Neural Networks Based Fractional Pixel Motion Estimation for HEVC

Ehab M. Ibrahim[1,3], Emad Badry[1], Ahmed M. Abdelsalam[2], Ibrahim L. Abdalla[3], Mohammed Sayed[1,3], Hossam Shalaby[1,4]

[1] ECE Department, Egypt-Japan University for Science and Technology, Alexandria, Egypt
[2] Computer and Software Engineering Department, Polytechnique Montreal, Montreal, Canada
[3] ECE Department, Zagazig University, Zagazig, Egypt
[4] Electrical Engineering Department, Alexandria University, Alexandria, Egypt
{ehab.ibrahim, emad.mahmoud, mohammed.sayed}@ejust.edu.eg,
ahmed.abdelsalam@polymtl.ca, ilotfy@zu.edu.eg, shalaby@ieee.org

Abstract—**High Efficiency Video Coding (HEVC) provides more compression than its predecessors. One of the modules that contributes to higher compression rates is the Motion Estimation module, which consists of Integer and Fractional pixel motion estimation. The Fractional Motion Estimation (FME) process performs interpolations to find sample values at fractional-pixel locations, which can be computationally demanding. In this paper, we propose an interpolation-free method for FME based on Artificial Neural Networks (ANNs). Our proposed method is implemented in HEVC reference software (HM-16.9). According to our results, ANNs can accomplish FME task with an average increase of 2.6% in BD-Rate and an average reduction of 0.09 dB in BD-PSNR.**

Keywords—Fractional Motion Estimation (FME), High Efficiency Video Coding (HEVC), Deep Learning, Artificial Neural Networks (ANNs)

I. INTRODUCTION

High Efficiency Video Coding (HEVC) is a new and trending video coding standard. It's a strong candidate to be a successor for Advanced Video Coding (AVC) standard, as it provides better compression of up to 50% bitrate reduction with the same video quality [1]. One of the techniques that contribute to better video compression is Motion Estimation (ME), where frames of the video are predicted depending on past or future reference frames. The ME process is achieved using two steps: Integer-pixel Motion Estimation (IME) and Fractional-pixel Motion Estimation (FME).

First, IME predicts the location of the best matched Prediction Block (PB) in a reference frame with integer-pixels precision. Second, FME is a refinement of the IME result that searches for a better match with sub-pixels precision. The FME block is computationally expensive since it interpolates the sub-pixels around the selected integer location of the MV. Additionally, the ME process demands 40-60% of the processing time of the whole HEVC encoder [2]. Therefore, simpler FME approaches are highly desired.

Deep Learning [3] have proven successful in lots of applications. Inspired by the huge success of deep learning, we propose an interpolation-free Artificial Neural Network (ANN) approach that performs FME and predicts the optimum MV with quarter-pixel precision. The ANN is fed with error values of the best integer location and eight surrounding integer points, along with the PB size. The network was trained by a dataset extracted from six well-known video sequences, using the traditional Stochastic Gradient Descent (SGD) backpropagation algorithm. Our results show that ANNs can effectively perform FME task at the cost of an average increase of 2.6% in BD-Rate and average reduction of 0.09 in BD-PSNR. It also shows great promise for optimizations in terms of prediction accuracy and computational cost.

The rest of the paper is organized as follows: Section II shows previous efforts to perform interpolation-free FME. Section III showcases the proposed ANN architecture, and the techniques used to enhance its performance. The experimental results are presented in Section IV, and Section V concludes the paper and presents ideas for future work.

II. RELATED WORKS

Due to the computationally intensive traditional interpolation method used in standard HEVC, different approaches have been presented to overcome this huge complexity. The basic idea is to estimate the optimum fractional pixel MV by modelling the matching error surface surrounding the best integer position. Several works use eight matching error values spaced one pixel from the center, which can be fitted as 2-D paraboloid surface, then get the minimum point as the best fractional location. The previous approaches model the error surface mathematically using any of 9 terms [4], 6 terms [5], or 3 terms [6]–[8].

