 ISO/IEC JTC 1/SC 29/WG 2 N282

**ISO/IEC JTC 1/SC 29/WG 2  
MPEG Technical requirements   
Convenorship: SFS (Finland)**

**Document type:** Output Document

**Title:** Call for Proposals on Feature Compression for Video Coding for

Machines

**Status:** Approved

**Date of document:** 2023-04-28

**Source:** ISO/IEC JTC 1/SC 29/WG 2

**Expected** **action:** None

**Action** **due date:** None

**No. of pages:** 48 (with cover page) + XLS file

**Email of Convenor:** igor.curcio@nokia.com

**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-2>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 2**

**MPEG TECHNICAL REQUIREMENTS**

**ISO/IEC JTC 1/SC 29/WG 2 N282**

**Antalya, Turkey – April 2023**

|  |  |
| --- | --- |
| **Title** | **Call for Proposals on Feature Compression for Video Coding for Machines** |
| **Source** | **WG 2, MPEG Technical requirements** |
| **Status** | **Approved** |
| **Serial Number** | **22664** |

**Table of Contents**

[1 Introduction 5](#_Toc133523684)

[2 Who may participate 6](#_Toc133523685)

[3 Code of conduct and rules of engagement 6](#_Toc133523686)

[4 Source code and IPR 6](#_Toc133523687)

[5 Testing Fee 6](#_Toc133523688)

[6 Definitions 6](#_Toc133523689)

[7 Documents of CfP package 6](#_Toc133523690)

[8 Submission Process 7](#_Toc133523691)

[8.1 CfP Timeline table 7](#_Toc133523692)

[8.2 Envisioned Timeline for the FCVCM Standard 8](#_Toc133523693)

[8.3 Register your participation 8](#_Toc133523694)

[8.4 Mandatory Equipment, Software and Data Components 9](#_Toc133523695)

[8.5 Access to Test Material 9](#_Toc133523696)

[8.6 Conduct Objective Evaluations 10](#_Toc133523697)

[8.7 Details of the Submission 11](#_Toc133523698)

[8.7.1 Submission details for datasets 12](#_Toc133523699)

[8.7.2 Submission timeline 15](#_Toc133523700)

[8.8 Cross-checking of results 16](#_Toc133523701)

[8.9 Submit Proponent Documentation 17](#_Toc133523702)

[8.10 Evaluate CfP Submissions and Select Technology 18](#_Toc133523703)

[8.11 Submit WD Specification and RM Source Code 18](#_Toc133523704)

[9 Call Administrator 19](#_Toc133523705)

[10 Test Administrator 19](#_Toc133523706)

[11 Email reflector 19](#_Toc133523707)

[12 References 20](#_Toc133523708)

[13 Copyright Header for FCVCM Reference Software 20](#_Toc133523709)

[Appendix A: Questionnaire 22](#_Toc133523710)

[Appendix B: Detailed description of test datasets 23](#_Toc133523711)

[B.1 Dataset 1: Tencent Video Dataset (TVD) 23](#_Toc133523712)

[B.2 Dataset 2: OpenImages v6 23](#_Toc133523713)

[B.3 Dataset 3: SFU-HW-Object-v1 23](#_Toc133523714)

[B.4 Dataset 4: HiEve videos 25](#_Toc133523715)

[Appendix C: Anchor metrics 27](#_Toc133523716)

[C.1 Bitrate measurement 27](#_Toc133523717)

[C.2 Task: Object Tracking 27](#_Toc133523718)

[C.3 Task: Instance segmentation / Object detection 27](#_Toc133523719)

[C.4 Runtime Measurement 28](#_Toc133523720)

[Appendix D: Inference and Training information 29](#_Toc133523721)

[D.1 Inference information 29](#_Toc133523722)

[D.2 Training information 29](#_Toc133523723)

[Appendix E Feature Anchor generation procedure 31](#_Toc133523724)

[E.1 Feature anchor generation 31](#_Toc133523725)

[E.2 Anchor generation for OpenImages object detection and instance segmentation tasks (feature) 33](#_Toc133523726)

[E.3 Anchor generation for object tracking task: TVD videos and HiEve videos (feature) 35](#_Toc133523727)

[E.4 Anchor generation for SFU-HW videos for object detection (feature) 37](#_Toc133523728)

[E.5 Informative anchor generation for instance segmentation task (Image) 39](#_Toc133523729)

[E.6 Informative anchor generation for object tracking task: TVD (Video) 39](#_Toc133523730)

[E.7 Informative anchor generation for video object detection: SFU-HW-Objects-v1 40](#_Toc133523731)

[E.7 Informative anchor generation for video object tracking: HiEve 40](#_Toc133523732)

[Appendix F Anchor generation environment 41](#_Toc133523733)

[F.1 Object tracking on TVD videos (feature anchor) 41](#_Toc133523734)

[F.2 Instance segmentation on OpenImages (feature anchor) 41](#_Toc133523735)

[F.3 Object detection on OpenImages (feature anchor) 41](#_Toc133523736)

[F.4 Object detection on SFU videos (feature anchor) 42](#_Toc133523737)

[F.5 Object tracking on HiEve videos (feature anchor) 42](#_Toc133523738)

[F.6 Object tracking on TVD (informative video anchor) 42](#_Toc133523739)

[F.7 Instance segmentation / object detection on OpenImages (informative anchor) 43](#_Toc133523740)

[F.8 Object detection on SFU videos (informative anchor) 43](#_Toc133523741)

[F.9 Object tracking on HiEve videos (informative anchor) 44](#_Toc133523742)

[Appendix G Feature dump functionality 45](#_Toc133523743)

[Appendix H. Local Feature anchor package addendum 48](#_Toc133523744)

[Appendix I. Cloud Feature anchor package addendum 49](#_Toc133523745)

# Introduction

In 2019 MPEG started an investigation into the area of video coding for machines. The focus of this exploration was to study the case where images and videos are compressed not to be looked at and evaluated by humans, but rather by machine vision algorithms. These algorithms can serve different purposes such as object detection, instance segmentation, or object tracking. As video compression standards such as HEVC or VVC are developed and optimized towards the human visual system, the existing standards may not be optimal for applications where the video is analyzed by machines. One aspect is the compression of intermediate features seen in a neural network.

Regarding feature compression, a formal call for evidence was issued in July 2022 and provided evidence that this can be achieved in different ways. This call for proposals is the start of a process which has the creation of a new international standard as its goal.

This work on “Feature Compression for Video Coding for Machines” (FCVCM) aims at compressing features for machine tasks. As networks increase in complexity architectures such as ‘Collaborative Intelligence’ (whereby a network is distributed across an edge device and the cloud) become advantageous. With the rise of newer network architectures being deployed amongst a heterogenous population of edge devices, such architectures bring flexibility to systems implementers. As a consequence of such architectures, there is a need to efficiently compress intermediate feature information for transport over wide area networks (WANs). As feature information differs substantially from conventional image or video data, coding technologies and solutions could be different from conventional ones in order to achieve optimized performance for machine usage. With the rise of machine learning technologies and machine vision applications, the amount of video and images consumed by machines has been rapidly growing. Typical use cases include intelligent transportation, smart city, intelligent content management, etc., which incorporate machine vision tasks such as object detection, instance segmentation, and object tracking. Due to the large volume of video data, it is essential to extract and compress the features from video for efficient transmission and storage. Feature compression technology solicited in this CfP can also be helpful in some other regards, such as computational offloading and privacy protection. This call focuses on the compression of features and thus responses are expected to produce decoded features that will be used to complete execution of a pre-defined set of machine vision algorithm to generate the performance results.

Over the last three years, MPEG has investigated potential technologies for efficient compression of feature data for machine vision tasks and established an evaluation mechanism that includes feature anchors, rate-distortion based metrics, and evaluation pipelines. Feature anchors developed by FCVCM utilize both image and video datasets.

This call requires proponents to submit technology suitable for compressing features. **Note that it is mandatory to provide results for machine vision tasks of object detection and object tracking for video datasets and object detection and instance segmentation for an image dataset.**

This document contains detailed information about the setup of this call, general rules for conduct, the planned timeline for both this call and the development of a standard, detailed information on what needs to be submitted, and the next steps following the evaluation of responses to the call.

# Who may participate

Proponents that respond to this call may include any persons whether they are or are not accredited delegates of ISO/IEC JTC 1/SC 29/WG 2. However, all proponents are required to attend the meetings at which their proposals are evaluated. The meeting during which proposals are evaluated is identified with an \* in Table 1 and Table 2. A one-time invitation may be extended to proponents to participate in the evaluation process if the proponent is not an accredited delegate of ISO/IEC JTC 1/SC 29/WG 2. If the proponent’s technology is accepted into the Working Draft of the Standard, then the proponents are required to participate in meetings identified with a † in Table 1 and Table 2. In such a case where the technology is accepted from a proponent who is not an accredited delegate of ISO/IEC JTC 1/SC 29/WG 2, the proponent is expected to initiate the process to join their National Body committees in order to become accredited to participate in subsequent meetings of WG 2. Information for how to join National Body committees and to become an accredited delegate for ISO/IEC JTC 1/SC 29/WG 2 is available at [How to Get Involved](https://www.iso.org/get-involved.html).

# Code of conduct and rules of engagement

All participants shall be required to familiarize themselves with relevant [ISO Policies and Procedures](https://www.iso.org/resources.html), including in particular [ISO Code of Conduct](https://www.iso.org/publication/PUB100397.html), [ISO Declaration for Participants in ISO Activities](https://www.iso.org/declaration-for-participants-in-iso-activities.html), [ISO Privacy and Copyright](https://www.iso.org/privacy-and-copyright.html) policy, and [ISO Policy on Communication of Committee Work](https://www.iso.org/publication/PUB100382.html), and to consent to be bound by these policies.

# Source code and IPR

By responding to a CfP, the proponent affirms that he or she is willing to make source code available for use as the starting point for collaborative standardization.

It is the responsibility of the proponent to obtain any necessary internal approvals in a timely manner, otherwise more readily available source code may be selected.

Furthermore, proponents are advised that this Call is being made subject to the common patent policy of ITU-T/ITU-R/ISO/IEC (refer to [www.itu.int/ITU-T/dbase/patent/patent-policy.html](http://www.itu.int/ITU-T/dbase/patent/patent-policy.html) or Appendix I of [ISO/IEC Directives Part 1](http://isotc.iso.org/livelink/livelink?func=ll&objId=4230455&objAction=browse&sort=subtype)).

# Testing Fee

Participating in this Call for Proposals is not associated with any fees.

# Definitions

The definitions for terms associated with this Call for Proposals can be found in section 1 of [1].

