A Study on Data-Driven Probability Estimator Design for Video Coding

Heiner Kirchhoffer*, Christian Rudat*, Michael Schäfer*, Jonathan Pfaff*, Heiko Schwarz*†, Detlev Marpe* and Thomas Wiegand*‡

*Fraunhofer Heinrich Hertz Institute, Berlin, Germany

†Institute for Computer Science, Free University of Berlin, Berlin, Germany

‡Chair of Media Technology, Technical University of Berlin, Berlin, Germany

Abstract

Multi-hypothesis probability estimation heavily influences the coding efficiency of the Versatile Video Coding (VVC) standard. In this paper, data-driven optimization techniques are used to push the limits of the VVC probability estimator. This includes an increase of the number of hypotheses, a generalization based on a weighted sum of previously observed binary symbols, as well as a combination with a neural network. Experimental results show coding efficiency gains for all of the investigated architectures.

1 Introduction

The state-of-the-art video compression standard VVC [1] features a sophisticated entropy coding stage known as Context-based Adaptive Binary Arithmetic Coding (CABAC) [2]. Here, the process of encoding (or decoding) syntax elements like transform coefficients, motion vectors, etc. can be structured into four separate stages: binarization, context model selection, probability estimation, and arithmetic encoding (or decoding). The binarization stage converts the respective syntax elements into a sequence of binary symbols (so-called *bins*). The context modeling stage associates each bin with one of a number of predefined context models. In the probability estimation stage, each context model assigns a probability estimate p of the bin being equal to 1 to each associated bin. The arithmetic coder encodes (or decodes) each bin with approximately $-\log_2(p)$ bits on average in the case the bin is equal to 1, or with approximately $-\log_2(1-p)$ bits on average in the case the bin is equal to 0.

The aim of this paper is to study the individual components of the probability estimation scheme of VVC in a data-driven way in order to explore the limits of the achievable compression efficiency. These studies were conducted using an extension of the VVC reference software (i.e. the Enhanced Compression Model version 3.1 [3]) which includes additional coding tools.

The remainder of the paper is structured as follows. An overview of the probability estimation in VVC is given in Section 2. Based on this, several data-driven approaches for probability estimation are developed in Section 3. Furthermore, Section 4 gives an overview of the model training and evaluation setup and experimental results are discussed. The paper concludes in Section 5.

2 Review of probability estimation in VVC

Context models in VVC carry out probability estimation for the associated bins in a backward-adaptive manner. An in-depth discussion can be found in [2]. Let $x(t)$ with $t \in \{1, \dots, M\}$ be the sequence of M bins that are associated with one context model. The VVC standard specifies for each context model two state variables pStateIdx0 and pStateIdx1 which will be denoted $S_1(t)$ and $S_2(t)$, respectively, in this paper, with $t \in \{1, \dots, M\}$. When a context model is initialized, for example, when the encoding or decoding of a picture begins, $S_1(1)$ and $S_2(1)$ are set to initial values. For encoding of a bin $x(t)$, a probability estimate $\bar{p}(t)$ of the bin being equal to 1 is calculated from the state variables according to $\bar{p}(t) = (p_1(t) + p_2(t))/2$ where $p_i(t) = S_i(t)/2^{b_i}$ with b_i being the number of bits to represent $S_i(t)$ as unsigned integer. The probability estimate $\bar{p}(t)$ is used for binary arithmetic encoding (or decoding). Note that $S_1(t)$ and $S_2(t)$ are represented as 10 and 14 bit unsigned integers, respectively, i.e., $b_1 = 10$ and $b_2 = 14$. Whenever a bin is encoded (or decoded), the state variables are updated according to

$$S_i(t+1) = S_i(t) - \left\lfloor \frac{S_i(t)}{2^{r_i}} \right\rfloor + \left\lfloor \frac{k_i \cdot x(t)}{2^{r_i}} \right\rfloor, \tag{1}$$

