UNIVERSITY OF CALIFORNIA,
IRVINE

FPGA-Optimized Neural Network for Cloud Detection from Satellite Images

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Science

by

Sehwan Hong

2022

DEDICATION

To

my parents and my sister

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT OF THE THESIS

FPGA-Optimized Neural Network for Cloud Detection from Satellite Images

by

Sehwan Hong

Master of Science in Computer Science

University of California, Irvine, 2022

Professor Sang-Woo Jun, Chair

This thesis presents a highly compact neural network model optimized for FPGA implementations, targeting real-time cloud detection from RGB satellite images. Our model uses an encoder and decoder structure without skip connections, and uses piecewise linear activation functions for low-resource hardware implementations. Detecting Clouds from images using deep learning has made a lot of progress with image recognition and computer vision, at the cost of intensive computation requirements. Due to the challenge of the complexity of state-of-the art neural networks, these networks often cannot be used to perform real-time processing on edge nodes. Hardware accelerators such as FPGAs can be helpful, but naively porting neural network models without considering hardware characteristics can result in inefficient use of hardware resources and high power consumption. In this thesis, I modify a highly compact neural network for detecting clouds, C-Unet++, for efficient hardware implementation on low-power FPGAs. The modified model has a slightly different model structure, is quantized for integer operations, and also uses piecewise linear activation functions to reduce eventual FPGA resource requirements. The model is trained using the Cloud-38 dataset of RGB satellite images. The accuracy of 32-bit floating point is 93.767%. The 16-bit quantized model achieved an accuracy of 89.856%

Chapter 1. Introduction

Cloud detection is an important topic for geospatial applications, as ephemeral cloud movement needs to be separated from the actual target data such as crop state or shoreline movement. Cloud detection in real time processing has a variety of challenges because there are many types of clouds in satellite images. Cloud thickness and distribution are uneven, which brings a lot of problems to detection. Also the background contains many ground objects such as snow, ice, trees, and mountains, which makes cloud detection even harder.

Major cloud detection approaches include the threshold method, cloud texture and spatial information based method, and deep learning based method.

Threshold method is the simplest traditional method for detecting clouds. Various researchers have studied to improve the accuracy of cloud detection using threshold methods. Limited by the spectrum of early remote sensing images, the physical threshold based cloud detection algorithms inherently have low detection accuracy [1, 2, 3, 4, 5]. However, using a physical threshold for cloud detection has acceptable accuracy when a background is a single tone image such as sea or desert images.

Texture information based detection methods[6,7] use the disparity between cloud texture and surface object texture. Cloud texture and spatial information based detection depends on fixed features and has low robustness.

With the improved deep learning technology, neural network models such as decision trees and support vector machines are used for cloud detection. The Fmask (Function of mask) method [8–10] and the automated cloud cover assessment method [11] uses a decision tree threshold to classify each pixel. The Fmask method is widely used for cloud detection and evaluation. Latry et al. [12] use support vector machines to make a decision boundary for cloud detection. The detection accuracy of heavy and complex cloud areas is low. M. Hughes et al. [13] were firstly to use a neural network-based method for cloud segmentation in Landsat-8 images.

Recently, a lot of study has been adopted on cloud segmentation using convolutional neural networks(CNN) [14, 15, 16, 17, 18]. However, many of these papers use heavy neural networks with computational complexity that are not compatible with real time processing uses, especially on resource-constrained environments such as cansats. S. Ghassemi et al. [19] have proposed a small strided U-Net architecture for onboard cloud segmentation. Reconfigurable hardware accelerators such as FPGAs may be helpful in reducing the performance overhead of these models, but heavy use of floating-point operators and nonlinear activation functions have high resource requirements for low-power FPGAs such as the Lattice ECP5.

This paper proposes a highly compact deep neural network model called HC-Unet, which modifies the highly compact, but still accurate U-Net architecture [16] for real time processing. HC-Unet is based on an improved version called C-Unet++ [32], which improved on U-Net by using the same encoder-decoder structure, but eliminates two stages on both encoder and decoder, but employing only three stage encoders and three stage decoders with reduced convolutional layers. In exploiting light weight encoders, C-Unet++ has also substituted conventional convolutional layers with depth-wise convolutional layers [20]. Through these modifications, C-Unet++ is a lightweight network, which contains a much smaller number of parameters, 9017 parameters compared to 3000 times less parameters than traditional U-net architecture [16]. HC-Unet optimizes C-Unet++ for FPGA implementation by quantizing the network for integer operations, and replacing the nonlinear sigmoid activations with the simplest piecewise linear activation, the Rectified Linear Unit (ReLU). To maintain accuracy despite these changes, HC-UNet also makes small changes to the convolution layer architecture.

Experimental results present HC-Unet has good cloud detection performance relative to the number of parameters. I use Tensorflow 2.5.0 with Python 3.6.9 on Intel I7-9700K with Nvidia GPS GTX 1080 Ti. The experimental results show that the model has achieved good detection results in cloud detection tasks. I demonstrate the potential of the solutions in the case of cloud detection of remote sensing images, in particular for extracting clouds from RGB satellite images. Before integer quantization, but ReLU activations, HC-Unet reaches 93.767% accuracy in cloud detection on 38-Cloud Data set. With integer quantization, the accuracy of 32-bit floating point is 93.767%. The 16-bit quantized model achieved an accuracy of 89.856%.

