一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
VLD2018-76,CPSY2018-86,RECONF2018-50(2019-01)

意味的領域分割のための組み込みシステム向け 疎な全畳み込みニューラルネットワークのFPGA実装の検討

下田　将之†　佐田　悠生†　中原　啓貴†

† 東京工業大学 工学院 情報通信系
E-mail: †{shimoda,youki}@reconf.ict.e.titech.ac.jp, ††nakahara@ict.e.titech.ac.jp

あらまし　本稿では枝刈り手法を適応した意味的領域分割のための疎な全畳み込みニューラルネットワークのFPGA実装を提案する．意味的領域分割とはピクセル単位にクラス識別を行うタスクのことであり, 障害物や人を正確に認識する必要のある自動運転等への活用が期待されている．意味的領域分割のためのモデルの多くは, 高い正解率を達成するために深い構造をとるものが多い．そのため, 演算に必要な重みパラメータの数が大くなり, リソースの限られた組み込みシステム上では実現が困難となっている．この問題に対し, レイヤー毎に重みをソートして昇順に重みを刈るものや, 閾値をあらかじめ決め閾値以下の重みを刈る手法が提案されている．しかし, それらの手法を適応したモデルを組み込みシステムへ実現する際に, フィルター毎に存在する重みの数が異なるため最も重みの数が多いフィルターに合わせた回路を作る必要がある．そのため, それ以外のフィルターは無駄な計算を行う必要があった．本研究では, ハードウェアにより適したフィルター毎にソートして決められた割合を昇順に刈る手法を提案する．加えて, それを適応した全畳み込みニューラルネットワークのFPGA実装を評価した．ベンチマークにはCamvidデータセット, FPGAにはXilinx zcu102評価ボードを用いた．その結果, リアルタイム処理要求 (30FPS) を満たした．

キーワード　FPGA, 全畳み込みニューラルネットワーク, 意味的領域分割

Filter-wise Pruning Approach to FPGA Implementation of Fully Convolutional Network for Semantic Segmentation

Masayuki SHIMODA†, Youki SADA†, and Hiroki NAKAHARA†

† Department of Information and Communications Engineering, School of Engineering, Tokyo Institute of Technology, Japan
E-mail: †{shimoda,youki}@reconf.ict.e.titech.ac.jp, ††nakahara@ict.e.titech.ac.jp

Abstract　This paper presents a hardware-aware sparse fully convolutional network (SFCN) for semantic segmentation on an FPGA. It is hard to implement the system on embedded systems since the number of weights for the SFCN is so large. Thus, embedded systems cannot store them using limited on-chip memory. To realize a balanced hardware with high speed and accuracy, we construct an AlexNet-based SFCN which has no skip connections and deconvolution layers to reduce the computation costs and the latency. Furthermore, we propose a filter-wise pruning technique that sorts the weights of each filter by their absolute values and prunes them by a preset percent filter-by–filter from a small order. It is more suitable for the hardware implementation since the number of computation of each filter becomes equal. We trained the AlexNet-based SFCN by using Camvid image dataset and implemented on Xilinx zcu102 evaluation board. The results show that the FPGA implementation achieves a real-time processing requirement.

Key words　FPGA, Fully Convolutional Network, Sparse Neural Network, Semantic Segmentation

1. はじめに

近年, 畳み込みニューラルネットワーク (CNNs) [1] は物体識別, 物体検出, 意味的領域分割などの幅広い画像処理分野において高い性能を達成している．中でも, 意味的領域分割は図 1 に示すようにピクセル単位にクラス識別を行うタスクのことであ

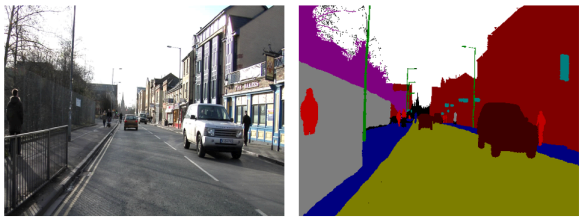

図 1　意味的領域分割の例.

