|  |
| --- |
| **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION INTERNATIONALE DE NORMALISATION ISO/IEC JTC 1/SC 29/WG 5 MPEG JOINT VIDEO CODING TEAM WITH ITU-T SG 16** |
| **ISO/IEC JTC 1 / SC 29 / WG 5 N 165** |
| **Mainz – 20–28 October 2022** |
| |  |  | | --- | --- | | **Title:** | **Description of algorithms and software in Neural Network-based Video Coding (NNVC)** | | **Source:** | **Convenor (Jens-Rainer Ohm)** | | **Type:** | **General** | | **Subtype:** | **Other** | | **Status:** | **Approved** | | **Date:** | **2022-12-02** | | **Expected Action:** | **Info** | | **Action due date:** | **N/A** | | **No. of pages** | **10** (without this cover page) | | **Email of convenor:** | **ohm @ ient . rwth-aachen . de** | | **Committee URL:** | **https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-5** | |

|  |  |
| --- | --- |
| **文本  中度可信度描述已自动生成徽标  描述已自动生成Joint Video Experts Team (JVET)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29**  28th Meeting, Mainz, DE, 20–28 October 2022 | Document: JVET-AB2019-v1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Algorithm Description for Neural Network-based Video Coding 3 (NNVC 3)** | | |
| *Status:* | Output document of JVET | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Yue Li  Hongtao Wang  Liqiang Wang  Franck Galpin Jacob Ström | Email: | [yue.li@bytedance.com](mailto:yue.li@bytedance.com) [hongtaow@qti.qualcomm.com](mailto:hongtaow@qti.qualcomm.com) [liqiangwang@tencent.com](mailto:liqiangwang@tencent.com) [franck.galpin@interdigital.com](mailto:franck.galpin@interdigital.com) [jacob.strom@ericsson.com](mailto:jacob.strom@ericsson.com) |
| *Source:* | Bytedance, Qualcomm, Tencent, InterDigital, Ericsson. | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This document is Neural Network-based Video Coding 3 (NNVC 3) software algorithm description. It includes the coding features and encoding methods implemented in NNVC-3.0 software that are under coordinated exploration study by the Joint Video Exploration Team (JVET) of ITU-T VCEG and ISO/IEC MPEG as potential neural network video coding technology beyond the capabilities of VVC. The groups are working together on this exploration activity in a joint collaboration effort known as the Joint Video Exploration Team (JVET) to evaluate compression technology designs proposed by their experts in this area.

Contents

[Abstract 1](#_Toc120011125)

[1 Introduction 2](#_Toc120011126)

[2 Scope 2](#_Toc120011127)

[3 Algorithm description of Neural Network-based Video Coding Software 2](#_Toc120011128)

[3.1 Neural network-based loop filter set 0 2](#_Toc120011130)

[3.1.1 Pre-processing and post-processing of chroma 2](#_Toc120011131)

[3.1.2 Neural network 3](#_Toc120011132)

[3.1.3 Combination with conventional filters 3](#_Toc120011133)

[3.1.4 Mode selection 4](#_Toc120011134)

[3.1.5 Base QP adjustment 4](#_Toc120011135)

[3.1.6 Inference details 5](#_Toc120011136)

[3.2 Neural network-based loop filter set 1 5](#_Toc120011137)

[3.2.1 Neural network for luma component 5](#_Toc120011138)

[3.2.2 Neural network for chroma component 6](#_Toc120011139)

[3.2.3 Adaptive inference granularity 6](#_Toc120011140)

[3.2.4 Parameter selection 6](#_Toc120011141)

[3.2.5 Residue scaling 7](#_Toc120011142)

[3.2.6 Combination with deblocking filter 7](#_Toc120011143)

[3.2.7 Encoder-only optimization 7](#_Toc120011144)

[3.2.8 Inference details 7](#_Toc120011145)

[3.3 Small ad-hoc deep learning (SADL) library 8](#_Toc120011146)

[4 Training description of Neural Network-based Video Coding Software 9](#_Toc120011147)

[4.1 Neural network-based loop filter set 0 9](#_Toc120011148)

[4.2 Neural network-based loop filter set 1 9](#_Toc120011149)

[5 References 10](#_Toc120011150)

# Introduction

This document provides algorithm description, encoding methods, and training methods of the neural network-based coding tools implemented in Neural Network-based Video Coding 3 (NNVC 3) software. The neural network-based tools (NN-based tools) are to enhance or replace conventional modules in the existing VVC design [1][2]. The implementation of NN-based tools in NNVC 3 are based on Small Ad-hoc Deep Learning (SADL) library [3]. It is recommended to refer to [4] for the detailed information of VTM-11.0, which is the base of NNVC 3 and [3] for the detailed usage of SADL.

