A Byte-based GPT-2 Model for Bit-flip JPEG Bitstream Restoration

Hao Qin, Haoran Sun, and Yi Wang
Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong SAR
E-mail: {fuyuyu.Qin, haoran-eee.sun}@connect.polyu.hk, yi-eie.wang@polyu.edu.hk

***Abstract*—In this paper, we investigate the application of large language models (LLMs) for the recovery of corrupted bitstreams, specifically focusing on JPEG image data. We propose a byte-based GPT-2 model that directly processes byte sequences and predicts the subsequent byte, enabling its application to JPEG bitstream recovery. This architecture allows the model to capture the relationships between consecutive byte data within the bitstream of a JPEG image, such that the model can restore the bit-flip errors due to the damaged storage and malicious attack. We evaluate the model's performance on bit-flip JPEG datasets with varying bit error rates (BERs). The experimental results demonstrate the model's ability to implicitly learn patterns in the bitstream and correct erroneous bytes, showcasing the potential of LLMs in binary processing tasks. Our findings highlight the promise of byte-based LLMs in addressing data corruption issues and open up new avenues for research in this domain.**

I. Introduction

In the face of increasing transmission bandwidth demand for streaming media, audio and video are transmitted under various lossy compression techniques [1], such as MPEG-4 [2] and JPEG [3]. These techniques provide the codec to compress the data into a stream of bytes at a specified compression ratio and decompress it back into images and audio. In scenarios where transmission and storage may be interfered with or intentionally attacked [4]–[6], bit flips or missing will occur in the stream which corrupts blocks or the entire bitstream. Recovering data from a corrupted bitstream is a challenging task. Typically, based on the optimization principle, the data segments in the bitstream of a lossy compression method do not contain redundant data for error correction and use irregular lengths and code words for different data[7].

On the one hand, researchers have focused their problem-solving efforts on coding techniques. Modern error correcting codes (ECC) such as LDPC [8] are used in wireless transmission and digital video broadcasting controlling errors in data transmission. Message-digest algorithms such as MD5 [9]are used for verifying data integrity and creating digital signatures. Other researchers have proposed models to extract pattern and structural features from the correctly decoded portion of the bitstream, complementing the image or interpolating it [10]. Nevertheless, corruption caused by bit flips or missing bits often results in relatively long segments of data that cannot be decoded. Repair of decoded content relies on the temporal and spatial continuity of the data, which is not always satisfactory.

In this work, we modified the GPT-2 [11] model to accept a sequence of bytes as input and predict the next byte. We intend to use the GPT-2 model for JPEG bitstream restoration. The idea is inspired by the fact that image bitstreams in JPEG format have a fixed metadata structure [3], and if the byte sequences in data segments are considered as statements, the codebook in the metadata defines all the semantics recurring in the bitstream. We modeled bit-flip corruption on JPEG image bitstream with existing image datasets such as ImageNet [12] and CIFAR-10 [13]. Trained on these datasets, our proposed binary model successfully recovers a portion of corrupted files. To further evaluate the performance, we restricted the bitstream corruption to occur in different regions and evaluated the model at different bit error rates (BERs). The results show that LLM can capture the relationship between stream metadata and data segments.

Overall, this paper makes the following contributions:

- We adopt Large Language Models (LLMs) to process byte sequences directly and predict the subsequent byte. The proposed byte-based GPT-2 model can be applied directly to the task of JPEG bitstream recovery.
- We evaluate our method on bit-flip JPEG datasets. Experimental results show the potential of the GPT-2 model in this field. To the best of our knowledge, this marks the first instance of utilizing an LLM to restore corrupted bitstreams.

II. Related Work

JPEG structure. The file bitstream of JPEG format is partitioned into two parts: the metadata segment and the image data segment. The image data is compressed by Discrete Cosine Transform (DCT) and then converted into a variable-length coding structure after quantization, run-length encoding (RLE), and Huffman coding as shown in Fig. 1. The metadata stores the important parts such as the Huffman coding table, quantization table, etc., which accounts for a small percentage of the whole bitstream. The data segments are sequence-dependent, so bit-flip and missing errors will seriously interfere with decoding, resulting in image corruption or even the inability to read the entire bitstream. Image corruption can happen in a variety of patterns.

