Lightweight Multitask Learning for Robust JND Prediction using Latent Space and Reconstructed Frames

Sanaz Nami¹, Farhad Pakdaman¹, Mahmoud Reza Hashemi¹, Shervin Shirmohammadi¹, and Moncef Gabbouj¹

¹Affiliation not available

January 2, 2024
Lightweight Multitask Learning for Robust JND Prediction using Latent Space and Reconstructed Frames

Sanaz Nami, Student Member, IEEE, Farhad Pakdaman, Member, IEEE, Mahmoud Reza Hashemi, Senior Member, IEEE, Shervin Shirmohammadi, Fellow, IEEE, and Moncef Gabbouj, Fellow, IEEE

Abstract—The Just Noticeable Difference (JND) refers to the smallest distortion in an image or video that can be perceived by Human Visual System (HVS), and is widely used in optimizing image/video compression. However, accurate JND modeling is very challenging due to its content dependence, and the complex nature of the HVS. Recent solutions train deep learning based JND prediction models, mainly based on a Quantization Parameter (QP) value, representing a single JND level, and train separate models to predict each JND level. We point out that a single QP-distance is insufficient to properly train a network with millions of parameters, for a complex content-dependent task. Inspired by recent advances in learned compression and multitask learning, we propose to address this problem by (1) learning to reconstruct the JND-quality frames, jointly with the QP prediction, and (2) jointly learning several JND levels to augment the learning performance. We propose a novel solution where first, an effective feature backbone is trained by learning to reconstruct JND-quality frames from the raw frames. Second, JND prediction models are trained based on features extracted from latent space (i.e., compressed domain), or reconstructed JND-quality frames. Third, a multi-JND model is designed, which jointly learns three JND levels, further reducing the prediction error. Extensive experimental results demonstrate that our multi-JND method outperforms the state-of-the-art and achieves an average JND prediction error of only 1.57 in QP, and 0.72 dB in PSNR. Moreover, the multitask learning approach, and compressed domain prediction facilitate light-weight inference by significantly reducing the complexity and the number of parameters.

Index Terms— Just Noticeable Difference (JND), Human Visual System (HVS), Multitask Learning, Compressed Domain.

I. INTRODUCTION

The fast growth and popularity of new multimedia technologies such as Virtual Reality (VR), Cloud Gaming, and Ultra-High-Definition (UHD) video streaming have led to an exponential increase in video traffic and the required bandwidth. This has intensified the need for efficient video compression techniques that can reduce the storage and bitrate of videos while maintaining their visual quality as much as possible. Perceptual Video Coding (PVC) is a promising approach that aims to further compress video content by eliminating imperceivable redundancies in the content according to the Human Visual System (HVS).

Just Noticeable Difference (JND) is one of the most efficient PVC approaches, which takes advantage of the HVS characteristics in perceiving visual quality in a discrete manner. This means that the HVS can only distinguish between a limited number of quality levels, known as JND levels, as measured by metrics such as Quantization Parameter (QP) [1][2]. As observed in Fig. 1, videos encoded with any QP value between two adjacent JND levels (the jump points in Fig. 1) are perceived with similar qualities.

JND is used in various multimedia applications, including perceptual coding [3][4], quality enhancement [5][6], quality assessment [7][8], face recognition [9], and recently for compression for machine vision [10][11]. By exploiting the limitations of the HVS in detecting small changes in video content, JND-based methods can allow imperceivable distortion levels, resulting in the lowest possible bitrate while maintaining the visual quality. Thus, high-bitrate applications such as Cloud Gaming (CG), 360 video, and UHD streaming, can benefit from JND-based methods. This is achieved by predicting the JND levels, and applying them to optimize video applications, such as video coding algorithms as depicted in Fig. 2.

Existing JND methods can be classified into three categories: pixel domain JND models, frequency domain JND models, and image/video domain JND models. Pixel domain JND models, [12][13][14], calculate a distortion threshold for each pixel by taking into account various key features such as Luminance Adaptation (LA) and Contrast Masking (CM). Frequency domain JND models, [15][16][17], operate on the frequency

Sanaz Nami, Farhad Pakdaman and Moncef Gabbouj are with the Faculty of Information Technology and Communication Sciences, Tampere University, 33720 Tampere, Finland (e-mail: sanaz.nami@tuni.fi; farhad.pakdaman@tuni.fi; moncef.gabbouj@tuni.fi). Shervin Shirmohammadi is with the DISCOVER Laboratory, School of Electrical and Computer Engineering, College of Engineering, University of Ottawa, Ottawa, ON K1N 6N5, Canada (email: shervin.s@uottawa.ca).
domain of the image. The image is first transformed into a frequency domain representation such as Discrete Cosine Transform (DCT) and then a threshold is calculated for each sub-band considering several features such as LA, CM, and Contrast Sensitive Function (CSF). While pixel and frequency domain JND methods showed good performance, they have limitations. (1) HVS perceives the quality of a video frame as a whole rather than individual pixels [2] and (2) these models overlook the impact of quantization during compression [1]. To this end, recent JND models have increasingly utilized Machine Learning (ML) approaches to calculate the JND threshold in terms of encoding parameters, for each image or video scene. Recent JND datasets such as MCL-JCI [18], VVC based JND dataset [19], and KonJND [20] for images, or MCL-JCV [21] and VideoSet [22] for videos, have facilitated such a data driven approach. JND prediction models are trained on such datasets, to predict the JND threshold in terms of Quality Factor (QF) or QP, by receiving the raw and distorted videos as input. While JND-based PVC approaches enhance the compression efficiency and privacy preservation. Moreover, to push the performance beyond each individual model for a single JND level, we jointly train three JND levels, with multitask learning. Not only this approach improves the performance, but also reduces the inference time and the complexity of predicting multiple JND levels, which is crucial for many applications such as live streaming [30]. Project codes and data will be available at https://github.com/sanaznami/MTL_JND.

