C-014

ViT の FPGA 実装に向けた活性化関数 GELU の軽量化
Lightweight Activation Function GELU for FPGA Implementation of ViT

土橋 尚弥† 黒木 修隆† 沼 昌宏†
Naoya Tsuchihashi Nobutaka Kuroki Masahiro Numa

1. はじめに

近年，画像認識分野において ViT（Vision Transformer）による深層学習が注目を集めている。ViT の処理には膨大な計算量が必要であり，その処理を高速化するアクセラレータとして GPU（Graphics Processing Unit）が一般的であるが，消費電力が大きいという問題がある。そこで，汎用性に優れながら GPU より低い消費電力で動作可能な FPGA（Field-Programmable Gate Array）上に ViT をハードウェアとして実装することにより，低消費電力化を実現する研究が注目されている。しかし，FPGA の回路規模と利用可能なリソース数には制限があるため，FPGA に実装することを考慮した新たな演算手法や回路構成が必要となる。

そこで本稿では，軽量かつ精度の高い ViT である ViT-Tiny を導入し，従来のモデルと比較して精度低下を抑えつつ軽量化を実現する ViT の構造を提案する。提案する ViT では，MLP 演算に用いられる活性化関数について，従来の活性化関数と比較して軽量な 2 のべき乗を利用する関数や区分的に線形近似した関数を適用することで，演算に必要なリソース数を削減する。また，提案するモデルの学習に関して，学習を段階的に行いつつ，関数を学習途中で変更することで精度低下を抑制する。

2. 提案手法

本章では，三つの活性化関数を提案する。また，精度低下抑制のための学習手法の工夫について述べる。

2.1 GELU_COV

GELU_COVは，GELU関数をシグモイド関数によって近似し，指数部について底を 2 に変換した活性化関数である。GELU 関数は，

$$\mathrm{GELU}(x) = x \cdot \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) dt \approx x \cdot \mathrm{sigmoid}(1.702x) \quad (1)$$

と近似できる。これに対して，$x' \leftarrow (1.702 \log_2 e)x$ と変数を変換した，

$$\mathrm{GELU}_{\mathrm{COV}}(x') = \frac{x'}{1.702 \log_2 e} \cdot \frac{1}{1 + \exp_2(-x')} \quad (2)$$

を提案する。これにより，指数関数の整数部の計算にシフト演算を用いることが可能となり，FPGA における演算コストの低減が期待できる。

2.2 Swish_log2e

Swish_log2e は，Swish 関数をもとに GELU_COV の勾配消失問題を回避した活性化関数である。Swish 関数は，

$$\mathrm{Swish}(x) = x \cdot \mathrm{sigmoid}(\beta x) \quad (3)$$

と表される。そこで，底を2とするため，$\beta = \log_2 e$ とした

$$\mathrm{Swish}_{\log_2 e}(x) = x \cdot \frac{1}{1 + \exp_2(-x)} \quad (4)$$

を提案する。

2.3 PLGELU

PLGELU は，GELU 関数を 4 つの範囲で区分線形近似した活性化関数である。近似する際の条件を，

i) $x \geq 0$ で $\mathrm{PLGELU}(x) = x$

ii) GELU 関数の最小値 $(-0.75, -0.17)$，負の領域の変曲点 $(-1.41, -0.11)$ を通る。

iii) ある定数 $c < 0$ に対して，$x < c$ で$\mathrm{PLGELU}(x) = 0$

とすると，

$$\mathrm{PLGELU}(x) = \begin{cases} 0, & x < -2.62 \\ -\frac{1}{11}x - \frac{131}{550}, & -2.62 \leq x < -0.75 \\ \frac{17}{75}x, & -0.75 \leq x < 0 \\ x, & x \geq 0 \end{cases} \quad (5)$$

と求められる。軽量化のため，係数等に対して 2 のべき乗を用いて近似を行うと，

$$\mathrm{PLGELU}(x) = \begin{cases} 0, & x < -\frac{11}{4} \\ -\frac{3}{32}x - \frac{33}{128}, & -\frac{11}{4} \leq x < -\frac{3}{4} \\ \frac{1}{4}x, & -\frac{3}{4} \leq x < 0 \\ x, & x \geq 0 \end{cases} \quad (6)$$

と定義できる。これにより，関数の演算をシフト演算と加算に置き換えることが可能となり，分岐条件がビットの比較になるため，FPGA 演算におけるコストの低減が期待できる。

2.4 事前チューニング

大規模データセットにおける事前学習によって獲得された重みは，ViT の汎化性能に寄与するが，特定の活性化関数の特性に最適化されており，活性化関数の変更により表現空間が変化することが考えられる。また，活性化関数を変更すると，勾配の大きさや分布が変わり，勾配消失や勾配爆発といった問題が発生する可能性がある。その結果，ファインチューニングの収束が遅れる場合や，性能が低下する場合がある。

そこで，初期段階ではもとの活性化関数のままファインチューニングを施す，事前チューニングを行う。事前チューニングが安定してきた段階で，新たな活性化関数に移行する。これにより，活性化関数の変更に伴う性能低下の抑制が期待できる。

