Very Low Complexity Convolutional Neural Network for Quadtree Structures

Adrian Caruana **Teresa Vidal-Calleja**
Centre for Autonomous Systems at the Faculty of Engineering and IT,
University of Technology Sydney
adrian.caruana@student.uts.edu.au, teresa.vidalcalleja@uts.edu.au

Abstract

In this paper, we present a Very Low Complexity Convolutional Neural Network (VLC-CNN) for the purpose of generating quadtree data structures for image segmentation. The use of quadtrees to encode images has applications including video encoding and robotic perception, with examples including the Coding Tree Unit in the High Efficiency Video Coding (HEVC) standard and Occupancy Grid Maps (OGM) as environment representations with variable grid-size. While some methods for determining quadtree structures include brute-force algorithms or heuristics, this paper describes the use of a Convolutional Neural Network (CNN) to predict the quadtree structure. CNNs traditionally require substantial computational and memory resources to operate, however, VLC-CNN exploits downsampling and integer-only quantised arithmetic to achieve minimal complexity. Therefore, VLC-CNN's minimal design makes it feasible for implementation in real-time or memory-constrained processing applications.

1 Introduction

Quadtrees are an effective data structure for representing images (and 2D occupancy and elevation maps, which can be treated as images) due to their inherent nature to efficiently store spatial information. They facilitate the allocation of additional resources for representing edges or high-contrast areas of an image and fewer resources for spatially homogeneous information. Therefore, they are used extensively in image representation applications, for example in the Coding Tree Unit (CTU) in the High Efficiency Video Coding (HEVC) standard. HEVC implements the Rate Distortion Optimisation (RDO) process for generation of each CTU - an exhaustive algorithm whereby a cost (RD-Cost) is calculated for each of the 85 possible Coding Units (CU) to optimise image quality and encoded bit-cost. The CU with the minimum RD-Cost is expressed as

$$\vec{p}_o = \arg\min_{\vec{p}}\{D(\vec{p}) + \lambda \cdot R(\vec{p})\}\,, \tag{1}$$

where $\vec{p}$ represents the candidate CU, $D(\vec{p})$ and $R(\vec{p})$ represent the distortion and rate-cost respectively, λ is the Lagrangian multiplier, and $\vec{p}_o$ is the CU with the minimum RD-cost. In order to obtain $D(\vec{p})$ and $R(\vec{p})$, each candidate $\vec{p}$ must first be encoded to yield $R(\vec{p})$, and then decoded and compared to the reference data to yield $D(\vec{p})$ - and so this leads to substantial RDO complexity.

Occupancy Grid Maps (OGM) are the most common method for environment representation and mapping in robotics [Thrun, 2003; Moravec and Elfes, 1985]. However, where a high degree of accuracy is required in environmental mapping, increased resolution results in a large increase in memory consumption and computational complexity, and a variable-resolution map aptly exploits the high-accuracy of modern sensor technology. This is due to the fixed cell size in OGMs. This has been addressed by using a variable cell size, where larger cells are used to represent spatially homogenous parts of the environment, thereby reducing memory and computational burden without compromising the maps resolution.

Quadtrees have been shown to be an effective data structure for representing spatial information in OGMs [Kraetzschmar *et al.*, 2004; Einhorn *et al.*, 2011; Wurm *et al.*, 2010; Li and Ruichek, 2013]. [Kraetzschmar *et al.*, 2004] introduces probabilistic quadtree for OGM which is generated off-line. [Wurm *et al.*, 2010] also used an off-line, probabilistic approach to model 3D environments using octrees. [Einhorn *et al.*, 2011] employed an Nd-tree (a *d*-dimensional generalisation of an quadtree/octree) to create OGMs at variable resolutions. [Li and Ruichek, 2013] extended the existing tree-based grid mapping techniques from off-line range sensor based to an online stereo-vision system based.

The objective for this paper is to predict if an image or grid map should be split into quadrants or not by using a CNN. This is achieved by training a CNN to predict CU partitions of CTUs in HEVC, as representing images using a quadtree structure is fundamentally an image compression and segmentation task.