Bitstream restoration. Much of the previous work has focused on using techniques for specific codecs for concealment or fixing. Error concealment is a common decoder-side post-processing technique used to repair erroneous regions in decoded videos [14], [15]. It can be divided into several

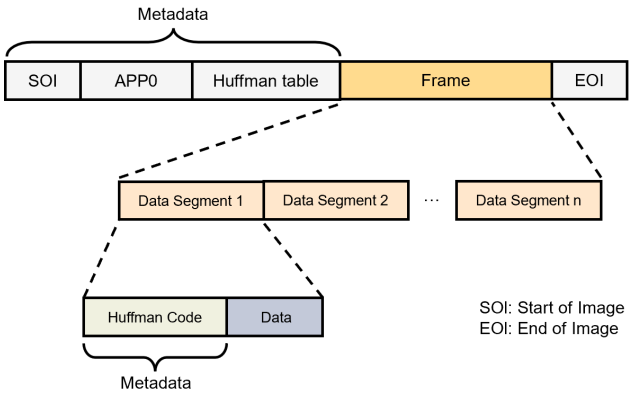

Fig. 1. The structure of JPEG bitstream.

types, including spatial, temporal, and hybrid spatiotemporal methods, covering both bitstream and pixel levels [15], [16]. In recent years, deep learning-based methods usually assume traditional damage patterns and use experimental masks to simulate stripes or block losses [14], [17]–[19]. However, this approach is not suitable for recovering videos with damaged bitstreams because the damage caused by actual packet loss is often difficult to predict and has no regularity. The approach in [20] constructed a robust decoder that attempted to skip over erroneous blocks and parse the remainder of the bitstream. The parsed corrupted images were restored using a two-stage compensation and alignment framework.

Byte models. From early LSTM to today's language models based on the Transformer architecture [21], they play a vital role in understanding, generating human language, and simulating intelligence. Text tokenization breaks down text into smaller units (such as words or subwords) as model input [21]. The Generative Pre-trained Transformer (GPT) model marks a major advancement, utilizing self-supervised learning and next-token prediction pre-training to capture the structure and semantics of language. Researchers are also exploring byte-level encoding methods to improve the performance of existing models. For example, compressing byte sequences using language models [22] offers new insights into leveraging large pre-trained models [23]. Byte-level byte pair encoding (BBPE)[24] has shown promise in enhancing multilingual model pre-training and machine translation [25]. ByT5 [26] builds upon this by processing byte sequences with a standard Transformer model, generalizing an unlabeled encoding approach for improved noise robustness and spelling sensitivity in multilingual scenarios.

III. Our Method

In this section, we describe the byte-based GPT-2 modeling. Processing data from the byte level leads to a model with high granularity and extremely long sequences. The computational effort of self-attentive scaling in Transformer-based models grows quadratically with the length of the sequence [27]. To avoid excessively long sequence lengths, bytes are partitioned into patches in a multi-layer transformer, referring to the work of Wu et al. [27].

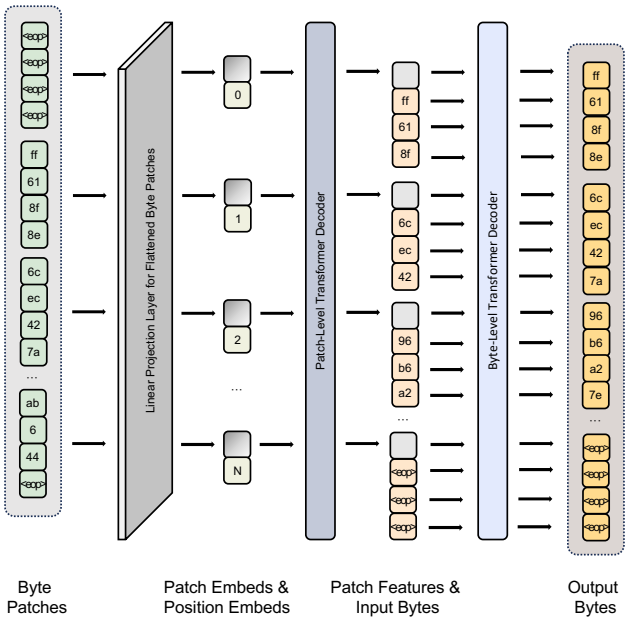

Fig. 2. The structure of byte-based two-layer GPT-2 model.

