Wyner-Ziv Video Coding using Hadamard Transform and Deep Learning

Jean-Paul Kouma

Ulrik Söderström
Department of Applied Physics and Electronics
Umea˚ University, Sweden

***Abstract*—Predictive schemes are current standards of video coding. Unfortunately they do not apply well for lightweight devices such as mobile phones. The high encoding complexity is the bottleneck of the Quality of Experience (QoE) of a video conversation between mobile phones. A considerable amount of research has been conducted towards tackling that bottleneck. Most of the schemes use the so-called Wyner-Ziv Video Coding Paradigm, with results still not comparable to those of predictive coding. This paper shows a novel approach for Wyner-Ziv video compression. It is based on the Reinforcement Learning and Hadamard Transform. Our Scheme shows very promising results.**

***Keywords*—Wyner-Ziv; video coding; rate distortion; Hadamard transform; Deep learning; Expectation Maximization**

I. Introduction

Video compression schemes such as MPEG4 and H.264 [1] are the current state-of-art, where correlation between or among frames are exploited at the encoder side. Such schemes usually achieve high compression with a fairly low complexity at the very expense of a high complexity encoder. Compression schemes like MPEG4 or H.264 are suitable for scenarios where the encoder has enough power computation, like video-on-demand servers.

Mobile phones are today the de facto device for communication. People want to do more and more with their mobile phone. They want to be able to have a real-time video communication experience comparable to that of computers. Unfortunately, current video compression technologies [1] barely permit it: **encoder complexity** is the bottleneck. Either comparable frame rate can not be achieved or conversation cannot last long because of battery is scarce. In either case quality of experience will be dropped.

The video community is aware of that issue as a great deal of research has been conducted since the emergence of camera-based mobile devices. The common insight toward tackle the issue is the so-called *Wyner-Ziv Video Coding (WZVC)* or *Distributed video coding*.

WZVC is the consequence of information-theoretic bounds established in the 1970s. First by Slepian and Wolf for distributed lossless coding [2], and then Wyner and Ziv for lossy coding with decoder side information [3].

Let $\{(X_k, Y_k)\}_{k=1}^{\infty}$ be a sequence of independent drawing of a pair of dependent variables (X, Y) taking values in the finite sets $\mathbb{X}$ and $\mathbb{Y}$, respectively. The decoder has access to the side information Y. Illustration is shown in figure 1. Wyner and Ziv suggest that whether or not the side information Y is available at the encoder, X can be compressed - to Z and decoded to $\hat{X}$ - at a rate $R_{X|Y}(D)$ where $D = \mathbb{E}[d(X, \hat{X})]$ is an acceptable distortion.

In WZCV, unlike that of predictive coding paradigms (i.e. H.264), individual frames are encoded separately but decoded conditionally. According to [3], the compression effectiveness of WZVC schemes should be comparable to that of predictive coding. A typical WZVC setup is shown in figure 2 where both terminals are lightweight modern mobile phone capable of decoding MPEG4 frames for example. The corresponding Wyner-Ziv decoder is thought to be powerful computer capable of exploiting statistics between frames and output MPEG4 streams in real-time, using much more complex algorithms.

Most of the conducted studies in the area of WZCV have been using binary codes. Major contributions come from Standford University [4] and UC Berkeley [5]. Both methods followed a common pattern; those methods were first developed to perform in pixel domain and later in transform domain (namely Discrete Cosine Transform). Those methods suffered from three major drawbacks:

- The overhead of working in binary domain - since DCT pixels or alternatively transform coefficients have to be converted back and forth from and to bit planes during decoding process.
- Rate control - All the pixels values, alternatively transform coefficients need to be converted to binary with the same amount of bits, making the rate control difficult.
- The decoding algorithms used - either generative or discriminative - were somewhat too simplistic and did not work well in practice.

We propose a new practical compression scheme, based on Hadamard transform and reinforcement learning. In contrast to previous works, our method deals with non-binary codes. The encoding is relatively of low complexity with an inherent rate control. We also show that our algorithm outperforms that of the state of art [4] and [5] and really is comparable to predictive coding schemes.

II. Low complexity video encoding

The encoding challenge is to implement an encoder with lower encoding complexity than that of predictive video coding methods [1] and still achieve comparable codec **effectiveness**.

