THE (NON)SENSE OF ARTIFICIAL INTELLIGENCE IN REAL-TIME VIDEO ENCODING

J. De Cock

Synamedia, Belgium

ABSTRACT

Thousands of articles have appeared over the last decade(s) claiming the benefits of AI and ML. Some of these are realistic, while others overpromise the potential benefits of ML techniques in a wide range of applications. So where are the real benefits, and where do our marketing departments cross the line into the realm of nonsense? And what about applications under stringent conditions, such as real-time (video) processing or encoding? How much of the net benefits do you keep once you take into account the computational overhead, latency and cost constraints?

This article takes a stab at distinguishing reality from fiction, where we see the benefits, and how the tools provided by AI enhance our encoder toolbox. After settling on AI/ML terminology, we give an overview of the state-of-the-art in the domain of video encoding. Next, we present examples and counterexamples of techniques that have worked, along with approaches that benefit our line-up of encoders.

INTRODUCTION

It is not my intention in this article to repeat a comprehensive overview of artificial intelligence, machine learning, or (deep) neural networks. Excellent books and articles have been written in this context e.g. by ‘Sze et al (1)’ and ‘Ding et al (2)’, and new papers are published faster than they can be read.

First of all, it’s helpful to get the terminology right. AI and ML are often used interchangeably. To avoid any confusion, we repeat the hierarchy of classes within *Artificial Intelligence* (AI). AI is the science and engineering of creating intelligent machines/programs. Within AI, *machine learning* (ML) has been defined as the field of study that gives computers the ability to learn without being explicitly programmed.

Many different types of ML algorithms are available, each having their merits, including Decision Trees, Random Forest and Support Vector Machines. Given their performance on a wide variety of tasks, the focus has shifted towards *brain-inspired* ML techniques, in which neurons are used that can take signals as input, perform a computation on those signals, and generate an output signal.

Neural networks (NNs) are the most common type of brain-inspired ML techniques, in which a neuron’s computation involves a weighted sum of the input values, after which a non-linear function is applied that generates an output only if the input crosses some threshold. Within the domain of neural networks, we find *deep learning*, in which the NNs

have more than 3 layers, in which at least one layer is hidden. These so-called *deep neural networks* (DNNs) have become increasingly popular, and often contain dozens or even hundreds of (convolutional) layers.

In general, we have seen an evolution from shallow networks using hand-crafted features as inputs towards “black box” deep neural networks with a multitude of convolutional filter layers. Especially for computer vision, using several convolutional layers seems to be very effective. It is tempting to expect the same evolution for video compression.

THE COMPLEXITY OF DEEP LEARNING NETWORKS

To set the expectations: despite all their potential, applying DNNs to video encoding, and hoping for magical jumps in compression performance to come out, simply does not happen. Either the resulting networks are too complex and require a lot of computational power – raising the cost of your solution; or they become very shallow, and the accuracy goes down. In that case, it might be more beneficial to stick with manually tailored heuristics in your established encoders. Hardware acceleration can work (but also raising the cost), but you lose the flexibility in deployment of your solution.

In literature, the complexity of DNNs keeps increasing. Fortunately, more and more publications are focusing on, or at least mentioning, the complexity of ML. So, what is a good measure of the complexity of machine learning? A reference point that is used more and more is to express the complexity of these architectures by the number of multiply-and-accumulate operations (MACs). These MACs can range from thousands to millions per second and are a good representation of the complexity (1). Even after hardware parallelization, a MAC remains a costly operation.

As an example, state-of-the-art CNNs that have been submitted to the ImageNet challenge are listed in Table 1, ranging from LeNet to ResNet and VGG, mentioning the number of layers and total number of MACs needed. For the more recent networks, operation counts are typically expressed in GigaOps, highlighting the explosion in computational power needed.