where $k_i = 2^{b_i} - 1$ and where r_i is an adaptation rate parameter. Note that the division by 2^{r_i} and the floor operation are efficiently implemented as a bit-shift operation in VVC. The choice of b_i controls the trade-off between modeling accuracy and memory consumption. The rounding error in (1) caused by the floor operators becomes less relevant when b_i is increased. According to (1), the probability estimate of a context model is updated based on the previously encoded bin in a backward-adaptive manner and the adaptation rate r_i controls the degree of change that one update step causes. For example, large values of r_i only slightly change S_i during one update while small values of r_i change S_i by a more significant amount. The adaptation rates are constant values that only depend on the context model and they fulfill the conditions $2 \leq r_i \leq 9$ (for $i \in \{1, 2\}$) and $r_2 \geq r_1 + 3$. It may, however, be beneficial for the compression efficiency to relax these conditions and allow larger or smaller values for r_i.

3 Data-driven probability estimator design

In the sequel, generalizations of the aformentioned probability estimation scheme of VVC are studied. The parameters of the considered estimators are derived in a data-driven way.

3.1 Data-driven hypotheses weighting

As discussed in Section 2, each context model uses a weighted sum of two hypotheses S_1 and S_2 for probability estimation which differ in the associated adaptation rates r_1 and r_2. The idea of data-driven hypotheses weighting (DHW) is to generalize this approach by using more than two hypotheses. For this purpose, b_i shall be assumed to be sufficiently large so that the rounding caused by the floor operation in (1) can be neglected. Furthermore, $S_i(t)$ shall be substituted by $p_i(t) \cdot 2^{b_i}$ and $k_i/2^{b_i}$ is approximated by 1 in (1) which yields

$$p_i(t+1) = p_i(t) - \frac{p_i(t)}{2^{r_i}} + \frac{x(t)}{2^{r_i}}. \tag{2}$$

with $p_i(1)$ being a predefined initial probability estimate. Next, a so-called inertia parameter is defined as $\alpha_i = 1 - 2^{-r_i}$. Substituting r_i with α_i in (2) then yields

$$p_i(t+1) = \alpha_i \cdot p_i(t) + (1 - \alpha_i) \cdot x(t), \tag{3}$$

which can also be written in a non-recursive way as

$$p_i(t+1) = \alpha_i^t \cdot p_i(1) + (1 - \alpha_i) \cdot \sum_{j=0}^{t-1} x(t-j) \cdot \alpha_i^j. \tag{4}$$

Let $u_i(t)$ be a probability estimate according to (4) for which $p_i(1)$ is assumed to be equal to 0, i.e.,

$$u_i(t+1) = (1 - \alpha_i) \cdot \sum_{j=0}^{t-1} x(t-j) \cdot \alpha_i^j. \tag{5}$$

Correspondingly, let $v_i(t)$ be a probability estimate according to (4) for which $p_i(1)$ is assumed to equal 1, i.e.,

$$v_i(t+1) = \alpha_i^t + u_i(t+1). \tag{6}$$

Then, the probability estimate satisfies

$$p_i(t) = p_i(1) \cdot v_i(t) + (1 - p_i(1)) \cdot u_i(t). \tag{7}$$

In other words, $p_i(t)$ can be expressed as a weighted sum of probability estimates $v_i(t)$ and $u_i(t)$ where the weighting factors are $p_i(1)$ and $1 - p_i(1)$, respectively.

Next, a probability estimator that consists of B weighted hypotheses according to (4) with different inertia parameters α_i and different initial probabilities $p_i(1)$ shall be defined as

$$\hat{p}(t) = \sum_{j=1}^{B} w_j \cdot p_j(t), \tag{8}$$

with $w_j \geq 0$ for all $j \in \{1, \ldots, B\}$ and $\sum_{j=1}^{B} w_i = 1$. Substituting (7) in (8) yields

$$\hat{p}(t) = \sum_{j=1}^{B} w_j \cdot (p_j(1) \cdot v_j(t) + (1 - p_j(1)) \cdot u_j(t)). \tag{9}$$