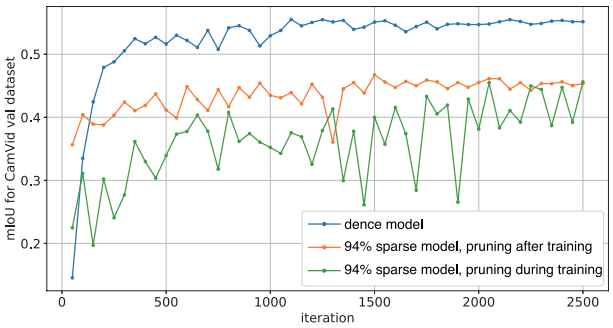

図 2　提案手法を学習中と学習後にそれぞれ適応した場合の学習曲線.

り, 障害物や人を正確に認識する必要のある自動運転等への活用が期待されている. 高い認識精度を達成している CNN の多くは, 畳み込み層を追加しているものや, 新しいアーキテクチャを導入しているものがある. その結果, モデルがとても深く複雑なものとなり, FPGA などのリソースに制限のある組み込みシステム上での実現が困難なものとなっている.

この問題に対し, [2] は意味的領域分割のための小さなモデルを提案し, それを FPGA 実装してリアルタイム処理要求を満たした. その一方, 小さなモデルで多クラスを高い認識精度で識別するのは困難なため, 対応しているのは道路のみとなっている. そのため, 依然として意味的領域分割向け CNN の組み込みシステム上への実現には課題が多い.

本研究では, フィルター毎に枝刈りを適応した AlexNet [3] をベースにした疎な CNN を提案する. このモデルは, 意味的領域分割のための CNN に多く見られるスキップ構造や逆畳み込み層を持たないモデルであり, これにより特徴マップのバッファー数や, メモリアクセス, 計算コストを大幅に減らすことができる. 加えて, 提案モデルにはフィルター毎に一定の割合の枝刈りを適応し, 各層のフィルターの非零の数を一定にしている. その結果, 各フィルターの計算量は同じとなり, 効率よく計算することができる. 本研究の貢献点は次の通りである.

（1） AlexNet [3] をベースにした疎な CNN の FPGA 実装を提案する. 同等の認識精度を達成するモデルと比べ, パラメータ数を大幅に削減した.

（2） フィルター毎に枝刈りをする手法を提案する. これにより, 対応する回路は効率的に計算できる. また, 提案手法を用いることで従来手法 [4] より多くの枝を刈れた.

（3） 提案モデルを FPGA 実装し評価した. 提案回路はリアルタイム処理要求を達成した.

2. 関 連 研 究

2.1 意味的領域分割

Fully Convolutional Network (FCN) [5] は画像から畳み込み演算を用いて粗いラベルマップを生成し, 逆畳み込み演算や線形補間を用いて粗いラベルマップを入力画像サイズへと変換することでピクセル毎のクラス識別を実現した. SegNet [6] は小さな物体を正確に識別するために畳み込み層から逆畳み込み層へのスキップ構造を導入した. Pyramid Scene Parsing Network (PSPNet) [7] はピラミッドプーリングモジュールを提案した. これは異なるサイズのプーリングを特徴マップへ適応し, それぞれから畳み込み演算を用いて特徴を抽出し連結することで様々な大きさの物体を正確に識別できるようにした. 加えて, PSPNet はアップサンプリングに OpenCV の resize 関数を用いて計算量を抑えている. ICNet [8] は様々な解像度の入力画像をそれぞれに対応するネットワークへと適応し, その結果を連結することで認識精度を保ちつつ実行速度を改善している.