	LeNet	AlexNet	GoogLeNet	ResNet-50	VGG-16
# of convolutional layers	2	5	57	53	13
Total weights	60k	61M	7M	25.5M	138M
Total MACs	341k	724M	1.43G	3.9G	15.5G

Table 1 – Overview of popular DNNs

Even on powerful (and expensive) GPUs, inference time for these networks can go up to more than 100 ms for a single 1024x1024 image. Also, for these more recent networks, accuracy per parameter goes down, and the more oversized networks do not fully take advantage of their learning ability (‘Canziani et al (3)’). Although this just gives an indication of the complexity of modern-day DNNs, how much can video compression benefit from such deep networks?

EVOLUTION AND COMPUTATIONAL COMPLEXITY OF VIDEO CODING

Video coding has come a long way since the first digital compression systems in the 80s. From H.261 over MPEG-2 and H.263, we're now in a multi-codec world of several compelling formats, including H.264/AVC, HEVC, VP9, AV1 and VVC (See e.g. 'DeCock (4)'). Typically, the introduction of video standards has gone hand-in-hand with increases in resolution. In particular in the broadcast world, there is a close connection between the next step in resolution and new standards (e.g. H.264/AVC and HD, HEVC and UHD). The introduction of VVC is expected to go hand-in-hand with the adoption of 8K.

Along with compression efficiency improvements, the complexity in video encoders and decoders has grown substantially. For real-time encoding, there are increasing challenges to process high-frame, high-resolution video in real-time with the latest standards. In real-life systems, such as in broadcast or live ABR distribution, the cost of the system is a very important parameter to remain competitive. In general, AI techniques can bring benefits to make encoder providers more competitive. But, as seen above, the complexity of many networks is prohibitive to put into practice, leading to net disadvantages when applied to encoding. It is this trade-off that we'll be exploring in the remainder of this paper, excluding some approaches from consideration, while discussing others that look promising. But as announced in the introduction, there is a gap between marketing promise and reality.

MACHINE LEARNING IN VIDEO CODING

Machine learning has been applied in image and video coding for multiple decades. In the late 80s, experiments were already ongoing with image compression based on multilayer perceptron (MLP) networks ('Cottrell et al (5)'). In the 90s, work shifted towards random neural networks for image ('Marsi et al (6)', 'Gelenbe and Sungur (7)') and video ('Cramer et al (8)') compression. These publications focused on alternatives to then-current compression schemes such as JPEG and H.261, with comparable computational complexity. Although they offered reasonable performance, they never made it into the field and the focus shifted to standardized encoding solutions such as MPEG-2 and H.264/AVC.

Since the early 2010s, machine learning work related to video coding has revived, and truly accelerated over the last years, leading to what you could describe as "deep schemes", "deep tools" and "standard-compliant" work. It is almost impossible to provide an exhaustive overview of the work in this direction, as each of these topics are extremely hot research topics with new publications appearing daily.

Deep schemes

Deep compression schemes are a radically different approach for encoding, providing a complete end-to-end solution. They build on the concept of dimensionality reduction and auto-encoding, enabling automated learning and eliminating the need for hand-crafted features ('Hinton et al (9)'). These auto-encoders have received growing attention over the last 5 years, with very promising results, especially in terms of perceptual quality. In 'Chen et al (10)', a CNN-based video compression framework (DeepCoder) was presented, with separate CNN networks for predictive and residual signals, after which scalar quantization and Huffman coding are applied. The authors obtained similar efficiency as x264 IPPP coding in terms of SSIM. No results on its computational requirements were mentioned though. Other work by 'Ballé et al (11)' has focused on end-to-end optimization and image compression, with potential to outperform formats such as JPEG and JPEG 2000 in terms

of compression performance. ‘Chen et al (12)’ focuses on CNNs to model spatio-temporal coherence to perform predictive coding, approaching the compression performance of H.264/AVC. But, overall computational complexity was reported to be 141x that of the H.264/AVC reference software (JM). ‘Lu et al (13)’ presented an end-to-end deep video compression network, with a complexity of about 11M parameters, reaching about 24 fps for CIF-sized videos.

In (2), an overview is given of end-to-end neural video coding solutions, along with case studies. It is clear though that these “deep schemes” are not ready for prime time in real-time video, and that they lack a standardized solution before they would be widely deployed.