Furthermore, this Call for Proposals uses the terms Working Draft (WD), Committee Draft (CD), Draft International Standard (DIS), Final Draft International Standard (FDIS) and International Standard (IS) according to the [ISO Stages and Resources for Standard Development](https://www.iso.org/stages-and-resources-for-standards-development.html).

# Documents of CfP package

The CfP package consists of the following documents:

* N190 Requirements and Use cases for Video Coding for Machines
  + This document describes the use cases where proposals can be applied and what requirements the final standard needs to fulfill.
* N282 Call for Proposals on Feature Compression for Video Coding for Machines (this document) and an excel template for reporting the test results attached to this document.
  + This document contains details about the submission process and the guidelines to follow. Furthermore, it describes who may participate and the rules for participation.

# Submission Process

## CfP Timeline table

Each entry in the table is described in a section below. WG 2 is the SC 29 working group WG 2 MPEG Technical Requirements.

Unless stated otherwise, deadlines refer to a specific day at **23:59 UTC (no grace period will be granted)**.

Table . CfP Timeline (\* indicates attendance at the meeting is required).

|  |  |  |  |
| --- | --- | --- | --- |
| **Meeting** | **Date** | **Who** | **Action** |
| 11 | April 2023  (24 - 28) | WG 2 | Issue Call for Proposals package. |
|  | 2023-05-02 | WG 2 | Verified version of test material is available (prospective proponents should begin their own feature anchor generation feature anchor runtimes, e.g., reproduced locally or in the cloud). |
|  | 2023-07-03 | Proponent | Registration deadline. |
|  | 2023-07-04 | Call Administrator | Inform Test Administrator of registrations. |
|  | 2023-07-11 | Test Administrator | Perform cross-check assignment. |
| 12 | July 2023 (17 - 21) | Co-chairs | Intermediate report on CfP progress issued. |
|  | 2023-09-13 | Proponent | Uploaded bitstream files, results and decoder/NN part 2 package for each network/split point. |
|  | 2023-09-13 | Test Administrator | Inform proponents which (anonymized ID) proposals they are cross checking. |
|  | 2023-09-14 | Volunteers and Proponents | Begin cross-checking others' results. |
|  | 2023-10-05 | Volunteers and Proponents | Completed cross-checking others results and communicated cross-check result to Test Administrator. |
|  | 2023-10-09 | Proponent | Submitted proponent documentation as a contribution to the 13th WG 2 meeting. |
|  | 2023-10-09 | Test Administrator | Submitted anonymized1 results of cross-checking as a contribution to the 13th WG 2 meeting. |
|  | 2023-10-13 | AHG | AHG prepare draft report on CfP responses. Topics: Requirements met, performance, new project initiation warranted, technology submitted. |
| 13\* | October 2023 (16 - 20) | WG 2 | Evaluate Call for Proposals submissions and select technology. Respondents must be present at the meeting and present the proposals for information sharing.  The source code of candidate technologies for the reference model will be published. |
|  | 2023-10-20 | Software coordinators | Source code of the selected technologies published on Git. |
|  | 2023-11-20 | AHG | Release reference model (RM) on Git with cross-checked performance results. |
|  | 2023-11-20 | AHG | Release working draft (WD). |

Notes:

1. The report provided to WG 2 identifies proponents only by their Proponent ID and not their company name.

## Envisioned Timeline for the FCVCM Standard

It is envisioned that the timetable for the progress of the FCVCM standard will be as follows. Note that not all exact dates for future meetings are set and AhG meetings might be in advance of the hosting WG meetings.

Table . Standardization Timeline (\* indicates attendance at the meeting is required. † indicates attendance is required if technology is selected to be included in the Working Draft and further stages.)

|  |  |  |
| --- | --- | --- |
| **WG 2 Meeting** | **Date** | **Action** |
| 11 | 2023-04-24/28 | CfP |
| 12\* | 2023-07-17/21 |  |
| 13\* | 2023-10-16/20 | Evaluation, RM, WD |
| 14† | 2024-01-01 |  |
| 15† | 2024-04-01 |  |
| 16† | 2024-07-01 |  |
| 17† | 2024-10-01 | CD |
| 18† | 2025-01-01 | DIS |
| 19† | 2025-04-01 | Verification Test |
| 20† | 2025-07-01 | FDIS |
| 21† | 2025-10-01 | IS |

## Register your participation

Proponent must register on or before the Registration deadline (see Table 1), an intention to participate in the CfP. Registering an intent is not binding and registered parties are not required to submit proposals. However, parties that do not register will not be able to submit proposals. Each organization shall only register once. Register by sending an email to the people detailed in section 9. This email shall indicate:

* Company/entity name(s) (all affiliations associated with the submission).
* Contact name and contact email address.
* The number of proposals the proponent plans to submit.
* The envisioned scope of the proposal(s) (e.g., will all or only some requirements be met). This envisioned scope is not binding and is not a restriction on proponent response but is rather for planning purposes only.

Proponents shall indicate which training datasets they will use (if applicable) in their response(s) in their registration email. Proponents are advised that training data used in standards development must be accessible by all participants. The use of training data which does not meet this criterion may affect a proponents eligibility to be selected following a CfP.

At the registration deadline the Call Administrator communicates the list of registrants (including any joint proposal registrations) to the Test Administrator.

Upon registration, the proponent also agrees to cross-check up to three (minimum of two) proposals from other proponents for each proposal the proponent is submitting. As an example, if a proponent submits three proposals, they agree to cross-check up to nine proposals from other proponents (in order to help ensure two cross-checks per proposal is met given multiple possible proposals from each organization). The number of cross-checks will depend on the number of submitted proposals and not the planned number of proposals stated in the registration.

After registration, the proponent will receive a number of “ProponentIDs” (<Proponent\_ID>) for use in the submission of bitstreams, decoders and other materials. This permits each proponent to have several submissions, e.g., each tuned to a different operating point. Furthermore, the proponent will receive information on how to access the verified version of the test material.

## Mandatory Equipment, Software and Data Components

The use of the software and data components for regenerating the anchors used in the CfP can be found in Appendix E. Where applicable, the same software and configuration shall be used for the proposed technology as for the anchors, e.g., for converting YCbCr files to PNG FFmpeg shall be used.

There are two computing methodologies: local methodology and cloud-based methodology. A proponent may choose either methodology with the constraint that the proposed technology and the provided anchor results (which show proponent-generated runtimes but must match the reporting template results for rate and task) are evaluated using the same methodology.

## Access to Test Material

Registered proponents can get access to a verified version of the test material by the date on which a verified version of test material is available (see Table 1. The datasets are currently publicly available. A verified collection of the data will be made available together with the feature anchor results. The scripts to generate the feature anchor results for the machine tasks from the datasets will also be included with the test material. Access credentials to obtain these materials are available from the FCVCM co-chairs upon request. The purpose of this verified collection of datasets is to be independent of changes made to the datasets, as MPEG does not own these datasets.

Access credentials to the datasets and inference scripts will also be distributed to other interested parties upon email request.

An overview of the datasets to be used can be found in Appendix B.

Information on how to access the datasets are as follows:

The following files and data are provided for proponents to assist with local feature anchor generation and CfP response simulations:

* Video/image datasets with accompanying ground truth (ground truth may be included with the corresponding feature anchor package)
* Feature anchor packages
* Feature anchor dump packages
* Input document template

These are available to all registered proponents at the following location:

* ftp://[fcvcm@mpeg.org](mailto:fcvcm@mpeg.org)

The password for this account is available from the FCVCM co-chairs.

See also Appendices F and H for additional information to facilitate locally regenerating feature anchors.

## Conduct Objective Evaluations

Proponents must provide the objective measurement of the following parameters. Details on how to measure the following metrics are given in Appendix C.

The main evaluation metrics are:

* machine vision task performance (mAP or MOTA), and
* rate (bitrate or BPP).

The following metrics shall be reported to provide additional information about the proposal and may be used in selecting the technology for the reference model:

* FCVCM decoder runtimes1
* decoder executable size2
* FCVCM encoder runtimes1
* encoder executable size2

Note 1: Includes feature conversion stage (if any, e.g., added trained layers or other processing) and codec stage (if any, e.g., VTM). Where a stage is not applicable to the response, runtime cells for that stage shall be populated with ‘0’ values.

Note 2: For the purpose of reporting executable size, third-party libraries, e.g., libraries installed from public ‘pip’ or ‘conda’ repositories are excluded and supporting data files, e.g., trained network weights, are included. Scripts and executables such as VTM are included.

A template for the submission of this information is given in the attached excel. The proponent shall include a summary of the objective results in the contribution document describing the proposed technology. This shall include a description about the hardware and software environment where the objective evaluation is performed.

The performance for the machine tasks shall be evaluated on the same split points in the neural network as the corresponding feature anchor.

The performance for the machine vision tasks shall be evaluated using the neural networks and split points listed as follows:

* Instance Segmentation:
  + Mask R-CNN [2] X101-FPN at the P-layer split point (part of the Detectron2 framework) at the P-layer split point (for OpenImages, see Appendices E.2 and F.2).
* Object tracking:
  + JDE-1088x608 at layer indices 36, 61, and 74 (the “DarkNet-53” split point) (for TVD videos, see Appendices E.3 and F.1) and alternative split point at layers indices 75, 90, and 105 (for HiEve videos, see Appendices E.3 and F.5).
* Object detection:
  + Faster-RCNN X101-FPN (part of the Detectron2 framework) at the P-layer split point, i.e., the same split point as used for instance segmentation, (for SFU see Appendices E.4 and F.4 and for OpenImages see Appendices E.2 and F.3).

## Details of the Submission

Proponents must submit proposal materials no later than the submission deadline (see Table 1). The materials shall be submitted via FTP with a site URL and username/password communicated by the Test Administrator to all registered proponents.

Figure 1 shows the pipeline of the encoder and the submitted decoder.