これらの従来手法はモデルサイズが大きく複雑なため, 組み込みシステムに不向きとなる. そのため, 本研究ではより単純なモデルを模索し, AlexNet [3] をベースにした疎な CNN を提案する. 提案するモデルは feed-forward 型であり, スキップ構造と逆畳み込み層を持たない構造である. アップサンプリングには PSPNet と同様の resize 関数を用いる. これにより, 実装回路は特徴マップのためのバッファーを削減できるため, 提案モデルは組み込みシステム向きなモデルとなる.

2.2 枝刈り手法

CNN は推論に大量の重みパラメータを用いた計算をするため, FPGA 等の組み込みのオンチップメモリでは全てを格納することができない. その問題に対し, 大きく分けて二つの手法が提案されてきた. 一つ目が CNN の学習中に枝刈りを適応する手法で, もう一方が学習後に枝を刈る手法である.

学習中に枝を刈る手法に関しては, [9] は徐々に刈る重みの割合を上げていき, 認識精度の低下を抑えつつ多くの重みを刈る手法を提案した. [10] は様々なドロップアウト層を畳み込み層へと適応し, 高スパース性を実現した. [11] は各層のパラメータ行列のランクを削減することで疎なモデルを実現した.

一方, 学習後に枝を刈る手法で代表的な [4] は量子化, 枝刈り, ハフマン符号化を用いてモデルを圧縮した手法で, 3-4 倍高速化した. [12] は枝 (重み) を刈るのではなく, ニューロンを刈ることによってシーケンシャルアクセスを維持したままパラメータを多く削減した. また, SIMD アーキテクチャ向けに連続するピクセルを一つのブロックとみなし, ブロック毎に刈ることで圧縮行格納方式 (CSR) 等の特定の格納方式に対しても高い並列度で計算を可能にした [13].

本研究では学習後にフィルター毎に枝を刈る手法を提案する. フィルター毎に枝を刈るため, 回路で実現した場合に効率的に計算を行える. 図 2 に提案手法を学習中と学習後にそれぞれ適応した時の学習曲線を示す. 学習中のものに比べ, 学習後に適応した場合の方が学習の収束が早い. このことから, 学習後に適応する方が提案手法により適していると考えた.

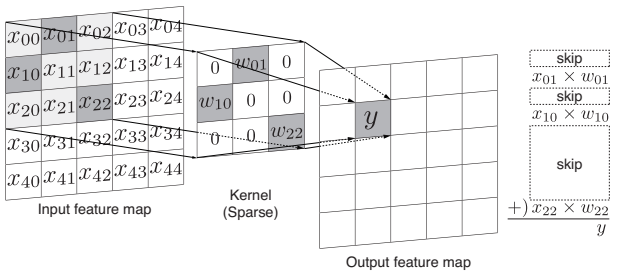

図 3　疎な畳み込み演算.

3. Fully Convolutional Network (FCN)

Fully convolutional network (FCN) とは畳み込み層のみで構成される CNN の一種である．全結合層を持たないため, 任意の画像サイズを入力できるのが特徴である．クラス分類をする場合, 全結合層の代わりに 1×1 の畳み込み層を用いる．

3.1　疎な畳み込み演算

図 3 に疎な畳み込み演算の図を示す．枝刈りを適応後, 多くの重みの値は零となる．そのスパースな行列を効率的に格納するため, 本研究では coordinate (COO) 形式を用いてメモリへと格納する．COO 形式を用いた時の畳み込み演算は次式で表される．

$$s_{x,y} = \sum_{i=0}^{N-1} X(ch_i, y + col_i, x + row_i) * data_i$$
$$z_{x,y} = f_{act}(s_{x,y}),$$

ここで, $X(ch, y, x)$ は特徴マップにおける $(ch,\ y,\ x)$ の座標に存在する値, s は中間変数, N は非零の数, f_{act} は活性化関数を表す．col_i, row_i, and $data_i$ は COO 形式で格納された列, 行, 重みにおけるインデックス i に存在する値をそれぞれ表す．

