Learning on entropy coded images with CNN

Rémi Piau, Thomas Maugey, Aline Roumy

▶ **To cite this version:**

Rémi Piau, Thomas Maugey, Aline Roumy. Learning on entropy coded images with CNN. ICASSP 2023 - IEEE International Conference on Acoustics, Speech and Signal Processing, Jun 2023, Rhodes, Greece. pp.1-5, 10.1109/ICASSP49357.2023.10095647 . hal-04133662

HAL Id: hal-04133662
https://hal.science/hal-04133662

Submitted on 20 Jun 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

LEARNING ON ENTROPY CODED IMAGES WITH CNN

Rémi Piau, Thomas Maugey, Aline Roumy

INRIA, Rennes, France
name.surname@inria.fr

ABSTRACT

We propose an empirical study to see whether learning with convolutional neural networks (CNNs) on entropy coded data is possible. First, we define spatial and semantic closeness, two key properties that we experimentally show to be necessary to guarantee the efficiency of the convolution. Then, we show that these properties are not satisfied by the data processed by an entropy coder. Despite this, our experimental results show that learning in such difficult conditions is still possible, and that the performance are far from a random guess. These results have been obtained thanks to the construction of CNN architectures designed for 1D data (one based on VGG, the other on ResNet). Finally, we propose some experiments that explain why CNN are still performing reasonably well on entropy coded data.

Index Terms— Image and Video Coding for Machine, Deep-learning, CNN, Entropy coding.

1. INTRODUCTION

There is now evidence that visual data will mostly be processed by machines [1]. To this end, the classical overall scheme consists in first compressing the data, and then decompressing them before being analyzed. Decompression before processing is a classical setup, as visual data analysis is usually performed on the 2D images, in the pixel domain. However, one can ask whether these two tasks, compression and analysis, can be efficiently combined.

There are several ways to achieve this. A common approach consists in designing a new codec optimized for both human and machine consumption [1, 2]. Note that it is then assumed that the task is known a priori (e.g., face coding [2]). But there are many cases, where the task is not known beforehand or video is recorded without any automatic analysis intent (e.g., TV recordings). Therefore, in this work, we assume that the video is compressed with an algorithm optimized for human vision, and that, *afterwards, an analysis is performed on the data*. But, as visualization is not needed when processing the data, we can wonder whether learning directly in the coded domain is possible without a need to decode.

Learning without decoding would however require analyzing the entropy coded stream directly. Indeed, entropy coding is ubiquitous in image and video standards such as JPEG [3], AVC [4], HEVC [5], and the more recent VVC [6]. Likewise, in deep learning based image compression [7], the latent representation provided by a variational autoencoder is entropy encoded, which leads to the compressed file. Learning on entropy coded data is difficult due to the loss of structure and variable length of this type of coding. Therefore, several contributions study learning on partially decoded data. In fact, learning on coded data has been studied under the condition that entropy decoding is performed. For instance, the analysis is performed either on the JPEG coefficients [8], or on the meta-data of the coded bitstream. For example, intra-coded prediction mode, inter-coded frame block mode, and block size can be used to detect scene changes [9]. Other methods, use both coefficients and metadata to perform analysis. For instance, intra-coded frame DCT coefficients and inter-coded frames motion vectors are used for video saliency detection in [10]. Similarly, the fusion of some fully decoded frames and entropy decoded motion vectors information allows CNN to achieve high-speed video classification [11]. More recently, a real-time video segmentation method that uses classical segmentation on intra-coded frames while computing inter-coded frame segmentation from both motion vectors and an embedding of last frame segmentation was proposed in [12]. Another method [13] use codec specific knowledge to align encoded JPEG block which can be done by parsing the file without decoding per se.