As shown in Fig. 2, the byte-based GPT-2 model contains three parts, the linear projection layer followed by positional coding to the sequence, the patch-level decoder, and the byte-level decoder.

Bitstream Preprocessing. In a bitstream, data is not always organized by bytes. There are many semi-byte blocks and flag bits in different bitstreams (e.g., communication protocols, standard file types, CPU states, etc.). In variable-length bitstream, at the end of the file, bits often cannot be rounded up to a byte, and padding is required to fill up the data. For a sequence of bytes (bitstream) $B = [b_0, b_1, b_2, b_3, ..., b_{T-1}], b_i \in \mathbb{Z}_{256} = \{0, 1, 2, ..., 255\}$ of length T, we pad $8 - |b_{T-1}|$ zeros at Byte b_{T-1}.

The direct use of B as input leads to long sequence lengths, and to solve this problem, inspired by previous studies[20], [28], we split the B into patches with a certain size S, forming a sequence of patches $P = [p_1, p_2, ..., p_K]$ with the length of K, where $K = \lceil \frac{T}{S} \rceil$. To locate the endings of the bitstreams, noting that the common data file storage formats usually have a start-of-file marker and an end-of-file marker, we add start-of-patches (SOP) and end-of-patches (EOP) marker to the beginning and end of the byte sequence. The SOP and EOP markers share one single symbol, each occupying a byte position. A special patch is formed by adding S markers to the beginning to indicate that this is the start patch, and at the end we need to add a sufficient number of markers to indicate that this is the end patch, and the number of patches to add is $S + Mod(-K, S)$.

Linear Projection Layer. Firstly, there are only 256 hexadecimal digits that can be represented by a byte, so a byte can

be represented by a one-hot vector of length 256, and adding a symbol to stand for the EOP marker gives a length of 257. A patch contains S bytes, which can be represented as a matrix of $S \times 257$. This matrix is flattened to a vector of length $S \times 257$ and linearly weighted to perform token embedding on patches. For error correction type applications, the byte vector will be in Gray code order. For each input patch, the linear projection layer maps the sparse input vector to a dense vector D of size (S, H), with H being the size of the hidden layer, and this process can be formulated as follows:

$$D_i = V_i \cdot W_{linear}, 1 \leq i \leq K \tag{1}$$

where W_{linear} denotes the weight matrix of the linear projection layer.

Byte Decoder. We used two layers of transformer decoders to make predictions for positionally encoded patches and bytes, respectively. First, the patch-level transformer makes predictions based on the sequence of embedded patches to get predicted features for the predicted patches. Then byte-level decoder predicts the next byte based on the previous patches' prediction features and the sequence of bytes within the current patch. Considering the existence of byte-level semantics between codewords in the byte stream, the computations performed at the byte decoder are not independent for each patch, unlike the work of Wu et al. [27].

Loss Function. Our goal is to predict the next byte of the sequence and try to keep it consistent with the uncorrupted sequence, given the existing corrupted sequence, which requires us to adopt a generative modeling framework for the process. For a sequence of bytes $B = \{b_1, b_2, b_3, ..., b_T\}$ of length T, the log-likelihood of the next byte needs to be maximized, so the loss function can be defined as:

$$\mathcal{L}(\Theta) = -\sum_{i=1}^{T-1} \log p(b_{i+1}|b_1, b_2, b_3, ..., b_i; \Theta) \tag{2}$$

where Θ denotes the model parameter $p(\cdot)$ denotes the conditional probability distribution of the next byte prediction. bitstream restoration is a special case of generative modeling, where the model predicts bytes consistent with the original content in most cases. In this case, not all bytes in the sequence are meaningful, and the sequence may have potential semantic sparsity, which remains to be investigated.