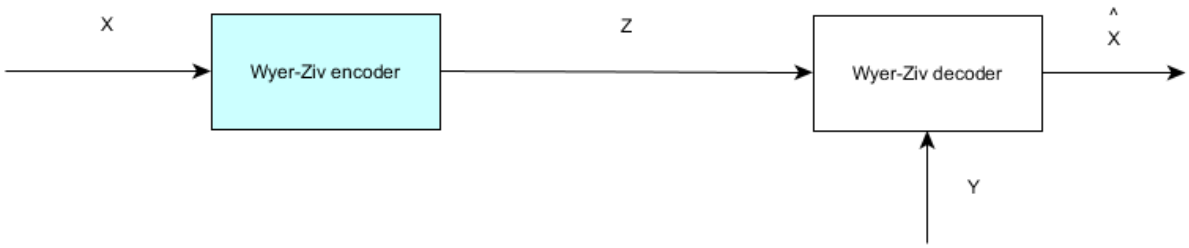

Fig. 1. Source coding with side information available at the decoder

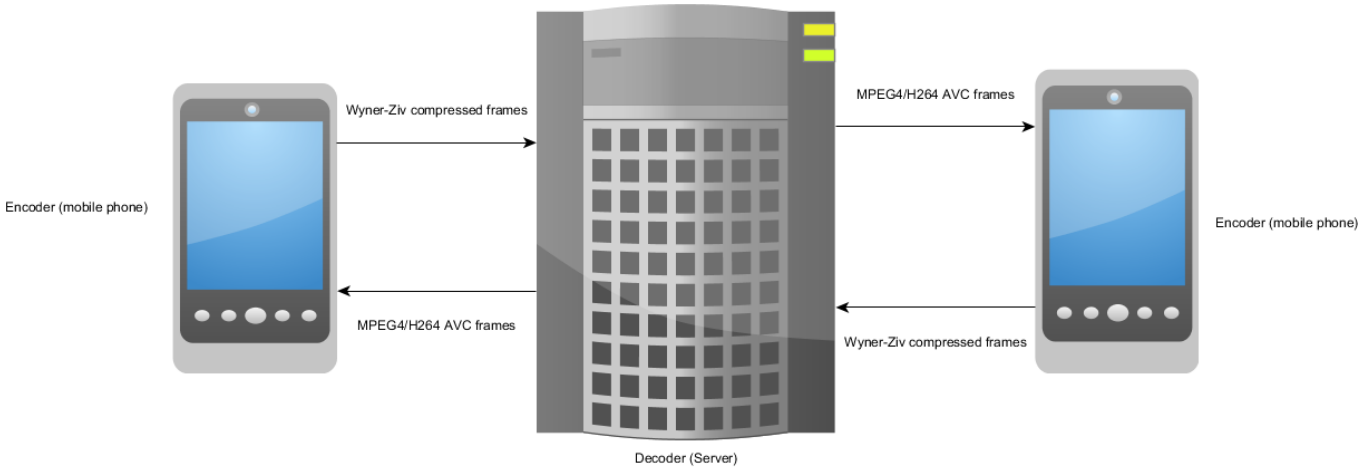

Fig. 2. Wyner-Ziv setup of video compression

A. Problem setup

Let Y and X be two consecutive video frames. X is to be encoded without the knowledge of Y. The challenge is to compress X at the bit rate R **bit/pixel**, where $R = H(Y|X) =$. $H(Y|X)$ is the entropy or the amount of information (in bits) needed to represent the frame X conditioned on that of Y. Usually, Y and X are highly correlated after motion compensation [6], $H(X|Y) < H(X)$. In practice the challenge is to encode X at a rate even lower than $R_\epsilon < H(Y|X)$ - leading to **lossy compression** - and still achieve reconstruction with satisfying fidelity, as suggested by Ziv et al [3].

B. The encoder

Let Z^* be the compressed frame or Wyner-Ziv frame from X. In our case, compression with compression ratio $n : m$ is achieved simply projecting the row version of frame $X \in \mathbb{R}^{1\times n}$ onto the $n : th$ first dimensions of the **orthogonal Hadamard vector basis** $G \in \mathbb{R}^{n\times m}$, where $m < n$. $Z^* = X \times G$. The final stream is $Z = [Z^* \quad \sigma^2]$ where σ^2 is the variance between the current frame X and the previous frame X_{-1}. Its given by

$$\sigma^2 = \sum_{n=0}^{N} X[n] - X_{-1}[n-1] \tag{1}$$

C. The Hadamard Transform

The Hadamard transform is an orthogonal transform that has been used in numerous image coding applications [7], [8]. The transform matrix of dimension 2^k for $k \in \mathbb{N}$ is given by the following recursive formula

$$H(2^k) = \begin{bmatrix} H(2^k-1) & H(2^k-1) \\ H(2^k-1) & -H(2^k-1) \end{bmatrix}$$

and

$$H(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

III. Decoding

A. Problem setup

The aim here is to reconstruct the encoded frame Z to X. At time t, the decoder has knowledge of the incoming Wyner-Ziv frame Z and previously decoded frame Y.

