

Concerning the patches extraction, we have adopted the random selection, performed online during training, where a random patch is selected from the available frames in all the 350 sequences, which are part of the training MPAI dataset.

Figure 16 is showing the training performances related to this activity.





**Figure 16:** PSNR (top) and MSE (bottom) functions of the training and validation datasets used for the fine-tuning of the original DRLN SR.

**Testing results**

**HD24K case**

The corresponding results are shown from Table 3 to Table 5 below and show combined gains, when compared to the EVC reference+bicubic interpolation (baseline), for the pre-trained DRLN (Table 3), the refined DRLN (Table 4) and the trained from zero DRLN (Table 5) approaches. The refined DRL and the trained from zero DRLN are trained using patches chosen randomly and using the importance sampling techniques. The results are showing a large gain with the refined DRLN using uniform sampling. This gain is 11.99% for the BD-rate, and of 0.41 dB for the BD-PSNR.

We have tries two approaches:

1. where the trained architecture DRLN for the user case HD24K is used.
2. where we have refined or re-trained from zero the DRLN architecture using the training set of SD2HD sequences.

|  |  |  |
| --- | --- | --- |
|  | **DRLN pre-trained** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -3.36 | 0.11 |
| CatRobot | -4.51 | 0.15 |
| DaylightRoad 2 | -8.92 | 0.21 |
| FoodMarket 4 | -5.46 | 0.22 |
| ParkRunning 3 | -0.75 | 0.03 |
| Tango 2 | -4.16 | 0.12 |
| **AVG** | **-4.53** | **0.14** |

**Table 3:** BD rates gain/loss when comparing DLRN network pre-trained vs. bicubic interpolation. The reference is the 4K EVC encoded sequence with bicubic upsampling.

**Case 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined (random sampling)** | | **DRLN refined (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -11.78 | 0.39 | -13.87 | 0.46 |
| CatRobot | -11.26 | 0.41 | -13.3 | 0.49 |
| DaylightRoad 2 | -13.95 | 0.36 | -15.64 | 0.41 |
| FoodMarket 4 | -8.11 | 0.34 | -9.22 | 0.39 |
| ParkRunning 3 | -7.79 | 0.31 | -9.97 | 0.39 |
| Tango 2 | -8.56 | 0.25 | -9.95 | 0.3 |
| **AVG** | **-10.24** | **0.34** | **-11.99** | **0.41** |

**Table 4:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (random sampling)** | | **DRLN trained TFZ (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -9.68 | 0.29 | -11.55 | 0.37 |
| CatRobot | -7.42 | 0.24 | -11.26 | 0.41 |
| DaylightRoad 2 | -9.13 | 0.21 | -13.89 | 0.35 |
| FoodMarket 4 | 1.08 | -0.07 | -8.55 | 0.36 |
| ParkRunning 3 | -6.40 | 0.23 | -7.92 | 0.31 |
| Tango 2 | -5.24 | 0.15 | -8.84 | 0.26 |
| **AVG** | **-6.13** | **0.17** | **-10.34** | **0.34** |

**Table 5:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**Case 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined (random sampling)** | | **DRLN refined (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -10.37 | 0.35 | -13.25 | 0.44 |
| CatRobot | -10.88 | 0.41 | -13 | 0.49 |
| DaylightRoad 2 | -13.75 | 0.36 | -15.38 | 0.41 |
| FoodMarket 4 | -8.5 | 0.36 | -7.33 | 0.31 |
| ParkRunning 3 | -7 | 0.27 | -8.47 | 0.32 |
| Tango 2 | -8.48 | 0.25 | -8.99 | 0.27 |
| **AVG** | **-9.83** | **0.33** | **-11.07** | **0.37** |

**Table 6:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (random sampling)** | | **DRLN trained TFZ (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -9.52 | 0.3 | -8.48 | 0.27 |
| CatRobot | -7.28 | 0.25 | -7.4 | 0.26 |
| DaylightRoad 2 | -8.98 | 0.21 | -10.98 | 0.27 |
| FoodMarket 4 | 0.82 | -0.06 | -6.49 | 0.27 |
| ParkRunning 3 | -5.24 | 0.2 | -2.49 | 0.09 |
| Tango 2 | -2.79 | 0.07 | -5.86 | 0.17 |
| **AVG** | **-5.50** | **0.16** | **-6.95** | **0.22** |

**Table 7:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**SD2HD case**

|  |  |  |
| --- | --- | --- |
|  | **DRLN pre-trained** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] |
| BasketBallDrive | -25.16 | 0.85 |
| BQTerrace | -43.30 | 1.47 |
| Cactus | -15.41 | 0.58 |
| MarketPlace | -3.09 | 0.09 |
| RitualDance | -3.00 | 0.13 |
| **AVG** | **-17.99** | **0.62** |

**Table 8:** BD rates gain/loss when comparing DLRN network pre-trained vs. bicubic interpolation. The reference is the HD EVC encoded sequence with bicubic upsampling.