Inference. For inference, the patches predicted by the model are unwrapped and the padded zero bits at the end of the bitstream are removed to obtain the predicted sequence.

IV. Experiments

A. Experimental Settings

1) Datasets: For bitstream corruption, there is no publicly available dataset other than a benchmark for video bitstream corruption recovery [29]. To evaluate the model in this paper, we use the ImageNet and CIFAR-10 datasets, which are image datasets used for image recognition and classification, to which we artificially add single-bit-flip errors to validate the viability of the model for JPEG bitstream recovery.

Bit errors happen randomly throughout the whole bitstream and the errors occurring in the Huffman coding table cause the entire file to be unreadable, while errors occurring in the data segments lead to multiple forms of error patterns. Among these, metadata such as the Huffman coding table accounts for less than 0.1% of the total bitstream size. With a low bit error rate (BER), i.e., no larger than 10^{-2}, the bit flips that occur in the metadata are extremely rare. Metadata has a fixed structure and logical order, as well as a limited number of possible values, and we can use its structural information for error correction. Hence, the bit flips in metadata do not have a significant problem for file reading. The bit-flip errors in metadata can be efficiently corrected using the robust decoder proposed in the study of [20].

To simulate corruption, we added noise to the JPEG image bitstream at several certain BER levels and augmented some of the files to suggest where different errors occurred for the same bitstream. Considering the training cost, we use the context length of 8k, which means that the size of the JPEG image file should not be more than 7.8 KB or 128×128. We filtered the images that do not exceed 7.8 KB in the ImageNet dataset to be used as the training data. The image resolution of the CIFAR-10 dataset is 32×32, which meets the demand.

2) Evaluation Metrics: In the field of communication, the quality of a digital signal after transmission can be evaluated in terms of peak-signal-to-noise ratio (PSNR). For the bitstream restoration problem in this paper, the upper bound of the signal-to-noise ratio ratio is the BER if the network does not produce incorrect predictions. It is one-sided to use only one single metric, the peak-signal-to-noise ratio, to evaluate the effectiveness of the model. For the bitstream recovery problem, we propose new definitions of accuracy and recall to replace PSNR, as follows.

$$Precision = 1 - \frac{\sum_i min\{L_i, L_i^{gt}\}}{\sum_i L_i^{gt}}, \tag{3}$$

$$Recall = 1 - \frac{\sum_i N_i^D}{\sum_i L_i^{gt}}, \tag{4}$$

where L_i^{gt} represents the Levenshtein distance [30] between the ground truth and the corrupted bitstream, L_i represents the Levenshtein distance between the predicted bitstream and the correct bitstream, and N_i^D denotes the number of successful fixes of the prediction result at the wrong bits. In addition, we use a metric of human sensing of the image to evaluate the quality of the recovered image. which is based on the PSNR of the pixels of the image to the original image after reading the bitstream as an image. This metric is specifically calculated as follows: the peak-signal-to-noise ratio of pixels to the ground truth image after the recovered stream is read as an RGB image.

3) Parameter Settings: The embedding and hidden state dimensions of Byte-Level are set to 3 and 768 respectively. The embedding and hidden state dimensions of Patch-Level were set to 12 and 768 respectively. The Adam optimizer [31]

TABLE I
OVERALL PERFORMANCE OF THE MODEL ON THE IMAGENET DATASET.

Bit error rates	ImageNet	
	Precision(%)	Recall(%)
10^{-2}	0.380	1.424
10^{-3}	8.950	13.719
10^{-4}	17.265	21.101
10^{-5}	22.430	22.544

TABLE II
OVERALL PERFORMANCE OF THE MODEL ON THE CIFAR-10 DATASET.

Bit error rates	CIFAR-10	
	Precision(%)	Recall(%)
10^{-2}	0.632	3.215
10^{-3}	8.950	13.719
10^{-4}	21.123	28.545
10^{-5}	19.989	22.468

was used. The initial learning rate was 0.0001 and the batch size was 32. A drop rate of 0.1% was used during training.