The limitations described above motivated us to propose a novel solution for JND prediction, that exploits JND-quality frames, and multitask learning for improved performance. We hypothesize that if a model is learned that can translate raw video into its JND-quality video, the learned features should contain efficient HVS-related features that can also predict the JND levels. With this in mind, we propose to jointly learn to reconstruct JND-quality videos from raw videos and to predict JND levels. Moreover, we claim that different JND levels represent correlated information, and jointly learn them in a multitask learning approach which improves the generalization ability and the accuracy of JND prediction by learning a richer feature space. In our recent work [23], we studied multitask learning for JND, and presented early proof of concept for the effectiveness of jointly learning multiple JND levels. This paper significantly extends our previous work and designs a novel solution for JND prediction which is outlined in next.

Inspired by recent works on learned compression [24][25], we design an autoencoder, namely a reconstruction backbone, which learns to convert raw video frames to their JND-quality representations. Then, two types of data modalities from the autoencoder are used for JND prediction, (1) the Latent space obtained from the encoder, and (2) the JND-quality frames reconstructed by the decoder. Each data modality is used for training JND prediction models based on two strategies: (1) learning the JND prediction sequentially, after learning the JND-quality videos reconstruction, and (2) training an End-to-End (E2E) model to learn both tasks simultaneously. We elaborate in the next sections that while the reconstructed data often leads to the best performance, the latent space-based methods are capable of delivering excellent performance, with a much lower computational complexity, hence enabling fast inference at the edge [26][27]. In fact, this is inspired by the upcoming JPEG-AI [28] and MPEG VCM [29] standards, that require supporting compressed domain analysis for energy efficiency and privacy preservation. Moreover, to push the performance beyond each individual model for a single JND level, we jointly train three JND levels, with multitask learning. Not only this approach improves the performance, but also reduces the inference time and the complexity of predicting multiple JND levels, which is crucial for many applications such as live streaming [30]. Project codes and data will be available at https://github.com/sanaznami/MTL_JND.

The main contributions of the paper are as follows:

(1) We propose a solution that reformulates the JND prediction problem into learning the JND-quality video jointly with JND QP prediction. This solution learns the best representative features by regularizing the QP loss with a frame-wise difference loss to guide the network. The learned features highly boost the performance of JND prediction compared to existing methods.

(2) We develop two deep learning-based methods that predict the JND levels by receiving either the decompressed JND-quality video frames, or directly in the compressed (latent space) frames. To the best of our knowledge, this is the first study that predicts JND levels based on latent space, or even
based on JND-quality frames.

(3) We propose to jointly learn multiple JND levels as correlated tasks, in a multitask learning setting. Compared to single-JND learning, this approach reduces the memory footprint, increases the inference speed, and most importantly improves the prediction performance.

(4) Using the latent space prediction, and the multi-JND method, we significantly reduce the computational complexity and number of parameters required for JND prediction. Moreover, the proposed methods only receive the raw frames, without the need for compression of each frame with multiple QPs, as in traditional methods.

(5) Through ablation studies and extensive experiments conducted on two distinct JND datasets, one for videos and one for images, we demonstrate the effectiveness of the proposed methods.

The rest of the paper is organized as follows. Prior works are introduced in Section II. The proposed methods are presented in Section III. The evaluation results are presented and discussed in Section IV, and finally the paper is concluded in Section V.

II. RELATED WORK

As previously stated, JND methods can be categorized into three groups: pixel domain JND models, frequency domain JND models, and image/video domain JND models. The following sections will provide a review of these methods.

A. Pixel domain JND models

Several traditional studies have used signal processing tools, and leveraged the characteristics of the HVS for quality perception such as LA and CM, to calculate a JND threshold value for each pixel. Wu et al. [12] have partitioned the image into regular and irregular contents, taking into account that HVS is less sensitive to irregular contents. To predict the JND, they employed distinct strategies for each content. For regular contents, they determined the JND threshold by considering the effects of LA and CM. For irregular contents, they analyzed the difference between the original image and its prediction obtained through an Autoregressive model. Wu et al. [13] have emphasized that contrast masking cannot accurately account for the complicated interaction among visual contents. For this reason, they considered Pattern Complexity (PC) as another factor to adopt the overall JND threshold. Wang et al. [14] proposed a new pixel domain JND model for screen content images. They considered the LA, CM, and structural distortion sensitivity in each edge profile. Then, they created the JND model based on the edge profile reconstruction with tolerable variations. JND models based on pattern masking function inferred from luminance contrast and structural uncertainty [31], and spatial masking function inferred from the structural regularity weighted luminance difference [32] are among other notable methods.