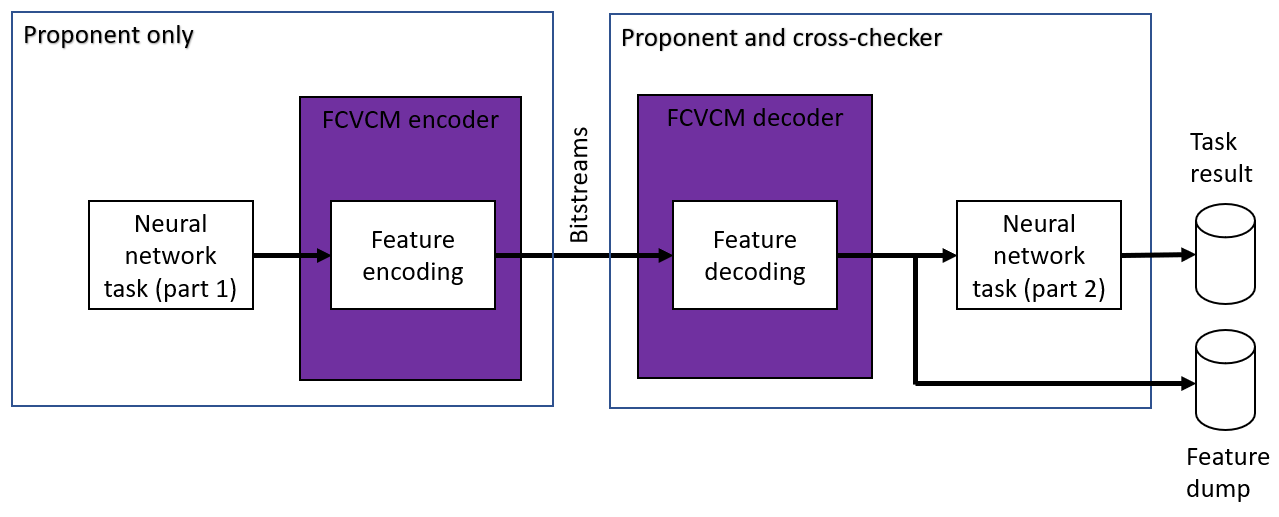


Figure 1. Pipeline of proposal (encoder and submitted decoder).

The provided decoder is expected to include the neural network task (part 2) as part of the package. The provided decoder will report summary-level data on the reconstructed features output from the FCVCM decoder stage (per-channel feature map mean and variance) in the form of a ‘feature dump’. See Appendix G for details.

The randomly assigned proponent IDs will be delivered to each proponent by the Test Administrator, sufficient to permit separate proponent submissions. The assignment of the IDs is confidential so that it is not possible to determine the source of a submission (i.e., name of the proponent) except for the Test Administrator and the proponent of that submission. A proponent or a cross-checker must not share any information that may lead to a disclosure of its own or another proponent’s ID, or information that may identify its own or another proponent’s submission otherwise (e.g., information about the method itself) before the submission of proponent documentation.

Proponents shall produce their own feature anchor results with runtimes captured. The anchor and response runtimes shall be obtained from CPU execution with ‘AVX512’ ISA extensions disabled and on the same platform to enable encoder time (ET)/decoder time (DT) ratios to be reported (see Appendix H and Appendix I). Proponents shall report the platform used.

Proponents shall not manually select nor hardcode the encoder and decoder algorithm or parts of the encoder and/or decoder algorithm according to the testing dataset, test data (input image or video/feature) or machine tasks.

Post-processing is only allowed if it is part of the decoding process.

No images or videos from the FCVCM validation datasets shall be used in training. Where a subset of the frames of a video is used for validation, it is not allowed to use other frames of the video for training.

Proponents shall submit proposal materials for all bitrate points on the image dataset and all video datasets. One bitstream file is to be provided for each performance point for each image in the image dataset and each video performance point in each video dataset.

Results shall be provided for the following datasets performing the indicated tasks

* OpenImages (instance segmentation),
* OpenImages (object detection),
* TVD (object tracking),
* HiEve (object tracking), and
* SFU-HW (object detection).

The proponent is to provide a ‘decoder package’ that executes the FCVCM decoder and the remainder of the neural network, i.e., NN Part 2, with NN Part 2 corresponding to what was used in the respective feature anchor.

Evaluation scripts:

* Task performance: The resulting output shall be capable of being evaluated for task performance using evaluation scripts provided with the corresponding feature anchor. Changes to the scripts are allowed but shall be described in detail.
* Bitrate/runtime: Proponents may provide their own scripts to extract this information (cross-checkers to verify script operation).

Feature data passing through the boundary between the FCVCM decoder and NN Part 2 shall be logged using the ‘feature dump’ approach as exemplified in the feature anchor package for each dataset and as described in Appendix G.

A diagram describing the expected pipeline of proposals can be found in Figure 1c, 1d and 1e of [1]. Note that for the input of the FCVCM encoder shall be feature maps rather than image or video, and so are the output of the FCVCM decoder.

The submission shall contain:

* One archive file per dataset containing compressed bitstreams and decoder package for Linux implementing the FCVCM decoder and NN part 2 and evaluation.
  + For image datasets, one bitstream per image per PP.
  + For video datasets, one bitstream per video per PP.
* One archive file per dataset containing feature dump log files.
* The reporting template with all mandatory rate and task performance information included.
* A short description of how to set up the decoding environment from a clean Linux (e.g., Ubuntu) and scripts to run the decoder to decode the bitstreams in the dataset archives, perform the remainder of the network (NN part 2) and output rate and task performance results.

No identity information should be in any of the submitted files. The submitted files should be marked with the proponent ID in the format of “P< Proponent\_ID>” to help the Test Administrator to identify the source of each file.

### Submission details for datasets

The response shall provide results for six “task performance points” for each image dataset and for each class of each video dataset. For each task performance point for each dataset, the achieved task performance shall lie within a range defined by the boundaries and at the granularity as shown in Table 3.

Table . Acceptance range for task performance point.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset – task | Minimum boundary | Maximum boundary | Granularity |
| OpenImages – object detection | PP – 3 | PP + 2 | Dataset |
| OpenImages – instance segmentation | PP – 3 | PP + 2 | Dataset |
| SFU – object detection | PP – 3 | PP + 2 | Class |
| TVD – object tracking | PP – 3 | PP + 2 | Class |
| HiEve – object tracking | PP – 3 | PP + 2 | Class |

Figure 2 shows an example of the acceptance ranges for each performance point for TVD object tracking. Note that for the BD-rate computation to produce a result, the proposal results must be monotonic.



Figure 2. Example of acceptance ranges for TVD object tracking performance points.

Feature anchor packages include scripts capable of synthesizing class-level results from sequence results.

These results will be used to enable BD-rate measurement.

The response must provide results that lie within the acceptance range for each respective performance point.

The performance point (task result) targets are defined in Table 4, Table 5, and Table 6.

Table  Task results on OpenImages and TVD

|  |  |  |  |
| --- | --- | --- | --- |
| Task performance point (PP) | OpenImages | | TVD |
| Instance segmentation: mAP [%] | Object detection: mAP [%] | Object tracking: MOTA [%] |
| OVERALL |
| PP0 | 80.682 | 78.871 | 49.84 |
| PP1 | 79.187 | 78.056 | 47.11 |
| PP2 | 77.049 | 77.051 | 43.18 |
| PP3 | 72.618 | 74.755 | 35.76 |
| PP4 | 64.247 | 69.653 | 29.47 |
| PP5 | 50.865 | 59.724 | 22.81 |

Table 5 Object tracking results on HiEve

|  |  |  |
| --- | --- | --- |
| HiEve | | |
| Object tracking: MOTA [%] | | |
|  | HIEVE-1080P | HIEVE-720P |
| PP0 | 31.12 | 35.75 |
| PP1 | 30.39 | 34.90 |
| PP2 | 29.29 | 33.66 |
| PP3 | 27.42 | 31.70 |
| PP4 | 24.61 | 29.74 |
| PP5 | 22.84 | 25.20 |

Table 6 Object detection results on SFU-HW

|  |  |  |  |
| --- | --- | --- | --- |
| SFU-HW | | | |
| Object Detection: MAP [%] | | | |
|  | Class A/B | Class C | Class D |
| PP0 | 45.215 | 45.035 | 39.895 |
| PP1 | 44.756 | 41.704 | 38.532 |
| PP2 | 41.160 | 39.138 | 35.869 |
| PP3 | 34.541 | 30.884 | 32.467 |
| PP4 | 28.119 | 17.354 | 23.480 |
| PP5 | 16.852 | 10.017 | 10.669 |

The video sequences shall be coded in a random-access configuration, meaning no more than 32 frames of structural delay, e.g., 32 pictures “group of pictures (GOP)”, and random-access intervals (e.g., IRAP picture) not less frequently than the value corresponding to the multiple of 32 that results in close to 1 second of frames at the sequence frame rate. For example, 64 pictures (or less) for a video sequence with a frame rate of 50 or 60 frames per second and 32 pictures (or less) for a video sequence with a frame rate of 24 or 30 frames per second.

### Submission timeline

To summarize the information regarding the submission, at the following dates the following information needs to be submitted:

|  |  |  |
| --- | --- | --- |
| **Date** | **What needs to be submitted** | **Where** |
| 2023-09-13 | Bitstreams for all performance points for TVD, OpenImages, SFU-HW, HiEve.  Decoder package (FCVCM decoder coupled to NN Part 2).  Excel template with results.  Feature dump files. | FTP server |
| 2023-10-05 | Cross-check results. | Email to the Test Administrator |
| 2023-10-09 | Input contribution describing technology, including:   * excel template with results and runtimes, * questionnaire on requirements, and * training details (if applicable). | Document system (MDMS) |

## Cross-checking of results

After the submission of bitstreams and decoders, each proponent shall cross-check the results reported by up to three and at least two other proponents (based on the cross-check assignment). Non-proponent participants might assist with cross-checking. A report on the results of this cross-checking activity shall be made available by the cross-checkers by the date listed in the CfP Timeline table above. This report shall be uploaded as an input contribution to the 13th WG 2 meeting.

At the time of registration, the Call Administrator communicates the companies responding and the number of responses to the Test Administrator.

The Test Administrator assigns and communicates the cross-checking tasks to the cross-checkers (i.e., proponents). The Test Administrator shall not assign a cross-check of a proposal to a related entity (i.e., separate entities that have registered a joint proposal).

A cross-checker shall not disclose information about the IDs of their cross-checking assignments or information about the methods in their cross-checking assignments until the cross-checking contribution to the MPEG has been submitted.

The cross-check for a proposal is considered successful if all of the following are met:

* Bitrates and task results produced by the cross-checker differ by no more than the amounts specified in Table 7.
* The volume of feature dump data produced by the cross-check run matches the volume provided by the proponent.
* Each task result falls within a lower and upper bound relative to the respective task performance point (see section 8.7.1).

Proposals not passing cross-check will not be further considered at the CfP evaluation.

When a relative amount is specified for a given result (i.e., BPP, bitrate, mAP, or MOTA), the cross-check result rc and proponent result rp must satisfy:

When an absolute amount is specified, the cross-check result rc and proponent result rp must satisfy:

These checks are implemented in the accompanying result template and a summary of these checks is presented in the ‘CrosscheckSummary’ worksheet. Note that although the spreadsheet implements checks for BD-rate, these are not part of the cross-check result.