To be able to reconstruct X, Z has somehow to contain enough information about Z, possibly together with Y; that is the case in predictive coding schemes, where Z carries information about Y, usually motion vectors (pixel configuration for that matter) and Huffman or Arithmetic coded residuals of the motion-compensated frames [1]. Since we do not really

know the real pixel setup X we will be left to "guessing" it out of the pixel space $\mathbf{\Lambda}$. Mathematically, the problem could be formulated as follow: at a time t, the decoder **observing** Z and Y, aims to find the **"best"** pixel configuration $\hat{X}$. . The term **"best"** used here actually points out that X is **unobserved**. This formation leaves us to a **Maximum Likelihood** problem

$$\hat{X} \leftarrow \arg\max_{\Lambda} P(\hat{X}, Z, Y) \tag{2}$$

B. Maximum Likelihood

Decoding X by simply estimating doing $X = Z \times G^{-1}$ is obviously not optimal, but it may be worth mentioning why at this point of the study. As X grows in size, pixels in it decrease in term of correlation. Consequently, the coefficients in Z won't explain X well. Estimating X become thus equivalent to solving an under-determined system of equations - fewer equations than unknowns.

To find the "best" estimate of X we have to model an optimal Maximum Likelihood Estimator (MLE). That is, designing the Maximum Likelihood estimator to "capture" as much decoding information in Y and Z as possible, that is capable of estimating the following joint quantity

$$MLE(\hat{X}) = P(\hat{X}, Z, Y; \Theta) \tag{3}$$

where Θ is the generative model. Hidden Markov Mode (HMM)l [9] and Reinforcement Learning models (such as Q-Learnining) [10] are two good candidates for such problem. But modelling such MLE problem could be rather complex if applying either HMM or Q-learning due to the dimensionality of the tuples. Fortunately Q-learning has a variant, using function approximators and experience replay [11], that has shown to deal well with high dimensions.

C. Q-learning

Q-learning is a deep learning technique. Generally spoken, the learning model tries to learn the optimal so-called action-selection policy. We are given an agent, states **S** and a set of actions per state **A**. At a time t, the agent receives a reward r_t by executing an action a_t being in state s_t. The goal of the agent is to maximize its total reward by learning optimal action for each state; that is the cumulative discounted long-term reward $Q(a, s)$, starting from the current state. During learning process, the $Q(a, s)$ value is updated as follow

$$Q_{t+1}(a_t, s_t) \leftarrow Q_t(a_t, s_t) + \tag{4}$$

$$\alpha \left[r_{t+1} + \lambda \arg\max_{a} Q_t(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{5}$$

Due to high dimensionality equation (4) cannot be applied directly to our ML problem. Instead we use a variant called *Q-learning with experience replay* [11].

Q-learning with experience replay, the agent's state-action pair is stored in a data set and re-sampled, with respect to some significance criteria, in some later episodes in conjunction with other selected, usually randomly, data set of state-action pairs.

Our Q-learning scheme is endorsing the same intuition and motivation albeit somewhat different in design; The following setup is adopted

- the output of our Q-learning scheme - Q-values - is two dimensional, as opposed to other schemes [11] outputting one-dimensional.
- An episode ends at either n iterations or p dB of PSNR. Whichever comes first. For example $n = 20$ or $p = 45$.
- The reward is delayed, i.e. until episode ends

Each column of the Q-values corresponds to the probability distributions - that are assumed to be Gaussian - over the co-located pixels candidates of X; at every position $X[i]$ in X, we consider $\lfloor \mathbf{2}\sigma \rfloor$ pixel candidates

$$X_j[i], \quad -\sigma \leq j \leq \sigma, \quad j \in \mathbb{Z}$$

Each pixel candidate $X_j[i]$ has an initial probability

$$P_{ij}^0 = \frac{1}{\sigma\sqrt{2\pi}} e^{-(j)^2/2\sigma^2}$$

Our Q-learning design is illustrated in figure 3.

D. Maximum Likelihood through Experience Replay

Recall that the idea behind this whole Q-learning business is to fit a likelihood function. We aim to find out how to capture information out of the previously reconstructed frame, an encoded stream (that is syndrome and mean-squared error) so as decoding is as effective as possible. A Maximum Likelihood function does permit us to estimate the (degree of) truthfulness of a pixel combination - possibly with some measure of confidence of interval.

