ISSN 2186-5647

P-23

KAN Mixers におけるノード統合手法の検討

日大生産工　○佐々木　陽登　　日大生産工　山内　ゆかり

1. まえがき

コンピュータビジョンの分野では、畳み込み層を基盤としているアーキテクチャである畳み込みニューラルネットワーク（Convolutional neural network:CNN）やself-attentionを用いたTransformerベースのアーキテクチャであるVision Transformer（ViT）が主流となっている。そんな中、2021年にGoogleが提案したMLP-Mixer[1]では、多層パーセプトロン（Multi-layer perceptron:MLP）のみを用いるシンプルな構造ながらCNNやViTと同等の精度、計算コストを達成することが可能とされている。一方で、洗練された特徴抽出には不向きな可能性があり、また、多量の学習データを必要とするという課題がある。JorgeらはMLP-MixerのMLP層に、高い精度と解釈性の向上が実現できるKolmogorov-Arnold Network（KAN）[2]を導入したKAN-Mixers[3]を提案し、MLP、MLP-Mixer、KAN単体のモデルを上回る性能を示したことが報告がされている。しかし、従来のKAN同様にモデルの訓練時間が長くなる傾向があるという問題がある。

本研究では、上述の問題において訓練時間の向上を目指すため、KAN-Mixersにノード統合を行うNodes Mergingの導入を提案し、実験により提案手法と従来手法を比較し、その実験結果について報告する。

2. 従来研究

2.1.MLP-Mixer[1]

2.1.1. MLP-Mixerアーキテクチャ

MLP-Mixerでは畳み込み層やself-attentionを持たず、多層パーセプトロン（Multi-layer perceptron:MLP）のみに基づく概念的にも技術的にもシンプルなアーキテクチャを持つ。具体的には、パッチ毎の全結合層、Mixer層、Global Average Pooling（GAP）層、GAP層の出力を用いて分類を行う全結合層によって構成される。また、Mixer層は、パッチ間にMLPを適用し、空間情報をミキシングするToken-Mixing MLPとパッチ毎に独立してMLPを適用し、各位置の特徴をミキシングするChannel-Mixing MLPの2層からなる。MLP-Mixerのアーキテクチャを Figure 1に示す。

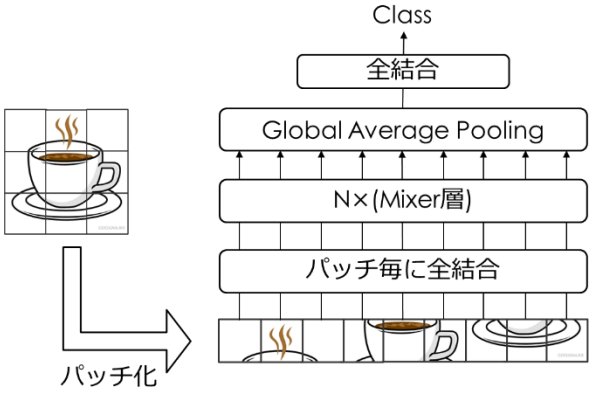

Figure 1　MLP-Mixerアーキテクチャ

2.1.2. Mixer層

MLP-Mixerの中核をなすMixer層は前述したToken-Mixing MLPとChannel-Mixing MLPに加え、それぞれにレイヤー正規化とスキップ接続を含み構成される。Mixer層のアーキテクチャをFigure 2および式(1),(2)に示す。

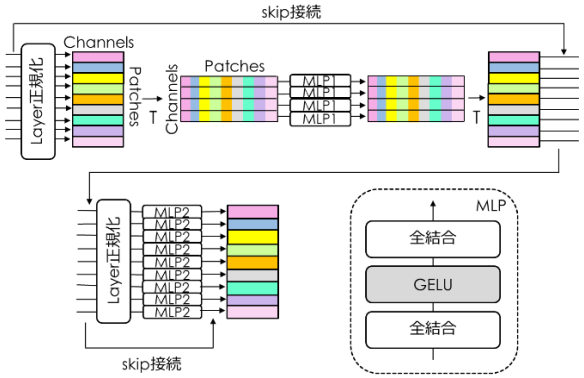

Figure 2　Mixer層のアーキテクチャ

$$U_{*,i} = X_{*,i} + W_2\sigma\left(W_1 LayerNorm(X)_{*,i}\right) \quad for\ i = 1 \ldots C, \tag{1}$$

$$Y_{j,*} = U_{j,*} + W_4\sigma\left(W_3 LayerNorm(U)_{j,*}\right) \quad for\ j = 1 \ldots S, \tag{2}$$