3.2　フィルター毎の枝刈り

既存の枝刈りの手法は，層ごとや CNN モデル全体の重み毎に適応される．結果として畳み込みフィルター毎のゼロ重みの比率が一定ではない．図 4 に AlexNet ベースの全畳込みモデルにおいて, 畳み込み層毎と畳み込みフィルター毎に枝刈りを適応した後の第一層目の畳み込みフィルタ毎のゼロ重みの比率を示す．このように各フィルターの非零の数の差が大きくなることがある．FPGA で疎である畳み込み層を計算行う際, 畳み込みフィルター毎のゼロ重みの数がばらついている場合, 回路は最悪のケースであるフィルター (非零の数が一番多いフィルター) に合わせて回路を作る必要がある．その結果, 他のフィルターの計算を行う際に 0 を用いた無駄な計算をするため効率が悪い．この問題に対し, 本稿では畳み込みフィルター毎の枝刈りを提案する．畳み込みフィルター毎に重みの絶対値ソートを行い, 畳み込みフィルター間で同じ数だけ値の小さな重みを枝刈りする．以下に本手法を用いた訓練方法を示す．

1)　浮動小数点数精度のモデルを訓練

2)　学習済みの重みをフィルター毎に絶対値ソート

3)　予め設定した比率に基づき，層ごとに枝刈りする重みの数を決定し, 算出した数だけ層ごとに重みの絶対値が小さい順から枝刈りする

4)　非ゼロ重みを再学習

上記の手順を踏むことで, フィルター毎の非零の数は同じになり, 対応する回路は効率的に計算を行うことができる．本稿では, 表 1 に示す通り提案モデル前段の畳み込み層を 94%, 後段のカーネルサイズ 1 の畳み込み層を 80%,75% のゼロ重みの比率を達成した．

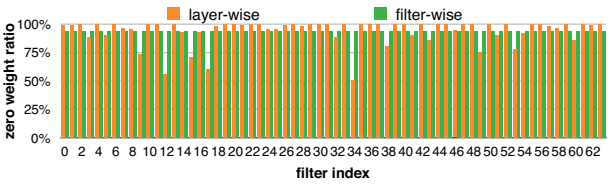

図 4　畳み込み第 1 層目のフィルター毎の非零の数.

3.3　蒸留枝刈り

蒸留 [14] とは，ソフトターゲットと呼ばれる複雑なモデルが推論したクラス確率を小さなモデルの訓練に用いることで，高い認識精度を獲得できるモデル圧縮手法である．既存研究として，蒸留をクラス認識 [15] やオブジェクト検出 [16] に適応したものがある．これに対して本稿では，意味的領域分割に対して蒸留を適応し，密な CNN モデルからゼロ重みの比率が非常に高い疎な CNN モデルを獲得する．疎な CNN モデルをハードターゲットであるアノテーションに加えソフトターゲットである密モデルが推論したクラス確率を用いて訓練することで高い認識精度を得ることが出来る．本稿では，図 5 に示すように, 学習の収束速度を向上させ高い認識精度を得るために 2 つの損失関数を用いた．

1)．ハードターゲット：ソフトマックス交差エントロピー誤差

ハードターゲット損失関数は，疎な CNN モデルの出力クラス確率とアノテーションとのピクセル単位でのソフトマックス交差エントロピー誤差

$$L_{sft} = -\frac{1}{H_{in}W_{in}} \sum_{ch=0}^{n_{class}-1} \sum_{x=0}^{W_{in}-1} \sum_{y=0}^{H_{in}-1} \log\left(\sigma\left(s_{x,y}\right)_{ch}\right) p_{ch,x,y}$$
$$\sigma\left(s_{x,y}\right)_{ch} = \frac{\exp(s_{ch,x,y})}{\sum_{k=0}^{n_{class}-1} \exp(s_{k,x,y})},$$

で定義する．ここで，H_{in}, W_{in} は入力画像サイズ，$\sigma(s_{x,y})_0, \ldots, \sigma(s_{x,y})_{n_{class}-1}$ は出力クラス確率，n_{class} はクラス数 (CamVid データセットの場合 11)，$p_{ch,x,y} \in \{1, 0\}$ はアノテーションのクラス確率である．