**Case 1.**

The corresponding results are shown from Table 7 to Table 8 below and show combined gains, when compared to the EVC reference+bicubic interpolation (baseline), the refined DRLN (Table 7) and the trained from zero DRLN (Table 8) approaches. The refined DRL and the trained from zero DRLN are trained using patches chosen randomly and using the importance sampling techniques. The results are showing a large gain with the refined DRLN using both random and importance sampling. This gain is 25.86 % for the BD-rate, and of 0.93 dB for the BD-PSNR.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined (random sampling)** | | **DRLN refined (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BQTerrace | -46.72 | 1.61 | -47.56 | 1.57 |
| BasketBallDrive | -29.44 | 1.05 | -30.61 | 1.07 |
| Cactus | -23.35 | 0.89 | -25.41 | 0.94 |
| MarketPlace | -9.03 | 0.26 | -10.75 | 0.32 |
| RitualDance | -12.31 | 0.61 | -14.95 | 0.75 |
| **AVG** | **-24.17** | **0.88** | **25.86** | **0.93** |

**Table 9:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

As shown in Table 8, re-training from zero the original architecture does not provide any improvement when compared with the results obtained with the refined approach (Table 7).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (random sampling)** | | **DRLN trained TFZ (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BQTerrace | -41.81 | 1.07 | -36.32 | 0.90 |
| BasketBallDrive | -24.82 | 0.74 | -26.45 | 0.83 |
| Cactus | -19.00 | 0.58 | -17.78 | 0.56 |
| MarketPlace | -6.31 | 0.17 | -6.35 | 0.18 |
| RitualDance | -6.72 | 0.29 | -9.42 | 0.43 |
| **AVG** | **-19.73** | **0.57** | **-19.26** | **0.58** |

**Table 10:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**Case 2.**

The corresponding results are shown from Table 9 to Table 10 below and show combined gains, when compared to the EVC reference+bicubic interpolation (baseline), the refined DRLN (Table 9) and the trained from zero DRLN (Table 10) approaches. The refined DRLN and the trained from zero DRLN are trained using patches chosen randomly and using the importance sampling techniques. The results are showing a large gain when compared with the baseline but are not improving the results obtained using the trained architecture for the user case HD24K.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined (random sampling)** | | **DRLN refined (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BQTerrace | -45.71 | 1.63 | -48.63 | 1.8 |
| BasketBallDrive | -29.16 | 1.11 | -31.37 | 1.19 |
| Cactus | -22.17 | 0.89 | -24.47 | 0.95 |
| MarketPlace | -8.38 | 0.25 | -10.42 | 0.31 |
| RitualDance | -11.06 | 0.55 | -13.57 | 0.68 |
| **AVG** | **-23.30** | **0.89** | **-25.69** | **0.99** |

**Table 11:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (random sampling)** | | **DRLN trained TFZ (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BQTerrace | -41.93 | 1.17 | -43.39 | 1.24 |
| BasketBallDrive | -19.89 | 0.54 | -24.46 | 0.74 |
| Cactus | -17.64 | 0.57 | -19.09 | 0.64 |
| MarketPlace | -1.43 | 0.04 | -5.17 | 0.14 |
| RitualDance | -0.57 | 0.02 | -8.38 | 0.38 |
| **AVG** | **-16.29** | **0.47** | **-20.10** | **0.63** |

**Table 12:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**Only SR (EVC codec)**

Here we report the results on the same test set but only for SR, using the same type of settings. We did an experiment to understand the performances of SR when applied alone and to understand the behaviour of the introduced improvement of performances when using deep learning approaches in more than one block in the existing decoding system, i.e. additive, multiplicative etc.

We have tried two approaches for both SR functions, i.e., SD2HD and HS24K:

1. where the trained architecture DRLN for the user case HD24K is used.
2. where we have refined or re-trained from zero the DRLN architecture using the training set of SD2HD sequences.

**HD24K (EVC codec)**

The results of both cases are then compared with the pre-trained original DRLN architecture (Table 11).

|  |  |  |
| --- | --- | --- |
|  | **DRLN pre-trained** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | 0.21 | 0.01 |
| CatRobot | 0.97 | -0.02 |
| DaylightRoad 2 | -2.05 | 0.06 |
| FoodMarket 4 | 1.79 | -0.07 |
| ParkRunning 3 | 1.48 | -0.04 |
| Tango 2 | 3.57 | -0.1 |
| **AVG** | **1.00** | **-0.03** |

**Table 13:** BD rates gain/loss when comparing DLRN network pre-trained vs. bicubic interpolation. The reference is the 4K EVC encoded sequence with bicubic upsampling.

**Case 1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined (random sampling)** | | **DRLN refined (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -9.71 | 0.33 | -12.04 | 0.4 |
| CatRobot | -7.03 | 0.27 | -9.32 | 0.35 |
| DaylightRoad 2 | -7.89 | 0.21 | -9.74 | 0.26 |
| FoodMarket 4 | -1.27 | 0.05 | -2.48 | 0.1 |
| ParkRunning 3 | -5.81 | 0.23 | -7.99 | 0.32 |
| Tango 2 | -1.55 | 0.05 | -3.08 | 0.09 |
| **AVG** | **-5.54** | **0.19** | **-7.44** | **0.25** |