However, all these methods still require entropy decoding or at least parsing and as such, loose the benefits of analyzing the bitstream directly. In this paper, we rather propose to study whether CNN based learning directly on entropy coded data is possible. The question is of scientific interest, as it would allow one to measure the impact of the variable length data format on the analysis accuracy. Besides, learning on the coded domain would provide many potential advantages: first decoding is avoided, and second, since the data have a compact representation, a faster processing can be expected. Indeed, modern image/video codecs, entropy coding is the final step that really reduces the size of the data. All previous steps such as transforms, prediction, mostly prepare the entropy coding step by exploiting correlation between data. But the number of coefficients during these previous steps remains the same or even increases, due to the use of metadata.

We first study the convolution, a key step in CNN, and identify two properties, that data format should satisfy. More precisely, to be processed by a convolution, the data fed to a CNN should be spatially but also semantically close. We then confirm these statements experimentally. Subsequently, we explain why and how the classical processing of entropy coding does not preserve these properties. For the sake of the study, we introduce two architectures, designed to process 1D vector data, with which we learn on both arithmetic and Huffman coded data. Surprisingly, we observe that analysis with CNN on entropy coded data is still possible, and that the accuracy are even far from a random guess. Finally, we propose to explain such good CNN performance with complementary experiments.

2. PROBLEM DEFINITION

We want to evaluate if a CNN can directly learn in the coded domain. To do so, we first present the joint coding and learning scheme, and compare the different setups: *(a) classical*, when learning is performed on the decoded data, and *(b) in the coded domain*, when learning is done directly on entropy coded data (see Fig. 1).

An image X is stored and transmitted in a compressed form X'.

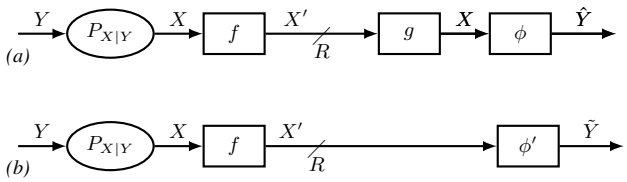

Fig. 1. Classical learning (a) vs learning in the coded domain (b).

The function f refers to the entropy coder that maps X to its compact description X'. In particular, f is controlled by a frequency table that determines the output length of each input symbol. The decoder g retrieves the image X from X'. Since we study entropy coding, the encoder decoder pair is lossless i.e., $g \circ f = Id$. To evaluate compression we define the mean compression ratio as:

$$\overline{r(X)} = \frac{\overline{|X|}}{\overline{|f(X)|}} \tag{1}$$

where $\overline{|X|}$ stands for the average length of X ($|.|$ the length in bits), computed over a set of images.

We denote Y the label associated with the image X, and model the relationship between Y and X as a probability distribution $P_{X|Y}$, as shown in Fig. 1.

In the classical setting (see Fig. 1*(a)*), a learning algorithm takes as input the image X, i.e., after the decoding step g. In this case, the learner determines the estimator ϕ, by solving the following optimization problem, with D a pairwise distance and T_{train} the training data:

$$\min_{\phi} \sum_{(X,Y)\in T_{\mathsf{train}}} D(\phi \circ g \circ f(X), Y). \tag{2}$$

In the "learning in the coded domain" setup, the label Y is estimated from the coded data X', as shown in Fig. 1*(b)*. More precisely, the learner determines ϕ', and solves the optimization problem:

$$\min_{\phi'} \sum_{(X,Y)\in T_{\mathsf{train}}} D(\phi' \circ f(X), Y). \tag{3}$$

Note that the estimators ϕ and ϕ' differ through their input spaces.

Since f is zero-error, we do not lose any information by encoding X. Thus, at first sight, we may expect the same performances for both learning algorithms: classical and in the coded domain. However, as stated in the next two sections, there are some incompatibilities between the data format needed to insure good performance of a learning algorithm, and the format of the data after entropy coding. The next section starts with a review of two key properties for CNNs that are not preserved after entropy coding. Which might cause some performance degradation, when learning is performed in the compressed domain.