Table . Cross-check acceptance thresholds.

|  |  |
| --- | --- |
|  | **Acceptance criteria** |
| **BPP/bitrate comparison** | Within or equal to 0.1% (relative) |
| **Task comparison** | Within or equal to 1.5% (relative) OR 0.1 (absolute difference) |

The cross-check shall also report the worst-case discrepancy found in the comparing feature dump log files provided by the proponent with those produced by the cross-checker to an accuracy of one digit after the decimal point (see Appendices G and H). Mismatch in the volume of logged data results in cross-check failure.

Non-proponents that are interested in participating in the cross-checking effort are welcome to contact the Call and Test Administrators and indicate the number of proposals they would like to cross-check.

## Submit Proponent Documentation

Proponents shall submit the following in a contribution to the MPEG meeting indicated in the CfP Timeline table above:

* A written description of the technology having sufficient detail to permit technical discussions.
* Objective test results, as indicated in section 8.6.
* Description of how the requirements in [1] are met using the questionnaire in Appendix A.
* Description of the training details (see Appendix D.2) if the proposal contains components that are learned.
* The encoder and decoder runtimes (except for object tracking on the HiEve dataset, which runtimes are optional), description of the computing platform, encoder size and decoder size.
* An archive of JSON file(s) including the dumped intermediate features, as indicated in Appendix G.

A template for reporting the training details can be found in Appendix D.

Proponents that are WG 2 members shall register and upload an input contribution to the WG 2 meeting and send title and author information to co-chairs of the group as indicated in section 9 prior to the contribution upload deadline. The contribution shall contain all documentation mentioned above.

However, proponents that are not WG 2 members shall email the documents to the convenor of WG 2 two weeks prior to the 13th WG 2 meeting, so that the documents can be registered and uploaded. The documents should be written in Microsoft Word, a template for input contributions can be found at the FTP site where the test data is available (see section 8.5). The Convenor of WG 2 will extend a one-time-only invitation to the WG 2 meeting so that a non-member proponent can present their contributions and participate in the selection process.

All proponents are urged to become members of the WG that will develop a potential FCVCM standard (see Section 2).

## Evaluate CfP Submissions and Select Technology

At the WG 2 meeting indicated in the CfP Timeline table above, submissions will be evaluated by the WG 2 experts to determine whether the CfP has received sufficiently performant proposal(s) in order to transfer the activity to a technical WG. It is strongly urged that proponents have experts familiar with the proposed technology attend in order to allow discussions on details of the proposals. It is envisioned that at least one submission will be selected as technology for the Working Draft of the FCVCM standard. Submissions shall be evaluated considering all submitted information.

The metrics for evaluating the submissions can be found in Section 8.6.

Proposals do not have to fulfill all optional requirements. Requirements that are not fulfilled by the selected technology will be addressed in the Core Experiment (CE) process, which may include CEs using other submitted technologies in order to address all requirements.

If by the assessment of the hosting technical WG experts there is no single best proposal, then the technical WG to which this activity is transferred will draft a workplan on how to merge the best-performing technologies into a single unified technology.

## Submit WD Specification and RM Source Code

This section provides a recommendation to the WG to which FCVCM is transferred on how to proceed.

At the host technical WG meeting indicated in the timetable above, the AHG shall submit an initial version for the Working Draft (WD) of the specification as an input contribution. Based on this contribution, the group will collaborate to create a WD.

The WD must include a normative specification of the FCVCM decoding process, signaling aspects and bitstream syntax.

The source code of the selected technologies shall be published in Git within 4 weeks after the 12th WG 2 meeting where the selected technology is announced. If the proponent(s) fail to publish the source code on time, new technologies will be selected, and a new work plan will be drafted by the hosting technical WG experts. Software coordinators shall be selected from proponent(s) of the selected technology and/or interested experts. If necessary, the merge activity shall be carried out as an open process (i.e., visible to all members of hosting technical WG) on the Git repository. The merging of the code shall be completed before the date indicated in the timeline and a contribution by the software coordinator shall describe the merging process, details of how to access the code, and how to jointly develop the code during the standardization process.

The RM shall be cross-checked by at least two independent technical WG member organizations. If a single proposal was chosen, the cross-checking shall verify that

* the bitrate and task performance meet the same thresholds for acceptance as stipulated in this document for CfP response cross-checks, and
* the reconstructed feature generated by the decoder in the RM match the reconstructed feature generated by the compiled decoder in the original CfP submission.

If the RM is not based on a single proposal, i.e., the RM is a combination of proposals, the proponents shall produce a new anchor for future work on the RM for the collaborative development of the FCVCM standard. The new anchor shall be cross-checked by at least two independent technical WG member organizations.

In addition, the contribution containing the WD shall include a description of the encoding and decoding algorithms. To match this, the RM must include source code that implements the described encoding and decoding algorithms. Note that this does not need to be the exact algorithm used in the proponent’s submission to the CfP but rather the merged software. Subsequent Core Experiment work for the collaborative development of FCVCM standard shall use the RM as the “baseline” (i.e., the new anchor) in CE performance comparisons*,* against which CE technology is evaluated.

# Call Administrator

This Call for Proposals is issued by WG 2. The WG 2 convenor serves as the Call Administrator:

Igor Curcio

Convenor, WG 2 MPEG Technical Requirements

[igor.curcio@nokia.com](mailto:igor.curcio@nokia.com)

For any questions related to this Call for Proposals or associated evaluation procedures please contact the co-chairs of the FCVCM AhG:

|  |  |
| --- | --- |
| Chris Rosewarne  Canon  [Chris.Rosewarne@canon.com.au](mailto:Chris.Rosewarne@canon.com.au) | Yuan Zhang  China Telecom  [zhangy666@chinatelecom.cn](mailto:zhangy666@chinatelecom.cn) |

# Test Administrator

The cross-checking activities for proposals are coordinated by Professor Zhibo Chen ([chenzhibo@ustc.edu.cn](mailto:chenzhibo@ustc.edu.cn))

# Email reflector

For communication, usage of the reflector is encouraged: [mpeg-vcm@lists.aau.at](mailto:mpeg-vcm@lists.aau.at)

You can subscribe etc. to the FCVCM reflector on the following webpage: <https://lists.aau.at/mailman/listinfo/mpeg-vcm>

Important information such as scheduling for AhG, BoG or WG 2 meetings may be shared via this reflector.

WG 2 is using the following reflector: [mpeg-req@lists.aau.at](mailto:mpeg-req@lists.aau.at)

You can subscribe etc. to the WG 2 reflector on the following webpage: <https://lists.aau.at/mailman/listinfo/mpeg-req>

# References

1. N190 Requirements and Use cases for Video Coding for Machines
2. Y. Wu, A. Kirillov, F. Massa, et al. "Detectron2,", https://github.com/facebookresearch/detectron2
3. Z. Wang, L. Zheng, Y. Liu, et al. "Towards real-time multi-object tracking," in European Conference on Computer Vision (ECCV). 2020: 107-122.

# Copyright Header for FCVCM Reference Software

All FCVCM Reference Software files shall contain the following header:

/\* The copyright in this software is being made available under the BSD

\* License, included below. This software may be subject to other third party

\* and contributor rights, including patent rights, and no such rights are

\* granted under this license.

\*

\* Copyright (c) 2023, ISO/IEC

\* All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions are met:

\*

\* \* Redistributions of source code must retain the above copyright notice,

\* this list of conditions and the following disclaimer.

\* \* Redistributions in binary form must reproduce the above copyright notice,

\* this list of conditions and the following disclaimer in the documentation

\* and/or other materials provided with the distribution.

\* \* Neither the name of the ISO/IEC nor the names of its contributors may

\* be used to endorse or promote products derived from this software without

\* specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

\* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

\* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS

\* BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

\* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

\* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

\* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

\* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

\* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF

\* THE POSSIBILITY OF SUCH DAMAGE.

\*/

# Appendix A: Questionnaire

Proponents are requested to copy the following questionnaire to their submission containing the description of the proposed technology and fill it out according to the following instructions:

The requirements for a FCVCM standard are defined in the Use cases and requirements document [1], with those applicable to feature compression listed below. Please use the check boxes in the “Fulfillment” column to indicate which requirements your proposal fulfills and add a short reasoning in the “Reasoning” column as to why your proposal fulfills the requirement.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Fulfillment** | **Reasoning** |
| b) FCVCM shall support feature coding. |  |  |
| d) FCVCM shall support a broad spectrum of encoding rates. |  |  |
| e) FCVCM shall support various degrees of delay configuration. |  |  |
| f) FCVCM shall be agnostic to network models. |  |  |
| g) FCVCM shall be agnostic to machine task types. |  |  |
| h) FCVCM shall provide description of the meaning or the recommended way of using the decoded data. |  |  |
| i) FCVCM should support the use and inclusion of information such as descriptors in its bitstream. |  |  |
| j) A single FCVCM bitstream shall support any number of instances of machine tasks. |  |  |
| m) FCVCM complexity shall allow for feasible implementation within the constraints of the available technology at the expected time of usage. |  |  |
| q) FCVCM shall support privacy and security. |  |  |

Appendix B: Detailed description of test datasets

B.1 Dataset 1: Tencent Video Dataset (TVD)

The Tencent Video Dataset (TVD) consists of 3 video sequences for object tracking. The three video sequences, TVD-01, TVD-02 and TVD-03 are used for the CfP for video coding for machines. TVD-01, TVD-02 and TVD-03 have 3000, 636 and 2334 frames respectively, and the resolutions of the videos are 1920x1080. The files are in MP4 format. The dataset is provided and labeled by Tencent.

Detailed information can be found on <https://multimedia.tencent.com/resources/tvd>.

B.2 Dataset 2: OpenImages v6

OpenImages V6 is a large-scale dataset, consists of 9 million training images, 41,620 validation samples, and 125,456 test samples. Note that all images are already compressed.

In this CfP, a subset of the OpenImages dataset is used. A total of 5000 images were selected for object detection and another 5000 images were selected for instance segmentation. While there is an overlap between the two subsets, these are not identical.

More information on the dataset can be found on <https://storage.googleapis.com/openimages/web/index.html>.

For the machine vision task performance, mAP@0.5 shall be used.