# Scope

The NNVC-3.0 reference software is provided to demonstrate a reference implementation of encoding techniques and the decoding process, as well as the training methods for neural network-based video coding explored in JVET. The reference software can be accessed via

<https://vcgit.hhi.fraunhofer.de/jvet-ahg-nnvc/VVCSoftware_VTM>.

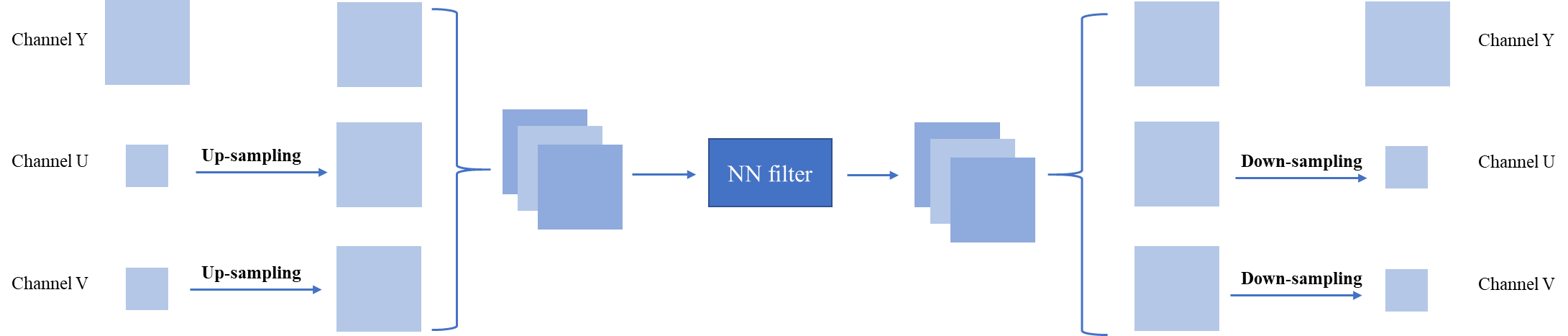
This document provides an algorithm description, an encoder-side description, and a training description of the NNVC-3.0, which serves as a tutorial for the algorithm and encoding model implemented in the NNVC-3.0 software, as well as the training method of the tools in NNVC-3.0 software. The purpose of this document is to share a common understanding of the coding features and the reference encoding methods supported in the NNVC-3.0 software, in order to facilitate the assessment of the technical impact of new technologies during the exploration work.

# Algorithm description of Neural Network-based Video Coding Software

## Neural network-based loop filter set 0

### Pre-processing and post-processing of chroma

In filter set 0, the filter with a single model is designed to process three components. Since the resolutions of luma and chroma are different, pre-processing and post-processing steps are introduced to up-sample and down-sample chroma components respectively as shown in Figure. 1. In the resampling process, the nearest-neighbor interpolation method is used.