There have been many previous attempts to reduce the complexity of the brute-force RDO algorithm by instead predicting the quadtree structure of CTUs [Leng *et al.*, 2011; Lu *et al.*, 2016; Xu *et al.*, 2017; Yu *et al.*, 2015]. [Leng *et al.*, 2011] used neighbouring and co-located CU information to reduce encode time by 55%, while [Lu *et al.*, 2016] employed a bi-threshold decision scheme with [Leng *et al.*, 2011] to reduce encoding time by 57%. [Zhu *et al.*, 2017] used a binary SVM based classifier to perform CU decision, achieving 66% to 68% reduction in encoding time, while [Xu *et al.*, 2017] introduced an Early Terminated Hierarchical Convolutional Neural Network (ETH-CNN) with a bi-threshold decision scheme that reduced complexity by 64% to 71%. [Yu *et al.*, 2015] introduced a VLSI friendly CNN, reducing implementation complexity by 61%[1].

There is a benefit in significantly reducing the complexity of a quadtree generation algorithm as to make it feasible for real-time or memory-constrained processing applications. Regarding low-complexity CNNs for quadtree generation, the CNN presented in [Yu *et al.*, 2015] offers particularly low complexity due to using only an 8×8 input matrix (utilising downsampling for input CUs greater than 8×8), and a subsampling operation between the first and second convolutional layers as to reduce the number of operations required for the remainder of the network. This results in the CNN using 3352 multiply, 3054 addition, and 298 tanh operations; and 1224 trainable parameters (4896 bytes).

The contributions of this paper are twofold: Firstly, VLC-CNN builds on [Yu *et al.*, 2015] by further reducing the CNN complexity by introducing quantised inference with 8-bit integer-only arithmetic, a QP-dependent threshold decision scheme, by using a linear (versus non-linear) function for the network activations, and a myriad of subsampling techniques. Second, this paper presents a novel approach to probabilistic quadtree-generation for OGMs which is suitable for online system implementation given its minimal complexity.

To our knowledge, this is the first published research into the use of conventional image compression techniques as a means for generating variable resolution OGMs, and also the first published application of 8-bit quantisation using integer-only arithmetic CNNs for image-based quadtree segmentation.

[1]Each cited performance result is in comparison to the HEVC reference software HM[JCT-VC, 2014], with varying loss in RDO-performance

2 Very Low Complexity Convolutional Neural Network (VLC-CNN)

Network Design

The overall network is shown in Figure 1. It contains 5 layers, taking an input of 8×8 pixels, and outputs an activation representing the probabilistic likelihood of splitting the image into quadrants. The network is similar in design to [Yu *et al.*, 2015], with key differences being that VLC-CNN uses stridden convolutional layers with a kernel size of 2×2 instead of unstridden 3×3 kernels, uses a single activation and is quantised to 8-bit integer operations (in favour of the 32-bit floating point).

- *AvgPool*: The first layer, takes a square image of $N \times N$ luminance pixel data (VLC-CNN requires the use of a YUV colour space) and uses local average subsampling with a receptive field of $n \times n$ pixels, where $\{N, n\} = \{\{8, 1\}, \{16, 2\}, \{32, 4\}\}$, resulting in an 8×8 downsampled result.
- *Conv2d_1*: The image is convolved by 8, 2×2 kernels with a stride of 2, yielding an $8 \times (4 \times 4)$ result, with ReLU activation (where $ReLU(x) = \max(0, x)$).
- *MaxPool*: The convolved result is downsampled using maximum pooling with a receptive field of 2×2, yielding an $8 \times (2 \times 2)$ result.
- *Conv2d_1*: The pooled result is convolved by 16, 2×2 kernels, yielding a $16 \times (1 \times 1)$ result, with ReLU activation.
- *Dense_0*: The convolved result progresses through 8 fully-connected activations, yielding an $8 \times (1)$ result, with ReLU activation.
- *Logits_0*: Finally, the Dense result is connected to a single activation representing the probabilistic likelihood for splitting, expressed using the Sigmoid function as

$$Sigmoid(x) = \frac{1}{\exp(-x) + 1}. \qquad (2)$$

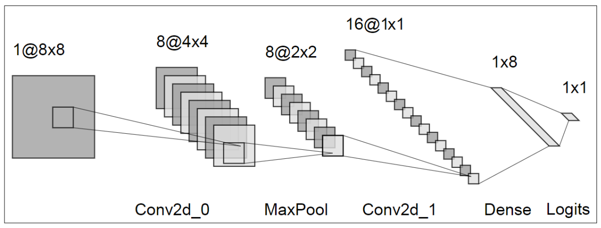

Figure 1: VLC-CNN Architecture

The CNN uses downsampling, quantisation, and a sigmoid activation function to minimise its burden on computational and memory resources:

- Downsampling: Layers *AvgPool* and *MaxPool*, as well as *Conv2d_0* (with its stride size of 2), act to downsample the data. This keeps the size of *Conv2d_0*, *Conv2d_1* and *Dense_0* from getting exceedingly large.
- Rectifier Activations: Layers *Conv2d_0*, *Conv2d_1* and *Dense_0* use Rectifier Linear Unit (ReLU) activations instead of more complex, non-linear logistic activations to reduce computational burden.
- Quantisation: The network implements quantisation to operate with 8-bit integer arithmetic (as opposed to 32-bit floating-point) on the learned network parameters. This results in 4× memory reduction while also facilitating the utilisation of simpler fixed-point arithmetic operations, with almost no impact on classification accuracy [Vanhoucke *et al.*, 2011; Jacob *et al.*, 2017]. Additionally, the quantisation enables the network to be implemented efficiently in hardware that is optimised for fixed-point arithmetic[2].
- Sigmoid: The logistic activation function for inferring the splitting probability at the output of the network is implemented efficiently as a lookup table. Its discrete, quantised nature and rotational symmetry make this method of implementation possible.

Training Procedure

To train the classifier, examples were drawn at random from the training dataset in batches of 1024. Oversampling and undersampling the minority and majority classes respectively eliminated class-imbalance. Each example included a random transformation[3] to improve training performance[4]. During training, the logistic loss of the Logits_0 activation is the probability error, calculated as:

$$Loss = \max(x, 0) - x \cdot z + \log(1 + \exp(-|x|))$$

Where x is the Logits_0 activation, z is the class-label for the training example, *log* is the natural logarithm function, and *exp* is the exponential function with base e [TFS, 2017]. The loss for a batch is the mean loss of the examples in the batch. The optimiser used is ADAM [Kingma and Ba, 2014]. Additionally, because the model is intended to be quantised for inference (as explained in Section 2), quantisation is simulated during the training loop using [TFF, 2017].

Quantisation Model

VLC-CNN is quantised using the quantisation scheme outlined in [Jacob *et al.*, 2017]. The small size of the network allows for the model to be trained offline without quantisation.

From [Jacob *et al.*, 2017], for each activation array and each weight array in the network, their floating point parameters (weights and biases) are quantised through an affine transform of their real values to the nearest 8-bit integer in the range $[0 \Rightarrow 255]$. That is

$$r = S(q - Z)\,, \tag{3}$$

where r and q represent the real and quantised values respectively, and S and Z are constants representing the quantisation scale and quantisation zero-point respectively. An $N \times N$ square quantised matrix is represented as

$$r_\alpha^{(i,j)} = S_\alpha(q_\alpha^{(i,j)} - Z_\alpha)\,, \tag{4}$$

where $1 \leqslant i, j \leqslant N$ and $q_\alpha^{(i,j)}$ denote the quantised entries. From Equation 4, the quantised result of the product of r_1 and r_2 can be expressed as

$$\begin{aligned} q_3^{(i,k)} = Z_3 + M \Bigg(& N Z_1 Z_2 - Z_1 a_2^{(k)} \\ & - Z_2 \bar{a}_1^{(i)} + \sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)} \Bigg)\,, \end{aligned} \tag{5}$$

where

$$a_2^{(k)} = \sum_{j=1}^{N} q_2^{(j,k)},\ \ \bar{a}_1^{(i)} = \sum_{j=1}^{N} q_1^{(i,j)},\ \ M = \frac{S_1 S_2}{S_3}\,,$$

and $q_3^{(i,k)}$ represents the quantised result.

For the application of quantisation to VLC-CNN, the input activations to the network are 8-bit luminance data, and the activation function is ReLU. Thus, the zero-point of the activations are always zero (*ie:* $Z_1 = 0$), and because the output of one layer is the input to the next, the input scale is one (*ie:* $S_1 = 1$), and so Equation 5 can be simplified to

$$q_3^{(i,k)} = \frac{S_2}{S_3}\Big(\sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)} - Z_2 \sum_{j=1}^{N} q_1^{(i,j)}\Big). \tag{6}$$

The quantisation process necessarily changes the dataflow through the network due to how the quantisation scheme augments the network parameters and operations. Because the convolution operates on `uint8`, the

[2]Modern examples include the Snapdragon 835 Hexagon 682 DSP [Qua, 2018], and the Xilinx DSP48E2 slice, which is capable of performing 2 8-bit integer MACC operations per cycle [Fu *et al.*, 2017].