B. Results

In our experiments, we evaluated the performance of the model at four BER levels on two datasets. In particular, a total of 2,402 images from the ImageNet dataset were used, which were allocated in a 10:1 ratio of training/test set. The training set was augmented by adding noise and the model was trained with both the corrupted and uncorrupted images, totaling 11020 images. For the CIFAR-10 dataset, a total of 50,000 images from 10 categories were used as the training set and 10,000 as the test set. Noised images were aligned with the uncorrupted images and a total of 100000 images were provided for training. For comparison with approaches related to bitstream restoration, we choose the stage-one robust decoder in [20] as the baseline, since this process is for bitstreams and not for image data.

1) Overall Performance: **Precision and recall.** The overall precision and recall of the prediction results for the ImageNet test set are shown in Table I and the overall results for the CIFAR dataset are shown in Table II. As can be seen in Tables I and II, the prediction accuracy of the model basically improves gradually as the BER decreases. Optimal performance is achieved on the CIFAR-10 dataset at a BER of 10^{-4} and on the ImageNet dataset at a BER of 10^{-5}. Theoretically, the lower the BER, the better the performance of the model. The reason that CIFAR-10 achieves optimal performance on 10^{-4} BER is that 10^{-4} and 10^{-5} will only produce bit errors below 1 bit on average for the image size 32×32 of the CIFAR-10 dataset. We only tested the files that produced bit errors, so it is conditionally the same for CIFAR-10. Precision and recall cannot be reasonably calculated for images recovered using the robust decoder in [20]. It adopts a zero-padding strategy to skip blocks that cannot be parsed. As a result, the precision and recall of the robust decoder have significantly reduced, and the comparison is not meaningful.

TABLE III
OVERALL PERCENTAGE AND AVERAGE PSNR OF IMAGES RECOVERED.

Bit error rates	Recovery Rate(%)		PSNR(dB)	
	ImageNet	CIFAR-10	ImageNet	CIFAR-10
10^{-2}	4.082	1.460	3.523	3.389
10^{-3}	16.226	13.510	12.063	11.221
10^{-4}	56.122	36.180	19.994	18.190

TABLE IV
COMPARISON OF DIFFERENT METHODS IN OVERALL PERCENTAGE AND AVERAGE PSNR OF IMAGES RECOVERED A BER LEVEL OF 10^{-4}.

Model	Recovery Rate(%)		PSNR(dB)	
	ImageNet	CIFAR-10	ImageNet	CIFAR-10
Robust decoder [20]	44.358	26.115	17.733	16.889
Our Model	56.122	36.180	19.994	18.190

Pixel's level performance. Since the metadata (e.g., Huffman code) segment is relatively sparse in variable-length bitstream, the JPEG stream is mostly filled with data segments. When the bits in the data segment are changed, it only affects the color, texture, and other data of the specific image block and does not affect the decoding of the bitstream, so errors occurring in data segments have less effect on the image. It is impossible to evaluate the quality of the image recovered from the metrics of bitstream restoration alone, so we use the PSNR as an evaluation metric. The image recovered with higher PSNR is considered as successfully recovered image, and the results are shown in Tables III. Compared to the results in Tables I and II, the performance of the model is better, this is because the model is unable to identify whether the data segments are noisy or not and thus cannot make an accurate prediction of the segment, which reduces the accuracy of the model. At an error rate of 10^{-5}, the model recovered 56.122% of the corrupted images on ImageNet and 36.180% of the corrupted images on CIFAR-10.

Table IV demonstrates the results of the performance comparison between our model and the robust decoder in [20] at a BER level of 10^{-4}. Compared to the robust decoder, our model achieves a performance improvement of 2.26dB PSNR and 11.76% recovery rate on ImageNet. These results show that our method is more advanced in bitstream restoration.