† 神戸大学，Kobe University

3. 実験と評価

本章では，本稿で提案した，MLP に用いられる活性化関数を軽量化した ViT モデルに関する評価を行う。

3.1 ソフトウェアによる ViT 学習

CIFAR-100 を用いた ViT 学習については，以下の 5 つの活性化関数を用いたモデルでの評価実験を行った。

i) GELU 関数（従来手法）
ii) ReLU 関数
iii) GELU_COV（提案手法 1）
iv) Swish_log2e（提案手法 2）
v) PLGELU（提案手法 3）

また，実験は

1) はじめに活性化関数を変更して 20 エポック学習する場合
2) GELU 関数のまま 10 エポックの事前チューニングを行った後，関数を変更してさらに 10 エポック学習する場合

について行った。評価項目として CIFAR-100 データセットに関する認識精度を評価した。

表 1 に，従来手法と提案手法の CIFAR-100 に対する認識精度を示す。提案手法 1，2，3 による認識精度を従来手法と比較した結果，1) 事前チューニングを行わなかった場合では，最大 7.15 pt 低下した。一方，2) 事前チューニングを行った場合では，認識精度の低下を最大 4.47 pt 縮小できることを確認した。また，従来手法である ReLU 関数を用いたときについても，事前チューニングによる認識精度低下の抑制効果を確認した。

提案手法 3 が最も精度の低下を抑えることができたことから，漸近線や極値等の関数形状が重要であると考えられる。具体的には，関数間のノルム等を用いて定量的に関数形状の類似度と精度の関係を評価することが，今後の課題として挙げられる。また，事前チューニングを行わない場合の認識精度低下の要因として，汎化性能が低下したことで，小規模なデータセットに対して過学習が発生したことが挙げられる。そのため，関数形状の大きく異なる提案手法 1，2 において大幅な精度低下が起こったと考えられる。

3.2 活性化関数層の FPGA 実装

32 bit 浮動小数点数で演算を行う活性化関数層の FPGA による実装・設計を行った。回路設計には C++，Verilog HDL を用い，HLS math ライブラリによって実現し，高位合成と論理合成を行った。動作シミュレーション・論理合成には AMD 社の Vitis 2023.2，Vivado 2023.2 を利用した。また，評価用の FPGA ボードとして ZCU102 を利用し，マッピングを行った。評価項目については，演算回路の利用リソース数，処理サイクル数とした。

表 2 に，活性化関数層をマッピングした結果を示す。提案手法 1，2，3 による利用リソース数を従来手法と比較すると，いずれも LUT を 90% 以上，FF を約 97%，DSP を 95% 以上削減できることを確認した。また，処理サイクル数は，いずれも 76% 以上削減されることを確認した。

表 1　CIFAR-100 の認識精度

活性化関数	認識精度 [%]	
	1) チューニングなし	2) チューニングあり
従来手法	83.95	83.71
ReLU	81.39	82.50
提案手法 1	76.87	81.10
提案手法 2	76.80	81.30
提案手法 3	82.75	83.49

表 2　活性化関数層のマッピング結果

活性化関数	利用リソース数				サイクル数
	LUT	FF	DSP	BRAM	
従来手法	20,489	25,519	253	0	85
ReLU	16	0	0	0	0
提案手法 1	1,705	746	11	1	20
提案手法 2	1,570	548	8	1	15
提案手法 3	377	0	0	0	0
利用可能数	274,080	548,160	2,520	912	

ReLU 関数，提案手法 3 のマッピング結果から，指数関数を用いないことで，組合せ回路により演算を実行できるようになり，LUT，FF，DSP の利用数を 0 に削減でき，サイクル数の短縮が可能となることが確認できる。また，BRAM について，従来手法では利用数が 0 であったのに対して，提案手法 1，提案手法 2 では利用数がどちらも 1 となり増加した。これは，変数変換に用いる値を保持するために利用されていると考えられる。

4. おわりに

本稿では，FPGA 実装に向けた ViT の軽量化と，軽量化による精度低下の抑制を目的として， MLP に用いられる活性化関数について，2 のべき乗を利用する関数や線形近似した関数を適用したモデルを提案した。さらに，関数の変更を学習途中とすることで，精度の低下を抑制した。

ソフトウェア上での学習・推論実験の結果，提案手法の一つである PLGELU において，従来手法と比較して認識精度が維持されることを確認した。また段階的に関数を変更するモデルは，初期設定した関数を変更しないモデルと比較して，認識精度の低下を最大 4.47 pt 抑えられることを確認し，従来の関数を適用したモデルでも，段階的な学習手法が有用であることを確認した。

さらに， C++，Verilog HDL による実装評価の結果，32 bit 浮動小数点数で演算を行う活性化関数層について提案手法は，従来手法に対して，LUT を 90% 以上，FF を約 97%，DSP を 95% 以上削減して実装できることを確認した。また，サイクル数を 76% 以上削減できることを確認した。

今後の課題として，ViT 全体の FPGA 実装や，他のネットワークへの PLGELU の適用が挙げられる。

参考文献

[1] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs),” arXiv:1606.08415, 2016.