Deep tools

Closer to existing standards is the work on so-called “deep tools” or “modularized neural video coding” (2), in which encoder tools are replaced by learned algorithms. Encoder tools are traditionally hand-designed and tweaked by engineers active in the standardization process. By learning, solutions can be found that can better adjust to the type of content, or that better exploit spatio-temporal coherence.

Over the last years, we’ve seen learned algorithm proposals for intra prediction, e.g. ‘Cui et al (14)’, where small gains were reported compared to HEVC reference software. ‘Li et al (15)’ mentions 1.1% bitrate savings at a cost of 148% and 290% relative to the HM encoding and decoding software, respectively. Complementary HEVC intra prediction modes were proposed in ‘Pfaff et al (16)’, which explicitly mentions the high cost in terms of multiplications, which grows with the block size (from 20 multiplications per pixel for 4x4 blocks to 132 multiplications per pixel for 32x32 blocks).

For temporal prediction, ‘Liu et al (17)’ present a fractional interpolation method based on a grouped variation convolutional neural network (GVCNN). This gives bitrate savings of 2.2% on average, but with encoder/decoder times of 6x and 1500x, again relative to reference software. Alternative schemes have been proposed, e.g., to predict texture without sending motion information between encoder and decoder (‘Choi and Bajić (18)’). Encoder time is 1.5x, while decoder time is more than 100x.

Standardization efforts

Currently, standardization efforts are on their way to bring ML tools into real-life. An activity in JPEG has been started (JPEG AI) to kick off explorations in the direction of learning-based encoding. Also, in MPEG/JVET there is a tendency towards using AI in video compression schemes. In the past JVET meetings, several contributions have been submitted in this direction, and an AHG was established with the goal of developing a potential VVC extension supporting learning-based coding tools (‘Liu et al (19)’). Tools that have been discussed in this AHG include intra prediction, in-loop filtering, post-processing, and super-resolution. Similar work is ongoing within the Alliance for Open Media, for improvements in the context of AV1 and AV2.

On the downside, many of these tools are again very complex. On the positive side, attention is paid to the complexity by optimizing the networks. In the JVET AHG, it is a requirement to mention the number of MACs per pixel. For many of these tools though, decoding time compared to VVC reference software goes up by two or three orders of magnitude, and more work is needed to reduce complexity to acceptable levels.

Work compliant with existing standards

Part of the more recent research has focused on improving and accelerating video encoding in a normative way (i.e., without modifying existing standards or tools). Starting with the introduction of HEVC, plenty of research can be found that tries to reduce encoder complexity. As an example, “data mining” approaches were applied, resulting in decision trees to reduce computational complexity of HEVC encoding (‘Correa et al (20)’). In ‘Xu et al (21)’, both a CNN-based and LSTM-based model are introduced to accelerate quadtree partitioning for intra and inter coding in HEVC. Still, the first model needs a total of 1.5M multiplications and additions, while the second requires about 760K. Although these models lead to reductions of 60-70% of encoding time compared to reference software, they are orders of magnitude too high for practical implementations.

Recently, worked has shifted towards acceleration of VP9 (‘Paul et al (22)’), AV1 (‘Chiang et al (23)’) and VVC (‘Tang et al (24)’, ‘Zhao et al (25)’) encoding. The smallest model in (22) used about 26K trainable parameters, requiring about 10M floating point operations per 64x64 superblock. While this results in speed-ups compared to VOD-type-settings, it is much harder to justify this number of operations for real-time encoding.

Pre- and post-processing

DNNs have been shown to be extremely powerful for (pre-)filtering applications. A range of publications have been written on pre-filtering techniques including sharpening, denoising, contrast enhancement, motion deblurring, and edge detection. A different approach focuses on video content semantics to assist in video encoding, such as object detection / segmentation, saliency prediction etc. More forward-looking are analysis-synthesis techniques that are more closely aligned to how e.g. texture is perceived by the human visual system. A texture-based video coding approach is presented in (2), along with its open issues, such as the accuracy of the analysis.