Note that each hypothesis j may have an individual value for the initial probability $p_j(1)$. However, it seems to be reasonable to initialize each hypothesis with the same initial value, $\forall j$, $p_j(1) = p(1)$. Based on (9), a probabity estimate that only uses one $p(1)$ can be defined as

$$\hat{p}'(t) = p(1) \cdot \sum_{j=1}^{B} w_j \cdot v_j(t) + (1 - p(1)) \cdot \sum_{j=1}^{B} w_j \cdot u_j(t). \tag{10}$$

Let

$$\hat{p}''(t) = p(1) \cdot \sum_{j=1}^{B} \gamma_j \cdot v_j(t) + (1 - p(1)) \cdot \sum_{j=1}^{B} \delta_j \cdot u_j(t). \tag{11}$$

Note that (11) is identical to (10) if $\gamma_j = \delta_j = w_i$. Equation (11) is used as basis for the DHW architecture with $p(1)$, γ_j, and δ_j being the parameters that shall be trained. Note that

$$\sum_{j=1}^{B} \gamma_j = 1, \quad \sum_{j=1}^{B} \delta_j = 1, \quad \text{and} \quad 0 \leq p(1) \leq 1 \tag{12}$$

must be satisfied when training the parameters. The softmax function can be used to achieve this for γ_j and δ_j while the sigmoid function is suitable to achieve this for $p(1)$ as explained in the following. Let

$$\gamma_j = e^{\gamma'_j} / \sum_{k=1}^{B} e^{\gamma'_k}, \quad \delta_j = e^{\delta'_j} / \sum_{k=1}^{B} e^{\delta'_k}, \quad \text{and} \quad p(1) = \frac{1}{1 + e^{-\mu}}. \tag{13}$$

Note that no such restrictions as defined in (12) apply to parameters γ'_j, δ'_j, and μ which makes them suitable to be used as trainable parameters of the architecture. Consequently, the DHW architecture contains $2B + 1$ trainable parameters. In this paper, an exemplary configuration with B set to 14 and with $r_i = i$ for $i \in \{1, \ldots, B\}$ will be used for all experiments.

3.2 Data-driven weighting of latest bins

As can be seen from (4), for each hypothesis, the probability estimate $p_i(t + 1)$ is a weighted sum of the initial probability estimate $p_i(1)$ and all observed bins $x(j)$ for all $j \in \{1, \ldots, t\}$. The idea of data-driven weighting of latest bins (DWLB) is to directly train these weights instead of calculating them from the inertia parameter α_i. To motivate this idea further, (4) is substituted in (8) which yields

$$\hat{p}(t + 1) = \sum_{i=1}^{B} w_i \cdot \alpha_i^t \cdot p_i(1) + \sum_{j=0}^{t-1} x(t - j) \cdot \sum_{i=1}^{B} w_i \cdot (1 - \alpha_i) \cdot \alpha_i^j. \tag{14}$$

Equation (14) shows that $\hat{p}(t+1)$ is a weighted sum of the initial probability estimates $p_i(1)$ and and all observed bins $x(j)$ for all $j \in \{1, \ldots, t\}$. Consequently, only one set of weights for all previously observed bins plus one initial probability value is sufficient to implement an arbitrary large number of weighted hypotheses according to (8). Therefore, DWLB can be considered as a generalization of DHW.

It is, however, challenging to train weights for a previous bins as their number is not constant. Hence, an approximation which uses a fixed number of previous bins only shall be derived as follows. Let

$$\tilde{x}(t) = \begin{cases} x(t) & \text{if } t > 0, \\ p(1) & \text{otherwise}. \end{cases} \tag{15}$$