The other parts of the paper are organized as follows. In Section 2, I introduce the related works of the model in this paper. In Section 3, I introduce an overview of the proposed neural network model. In Section 4, I visualize the experimental results of cloud detection. In Section 5, I describe a conclusion.

Chapter 2. Background and Related works

In this section, I introduce three different background information. Neural network architecture for image segmentation is the backbone of the presented neural networks. Computationally efficient convolution portrays the details of the presented neural network. Finally, quantizing the Neural Networks extends the thought to the future works.

2.1. Neural network architecture for image segmentation

One of the famous neural network structures for image segmentation is using encoder-decoder structures. Using encoder-decoder structures, many researchers have developed state of the art image segmentation neural networks such as U-Net [16], SegNet [21], or DeepLabv3+ [22]. Based on Unet, segNet, and Deep Labv3+, the encoder of the neural network consists of multiple layers of convolutional neural networks. In each convolutional neural network stage of the encoder composed of single or multiple convolutional layers followed by the Pooling layers that allows networks to extract features at different scales. In each convolutional stage, various convolutional neural networks structures such as ResNet[23], VGG16[24], or MobileNet[25] could be used to extract the features of the images. Decoder in image segmentation network is responsible for creating the final result, a labeled masked image that includes the boundary of the images. Decoder decypher the features extracted through encoders by up-sampling the results until the dimensions of output equals the original images. To up-sample the images from smaller images, decoder implements deconvolution introduced by H. Noh et al. [26]. Transposed convolution expands the pixels by training the parameters to predict neighborhood pixels.

2.2. Computationally efficient convolution

Increasing the number of uses in edge devices such as smartphones, IoT devices and many other smart devices have required researchers to develop artificial intelligence that is applicable in edge devices. Various low-complexity deep neural networks have emerged to have computationally simple networks but have high performances similar to state of the art neural networks, such as Inception [27], Xception [28], ShuffleNet [29], or ESPnet [30]. These networks are based on the principle of multiple simpler convolution operations. In contrast, I use depth-wise separable convolutions [25, 28]. The famous neural networks that use depth-wise separable convolutions is MobileNets [25]. Compared with the conventional convolution operation, depth-wise separable convolution splits calculations into two parts: depthwise convolution and pointwise convolution. Depthwise convolution calculates the features within a single channel. After each feature is extracted out, point-wise convolution merges features in every channel and generates a larger feature map using less computation time.

2.3. Quantizing Neural Network

Another way to simplify the computation is to use integer computation rather than floating point computation. This process is called quantization [31]. The paper [31] has introduced two different types of quantization: affine quantization and scale quantization. Affine quantization defines lower and upper bound and quantizes the values in between specific ranges defined by carefully selected lower and upper bound. Scale quantization only defines a floating point range and quantizes the values around the zero. I have used Scale quantization in this paper since scale quantization has less parameters to calibrate. The author of [31] includes different methods to quantize tensors. For simplicity, I chose to use a single parameter to quantize weights and input tensors.

Chapter 3. The Neural Network Model

I modify a highly compact cloud detection neural network model, C-Unet++, for efficient FPGA implementation. C-Unet++ is based on an encoder and decoder structure with depth-wise separable convolutions. The encoder and decoder structure has roughly 3 stage structures. The size of the networks does not depend on the size of input images. I use 192x192 images as training data and 384x384 images as testing data. The size of the images can differ in training data and the testing data.

3.1. Highly compact neural network, HC-Unet

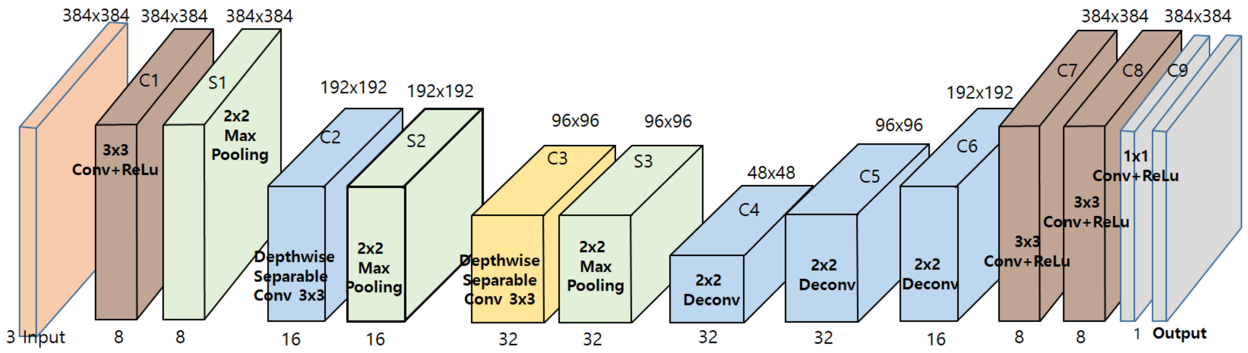

Figure 1: The details of Highly Compact Unet(HC-Unet) model

The design of HC-Unet is a revised version of C-Unet++ model [32], which has three components: a 3-layer extraction stage, a 3-layer expansion stage, and a 3-layer convolution stage(Figure 1). The extraction stage consists of one standard 3x3 convolution with the relu and two depthwise separable convolution in the encoder structure. This extraction part has three fewer convolution layers compared with the original Unet architecture that has heavy computation time. The expanding section has three 2x2 deconvolution operations. The final 3-layer convolution contains two 3x3 standard convolutions with the relu and one 1x1 convolution with the relu. When the input image is 384x384 pixels, the output of the encoder is 32 channels of 48x48 pixels. C-Unet++, and as a result, HC-Unet, reaches cloud detection accuracy as high as that of Unet despite having only 9,017 parameters, over 3,000x lower compared to U-Net.