**Table 14:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ ( random sampling)** | | **DRLN trained TFZ (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -7.89 | 0.24 | -9.7 | 0.31 |
| CatRobot | -3.43 | 0.11 | -7.14 | 0.26 |
| DaylightRoad 2 | -3.27 | 0.08 | -7.89 | 0.2 |
| FoodMarket 4 | 8.48 | -0.34 | -1.8 | 0.08 |
| ParkRunning 3 | -4.5 | 0.16 | -5.94 | 0.23 |
| Tango 2 | 1.76 | -0.05 | -1.9 | 0.06 |
| **AVG** | **-1.48** | **0.03** | **-5.73** | **0.19** |

**Table 15:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**Case 2.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined (random sampling)** | | **DRLN refined (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -7.92 | 0.27 | -10.99 | 0.37 |
| CatRobot | -6.43 | 0.25 | -8.62 | 0.33 |
| DaylightRoad 2 | -7.52 | 0.2 | -8.96 | 0.25 |
| FoodMarket 4 | -1.58 | 0.07 | -0.36 | 0.02 |
| ParkRunning 3 | -4.96 | 0.2 | -6.32 | 0.24 |
| Tango 2 | -1.34 | 0.04 | -1.92 | 0.06 |
| **AVG** | **-4.96** | **0.17** | **-6.20** | **0.21** |

**Table 16:** Patches refined training -using SD2HD trained network for the HD24K user case:BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (random sampling)** | | **DRLN trained TFZ (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -7.43 | 0.24 | -6.15 | 0.2 |
| CatRobot | -2.86 | 0.1 | -2.73 | 0.1 |
| DaylightRoad 2 | -2.66 | 0.06 | -4.57 | 0.12 |
| FoodMarket 4 | 8.31 | -0.34 | 0.53 | -0.02 |
| ParkRunning 3 | -3.21 | 0.13 | -0.42 | 0.02 |
| Tango 2 | 4.66 | -0.13 | 1.44 | -0.04 |
| **AVG** | **-0.53** | **0.01** | **-1.98** | **0.06** |

**Table 17:** Patches training from zero - using SD2HD trained network for the HD24K user case: BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**SD2HD (EVC codec)**

The results of both cases are then compared with the pre-trained original DRLN architecture (Table 16).

|  |  |  |
| --- | --- | --- |
|  | **DRLN pre-trained** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] |
| BasketBallDrive | -20.01 | 0.70 |
| BQTerrace | -40.55 | 1.38 |
| Cactus | -9.39 | 0.42 |
| MarketPlace | 0.8 | -0.02 |
| RitualDance | 7.4 | -0.29 |
| **AVG** | **-12.27** | **0.44** |

**Table 18:** BD rates gain/loss when comparing DLRN network pre-trained vs. bicubic interpolation. The reference is the HD EVC encoded sequence with bicubic upsampling.

**Case 1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined (random sampling)** | | **DRLN refined (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BQTerrace | -34.68 | 1.34 | -45.34 | 1.51 |
| BasketBallDrive | -19.31 | 0.73 | -26.57 | 0.95 |
| Cactus | -15.53 | 0.65 | -21.57 | 0.83 |
| MarketPlace | -5.51 | 0.17 | -7.29 | 0.22 |
| RitualDance | -4.32 | 0.24 | -7.34 | 0.39 |
| **AVG** | **-15.87** | **0.63** | **-21.62** | **0.78** |

**Table 19:** Patches refined training -using HD24K trained network for the SD2HD user case:BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (random sampling)** | | **DRLN trained TFZ (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BQTerrace | -29.70 | 0.85 | -42.69 | 1.24 |
| BasketBallDrive | -16.20 | 0.51 | -25.46 | 0.87 |
| Cactus | -12.28 | 0.41 | -18.46 | 0.65 |
| MarketPlace | -2.90 | 0.08 | -5.67 | 0.17 |
| RitualDance | -1.03 | -0.04 | -5.18 | 0.27 |
| **AVG** | **-12.01** | **0.36** | **-19.49** | **0.64** |

**Table 20:** Patches training from zero - using HD24K trained network for the SD2HD user case BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**Case 2.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined (random sampling)** | | **DRLN refined (uniform sampling)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BQTerrace | -33.68 | 1.37 | -46.07 | 1.71 |
| BasketBallDrive | -19.33 | 0.80 | -27.04 | 1.05 |
| Cactus | -13.87 | 0.63 | -19.68 | 0.81 |
| MarketPlace | -4.80 | 0.15 | -6.75 | 0.2 |
| RitualDance | -2.5 | 0.16 | -5.2 | 0.29 |
| **AVG** | -14.84 | **0.62** | **-20.95** | **0.81** |

**Table 21:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (random sampling)** | | **DRLN trained TFZ (uniform sampling** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BQTerrace | -29.51 | 0.91 | -40.72 | 1.17 |
| BasketBallDrive | -8.81 | 0.25 | -19.79 | 0.61 |
| Cactus | -9.44 | 0.34 | -14.71 | 0.52 |
| MarketPlace | 2.35 | -0.06 | -1.45 | 0.04 |
| RitualDance | 8.38 | -0.32 | 0.33 | 0.01 |
| **AVG** | **-7.41** | **0.22** | **-15.27** | **0.47** |

**Table 22:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

On average, the general behaviour is that the two improvements provided by the two deeplearning blocks intra and super resolution are additive.

**Extension on VVC**

This experiment is helping to understand if the SR neural network has potential to be "codec agnostic," meaning that it can improve video quality across a variety of video coding standards, including the emerging Versatile Video Coding (VVC).