Even though likelihood could be measured for every pixel configuration of X in Λ estimating the best configuration still remains intractable as X is of high dimension. We use ***Expectation-Maximization (EM)*** [9] to estimate the Maximum Likelihood. In our setup, the **E-step** correspond to the estimation of the Q-values, while the **M-step** choose the pixel combination that maximizes their probabilities:

1) E-step: In the E-step the derived Q-values at each iteration are used to perform a probability update of the side information as follow

$$P_{ij}^{t+1}(x_i j) = \frac{q_{ij} P_{ij}^t}{\sum_{i=0}^{I} q_{ij}}$$

where i is the ith side information and j is the jth distribution from the ith side information

2) M-step: The best pixel combination with respect to recent probability distribution update is selected according to

$$\hat{x_i} \leftarrow \arg\max P_{ij}(x_{ij})$$

The Learning process is depicted in figure 4

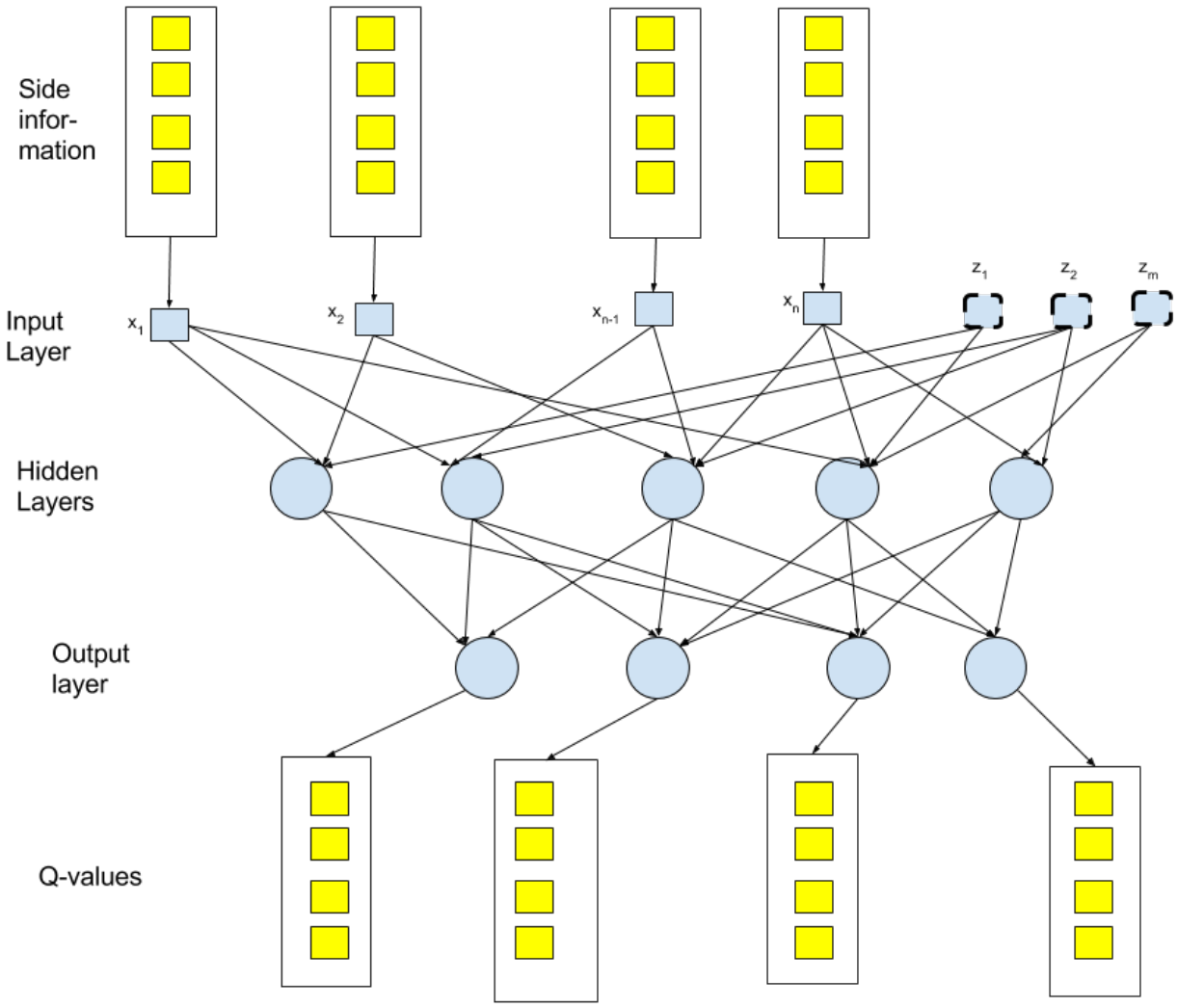

Fig. 3. Setup for Neural Network based Q-learning

IV. Experiment Results and Discussion

As mentioned in section III-C, the size of the Function Approximator's output is related the variance information. Thus a side information can in theory be up to $n = 256$ pixel away from its counterpart. This means that the size of our Q-values is $M \times N \times 256$, where M is the frame height and N is the width. That is, for grayscale QCIF video sequences with $M = 144$ and $N = 176$, as used in our experiments, the size of our Q-values will be $144 \times 176 \times 256 =$ **6 488 064**. That means the Q-values alone require a more than **50 Gigabytes** of RAM memory! As were running the experiment on a personal computer with 8 GB $(4 \times 2GB)$ of RAM, we figured that only around 1024×20 Q-values could fit at a time. We performed therefore a "cherry-picking" procedure for the sake of assessing the effectiveness of our novel algorithm.