The authors of [5] derived the mathematical model terms from 9 matching error values by solving overdetermined equations using convex optimization method. In [9], the fractional MV is estimated by the intersection of the two main parabolas derived from the parallel horizontal and vertical planes. The algorithm is then improved by rejecting the outliers amidst the 9 integer-pixel locations. The previous results were improved in the work of [10] by analyzing the error surface and investigating the vertex direction. In [11], the author evaluates the best fractional matching MV from among four different directional patterns. Works of [12] proposes estimating the fractional MV using 25 matching error values, which incur additional overhead in the case of IME fast search mode, since it does not estimate the required 25 matching error values.

Authors of [13] proposed performing fractional-pixel Motion Compensation (MC) using a Convolutional Neural Network. The difference between our approaches is that in [13], the interpolation-based ME algorithm is the same as the standard, and only the MC algorithm is replaced. While in our approach, we replace the ME algorithm with an interpolation-free ANN that predicts the output MV.

All aforementioned schemes model the error surface mathematically without considering different video characteristics. To the best of our knowledge, our work is the first attempt to utilize deep learning in performing interpolation-free FME task, and to predict the best fractional MV without modelling the error surface mathematically.

III. Proposed Architecture

In this section, we describe the architecture of our proposed ANN. We justify the design choices we took for the network's hyperparameters, as well as the optimization techniques used to improve both training time and prediction accuracy. Information about the data used in our network's training and validation are also presented.

A. Artificial Neural Network Architecture

At first, we need to define the inputs and expected output of our network. The inputs are nine matching error values of integer-pixel locations, plus the height and width of the assigned PB. We approach the FME problem as a multi-class classification problem, where the output can only be one of 49 points - the center integer point and surrounding 48 pixel locations with quarter precision - as shown in Fig. 1. The most suitable deep learning architecture for that kind of problem is a Fully Connected (FC) ANN. The network has a total of 11 inputs, and its output is a Log-Softmax layer with 49 outputs, which predicts the most probable quarter-pixel location for the given inputs. We trained our network with the traditional SGD backpropagation algorithm.

Several deep learning optimization techniques have been employed, such as: a) Dropout [14], b) Batch Normalization (BN) [15], c) Entity Embeddings [16]. Dropout is the process of dropping a fraction of the network's activations at the start of each training epoch, so as to prevent overfitting. BN handles normalizing the activations of each layer, and scaling according to a trainable parameter, which accelerates the training process. BN also has a slight regularization effect which contributes in preventing overfitting. Both dropout and BN are used in all layers of our network.

Entity Embeddings are used to handle the input values of PB width and height, which can only be one of certain values. Such inputs are also known as categorical variables.

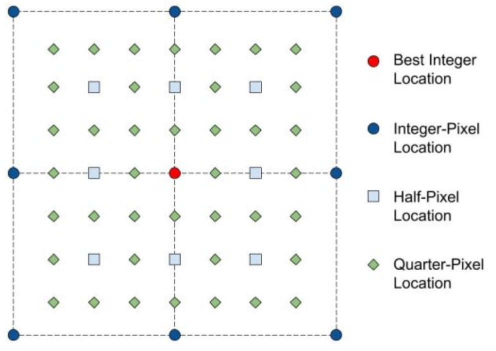


Figure 1. Integer and Fractional pixel locations

With Entity Embeddings, each categorical variable gets replaced with several floating-point numbers (in our case, four floating-point numbers) that are trainable along with the network's weights and biases.

One ANN was trained for each Quantization Parameter (QP), having a total of four trained ANNs. An illustration of the architecture is shown in Fig. 2. The training process was carried out using FastAI library, which is based on PyTorch.