[3]A random transformation is defined as rotating the example image by θ, where ($\theta = \{0, 90, 180, 270\}^\circ$).

[4]These transformations help reduce overfitting by increasing the size of the dataset, and prevent the network from inheriting any rotation-bias in the dataset.

Table 1: Computational complexity for each operation for VLC-CNN

Layer	Multiplys	Adds	ROM (Bytes)
Conv2d_0	512	432	67
Conv2d_1	513	544	579
Dense_0	129	135	163
Logits_0	9	15	15
Sigmoid	0	1	128
Total:	**1179**	**1127**	**952**

accumulator must necessarily contain greater precision than 8-bits - thus an `int32` accumulator is used. Finally, because the result of the convolution operation must yield an `uint8`, the `int32` activation must be rescaled back to 8-bit. The data-flow of a quantised convolution[5] operation is outlined in Figure 2.

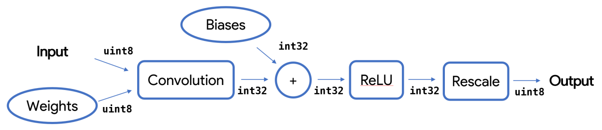

Figure 2: Data-flow and data-types of a quantised convolution operation.

Computational Complexity

Table 1 shows the computational complexity of VLC-CNN and each of its layers. VLC-CNN requires significantly fewer multiply/add operations and memory requirement than other models [Xu *et al.*, 2017; Katayama *et al.*, 2018; Yu *et al.*, 2015] - VLC-CNN's complexity is closest in comparison to [Yu *et al.*, 2015], requiring 36% as many operations and 19% as much memory. However, this does not take into account the increased efficiency of 8-bit fixed-point arithmetic compared to the other models floating-point arithmetic. Additionally, both [Yu *et al.*, 2015] and VLC-CNN utilise the same CNN architecture for different size CUs (as to reuse hardware logic in an FPGA implementation), however, instead of having different CNN parameters for CUs of different size, VLC-CNN uses the same learned parameters for each CU size to minimise the ROM size, which is compensated for using the QP-dependent threshold scheme outlined in Section 2.

Quadtree Generation

VLC-CNN generates quadtrees with a top-down approach. The quadtree for a given CTU is generated as follows:

[5]The convolution can be implemented as a dot-product, the same data-flow is used for fully-connected layers.

Algorithm 1 Decides whether to split a CU into its children CUs

```
$split \Leftarrow False$
if ($top\_prob \geq top\_thresh$) or ($\max(child\_probs) \geq child\_thresh$) then
    $split \Leftarrow True$
else
    $split \Leftarrow False$
end if

return $split$
```

1. For each depth, a splitting threshold is defined. As VLC-CNN returns an 8-bit quantised splitting probability (whereby probabilities in the range $\{0 \Rightarrow 1\}$ are linearly mapped to integers in the range $\{0 \Rightarrow 255\}$) the threshold is in the range $\{0 \Rightarrow 255\}$).

2. Perform an inference for the top-level ($2N \times 2N$) CU and for each of its four quadrants ($N \times N$). For CU sizes greater than 8×8, downsample to 8×8 using *AvgPool* layer before inference.

3. Apply Algorithm 1 to the results from step 2 (where *top_prob* and *top_thresh* are the splitting probability and the splitting threshold respectively of the $2N \times 2N$ CU, and *child_prob* and *child_thresh* are the splitting probabilities and the splitting threshold respectively of the $N \times N$ CUs). If Algorithm 1 returns *True*, perform steps 2 *to* 3 for each of the four quadrants, repeating recursively until either $depth = max\_depth$ or Algorithm 1 returns *False*.

This algorithm includes early termination (similar to [Xu *et al.*, 2017]), which reduces processing time in cases where the quadtree has a depth of 0.

