Deep-BVQM: A Deep-learning Bitstream-based Video Quality Model

Nasim Jamshidi Avanaki
Quality and Usability Lab
Technische Universität Berlin

May 2022
Outline

- Introduction
- Motivation
- Deep-BVQM
  - Frame level
  - Video level
- Performance Analysis
- Video Quality Prediction Pipeline
- Discussion
- Conclusion
Introduction

- Video Quality
- Video Quality Assessment (VQA)
- Types of VQA methods
  - Subjective quality assessment
  - pseudo-objective models
    - planning models
    - bitstream-based models
    - signal-based models
Motivation

- Bitstream-based models are simple, accurate and good choice for monitoring models.
- Each bitstream model takes a few years to be developed.
- Require collecting massive subjective tests.
- Any codec updates requires development of a new model.
- They predict the quality at minimum 1 second not at frame-level.
P.120X Quality Prediction

"bitrate": 3231.07,
"codec": "h264",
"duration": 7.0 sec,
"fps": 30.0,
Quality variation in gaming video

Frame-level Quality Score

Frame Index

1 sec

30 seconds
Deep-BVQM has been developed at two levels:

- Frame Level
  - Using CNN
- Video Level
  - Using LSTM
Deep-BVQM (frame-level)

- Take 89 x 89 patches from the QP values.
- Train a CNN model
  - Deep model
  - Simple model
- Use VMAF classes for training

- [0–20]  [0–17]
- [20–40]  [23–37]
- [40–60]  [43–57]
- [60–80]  [63–77]
- [80–100] [83–100]
Deep-BVQM (frame-level)(incl. CNN architecture, dataset)

Training Dataset used for frame-level module

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Quantity</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source videos</td>
<td>38 SRC</td>
<td>CGVDS + GVSET</td>
</tr>
<tr>
<td>Rate controller</td>
<td>1</td>
<td>CBR</td>
</tr>
<tr>
<td>Preset</td>
<td>3</td>
<td>Slow, veryfast, Llhq</td>
</tr>
<tr>
<td>Bitrate</td>
<td>9</td>
<td>300 kbps to 10 mbps</td>
</tr>
<tr>
<td>Codec</td>
<td>1</td>
<td>H.264</td>
</tr>
<tr>
<td>Resolution</td>
<td>1</td>
<td>1080p</td>
</tr>
<tr>
<td>Frame rate</td>
<td>1</td>
<td>30, 60 fps</td>
</tr>
</tbody>
</table>

38 x 1 x 3x 9x 1 x 1 x 1 = 1026 video files
≈ 300 k frames (training)