B. Frequency JND models

Frequency based JND models perform analysis mostly in frequency bands, instead of spatial domain. Several features such as CSF, LA, and CM are used in these models to predict the JND threshold. Wei et al. [15] proposed a spatio-temporal JND model based on spatial and temporal CSF, LA, CM, and the retina movement compensation. Experimental results show the proposed model has much better perceptual quality compared to pixel domain JND models. Due to the use of a fixed kernel size in earlier works, Bae et al. [16] proposed a DCT-based local distortion detection probability (LDDP) model for variable sized DCT kernels. LDDP estimates a degree of distortion visibility for any distribution of the transform coefficients. Zhang et al. [3] developed a JND-based rate-distortion optimization method where a JND-based distortion metric is used instead of the mean squares error. Moreover, Bae et al. [17] proposed a JND model that jointly considers the temporal masking and foveated masking effects in the DCT domain.

C. Image/video domain JND models

Although pixel domain and frequency domain JND methods demonstrated good performance, their shortcomings have led recent research to shift towards an image or video-wise JND prediction. Firstly, it is understood that HVS perceives the quality of the entire video frame rather than in pixel grain. Secondly, the handcrafted features used by these methods may not be usable in different applications [2]. Thirdly, the impact of the quantization process during encoding is disregarded by these models [1][33][34].

Image/video-wise methods predict JND thresholds for each image or video, and can be categorized into subjective and objective methods. Several studies have generated JND-based image/video datasets using subjective experiments. For example, Jin et al. constructed MCL-JCI, a JND dataset consisting of 50 JPEG-encoded images with 3 to 7 JND levels per image based on QF [18]. Then, Shen et al. [19] established an image-wise JND dataset for 202 images based on the next generation video coding standard Versatile Video Coding (VVC). Finally, Lin et al. [20] built a large image-wise dataset, KonJND-1k, with two compression schemes, JPEG and BPG. Also, there are two video-wise JND datasets called MCL-JCV [21] and VideoSet [22]. MCL-JCV and VideoSet consist of 3 JND levels for 30 and 220 videos, respectively.

Due to the time-consuming nature of subjective experiments, JND models calculate the JND threshold mainly based on Machine Learning (ML) approaches with subjective experiments serving as the ground truth. Liu et al. [35] formulated the JND prediction as a binary classification problem and then developed a Convolutional Neural Network (CNN)-based predictor on MCL-JCI dataset, which can estimate whether or not an image is perceptually lossy compared to its raw image. Tian et al. [36] also used MCL-JCI dataset and constructed a JND prediction using a CNN which estimates the value of first and last JND levels. Then, the authors have developed an SVR-based model which predicts the number of JND levels. Finally, they designed an algorithm which assigns the QF for each JND level. Nami et al. [37] proposed a JND method on MCL-JCI dataset, which combines saliency and JND concepts. They first developed a JND
predictor based on various quality metrics. Then, they measured the visual importance of each block and finally designed an algorithm which assigns a QP to each block. The use of an image quality assessment network to predict JND has been studied by Fan et al. [38].

Further studies have used video-based JND dataset including MCL-JCV or VideoSet to develop a JND predictive model. Takeuchi et al. [39] developed an SVR-based JND estimator using quality features including PSNR, SSIM and VMAF. Nami et al. [1] developed a framework for block-level JND prediction, where first a JND mapping method is used to derive block-level JNDs from frame-level information. Then, a CNN-based model is developed which estimates JND levels for each block according to spatial and temporal features. Finally, an efficient quantization control algorithm assigns a QP to each block, according to their JND levels, and visual importance. Zhang et al. [2] consider both spatial and temporal information, where temporal information is extracted using an optical flow algorithm [40], and then used it in a CNN-based JND predictive model. Finally, Nami et al. [23] proposed a multitask learning, to jointly learn JND prediction with visual attention modeling, or jointly learning multiple JND levels. It was demonstrated that joint learning of correlated tasks such as multiple JND levels, can improve the JND prediction performance.

As mentioned in Section I, existing JND methods are limited in the sense that they train the models based on a single JND level (e.g., a QP value). Given the limited size of existing JND datasets, and content-dependence of JND, this may not be sufficient to train a deep network for such a task. Moreover, different JND levels represent correlated information, all existing methods model each JND level independently. To address these shortcomings, in the next section we detail our proposed multitask solution, with two main ideas: (1) learning to reconstruct the JND-quality frames from raw frames to assist the JND prediction task, and (2) jointly learning multiple JND levels to enhance the learning performance and achieve computational efficiency.

### III. Proposed Method

#### A. Overview

Fig. 3 illustrates an overview of the proposed methods in comparison with existing methods. From Fig. 3(a), most of the previous studies follow a process where a video is first encoded with all (or multiple) QP values. These features are then used to predict the JND levels as (1):

$$ q_{p_{JND}} = f_P(X_{raw}, X_{qp1}, \cdots, X_{qp_{51}}) $$

(1)

where \( f_P \) is a function which predicts JND levels based on features of the raw frame \( X_{raw} \), and frames encoded with QPs from 1 to 51, i.e., \( q_{p1} \) to \( q_{p_{51}} \).

We propose a novel solution for JND prediction, where the reconstruction of JND-quality frames is first learned by an autoencoder-based model. Then, two types of data modalities from the autoencoder, the latent space obtained from the Encoder (E), and the JND-quality frames reconstructed by the Decoder (D), can be used for JND level learning, which can be described as (2):

$$ q_{p_{JND}} = f(X), \text{ where } X = \hat{X}_{JND} \text{ or } Y_{Lat} $$

(2)

where \( f \) is a function to predict JND levels based on features of JND-quality decoded frames \( \hat{X}_{JND} \), or latent space \( Y_{Lat} \) (depending on the method).