Note that for $\alpha \neq 1$,

$$\alpha^t = (1-\alpha) \cdot \sum_{j=t}^{\infty} \alpha^j \tag{16}$$

holds. Substituting (15) and (16) in (4) and neglecting index i yields

$$p(t+1) = (1-\alpha) \cdot \sum_{j=0}^{\infty} \tilde{x}(t-j) \cdot \alpha^j = \sum_{j=0}^{\infty} \tilde{x}(t-j) \cdot \phi_j \tag{17}$$

where $\phi_j = (1-\alpha) \cdot \alpha^j$. This equation can be interpreted as follows. The probability estimate is a weighted sum of all previously observed bins and infinitely many further bins which are all equal to $p(1)$. Note that (17) defines infinitely many parameters ϕ_j which is difficult to implement in practice. As a workaround, the infinite sum in (17) is replaced with a finite sum of sufficiently many elements. Note that the weights ϕ_j are exponentially decaying with increasing index j. In particular, 'former' bins have a smaller weight than 'more recent' bins. Therefore, it is reasonable to only consider a fixed number of D latest bins $\tilde{x}(t-j)$ for all $0 \leq j < D$ and still get a sufficiently accurate probability estimate. Let

$$\tilde{p}(t+1) = p(1) \cdot \theta + \sum_{j=0}^{D-1} \tilde{x}(t-j) \cdot \phi_j, \quad \text{with} \quad \theta = \sum_{j=D}^{\infty} \phi_j. \tag{18}$$

In (18), only up to D previous bins are considered. All other bins are assumed to equal $p(1)$. If D is sufficiently large, (18) can be used as a good approximation of (17). The contribution of the neglected bins to the probability estimate (17) does obviously not exceed θ. The aim of DWLB is to train parameters $p(1)$, θ and ϕ_j for $j < D$. Note that

$$\theta + \sum_{j=0}^{D-1} \phi_j = 1 \quad \text{and} \quad 0 \leq p(1) \leq 1 \tag{19}$$

must be satisfied when training the parameters. The softmax function can be used over the set $\{\theta, \phi_j \mid 0 \leq j < D\}$ and the sigmoid function can be used for $p(1)$ to satisfy (19) as follows. Let

$$\theta = \frac{e^{\theta'}}{e^{\theta'} + \sum_{k=0}^{D-1} e^{\phi'_k}}, \quad \phi_j = \frac{e^{\phi'_j}}{e^{\theta'} + \sum_{k=0}^{D-1} e^{\phi'_k}}, \quad \text{and} \quad p(1) = \frac{1}{1+e^{-\mu}}. \tag{20}$$

Parameters θ', ϕ'_j, and μ are the trainable parameters of the architecture. Hence, the number of trainable parameters for DWLB is $D+2$. In an experimental configuration,

D can be chosen such that θ does not exceed a predefined threshold. A value of 2048 for D turned out to be sufficiently large for the experiments described in this paper and is thus used for all experiments. Vice versa, larger values of D did not further improve the results.

3.3 Constant one and zero inputs for DHW and DWLB

The minimum value for the probability estimate of the DHW and the DWLB architecture can approach 0 arbitrarily close while the maximum value can approach 1 arbitrarily close. The arithmetic coding engine of VVC is, however, limited in its capability to properly reflect extreme probabilities (very close to 1 or 0). Therefore, it makes sense to avoid extreme probabilities in the estimation. One way to achieve this is to define maximum and minimum values for the probability estimate and then apply a clipping operation. However, an experimental evaluation showed that it may be beneficial to adapt the clipping limits to the context models. In the following, an alternative technique for avoiding extreme probabilities is presented. Given a probability estimate $p(t)$ let

$$\ddot{p}(t) = a_0 \cdot p(t) + a_1 \tag{21}$$

be the probability estimate with limited maximum and minimum value. Three trainable parameters a'_0, a'_1, and a'_2 are defined from which a_j is derived according to the softmax function

$$a_j = e^{a'_j} / \sum_{k=0}^{2} e^{a'_k}. \tag{22}$$

As can be seen from (21), a_1 corresponds to the minimum value for the probability estimate $\ddot{p}(t)$. Since $a_0 + a_1 + a_2 = 1$, the maximum value for the probability estimate $\ddot{p}(t)$ is equal to $1 - a_2$. This increases the number of trainable parameters by 3 for each architecture it is applied to. All experimental results for DHW and DWLB in this paper are created using this constant one and zero concept.