2)．ソフトターゲット：平均二乗誤差

ソフトターゲットは，疎なモデルの特徴マップと蜜モデルの特徴マップの平均二乗誤差で定義される．ハードターゲットも含めた全体の誤差関数は

$$L = \frac{1}{M} \sum_{j=0}^{M-1} \frac{\alpha_j}{C_j H_j W_j} \sum_{ch=0}^{C_j-1} \sum_{x=0}^{W_j-1} \sum_{y=0}^{H_j-1} \left(s_{ch,x,y}^{t_j} - s_{ch,x,y}^{'t_j}\right)^2$$

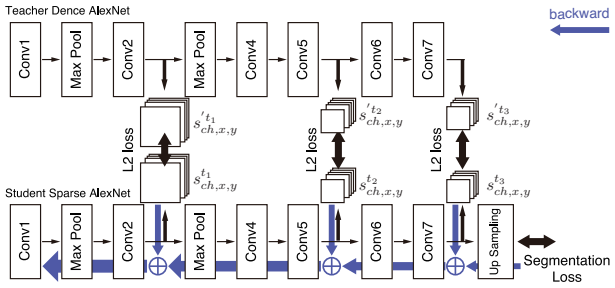

図 5　蒸留枝刈り.

$$+ \beta L_{sft},$$

で定義される．ただし，M は蒸留を適応する特徴マップの数，α_j, β は学習率を制御するためのハイパーパラメータ，s, s' はそれぞれ疎なモデルと密なモデルの特徴マップのピクセル値，L_{sft} はソフトターゲット誤差関数である．図 4 に示すように，実験では $M = 3$ とし，第一層目，第二層目，第七層目の出力特徴マップを用いた．以上の蒸留枝刈りを AlexNet ベースの全畳込みモデルに適応した．ソフトターゲットを用いない疎なモデルの訓練では 40.53% [mIoU] であり，ソフトターゲットとハードターゲット用いた訓練では 44.90% [mIoU] であった．ただし，2 つの実験とも CNN モデルのゼロ重みの比率は 93.6% に設定した．ソフトターゲットとハードターゲット用いた訓練によって，密モデルに対して mIoU をわずか 0.14% の減少に留めることが出来た．

3.4　AlexNet-based Fully Convolutional Network

高い認識精度を誇る意味的領域分割のためのモデルの多くは複雑なスキームで深い構造を取るため FPGA 実装に不向きである．そのため，本稿では FPGA 実装に適した AlexNet をベースにしたモデルを用いる．モデルの詳細を表 1 に示す．バッチ正規化 (BN) [17] を各畳み込み層の後段に挿入する．認識精度の低下を防ぐために，AlexNet の最後の最大値プーリング層を取り除き，1×1 の畳み込み層における枝を刈る割合を前段に比べ小さく設定した．加えて，メモリアクセスを多く引き起こす逆畳み込み層の代わりに OpenCV の resize 関数は PL 側には実現せず PS 側で行う．表 2 に従来手法との浮動小数点数精度での比較結果を示す．提案モデルは同等の認識精度を維持したままパラメータ数を大幅に削減することができた．本研究では，この提案モデルをベースラインとして扱う．

4.　実 装 回 路

図 6 に実装した回路の全体像を示す．回路は大きく分けて重み等のパラメータをキャッシュしておくバッファ部，畳み込み演算をするブロック (CB) 部があり，CB 部を複数並べてタスクレベルのパイプラインアーキテクチャを構成している．重みやバイアス等全てのパラメータは DDR3 メモリからオンチップメモリのブロック RAM (BRAM) へ送信される．その後，プロセッサは入力画像を PL 側 (ハードウェア) へと送信し，最初の CB で一段目の畳み込み演算が行われる．各 CB の出力特徴マップは次の段の CB 内にあるピンポンバッファへと送

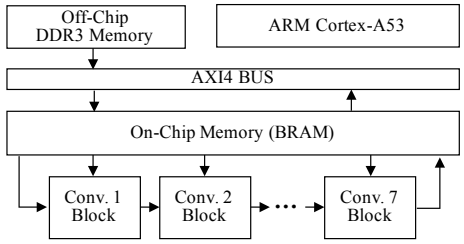

図 6　Overall architecture.