A promising direction for some time has been on super-resolution, where input videos are downsampled before encoding, after which the reconstructed frames are upsampled again at the receiver. These approaches can save bits at acceptable complexity requirements (see ‘Yang et al (26)’ for an overview). The benefit of these approaches is that they can work in a standard-compliant way, without modifications to the underlying encoders or infrastructure. A downside is that the super-resolution upscaling algorithms need to be applied at the receiver side, making not all (legacy) clients suitable for this type of distribution.

Complexity considerations

As stated in (2), “All of these issues present serious barriers to the market adoption of DNN-based tools, particularly on energy-efficient mobile platforms. One promising solution is to design specialized hardware for the acceleration of DNN algorithms”. And according to ‘Liu et al (27)’, “Comparing the existing deep tools with their counterparts in the traditional non-deep schemes, one may easily notice that the computational complexity of the former is much higher than the latter. High complexity is indeed a general issue of deep learning, and a critical issue that hinders the adoption of deep networks in scenarios of limited computing resource, e.g. mobile phones.”

Despite all their potential, we’re still far off from applying these schemes and tools in real-time encoders. While plenty of research and engineering has been spent on accelerating *traditional decision making in encoders*, there is a lot of work needed towards accelerating

DNN-based encoder decision making. Throwing in a GPU or parallel hardware does not resolve the situation, since it results in a very expensive solution, which is a multiple of currently deployed software on generic CPUs. It is a good evolution that in more and more publications the complexity of networks is explicitly mentioned, seeing how far we are still away from practical deployment for many networks.

SO WHY ISN'T THIS WORKING (YET)?

Theory vs practice

For most of the publications described above, significant improvements were reported, with impressive speed gains compared to the open-source *reference* software. As we all know though, these reference code bases are far from optimized, with speeds expressed in "seconds per frame" rather than "frames per second". It is easy to demonstrate speed-ups relative to these code bases, but it becomes way harder when compared to optimized encoders that have been tuned by experts, optimized with intrinsics and that operate at high frame rates and resolutions in real-time.

Arguably, it makes sense that academic publications focus on comparison with reference software, since these are established points of reference, as they have been for years. Still, the danger of this practice is that the complexity of the reference software and non-real-time encoders hides the complexity of the machine learning networks used in these papers, hence over-promising their potential. As a result, applying these as such to practical encoders leads to limited gains. Techniques to simplify and optimize encoders are well-known in the industry. Accelerating repetitive MAC operations in DNNs on the other hand is limited by capabilities of hardware, parallelization options and memory access. Even though accelerators are available, more time and effort need to be spent on the (co-)design of smart networks for real-time processing.

Constraints of real-time video

It is important to make a distinction between *training* effort and *inference* effort. It is primarily the latter that we're focusing on, while the training phase can be orders of magnitude more expensive. The focus and target in this paper is on *real-time* inference for video encoding and processing, that is widely deployable, preferably on generic CPUs. In that way, you can reach deployment on any platform, including on the widest selection of cloud instances.

There is of course a difference between offline encoding for VOD services, and encoding for live video (broadcast, live events, live ABR, web conferencing for example). For the former, more effort and compute cycles can be spent to encode and prepare episodes and movies. This is an encode-once, decode-millions-of-times scenario, and it is ok to spend hours per encode if necessary.

For real-time encoding, millions of decisions have to be made every second, and compute capacity is limited, leading to optimization in a three-dimensional rate-quality-complexity space. For live deployments, you also want to limit the financial cost. Any increase in complexity has to be justified – adding a GPU to the system can quickly escalate the cost of your server and hurt your competitiveness. HW/SW interaction can also complicate interaction and transfers between processes, and complicate the implementation. Moving purely to hardware can limit the flexibility and features that can be supported in your video pipeline. With HW implementations, there is limited space for VQ improvements and

innovation cycles slow down. All of these will influence the decision to move towards (partial) HW implementations, or to stay fully focused on software.

ACCELERATING NEURAL NETWORKS

It's clear that the networks described above cannot be applied as such as in real-time systems with acceptable cost. What has been done to simplify NNs so far, or to make their potential more accessible?