From Fig. 3(b), latent space information is used to train the LAT network which predicts the JND level directly from compressed frames based on two training strategies, called LAT, and E2E-LAT. Two REC networks (REC and E2E-REC) are trained on decompressed JND-quality frames, shown in Fig. 3(c). These methods are learned separately for each JND level. Moreover, two different methods are designed for Multi-JND (MJ) learning using the latent space of 3 JND levels as in Fig. 3(d), or reconstructed JND-quality frames as in Fig. 3(e). These
models are called MJ-LAT and MJ-REC, respectively. The following subsections detail these models, and Table I summarizes the notations used throughout the paper.

B. Reconstruction of JND-quality frames

An auto-encoder based network can convert raw frames into JND-quality frames. Therefore, we develop an auto-encoder based network that takes raw frames $X_{\text{Raw}}$ as input and outputs the latent space $Y_{\text{Lat}}$, and the decompressed JND-quality frames $X_{\text{JND}}$. As shown in Fig. 4, this reconstruction consists of two parts. (1) The encoder generates a compressed representation of the frame with lower dimensional features $Y_{\text{Lat}}$, and (2) the decoder converts the compressed data into JND-quality frames $X_{\text{JND}}$.

$$Y_{\text{Lat}} = E(X_{\text{Raw}}, \theta_e)$$

(3)

$$X_{\text{JND}} = D(Y_{\text{Lat}}, \theta_d)$$

(4)

Here, $E(\cdot)$ and $D(\cdot)$ are the encoder and decoder, respectively, while $\theta_e$ and $\theta_d$ are the weight parameters of the encoder and decoder, respectively.

**Network Architecture:** The auto-encoder based reconstruction model is depicted in Fig. 4. This network architecture is inspired by [24] and simplified for this task. We do not use a separate context model (such as a hyperprior network), as such a compression model is not the main goal of this network, and such a model would significantly increase the computational complexity [41]. A $5 \times 5$ kernel and $2 \times 2$ strides are used in all Convolutional (Conv) and DeConvolution (DeConv) layers. The activation function of all layers is set as Leaky Rectified Linear Unit (LReLU). Also, zero-padding is used to produce an output of the same size as the input.

Training this network requires a large number of samples. However, existing JND datasets only contain a limited number of samples that are insufficient for training. To remedy this, we first train the model on the large JPEG-AI dataset to learn high quality image reconstruction. Then the model is fine-tuned on the available JND dataset and learn to translate the frames to their JND-quality frames. Training is done on randomly selected patches of 256×256 pixels.

**Loss function:** The autoencoder is trained end-to-end, and its weights are updated using the Mean Squared Error (MSE) as the loss function, which is defined as follows:

$$L_{\text{IR}}(\theta_e, \theta_d) = \|X_{\text{JND}} - D(E(X_{\text{Raw}}, \theta_e), \theta_d)\|$$

(5)

where $X_{\text{JND}}$ is the frame encoded with the ground truth JND level. After training, the latent space and the decompressed frames from the auto-encoder are used for JND prediction, as detailed in the next subsections.

C. JND prediction based on latent space features

In this section, JND prediction is developed based on latent space and is modeled as a regression problem by (6):

$$\hat{q}_{\text{JND}} = f_{\text{Lat}}(E(X_{\text{Raw}}, \theta_e), \theta_t)$$

(6)

where $f_{\text{Lat}}(\cdot)$ and $\theta_t$ are the mapping function and the weight parameter of this model, respectively. Similar to most Video Quality Assessment (VQA) approaches, we use CNN for $f_{\text{Lat}}(\cdot)$ mapping function on spatial features.

---

### Table I

**IMPORTANT NOTATIONS USED THROUGHOUT THE PAPER.**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{\text{Raw}}$</td>
<td>Raw video frame</td>
</tr>
<tr>
<td>$X_{\text{QP}}$</td>
<td>Video frame encoded with qp</td>
</tr>
<tr>
<td>$X_{\text{JND}}$</td>
<td>Reconstructed $i$th JND-quality frame</td>
</tr>
<tr>
<td>$Y_{\text{Lat}}$</td>
<td>Latent space of video frame for $i$th JND level</td>
</tr>
<tr>
<td>$\hat{q}_i$</td>
<td>Predicted qp for $i$th JND level</td>
</tr>
<tr>
<td>$\theta_e$</td>
<td>Parameters of the encoder</td>
</tr>
<tr>
<td>$\theta_d$</td>
<td>Parameters of the decoder network</td>
</tr>
<tr>
<td>$\theta_l$</td>
<td>Parameters of the LAT network</td>
</tr>
<tr>
<td>$L_{\text{Lat}}$</td>
<td>Loss function for $i$th JND-quality frame reconstruction</td>
</tr>
<tr>
<td>$L_{\text{IR}}$</td>
<td>Loss function of Lat method for $i$th JND level</td>
</tr>
<tr>
<td>$L_{\text{EI}}$</td>
<td>Loss function of E2E-Lat method for $i$th JND level</td>
</tr>
<tr>
<td>$L_{\text{IR}_L}$</td>
<td>Loss function of Rec method for $i$th JND level</td>
</tr>
<tr>
<td>$L_{\text{ML}}$</td>
<td>Loss function of MJ-Lat method</td>
</tr>
<tr>
<td>$L_{\text{MR}}$</td>
<td>Loss function of MJ-Rec method</td>
</tr>
</tbody>
</table>

---
Network Architecture: The architecture of JND predictive model is depicted in Fig. 5. As mentioned earlier, the latent space obtained from the encoder side of the auto-encoder model is the input, and the QP corresponding to the JND level is the output. As shown in Fig. 5, the predictor consists of Conv, Batch Normalization (BN), Max Pooling (MP), Fully Connected (FC), and Channel-Attention (CA) layers. The kernel size is 3 × 3 in all three Conv layers. Each Conv layer is activated by ReLU. After every two Conv layers, MP layer with kernel size is 3 × 3 in all three Conv layers. Each Conv layer is activated by ReLU. After every two Conv layers, MP layer with 2 × 2 kernel size is used.