The dataset is available using the following license text:

*The annotations are licensed by Google LLC under the* [*CC BY 4.0*](https://creativecommons.org/licenses/by/4.0/) *license. The images are listed as having a* [*CC BY 2.0*](https://creativecommons.org/licenses/by/2.0/) *license.* ***Note:*** *while we tried to identify images that are licensed under a Creative Commons Attribution license, we make no representations or warranties regarding the license status of each image and you should verify the license for each image yourself.*

B.3 Dataset 3: SFU-HW-Object-v1

SFU-HW-Object-v1 is a labeled video data with object labeled on raw video sequences. It has already been used for MPEG (HEVC). It can be used for compression and object detection simultaneously. The dataset is provided under the Creative Commons license BY 4.0 (CC BY 4.0). Videos and labels of this dataset can be found in the following links:

Video (source for FCVCM activity): [ftp://mpeg.org](ftp://mpeg.org/) (contact Feature Compression for Video Coding for Machines chairs for login information)

Video (original source): <ftp://hevc@mpeg.tnt.uni-hannover.de/testsequences/>

Label: <https://dx.doi.org/10.25314/7d8efc0a-3943-4738-b7a5-72badb04d765>

Clips (specific coded frame range) of each video are used, as follows:

**Class A:** Size 2560x1600p 30 fps

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Original size, framerate** | **Duration** | **Used frames** | **Cropped area position** |
| Traffic | 2560x1600p 30 fps | 5s | 117 to 149 | Line 80,  Column 1200 |

**Class B:** Size 1920x1080p 24-60 fps

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **fps** | **Duration** | **Used frames** |
| Kimono | 24 | 10s | 207 to 239 |
| ParkScene | 24 | 10s | 207 to 239 |
| Cactus | 50 | 10s | 403 to 499 |
| BasketballDrive | 50 | 10s | 471 to 599 |
| BQTerrace | 60 | 10s | 403 to 499 |

**Class C:** Size 832x480p (WVGA) 30-60 fps

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **fps** | **Duration** | **Used frames** |
| BasketballDrill | 50 | 10s | 471 to 599 |
| BQMall | 60 | 10s | 403 to 499 |
| PartyScene | 50 | 10s | 403 to 499 |
| RaceHorses | 30 | 10s | 235 to 299 |

**Class D:** Size 416x240p (WQVGA) 30-60 fps

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **fps** | **Duration** | **Used frames** |
| BasketballPass | 50 | 10s | 471 to 599 |
| BQSquare | 60 | 10s | 403 to 499 |
| BlowingBubbles | 50 | 10s | 403 to 499 |
| RaceHorses | 30 | 10s | 235 to 299 |

Note that the videos “FourPeople”, “Johnny”, and “KristenAndSara”, known as ‘Class E’ are not used in this CfP.

B.4 Dataset 4: HiEve videos

“Human In Events” (“HiEve”) is a labelled video dataset with labels for object tracking, post estimation, and action recognition. This CfP uses the HiEve dataset for object tracking.

The HiEve video anchor package includes the video files in MP4 format, which is the input format for the HiEve feature anchor package. The five videos used from the HiEve dataset are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sequence number** | **Descriptive name** | **Resolution** | **Frame count** |
| 2 | Human in lab1 | 1280×720 | 4819 |
| 13 | hm\_in\_playground | 1920×1080 | 1416 |
| 16 | hm\_in\_road | 1920×1080 | 700 |
| 17 | hm\_in\_square2 | 1280×720 | 966 |
| 18 | hm\_in\_stair2 | 1280×720 | 1614 |

The feature anchor divides the five videos into two classes based on resolution as follows:

* HIEVE\_1080P: 13, 16.
* HIEVE\_720P: 2, 17, 18.

All frames of each video are to be coded and the frame rate is set as 30fps regardless of the indicated frame rate in the provided MP4/MOV file.

The HiEve feature anchor package includes scripts to compute classwise results for the HiEve informative video anchor using provided detections files in the HiEve video anchor package.

The HiEve dataset (videos and ground truth) are available from the FTP server.

**NOTE**: Use of the HiEve videos requires returning a signed copy of a license agreement to Shanghai Jiao Tong University (SJTU), a copy of which is included with the test material.

Appendix C: Anchor metrics

## C.1 Bitrate measurement

For each image dataset, bits per pixel (BPP) shall be used. BPP is the number of bits occupied by each pixel, which is defined by:

refers to the total number of bits overall images and refers to the total number of pixels overall images at their original resolution.

For each video sequence, the bitrate shall be measured in kilobits per second (kbps). This is defined as:

Here refers to the total number of bits of the whole video sequence, *fps* denotes the number of frames per second of the video sequence and *frames* denote the number of encoded frames of the video sequence.

For classwise bitrate reporting, the sequence-length weighted average bitrate of the calculated sequence bitrates is to be reported. This is defined as:

Here *lengthCLS* refers to the total length of sequneces in the class in seconds, *bitrateCLS* refers to the classwise bitrate, *bitraten* refers to the bitrate of sequence *n* in class CLS containing *N* sequences, *FramesToBeEncodedn* refers to the count of frames to be encoded for sequence *n* and *FrameRaten* refers to the fame rate of sequence *n*.

## C.2 Task: Object Tracking

For the object tracking task, Multiple Object Tracking Accuracy (MOTA) [1] shall be used to measure performance.

The MOTA accounts for all object configuration errors made by the tracker, false positives, misses (true negative), mismatches, and overall frames.

where , , and are the number of false negatives, the number of false positives, the number of mismatch error (ID Switching between 2 successive frames), and the number of objects in the ground truth respectively at time .

## C.3 Task: Instance segmentation / Object detection

For both object detection and instance segmentation, mean Average Precision (mAP) [2] [3] shall be used to measure the performance of the network.

For a given category of object, true positive , false positive , false negative , and true negative are defined with an Intersection over Union (IoU) threshold for that category, where true/false represents the output of the neural network, positive/negative represents the label in the ground truth.

Then, recall of the given IoU threshold is defined as the proportion of all true positive examples in all true positive and false negative examples corresponding to that IoU threshold:

The precision of the given IoU threshold is the proportion of all true positive examples which are from all positive examples:

A neural network of segmentation may achieve several pairs of recall and precision values corresponding to a certain IoU threshold and different confidence levels. For each recall value in the pairs, let takes the maximum precision value in all precision values for which the corresponding recall values are above the given recall value :

Average Precision (AP) of a given category of object is defined as the average value of for all recall values provided by the neural network, which can characterize the area of the entire precision-recall curve.

Mean Average Precision (mAP) is an averaged AP over all categories of objects and in a range of IoU thresholds. As an example, in MS COCO 2017 dataset, 10 IoU thresholds are taken at equal intervals from 0.50 to 0.95. In particular, AP50 and AP75 generally present the mAP when the IoU threshold is 0.50 and 0.75 respectively.

The following variants of mAP is used:

* [mAP@0.5](mailto:mAP@0.5): the mAP when the IoU threshold is 0.5

## C.4 Runtime Measurement

Runtime includes Encoding time (EncT), Decoding time (DecT) and Task time for part 1 and part 2 of the network (TaskT1, TaskT2) for complexity measurement. The proposed runtime measurements for a FCVCM solution are:

* **TaskT1**: Time needed to perform part 1 of the network (e.g., the backbone) to produce features.
* **EncT:** Time needed to convert feature input to bitstream.
* **DecT:** Time needed to convert bitstream to decoded features.
* **TaskT2**: Time needed to perform part 2 of the network (e.g., the head) to complete the task based on the decoded features.

For the purpose of reporting encoding and decoding running times, the feature anchor and proposal should be simulated on the same platform, e.g., the same CPU, to have reliable runtime comparison.

For summary runtime reporting, an “overall” result averaging the classwise runtimes from each dataset excluding the HiEve dataset is computed.

Appendix D: Inference and Training information

## D.1 Inference information

The information described below is required to be provided for the inference process for both encoding and decoding processes.

* **Network Visualization:** Graphical representation of the neural network
* **Param. Number**: Total numbers of parameters in the neural network.
* **Param. Precision**: Bits for storing one parameter. Additionally, use “I” for indicating an integer parameter and use “F” to indicate a floating-point number. For example, if the proposed method uses 16-bit integer to represent a parameter, you can report this information as “16 (I)”.
* **MAC (Kilo):** Number of multiply–accumulate (MAC) operations per pixel in the worst case for the inference stage, where the multiply–accumulate operation is a common step that computes the product of two numbers and adds that product to an accumulator. Since different size of input may influence the value, it is suggested to use 3840x2160 as the input frame size for unification.
* **Mem.T (MB):** Temporary memory. It denotes the memory used to store the output feature map for all intermediate layers (forward pass). Since different size of input may influence the value, it is suggested to use 3840x2160 as the input frame size for unification. For reporting Mem.T (MB) the calculation process is also suggested to be provided for crosschecking.
* **Patch Size**: The size of input to the neural networks during inference (patchW×patchH×patchT, e.g., 64x64x3) where applicable (e.g., when patch-wise processing of features is performed).

## D.2 Training information

When applicable, it is required to report and discuss the following information for the training process.

* **Epoch**: The number of complete passes through the training data (e.g., 100)
* **Batch Size**: The number of samples processed before the model is updated. (e.g., 4Kx16frames)
* **Training Time**: CPU and/or GPU (e.g., 48h) and hardware such as CPU/GPU model and count (if different to that used for inferencing).
* **Learning Curve:** Plot of the training loss and validation loss (or similar) versus the number of epochs
* **Training Sets**: If a pre-trained model is used, the source of the pre-trained model and its training sets should be reported in detail. The size (number of images or videos) used in each training dataset shall be reported.
* **Training Configuration per Rate-Distortion Point**: Any changes in the requested information used to generate different rate-distortion points

The following additional training information could help to better understand proposed neural network-based methods:

* **Number of Iterations:** number of gradient updates within an epoch
* **Patch Size**: size of input to the neural networks (patchW×patchH×patchT, e.g., 64x64x3) where applicable (e.g., when patch-wise training using features is performed).
* **Learning Rate**: The amount that the weights are updated during training (e.g., 5e-4)
* **Optimizer**: The algorithm used to change the attributes of proposed neural networks (e.g., ADAM)
* **Loss Function**: The function to calculate the model error during training and optimization (e.g., L1, L2, etc.)
* **Preprocessing**: (e.g., preprocessing procedure, normalization, cropping method, rotation, zoom etc.)

Appendix E Feature Anchor generation procedure

## E.1 Feature anchor generation

Each feature anchor is generated by running a neural network in two parts, with additional operations for quantization and frame packing included in these two parts. VTM is used to encode and decode the packed frames into bitstreams.

For each feature anchor a package is provided that is capable of producing the feature anchor result. For feature anchors for video datasets, the packed frames are stored as a YUV video sequence and coded using Random access (RA) configuration. Scripts for RA encoding divide the encoding into a set of segments, each of which may be run in parallel. Each segment encodes frames corresponding to one ‘intra period’ worth of pictures plus one training intra picture.

Feature anchors for the OpenImages dataset encode each image as a separate one-frame YUV file and use All intra (AI) configuration

**Stage 1: NN part 1**

The neural network is run in a mode where layers up to the specified split point (which may include multiple specific layers in the network) and feature data at this point is captured. Layers not required for production of feature data at the split point are not run (e.g., by early termination of the run).