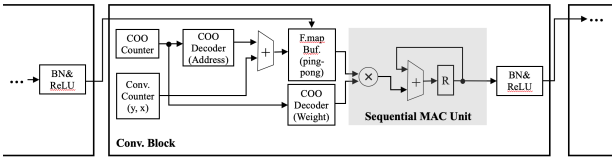

図 7　Convolutional block.

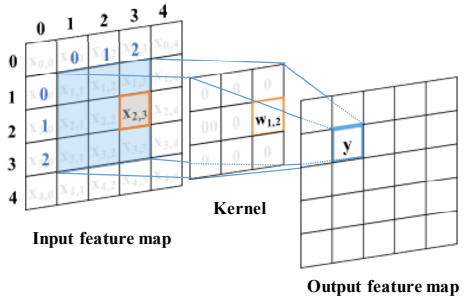

図 8　絶対座標と相対座標の例．絶対座標 (y, x) = (1,1) は畳み込み演算を行う座標を示し，青で示した相対座標 (col, row, ch) = (2,1,0) はフィルターの左上を基準とした座標を表す.

信され，畳み込み演算が行われる．最後の CB の計算が終了後，結果を PS 側 (ソフトウェア) の ARM プロセッサへと送信し，OpenCV を用いて画像を入力画像サイズへとリサイズする．全ての重みパラメータは COO 形式で BRAM へ格納されているため，電力効率に優れた回路を実現できる．

4.1　Convolutional Block

図 7 に CB 回路を示す．CB 回路は COO デコーダー，COO カウンター，畳み込み演算を行う座標 (絶対座標) をカウントするカウンター，ピンポンバッファー，逐次 MAC 回路，そして BN と活性化関数の ReLU を計算する処理単位 (PE) から構成される．COO カウンターは各フィルターの非零の重みの数を数えていき，そのカウントは COO デコーダへと入力され相対座標 (col, row, channel) を得る．その後，絶対座標と相対座標から対応する特徴マップの値を読み出す．図 8 に例を示す．説明のため channel 軸を削除している．畳み込み演算を行なっている座標 (y, x) は (1, 1) であり，相対座標 (col, row) は (2, 1) となっている．その場合，対応する特徴マップの座標は (y+col, x+row) = (3,2) と計算できる．上記の例のように計算して対応する値を取得した後，重みと共に逐次 MAC 回路へと入力され，その後 BN と ReLU を行う PE で計算が行われる．最後に，出力は次段の CB のピンポンバッファへと出力される．本研究では，CB 内の逐次 MAC 回路と BN 回路は半浮動小数点数精度で実現し，図 9 に示すように並列化した．表 3 に各 CB の並列

表 1　提案モデルの構成とスパース率

Layer	#In. F.maps	#Out. F.maps	In. F.Size	Kernel Size	Stride	Padding	Zero Weight Ratio
Hardware part:							
Conv	3	64	360×480	11×11	4	0	21,888/23,232 (94.2%)
MaxPool	64	64	89×119	3×3	2	0	-
Conv	64	64	44×59	5×5	1	2	96,320/102,400 (94.1%)
MaxPool	64	64	44×59	3×3	2	0	-
Conv	64	128	22×29	3×3	1	1	69,376/73,728 (94.1%)
Conv	128	128	22×29	3×3	1	1	138,624/147,456 (94.0%)
Conv	128	128	22×29	3×3	1	1	138,624/147,456 (94.0%)
Conv	128	128	22×29	1×1	1	0	13,184 /16,384 (80.5%)
Conv	128	11	22×29	1×1	1	0	1,067 /1,408 (75.8%)
Software part:							
Resize	11	11	22×29	-	-	-	-
Total	-	-	-	-	-	-	479,083/512,064 (93.6%)

表 2　Comparison with existing results on CamVid dataset.