CNN Architecture

<table>
<thead>
<tr>
<th>Layer</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>1x89x89</td>
</tr>
<tr>
<td>Pool</td>
<td>1x44x44</td>
</tr>
<tr>
<td>Conv1</td>
<td>16x44x44</td>
</tr>
<tr>
<td>Pool</td>
<td>16x22x22</td>
</tr>
<tr>
<td>Conv2</td>
<td>32x22x22</td>
</tr>
<tr>
<td>Conv3</td>
<td>64x22x22</td>
</tr>
<tr>
<td>Pool/Dropout (20%)</td>
<td>64x11x11</td>
</tr>
<tr>
<td>Conv4</td>
<td>64x11x11</td>
</tr>
<tr>
<td>Fc1</td>
<td>7744</td>
</tr>
<tr>
<td>Fc2</td>
<td>1000</td>
</tr>
<tr>
<td>Fc3</td>
<td>5</td>
</tr>
</tbody>
</table>
Deep-BVQM (frame-level)

```
<table>
<thead>
<tr>
<th>Frame type</th>
<th>Frame size + seq_img</th>
<th>QP values</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>1024x768/4:2:0:0/4s</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>1280x720/4:2:0:0/4s</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>1920x1080/4:2:0:0/4s</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>2560x1440/4:2:0:0/4s</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>3840x2160/4:2:0:0/4s</td>
<td></td>
</tr>
</tbody>
</table>

QP values

Bitstream Information of Frames

Frame size + Frame type

Fully Connected (FC_i)

Latent features

VMAF Classes

A
B
C
D
E

Concatenate (Frame size, Frame type, preset)

Scheme of the model design at frame level

CNN

12
Using the `ffmpeg_debug_qp` to extract the bitstream information.

- Requires pre-processing to be used for deep learning task.
- Each row conveys the information including Frame type, Frame size, Quantization parameter (Qp) average, and Qp values.
Results on validation set – frame level (1)

Use a Random Forest model to see how well the latent features are trained.

- **Per frame**
  - PLCC = 0.97
  - RMSE = 0.33

- **Per video**
Comparison of predicted VMAF, blue line, vs actual VMAF scores, grey line, over 900 frames.
**Deep-BVQM (video-level)**

**Video quality prediction using LSTM**

[Diagram of LSTM network for video quality prediction]

- **Inputs:** Frame 1, Frame 2, ..., Frame n
- **Predicted Quality:**
Deep-BVQM (video-level)

Properties of LSTM network:

- Input window: 15 frames taken from 1 sec duration
- Each frame vector: 8 features
  [5 Latent features, Frame type, Frame size, Preset]
- Training based on a small dataset of 900 input windows (≈13’500 frame vectors) and validated on unknown datasets to the model.
- Ground truth: MOS values
Deep-BVQM (video-level) (incl. LSTM architecture)

Video quality prediction using LSTM

LSTM Architecture

<table>
<thead>
<tr>
<th>Layer</th>
<th>Output Shape</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>1 x 15 x 8</td>
<td>0</td>
</tr>
<tr>
<td>LSTM₁</td>
<td>(None, 15, 64)</td>
<td>18688</td>
</tr>
<tr>
<td>LSTM₂</td>
<td>(None, 15, 32)</td>
<td>12416</td>
</tr>
<tr>
<td>LSTM₃</td>
<td>(None, 15, 32)</td>
<td>8320</td>
</tr>
<tr>
<td>LSTM₄</td>
<td>(None, 16)</td>
<td>3136</td>
</tr>
<tr>
<td>Dropout</td>
<td>(None, 16)</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>(None, 17)</td>
<td>17</td>
</tr>
</tbody>
</table>

Total trainable params: 42,577
Performance on two gaming datasets – video level

GVSET Dataset

KUGVD Dataset
## Performance of bitstream-based models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>( \text{P1203 m1} )</th>
<th>( \text{P1203 m3} )</th>
<th>( \text{P1204.3} )</th>
<th>BQGV</th>
<th>Proposed Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>GVSET</td>
<td>PLCC 0.67</td>
<td>0.91</td>
<td>0.92</td>
<td>0.68</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>RMSE 1.53</td>
<td>0.86</td>
<td>0.53</td>
<td>0.98</td>
<td>0.36</td>
</tr>
<tr>
<td>KUGVD</td>
<td>PLCC 0.65</td>
<td>0.92</td>
<td>0.96</td>
<td>0.74</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>RMSE 1.45</td>
<td>0.59</td>
<td>0.41</td>
<td>0.87</td>
<td>0.32</td>
</tr>
<tr>
<td>CGVDS</td>
<td>PLCC 0.80</td>
<td>0.86</td>
<td>0.83</td>
<td>0.88</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>RMSE 0.70</td>
<td>0.49</td>
<td>0.52</td>
<td>0.40</td>
<td>0.24</td>
</tr>
</tbody>
</table>

Performance of bitstream-based models on all three gaming dataset, only at 1080p resolution.
Model Prediction

Video Quality MOS: 2.87

Predicted Video Quality: 2.54

Frame Index

VMAF Score

1 sec

30 seconds
Video Quality Prediction Pipeline (Pre-processing, data extraction)

Bitstream Parser

I
QP = [38, ..40]
Fram size = 20249 bits

P
QP = [35, ..34]
Fram size = 2658 bits

P
QP = [37, ..38]
Fram size = 3374 bits

P
QP = [39, ..38]
Fram size = 3501 bits
Video Quality Prediction Pipeline (at frame level)

- **Bitstream Parser**
  - QP = [38, 35, 37, 39]
  - Fram size = 20249, 2658, 3374, 3501 bits

- **Random forest model**
  - vmaf = 91
  - vmaf = 88
  - vmaf = 88
  - vmaf = 87

- **Latent feature extraction**

- **CNN**

- **Frame level quality prediction**
  - Preprocessing, data extraction
Video Quality Prediction Pipeline (in 1-sec duration)

- **Bitstream Parser**
  - Random forest model
  - Latent feature extraction
  - LSTM
  - CNN

**Input Parameters**
- QP = [38, ..40]
- Fram size = 20249 bits
- QP = [35, ..34]
- Fram size = 2658 bits
- QP = [37, ..38]
- Fram size = 3374 bits
- QP = [39, ..38]
- Fram size = 3501 bits

**Quality Assessment**
- vmaf = 91
- vmaf = 88
- vmaf = 88
- vmaf = 87

**Output**
- 1-Second Segment

**Steps**
- Preprocessing, data extraction
- Frame level quality prediction
- 1-second Video Quality Prediction
**Discussion**

**Future Extension**

- Extend the codec type select a constant block size.
- Extend the resolution ranges development of multiple models for different resolutions.
- rescaling the input to $89 \times 89$ that is used for 1080p resolution.
- upscaling 720p $\rightarrow$ PLCC = 0.92, RMSE = 0.49

**Capability to predict non-gaming content**

- Used a dataset with 2160p (4k) resolution and different encoding setting
  $\rightarrow$ PLCC = 0.74, RMSE = 0.67

**Computational Complexity**

- Less than 1 second on GPU
Conclusion

Deep-BVQM is a bitstream-based model which...

- is capable to predict at the frame level.
- is developed for gaming encoding setting.
- simplifies the model extension and expansion.
- is a lightweight prediction method.
Thanks!

Any questions?

You can find me at:

n.jamshidiavanaki@tu-berlin.de