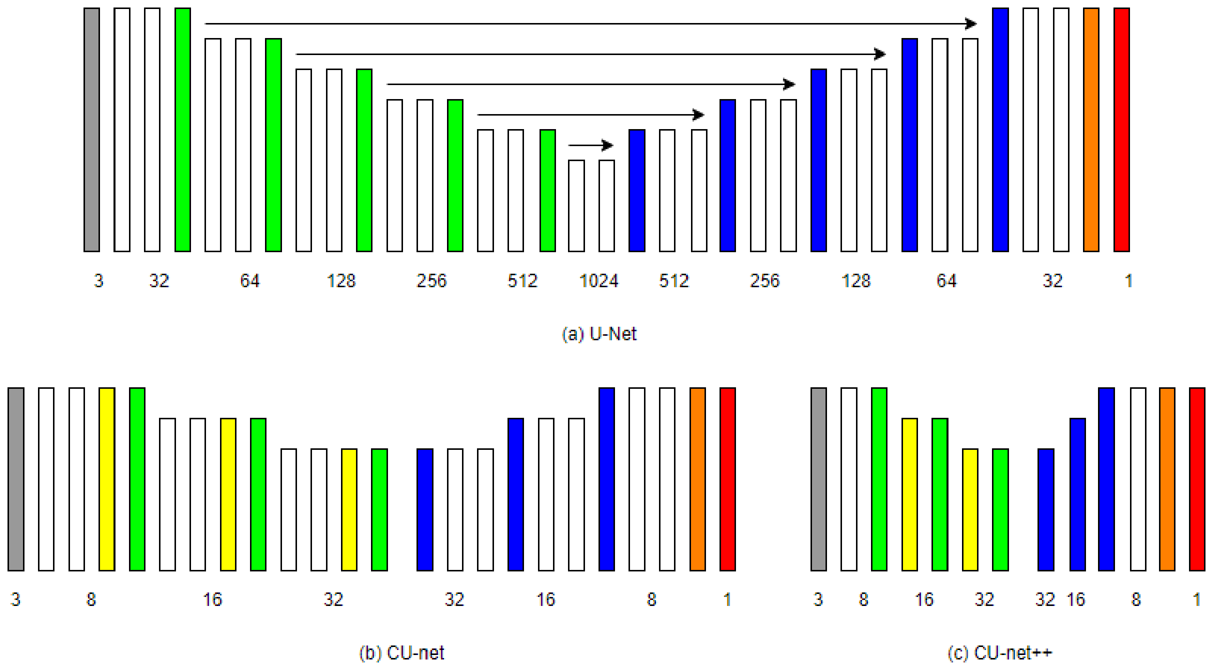

Figure 2: U-Net [16] compared to C-UNet and C-UNet++[32]. Gray: input, white: conv3x3 + ReLU, yellow: depthwise separable conv3x3 + ReLU, green: 2x2 max pool, orange: conv1x1 sigmoid, blue: 2x2 deconvolution, red: output, arrows: skip connections. Numbers indicate the number of feature maps at each stage, as introduced in [32].

HC-UNet modifies the C-UNet++ for efficient FPGA implementation without much accuracy loss. Instead of the non-linear sigmoid activation functions, HC-UNet uses the Rectified Linear function. In order to maintain accuracy despite the activation function change and eventual quantization, HC-UNet also changes the number of 3x3 convolutional layers. As shown in Figure 2, C-Unet++ upsampling step has three deconvolution layers and two convolution layers; the last two convolution layers have a 3X3 convolution layer and a 1X1 standard convolution layer with sigmoid activation. While for HC-Unet, the decoder has three transposed convolution operations and three convolution layers.

Layer(type)	Output feature map	Num. of parameters
Input	(384, 384, 3)	0
C1(Conv. layer+Relu)	(384, 384, 8)	224
S1(Max Pooling)	(192, 192, 8)	0
C2(Separable Conv.+Relu)	(192, 192, 16)	216
S2(Max Pooling)	(96, 96, 16)	0
C3(Separable Conv.+Relu)	(96, 96, 32)	688
S3(Max Pooling)	(48, 48, 32)	0
C4(Deconvolution layer)	(96, 96, 32)	4128
C5(Deconvolution layer)	(192, 192, 16)	2064
C6(Deconvolution layer)	(384, 384, 8)	520
C7(Conv. layer+Relu)	(384, 384, 8)	584
C8(Conv. layer+Relu)	(384, 384, 8)	584
C9(Conv. layer+Relu)	(384, 384, 1)	9
Output	(384, 384, 1)	0

Table 1: Proposed Neural Network model with the number of parameters

Architecture	Num. of parameters	Storage (MB)
HC-UNet	9017	0.193
C-Unet++	9129	0.172
C-UNet	51113	0.735
mobUNet	52657	0.724
StridedUNet	79785	1.0
ESPNet A	200193	2.7
U-Net	31094497	357

Table 2: The Comparison of Network parameters and storage size. HC-Unet model has 9,017 parameters taking 0.193 MB in storage. Storage size is the size of the Keras h5 file for each model in MegaBytes. C-Unet, C-Unet++[32], mobUnet[33], StridedUNet[19], ESPNet A[30], and U-Net[16]

As for the number of parameters, in the Unet family, HC-Unet has the least number of parameters. HC-Unet has 112 less parameters than C-Unet++. HC-Unet has 5.67 times less number of parameters than C-Unet [32], and 3448 times less number of parameters than U-Net [16].