Captured feature data is quantized from floating-point domain into 10b samples using the worst-case quantization range encountered in the dataset at this split point.

Quantized feature data is packed into YUV400 feature videos with that feature maps are packed in channel order.

**Stage 2: VTM encoding/decoding**

The default configuration files are provided with the VTM software and should be used for the anchor of FCVCM. There are two default test configurations provided as follows:

* **All Intra (AI)** used for image dataset: encoder\_intra\_vtm.cfg
* **Random Access (RA)** used for video datasets: encoder\_randomaccess\_vtm.cfg

The following defines the parameters to be changed for each test point are:

* **InputFile** to reflect the location of the source image/video sequence on the test system
* **FrameRate** to reflect the frame rate of a given image/video sequence
* **SourceWidth** to reflect the width of the feature image/video
* **SourceHeight** to reflect the height of the feature image/video
* **FramesToBeEncoded** to reflect the frame count of a given image/video sequence
* **IntraPeriod** to reflect the intra refresh period in the random-access configuration. The intra refresh period is dependent on the frame rate of the source and the GOP size in use: a value 32 shall be used for sequences with a frame rate equal to 20fps, 24fps, 25fps and 30fps, 64 for 50fps, and 60fps, and 96 for 100fps.
* **QP** to reflect the quantization parameter value
* **InputBitDepth** to reflect the bit depth of a feature video, i.e., 10 bits.
* **(optional) ConformanceWindowMode**=1 to use automatic padding mode
* **InternalBitDepth**=10 to use 10-bit internal bit-depth

Table 2 shows the configurations that need to be indicated for each test dataset. The other parameters need to be defined according to the specific test sequence.

Table 2. The configurations of the anchor for each dataset

|  |  |  |  |
| --- | --- | --- | --- |
| **Machine Task** | **Configuration** | **QP** | |
| Object detection on OpenImages | AI | 35, 37, 39, 41, 43, 45 | |
| Instance segmentation on OpenImages | AI | 35, 37, 39, 41, 43, 45 | |
| Object tracking on TVD | RA | TVD-01 | 22, 24, 26, 29, 31, 33 |
| TVD-02 | 23, 25, 27, 28, 30, 31 |
| TVD-03 | 25, 26, 27, 29, 30, 31 |
| Object Detection on SFU | RA | Traffic | 37, 39, 41, 43, 45, 47 |
| Kimono | 41, 42, 43, 44, 45, 46 |
| ParkScene | 32, 35, 38, 41, 44, 47 |
| Cactus | 40, 42, 44, 48, 49, 51 |
| BasketballDrive | 32, 35, 38, 41, 44, 47 |
| BQTerrace | 32, 35, 38, 41, 44, 47 |
| BasketballDrill | 32, 35, 38, 41, 44, 47 |
| BQMall | 37, 39, 41, 43, 45, 46 |
| PartyScene | 37, 40, 43, 46, 49, 51 |
| RaceHorsesC | 39, 41, 43, 46, 49, 52 |
| BasketballPass | 37, 39, 41, 43, 45, 46 |
| BQSquare | 32, 35, 38, 41, 44, 47 |
| BlowingBubbles | 40, 42, 44, 46, 49, 52 |
| RaceHorses | 39, 41, 43, 46, 49, 52 |
| Object tracking on HiEve | RA | 2 | 22, 25, 27, 29, 31, 34 |
| 13 | 20, 22, 24, 26, 28, 29 |
| 16 | 22, 24, 26, 28, 30, 31 |
| 17 | 22, 23, 24, 26, 27, 28 |
| 18 | 22, 25, 27, 29, 31, 34 |

Due to potential high frame counts of videos, feature anchors support parallel encoding of RA for videos.

Where parallel encoding is performed, the feature anchor includes a script to assemble a sequence-level bitstream from each of the bitstreams resulting from parallel execution.

VTM is used to decode the sequence-level bitstreams to produce sequence-level decoded feature videos.

**Stage 3: NN Part 2**

The neural network is run in a mode whereby layers corresponding to ‘NN part 2’ are skipped. Decoded feature videos are read, unpacked, and inverse quantized to produce features for insertion as the output of the layers corresponding to the specified split point. Then, the remainder of the network is run as normal to complete the inferencing operation.

## E.2 Anchor generation for OpenImages object detection and instance segmentation tasks (feature)

NOTE: The FTP site provides two sets of 5000 images, one for object detection and one for instance segmentation. The OpenImages feature anchor package accesses a single directory for both tasks. Both archives may be extracted into the same directory and where overlapping filenames are encountered, the two files are identical.

The evaluation pipeline of the experiments is shown in the following figure:



The outputs of P-layers in Faster R-CNN X101-FPN and Mask R-CNN X101-FPN networks for object detection and instance segmentation, respectively, are used as compression target, and the compression and decompression are performed via VTM 12.0 reference software. Specifically, the P-layer feature maps are extracted from the front part of Faster R-CNN and Mask R-CNN networks, denoted as part A in the above figure. To feed these P-layer outputs into the VTM reference software, the P-layer outputs of different sizes are tiled into a single frame and then uniformly quantized. In the decoder side, corresponding dequantization and unpacking operations are performed, and then the reconstructed P-layer feature maps are fed into the rest parts of the machine task network, denoted as part B in the above figure, for measuring mean average precision ([mAP@0.5](mailto:mAP@0.5)).

Both for object detection and instance segmentation the quantization range used is:

[-26.426828384399414, 28.397470474243164]

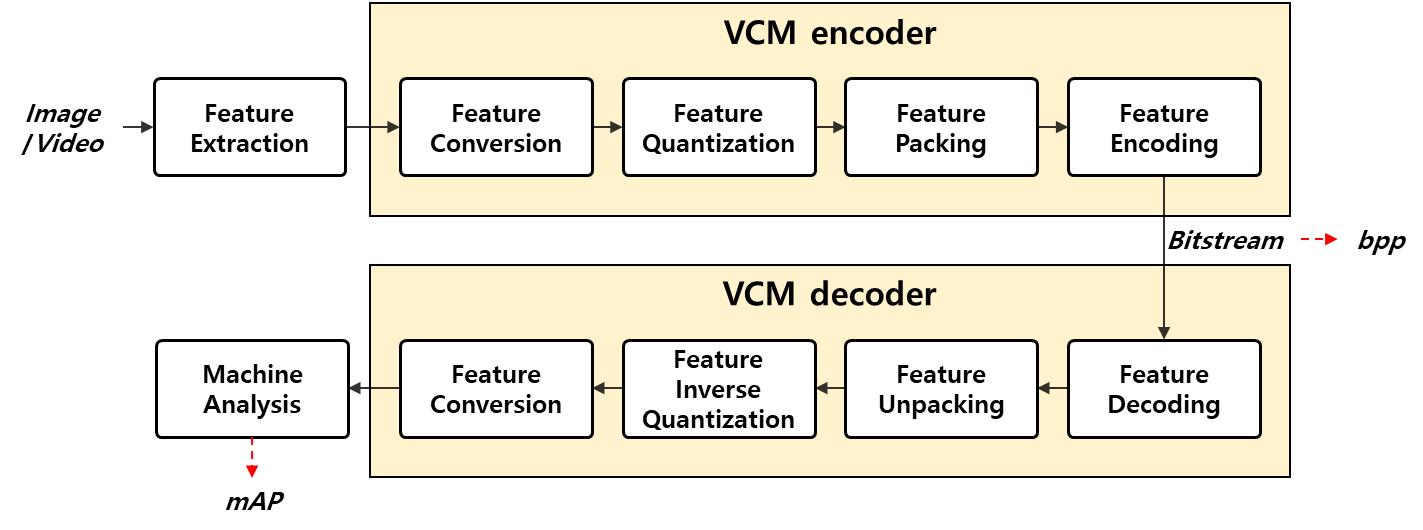
The setup for this anchor is as follows:

TVD feature anchor setup is as follows:

|  |  |
| --- | --- |
| Machine Tasks | Object detection and instance segmentation |
| Codec | VTM-12.0 |
| Configuration | All Intra (one bitstream per image per QP) |
| Dataset | OpenImagesV6 validation set – two subsets of 5K images are used for each task |
| Bitstream Cost | BPP |
| Task Network | FasterRCNN-X101-FPN for object detection and MaskRCNN-X101-FPN for instance segmentation (both from Detectron2 framework) |
| Network split point | P2-P5 |
| Performance Metric | mAP |

## E.3 Anchor generation for object tracking task: TVD videos and HiEve videos (feature)

The following figure shows the pipeline of the anchor generation for object tracking task on the TVD (Video) and HiEve dataset. The original format of video is mp4.



For the TVD feature anchor, the split point was chosen as the boundary between ‘Darknet-53’ backbone and the remainder of the YOLOv3 network implemented within the JDE framework. This split point provides an arguably well-accepted definition for ‘backbone’ for this network. The tensor dimensionality and resulting frame area at this point is shown in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Width × height** | **Channel count** | **Luma samples** |
| 0 | 136×76 | 256 | 2646016 |
| 1 | 68×38 | 512 | 1323008 |
| 2 | 34×19 | 1024 | 661504 |
| Total | | | 4630528 |

For the HiEve feature anchor, the split point was selected as defined at layers 75, 90, and 105 in the Towards-Realtime-MOT (“JDE”) framework. The tensor dimensionality and resulting frame area at this point is shown in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Width × height** | **Channel count** | **Luma samples** |
| 0 | 136×76 | 128 | 1323008 |
| 1 | 68×38 | 256 | 661504 |
| 2 | 34×19 | 512 | 330752 |
| Total | | | 2315264 |

For both anchors using JDE, feature maps were packed in a raster-scan arrangement into YUV400 videos with a bit-depth of 10 bits, with increasing channel order.

Quantization range for the TVD videos (Darknet-53 split point) is: [-4.722218990325928, 48.58344268798828].

TVD feature anchor setup is as follows:

|  |  |
| --- | --- |
| Machine Task | Object Tracking |
| Codec | VTM-12.0 |
| Configuration | RandomAccess (RA) - GOP32 with IntraPeriod = 64 |
| Dataset | TVD dataset |
| Bitstream Cost | Bitrate |
| Task Network | JDE-1088x608 [3] |
| Network split point | Darknet-53 |
| Performance Metric | MOTA |

Quantization range for the HiEve videos (alternative split point) is: [-1.0795, 11.8232].