**Tested on VVC sequences**

**Only(SR)**

**SD2HD (VVC codec)**

In this experiment we have tested the SR step on the VVC encoded sequences using the pretrained DRLN approach (vanilla test flavour) with its own weight, for the case SD2HD the Table 21 is showing the final results:

|  |  |  |
| --- | --- | --- |
|  | **DRLN pre-trained** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] |
| **BasketBallDrive** | -43.12 | 1.57 |
| **BQTerrace** | -49.43 | 1.51 |
| **Cactus** | -22.78 | 0.72 |
| MarketPlace | -3.2 | 0.08 |
| RitualDance | -5.09 | 0.22 |
| **AVG** | **-22.09** | **0.96** |

**Table 23:** BD rates gain/loss when comparing DLRN network pre-trained vs. bicubic interpolation. The reference is the HD VVC encoded sequence with bicubic upsampling.

During the experiments we noticed that three of the above test sequences we were using were 8-bit sequences (bold sequence name) while the references were 10-bit sequences. To compensate for it in the PSNR calculation, we have converted (put into a 10-bit container) to 10-bit the sequences (Table 23).

To avoid any doubt, we have repeated the PSNR computation with the encoded sequences at 10-bit and the table 24 shows the differences between the two types of sequences (**this will be reference table for the rest of the experiments**):

|  |  |  |
| --- | --- | --- |
|  | **DRLN pre-trained** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] |
| **BasketBallDrive** | -27.09 | 0.8 |
| **BQTerrace** | -49.23 | 1.41 |
| **Cactus** | -23.12 | 0.72 |
| MarketPlace | -3.2 | 0.08 |
| RitualDance | -5.09 | 0.22 |
| **AVG** | **-21.54** | **0.65** |

**Table 24:** BD rates gain/loss when comparing DLRN network pre-trained vs. bicubic interpolation as Table 21 but the bold sequences are native 10-bit. The reference is the HD VVC encoded sequence with bicubic upsampling.

As in the EVC experiment, we have tried two approaches:

1. where the trained architecture DRLN for the user case HD24K is used.
2. where we have refined or re-trained from zero the DRLN architecture using the training set of SD2HD sequences.

The results of both cases are then compared with the pre-trained original DRLN architecture (Table 24).

**Case 1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined patches random** | | **DRLN refined patches non uniform** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BasketBallDrive | -32.12 | 1.00 |  |  |
| BQTerrace | -54.65 | 1.73 |  |  |
| Cactus | -28.47 | 0.97 |  |  |
| MarketPlace | -6.13 | 0.17 |  |  |
| RitualDance | -8.9 | 0.42 |  |  |
| **AVG** | **-26.05** | **0.86** |  |  |

**Table 25:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (patches random)** | | **DRLN trained TFZ (patches non uniform)** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BasketBallDrive | -23.18 | 0.55 |  |  |
| BQTerrace | -46.6 | 1.02 |  |  |
| Cactus | -19.84 | 0.50 |  |  |
| MarketPlace | -3.09 | 0.08 |  |  |
| RitualDance | -1.44 | 0.06 |  |  |
| **AVG** | **-18.85** | **0.44** |  |  |

**Table 26:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**Case 2.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined patches random** | | **DRLN refined patches non uniform** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BasketBallDrive | -32.34 | 1.08 |  |  |
| BQTerrace | -55.30 | 1.86 |  |  |
| Cactus | -27.93 | 0.96 |  |  |
| MarketPlace | -5.77 | 0.16 |  |  |
| RitualDance | -8.58 | 0.40 |  |  |
| **AVG** | **-25.98** | **0.89** |  |  |

**Table 27:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ (patches random)** | | **DRLN trained TFZ (patches non uniform** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| BasketBallDrive | -29.17 | 0.55 |  |  |
| BQTerrace | -51.82 | 1.37 |  |  |
| Cactus | -23.80 | 0.70 |  |  |
| MarketPlace | -2.56 | 0.07 |  |  |
| RitualDance | -3.43 | 0.14 |  |  |
| **AVG** | **-20.96** | **0.57** |  |  |

**Table 28:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**HD24K (VVC codec)**

In this experiment we have tested the SR step on the VVC encoded sequences using the pretrained DRLN approach (vanilla test flavour) with its own weight, for the case HD24K the Table 26 is reporting the final results:

|  |  |  |
| --- | --- | --- |
|  | **DRLN pre-trained** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -6.79 | 0.16 |
| CatRobot | -6.56 | 0.19 |
| DaylightRoad2 | -6.21 | 0.13 |
| FoodMarket4 | 1.01 | -0.03 |
| ParkRunning3 | -3.3 | 0.10 |
| Tango2 | 0.49 | -0.01 |
| **AVG** | **-3.56** | **0.09** |

**Table 29:** BD rates gain/loss when comparing DLRN network pre-trained vs. bicubic interpolation. The reference is the 4K VVC encoded sequence with bicubic upsampling.