Chapter 4. Experiments on cloud detection

4.1. Data Selection and Evaluation Environment

4.1.1. Dataset

I use a public cloud image dataset, the 38-Cloud dataset [18], to train and test the models. I use RGB bands. In the dataset, approximately half of the satellite images do not have cloud information. However, I did not exclude such images because those contain the features of clear ground. Therefore, training and validation sets have 8,400 patches of 384x384 pixel images. The testing set includes 9,200 patches. This dataset contains 38 Landsat 8 scene images and their manually extracted pixel-level ground truths for cloud detection. The images of these scenes are cropped into multiple 384*384 patches to be proper for neural network algorithms

4.1.2. Training procedure

The model is implemented using the Keras framework using Tensorflow 2.5.0 as a backend in Python 3.6.9. Training and validation sets are shuffled and divided to have 80% of training data and 20% of validation data. Before feeding the inputs into the model, images are randomly flipped, rotated or cropped for data augmentation. The model is trained using a batch-size of 12. The neural network uses Adam optimizer with a learning rate of 0.0001. Learning rate is decreased by the factor of 0.7 when the training loss is on a plateau with patients of 15. Early stopping is implemented when the learning rate becomes less than 1e-8. The loss function of the network is Jaccard Coefficient.

4.1.3. Quantized Model

4.1.3.1 Quantization method

Inspired by the paper [31], I use scale quantization instead of affine quantiation. Compared to affine quantization which requires two variables, the upper and lower limits, scale quantization uses a single floating point to define a representable real value range around zero. Since the range of output values of the hidden layers are unknown, defining two parameters is more difficult to calibrate than selecting a single variable that would define range. Another reason behind using a scale quantization is that the model contains the rectified linear units that would eliminate the values that would convert all negative values to 0 while positive values remain unchanged. Since some of the kernels contain negative values, selecting an equal size positive and negative range would be a better selection than one sided floating point range.

4.1.3.2 Weight and Input Quantization

Using scale quantization, defined in the related works section, I quantized both the weights and the input values. There are different methods of quantizing the tensors across the values as presented in [31]. Out of these various methods, I selected a single parameter to quantize all values across the weights and the inputs. The reason behind using a single parameter is to eliminate the process of quantization and dequantization in every layer to match quantization levels.

4.1.3.3 Calibration

Calibration is fine-tuning of parameters to achieve highest accuracy from the model. For calibrating the quantization, there are two parameters that I adjusted to obtain minimum accuracy loss in quantization. The first and the most important parameter is the level of quantization. The quantization level I experimented with is 32-bit integer, 16-bit integer and finally the 8-bit integer. Another parameter is a floating point value which is used to define a range of representable real numbers for scale quantization [31].

4.2. Experimental Results

Evaluation matrix, experimental details, and experimental results and its analysis are explained in this section

4.2.1. Evaluation Metrics

Evaluation Metrics are Accuracy, Recall, Precision, Specificity, Jaccard Index, and F1-Score as evaluation indicators. The Precision indicator is used to obtain the precision of the neural network model, and the Recall indicator is used to get the precision of the model. The Specificity index is used to measure the completeness of the error prediction, and the Accuracy index is used to measure the accuracy of the two classifications. The Jaccard Index indicator is used to judge the similarity between the predicted mask and the real mask.

Jaccard Index = $\frac{TP}{TP+FP+FN}$

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$

In the above equation representation, TP represents true positive, where ground truth is true and prediction is positive. TN is true negative, where ground truth is true but the prediction value is false. Similarly, FP is false positive and FN is false negative. False positive is when the ground truth is false and predicts true. False Negative is when the ground truth is false and prediction value is false.

To make a simple comparison, I use accuracy to describe the performance of cloud detection algorithms in the model and Jaccard index to get the loss function for neural network training.

4.2.2. Experimental Details

In this study, all experiments are implemented using the Keras framework using Tensorflow 2.5.0 as a backend with NVIDIA GTX 1080 Ti GPU. The experiment uses python 3.6.9 as the software environment. In the experiment, I use the 384×384 pixel RGB 3-channel patch images in 38-cloud[18] as input to the neural network. As of training, the network until the early stop took approximately 48 hours in the environment. The training prematurely ceased at 664 epochs as the learning rate dropped under the value of 1e-8. The quantitative result of the HC-Unet shows that the accuracy of 32-bit floating point is 93.767% and Jaccard index of 85.8%.

4.2.3. Experimental Result Analysis

4.2.3.1 Qualitative Results

Evaluating the performance of HC-Unet for cloud extraction in RGB remote sensing images is a difficult task. Accurate detection of clouds is not an easy task, especially when a limited number of spectral bands is available, objects with similar radiometric properties are categorized into similar objects. For instance, since both clouds and snow are pure white color, it is hard to differentiate cloud and snow using only RGB images.