Dataset Generation

Learning the parameters of VLC-CNN requires the generation of a dataset representing the desired target function. The dataset is used to train, validate and evaluate the network. The dataset examples were sourced from the images from the RAISE Raw Image Dataset [Pasquini *et al.*, 2015] and classified using the HEVC HM Reference Software [JCT-VC, 2014]. The images were encoded with Quantisation Parameter $QP = 32$, `MaxCUWidth` $= 32$, `MaxPartitionDepth` $= 2$. This yielded training examples for CUs of $depth = \{0, 1\}$, and each example has $label = \{0, 1\}$, where 0 indicates that the CU was not split, and 1 indicates that the CU was split. Figure 3 depicts some examples of encoded frames which comprise the dataset.

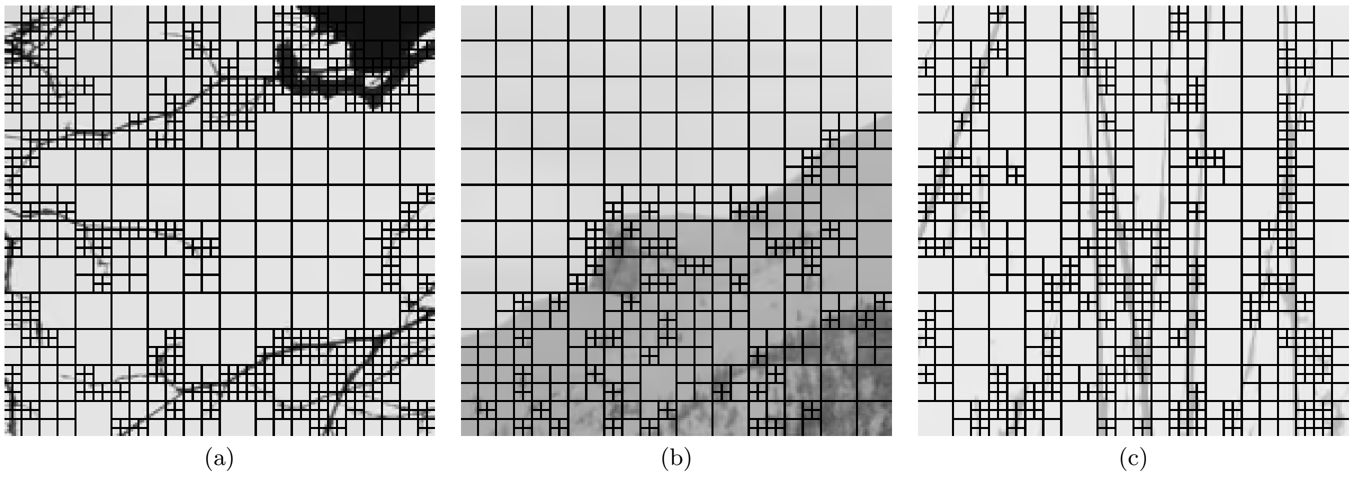

Figure 3: Examples of encoded frames used for the dataset. The training examples are 32×32 and 16×16 8-bit Luma CUs from the raw data, with labels drawn from the HM splitting decision.

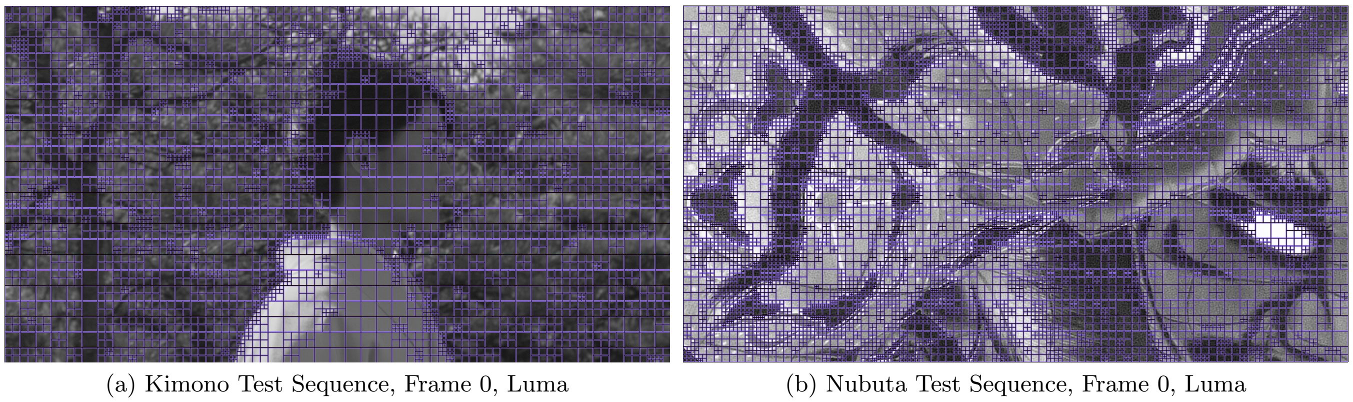

Figure 4: Visualisation of the quadtrees generated by VLC-CNN for encoding the frames at $QP = 12$.