Hardware acceleration

Fortunately, we have seen several efforts to enable inference in real-time on deep learning accelerators, e.g. research chips such as Stanford's EIE and TETRIS chips; the work by Chen et al. at MIT on the Eyeriss reconfigurable DNN accelerator; and the DianNao series of research chips. 'Reuther et al (28)' give an overview with a distinction between research chips, very low power chips, embedded chips, data center chips and autonomous systems currently in development or use.

Although these accelerators will certainly help in dedicated use cases, we want to deploy our encoders as broadly as possible. For common CPUs, approaches such as TensorFlow Lite can also help in efficiently deploying models, as can efficient instruction sets such as VNNI instructions on AVX-512.

Co-design

In general, co-design of DNN models and hardware (1) will help in making these models more accessible, by using a combination of the following:

- *Network quantization*. In many cases, floating-point accuracy is overkill for DNNs. By reducing precision of operations and operands, and by representing floating-point weights and/or activations with fewer bits, network calculations can be accelerated. A trade-off then needs to be found between acceleration and network accuracy. This is in line with what e.g. Intel is doing by providing conversion software from 32-bit floating point precision. Academic work in this direction is found in e.g. 'Zhang et al (29)', where a method was proposed to quantize both the weights and activations to arbitrary bit-widths. (1) summarizes different techniques to reduce prediction along with accuracy loss compared to 32-bit float operations.
- *Compact network design* can be achieved by reducing the number of operations and model size. This can be done by exploiting activation statistics (for zero or low-valued activations), or by network pruning (setting redundant weights to zero). The latter builds on early work in 'Le Cun et al (30)'. More recent applications are described in e.g. 'Li et al (31)'.

TinyML

The co-design described in the previous section is aligned with the goals of "TinyML", a field which targets hardware, algorithms and software capable of performing on-device (sensor) data analytics at extremely low power. As described by 'Verhelst and Moons (32)', "the combination of pruning, weight sharing, and Huffman compression compresses state-of-the-art networks by 50 times in memory size". It is indeed necessary to reduce the complexity of deep networks by several orders of magnitude to allow their implementation on low-power devices.

In the end, the complexity that we're searching for in video encoders, is similar to networks that this research is trying to accomplish on low-power devices. Within a video encoder, many networks can run in parallel to make thousands of decisions per second. Each of these networks is allowed to only occupy a small part of the overall power of the CPU.

Interpretability: back to feature engineering

In the beginning years of machine learning, research focused on features to feed into neural networks. Later, black-box-type models were introduced that gave more flexibility, but also higher complexity. Recently, the ability to interpret what a model has learned is receiving an increasing amount of attention ('Murdoch et al (33)'). Among the interpretability methods, domain-based and model-based feature engineering are presented. Domain/expert knowledge and insights help to identify features that are domain specific and help interpretation,

Recent papers have focused on interpretability of CNNs for video coding ('Murn et al (34)'). This approach tries to reduce the complexity of CNNs by interpreting the learned parameters to build simpler models. As stated in (34), interpreting and understanding relationships learned by the network enables the derivation of streamlined, less complex algorithms which achieve simpler performance to the original models.

SO, WHAT DOES WORK?

Fortunately, it is not all bad news, and the trend toward simplification is leading to good examples of what can be achieved with ML, in a trade-off between accuracy and computational complexity. Although most of the deep networks presented in literature and described above cannot be applied as such in a real-time encoding workflow at acceptable cost, we do see benefits in the following areas in the near future.