Among the several features maps constructed at each Conv layer, some are more important for the task at hand and are needed to be assigned a higher degree of importance. Hence, a channel-wise attention [42] is used which exploits the inter-channel relationship of the features and assigns a weight to each channel. The attention vector according to (7).

\[ Y' = \sigma(MLP(Y_{MP}) + MLP(Y_{AP})) \otimes Y \]  

(7)

Loss Function: As mentioned in Section I, we propose two strategies for learning JND-quality frame reconstruction, and the JND prediction task. The two tasks can either be learned sequentially or jointly, i.e., end-to-end. For the first strategy, the reconstruction network is first trained on JND-quality frames, based on (5). Then the JND prediction method, LAT, is trained based on the following Mean Absolute Error (MAE) loss function, where \( q_p \text{JND} \) is the ground truth JND.

\[ L_L(\theta_L) = |q_p \text{JND} - f_{\text{Lat}}(Y_{\text{Lat}}, \theta_L)| \]  

(8)

In the second strategy, E2E-LAT, both JND-quality frame reconstruction and JND prediction tasks are jointly learned with the loss function defined in (9), where \( \lambda_{IR} \) and \( \lambda_L \) are two nonnegative weights. It is worth noting that the reconstruction (decompression) task in LAT and E2E-LAT methods is trained only to assist the JND prediction task. In other words, the decoder network is discarded/deactivated after training, to save computation.

\[ L_{EL}(\theta_e, \theta_d, \theta_f) = \lambda_{IR} L_{IR} + \lambda_L L_L \]  

(9)

D. JND prediction based on reconstructed JND-quality frames

In this section, JND model is built based on reconstructed JND-quality frames, and is modeled as a regression problem by (10):

\[ q_p \text{JND} = f_{\text{Rec}}(D(E(X_{\text{Raw}}, \theta_d), \theta_r)) \]  

(10)

where \( f_{\text{Rec}}(\cdot) \) and \( \theta_r \) are mapping function and the weight parameter of this model, respectively.

Network Architecture: The architecture of JND predictive model is shown in Fig. 7. Recomposed frames from the decoder are input to the model, and the JND levels in terms of QP are the output. The predictor consists of Conv, BN, MP, and FC layers. The kernel size is 3 × 3 in all Conv layers, and 2 × 2 for all MP layers. Conv layers are activated by ReLU, and padding is used, as described earlier.

Loss Function: As mentioned earlier, the JND-quality frame reconstruction and JND prediction tasks can be trained either sequentially or jointly. For the first strategy, the reconstruction network is trained on JND-quality frames, based on (5). Then the JND prediction method, REC, is trained based on (11).

\[ L_R(\theta_r) = |q_p \text{JND} - f_{\text{Rec}}(X_{\text{JND}}, \theta_r)| \]  

(11)

In the second strategy, E2E-REC, both JND-quality frame reconstruction and JND prediction task are jointly learned with the loss function in (12), where \( \lambda_{IR} \), and \( L_I \) are two nonnegative weights.

\[ L_{ER}(\theta_e, \theta_d, \theta_f) = \lambda_{IR} L_{IR} + \lambda_L L_L \]  

(12)

E. Multi-JND learning based on the latent space

This section presents joint prediction of three JND levels, using multitask learning. Latent space features from three JND levels are fused, and used to model JND prediction as a regression problem. As explained in Section I, different JND levels are correlated, and jointly learning them improves generalization by learning a richer feature space.
Network Architecture: The architecture of JND predictive model, called MJ-LAT, is shown in Fig. 8. A feature fusion module [43] is designed that merges the latent features of 3 JND levels by element-wise multiplication operation. This fusion module can amplify or attenuate the importance of different features based on their relations with other features and has shown good performance in similar computer vision tasks, such as object detection [44], and video quality assessment [45]. The fused features are then processed with a shared network and passed to three identical decision tails to predict the three JND levels. The MJ-LAT network architecture is designed to be similar to the LAT network for each JND level.

Loss Function: In MJ-LAT, the weights of the reconstruction are first updated based on (5) and the weights of JND prediction are then updated based on a weighted sum as in (13), where \( \lambda_i \) is nonnegative weight for \( i^{th} \) JND level, and \( L_{Li} \) is the MAE for the \( i^{th} \) decision tail, according to (7).

\[
L_{M\ell}(\theta_i) = \sum_{i=1}^{3} \lambda_i L_{Li}
\]

(13)

F. Multi-JND learning based on reconstructed JND-quality frames
Multitask JND prediction is developed next based on features of reconstructed JND-quality frames of 3 JND levels.

Network Architecture: Fig. 9 shows the architecture of this method, called MJ-REC. Reconstructed frames of 3 JND levels obtained from the decoder side of the autoencoder are the inputs, and the JND levels are the output. Similar to REC network, MJ-REC is designed to fuse features from three branches prior to processing them with the decision tails.