Recall that we aim to decode a frame X given its side information Y. To fit the likelihood function, a set of QCIF video sequences were used as training samples. X and Y were divided in blocks and paired up $x_1, x_2, x_3, ..., x_K$ and $y_1, y_2, y_3, ..., y_K$ respectively. The pair (x_i,y_i) was selected as a training sample if their variance was less than 100. That means pixels in x_i is are at most 10 intensities away from pixels in y_i. The length of the block were chosen to 1024 as the width of the Hadamard matrix has to be a power of 2. Recall that the encoder sends z_i and the decoder only has access to y_i and z_i and tries to estimate x_i. Thus $[x_i z_i]$ will be the input to our Function Approximator.

During training phase, we used minibatches of size 1000, while adopting a constant $\epsilon - greedy$ algorithm of 0.1. The input was scaled between -1 and 1 prior entering the Function Approximator. We used 5 hidden layers - with $tanh$ activation function - with 200 nodes per layer. Figures 5 and 6 show the learning ability in terms of rate distortion of the Function Approximator though iterations/episodes. We notice the increase of the PSNR at each episode.

The same "cherry-picking" procedure was used for testing purpose, since we were computationally limited. We tested the algorithm on QCIF video frames for the sequences *Salesman*

		Variance information				
		$\sigma^2 = 20$	$\sigma^2 = 50$	$\sigma^2 = 100$	$\sigma^2 = 150$	$\sigma^2 = 200$
Compression ratio	**1024:3**	38,60	32,60	31,88	29,82	24,73
	1024:10	43,85	37,47	36,50	33,81	29,26
	1024:20	62,66	54,92	45,13	40,12	34,64

TABLE I. TABLE SHOWING THE RATE DISTORTION PERFORMANCE FOR SALESMAN VIDEO SEQUENCE

		Variance information				
		$\sigma^2 = 20$	$\sigma^2 = 50$	$\sigma^2 = 100$	$\sigma^2 = 150$	$\sigma^2 = 200$
Compression ratio	**1024:3**	38,60	32,60	31,88	29,82	24,73
	1024:10	43,85	37,47	36,50	33,81	29,26
	1024:20	62,66	54,92	45,13	40,12	34,64

TABLE II. TABLE SHOWING THE RATE DISTORTION PERFORMANCE FOR HALL VIDEO SEQUENCE

and *Hall Monitor*. Frames blocks with variance information ranging from *around* 10 to *around* 200 were selected. The rate distortion performances are given in tables I and II for the Salesman and Hall video sequences. It is important to notice that even though the Function Approximator is trained on variance information less than 100, we tested our algorithm on variance greater than 100. The reconstruction quality is still good to very good for compression ration 1024:10 and 1024:20, respectively. Compression ratio 1024:3 was also to assess the compression limit. The idea was to check if we could still achieve reasonable distortion by minimizing the number of bits to send.

For each frame block, the encoder generates 3 syndrome coefficients (3 integers = 3 bits) and 1 variance information (1 double =1 bits), 25 blocks and 12 fps. A full frame has 25 blocks. Arguably, compression ration 1024:3, 1024:10 and 1024:20 could thus be comparable to 33, 100 and 200 kbps respectively. This insight shows the high potential of our scheme for **low complexity, low bitrate and low distortion** video coding.

V. CONCLUSION

We have presented a new and practical Video compression scheme based on the Wyner-Ziv framework [3]. The novelty in our scheme lies mostly in the integration of Q-learning in the decoding process. The Wyner-Ziv coding problem has been subject to a great deal of research for at least the past 15 years. The mainstream of in Wyner-Ziv Video Coding has been based on binary codes [4], [12], [13]. Our algorithm is the first really dealing with non-binary codes. A second advantage is its inherent scalability. Previous schemes, such as punctured codes [12] have used different methods for rate control. Non-binary codes have he advantage of reducing the computation complexity at the both at the encoded and decoder, since calculations do not have to be performed at a bit level as in [14] for example.

We also showed that the Wyner-Ziv problem - at least in our case - can be solved using Q-learning algorithm as Likelihood Estimator with a inherent embedding of the EM framework.

However, due our computational limitation, we assessed the algorithm in a ”cherry-picking” manner. The results shown are very good. We arguably showed that our Video Coding scheme was that of **low complexity, low bitrate and low distortion** is specially meaningful for lightweight devices such as surveillance cameras, mobile phones or probably Google^{TM} Watches or Google^{TM} Glasses in the near future when provided with cameras.