Optimized encoder decisions

Given the huge search space, encoder decisions are good candidates for speed-up. A number of techniques have already been mentioned above, some of which only work on reference software. But, more shallow neural networks have the potential to provide a good trade-off between complexity and accuracy. In particular the more recent formats such as HEVC, AV1 and VVC will benefit most of NN-based acceleration, given their many degrees of freedom. Promising work has been published in the following directions, among others:

- *Intra prediction*. The work by 'Santamaria et al (35)' presents NN-based intra prediction modes along with simplifications that lead to multiplications in the order of 100s up to 10,000s for 16x16 blocks. This builds on the work of (16), and an interpretation analysis is run to come to simpler, explainable predictors that are easy to implement. The result is NN-based modes that are much closer to real-life usage.
- *Inter Prediction.* Although much of the recent work has focused on NNs, other ML techniques such as Decision Trees prove to be efficient ways to optimize encoder decisions, as in 'Kim et al (36)', where inter prediction is accelerated for AV1.
- *Mode decision*. 'Liu et al (37)' presented a CNN-based CU partition size decision with a reasonable complexity of 3,000 MACs, along with a hardware implementation.

- *Transform selection*. The transform search for AV1 is accelerated in 'Su et al (38)', based on a neural network with one hidden layer. For transform kernel prediction, two shallow networks are used which are combined into a score for the 2D transform.

The message from these papers, along with our findings, are that fairly simple neural networks can produce accurate results, and at acceptable computational complexity.

Video quality measurement

Several metrics have been introduced that are trained based on neural networks, for example to predict PSNR, SSIM, subjective scores or for QoE monitoring. The popular VMAF metric was trained based on Support Vector Machine regression. Still, such metrics can be very complex, making it challenging to calculate them in real-time, or deep inside the encoder. It is however possible to approximate these metrics based on low-complexity networks and features, as described in 'Barman et al (39)'.

Real-time encoding in practice

Rate control is one of the algorithms in real-time encoders that can truly make a difference in video quality. Rate control determines how to optimally allocate bits between GOPs, frames and blocks within a frame, hereby maximizing visual quality. For frame-level rate control, we have seen promising improvements in estimation accuracy, that make rate control more adaptive to different types of content, and helping eliminate misprediction in case of outliers, e.g. rapid transitions between easy and difficult types of content.

In 'Li et al (40)', a CNN is used to predict the Lagrange multiplier, leading to more accurate rate control. Saliency-based encoding is a promising way to direct bits to where they matter most. In 'Lyudvichenko et al (41)', the saliency maps are fed into the x264 encoder, to assist its rate control. Importance maps such as in 'Li et al (42)' can further help adaptive quantization.

In the broadcast world, statistical multiplexing algorithms provide a powerful way to fully utilize the available bandwidth for a bundle of channels. ML-based complexity estimation can help to allocate bitrate to different channels. For VOD applications, ML helps to optimize video quality in a content-adaptive way. By including low-complexity video quality measurement, VQ can be steered in a real-time fashion, leading to decisions optimized down to shot level.

Throttling helps to automatically adjust the complexity of the encoder depending on the available resources on the server. When the CPU load goes up, the encoder can scale the search operations down. In extreme cases, this could have an impact on VQ. Throttling can be assisted by ML operations to find an optimal balance between CPU load, encoder decisions and VQ, resulting in more robust encoders.

CONCLUSIONS

Machine learning techniques have made substantial jumps forward over the last decades. As a side-effect, also the complexity has grown manifold. Although deep neural networks can provide excellent accuracy for a variety of tasks, they come with substantial computational, and hence financial cost. To cope with this cost, there are tendencies toward simplification, efficient co-design and hardware acceleration for DNN inference networks.

Video compression, in particular real-time encoding, poses challenges for any platform. Millions of pixels, and thousands of decisions have to be processed every second. A trade-off needs to be made between handcrafted techniques and deep networks that hide some of the 'interpretability'. Unfortunately, many of the techniques described in literature only perform well on very complex encoders, such as poorly optimized reference software. Once you apply these to fast encoders, the true complexity of these DNNs becomes apparent.

It is clear that more work is needed to make these approaches more accessible, and at a cost that is close to what can be achieved with expert-tuned heuristics. At least some good examples can be found, but more publications need to be explicit about the applicability and focus on computational complexity. Nonetheless, good examples can be found of where ML can be applied in real-time video encoding, leading to faster, more adaptive and robust encoders.