3. LEARNING WITH CNN: REQUIREMENTS

3.1. Required properties: spatial and semantic closeness

When using CNN, the main tool of the feature extraction layers is convolution. A key ingredient of the convolution is a weighted average function performed on a neighbourhood, at a given position of a fixed and regular grid. Therefore, the format of the data fed to a CNN should satisfy the following properties:

Properties lost	**Accuracy**
None	0.62098
(i)	0.38089 ± 0.00505
(ii)	0.24406 ± 0.01656
(i) and (ii)	0.12392 ± 0.02151

Table 1: VGG11 network mean accuracy on YCIFAR-10 when breaking closeness properties via random permutations (95% confidence interval).

- (i) *Spatial closeness*: two neighboring pixels on the pixel grid should, in the representation, be placed at adjacent and identifiable positions,
- (ii) *Semantic closeness*: neighbor data cell values (pixel values) that are close in terms of meaning (e.g., close in terms of perceived physical color) should be close in terms of representation (integer color value).

In order to check the impact of these properties, we introduce lossless perturbations that alter the aforementioned properties:

- A permutation on positions that shuffles the pixel's location (i.e., we lose spatial closeness (i)).
- A permutation on values in this case the pixel value range ($\{0, \dots 255\}$) (i.e., we lose semantic closeness (ii)).

Next, we apply these permutations on the data and perform learning on the permuted data. Table 1 shows the accuracy of the classification network VGG11 [14] on the uncoded greyscale YCIFAR-10 dataset (see 4.2).

Interestingly, the permutation on positions, which breaks the spatial closeness property (i), and the permutation on values, which breaks the semantic closeness property (ii), both create a drop in learning accuracy. Moreover, applying both permutations at the same time leads to a total learning failure as accuracy (0.12) becomes close to a random guess $(0.1$, for 10 classes).

Although CNNs have demonstrated to be a powerful tool, we show here that some properties on the input data have to be respected for them to work well.

3.2. The challenge to learn on entropy coded data

Entropy coding modifies significantly the structure of the input data for two reasons. First, the encoder maps a vector of symbols to an encoded representation. So, within a vector, no clear delimitation between encoded pixels are present. Therefore, one can not determine pixels representation boundaries, and decide whether two pixels are neighbors, unless decoding is performed. Second, the encoded representation (of a vector) is variable length. Therefore, the boundaries between encoded vectors is not fixed, and can only be retrieved upon decoding. As a consequence, we loose the spatial closeness property (i) needed for convolution to work well.

Moreover, during encoding, we also obfuscate the individual values of each pixel. Indeed, the coded output of a given pixel depends on its occurrence frequency thus two similar pixels may be mapped to wildly different outputs thus breaking the semantic closeness property (ii). For example, two different shades of red, one occurring frequently and one sparsely occurring, may be close in the pixel domain but will be mapped to strongly different outputs by the entropy coder as it only takes into account the frequencies, not the semantic.

For all these reasons, we conclude that learning with CNN on entropy coded data is *a priori* a difficult (nearly impossible) task,

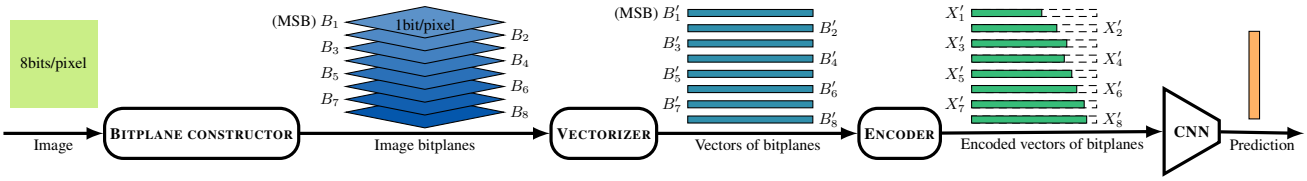

Fig. 2. Experimental pipeline for learning on coded data.

since the entropy coded format does not satisfy the key properties (i) and (ii).