*Figure. 1 the pre-processing and post-processing units*

### Neural network

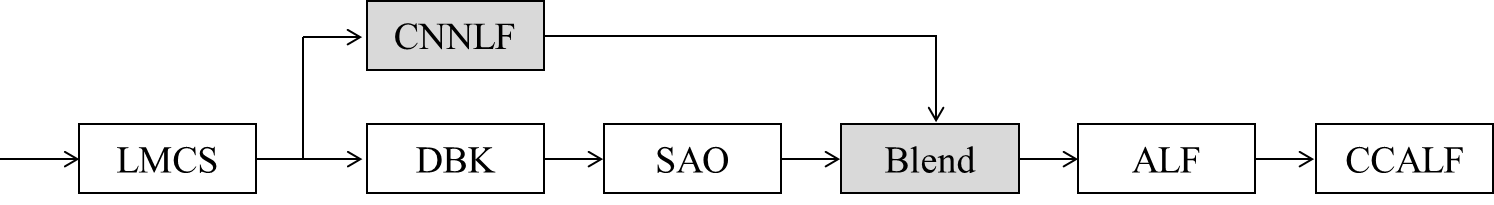
The network structure of the CNN filter is shown in Figure. 2. Along with the reconstructed image (rec\_yuv), additional side information is also fed into the network, such as the prediction image (pred\_yuv), slice QP, base QP and slice type. In the ResBlock, the number of channels firstly goes up before the activation layer, and then goes down after the activation layer. Specifically, K and M are set to 64 and 160 respectively, and the number of Resblock is set to 32.

图示

描述已自动生成

*Figure. 2*. *Architecture of the CNN in filter set 0.*

### Combination with conventional filters



*Figure. 3*. *Implementation of the CNN in filter set 0.*

As shown in Figure.3, the reconstructed samples before DBK are fed into the CNN based filter (CNNLF), then final filtered samples are generated by blending the result of CNNLF and SAO. This blending process can be briefly formulated as:

There are four candidates, 1, 0.75, 0.5 and an adaptive weight, for the blending weight. With regard to the adaptive weight, its derivation is based on least square method. If the adaptive weight is selected, the blending weight is signaled for each color component in the slice header.

### Mode selection

The CNN filter can be turned on/off at the CTU level and slice level. For each enabling type, there are four blending ways. Therefore, there are nine modes to be evaluated by RDO at encoder. The final selected mode would be signaled in the slice header.

Table 1. Parameter selection of filter set 0

|  |  |  |
| --- | --- | --- |
| Mode | On/off type | Blending weight (w) |
| 0 | Disable at slice level | None |
| 1 | Enable at slice level | Adaptive weight |
| 2 | 1 |
| 3 | 0.75 |
| 4 | 0.5 |
| 5 | Enable at CTU level | Adaptive weight |
| 6 | 1 |
| 7 | 0.75 |
| 8 | 0.5 |

### Base QP adjustment

Base QP is fed into the CNN filter as shown in Figure. 2. To improve adaptation, an offset can be added to the base QP (the adjusted base QP is used as the input to the NN filter) at slice level. The offset candidates are {-5, 5}. For example given the offset -5, the actual input base QP to the filter becomes (BaseQP - 5) for the current slice.

**Encoder approach**

The proposed encoder only filters one out of every four CTUs during the process of selecting the best base QP offset to save encoding time. As shown in Figure 4, only shaded CTUs are considered for calculating distortions of using different BaseQP candidates {BaseQP, BaseQP-5, BaseQP+5}. After the candidate with the smallest cost is selected, the encoder filters the rest of CTUs (non-shaded ones in Figure. 4) by appying the best offset to the base QP.

图片包含 背景图案

描述已自动生成

*Figure. 4. Encoder optimization 2.*

### Inference details

SADL (see Section 1.3) is used for performing the inference of the CNN filters. Both floating point-based and fixed point-based implementations are supported. In the fixed-point implementation, both weights and feature maps are represented with int16 precision using a static quantization method. The network information in the inference stage is provided in Table 2.

Table 2. Network Information of filter set 0 in Inference Stage

|  |  |  |
| --- | --- | --- |
| **Network Information in Inference Stage** | | |
| Mandatory | HW environment: | |
| GPU Type | N/A |
| Framework: | SADL |
| Number of GPUs per Task | 0 |
|  |  |
| Number of Parameters (Each Model) | 1.9M |
| Total Parameter Number | 1.9M |
| Parameter Precision (Bits) | float: 32  int: 16 |
| Memory Parameter (MB) | float: 7.6MB  int: 3.8MB |
| Multiply Accumulate (kMAC/pixel) | 485 (assuming frame-level input)  615 (assuming block-level input) |
| Optional |  |  |
| Total Conv. Layers | 101 |
| Total FC Layers | 0 |
| Total Memory (MB) |  |
| Batch size: | 1 |
| Patch size | 144144 |
| Changes to network configuration or weights required to generate rate points |  |
| Peak Memory Usage |  |
| Other information: |  |