B. Choice of Hyperparameters

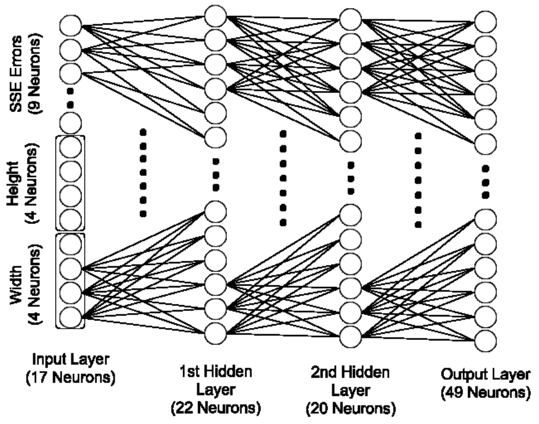
FC networks may consist of several hidden layers. We have experimented with multiple combinations of hidden layers and found that two layers provided the best trade-off between prediction accuracy and computational cost. Hence, only two hidden layers were used for our work. Our layers consist of 22 and 20 neurons respectively. The choice of the number of neurons per layer was mostly arbitrary, where we only considered having a smaller number of neurons on the second layer to reduce the number of computations.

To find the optimum learning rate, we used the cyclical learning rate method proposed in [17], where we found it to be equal to $1\text{x}10^{-3}$. As a result, we trained our network with $1\text{x}10^{-3}$ learning rate for 44 epochs. Furthermore, our model was trained for 6 more epochs with $1\text{x}10^{-4}$ learning rate to achieve extra fine-tuning, resulting in a total of 50 training epochs for each network.

C. Training Data

Our approach when extracting the training data was to generalize the ANN to work well under all conditions. Therefore, the data was extracted from six video sequences. The selected sequences were a mixture of high and low resolutions, and with fast and slow movements. To balance the number of error values pulled from each sequence, we chose a lower number of frames for higher resolution videos, and vice versa. Table I shows the selected video sequences, along with the specific frames in each sequence.

Four sets of data were extracted for QP values of {22, 27, 32, 37}, and each set was used to train an independent ANN. First, each data set was normalized by subtracting the mean and dividing by the standard deviation of each of the inputs, resulting in input error values with zero mean and unit



SSE Errors (9 Neurons)
Height (4 Neurons)
Width (4 Neurons)
Input Layer (17 Neurons)
1st Hidden Layer (22 Neurons)
2nd Hidden Layer (20 Neurons)
Output Layer (49 Neurons)

Figure 2. Neural Network Illustration

TABLE I. TRAINING DATA EXTRACTION

Class	Video Sequence	Number of Frames	Frame Skip
Class B	Kimono	5	60
Class C	PartyScene	20	60
Class D	BlowingBubbles	30	60
Class E	Johnny	10	60
Class F	SlideEditing	10	60
	SlideShow	10	60

variance. The values of mean and standard deviation for each input were stored for later, to perform normalization of error values during run-time. Then, for each set, 80% of the data were used for training, and 20% were used for validation. The validation set was chosen randomly.

D. Computational Cost

Training process of our network is done offline, and only the forward pass is executed during video encoding. The predicted output of the network only depends on the error values computed by the IME step, hence saving the complexity of the filters used for interpolation process used by HEVC standard. Instead, ANNs operate by performing consecutive multiplications and additions.

Our network, with two hidden layers, requires a total of 1936 additions and 1854 multiplications per prediction. A breakdown of the used operations is shown in Table II. Data normalization is the process of subtracting the mean and dividing by the standard deviation for each input. Linear layers describe multiplying weights and adding biases for each layer, and BN is a technique used for accelerating the training process.

Having four independent networks in our implementation does not incur any additional overhead in terms of computational resources. The only overhead is in terms of memory used to store all four sets of network parameters, and the slight delay of initializing the parameters depending on the used QP, which happens only once per video.

It's worth mentioning that several techniques can be used to reduce the number of used operations. For example, ANN pruning can be used to decrease the number of neurons, resulting in smaller and faster networks. Quantizing trainable parameters can also be used to reduce multiplication complexity with only a slight reduction in prediction accuracy. The implications of pruning and quantization are yet to be studied and are out of the scope of this paper.