Our encoding scheme is of a very low complexity compared to that of motion estimation based video encoders. The bulk of computation is shifted from the encoder to decoder. The decoder is thought to be powerful server station. This is of a great advantage, especially on lightweight devices, such as mobile phones.

REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the h. 264/avc video coding standard,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, 2003.

[2] D. Slepian and J. K. Wolf, “Noiseless coding of correlated information sources,” *Information Theory, IEEE Transactions on*, vol. 19, no. 4, pp. 471–480, 1973. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1055037

[3] A. Wyner and J. Ziv, “The rate-distortion function for source coding with side information at the decoder,” *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, 1976. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1055508

[4] A. Aaron, R. Zhang, and B. Girod, “Wyner-ziv coding of motion video,” pp. 240–244, 2002.

[5] R. Puri, A. Majumdar, and K. Ramchandran, “Prism: a video coding paradigm with motion estimation at the decoder,” *Image Processing, IEEE Transactions on*, vol. 16, no. 10, pp. 2436–2448, 2007.

[6] J.-P. Kouma and H. Li, *Large-Scale Face Image Retrieval: A Wyner-Ziv Coding Approach*. InTech, 2011, ch. 2, p. 29. [Online]. Available: http://www.intechopen.com/books/new-approaches-to-characterization-and-recognition-of-faces/large-scale-face-image-retrieval-a-wyner-ziv-coding-approach

[7] C. Kim, H.-H. Shih, and C.-C. J. Kuo, “Fast h. 264 intra-prediction mode selection using joint spatial and transform domain features,” *Journal of Visual Communication and Image Representation*, vol. 17, no. 2, pp. 291–310, 2006.

[8] Z. Liu, L. Li, Y. Song, S. Li, S. Goto, and T. Ikenaga, “Motion feature and hadamard coefficient-based fast multiple reference frame motion estimation for h. 264,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 5, pp. 620–632, 2008.

[9] A. P. Dempster, N. M. Laird, and D. B. Rdin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38, 1977.

[10] C. J. C. H. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, University of Cambridge England, 1989.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[12] J. Sun and H. Li, “Trellis-based reconstruction in wyner-ziv codec for video,” in *IASTED International Conference on Internet and Multimedia Systems and Applications*, 2005.

[13] A. Aaron and B. Girod, “Wyner-ziv video coding with low-encoder complexity,” in *Proc. Picture Coding Symposium*, 2004.

[14] A. Aaron, S. Rane, E. Setton, and B. Girod, “Transform-domain wynerziv codec for video,” in *in Proc. SPIE Visual Communications and Image Processing*, 2004, pp. 520–528.