Table 27 and 28 are showing respectively the results for the fine tuning of the DRLN network as well as the training from zero.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN refined patches random** | | **DRLN refined patches non uniform** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -9.99 | 0.26 |  |  |
| CatRobot | -10.70 | 0.33 |  |  |
| DaylightRoad2 | -10.03 | 0.23 |  |  |
| FoodMarket4 | 0.02 | 0.00 |  |  |
| ParkRunning3 | -6.79 | 0.23 |  |  |
| Tango2 | -0.66 | 0.02 |  |  |
| **AVG** | **-6.36** | **0.18** |  |  |

**Table 30:** Patches refined training -BD rates gain/loss when comparing DLRN network vs. bicubic interpolation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DRLN trained TFZ patches random** | | **DRLN trained TFZ patches non uniform** | |
| Sequences Name | BD-Rate [%] | BD-PSNR [dB] | BD-Rate [%] | BD-PSNR [dB] |
| Campfire | -3.64 | 0.08 |  |  |
| CatRobot | -4.57 | 0.12 |  |  |
| DaylightRoad2 | -4.31 | 0.09 |  |  |
| FoodMarket4 | 10.11 | -0.33 |  |  |
| ParkRunning3 | -4.11 | 0.13 |  |  |
| Tango2 | 2.15 | -0.05 |  |  |
| **AVG** | **-0.73** | **0.01** |  |  |

**Table 31:** Patches training from zero - BD rates gain/loss when comparing DLRN network trained from zero vs. bicubic interpolation.

**In-loop filter**

The starting point is the the paper A Deep Learning Approach for Multi-Frame In-Loop Filter of HEVC <https://github.com/tianyili2017/MultiFrame-InLoop-Filter>

The group has evaluatend the NN and the results are shown in Table 4: ‘HEVC\_16.5’ is the encoding with HEVC HM 16.5 with all in-loop filters turned on (Deblocking and Sample Adaptive Offset filter). ‘HEVC CNN’ is the configuration that enhances the frame in which the neural network is always active; MIF-NET is the configuration in which a metric chooses among the best approaches. Table 4 shows that the MIF-NET configuration is the best because the PSNR is higher, with the same bitrate. CNN does a better job than HEVC filters. The assessment on the NN gives BD-rate: -9.26.

This approach is implemented in HEVC, and we have done the porting into the EVC codec.

The group has done the following work:

* Porting from HEVC to EVC
  + Image extraction before EVC filtering
  + Image enhancement with the MIF-NET neural network
  + Integrate the enhanced image into the EVC pipeline

MIF-NET requires, in the training and testing phase, the retrieval of partition information from the EVC codec. Currently the group is considering replacing it with an attention mechanism.

The group decided to take a step back and work on the HEVC codec because in the article author's github it uses two reference images, while the previous porting to EVC uses only one image. Two reference images ensure better results in image enhancement. The purpose of replacing the partition information with the attention mechanism is that no signals will need to be sent to the decoder, thus saving bits. The group then completed the process of writing the training part of the neural network, which was not available on github at the time.

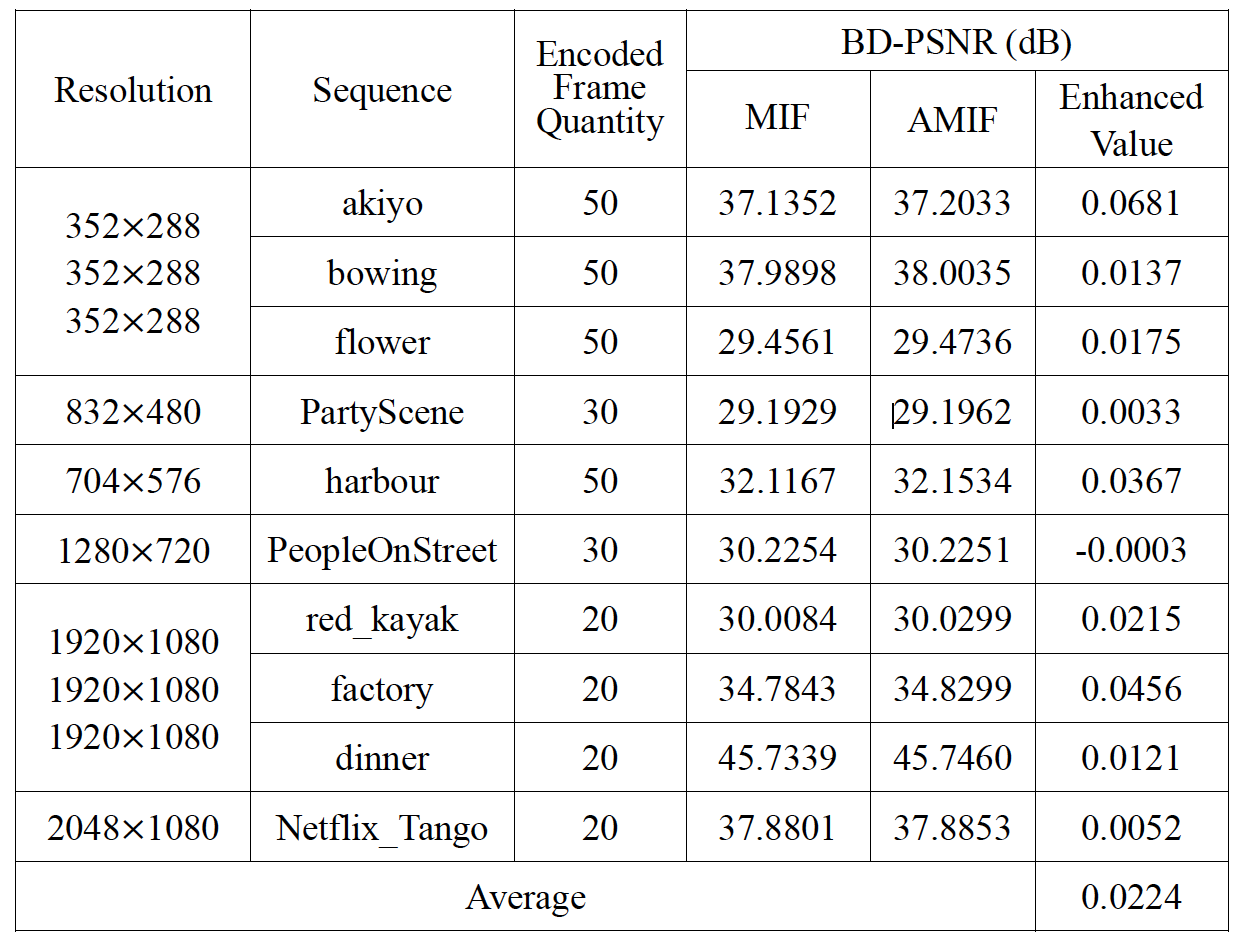
Table 5 shows the preliminary result. As a first step, the attention mechanism was built on top of the quadtreee partition meahcnis. Table 5 shows a BD-rate improvement of 0.0225% with the addition of the attention mechanism (AMIF) compared with the quadtree partition (MIF) alone.