Visual Results. Fig. 3 and Fig. 4 show a comparison of the visual quality of the images before and after the restoration of the robust decoder [20] and our model on the two datasets, where the recovery gets progressively better from left to right. It is worth noting that the robust decoder is unable to produce effective parsing results because the image size of CIFAR-10 is too small. The samples selected in Fig. 3 and Fig. 4 are selected from corrupted bitstreams that can still be decoded, albeit with severe discoloration and block artifacts. It is important to note that these samples represent a subset of the corrupted files, as some corrupted bitstreams cannot be decoded. From the figures, it can be seen that after the recovery by our model, some images are completely restored while others are

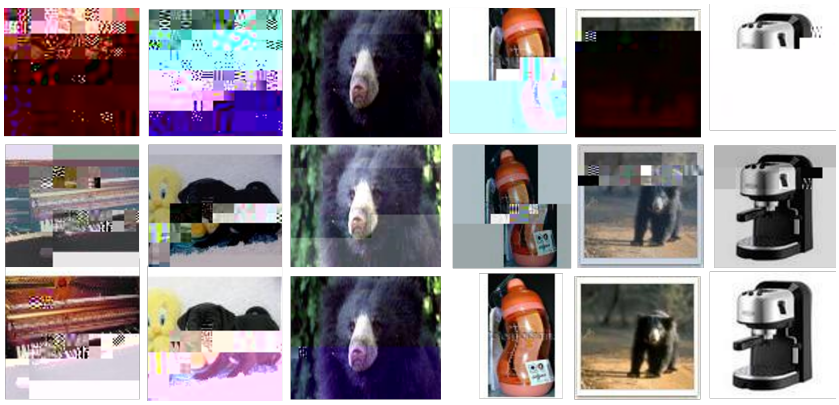

Fig. 3. Visual comparison of the images before (at the top) and after restoration of the robust decoder [20] (at the middle) and our model (at the bottom) on ImageNet at a BER level of 10^{-4}.

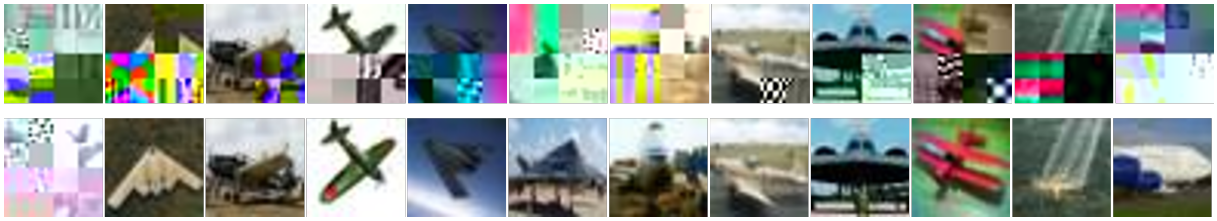

Fig. 4. Visual comparison of the images before (at the top) and after (at the bottom) restoration on CIFAR-10 at a BER level of 10^{-4}.

TABLE V
RECOVERY RATE ON THE BENCHMARK OF NOISY METADATA SEGMENTS.

Bit error rates	Dataset	
	ImageNet	Cifar
10^{-2}	0.000%	0.000%
10^{-3}	0.090%	0.810%
10^{-4}	17.440%	20.010%
10^{-5}	62.190%	81.780%

compensated for the color distortion and block shifts, thus providing better visual effects.

2) Recovery in Metadata of Bitstream: In order to verify that the model achieves recovery of the bitstream and not only a pseudo-positive due to bit errors in data segments, we constructed a benchmark using the test set of both datasets. We made twenty copies of all images and added errors at different locations in the metadata segment. From Table V, it can be seen that for samples corrupted only in the metadata segment, the model still recovers the data, which indicates that the model learned the inter-byte relationship of metadata from the bitstream.

V. CONCLUSIONS

In this paper, we propose a novel JPEG bitstream recovery method based on GPT-2, which explores the potential of large language models (LLMs) in the restoration of corrupted bitstreams. Our model employs a large language model to process binary information, achieving an end-to-end processing flow. Experimental results demonstrate that the proposed method achieves a 56% recovery rate using an ImageNet dataset at low BER levels. In future work, we aim to explore more LLMs and variable content-length embedding for generalized bitstreams to enhance the model's prediction performance and improve the quality of recovered JPEG images.