Table 2: BD-PSNR (BP) and BD-Rate (BR) results from various encoded sequences using VLC-CNN for CTU quadtree prediction compared to HM.

Clip	BP (dB)	BR (%)
NebutaFestival	-0.69	8.61
Trafficandbuilding	-0.45	9.64
SteamLocomotiveTrain	-0.61	9.14
Kimono	-0.34	8.47
ParkScene	-0.59	10.86
Average	**-0.54**	**9.34**

3 Evaluation

VLC-CNN for Video Encoding

For evaluation using video encoding, VLC-CNN predicted CTUs for encoding using the HEVC reference software [JCT-VC, 2014], and the results were compared against the reference software itself as a control. The frames were encoded at multiple QPs ($QP = \{12, 17, 22, 27, 32, 37\}$), with peak signal-to-noise ratio (PSNR) and bit-rate recorded for each separate encode. Finally, the Bjøntegaard-delta (BD) metric [Bjontegaard, 2001] was used to calculate the BD-PSNR (BP) and BD-Rate (BR) differences between the two implementations. Figure 4 depicts CTU splits for the Nubuta and Kimono sequences at $QP = 12$. Additionally, Table 2 shows BD results for the first frame of a variety of test sequences. These results were generated with the splitting thresholds (as described in Section 2) given by

$$t_{d0} = round(3.5 \times QP + 84)\,, \tag{7}$$

$$t_{d1} = t_{d0} + 51\,, \tag{8}$$

$$t_{d2} = t_{d1} + 51\,, \tag{9}$$

where t_{dx} represents the threshold for a particular

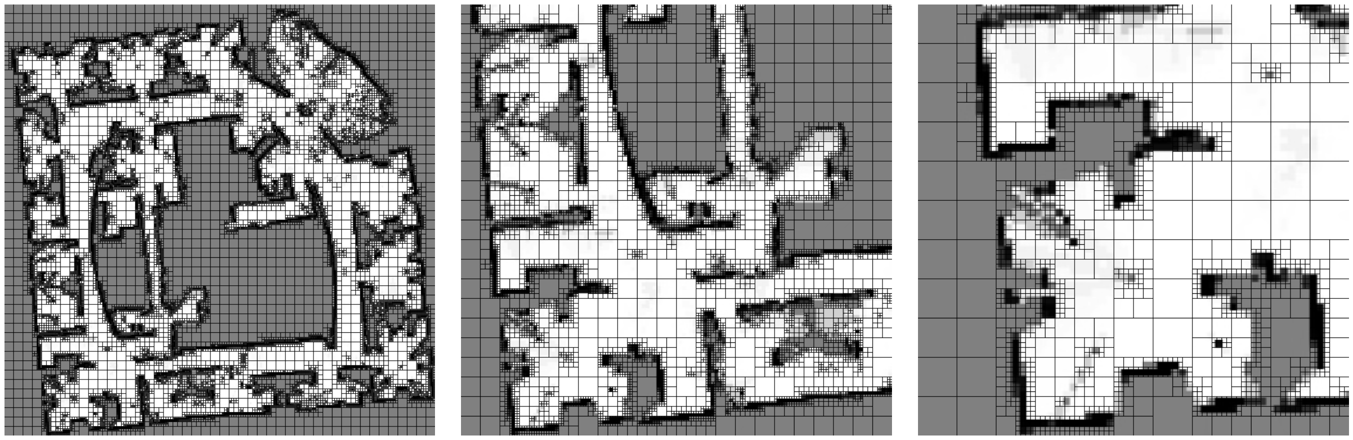
(a) Zoom: 1× (b) Zoom: 2× (c) Zoom: 4×

Figure 5: VLC-CNN applied to the Intel Dataset generated by [Jadidi *et al.*, 2016] at three diferent magnifications.