	SegNet [6]	Ours
Params [M]	1.425	0.515
Pixel-wiseAcc	84.0%	77.6%
mClassAcc	54.6%	65.9%
mIoU	46.3%	45.0%

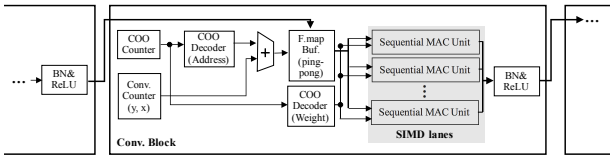

図 9　SIMD Convolutional block.

度を示す.

5. 実験結果

本実験では Chainer [18] と ChainerCV [19] を用いて提案モデルの学習と枝刈りを行なった. データセットには自動運転向けの意味的領域分割タスクのベンチマークである CamVid [20] を使用した. このデータセットは 11 カテゴリのクラスをピクセル単位で識別するものである. 認識精度を比較する実験での入力画像サイズは $(height, width) = (360, 480)$ とした.

本研究では意味的領域分割タスクを行うモデルの評価で一般的に使用される指標を用いた. n_{ij} をクラス j と識別されたクラス i のピクセル数, n_{class} は全クラス数とする. その時, 評価指標は次の 3 つとなる.

Pixel-wise accuracy (Pixel-wiseAcc):

$$\frac{\sum_{i=1}^{n_{class}} n_{ii}}{\sum_{i=1}^{n_{class}} \sum_{j=1}^{n_{class}} n_{ij}} = \frac{Accurate\ area}{All\ area}$$

Mean class accuracy (mClassAcc):

$$\frac{1}{n_{class}} \sum_{i=1}^{n_{class}} \frac{n_{ii}}{\sum_{j=1}^{n_{class}} n_{ij}} = \frac{Area\ predicted\ as\ class\ i}{Truth\ area\ of\ class\ i}$$

Mean intersection over union (mIoU):

$$\frac{1}{n_{class}} \sum_{i=1}^{n_{class}} \frac{n_{ii}}{\sum_{j=1}^{n_{class}} (n_{ij} + n_{ji}) - n_{ii}}$$

表 4 に認識精度比較の結果を示す. 提案した疎なモデルは密なモデルのパフォーマンスに近い精度を達成できた. この結果から, フィルター毎に枝を刈った後に蒸留を適応して再学習することで, 認識精度の大幅な低下を抑えつつモデルを大きく圧縮することができると考える.

次に, 様々な入力サイズに対する疎な提案モデルをそれぞれ FPGA で実現し, 回路面積を比較した. 実装の際には Xilinx の SDSoC 2017.4 を使用し動作周波数を 99.9 MHz に設定した. 使用したボードは Xilinx の Zynq UltraScale+ MPSoC zcu102 評価ボードであり, このボードには Xilinx Zynq UltraScale+ MPSoC FPGA (ZU9EG, 68,520 Slices, 269,200 FFs, 1,824 18Kb BRAMs, 2,520DSP48Es) が搭載されている. 表 5 に実装回路のリソース使用率, 表 6 に速度の見積もり結果を示す. 提案した回路は特徴マップを格納するメモリにピンポンバッファを用いているため, BRAM の使用率が支配的となった. そのため, zcu ボードでは (180 × 240) より大きなサイズに対応する FCN の回路を実現できなかった. その一方, 実装回路はリアルタイム処理要求を (30FPS) 満たした.