TABLE II. BREAKDOWN OF COMPUTATIONAL RESOURCES

	Additions	Multiplications
Data Normalization	9	9
Linear Layers	1885	1794
Batch Normalization (BN)	42	51
Total	**1936**	**1854**

IV. EXPERIMENTAL RESULTS

To evaluate our proposed architecture, we' implemented it in HEVC standard software (HM-16.9). The QP values are {22, 27, 32, 37}. The configuration used for our implementation is "encoder_lowdelay_P_main", with fast search algorithm for IME and search range of 64. Finally, the error criterion used is the Error Sum of Squares (SSE).

Table III shows the computed BD-Rate and BD-PSNR for our proposed implementation. Our results are compared with works of [5], [9], [10], which are all interpolation-free and require only 9 matching error values. We have implemented the work of [5] and ran it on the same configuration of our architecture. The algorithms of [9], [10] have been implemented in the work of [11]. It's shown that our proposed network achieves an average increase of 2.6% in BD-Rate, and an average reduction of 0.09 dB in BD-PSNR. Compared to [5], our method was able to achieve an average of 0.7% lower BD-Rate and 0.04 dB higher BD-PSNR. The BD-PSNR was not reported in the works of [9] and [10], and only the BD-Rates of classes B, C and D were reported. For the average BD-Rate of classes B, C, and D, our proposed network achieved 0.5% lower BD-Rate than [8] and 0.1% lower BD-Rate than [9].

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new technique for performing interpolation-free FME, which utilizes deep learning. The inputs for our ANN are the SSE error values for the best integer-pixel location and eight surrounding integer locations, plus the PB height and width. Our network predicts the best point of 49 fractional locations, including the best integer location, with quarter-pixel precision. Results from implementing our method shows an average increase of 2.6% in BD-Rate, and an average reduction of 0.09 dB in BD-PSNR. Using deep learning in FME shows promise for reducing computational resources, hence making it more hardware friendly.

This work can be extended in many ways, the most obvious being optimizing the ANN for better results, which is presumably related to finding the optimum training data. Next, we can research ANN pruning and quantization, and how would they affect both prediction accuracy and computational resources. Finally, tailoring the ANN to specific applications can be studied. For example, the ANN can be tailored for teleconference applications by simply changing the training data to reflect its nature. With the rapid pace of deep learning research, ANNs will only get better at performing FME.

ACKNOWLEDGMENT

We would like to thank Egypt-Japan University of Science and Technology (E-JUST) for the continuous support and the Egyptian Ministry of Higher Education (MoHE) and National Telecom Regulatory Authority (NTRA) of Egypt for funding this work.

TABLE III. RESULTS OF PROPOSED ARCHITECTURE

		[9]	**[10]**	**[5]**		**Proposed**	
		BD-Rate (%)	***BD-Rate (%)***	***BD-Rate (%)***	***BD-PSNR (dB)***	***BD-Rate (%)***	***BD-PSNR (dB)***
Class B	BQTerrace	2.1	2.0	4.2	-0.05	4.5	-0.06
	BasketballDrive	2.3	2.1	1.8	-0.04	1.9	-0.04
	Cactus	2.5	2.2	2.3	-0.05	1.7	-0.03
	Kimono	1.2	1.0	0.7	-0.02	1.2	-0.03
	ParkScene	2.0	1.8	1.4	-0.04	1.7	-0.05
Average		**2.0**	**1.8**	**2**	**-0.04**	**2.2**	**-0.04**
Class C	BQMall	3.7	3.3	2.9	-0.1	2.6	-0.09
	BasketballDrill	3.5	3.1	2.4	-0.09	2.1	-0.07
	PartyScene	3.1	2.8	4.4	-0.15	3.0	-0.1
	RaceHorses	4.7	4.3	3.6	-0.12	2.9	-0.1
Average		**3.7**	**3.4**	**3.3**	**-0.11**	**2.7**	**-0.09**
Class D	BQSquare	3.0	2.7	8.5	-0.25	6.0	-0.18
	BasketballPass	4.4	3.8	3.1	-0.16	2.5	-0.11
	BlowingBubbles	3.8	3.5	4.6	-0.14	3	-0.1
	RaceHorses	6.7	5.7	4.5	-0.18	3.7	-0.15
Average		**4.5**	**3.9**	**5.1**	**-0.18**	**3.8**	**-0.14**
Averge of Class B,C,D		**3.4**	**3.0**	**3.5**	**-0.11**	**2.9**	**-0.09**
Class E	FourPeople	Not Reported	Not Reported	1.6	-0.05	2.0	-0.06
	Johnny			2.5	-0.05	3.0	-0.06
	KristenAndSara			1.6	-0.04	2.3	-0.06
Average		**-**	**-**	**1.9**	**-0.04**	**2.5**	**-0.06**
Class F	BasketballDrillText	Not Reported	Not Reported	2.8	-0.1	2.4	-0.09
	ChinaSpeed			5	-0.23	1.2	-0.05
	SlideEditing			3.1	-0.4	1.0	-0.12
	SlideShow			6.4	-0.4	3.7	-0.26
Average		**-**	**-**	**4.3**	**-0.28**	**2.1**	**-0.13**
Total Average		**-**	**-**	**3.3**	**-0.13**	**2.6**	**-0.09**