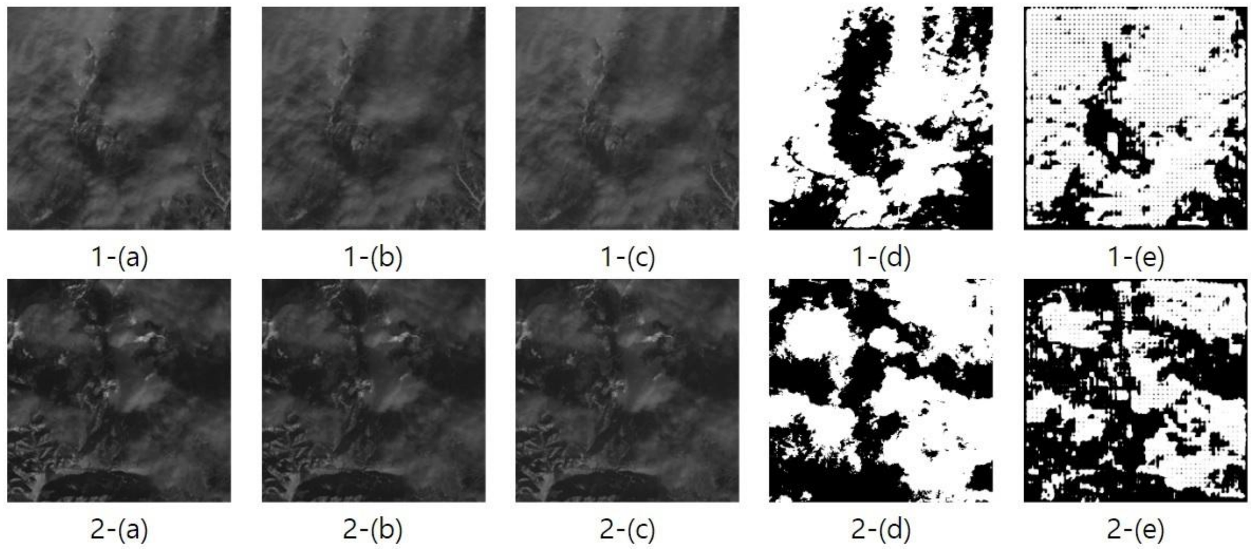

Figure 3: Qualitative experimental results of cloud detection. Numbers indicate that they are the same images. Following alphabet represents (a) red channel (b) green channel (c) blue channel (d) ground truth (e) HC-Unet on 38-Cloud dataset [18].

Figure 3 represents the good results of cloud detection in HC-Unet. Comparing the ground truth data and the predicted data of Figure 3 1-(d) and Figure 3 1-(e), HC-Unet has predicted and created accurate cloud masks. Similarly, Figure 3 2-(d) and Figure 3 2-(e) represent decent results predicting locations of large chunks of clouds correctly. However, there are some parts of the predictions that are imprecise. This inaccuracy is due to an image part with thin clouds so that the network detects them as ground.

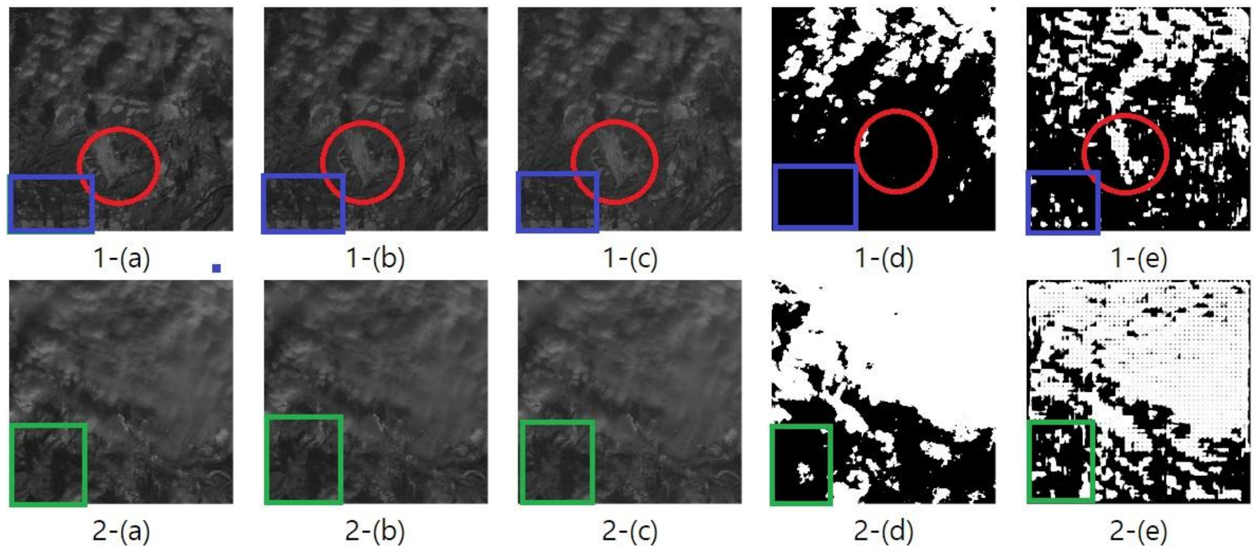

Figure 4: Predicting snow as clouds. Numbers indicate that they are the same images. Following alphabet represents (a) red channel (b) green channel (c) blue channel (d) ground truth (e) HC-Unet on 38-Cloud dataset [18].