6. まとめ

本稿はより FPGA に適した枝刈り手法を提案し, FCN へ適応しそれを FPGA 実装した. 枝刈り後の再学習で蒸留を用いることで, 認識精度の低下を抑えつつ従来手法より枝を刈り, より組み込みシステムに適したモデルを実現できた.

謝辞　本研究は, 一部, 日本学術振興会・科学研究費補助金 (若手 (A)), 国立研究開発法人・新エネルギー・産業技術総合開発機構 (NEDO), "次世代人工知能・ロボット中核技術開発", Xilinx 社 University Program, Intel 社 University Program, NVidia 社 GPU Grant Program による.

文　献

[1] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015.

表 3　SIMD configuration for (180, 240) images.

	Conv. 1	Conv. 2	Conv. 3	Conv. 4	Conv. 5	Conv. 6	Conv. 7
#SIMD lanes	240	29	15	15	15	15	15

表 4　Semantic segmentation result

	Dence Model	Sparse Model
Zero Weight Ratio	- %	93.6%
mIoU	45.04%	44.90%
mClassAcc	65.92%	61.62%
Pixel-wiseAcc	77.64%	79.38%
Sky	85.58%	85.78%
Building	53.36%	60.80%
Pole	8.85%	8.28%
Road	86.51%	84.15%
Pavement	63.66%	59.26%
Tree	57.26%	60.66%
Sign Symbol	12.75%	16.41%
Fence	16.88%	13.56%
Car	63.20%	61.46%
Pedestrian	21.75%	19.01%
Bycyclist	25.65%	24.53%

表 5　様々な入力画像に対する FCN の FPGA 実装結果 (zcu102 評価ボードの使用率). SDSoC 2017.4 を用いて実装した.

Image Size ($H \times W$)	(90×120)	(135×180)	(180×240)
18 Kbit BRAM	537(29.4)	1,130(62.0)	1,807(99.1)
DSP48E	211(8.4)	301(11.9)	338(15.4)
FF	50,876(9.3)	72,185(13.2)	86,340(15.8)
LUT	48,204(17.6)	65,243(23.8)	79,445(29.0)
mIoU	29.23%	35.91%	40.79%

表 6　実装した回路の速度見積もり (Vivado HLS2017.4 を使用).

Image Size ($H \times W$)	(180×240)
Throughput	449,154
Clock cycle	2,466,675

[2] Y. Lyu, L. Bai, and X. Huang. Real-time road segmentation using lidar data processing on an fpga. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2018.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[4] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[5] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, April 2017.

[6] 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*, 39(12):2481–2495, Dec 2017.

[7] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, July 2017.

[8] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for real-time semantic segmentation on high-resolution images. In *ECCV*, 2018.

[9] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *CoRR*, abs/1710.01878, 2017.

[10] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*, 2017.

[11] Jose M Alvarez and Mathieu Salzmann. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*, pages 856–867, 2017.

[12] Tomoya Fujii, Simpei Sato, Hiroki Nakahara, and Masato Motomura. An fpga realization of a deep convolutional neural network using a threshold neuron pruning. In Stephan Wong, Antonio Carlos Beck, Koen Bertels, and Luigi Carro, editors, *Applied Reconfigurable Computing*, pages 268–280, Cham, 2017. Springer International Publishing.

[13] Jiecao Yu, Andrew Lukefahr, David Palframan, Ganesh Dasika, Reetuparna Das, and Scott Mahlke. Scalpel: Customizing dnn pruning to the underlying hardware parallelism. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 548–560, New York, NY, USA, 2017. ACM.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[15] Jiyang Gao, Zhen Li, Ram Nevatia, et al. Knowledge concentration: Learning 100k object classifiers in a single cnn. *arXiv preprint arXiv:1711.07607*, 2017.

[16] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 742–751. Curran Associates, Inc., 2017.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

[19] Yusuke Niitani, Toru Ogawa, Shunta Saito, and Masaki Saito. Chainercv: a library for deep learning in computer vision. In *ACM Multimedia*, 2017.

[20] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. 30:88–97, 01 2009.