HiEve feature anchor setup is as follows:

|  |  |
| --- | --- |
| Machine Task | Object tracking |
| Codec | VTM-12.0 |
| Configuration | RandomAccess (RA) (GOP32) |
| Dataset | HiEve dataset |
| Bitstream Cost | Bitrate |
| Task Network | JDE-1088x608 |
| Network split point | L0-2 at points 75, 90, and 105 |
| Performance Metric | MOTA |

The HiEve videos are all treated as 30fps, which results in an IntraPeriod setting of 32.

The provided feature anchor packages use 'pymotmetrics’ to produce MOTA scores. Note that the ‘munkres’ LAP solver is used when producing HiEve results.

## E.4 Anchor generation for SFU-HW videos for object detection (feature)

The pipeline for the SFU-HW video object detection feature anchor is as follows (note: basic quantization and packing serves as ‘FCVCM encoder’ and ‘FCVCM decoder’ in the feature anchor):

FCVCM encoder

FCVCM decoder

Neural Network Task (part 2)

Video

Bit stream

Reconstructed Feature

Feature Conversion

Neural Network Task (part 1)

Video Decoding

Inverse Conversion

Video Encoding

Table 8 shows P-layer tensor dimensionality for BasketballDrill, BasketballDrill, BasketballPass, BlowingBubbles, BQMall, BQSquare, PartyScene, RaceHorsesC, and RaceHorses (D).

Table 8. Feature map area at split point.

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Width × height** | **Channel count** | **Luma samples** |
| P2 | 336×200 | 256 | 17203200 |
| P3 | 168×100 | 256 | 4300800 |
| P4 | 84×50 | 256 | 1075200 |
| P5 | 42×25 | 256 | 268800 |
| Total | | | 22848000 |

The packed frame size is 5376×4256

Table 9 shows P-layer tensor dimensionality for BasketballDrive, BQTerrace, Kimono, and ParkScene.

Table 9. Feature map area at split point.

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Width × height** | **Channel count** | **Luma samples** |
| P2 | 336×192 | 256 | 16515072 |
| P3 | 168×96 | 256 | 4128768 |
| P4 | 84×48 | 256 | 1032192 |
| P5 | 42×24 | 256 | 258048 |
| Total | | | 21934080 |

The packed frame size is 5376×4080

Table 10 shows P-layer tensor dimensionality for Traffic.

Table 10. Feature map area at split point.

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Width × height** | **Channel count** | **Luma samples** |
| P2 | 320×200 | 256 | 16384000 |
| P3 | 160×100 | 256 | 4096000 |
| P4 | 80×50 | 256 | 1024000 |
| P5 | 40×25 | 256 | 256000 |
| Total | | | 21760000 |

The packed frame size is 5120×4256.

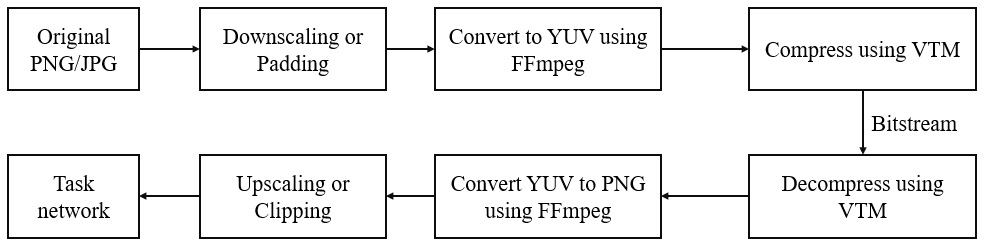
The feature anchor setup is as follows:

|  |  |
| --- | --- |
| Machine Task | Object detection |
| Codec | VTM-12.0 |
| Configuration | RandomAccess (RA) (GOP32), Intra period set based on frame rate |
| Dataset | SFU-HW dataset |
| Bitstream Cost | Bitrate |
| Task Network | FasterRCNN X101-FPN |
| Network split point | P-layer (P2-P5) |
| Performance Metric | mAP@0.5 |

The quantization range applied for SFU-HW sequences is [-17.8848, 16.69417].

## E.5 Informative anchor generation for instance segmentation task (Image)

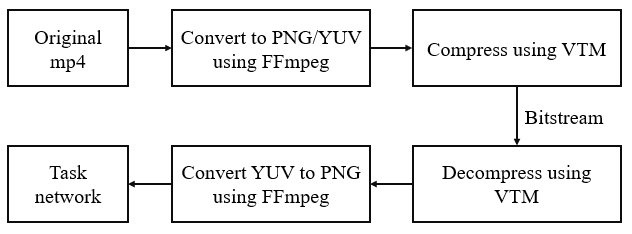
* The following figure shows the pipeline of the anchor generation for instance segmentation task:



The original images with PNG format should be first downscaled or padded and converted to YUV420 format using FFmpeg. The generated images in YUV format are compressed and decompressed using VTM software with the configuration predefined. The reconstructed YUV files with bit depth of 10 are then converted and upscaled or clipped to the same format and resolution of the input for the machine tasks.

## E.6 Informative anchor generation for object tracking task: TVD (Video)

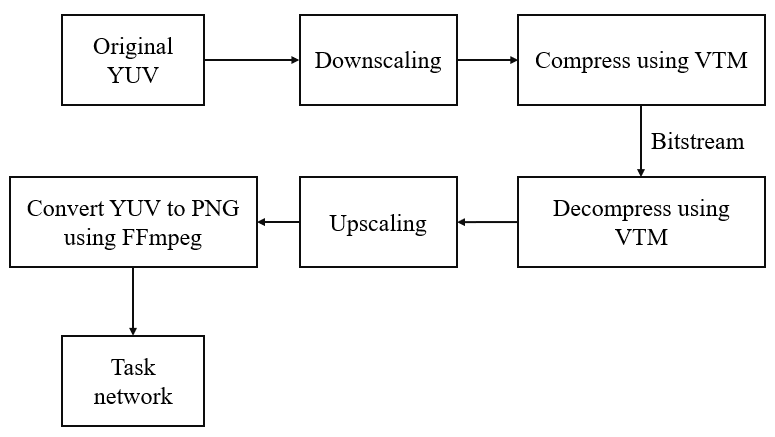
The following figure shows the pipeline of the anchor generation for object tracking task on the TVD (Video) dataset. The original format of video is mp4.



The source video sequence in mp4 format should be first converted to PNG and YUV format using FFmpeg, and then compressed and decompressed using VTM software with the configuration predefined. The reconstructed YUV files should be converted to PNG and then fed into the task object tracking networks.

## E.7 Informative anchor generation for video object detection: SFU-HW-Objects-v1

The following figure shows the pipeline of the anchor generation for video object detection task on the SFU-HW-Objects-v1 dataset. The input format of video is YUV420p. The reconstructed video should be converted to PNG format before running the task network.



The source video sequence in yuv420 format should be first downscaled using FFmpeg, and then compressed and decompressed using VTM software with the configuration predefined. The generated YUV files with bit depth of 10 are then converted and upscaled to the same format and resolution of the input using FFmpeg for the machine tasks.

## E.7 Informative anchor generation for video object tracking: HiEve

Informative (video) anchors for HiEve provide results at a classwise granularity.

The HiEve feature anchor package on the FTP site includes scripts to produce classwise informative anchor results from sequence results from the HiEve video anchor.

Classwise bitrate is computed as a sequence-length-weighted average of the sequence bitrates for the sequenes in the class.

Classwise MOTA scores are computed using the ‘OVERALL’ result produced by ‘pymotmetrics’ when processing results from each sequence in the class. Note that the ‘munkres’ LAP solver is used when producing HiEve results.

Appendix F Anchor generation environment

## F.1 Object tracking on TVD videos (feature anchor)

Software package versions used to generate the object tracking feature anchor are as follows:

* CentOS 7.9.2009
* gcc-9.3.1 (via ‘devtoolset-9’)
* CUDA 11.7
* Nvidia driver 515.86.01
* Python 3.8.13 (via ‘rh-python38’)
* Numpy 1.22.4
* PyTorch 1.13.1+cu117
* Pandas 1.1.5
* PIL 8.2.0
* GNU 'parallel' utility (for parallel VTM, inferencing execution)
* ffmpeg 4.2.2
* VTM-12.0 software (extracted and compiled under the `build` directory as 'VTM-12.0' subdirectory)

## F.2 Instance segmentation on OpenImages (feature anchor)

Software package versions used to generate the instance segmentation feature anchor are as follows:

* Xeon Gold6244 CPU @ 3.6GHz(8Core) x 2
* Ubuntu 18.04.5 LTS
* Nvidia Driver version: 460.91.03
* CUDA 11.2
* Python 3.6.9
* Torch 1.8.0
* Tensorflow 2.6.0
* Detectron2 0.4
* VTM: 12.0

Detectron2 and Tensorflow Object Detection API

* Install detectron2 following instructions from the webpage <https://github.com/facebookresearch/detectron2/releases/tag/v0.4>
* Install Tensorflow Object Detection API by following the webpage: <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html#tensorflow-object-detection-api-installation>

## F.3 Object detection on OpenImages (feature anchor)

Software package versions used to generate the object detection feature anchor are as follows:

* Xeon Gold6244 CPU @ 3.6GHz(8Core) x 2
* Ubuntu 18.04.5 LTS
* Nvidia Driver version: 460.91.03
* CUDA 11.2
* Python 3.6.9
* Torch 1.8.0
* Tensorflow 2.6.0
* Detectron2 0.4
* VTM: 12.0

## F.4 Object detection on SFU videos (feature anchor)

Software package versions used to generate the object tracking feature anchor are as follows:

* CentOS 7.9.2009
* gcc-9.3.1 (via ‘devtoolset-9’)
* CUDA 11.7
* Nvidia driver 515.86.01
* Python 3.8.13 (via ‘rh-python38’)
* Numpy 1.22.4
* PyTorch 1.13.1+cu117
* Pandas 1.5.3
* PIL 9.4.0
* GNU 'parallel' utility (for parallel VTM, inferencing execution)
* ffmpeg 4.2.2
* VTM-12.0 software (extracted and compiled under the `build` directory as 'VTM-12.0' subdirectory)

## F.5 Object tracking on HiEve videos (feature anchor)

Software package versions used to generate the object tracking feature anchor are as follows:

* CentOS 7.9.2009
* gcc-9.3.1 (via ‘devtoolset-9’)
* CUDA 11.7
* Nvidia driver 515.86.01
* Python 3.8.13 (via ‘rh-python38’)
* Numpy 1.22.4
* PyTorch 1.13.1+cu117
* Pandas 1.5.3
* PIL 9.4.0
* GNU 'parallel' utility (for parallel VTM, inferencing execution)
* ffmpeg 4.2.2
* VTM-12.0 software (extracted and compiled under the `build` directory as 'VTM-12.0' subdirectory)