Loss Function: here, the weights of the reconstruction are first updated based on (5) and the weights of JND prediction are then updated based on a weighted sum as in (14), where \( \lambda_i \) is nonnegative weight for \( i^{th} \) JND level, and \( L_{Ri} \) is the MAE for the \( i^{th} \) decision tail, according to (11).

\[
L_{MRe}(\theta_r) = \sum_{i=1}^{3} \lambda_i L_{Ri}
\]

(14)

IV. EXPERIMENTAL RESULTS
The experimental setup is first described followed by a detailed presentation and discussion of the experimental results. Subsection (a) introduces the datasets followed by the experimental setup in (b). (c) evaluates the compression and reconstruction backbone, (d) assesses the prediction error of the proposed JND predictive methods and compares them to existing methods on a video dataset, (e) compares the proposed methods with competing methods on an image dataset, (f) evaluates the compression performance of the proposed methods compared to existing methods, (g) evaluates the computational complexity, and (h) conducts an ablation study to further measure the effectiveness of the contributions.

A. Dataset
JPEG-AI [28] is used for (pre) training the auto-encoder network. Then VideoSet [22] is used for fine-tuning the trained auto-encoder network on JND-quality frames, and for training the JND prediction networks. Finally, MCL-JCI [18] is used for training the JND prediction networks for images.

JPEG-AI consists of 5264 images with high diversity for training and 350 images for validation. The dataset is mostly used for learning-based image codecs.

VideoSet is a video-wise JND dataset, consisting of 220 source videos with 3 JND levels, in terms of QP values.

MCL-JCI is an image-wise JND dataset, consisting of 50 source images with several JND levels, in terms of QF values.

B. Experimental setup
For fair comparison with competing methods, test set separation was done similarly for all methods, according to the test image/video numbers reported in [2] for VideoSet, and in [36] for MCL-JCI. This accounts for non-overlapping 60%, 20%, and 20% partitions for training, validation, and testing, respectively. Adam optimization is used to train the proposed methods. The learning rate and the number of epochs for all methods are set to 10^{-5} and 300, respectively. The only exception is the compression and reconstruction network on JPEG-AI dataset, for which these numbers are set to 10^{-4} and 500, respectively. The final model for all methods is chosen based on the one with the lowest validation loss. All methods are implemented in Tensorflow 2.4.1 and Python 3.8 and are trained with Tesla V100, with the exception of timing evaluations in subsection G, for which an older GPU is used, for the sake of fair comparison (details in IV.G).

C. Evaluating the compression and reconstruction network
While image reconstruction is not the main task in this work and is learned to assist the JND prediction, we measure and report the reconstruction results in Table II. The first row measures the reconstructed images against the raw images in JPEG-AI dataset, in terms of PSNR, SSIM and MAE. The next rows measure the quality for the network fine-tuned on JND-quality frames of VideoSet. Hence, the reconstructed JND-quality frames are measured against the ground truth JND-quality frames. It is observed that JND reconstruction task is learned with a high quality to assist the JND prediction task. For
JND\(_2\) and JND\(_3\), the quality is slightly lower, which is explained by the accumulated error in the dataset collection phase [22], and it is consistent with their lower accuracy of prediction in previous works [1][37].

### D. JND prediction evaluation on Video dataset

The JND prediction error of the proposed methods are reported in Table III in terms of \(|\Delta q_p|\), \(|\Delta PSNR|\), \(|\Delta VMAF|\), \(|\Delta SSIM|\), and \(|\Delta MS-SSIM|\). These values represent the absolute differences between the quality of the video encoded with the ground truth JND level, and the one encoded with the predicted JND level (smaller is better). While all proposed methods achieve good results, it is observed from Table III that (1) multi-JND learning methods, MJ-LAT and MJ-REC, achieve better results than their equivalent single-JND methods. This due to the fact that they are trained with complementary information of 3 JND levels which improves their generalization ability. For MJ-REC the sum of \(|\Delta q_p|\) from 3 JND levels is improved by 0.46. (2) JND prediction methods based on reconstructed JND-quality frames (REC, E2E-REC, and MJ-REC) achieve smaller prediction errors compared to methods based on latent space (LAT, E2E-LAT, and MJ-LAT). These methods have access to the reconstructed JND-quality frames and extract efficient features which represent the quality of the frames. However, we discuss in Section IV.G that latent space methods require much less computation while still achieving comparable performance that surpass most previous works. (3) The proposed MJ-REC method shows the best performance in almost all the evaluation metrics. (4) The E2E training strategy achieves a better performance compared to sequential training in all cases but one.

To further illustrate the improved performance of multi-JND learning method, we visualize the sum of mid-level feature maps (R-Block, 256 after ReLU activation layer) learned by REC and MJ-REC for a few frames of VideoSet videos. It can be observed that MJ-REC learns semantically richer features compared to REC method according to Fig. 10.

In Table IV, the proposed MJ-REC is compared with 6 existing JND methods, namely EJNDM [13], SUR-Net [38], BL-JUNIPER [1], PW-JND [35], MT-3LJND [23], and STSUR-QF [2] in terms of \(|\Delta q_p|\), \(|\Delta PSNR|\), and \(|\Delta SSIM|\). All methods are tested with similar settings and test sets for fair
Fig. 12. The correlation between predicted and ground truth PSNRs for (a) JND₁, (b) JND₂, and (c) JND₃.