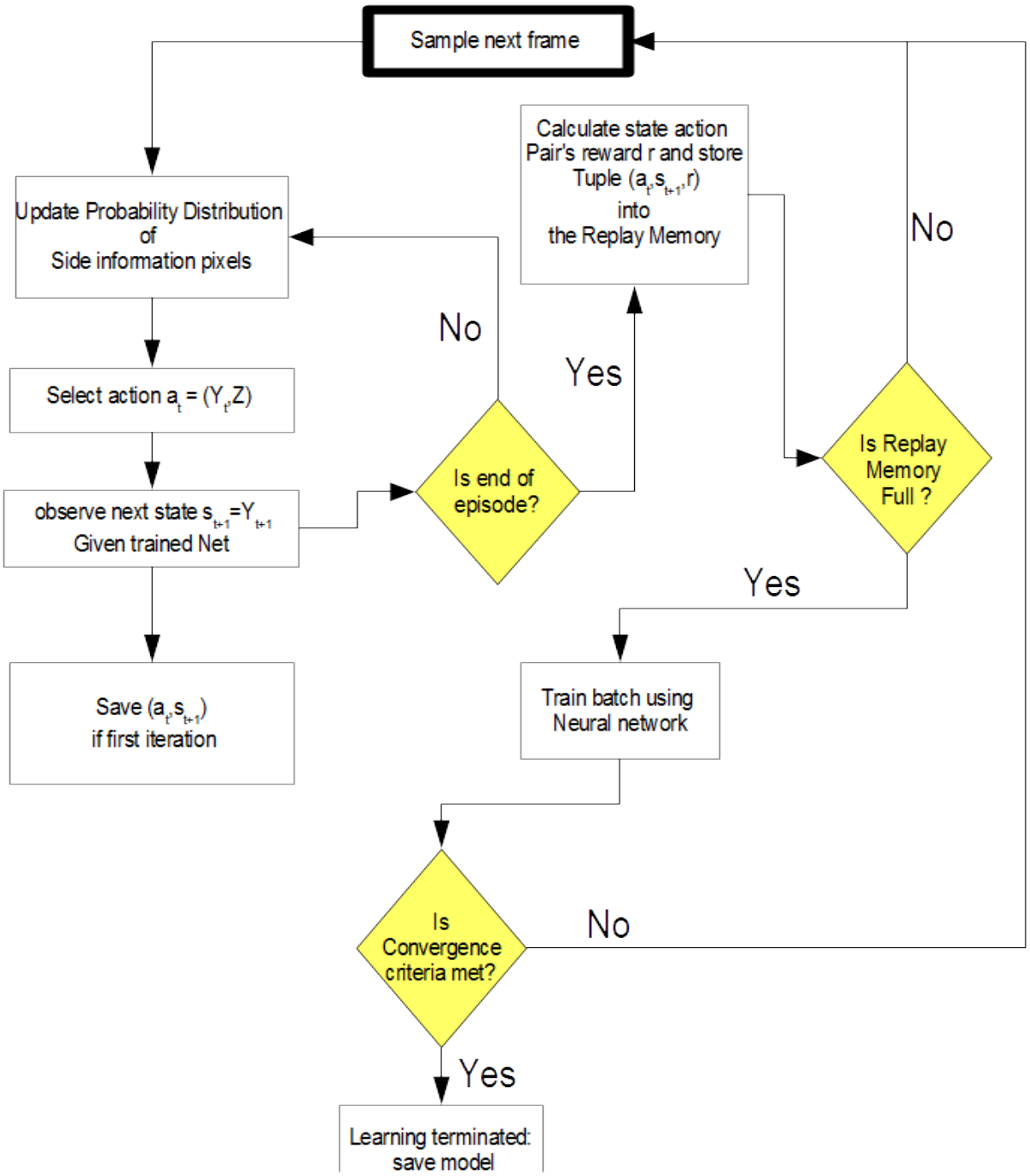

Fig. 4. Learning process of our video codec

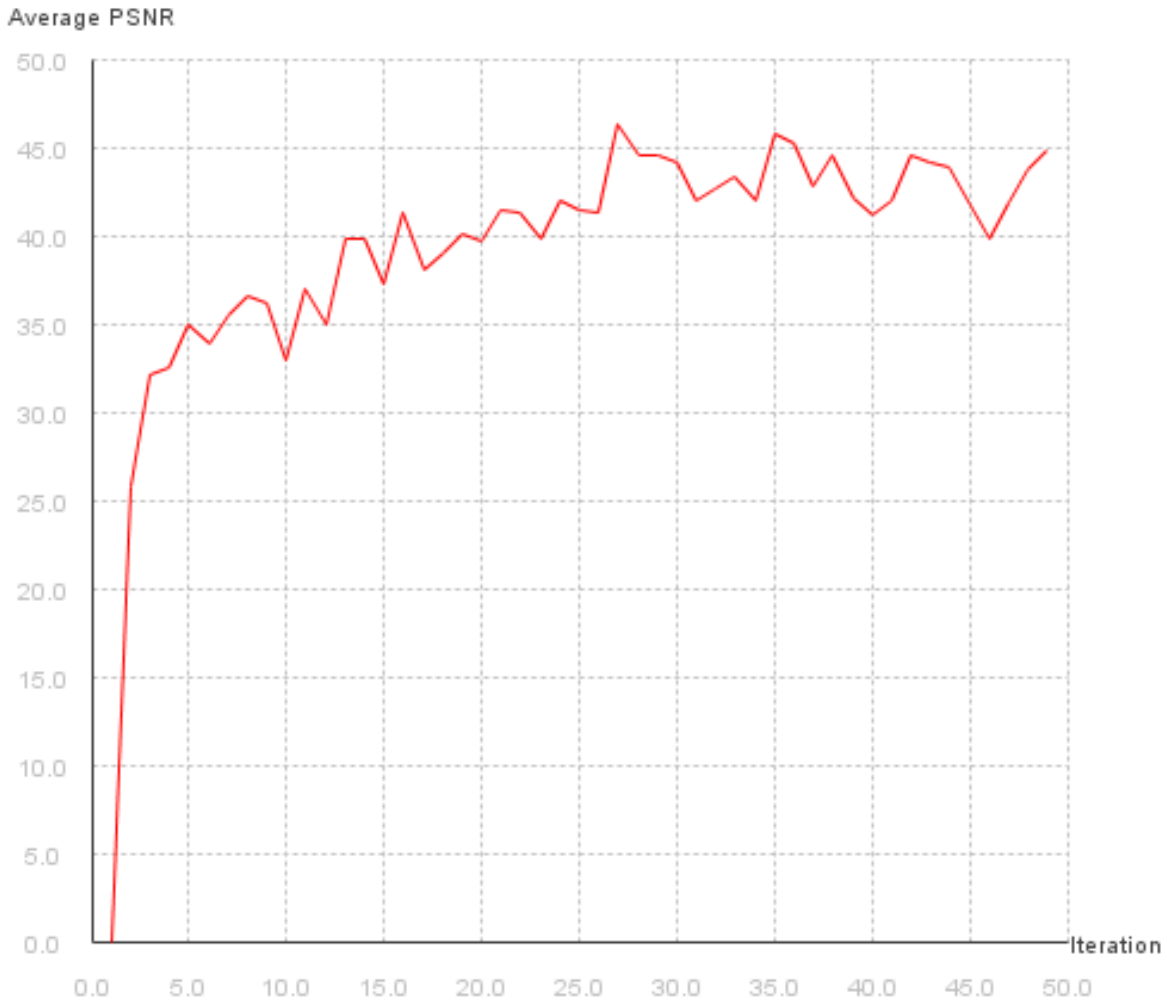

Fig. 5. Average Learning ability of our Function Approximator