## F.6 Object tracking on TVD (informative video anchor)

Software package versions used to generate the object tracking video anchor are as follows:

* Ubuntu 18.04.3 LTS
* Nvidia Driver version: 450.80.02
* CUDA: 11.0
* Python: 3.8.8
* Torch: 1.8.1
* Detectron2: 0.4
* VTM 12.0i
* FFMPEG: 4.2.2

## F.7 Instance segmentation / object detection on OpenImages (informative anchor)

Software package versions used to generate the instance segmentation and object detection anchor are as follows:

* Ubuntu 18.04.4 LTS
* Nvidia Driver version: 440.100 or above
* CUDA: 10.2
* Python: 3.7.4
* Torch: 1.6.0
* Tensorflow: 2.7.0
* Detectron2: 0.2.1
* VTM: 12.0
* pandas: 1.2.4
* opencv-python: 4.5.2.52

Detectron2 and Tensorflow Object Detection API

* Install detectron2 0.2.1 following instructions from the webpage: <https://github.com/facebookresearch/detectron2/releases/tag/v0.2.1>
* Install Tensorflow Object Detection API by following the webpage: <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html#tensorflow-object-detection-api-installation>

## F.8 Object detection on SFU videos (informative anchor)

Software package versions used to generate the object tracking feature anchor are as follows:

* CentOS 7.9.2009
* CUDA 11.3
* Nvidia driver 465.19.01
* Python 3.6.8
* Numpy 1.19.5
* PyTorch 1.7.1
* Pandas 1.1.5
* PIL 8.2.0
* GNU 'parallel' utility (for parallel VTM, inferencing execution)
* ffmpeg 4.2.2
* VTM-12.0 software (extracted and compiled under the `build` directory as 'VTM-12.0' subdirectory)

## F.9 Object tracking on HiEve videos (informative anchor)

* Ubuntu18.04 LTS
* Pytorch1.9.1
* Geforce 3090

Appendix G Feature dump functionality

Figure 3 shows the methodology for verifying a decoder-side package, i.e., a package that includes VCM decoder and NN part 2.

For CfP responses, the encoder-side package is run by the proponent, i.e., including NN part 1 and the VCM encoder. The decoder-side package is run by both the proponent and the cross-checker. As the VCM decoder may include trained layers or other functionality that executes in the same process as NN part 2, it is necessary to verify that this interface between the VCM decoder and NN part 2 exists. Feature dumping provides a way to demonstrate data is passing through this interface without requiring writing and then reading a prohibitive volume of feature data to/from disk.

Feature contents are summarized in JSON files and a ‘compare.py’ script is used to determine the degree to which they match, by reporting the worst-case absolute different encountered amongst the logged data.

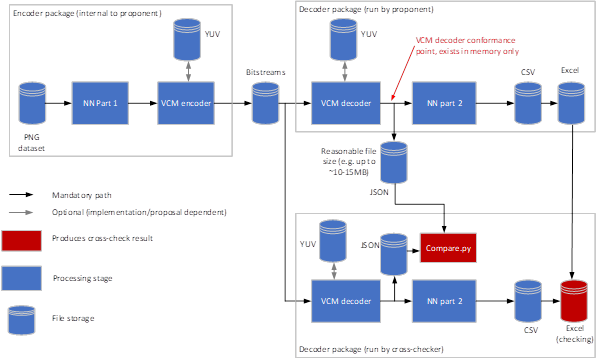


Figure . Methodology for verifying feature dump.

Feature dump JSON file contents are as follows:

A text file is written with one JSON object per line for each video sequence and each image. Each JSON object includes a list of dictionaries, one entry per FPN layer for one frame of the video. Each dictionary includes ‘mean’ and ‘variance’ keys, with values being a list of floating-point values.

A script ‘compare.py’ is included in the anchor package that is capable of comparing two sets of these files and reporting any cases where the absolute difference of the mean or variance differ by more than a specified threshold (0.001 by default).

Logging mean and variable for every channel resulted in an archive containing these files for this feature anchor occupies 128MB once compressed in tar.gz format. This file size just for one anchor seems impractical for file exchange via email or as attachment to a contribution.

Subsampling of the logged channels is performed to prevent the compressed file size of the feature dump JSON files from exceeding approximately 10-15MB (an amount more easily shared e.g. via email or inclusion in an input document). Subsampling is performed at the following rates based on the dataset:

* SFU: Every 9th channel
* HiEve: Every 90th channel
* TVD videos: Every 9th channel.
* OpenImages: Every 9th channel.

An example of the format is shown below (with subsampling to every 9th channel and 256 channel tensors as seen in the FasterRCNN network):

[{“means” : [p2\_chan0\_mean, p2\_chan8\_mean, …, p2\_chan252\_mean],

“variances” : [p2\_chan0\_var, p2\_chan8\_var, …, p2\_chan252\_var]},

[{“means” : [p2\_chan5\_mean, p2\_chan13\_mean, …, p2\_chan248\_mean],

“variances” : [p2\_chan5\_var, p2\_chan13\_var, …, p2\_chan248\_var]},…

Note that the remainder of the offset from subsampling along the channel dimension is carried forward to the next tensor being logged within the file (i.e., within the scope of one video or one image).

This feature dump methodology is exemplified in the feature anchors for each video dataset. See file ‘libvcm/utils.py’ class ‘FeatureDump’ in any video dataset feature anchor package.

The video feature anchors include a script ‘bin/compare.py’ that enables comparing the feature dump files from two separate runs (e.g., from a proponent decoder run and from a cross-checker decoder run). The volume of data is checked and must exactly match and the worst-case absolute discrepancy is reported on a per-file basis. A threshold argument is provided to provide a pass/fail indication of the threshold check (differing data volume directly results in a ‘fail’ result).

The ‘compare.py’ script can also be used for checking feature dumps produced from running the OpenImages feature anchors, however you will have to check each QP separately as the script compares contents of two directories and OpenImages feature anchors place feature dump files into separate directories for each QP.

The ‘compare.py’ script has an argument ‘--threshold’ that may be used to provide an ‘OK’, ‘Not OK’ check against the two sets of feature dump logs being checked.

Appendix H. Local Feature anchor package addendum

This addendum contains extra information to assist when locally generating feature anchor packages. Locally generated feature anchor packages need to match the published feature anchor results within thresholds for cross-check acceptance.

Locally generated feature anchors provide reference runtime results against which a proponent can report encoder/decoder time ratios of their own proposals and any cross-checks they perform.

Note that all feature anchor packages are configured to run inferencing stages using CPU with the following setting applied to prohibit use of AVX512 instruction set extensions:

export DNNL\_MAX\_CPU\_ISA=AVX2

Use of AVX512 was found to affect results obtained from the feature anchor package. As AVX512 is less widely available, it is disabled in the feature anchor packages. If enabled, results are highly likely to exceed thresholds for cross-check acceptance.

**FTP site anchor packages:**

* OpenImages: P\_layer\_Det\_Seg\_Anchor\_distribute\_r1.zip
* HiEve: hieve-objtrk-anchor.tar.gz
* SFU: sfu-objdet-anchor-noavx512.tar.gz
* TVD: tvd-objtrk-anchor.tar.gz

**FTP site feature dump archives:**

* OpenImages: OpenImages\_dumps.zip
* HiEve: hieve\_feature\_dump.tar.gz
* SFU: sfu\_feature\_dump.tar.gz
* TVD: tvd\_feature\_dump.tar.gz

**Network portion 1 (backbone) simulation**

Output from this simulation is stored in a subdirectory under ‘streams’ (see README.md for the directory specific to this feature anchor). If any JSON files in this subdirectory are zero in size, later stages will fail.

If any zero-sized JSON files are found, please check the corresponding ‘stderr’ and ‘stdout’ files under the corresponding subdirectory under the ‘sim’ directory for any indication of the cause of failure.

**VTM parallel chunk-based encode**

VTM is very memory intensive (cases of 10GB/process have been observed). If the OS runs out of memory during a VTM encoding operation, the corresponding stderr file has been observed to contain the text ‘Killed’ along with the process ID and executable name (EncoderAppStatic). Please rerun any killed cases.

If the VTM encoding job succeeds the corresponding stderr file will be empty. It is recommended to check all stderr files are zero-sized to confirm no ‘Killed’ cases are present. An example command to do this is as follows:

find sim/<vtm\_encode\_dir\_name> ‘\*.stderr ‘ -exec ls -s “{} “ \;

Note that even if some VTM encode jobs were killed the resulting truncated chunk bitstreams are still concatenated into sequence-level bitstreams, which may decode to produce a decoded YUV with missing frames. Thus, it is recommended to check stderr files from the VTM encode stage before progressing to subsequent stages.

**Job execution notes**

The feature anchor package is provided to run as a standalone package on one server, with helper scripts ‘seqrun.sh’ and ‘parallel.sh’ to co-ordinate job submission on the server. VTM jobs are single-threaded and hence may be run in parallel. Jobs involving Pytorch (NN part 1 and 2) are multithreaded and are run sequentially.

The jobs may also be run on a grid submission platform (e.g., SGE). If doing so, to ensure correct operation please ensure all relevant environment variables (see env\_anchor.sh) are passed to the jobs. Please also ensure stdout and stderr are captured to files named and placed as they would be were the helper scripts used (i.e., same directory as the script, different extension).

**Feature dump log files**

Due to the large size of the feature dump archives they are provided as separate files for download from the FTP site rather than combined with the feature anchor package itself.

Appendix I. Cloud Feature anchor package addendum

This addendum contains extra information to assist when generating feature anchor packages in a public cloud. Feature anchor packages generated in the cloud need to match the published feature anchor results within thresholds for cross-check acceptance.

The main difference between generating the anchor locally or in the cloud is in how the software and hardware is configured. Public clouds provide various configurations on the software and hardware. For example, in Amazon Elastic Compute Cloud (EC2), a user can select an Amazon Machine Image (AMI) as an initial configuration of the software. For hardware configuration, a user selects a (virtual server) instance. EC2 then installs the AMI to the selected hardware. After that, a user can install additional software listed in Appendix F to complete the software configuration.

To facilitate crosscheck, proponents are encouraged to submit a containerized image of their evaluation setup, e.g., in form of a custom AMI or Dockerfile.

If using cloud-based methodology, it is highly encouraged to use the option whereby dedicated machine resources (‘reserved instance’ in Amazon terminology), i.e., not shared VM (‘spot instance’), are used so that runtime measurements are reliable. The type of instance shall be reported.

Note: Proponents and cross-checkers are free to use this methodology but cannot mandate others to use it (for example, a proponent using this methodology cannot mandate that their cross-checker also use the cloud-based methodology).