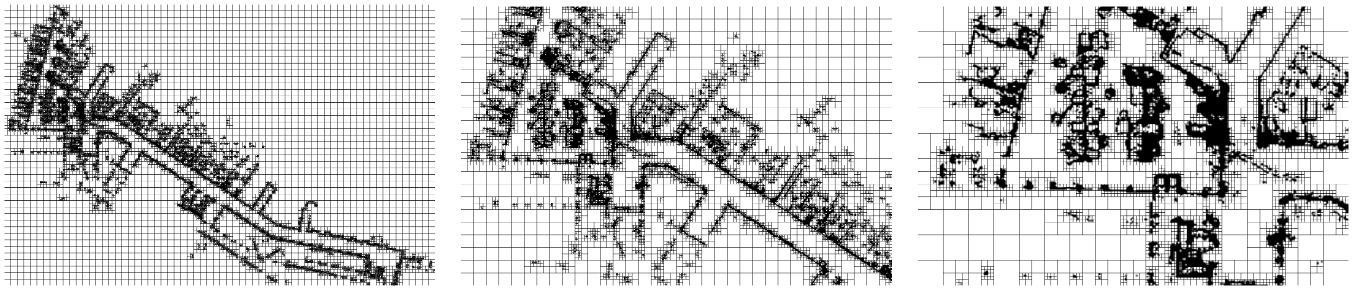
(a) Zoom: 1× (b) Zoom: 2× (c) Zoom: 4×

Figure 6: VLC-CNN applied to the an OGM (generated at UTS, Building 11, Level 9) at three diferent magnifications

depth, QP is the quantisation parameter, and *round* rounds the result to the nearest integer value. These equations are also depicted in Figure 7.

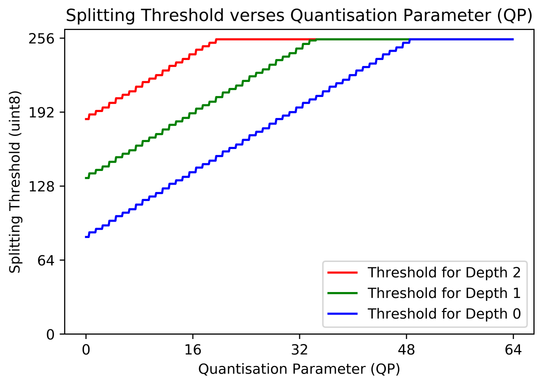

Figure 7: Splitting threshold verses Quantisation Parameter (QP) for depths $d = \{0, 1, 2\}$.

The results show that while VLC-CNN performs competitively with other implementations of low-complexity [Yu *et al.*, 2015], with BP and BR performance slightly compromised at the expense of substantial complexity reduction.

VLC-CNN for Occupancy Grid Maps

Evaluating VLC-CNN for generating quadtrees to represent OGMs was done by applying the quadtree generation algorithm (Section 2) to the OGMs. Figures 5 and 6 show an Intel Dataset [Andrew and Nicholas, 2003] OGM and an OGM generated at UTS, Building 11, Level 9 respectively. These OGM-based quadtrees were generated with splitting thresholds outlined in Equations 7, 8 and 9 with $QP = 12$, which yields more splitting for high-contrast images (which is the case for OGMs).

The use of quadtrees in the Intel and UTS OGMs - which contain 2.26×10^6 and 7.07×10^6 cells (pixels) respectively - require only 5.10×10^5 and 4.69×10^5 quadtree child nodes (where each child node has a resolution of 4×4 as to maintain the maximum accuracy for cells of depth $d = 3$) respectively. This substantially reduces the number of cells required to represent the image by using fewer cells to represent parts of the image which are of minimal contrast.

4 Conclusion

In this paper, we have presented a Very Low Complexity Convolutional Neural Network, that is capable of generating quadtree structures of images for the purpose of image segmentation and compression. This method takes a probabilistic approach to quadtree generation using a QP-based splitting-threshold scheme. VLC-CNN's design makes it suitable for applications where generating quadtree structures of images needs to be done with minimal burden on computational and memory resources, as its minimal complexity and potential for efficient implementation make this possible.

VLC-CNN was shown to be an effective method for determining quadtrees for use as CTUs in HEVC, while also substantially reducing the number of cells required to represent OGMs when compared to fixed grid-size OGMs.

References

[Andrew and Nicholas, 2003] Howard Andrew and Roy Nicholas. The Robotics Data Set Repository (Radish). 2003.