3.4 Training of adaptation rates

The DHW architecture is able to train parameters for weighting pre-defined hypotheses while the DWLB architecture trains weights for previous bins. A further interesting architecture would be one that directly trains the adaptation rates. The DWLB architecture can be modified to directly train parameters α_i instead of the individual weights for the latest bins. The only required change is to train α instead of θ and ϕ_j. Doing so would correspond to one hypothesis with inertia parameter α. The sigmoid function is used to ensure that $0 < \alpha < 1$. I.e., a trainable variable α' is defined from which α is derived according to $\alpha = 1/(1 + e^{-\alpha'})$, which is the sigmoid function. In order to increase the number of hypotheses trained in this way, the architecture is duplicated for each additional desired hypothesis, each hypothesis being associated with an α'_i and having an associated probability estimate $\tilde{p}_i(t)$ according to (18). A weighted sum is defined as

$$\tilde{p}'(t) = \sum_{i=1}^{G} w_i' \cdot \tilde{p}_i(t) \tag{23}$$

with G being the number of hypotheses. The resulting architecture shall be denoted data-driven training of alpha (DTA) and $\tilde{p}'(t)$ is the associated probability estimate. In addition to trainable parameters α_i', also the parameters w_i' need to be trained and the sum over weights w_i' needs to equal 1 which is achieved by defining corresponding weights $\hat{w}_i$ so that the parameters w_i' are derived from $\hat{w}_i$ using the softmax function.

Two configurations are tested in this paper: DTA_2 and DTA_3 with 2 and 3 hypotheses, respectively. The constant one and zero concept of Section 3.3 is also used for DTA which leads to $2G + 3$ trainable parameters.

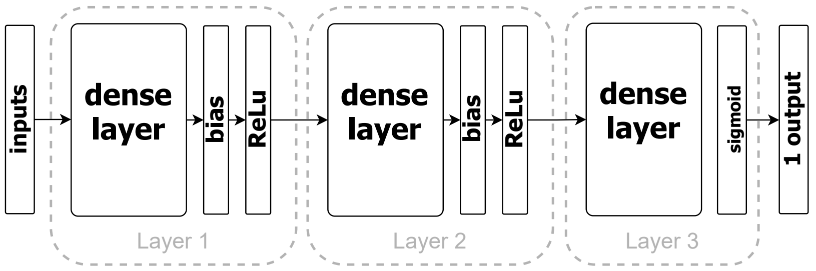
3.5 Combination of neural network-based weighting of latest bins and hypotheses weighting

Building up on ideas presented in Sections 3.1, 3.2 and 3.3, another approach denoted neural network-based weighting of latest bins (NNWLB) was examined. The final probability estimation $\check{p}(t)$ is given as the weighted sum of two probability hypotheses $\bar{p}(t)$ and $\dot{p}(t)$

$$\check{p}(t) = \beta \cdot \bar{p}(t) + (1 - \beta) \cdot \dot{p}(t) \tag{24}$$

with $0 \leq \beta \leq 1$ and β being a trainable parameter. Furthermore, the hypothesis $\bar{p}(t)$ is not trainable and may itself be, for example, a weighted sum of state-based hypotheses like (8). Finally, the hypothesis $\dot{p}(t)$ is the output of a trainable neural network with the latest K bins as inputs. In essence, a pre-existing probability estimation $\bar{p}(t)$ and the latest K bins $x(t)$ are fed into a system that yields a final probability estimation $\check{p}(t)$ which — if trained correctly — should at least match the coding efficiency achieved through $\bar{p}(t)$ if not even further improve it. In an