Figure 4 visualizes the prediction of snow as clouds. In Figure 4, the red circled area and blue rectangle area both represent what HC-Unet have predicted as clouds. In the colored images, Figure 4 1-(a,b,c), both the red circle and blue rectangle area is highlighted in all spectrum of images. This represents that red circled area is a bright white colored pixel. Investigating deeper, these areas are covered with snow. Instead of predicting highlighted areas as snow, HC-Unet predicts that those are cloud images. Similarly, the green rectangle area of Figure 4 2-(e) is also covered with snow. Through these images, HC-Unet generally predicts clouds well. Since only using the RGB images, HC-Unet has a hard time differentiating snow and clouds.

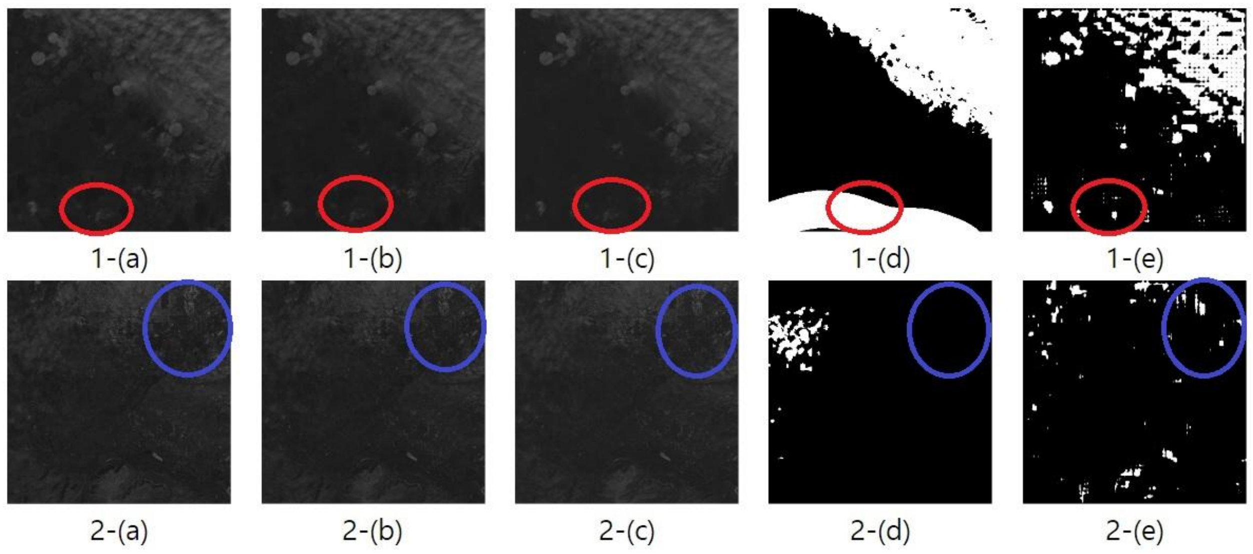

Figure 5: Miscellaneous errors in HC-Unet. Numbers indicate that they are the same images. Following alphabet represents (a) red channel (b) green channel (c) blue channel (d) ground truth (e) CUnet++ on 38-Cloud dataset [18].

Figure 5 represents various errors in using HC-Unet in 38-Cloud dataset. The first set of images is an error occurred by humans or error in the ground truth. As seen in the color images, Figure 5 1-(a, b, c), there is no cloud present in the red circled area of the image set. I found out that the red circled part of the Figure 5 1-(d) was human drawn to highlight a small cloud that was present in the red circled area. Another error is that even though there is no cloud present, HC-Unet sometimes predicts that the clouds exist. The blue circle area of Figure 5 2-(e) represents that HC-Unet has predicted that there is a cloud in this area.

4.2.3.2 Quantization Results

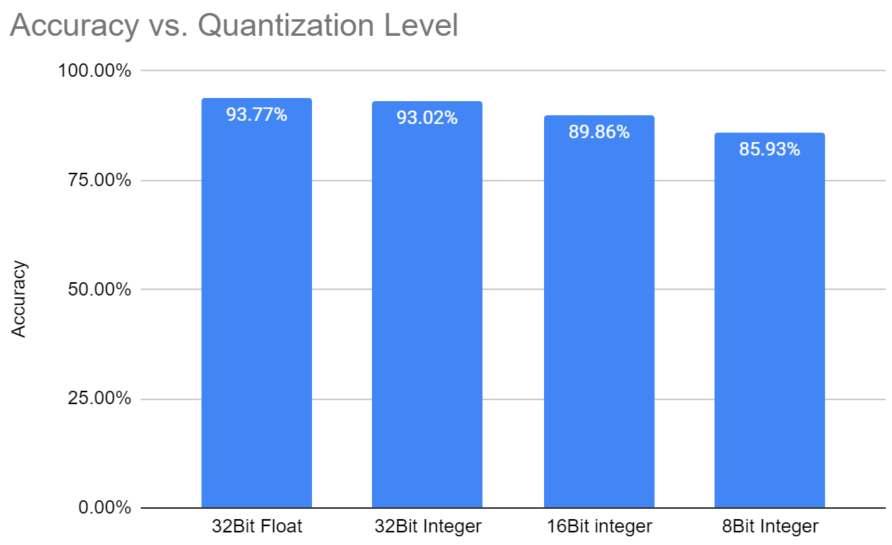

Figure 6 : Accuracy to level of quantization. As the level of quantization increases, Accuracy rate drops significantly.