Table 6 shows the increase in complexity due to the attention mechanis neural network.

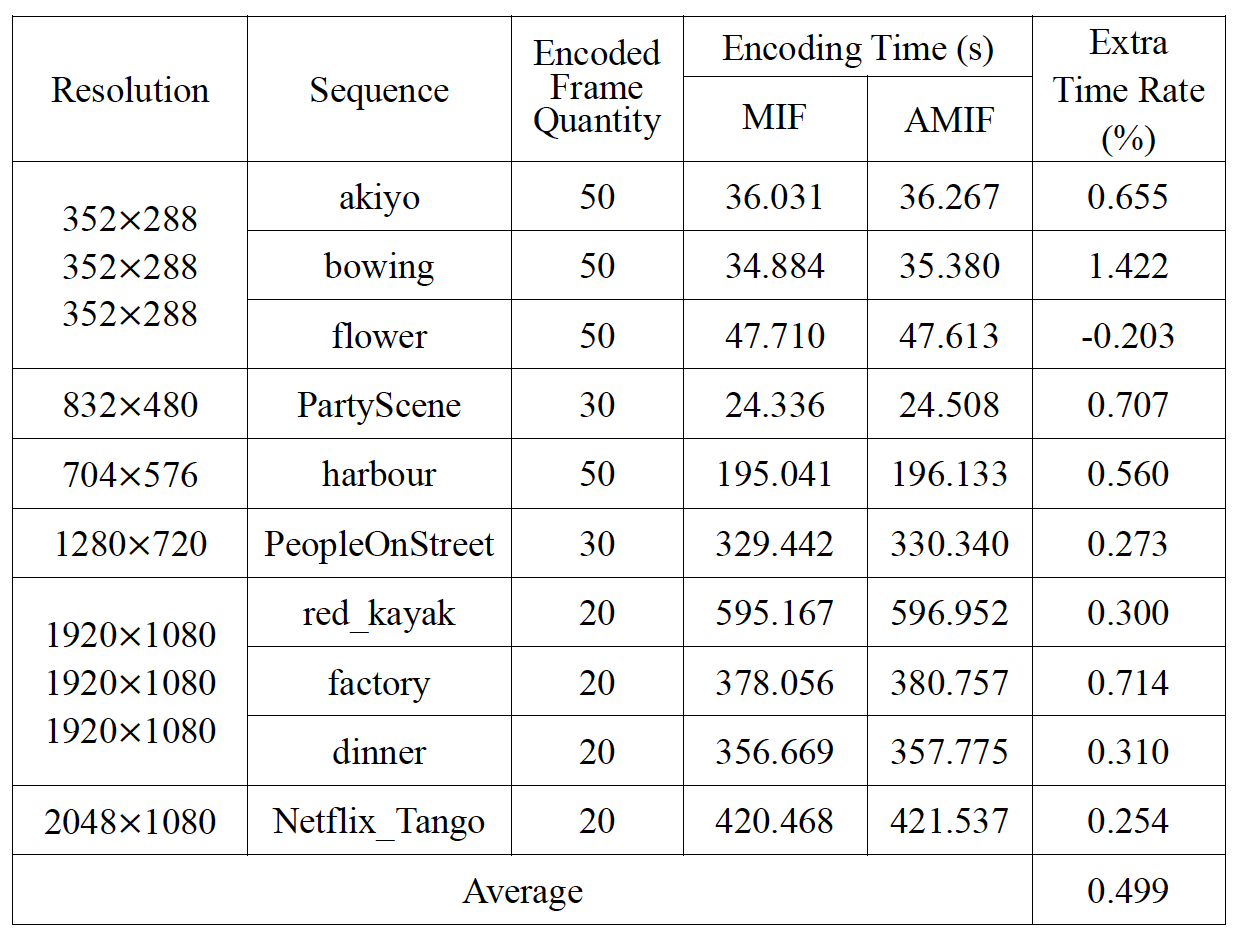
Immagine che contiene tavolo

Descrizione generata automaticamente

**Table 4:** Preliminary results of the assessment of the NN



**Table 5:** BD-rate performance comparison for in-loop filer



**Table 6:** Encoding time complexity comparison for in-loop filer

**Inter prediction**

The group started the work on inter prediction with the aim of improving it by exploiting deep learning.