TABLE III
THE PERFORMANCE OF PROPOSED JND PREDICTION METHODS.

<table>
<thead>
<tr>
<th>Method</th>
<th>LAT</th>
<th>E2E-LAT</th>
<th>MJ-LAT</th>
<th>REC</th>
<th>E2E-REC</th>
<th>MJ-REC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JND₁</td>
<td>JND₂</td>
<td>JND₃</td>
<td>JND₁</td>
<td>JND₂</td>
<td>JND₃</td>
</tr>
<tr>
<td>(\Delta q_p)</td>
<td>1.95</td>
<td>2.41</td>
<td>2.34</td>
<td>1.91</td>
<td>2.39</td>
<td>2.23</td>
</tr>
<tr>
<td>Sum of 3 (\Delta q_p)</td>
<td>6.7</td>
<td>6.53</td>
<td>6.33</td>
<td>6.22</td>
<td>6.25</td>
<td>5.79</td>
</tr>
<tr>
<td>(\Delta \text{PSNR})</td>
<td>0.93</td>
<td>1.24</td>
<td>1.23</td>
<td>0.92</td>
<td>1.24</td>
<td>1.17</td>
</tr>
<tr>
<td>(\Delta \text{SSIM}\times10^{-3})</td>
<td>1.74</td>
<td>4.41</td>
<td>7.29</td>
<td>1.68</td>
<td>4.08</td>
<td>6.91</td>
</tr>
<tr>
<td>(\Delta \text{MS-SSIM}\times10^{-3})</td>
<td>2.14</td>
<td>4.87</td>
<td>7.55</td>
<td>2.05</td>
<td>4.57</td>
<td>7.17</td>
</tr>
</tbody>
</table>

TABLE IV
COMPARING THE PROPOSED METHOD (MJ-REC) WITH SIX EXISTING METHODS

(Results marked with * are reported from a similar implementation in [2]).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JND₁</td>
<td>JND₂</td>
<td>JND₃</td>
<td>JND₁</td>
<td>JND₂</td>
<td>JND₃</td>
<td>JND₁</td>
</tr>
<tr>
<td>(\Delta q_p)</td>
<td>1.57</td>
<td>2.11</td>
<td>2.11</td>
<td>2.64</td>
<td>2.93</td>
<td>2.82</td>
<td>1.84</td>
</tr>
<tr>
<td>Sum 3 (\Delta q_p)</td>
<td>5.79</td>
<td>8.39</td>
<td>6.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\Delta \text{PSNR})</td>
<td>0.72</td>
<td>1.06</td>
<td>1.08</td>
<td>1.31</td>
<td>1.3</td>
<td>1.47</td>
<td>0.88</td>
</tr>
<tr>
<td>(\Delta \text{SSIM}\times10^{-3})</td>
<td>1.39</td>
<td>3.97</td>
<td>7.09</td>
<td>2.71</td>
<td>2.71</td>
<td>5.64</td>
<td>8.42</td>
</tr>
</tbody>
</table>

comparison. BL-JUNIPER is reimplemented on a frame-level to match other methods. Results of EJNDM, PW-JND, and SUR-NET are reported from a similar implementation in [2]. It is observed that (1) our proposed MJ-REC obtains the lowest prediction error, surpassing the current state-of-the-art STSUR-QF [2]. (2) The pixel domain JND prediction method EJNDM achieves the worst results, as it cannot express the HVS opinion on the entire frames.

To show video specific performances, Fig. 11 summarizes the prediction error of first JND level versus the ground truth in terms of (15):

\[
\Delta q_p = q_p^{\text{JND}_1} - \bar{q}_p^{\text{JND}_1}
\]

(15)

where the horizontal axis shows the test video indexes and the vertical axis represents \(\Delta q_p\) (closer to zero is better). It can be observed that MJ-REC, STSUR-QF, and MT-3LJND achieve the smallest prediction errors for most videos, and in most cases, MJ-REC achieves the best results.

Moreover, we measure the coefficient of determination (R²) for PSNR of the predicted and ground truth JND levels, and compare it among MJ-REC, MT-3LJND, and BL-JUNIPER in Figs. 12(a), (b), and (c) for first, second, and third JND level, respectively. The horizontal axis shows the ground truth PSNR, and the vertical axis represents the predicted PSNR. R² values for MJ-REC, MT-3LJND, and BL-JUNIPER are 0.91, 0.89, and 0.71 in JND₁, 0.89, 0.86, and 0.80 in JND₂, 0.89, 0.86, and 0.80 in JND₃, respectively. The high R² values observed for all JND levels further confirm the prediction accuracies reported earlier in this paper.

E. JND prediction evaluation on image dataset

To further evaluate the proposed method, we compared MJ-REC with 2 existing solutions, PJND [36] and MT-3LJND [23] in terms of \(\Delta q_p\) and \(\Delta \text{PSNR}\) on the image JND dataset, MCL-JCI. Fig. 13 shows that MJ-REC obtains the smallest prediction error in terms of \(\Delta \text{PSNR}\) for all JND levels. Also, MJ-REC achieves the smallest \(\Delta q_p\) for JND₂ and JND₃, and close to the best for JND₁. The average \(\Delta q_p\) for the 3 JND levels are 5.8, 6.03, and 7.4 for MJ-REC, MT-3LJND, and PJND, respectively.