The model allows fast processing of cloud detection with a Python 32-bit floating point implementation. This processing could be even faster by using quantized weights.
Using the HC-Unet model in 38-Cloud data, the accuracy of 32 bit floating point is 93.767% in python 3.6.9. I have tried various levels of quantization to find the sweet spot for quantization versus accuracy. As the level of quantization increases, the accuracy decreases significantly between 32-bit integer and 16-bit integer. Compared to the decrease in accuracy for other neural networks [34], I suppose that the drop of accuracy happens because the floating point range I used to calibrate the values are large numbers. From

16-bits to 8 bits, there is another drop of accuracy. However, this drop happens because all values are predicting a single value 0. Since training data includes images that predict the image does not contain any clouds, this tendency happens toward the test data and predicting that image has no cloud would get certain accuracy.

Chapter 5. Future Work : FPGA implementation

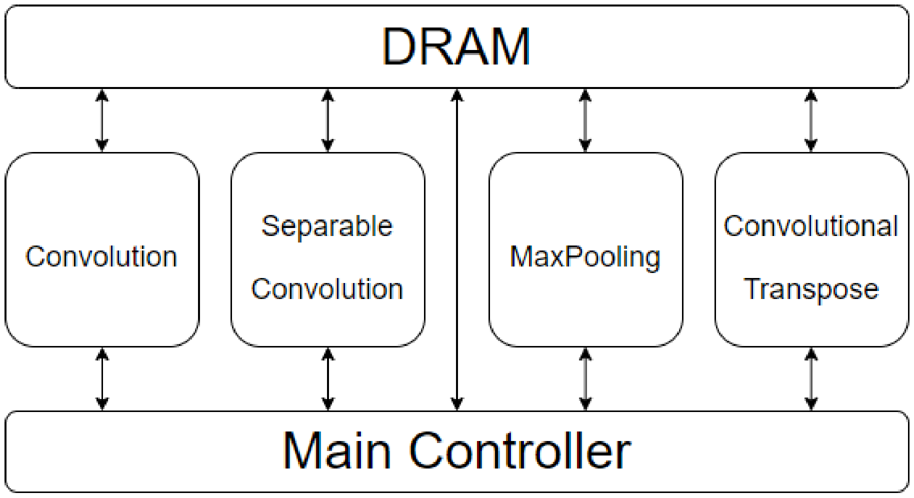

Figure 7. Structure of the FPGA for HC-Unet. They have four main modules which processes information given from DRAM and writes back to DRAM. These modules are calculated in parallel.

The planned structure of the FPGA for HC-Unet is described in Figure 7. As shown in the figure, FPGA contains the main controller, DRAM, and four main modules : Convolution, Separable Convolution, MaxPooling, and Convolutional Transpose. To process the images, DRAM is needed to store the values between each layer because there is not enough space in the FPGA on-chip memory to store the intermediate values. As shown in Table 1, the maximum values to store during the process are $384 \times 384 \times 8 = 1179648$. Using 16 bit quantization, a single value is 2 Bytes. The total memory usage for the maximum layer is 2.25MB. There is no possible way to store 2 MB of values in the FPGA. Therefore, FPGA needs a DRAM which temporarily stores intermediate values between the layers.

According to the [32], for real time performance, the FPGA must process seven frames per second. The total data read calculated using the input shape given in Table 1 is 14.2 MB. The total data write calculated similarly using the output shape is 13.6 MB. The total data read and write is 27.8MB for a single frame. Thus, calculating the required memory bandwidth to support HC-Unet is $27.8\text{MB/f} \times 7\text{fps} = 195\text{MB/s}$. There is no low-power FPGA development board in the current market that supports the required memory bandwidth. We discovered that in order to support HC-Unet, I need to design a new platform hardware with faster memory to support the necessary I/O speed.

Chapter 6. Conclusion

I have created a Highly Compact U-net, called HC-Unet for detecting clouds using RGB satellite images using an encoder and decoder structure. The model consists of three main stages: the extracting operations, the expanding operations and convolution operations. HC-UNet is optimized for eventual FPGA implementation, using piecewise linear ReLU activations instead of sigmoid, and supporting high accuracy even with quantization. I use the Cloud-38 data set publicly provided by the Landsat 8 satellite for the experiments. In the experiments, the qualitative results of HC-Unet shows excellent results of extracting clouds. The quantitative results of the HC-Unet shows that the accuracy of 32 bit floating point is 93.767% and Jaccard index of 85.8%. In the future, I will implement HC-Unet on an FPGA. Every convolutional layer could be built based on [35] and slight modification in multiplication structures could generate depth-wise convolutional layers.

REFERENCES

[1] R. A. Schiffer and W. B. Rossow, “The International Satellite Cloud Climatology Project (ISCCP): The first project of the world climate research programme,” Bulletin of the American Meteorological Society, vol. 64, no. 7, pp. 779–784, 1983.