REFERENCES

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649–1668, 2012.

[2] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1685–1696, 2012.

[3] Y. LeCun, B. Yoshua, and H. Geoffrey, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[4] S. Dikbas, T. Arici, and Y. Altunbasak, "Fast motion estimation with interpolation-free sub-sample accuracy," IEEE Trans. Circuits Syst. Video Technol., vol. 20, no. 7, pp. 1047–1051, 2010.

[5] Y. Li, Z. Liu, X. Ji, and D. Wang, "HEVC fast FME algorithm using IME RD-costs based error surface fitting scheme," VCIP 2016 - 30th Anniv. Vis. Commun. Image Process.

[6] M. S. Sayed, W. Badawy, and G. Jullien, "Interpolation-Free Fractional-Pixel Motion Estimation Algorithms with Efficient Hardware Implementation," J. Signal Process. Syst. Signal Image Video Technol., vol. 67, no. 2, pp. 139–155, 2012.

[7] E. Badry, A. Shalaby, and M. S. Sayed, "A hardware friendly fractional-pixel motion estimation algorithm based on adaptive weighted model," Proc. Int. Conf. Microelectron. ICM, vol. 2017–Decem, pp. 1–4, 2018.

[8] M. Sayed, W. Badawy, and G. Jullien, "Low-complexity algorithm for fractional-pixel motion estimation," Proc. - Int. Conf. Image Process. ICIP, pp. 1565–1568, 2009.

[9] W. Dai, O. C. Au, W. Zhu, W. Hu, P. Wan, and J. Li, "A robust interpolation-free approach for sub-pixel accuracy motion estimation," 2013 IEEE Int. Conf. Image Process. ICIP 2013 - Proc., pp. 1767–1771, 2013.

[10] X. Zuo and L. Yu, "A novel interpolation-free scheme for fractional pixel motion estimation," 2015 Pict. Coding Symp. PCS 2015 - with 2015 Pack. Video Work. PV 2015 - Proc., pp. 80–84, 2015.

[11] R. Fan, Y. Zhang, B. Li, and G. Wang, "Multidirectional parabolic prediction-based interpolation-free sub-pixel motion estimation," Signal Process. Image Commun., vol. 53, pp. 123–134, 2017.

[12] E. Badry, A. Shalaby, and M. S. Sayed, "Fast fractional-pixel motion estimation using Lagrangian-based error surface interpolation," 2017 IEEE Glob. Conf. Signal Inf. Process. Glob. 2017 - Proc., vol. 2018–Janua, pp. 151–155, 2018.

[13] N. Yan, D. Liu, H. Li, B. Li, L. Li, and F. Wu, "Convolutional Neural Network-Based Fractional-Pixel Motion Compensation," IEEE Trans. Circuits Syst. Video Technol., vol. 8215, no. c, pp. 1–1, 2018.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," J. Mach. Learn. Res., vol. 15, pp. 1929–1958, 2014.

[15] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Feb. 2015.

[16] C. Guo and F. Berkhahn, "Entity Embeddings of Categorical Variables," no. 1, pp. 1–9, 2016.

[17] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," no. April, 2015.