4. EXPERIMENTAL SETUP

In this section, we present our experimental pipeline for learning on entropy coded data. We will also present some intermediary results that validate key choices. Our experimental pipeline has the following outline (see Fig. 2). Using grey level images (luminance) with 8 bits/pixel (see Section 4.2) as input data, the *bitplane constructor* first splits the images into bitplanes. Then the *vectorizer* transforms the square bitplanes of the image into 1D vectors. Afterwards, the *encoder* encodes each bitplane vector separately. More precisely, there is one coder per bitplane with its associated frequency table computed over the training set bitplanes. Finally, after padding encoded vectors to fixed length, the *CNN* can learn on them.

During the training phase, one wants to optimize the weights of the network such as it minimizes the empirical risk. The network takes as input a tensor in $\mathbb{R}^{B_{\text{in}} \times C_{\text{in}} \times O_{\text{in}}}$ where B_{in} is the batch size, C_{in} the number of channel that is the number of bitplanes, O_{in} the original size of a decoded bitplane vector (as encoded vectors are padded to this length).

Meanwhile, the output of the network is a vector of probability of length C, with C being the number of classes we want to distinguish. Each the value of each coordinate $i \in \{0, \dots C\}$ is the probability of the input being in classified as in class i by the network. For classification one can get the class as the indices of the max coordinate of the output vector (argmax) as we have mutually exclusive classes in our dataset.

To optimize the weights of our network we use backpropagation with the cross entropy loss. Moreover, we use a validation set: a 10% subset of our training set to avoid over-fitting. This validation subset is thus not used during backpropagation. Lastly, we keep the model with the lowest validation score as the optimal model and use it for testing. We train all networks using a stochastic gradient descent (ADAM) [15] with a learning rate of 10^{-4}.

4.1. Unidimensional VGG & Resnet

We choose a commonly used architectures that represent well convolutional networks successfully used for 2D image classification:

- VGG [14] that is composed of 2 parts: first an interleaved convolutional and maxpooling layer, second a fully connected layer to reduce extracted features to label,
- RESNET [16] a well performing widely used convolutional network with skip layer connections.

Our encoded data is of variable size and cannot be meaningfully put back in the form of a matrix as vertical correlations may not be preserved as rows alignments and boundaries are lost during coding.

We want similar networks to work on 1D data instead of 2D images. As such we re-implement VGG11 and Resnet18 using only 1D convolutions. This gives rise to the Unidimensional-VGG11 (**UVGG11**) and Unidimensional-RESNET18 (**URESNET18**) networks. As input of these networks there is C_{in} bitplanes. That is C_{in} 1D vectors fed into a channel each.

4.2. Input data: greyscale matrix dataset

For our experiments, we consider three 8 bits greyscale datasets with growing complexity. The first dataset is MNIST [17]. Its images are handwritten digits on a uniform background. It thus has 10 classes.

The second one is Fashion-MNIST [18] where the images depict 10 different types of clothing on a uniform background.

The last dataset we use is an adaptation of CIFAR-10 [19]. As we learn on greyscale images in these experiments CIFAR-10 is transformed into *YCIFAR-10* by converting the images to greyscale (full swing BT.601 RGB to YUV conversion [20]). Images of YCIFAR-10 are more complex than Fashion-MNIST ones but also possess a non-uniform background, making class discrimination more complex.

4.3. Coding bitplanes

Before tackling learning (on coded data or not) we want to briefly discuss the effect of coding on each bitplane. In Table 2 we show the mean compression ratio $\overline{r}$ for each bitplane p for the entropy coded YCIFAR-10 dataset. We can see that most significant bits are more correlated and as such their bitplane is more compressed (higher ratio). The least significant the bit position is, the least its value is correlated with its neighbors thus the lower its bitplane compression is. This holds true for all coding methods and dataset used in this study. This method of compression leads to better compression efficiency while further breaking the spatial closeness property (i) between bitplanes.

The compression ratio for Fashion-MNIST and MNIST are respectively higher and much higher. Indeed, the compression ratio increases as the complexity of the database decreases.

On all datasets, Huffman coding leads to relatively smaller compression ratios compared to arithmetic coding. This is expected as by construction, Huffman coding [21, Chap. 3] cannot account for previously encoded data by itself whereas arithmetic coding can [21, Chap. 4].