Figure 1: Illustration of the NN path of NNWLB yielding $\dot{p}(t)$



experimental configuration, the network architecture was chosen as follows. The value $\bar{p}(t)$ is fed with the output of a pre-trained DHW probability estimation stage ($2B+4$ trainable parameters including constant one/zero) as described in Section 3.1 and 3.3. During the training of NNWLB, the parameters of the DHW path are not trainable and therefore fixed. The second hypothesis $\dot{p}(t)$ is generated by a fully connected, multi-layer neural network, as illustrated in Figure 1. The input layer is formed by concatenating the latest K bins with a constant one and a constant zero (see Section 3.3) resulting in $K + 2$ input nodes. Now, two hidden dense layers follow each featuring N nodes, N biases and ReLU activations ($(K + 2) \cdot N + N$ and $N \cdot N + N$ trainable parameters). The final layer has just one output followed by a sigmoid activation that ensures the output value $\dot{p}(t)$ is bound by zero and one (N trainable parameters). Finally, the weighted average of both hypotheses as described in (24) was implemented using two trainable parameters $\bar{c}$ and $\dot{c}$ that are normalized by applying a softmax function. In contrast to DWLB a value of $K = 200$ for the number of latest bins seems to be a good (experimentally derived) trade-off. This results in 202 nodes at the input layer of the neural-network path. In this paper, two specific architectures NNWLB_10 and NNWLB_100 are examined with $N = 10$ and $N = 100$, respectively.

4 Experimental Evaluation

The Enhanced Compression Model 3.1 (ECM) software [3] is used for the generation of training and evaluation data is used for all experiments. It contains several enhanced compression tools beyond VVC that further improve the compression efficiency. The probability estimation stage features two equally weighted hypotheses per context model (as in VVC) where the associated adaptation rates r_i are optimized for each context model. For all simulations, b_i is set to 63 in order to minimize rounding effects. Bjøntegaard delta bitrate (BD rate) [4] is used for assessing the compression efficiency. All BD rates are based on the PSNR_YUV = (6 · PSNR_Y + PSNR_U + PSNR_V)/8 as specified in [5].

4.1 Training and validation bitstream generation

Two disjoint sets of video sequences are used. All validation bitstreams are based on the video sequences as specified in the ECM common test conditions (CTC) of [6]. For training set generation, video sequences of the BVI-DVC testset [7] are employed. Both sets are encoded using encoder settings as specified in [6] in intra only (AI), random access (RA), and the low delay (LB) setting. Each video sequence is encoded using the four quantization parameter (QP) values 22, 27, 32, and 37. The BVI-DVC testset consists of 800 video sequences which results in 3200 bit streams while the validation set only consists of 22 video sequences (sequences from classes F and TGM are not used) which corresponds to 88 bit streams.

4.2 Re-training of context model initializations

In order to create a defined starting point for all simulations, the parameters of all context models in ECM have been re-trained [8] based on a set of BVI-DVC bitstreams that were encoded using the common test conditions. This re-training already creates an overall BD rate reduction of 0.07% for AI, 0.06% for RA, and 0.05% for LB on the CTC set. ECM with these re-trained context model initializations is used as a reference for all simulations and for the generation of training and validation sets.

Table 1: BD rate comparison of the various architectures (smaller values correspond to a better compression). Results for NNWLB_10 and NNWLB_100 are not available for RA and LB due to extreme encoding runtimes.

Architecture (number of trainable parameters)	Transcoding			Estimation and coding		
	AI	RA	LB	AI	RA	LB
DHW (32)	-0.07%	-0.13%	-0.16%	-0.10%	-0.19%	-0.32%
DWLB (2053)	-0.08%	-0.14%	-0.17%	-0.11%	-0.22%	-0.35%
DTA_2 (7)	-0.06%	-0.11%	-0.14%	-0.06%	-0.16%	-0.22%
DTA_3 (9)	-0.07%	-0.12%	-0.15%	-0.08%	-0.18%	-0.28%
NNWLB_10 (2184)	-0.10%	-0.17%	-0.19%	-0.13%		
NNWLB_100 (30534)	-0.11%	-0.18%	-0.20%	-0.14%		

4.3 Model training setup

The described model architectures have been implemented using TensorFlow [9] version 1.15.5 and Python [10] version 3.6. The Adam optimizer [11] was employed for training. The cost function is $-\log_2(p(t-1))$ if a bin $x(t)$ is equal to 1 and $-\log_2(1-p(t-1))$ otherwise, where $p(t-1)$ is the probability estimate of the respective model architecture. Each context model uses an individual instance of the respective model architecture. Note that VVC uses initial probabilities that depend on the quantization parameter (QP) which is an integer between 0 and 63, inclusively. To achieve a similar configuration, the initial probability values $p(1)$ are trained independently for each QP value. The three parameters associated with the constant 1 and 0 as described in 3.3 are also trained independently for each QP value while all other parameters are trained jointly over all QPs.