## Neural network-based loop filter set 1

### Neural network for luma component

The backbone of the CNN filter is shown in Figure. 5. The calculation process in the attention module can be written as:

*F\_out = F\_in f (Rec, Pred, BS, QP) + F\_in*

where *F\_in and F\_out* denote the input and the output of the attention module, respectively. *Rec*, *Pred*, *BS*, and *QP* stand for the reconstruction, the prediction, the boundary strengths, and the sequence-level input quantization parameter respectively. *f* comprises 2 convolutional layers, where an activation function is applied after the first convolutional layer. The objective of *f* is to generate a spatial attention map from external information, which then recalibrates the feature maps *F\_in. Note that the attention module has been removed from the luma intra model.*

The input of the network comprises reconstruction, prediction, boundary strengths, and QP. The numbers of feature maps and attention residual blocks are set as 96 and 8 respectively.



*Figure. 5*. *(a)* *Architecture of the CNN in filter set 1. M denotes the number of feature maps. N stands for the number of samples in one dimension. (b) Construction of Attention Residual Block in (a)*.

### Neural network for chroma component

Luma information is taken as additional input for the in-loop filtering of chroma. In addition, the partitioning information is fed into the chroma intra model. Considering the resolution of luma is higher than chroma in YUV 4:2:0 format, features are first extracted separately from luma and chroma. Then luma features are down-sampled and concatenated with chroma features. Regarding network backbone, chroma components use the same one as luma.

### Adaptive inference granularity

The granularity of the filter determination and the parameter selection is dependent on resolution and QP. Given a higher resolution and a larger QP, the determination and selection will be performed in a larger region.

### Parameter selection

Each slice or block could determine whether to apply the CNN-based filter or not. When the CNN-based filter is determined to be applied to a slice/block, which conditional parameter from a candidate list including three candidates derived from QP could be further decided. Denote the sequence level QP as q, the candidate list includes conditional parameters {Param\_1, Param\_2, Param\_3}. For low temporal layers, Param\_1 = q, Param\_2 = q5, Param\_3 = q10. For high temporal layers, Param\_1 = q, Param\_2 = q5, Param\_3 = q5. In other words, the third candidate is different across different temporal layers.

The selection process is based on the rate-distortion cost at the encoder side. Indication of on/off control as well as the conditional parameter index, if needed, are signalled in the bitstream. Figure. 6 shows the diagram of parameter selection at encoder and decoder sides. All blocks in the current frame need to be processed with three conditional parameters first. Then five costs, i.e. Cost\_0, ..., Cost\_5, are calculated and compared against each other to achieve optimum rate-distortion performance. In Cost\_0, CNN-based filter is prohibited for all blocks. In Cost\_i, {i = 1, 2, 3}, the parameter Param\_i is used for all blocks. In Cost\_4, different blocks may prefer different parameters, and the information regarding whether to use CNN-based filter or which parameter to be used is signaled for each block. At decoder side, whether to use CNN-based filter or which parameter to be used for a block is based on the Param\_Id parsed from the bit-stream as shown in Figure. 6 (b).

Note that for all-intra configuration, parameter selection is disabled while filter on/off control is still preserved. A shared conditional parameter is used for the two chroma components to ease the burden in worst case at decoder side. In addition, the max number of conditional parameter candidates could be specified at encoder side.



*Figure. 6*. *(a)* *Parameter selection at encoder side. (b) Parameter selection at decoder side*.

### Residue scaling

When a NN filter is being applied to reconstructed pictures, a scaling factor is derived and signaled for each color component in the slice header. The derivation is based on least square method. The difference between the input samples and the NN filtered samples (residues) are scaled by the scaling factors before being added to input samples.

### Combination with deblocking filter

To enable a combination with deblocking, the input samples used in the residual scaling is the output of deblocking filtering. The residual scaling process is shown below, where and refer to the outputs of NN filtering and deblocking filtering respectively.