[2] J. C. Price, “Land surface temperature measurements from the split window channels of the NOAA 7 Advanced Very High Resolution Radiometer,” Journal of Geophysical Research: Atmospheres, vol. 89, no. D5, pp. 7231–7237, 1984.

[3] W. Li and D. Li, “The universal cloud detection algorithm of MODIS data,” in Geoinformatics 2006: Remotely Sensed Data and Information, 2006, vol. 6419, pp. 108–113.

[4] X. Wu and Q. Cheng, “Study on methods of cloud identification and data recovery for MODIS data,” in Remote Sensing of Clouds and the Atmosphere XII, 2007, vol. 6745, pp. 198–205.

[5] R. Ren, S. Guo, L. Gu, L. Wang, and X. Wang, “An effective method for the detection and removal of thin clouds from MODIS image,” in Satellite Data Compression, Communication, and Processing V, 2009, vol. 7455, pp. 252–260.

[6] N. Shan, T. Zheng, Z. Wang, and others, “High-speed and high-accuracy algorithm for cloud detection and its application,” Journal of remote sensing, vol. 13, no. 6, pp. 1138–1146, 2009.

[7] P. Chen, R. Zhang, and Z. Liu, “Feature detection for cloud classification in remote sensing images,” Journal of University of Science and Technology of China, vol. 39, no. 5, pp. 484–488, 2009.

[8] Z. Zhu and C. E. Woodcock, “Object-based cloud and cloud shadow detection in Landsat imagery,” Remote sensing of environment, vol. 118, pp. 83–94, 2012.

[9] Z. Zhu, S. Wang, and C. E. Woodcock, “Improvement and expansion of the Fmask algorithm: Cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images,” Remote sensing of Environment, vol. 159, pp. 269–277, 2015.

[10] S. Qiu, Z. Zhu, and B. He, “Fmask 4.0: Improved cloud and cloud shadow detection in Landsats 4–8 and Sentinel-2 imagery,” Remote Sensing of Environment, vol. 231, p. 111205, 2019.

[11] R. R. Irish, J. L. Barker, S. N. Goward, and T. Arvidson, “Characterization of the Landsat-7 ETM+ automated cloud-cover assessment (ACCA) algorithm,” Photogrammetric engineering & remote sensing, vol. 72, no. 10, pp. 1179–1188, 2006.

[12] C. Latry, C. Panem, and P. Dejean, “Cloud detection with SVM technique,” in 2007 IEEE International Geoscience and Remote Sensing Symposium, 2007, pp. 448–451.

[13] W. Qu et al., “Application of single-band brightness variance ratio to the interference dissociation of cloud for satellite data,” Guang pu xue yu Guang pu fen xi= Guang pu, vol. 26, no. 11, pp. 2011–2015, 2006.

[14] M. Segal-Rozenhaimer, A. Li, K. Das, and V. Chirayath, “Cloud detection algorithm for multi-modal satellite imagery using convolutional neural-networks (CNN),” Remote Sensing of Environment, vol. 237, p. 111446, 2020.

[15] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2881–2890.

[16] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical image computing and computer-assisted intervention, 2015, pp. 234–241.

[17] T. Friedrich and A. Oschlies, “Neural network-based estimates of North Atlantic surface pCO2 from satellite data: A methodological study,” Journal of Geophysical Research: Oceans, vol. 114, no. C3, 2009.

[18] S. Mohajerani, T. A. Krammer, and P. Saeedi, “Cloud detection algorithm for remote sensing images using fully convolutional neural networks,” arXiv preprint arXiv:1810.05782, 2018.

[19] S. Ghassemi and E. Magli, “Convolutional neural networks for on-board cloud screening,” Remote Sensing, vol. 11, no. 12, p. 1417, 2019.

[20] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, “ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks,” CoRR, vol. abs/1910.03151, 2019, [Online] . Available: http://arxiv.org/abs/1910.03151

[21] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 12, pp. 2481–2495, 2017.

[22] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 801–818.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[25] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[26] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1520–1528.

[27] C. Szegedy et al., "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[28] F. Chollet, "Xception: Deep learning with depth-wise separable convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

[29] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848–6856.

[30] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in Proceedings of the european conference on computer vision (ECCV), 2018, pp. 552–568.

[31] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, “Integer quantization for deep learning inference: Principles and empirical evaluation,” arXiv preprint arXiv:2004.09602, 2020.

[32] G. Bahl, L. Daniel, M. Moretti, and F. Lafarge, “Low-power neural networks for semantic segmentation of satellite images,” in Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019, pp. 0–0.

[33] Z. Zhang, G. Xu, and J. Song, “CubeSat cloud detection based on JPEG2000 compression and deep learning,” Advances in Mechanical Engineering, vol. 10, no. 10, p. 1687814018808178, 2018.

[34] J. Chen, S. Hong, W. He, J. Moon, and S.-W. Jun, “Eciton: Very Low-Power LSTM Neural Network Accelerator for Predictive Maintenance at the Edge,” in 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), 2021, pp. 1–8.

[35] A. Shawahna, S. M. Sait, and A. El-Maleh, “FPGA-based accelerators of deep learning networks for learning and classification: A review,” ieee Access, vol. 7, pp. 7823–7859, 2018.