F. Compression performance evaluation

Perceptual video coding algorithms usually optimize the compression settings based on the predicted JND values, and the target QP. Previous studies demonstrated that in case of accurate JND prediction, significant bitrate reduction can be achieved, with negligible loss of visual quality [1][33][46], and hence, compression efficiency can partially be achieved given accurate JND prediction. However, to compare the compression
To measure $G$, we used a QP allocation algorithm that assigns the JND-level QP closest but not smaller than the target QP, as the optimal compression point [46]. Table V compares $G$ among MJ-REC, MT-3LJND, and BL-JUNIPER for four target QPs of 22, 27, 32, and 37. All three JND levels were used for QP allocation. It is observed that MJ-REC achieves the highest gain in all cases, with an average of 8.59.

G. Computational complexity analysis

JND-based methods often need to be executed several times, and for multiple JND levels for bitrate ladder generation or quality evaluation, which can happen over limited resources, such as edge servers [48]. Hence, computational complexity is an essential factor in evaluating JND methods.

We measure the complexity in terms of inference time and the number of network parameters, which are summarized in Table VI. For fair comparison, all methods have been tested on an Nvidia GTX1080Ti GPU. Moreover, the JND accuracy is included for each method, to compare the cost-efficiency. It can be observed that: (1) for single-JND prediction, LAT and REC methods need very low inference times, which are 85% and 77% lower than the current state of the art STSUR-QF [2]. Although these inference times are higher than those of BL-JUNIPER [1], their smaller $|\Delta q|$ and their number of parameters which are multiple times smaller than BL-JUNIPER, makes up for their longer inference time. (2) for predicting 3 JND levels, all single-JND methods run separate networks, which require roughly three times the complexity of a single JND level. However, multi-JND methods (MJ-LAT and MJ-REC) share the computation and parameter for all three tasks, leading to the lowest inference times. Compared to REC, MJ-REC saves 40% in inference time, and 42% in the number of parameters, while achieving a higher accuracy. Compared to STSUR-QF, MJ-REC requires 86% lower inference time and 87% fewer parameters. (3) The latent space prediction methods (LAT and MJ-LAT) achieve the lowest inference times for 1 and 3 JND levels, respectively. Compared to STSUR-QF, MJ-REC requires 86% lower inference time and 87% fewer parameters. Compared to already fast REC and MJ-REC methods, these methods achieve 33% and 46% time saving, respectively, as they operate directly in the compressed frames, without decompression. Even though these methods achieve lower accuracies compared to the proposed reconstruction-based methods, their performance is still better than most competing methods and comparable to the best.
results, while being much faster. Moreover, as they operate in the compressed domain, they require less memory and can potentially improve privacy.

Finally, it is worth noting that the reported times for STSUR-QF includes only the model inference time, which is added on top of the time required for compression and decompression of videos with seven QP values, as a preprocessing step. In contrast, the proposed methods are applied directly on the raw frames, with no further steps.

H. Ablation study and sensitivity test

To further examine the effectiveness of the proposed contributions, we define three Baseline (B) methods. (1) B-LAT is designed with the same architecture as LAT, but on a general autoencoder trained on raw frames (i.e., it has not seen the JND-quality frames). (2) Similarly, B-REC is similar to REC, but the autoencoder is trained on raw frames. (3) B-Raw is a JND model which predicts the JND levels based on raw frames, with a network identical to the REC. Moreover, we trained our proposed End-to-End methods with 4 different sets of weight, to measure their sensitivity to the choice of hyperparameters (weights). Four pairs of weights (λ₁, λ₂) are used in all experiments which are W₁ = [0.005, 0.995], W₂ = [0.01, 0.99], W₃ = [0.1, 0.9], and W₄ = [0.001, 0.999]. Figs. 15 and 16 summarize these experiments for latent space-based and reconstructed-based methods, respectively.

It can be observed that (1) all proposed methods consistently outperform the corresponding baseline methods in terms of prediction error, |Δqp| and |ΔPSNR|, in all JND levels, which shows the importance of learning the JND-quality frames, (2) the proposed End-to-End methods consistently show improved performance compared to their baselines, regardless of the choice of weights. This indicates that these methods are robust to the choice of hyperparameters. (3) B-Raw achieves better performance compared to B-REC, while REC outperforms both with a large margin. This interesting observation shows that not only learning features from the best (raw) quality frames does not help the JND prediction, but it can degrade the prediction compared to directly using raw frames. In contrast, learning from the JND-quality frames, provides representative features that even outperform the raw frame features. This affirms the underlying motivations introduced in Section I.

V. CONCLUSION

This paper proposed a novel framework for accurate JND prediction, where we learn to reconstruct the JND-quality video frames, to assist the JDN prediction task. Two types of methods were presented that predict JND levels based on either the latent space (compressed) data, or the reconstructed frames. Moreover, we proposed multi-JND methods that jointly learn three JND level, boosting the prediction performance. Through extensive experiments, it was demonstrated that: (1) learning JND-quality frames for JND prediction can extract more effective features and improves JND prediction performance. (2) Predicting JND levels from JND-quality frames boosts the learning performance beyond the raw frames. (3) Joint learning of multiple JND levels extracts features that are relevant to all tasks simultaneously, leading to improved performance. Moreover, this leads to major reduction of inference time and number of parameters. (4) JND prediction can be developed based on latent space features, providing a lightweight and efficient method. As this method is applied directly on the
compressed frames, it is useful for edge inference and privacy preservation. (5) Extensive experimental results over two JND datasets (MCL-JCI and VideoSet) confirm that the proposed framework outperforms existing methods.

REFERENCES