[Bjontegaard, 2001] Gisle Bjontegaard. Calculation of average PSNR differences between RD-curves. 2001.

[Einhorn *et al.*, 2011] Erik Einhorn, Christof Schroter, and Horst-Michael Gross. Finding the adequate resolution for grid mapping - Cell sizes locally adapting on-the-fly. In *2011 IEEE International Conference on Robotics and Automation*, pages 1843–1848. IEEE, may 2011.

[Fu *et al.*, 2017] Yao Fu, Ephrem Wu, Ashish Sirasao, Sedny Attia, Kamran Khan, and Ralph Wittig. Deep Learning with INT8 Optimization on Xilinx Devices White Paper (WP485). 2017.

[Jacob *et al.*, 2017] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. 2017.

[Jadidi *et al.*, 2016] Maani Ghaffari Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Gaussian Process Autonomous Mapping and Exploration for Range Sensing Mobile Robots. 2016.

[JCT-VC, 2014] JCT-VC. HM Software, 2014.

[Katayama *et al.*, 2018] Takafumi Katayama, Kazuki Kuroda, Wen Shi, Tian Song, and Takashi Shimamoto. Low-complexity intra coding algorithm based on convolutional neural network for HEVC. In *2018 International Conference on Information and Computer Technologies (ICICT)*, pages 115–118. IEEE, mar 2018.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Lei Ba. ADAM: A Method for Stochastic Optimization. 2014.

[Kraetzschmar *et al.*, 2004] Gerhard K. Kraetzschmar, Guillem Pagès Gassull, and Klaus Uhl. Probabilistic quadtrees for variable-resolution mapping of large environments. *IFAC Proceedings Volumes*, 37(8):675–680, jul 2004.

[Leng *et al.*, 2011] Jie Leng, Lei Sun, Takeshi Ikenaga, and Shinichi Sakaida. Content Based Hierarchical Fast Coding Unit Decision Algorithm for HEVC. In *2011 International Conference on Multimedia and Signal Processing*, pages 56–59. IEEE, may 2011.

[Li and Ruichek, 2013] You Li and Yassine Ruichek. Building variable resolution occupancy grid map from stereoscopic system A quadtree based approach. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 744–749. IEEE, jun 2013.

[Lu *et al.*, 2016] Xin Lu, Nan Xiao, Yue Hu, Graham Martin, Xuesong Jin, and Zhilu Wu. A hierarchical fast coding unit depth decision algorithm for HEVC intra coding. In *2016 Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, nov 2016.

[Moravec and Elfes, 1985] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121. Institute of Electrical and Electronics Engineers, 1985.

[Pasquini *et al.*, 2015] Cecilia Pasquini, Valentina Conotter, and Giulia Boato. RAISE - A Raw Images Dataset for Digital Image Forensics. *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 219–224, 2015.

[Qua, 2018] Products — Qualcomm, 2018.

[TFF, 2017] Fixed Point Quantization — TensorFlow, 2017.

[TFS, 2017] tf.nn.sigmoid_cross_entropy_with_logits — TensorFlow, 2017.

[Thrun, 2003] Sebastian Thrun. Learning Occupancy Grid Maps with Forward Sensor Models. Technical report, 2003.

[Vanhoucke *et al.*, 2011] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on CPUs, 2011.

[Wurm *et al.*, 2010] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. Technical report, 2010.

[Xu *et al.*, 2017] Mai Xu, Tianyi Li, Zulin Wang, Xin Deng, and Zhenyu Guan. Reducing Complexity of HEVC: A Deep Learning Approach. 2017.

[Yu *et al.*, 2015] Xianyu Yu, Zhenyu Liu, Junjie Liu, Yuan Gao, and Dongsheng Wang. VLSI friendly fast CU/PU mode decision for HEVC intra encoding: Leveraging convolution neural network. In *Proceedings - International Conference on Image Processing, ICIP*, volume 2015-Decem, pages 1285–1289. IEEE, sep 2015.

[Zhu *et al.*, 2017] Linwei Zhu, Yun Zhang, Zhaoqing Pan, Ran Wang, Sam Kwong, and Zongju Peng. Binary and Multi-Class Learning Based Low Complexity Optimization for HEVC Encoding. *IEEE Transactions on Broadcasting*, 63(3):547–561, sep 2017.