=

### Encoder-only optimization

Different from NNVC-2.0, EncDbOpt is also enabled for AI configuration.

For a better estimation of rate-distortion (RD) cost in the case the NN filter is used, the proposed encoder introduces NN-based filtering into the rate-distortion optimization (RDO) process of partitioning mode selection. Specifically, a refined distortion is calculated by comparing the NN filtered samples and the original samples. The partitioning mode with the smallest rate-refined distortion cost is selecte as the optimal one. To reduce complexity, several fast algorithms are applied. First, NN model is simplified by using a less number of residual blocks. Second, parameter selection is not allowed for the NN filtering in the RDO process Third, the proposed technique is only applied to the coding units with height and width no larger than 64. The NN filter used in the RDO process is also implemented with SADL using fixed point-based calculation. This NN-based encoder-only method is disabled by default.

### Inference details

SADL (see Section 1.3) is used for performing the inference of the CNN filters. Both floating point-based and fixed point-based implementations are supported. In the fixed-point implementation, both weights and feature maps are represented with int16 precision using a static quantization method. The network information in the inference stage is provided in Table 3.

Table 3. Network Information of filter set 1 in Inference Stage

|  |  |  |
| --- | --- | --- |
| **Network Information in Inference Stage** | | |
| Mandatory | HW environment: | |
| GPU Type | N/A |
| Framework: | SADL |
| Number of GPUs per Task | 0 |
|  |  |
| Total Parameter Number | 1.56M/model, 4 models in total for all tests |
| Parameter Precision (Bits) | float: 32  int: 16 |
| Memory Parameter (MB) | float: 6.3MB/model, 4 models  int: 3.1MB/model, 4 models |
| Multiply Accumulate (kMAC/pixel) | 539 (assuming frame-level input)  682 (assuming block-level input) |
| Optional |  |  |
| Total Conv. Layers | 25+16 |
| Total FC Layers | 0 |
| Total Memory (MB) |  |
| Batch size: | 1 |
| Patch size | 144144, 272272 |
| Changes to network configuration or weights required to generate rate points |  |
| Peak Memory Usage |  |
| Other information: |  |

## Small ad-hoc deep learning (SADL) library

SADL (Small Ad-hoc Deep-Learning Library) is a header only small library for inference of neural networks. SADL provides both floating-point-based and integer-based inference capabilities. The inference of neural networks in NNVC is based on the SADL.

The table below summarizes the framework characteristics.

Table 4. Characteristics of SADL

|  |  |
| --- | --- |
| Language | Pure C++, header only. |
| Footprint | 5200 LOC, library ~200kB, no dependency |
| Optimization | Some SIMD at hot spots (best effort) |
| Compatibility | ONNX file |
| Layer Supports | constants, add, maxPool, matMul, reshape, ReLU, conv2D, mul, concat, max, leakyReLU, shape, expand, transpose, conv2DTranspose |
| Type support | float, int32, int16, int8 |
| Quantization | Support adaptive quantizer per layer |
| License | BSD 3-Clause |

NNVC repository uses SADL as a submodule, pointing to the repository here: <https://vcgit.hhi.fraunhofer.de/jvet-ahg-nnvc/sadl>.

Documentation is available in the doc directory of the repository.

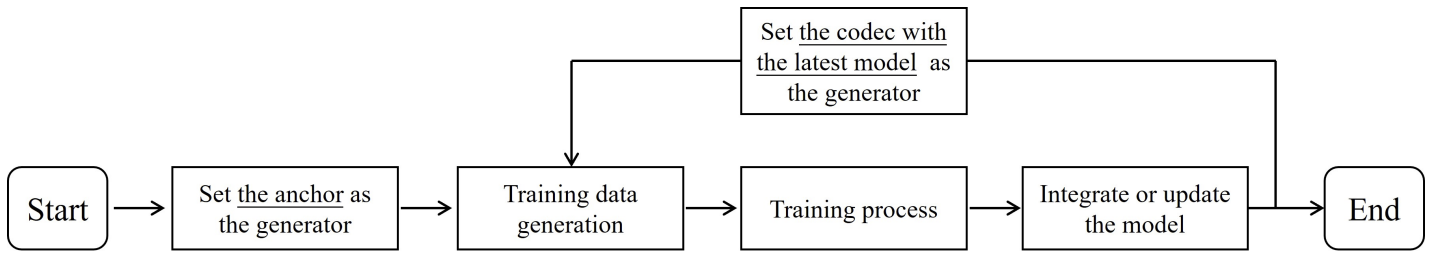
# Training description of Neural Network-based Video Coding Software