Bitplane	$\overline{r}$	**Bitplane**	$\overline{r}$
(MSB) 1	1.9884	5	1.0256
2	1.4095	6	1.0085
3	1.1732	7	1.0030
4	1.0695	(LSB) 8	1.0016

Table 2: Example of mean compression ratio per bitplane for arithmetic coded YCIFAR-10.

		Coding Type			
Dataset	Network	None	Huffman	Arithmetic	JPEG
MNIST	UVGG11	0.98911	0.83234	0.63130	-
	URESNET18	0.98753	0.74503	0.59498	-
Fashion-MNIST	UVGG11	0.90189	0.76347	0.68987	-
	URESNET18	0.84972	0.68620	0.61162	-
YCIFAR-10	UVGG11	0.56573	0.36062	0.29762	0.32459
	URESNET18	0.38368	0.25913	0.24325	-

Table 4: 1D-Network accuracy on entropy coded data.

	Network Type			
	VGG11		RESNET18	
Dataset	2D	1D+Bitplanes	2D	1D+Bitplanes
MNIST	0.99188	0.98911	0.99328	0.98753
Fashion-MNIST	0.90433	0.90189	0.89373	0.84972
YCIFAR-10	0.66489	0.56573	0.53206	0.38368

Table 3: Accuracy of 2D Networks and binary 1D Networks.

	Accuracy Increase
Huffman	0.05663
JPEG to JPEG-DC	0.00956

Table 5: VGG11 network accuracy increase when learning on YCIFAR-10 with coded data aligned on 2D grid compared the accuracy of 1D unaligned data of Table 4.

4.4. From 2D networks with pixel input to 1D networks with binary input

In this section, we evaluate the impact of representing a 2D grid of pixels by a set of 1D binary vectors. We perform experiments without any entropy coding. Binarization and above all vectorization modify the spatial closeness property, such that the accuracy of the classification could decrease. Interestingly, we observe in Table 3 that the accuracy remain quite stable. This might be explained by the efficiency of the pooling/unpooling, that still succeeds into extracting information positioned further away.

One might think that using larger kernel for the first layer of the network would capture enough bits as once to capture several coded symbols, but our testing shows that it just lead to a slight accuracy degradation.

5. EXPERIMENTAL RESULTS AND ANALYSIS

Our main experimental results can be found in Table 4. In this Table, we compare the accuracy achieved by three 1D-networks, that differ through the representation of the input data. In the first column, no entropy coding is performed (i.e, as if the learning is done after decoding), whereas in the second and third columns, the input data are compressed with a Huffman and an Arithmetic encoder, respectively and not decoded before learning.

Interestingly, despite the fact that the two key properties of the convolution (spatial (i) and semantic (ii) closeness) are not preserved, we can observe that the CNN can still learn on entropy coded data. Indeed, the accuracy is far from a random guess (which is 10% since, in all datasets used here, there are 10 classes). We also observe that it is easier to differentiate between Huffman encoded data than with arithmetic encoded data. This might be explained by the fact that, in our implementation, Huffman processes data of length 8, whereas the arithmetic processes a bitplane of length equals to the whole image. Therefore, and for the reasons explained in Sec.3.2, the Huffman encoder better preserves the spatial closeness property than the Arithmetic encoder. We use the best performing 1D-network (UVGG11) on the hardest dataset considered (YCIFAR-10), to learn directly on the bitstream of JPEG coded images. It is less accurate than learning on Huffman coded data but more than learning on arithmetic coded data for the same dataset. This stems from the fact a JPEG coder first performs a transform, which further breaks the spatial closeness property (i), and then apply an entropy coder, similar to Huffman coding. Therefore, the accuracy decreases with respect to Huffman coding only. However, learning on JPEG encoded data still outperforms learning on Arithmetic encoded data.