The group has chosen:

Paper 1:

Neural\_Reference\_Synthesis (NRS) for\_Inter\_Frame\_Coding

as a starting point for neural networks applied to inter prediction. The code is available and we have contacted the authors to ask for support in better understanding the code.

We had a presentation by the author of the article who shared with us his findings and his thoughts on the transition from HEVC to EVC.

We invited the author to assist in the integration of their NRS module into EVC. But they didn’t participate in the last 2 meeting cycles. So, we have planned to contact the authors of further shortlisted papers/journals promising inter prediction improvements.

Update: We got the source code which is obsolete and didn’t run. No further support available from their side. So we paused this activity.

Paper 2:

Affine Transformation-Based Deep Frame Prediction by Hyomin Choi , Member, IEEE, and Ivan V. Bajić , Senior Member, IEEE

We had 2 rounds of technical calls with the authors and got the reference software showcasing the unit level functionality (MIT License, Copyright (c) 2017-2022 Hyomin Choi, Permission granted, free of charge, to any person obtaining a copy of this software). Able to run this unit level software under Ubuntu pytorch environment with the example provided.

Requested the authors to provide the HEVC integrated setup as well as training related software. But these are not available for evaluation. So, these software need to be developed again with more research and development effort.

Paper 3:

<http://cs231n.stanford.edu/reports/2022/pdfs/29.pdf>

Video Frame Prediction with Deep Learning, with Convolutional LSTM network and a GAN model. Each of these models outputs a sequence of future video frames conditioned on a sequence of past video frames

This activity is currently in standby.

**Complexity of the tools**

The group is exploring the significance of understanding the complexity associated with neural networks applied to video coding.

The complexity of neural networks directly affects the computational resources required for training and inference. Deep and wide networks with numerous parameters demand substantial computational power and memory resources. Understanding the trade-offs between model complexity and available resources is crucial for developing a video codec.

It was computed the complexity of the SR neural network using the pythorch internal libraries, and the results for bot transformation SD-HD and HD-4K are reported below:

# Evaluating SD-HD (Channels:3, Model: **original**)

Average Inference time on 20 inputs is: 0.6479169964790344 Seconds

Number of model parameters: 34430155

FLOPs (M): 35398323.50976

# Evaluating HD-4K (Channels:3, Model: **original**)

Average Inference time on 20 inputs is: 6.225010085105896 Seconds

Number of model parameters: 34430155

FLOPs (M): 154130206.72176

We may have noticed that the experiment is done using the original network, using an input of three channels. In our case, since we are working on the luminance channel, all the three channels are replicating the luminance value.

Potentially, the complexity can be reduced by modifying the network to work with a single channel input.

**New metrics**

The group proposes to use the following objective metrics to measure the similarity between two images or video frames. These metrics are widely used in image and video processing applications, such as image compression, quality assessment and image restoration. They are full-reference metrics, i.e., they require access to the original undistorted image for comparison.

**PSNR**

The Peak Signal-to-Noise Ratio (PSNR) is a metric used to quantify the fidelity of a signal representation, calculated as the ratio of the maximum possible power of a signal to the power of corrupting noise. It is often employed to assess the quality of digital signal transmission. In the case of digital images, each pixel can be considered as a component of a signal with 8-bit or 10-bit RGB values.

Where, is the maximum valid value for a pixel and MSE is the Mean Squared error between the high resolution image and the super resolved image. It is a pixel-by-pixel comparison over the entire image. The PSNR index ranges between 0 and ∞. Acceptable value is >40.

**SSIM**

SSIM stands for Structural Similarity Index. SSIM takes into account both structural information and image pixel values, with the goal of capturing perceived visual quality.

The SSIM algorithm compares the similarities in luminance, contrast and structure between two images. To this end, it divides the images into small windows and calculates the similarity measures for each window. The final SSIM score is then calculated as the average of the similarity measures across all windows.

The SSIM index varies between 0 and 1, where 1 indicates perfect similarity between the images, while 0 means no similarity at all. The closer the SSIM score is to 1, the more similar the images are perceived to be.

**VMAF**

VMAF stands for Video Multimethod Assessment Fusion. It is a widely used objective video quality metric developed by Netflix.

VMAF is based on a machine learning model that was trained using a large dataset of subjective human ratings. The model takes into account various characteristics of video frames, including spatial and temporal factors as well as perceptual properties such as contrast and texture.

The VMAF score is generally reported on a scale from 0 to 100, where higher scores indicate better perceived quality.

The computation of the new metrics is ongoing.