REFERENCES

1. V. Sze, Y. Chen, T. Yang and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," in Proc. of the IEEE, vol. 105 (12), pp. 2295-2329, Dec. 2017.
2. D. Ding, Z. Ma, D. Chen, Q. Chen, Z. Liu and F. Zhu, "Advances in Video Compression System Using Deep Neural Network: A Review and Case Studies," in Proc. of the IEEE, March 2021.
3. A. Canziani, E. Culurciello and A. Paszke, "Evaluation of neural network architectures for embedded systems," IEEE Int. Symposium on Circuits and Systems (ISCAS), 2017.
4. J. De Cock, "Navigating a Multi-Codec World", Streaming Media Magazine, https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=145580.
5. G. W. Cottrell, P. Munro and D. Zipser, "Image compression by back propagation: an example of extensional programming," Models of cognition: rev. of cognitive science, vol. 1 (208), 1989.
6. S. Marsi, G. Ramponi and G. L. Sicuranza, "Improved neural structures for image compression," Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), 1991, pp. 2821-2824.
7. E. Gelenbe and M. Sungur, "Random network learning and image compression," in IEEE Int. Conf. on Neural Networks (ICNN), vol. 6, 1994, pp. 3996–3999.
8. C. Cramer, E. Gelenbe and H. Bakircloglu, "Low bit-rate video compression with neural networks and temporal subsampling," in Proceedings of the IEEE, vol. 84, no. 10, pp. 1529-1543, Oct. 1996.
9. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504–507, 2006.
10. T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, Z. Ma, "DeepCoder: A deep neural network based video compression". IEEE Visual Communications and Image Processing Conference (VCIP), 2017.
11. J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression", International Conference on Learning Representations (ICLR), April 2017.

12. Z. Chen, T. He, X. Jin and F. Wu, “Learning for Video Compression”, in IEEE Trans. on Circuits and Systems for Video Technology, vol. 30 (2), pp. 566-576, Feb. 2020.
13. G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai and Z. Gao. “DVC: An End-To-End Deep Video Compression Framework.” IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
14. W. Cui et al., “Convolutional Neural Networks Based Intra Prediction for HEVC”, Data Compression Conference (DCC), 2017, pp. 436-436.
15. J. Li, B. Li, J. Xu and R. Xiong, “Intra prediction using fully connected network for video coding”, IEEE International Conference on Image Processing (ICIP), 2017.
16. J. Pfaff, P. Helle, D. Maniry, S. Kaltenstadler, W. Samek, H. Schwarz, D. Marpe, and T. Wiegand, “Neural network based intra prediction for video coding,” SPIE Applications of Digital Image Processing XLI, vol. 10752, 2018.
17. J. Liu, S. Xia, W. Yang, M. Li, and D. Liu, “One-for-all: Grouped variation network-based fractional interpolation in video coding,” IEEE Transactions on Image Processing, vol. 28 (5), pp. 2140–2151, 2018.
18. H. Choi and I. V. Bajić, “Deep Frame Prediction for Video Coding”, in IEEE Trans. on Circuits and Systems for Video Technology, vol. 30 (7), pp. 1843-1855, July 2020.
19. S. Liu et al., “AhG on neural network based coding tools”, JVET-S0267, June 2020.
20. G. Correa, P. A. Assuncao, L. V. Agostini and L. A. da Silva Cruz, “Fast HEVC encoding decisions using data mining,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 25 (4), pp. 660– 673, Apr. 2015.
21. M. Xu, T. Li, Z. Wang, X. Deng, R. Yang and Z. Guan, "Reducing Complexity of HEVC: A Deep Learning Approach," in IEEE Transactions on Image Processing, vol. 27 (10), pp. 5044-5059, Oct. 2018.
22. S. Paul, A. Norkin and A. C. Bovik, “Speeding Up VP9 Intra Encoder with Hierarchical Deep Learning-Based Partition Prediction”, in IEEE Transactions on Image Processing, vol. 29, pp. 8134-8148, 2020.
23. C. Chiang, J. Han and Y. Xu, “A Multi-Pass Coding Mode Search Framework for AV1 Encoder Optimization”, Data Compression Conference (DCC), 2019.
24. G. Tang, M. Jing, X. Zeng and Y. Fan, “Adaptive CU Split Decision with Pooling-variable CNN for VVC Intra Encoding”, IEEE Visual Communications and Image Processing (VCIP), 2019.
25. J. Zhao, Y. Wang, Q. Zhang, “Adaptive CU Split Decision Based on Deep Learning and Multifeature Fusion for H.266/VVC”, Scientific Programming, 2020.
26. W. Yang, X. Zhang, Y. Tian, W. Wang, J. Xue and Q. Liao, "Deep Learning for Single Image Super-Resolution: A Brief Review," in IEEE Transactions on Multimedia, vol. 21 (12), pp. 3106-3121, Dec. 2019.
27. D. Liu, Y. Li, J. Lin, H. Li, F. Wu, “Deep Learning-Based Video Coding: A Review and A Case Study”, ACM Computing Surveys 53 (1), pp. 1-35, Feb. 2020.
28. A. Reuther, P. Michalea, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, “Survey of Machine Learning Accelerators”, IEEE High Performance Extreme Computing Conference (HPEC), 2020.