through episodes: *Compression ratio 1024:3*

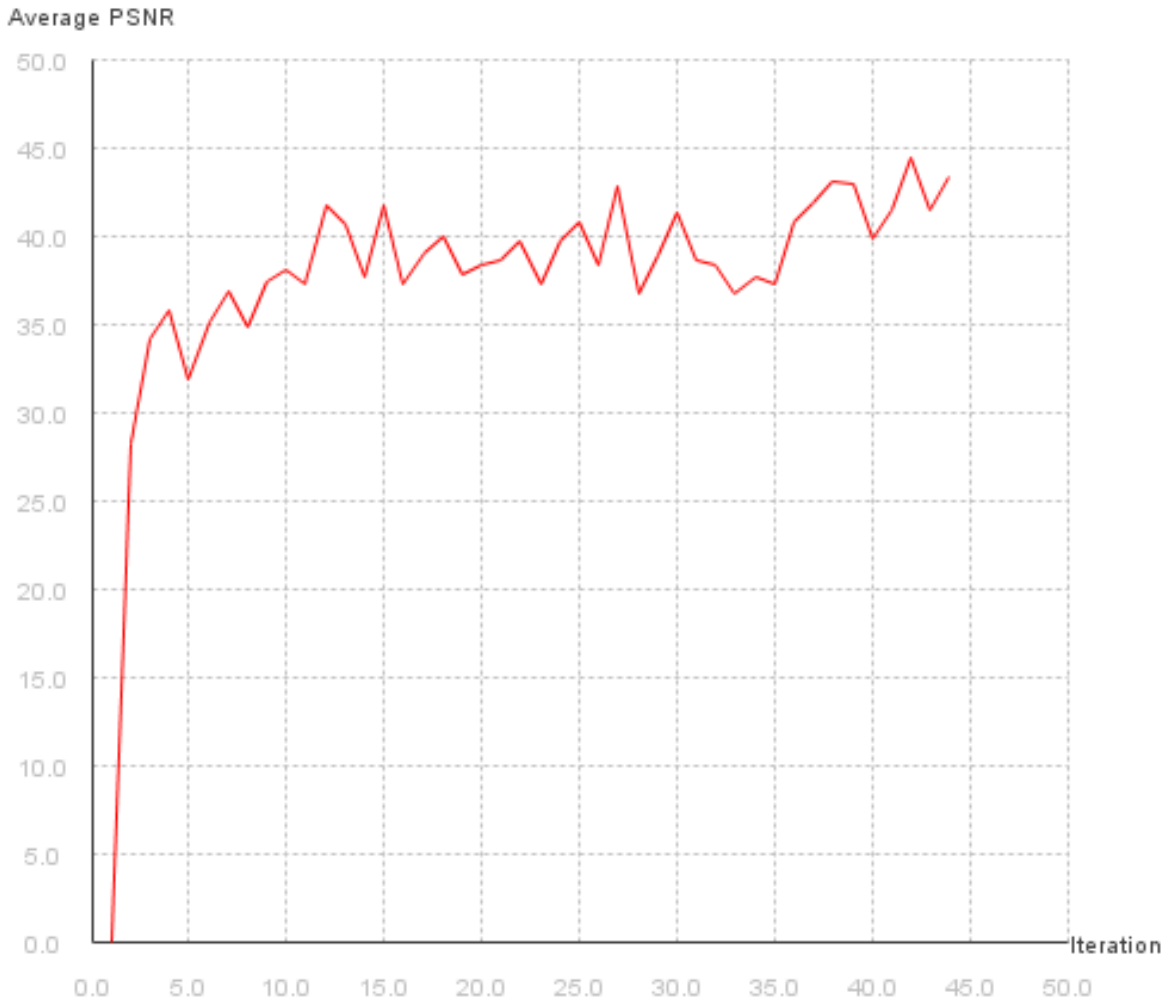

Fig. 6. Average Learning ability of our Function Approximator through episodes: *Compression ratio 1024:5*