4.4 Experimental results

The BD rates are shown in Table 1 for AI, RA, and LB. Results in category 'estimation and coding' are produced with the ECM software where the respective model architecture has been implemented. As the ECM software employs Rate-Distortion-optimization, the probability estimation also has an impact on the encoder decisions. In contrast, results in the 'transcoding' category are produced without changing the encoder decisions. This can be interpreted as transcoding where only the probability estimator of existing bitstreams are exchanged with the respective architecture. As can be seen in Table 1, 'estimation and coding' gains are in general larger than the 'transcoding' gains. Additionally, the numbers of trainable parameters per context model are shown in Table 1. The DWLB architecture has a slightly higher coding efficiency than DHW. This is expected as DWLB is a generalization of DHW (see Section 3.2). A small coding gain can be achieved with the DTA_2 architecture although the VVC estimator (which is used as reference for the BD rates) uses 2 hypotheses as well. With 3 hypotheses as used in DTA_3, only a slight additional gain is achieved and it has a performance comparable to DHW. A further increase of the number of hypotheses of DTA did not lead of an improvement of the coding efficiency. A more substantial increase of the gain can be achieved with the NNWLB architecture. Note that the computational complexity and the memory requirements of DWLB and NNWLB are significantly higher than for DHW and DTA. Consequently, the trade-off between the computational complexity, the memory requirements, and the coding gain is much more attractive for DHW and DTA. In particular, DTA_2 might be an interesting alternative for the probability estimator of VVC.

5 Conclusion

Various probability estimator architectures are derived and optimized by employing a data-driven approach. An increase of the number of hypotheses, as present in the DHW and the DTA architecture, causes an increase of the compression efficiency. The more general DWLB architecture adds a further, slight improvement. More coding gain can be achieved when combining the DHW approach with a neural network. This suggests that more complex neural network architectures may be an interesting topic for future work.

6 References

[1] ITU-T, *Rec. H.266: Versatile video coding (VVC)*, International Telecommunication Union, Apr. 2022.

[2] Heiko Schwarz et al., "Quantization and entropy coding in the versatile video coding (vvc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3891–3906, 2021.

[3] Muhammed Coban et al., *Algorithm description of Enhanced Compression Model 3 (ECM 3)*, Joint Video Experts Team (JVET), JVET-X2025, 24th Meeting, Oct. 2021.

[4] Gisle Bjøntegaard, *Improvements of the BD-PSNR model*, ITU-T Q6/16, Video Coding Experts Group (VCEG), VCEG-AI11, Berlin, Germany, 16-18 Jul. 2008.

[5] ITU-T, *Working practices using objective metrics for evaluation of video coding efficiency experiments*, Technical Paper ITU-T HSTP-VID-WPOM, Jul. 2020.

[6] Marta Karczewicz and Yan Ye, *Common test conditions and evaluation procedures for enhanced compression tool testing*, Joint Video Experts Team (JVET), JVET-X2017, 24th Meeting, Oct. 2021.

[7] Di Ma, Fan Zhang, and David Bull, "Bvi-dvc: A training database for deep video compression," *IEEE Transactions on Multimedia*, pp. 1–1, 2021.

[8] Frank Bossen, Jie Dong, and Heiner Kirchhoffer, *Description of Core Experiment 1 (CE1): CABAC Initialization*, Joint Video Experts Team (JVET), JVET-N1021, 14th Meeting, Mar. 2019.

[9] Martín Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[10] Guido van Rossum and Python Dev Team, *Python 3.6 Language Reference*, Samurai Media Limited, London, GBR, 2016.

[11] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.