29. Zhang, D., J. Yang, Dongqiangzi Ye and G. Hua. “LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks”, European Conference on Computer Vision (ECCV), pp. 365-382, 2018.
30. Y. LeCun, J. S. Denker and S. A. Solla, “Optimal Brain Damage,” in NIPS, 1990.
31. H. Li, K. Asim, I. Durdanovic, H. Samet and H. Graf. “Pruning Filters for Efficient ConvNets.”, ICLR 2017.
32. M. Verhelst and B. Moons, "Embedded Deep Neural Network Processing: Algorithmic and Processor Techniques Bring Deep Learning to IoT and Edge Devices," in IEEE Solid-State Circuits Magazine, vol. 9 (4), pp. 55-65, Fall 2017.
33. W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, B. Yu, “Definitions, methods, and applications in interpretable machine learning”, Proc. of the National Academy of Sciences, vol. 116 (44), Oct 2019, pp. 22071-22080.
34. L. Murn, S. Blasi, A. F. Smeaton, N. E. O’Connor and M. Mrak, “Interpreting CNN For Low Complexity Learned Sub-Pixel Motion Compensation in Video Coding”, IEEE International Conference on Image Processing (ICIP), 2020, pp. 798-802.
35. M. Santamaria, S. Blasi, E. Izquierdo and M. Mrak, “Analytic Simplification of Neural Network Based Intra-Prediction Modes for Video Compression”, IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 2020.
36. J. Kim, S. Blasi, A. S. Dias, M. Mrak and E. Izquierdo, “Fast Inter-prediction Based on Decision Trees for AV1 Encoding”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 1627-1631.
37. Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji and D. Wang, “CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network”, IEEE Transactions on Image Processing, vol. 25 (11), pp. 5088-5103, Nov. 2016.
38. H. Su, M. Chen, A. Bokov, D. Mukherjee, Y. Wang and Y. Chen, "Machine Learning Accelerated Transform Search for AV1," Picture Coding Symposium (PCS), 2019.
39. N. Barman, E. Jammeh, S. A. Ghorashi and M. G. Martini, “No-Reference Video Quality Estimation Based on Machine Learning for Passive Gaming Video Streaming Applications”, in IEEE Access, vol. 7, pp. 74511-74527, 2019.
40. Y. Li, B. Li, D. Liu and Z. Chen, "A convolutional neural network-based approach to rate control in HEVC intra coding," IEEE Visual Communications and Image Processing (VCIP), 2017.
41. V. Lyudvichenko, M. Erofeev, A. Ploshkin, and D. Vatolin, “Improving video compression with deep visual-attention models,” International Conference on Intelligent Medicine and Image Processing, 2019, pp. 88–94.
42. M. Li, W. Zuo, S. Gu, D. Zhao, D. Zhang, “Learning convolutional networks for content-weighted image compression”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, 3214-3223.