**Project Specification: VCM-ST-NPP**

**🌍 Project Title**

**Spatio-Temporal Neural Preprocessing for Standard-Compatible Video Coding in Machine Vision Tasks**

**🌐 Goal**

Build a complete pipeline that integrates a spatio-temporal neural preprocessing module with a standard video codec (HEVC/VVC) for optimized performance on downstream machine vision tasks (object detection, semantic segmentation, and multi-object tracking), while keeping the output bitstream fully compatible with standard codecs.

**🔢 System Overview**

Raw Video Input

⬇

ST-NPP Module (Temporal + Spatial + QAL)

⬇

Compressed via Standard Codec (e.g., HEVC)

⬇

Decoded via Standard Decoder

⬇

Input to Task Model (Detection / Segmentation / Tracking)

**🔧 Component Specification**

**✏️ models/st\_npp.py**

* **Function:** Define the Spatio-Temporal Neural Preprocessing Module
* **Classes/Functions:**
  + class STNPP(nn.Module)
    - forward(x, qp)
      * Apply spatial branch (Conv2D)
      * Apply temporal branch (Conv3D)
      * Fuse with skip connections
      * Use QAL (see below)
      * Output: enhanced frame x\_hat

**🔧 models/qal.py**

* **Function:** Quantization Adaptation Layer
* **Classes/Functions:**
  + class QAL(nn.Module)
    - forward(qp)
      * Input: scalar QP
      * Output: channel-wise scaling vector
      * Uses MLP with ReLU activations

**🔧 models/proxy\_codec.py**

* **Function:** Learnable differentiable proxy that mimics standard codec
* **Classes/Functions:**
  + ProxyEncoder
  + ProxyDecoder
  + ProxyCompressionModel
  + forward(x\_hat)
    - Outputs: recon\_frame, bitrate

**🔧 models/combined\_model.py**

* **Function:** Combine ST-NPP + QAL + Proxy Codec + Task Head
* **Classes/Functions:**
  + class CombinedModel(nn.Module)
    - forward(x, qp)
      * Output: compressed frame + task outputs

**🔧 models/task\_networks/detector.py**

* **Function:** Lightweight CNN detector head (dummy task head)
* **Classes/Functions:**
  + class DummyDetector(nn.Module)
    - forward(x)
      * Output: class logits or bounding boxes

**📊 utils/**

**model\_utils.py**

* Save/load model checkpoints
* save\_model(model, path)
* load\_model(path)

**video\_utils.py**

* Frame extraction, batching, reconstruction
* extract\_frames(video\_path)
* reconstruct\_video(frames)

**codec\_utils.py**

* Interface with FFmpeg
* Check available encoders
* Compress video with selected codec (x264/x265)
* compress\_with\_ffmpeg()

**data\_utils.py (TO CREATE)**

* Dataloader for COCO / KITTI / MOTChallenge
* get\_dataloader(dataset\_name)
* Handle batching, augmentation, and formatting

**loss\_utils.py (TO CREATE)**

* Compute total loss
* get\_loss(task\_outputs, labels, bitrate, pre\_loss)
  + task\_loss + λ bitrate + β pixel\_loss

**metric\_utils.py (TO CREATE)**

* mAP, mIoU, MOTA
* evaluate\_detection()
* evaluate\_segmentation()
* evaluate\_tracking()

**📁 scripts/**

**run\_comprehensive\_evaluation.sh**

* Shell script to evaluate everything end-to-end
* Run: ST-NPP -> Codec -> Task Inference -> Save results

**🔬 train.py**

* Args: --dataset, --epochs, --batch\_size, --lr, --qp
* Loads: CombinedModel
* Uses proxy codec for training
* Applies detection + segmentation losses

**📊 evaluate.py**

* Evaluate task model on compressed video
* Output mAP, mIoU, MOTA at different bitrates (QP levels)

**🌐 Expected Datasets**

* COCO 2017 (Detection, Segmentation)
* KITTI (Object Detection)
* MOT17 (Tracking)

**✨ Final Goals**

* Achieve >30% BD-Rate savings vs. HEVC at equal task accuracy
* Modular framework: replace codec or task head easily
* Visualization tools for preprocessed frames and final output

**🔐 Notes for Cursor AI**

* All models must be modular and torch.nn.Module
* Ensure all inputs/outputs are annotated
* Save sample frames and task outputs after compression
* Structure output folders: checkpoints/, results/, logs/
* Proxy codec only used in training
* At test time, must use standard ffmpeg-based codec

**🔔 Reminder**

Please ensure requirements.txt contains:

* torch, torchvision, opencv-python
* ffmpeg-python
* numpy, matplotlib
* pycocotools
* scipy, scikit-image