We can regain accuracy by aligning coded data on the image grid by aligning each coded pixel on the 2D grid using one channel per coded pixel bit. Arithmetic coded data is not separable per symbol but Huffman coded data is. Although one need to separate and align symbols during encoding as symbol boundaries are lost when coding. On the other hand JPEG coded DC coefficients can easily be extracted from file with a simple parsing and their order being fixed by the standard make them easy to map onto a grid. Table 5 contains the results of such experiments using UVGG11 on YCIFAR-10. The observed accuracy for aligned Huffman coded data aligned on 2D grid is greater than the one from 1D unaligned Huffman coded data. This is expected as we are recovering the spatial closeness property (i). But the increase is not that great, showing again that entropy coding does not disturb spatial closeness (i) as much as we though. Learning on only aligned JPEG DC coefficients on a 2D grid leads to a slightly better accuracy than learning on JPEG as a 1D bitstream. This can be explained as we both recover some spatial closeness (i) but as the same time we lose data as we are now working on a grid downscaled by 8 compared to the size of the original image. In [13] they try to learn on the whole JPEG coded block (DC+AC coefficients) after aligning them on the grid. This approach also suffers from an accuracy loss of about 15% compared to the baseline which is in line with what we obtain here.

6. CONCLUSION

In this paper, we introduced two data properties necessary for CNN to work well and validated them experimentally. Indeed, as entropy coded data lose these properties, we studied the impact of learning with such coded data onto the classification accuracy of a CNN network. For this purpose, 1D-binary input CNN have been designed. Surprisingly, even if the structure of the data processed by the network have been significantly modified, the CNN can still learn on these data. Entropy coding does not impact these properties as much as we though proving learning on entropy coded data feasible.

7. REFERENCES

[1] Lingyu Duan, Jiaying Liu, Wenhan Yang, Tiejun Huang, and Wen Gao, "Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics," *IEEE Transactions on Image Processing*, vol. 29, pp. 8680–8695, 2020.

[2] Shuai Yang, Yueyu Hu, Wenhan Yang, Ling-Yu Duan, and Jiaying Liu, "Towards Coding for Human and Machine Vision: Scalable Face Image Coding," *IEEE Transactions on Multimedia*, vol. 23, pp. 2957–2971, 2021.

[3] G.K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.

[4] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[5] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[6] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm, "Overview of the Versatile Video Coding (VVC) Standard and its Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, Oct. 2021.

[7] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations*, Feb. 2022.

[8] David Edmundson and Gerald Schaefer, "An overview and evaluation of JPEG compressed domain retrieval techniques," in *Proceedings ELMAR-2012*, Sept. 2012, pp. 75–78.

[9] Yumi Eom, Sangil Park, Sunggeun Yoo, Jin Soo Choi, and Sukhee Cho, "An analysis of scene change detection in HEVC bitstream," in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, Feb. 2015, pp. 470–474.

[10] Yuming Fang, Weisi Lin, Zhenzhong Chen, Chia-Ming Tsai, and Chia-Wen Lin, "A Video Saliency Detection Model in Compressed Domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 27–38, Jan. 2014.

[11] Aaron Chadha, Alhabib Abbas, and Yiannis Andreopoulos, "Video Classification With CNNs: Using the Codec as a Spatio-Temporal Activity Sensor," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 475–485, Feb. 2019.

[12] Junyi Feng, Songyuan Li, Xi Li, Fei Wu, Qi Tian, Ming-Hsuan Yang, and Haibin Ling, "TapLab: A Fast Framework for Semantic Video Segmentation Tapping Into Compressed-Domain Knowledge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1591–1603, Mar. 2022.

[13] P. R. Hill and D. R. Bull, "Transform and Bitstream Domain Image Classification," Oct. 2021.

[14] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[15] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 2014.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[17] Yann LeCun, "THE MNIST DATABASE of handwritten digits," https://yann.lecun.com/exdb/mnist/.

[18] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," Sept. 2017.

[19] Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Tech. Rep., University of Toronto, Apr. 2009.

[20] Radio Communication Sector of ITU, "RECOMMENDATION ITU-R BT.601-7 – Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios," Mar. 2011.

[21] Khalid Sayood, *Introduction to Data Compression*, Elsevier, 2018.