**Video coding standard**

Video encoding standards play a key role in the world of digital multimedia, providing the basis for efficient compression, transmission and playback of video content. Having achieved a 25% improvement in encoding gain over the basic MPEG5-EVC profile, we are beginning to explore what the video standard looks like. In particular, the discussion focuses on the relationship between downsampling and upsampling of Super-Resolution (SR) (Figure 17).

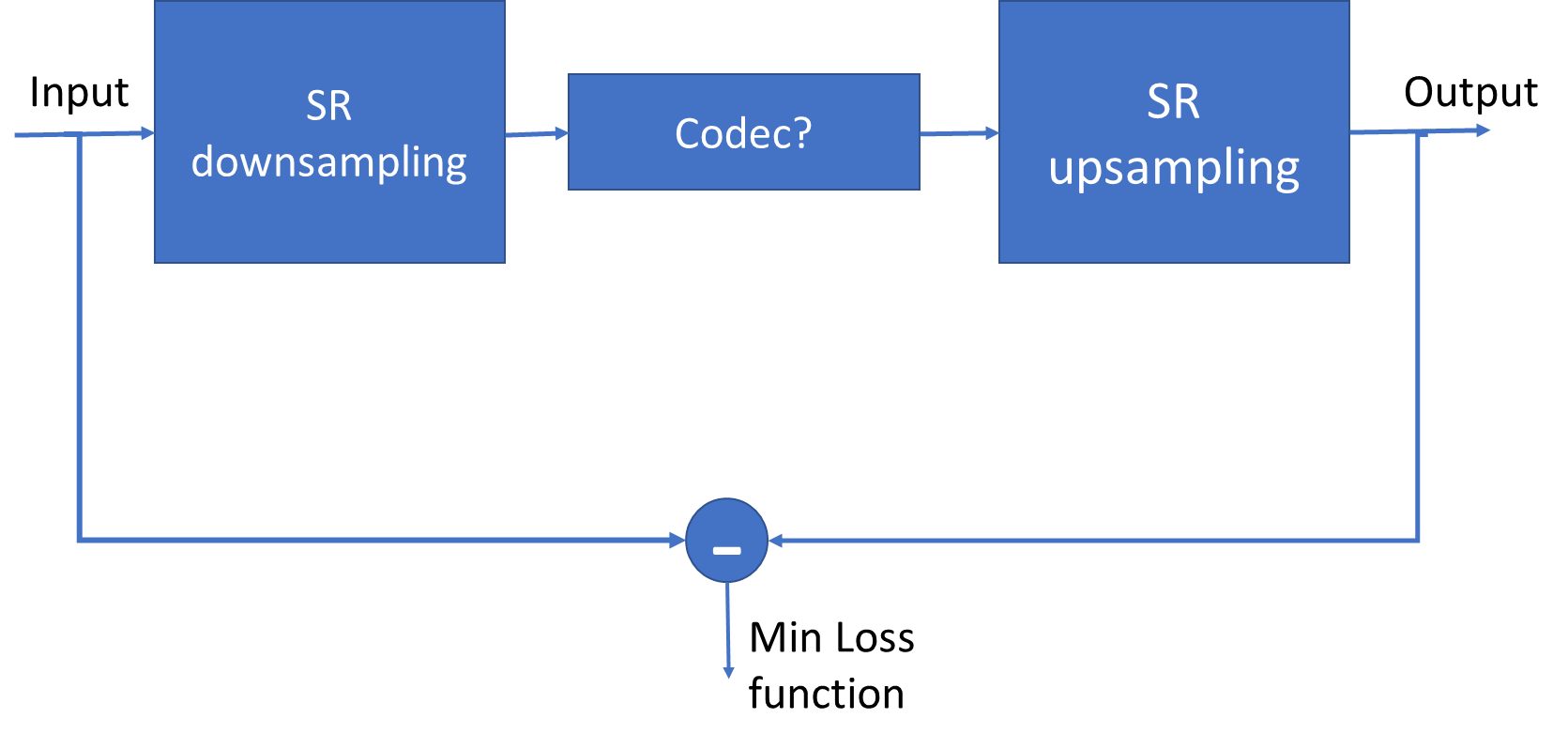


Figure 17: schema of a possible relation between SR downsampling and SR upsampling

In the context of video encoding, downsampling is an operation that takes place on the sides of the encoder. It involves reducing the spatial resolution of video frames to minimise the size of the data. Although this operation significantly reduces the size of the data, it inevitably leads to a loss of fine detail. This is where Super-Resolution (SR) technology comes into play. SR algorithms, on the decoder side, specialise in upsampling low-resolution images to recover lost detail and improve visual quality.

We believe that this connection between downsampling and upsampling SR is beneficial in the design of a video encoding standard.

The next steps are:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool** | **Date** | **Topic** | **Who** |
| Paper | x meeting cycle | Writing a paper | All |
| SR | 2 meeting cycle | Training SR to the VVC codec | Alessandro, Mattia, Giovanni |

**Future Plan**

* motion compensation: improve the motion compensation using NN architecture
* quantization: uniform scalar quantization used in classical video codec standards does not conform to the characteristics of the human visual system. It is possible to use a quantization strategy based on neural networks.
* arithmetic encoder: improve the CABAC performance by leveraging NN to directly predict the probability distribution of intra modes instead of the handcraft context models