REFERENCES

[1] A. J. Hussain, A. Al-Fayadh, and N. Radi, "Image compression techniques: A survey in lossless and lossy algorithms," *Neurocomputing*, vol. 300, pp. 44–69, 2018.

[2] D. Le Gall, "Mpeg: A video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991.

[3] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[4] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1211–1220.

[5] F. Carpi, C. Häger, M. Martalò, R. Raheli, and H. D. Pfister, "Reinforcement learning for channel coding: Learned bit-flipping decoding," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2019, pp. 922–929.

[6] J. Dong, H. Qiu, Y. Li, *et al.*, "One-bit flip is all you need: When bit-flip attack meets model training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4688–4698.

[7] I. E. Richardson, *Video codec design: developing image and video compression systems*. John Wiley & Sons, 2002.

[8] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[9] R. Rivest, *Rfc1321: The md5 message-digest algorithm*, 1992.

[10] A. S. Parihar, D. Varshney, K. Pandya, and A. Aggarwal, "A comprehensive survey on video frame interpolation techniques," *The Visual Computer*, vol. 38, no. 1, pp. 295–319, 2022.

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[13] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[14] S. Ye, M. Ouaret, F. Dufaux, and T. Ebrahimi, "Hybrid spatial and temporal error concealment for distributed video coding," in *2008 IEEE International Conference on Multimedia and Expo*, IEEE, 2008, pp. 633–636.

[15] J. Koloda, J. Østergaard, S. H. Jensen, V. Sánchez, and A. M. Peinado, "Sequential error concealment for video/images by sparse linear prediction," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 957–969, 2013.

[16] M. Kazemi, M. Ghanbari, and S. Shirmohammadi, "A review of temporal video error concealment techniques and their suitability for hevc and vvc," *Multimedia Tools and Applications*, vol. 80, no. 8, pp. 12 685–12 730, 2021.

[17] A. Sankisa, A. Punjabi, and A. K. Katsaggelos, "Video error concealment using deep neural networks," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2018, pp. 380–384.

[18] C. Xiang, J. Xu, C. Yan, Q. Peng, and X. Wu, "Generative adversarial networks based error concealment for low resolution video," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 1827–1831.

[19] Y. Wang, K. Wu, W. Liu, K.-H. Yap, and L.-P. Chau, "Image representation and deep inception-attention for file-type and malware classification," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2023, pp. 1–5.

[20] W. Liu, Y. Wang, K.-H. Yap, and L.-P. Chau, "Bitstream-corrupted jpeg images are restorable: Two-stage compensation and alignment framework for image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9979–9988.

[21] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] G. Delétang, A. Ruoss, P.-A. Duquenne, *et al.*, "Language modeling is compression," *arXiv preprint arXiv:2309.10668*, 2023.

[23] J. Hoffmann, S. Borgeaud, A. Mensch, *et al.*, "Training compute-optimal large language models. arxiv 2022," *arXiv preprint arXiv:2203.15556*, vol. 10, 2022.

[24] J. Wei, Q. Liu, Y. Guo, and X. Jiang, "Training multilingual pre-trained language model with byte-level subwords," *arXiv preprint arXiv:2101.09469*, 2021.

[25] C. Wang, K. Cho, and J. Gu, "Neural machine translation with byte-level subwords," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 9154–9160.

[26] L. Xue, A. Barua, N. Constant, *et al.*, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 291–306, 2022.

[27] S. Wu, X. Tan, Z. Wang, R. Wang, X. Li, and M. Sun, "Beyond language models: Byte models are digital world simulators," *arXiv preprint arXiv:2402.19155*, 2024.

[28] Y. Wang, W. Liu, K. Wu, K.-H. Yap, and L.-P. Chau, "Intra-and inter-sector contextual information fusion with joint self-attention for file fragment classification," *Knowledge-Based Systems*, vol. 291, p. 111 565, 2024.

[29] T. Liu, K. Wu, Y. Wang, W. Liu, K.-H. Yap, and L.-P. Chau, "Bitstream-corrupted video recovery: A novel benchmark dataset and method," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[30] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1091–1095, 2007.

[31] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, Ieee, 2018, pp. 1–2.