The training of NN-based tool typical involve three steps: compressing a dataset and dumping necessary training data, loading training data, training using the loaded training data. The first two steps are usually based on the data dumper and data loader in NNVC-3.0, while the third step relies on the training method designed for each specific tool.

A high-level description of training methods for NN-based tools is provided below. To reproduce the training of a specific tool, it is recommended to refer to the training scripts in the NNVC-3.0 software.

## Neural network-based loop filter set 0

In order to effectively enhance the NN model generalization, an iterative training method is designed to better maintain consistency between the training process and the inference process. As shown in Figure 7, the proposed iterative training method contains the initial training stage and the iterative training stage. In the initial training stage, the training data is generated by the anchor configured with the common test conditions. In the iterative training stage, the NN model obtained from the previous training process is integrated into anchor and the training data is generated by the NN-filter based codec.



*Figure. 7. The Iterative training method*

In the initial training stage, NNVC-3.0 (--NnlfOption=0) is used to compress all training images under all-intra and random-access configuration, respectively. The reconstructed images together with additional side information are generated and utilized to train the NN filter.

In the iterative training stage, NNVC-3.0 (--NnlfOption=1) with the integrated NN model from previous training stage is used to compress all training images under random-access configuration and the training data for I slices is still used from initial training stage. Theoretically, the more times of the iterative training stages, the better performance. However, only up to two times of training stages including the initial training stage are used.

In addition to the above real-iterative training method, pseudo-iterative method training can also be used. That’s to say, the training data used for each training stage can be generated by the codec whose performance is comparable with the codec enhanced by the latest training model, then the initial training stage can be skipped.

## Neural network-based loop filter set 1

To effectively train the NN models, an iteratively conducted three-stage training is adopted to better align the settings during training and testing as shown in Figure. 8. The first-stage training gives the models for intra slice. In the second-stage training, the intra models from the first stage are integrated into VTM to generate the training data for inter slice. Similarly, the third stage plugs the intra-slice models from the first stage and inter-slice models from the second stage into VTM anchor to generate the training data for inter slices again.



*Figure. 8. Iterative training, where VTM encoder is used to generate training samples in phase I. Then trained models from the last phase are integrated into VTM for generating training samples in the next phase. Note that the blue and yellow lines indicate integrations of the intra and inter slice models respectively.*

In training phase I, NNVC-3.0 (--NnlfOption=0) is used to compress training images under all-intra configuration. The reconstructed images together with other auxiliary information are collected and utilized for training intra frame filters.

In training phase II, NNVC-3.0 (--NnlfOption=2) equipped with the models from the previous training phsse is used to compress training videos under random-access setting. That is to say, the intra frames will be processed by the filters obtained in training phase I. Then the reconstruction together with other auxiliary information are collected and utilized for training inter frame filters.

In training phase III, NNVC-3.0 (--NnlfOption=2) equipped with the models from the previous training phases is used to compress training videos under random-access setting. That is to say, the intra frames will be processed by the filters obtained in training phase I and the inter frames will be processed by the filters obtained in training phase II. Then the reconstruction together with other auxiliary information are collected and utilized for training the final inter frame filters.

# References

1. B. Bross, J. Chen, S. Liu, and Y.-K. Wang "Versatile Video Coding Editorial Refinements on Draft 10," document JVET-T2001, 20th JVET meeting: by teleconference, 7 October – 16 October 2020.
2. J. Chen, Y. Ye, S. Kim "Algorithm description for Versatile Video Coding and Test Model 11 (VTM 11) ," document JVET-T2002, 20th JVET meeting, by teleconference, 7 October - 16 October 2020.
3. <https://vcgit.hhi.fraunhofer.de/jvet-ahg-nnvc/sadl>.
4. <https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-11.0>.