ここで、式(1)においてXはToken-Mixing MLPの入力、Uは出力、W_1はMLP層の1つ目の全結合層の重み、σは活性化関数、W_2は2つ目の全結合層の重み、Cはチャンネル数であり、Token-Mixing MLPではチャンネル方向の入力をレイヤー正規化したものを、入力層、活性化関数GELUを持つ中間層、出力層からなるMLPに通し、スキップ接続により入力と足し合わせたものを出力とすることで、パッチ間をミキシング

Investigation of Node Merging Methods for KAN-Mixers

Haruto SASAKI and Yukari YAMAUCHI

することを表している。また、式(2)においてUはChannel-Mixing MLPの入力、Yは出力、W_3はMLP層の1つ目の全結合の重み、σは活性化関数、W_4は2つ目の全結合の重み、Sはパッチ数であり、Channel-Mixing MLPではToken-Mixing MLPの出力であるパッチ方向の入力を用いて同様の処理を行うことで、パッチ毎にミキシングすることを表している。

2.2. Kolmogorov-Arnold Network（KAN）[2]

2.2.1. KANアーキテクチャ

現在主流となっているMLPは普遍近似定理に着想を得ており、ノード上に固定的な活性化関数を持つ。一方、KANでは入力層、中間層、出力層を用いた構造は同様だが、Kolmogorov-Arnold表現定理に着想を得て、エッジ上に学習可能な活性化関数を持つ。各重みはスプラインを用いた活性化関数によりパラメータ化しているため、ノード上には線形な重みは存在せず、入力信号の合計のみを行っている。KANのアーキテクチャをFigure 3に示す。

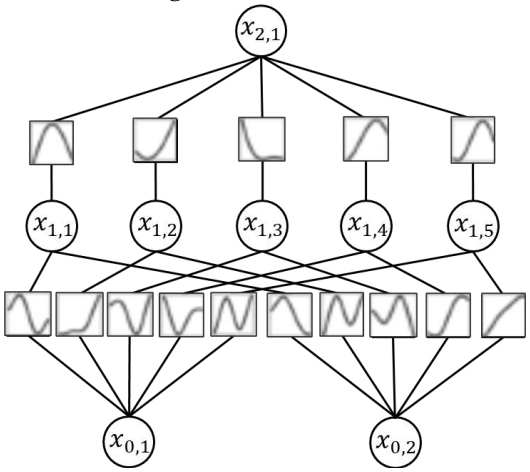

Figure 3　KANアーキテクチャ

2.2.2. Kolmogorov-Arnold表現定理

Kolmogorov-Arnold表現定理を式(3)に示す。

$$f(x)=\sum_{q=1}^{2n+1}\Phi_q\left(\sum_{p=1}^{n}\phi_{q,p}(x_p)\right) \tag{3}$$

ここで、$f(x)$はn次多変数関数であり、Φ_qと$\phi_{q,p}$は一変数関数である。つまり、Kolmogorov-Arnold表現定理とは任意の多変数関数は一変数関数の合成で表現することができるという定理である。また、xを1層目となる関数$\phi_{q,p}$に入力し合計した結果を、2層目となるノード数が2n+1個の関数Φ_qに入力し合計するという、2層のKANとなっている。

しかし、Kolmogorov-Arnold表現定理では幅2n+1と深さ2で固定であり、また、関数も未定義であるため、十分な表現力を持ったニューラルネットワークを構成することができない。そのためKANでは、Kolmogorov-Arnold表現定理を任意の幅と深さに一般化し、関数を多層に重ね合わせることで表現力を向上させている。Kolmogorov-Arnold表現定理の一般化を式(4)に示す。

$$f(x)=\sum_{i_{L-1}=1}^{n_{L-1}}\phi_{L-1,i_L,i_{L-1}}\left(\sum_{i_{L-2}=1}^{n_{L-2}}\cdots\left(\sum_{i_1=1}^{n_1}\phi_{1,i_2,i_1}\left(\sum_{i_0=1}^{n_0}\phi_{0,i_1,i_0}(x_{i_0})\right)\right)\cdots\right) \tag{4}$$

式(3)とは異なり、複数の一変数関数であるϕの合成の形となっており、幅と深さが任意となっている。

2.2.3. 活性化関数

前述したようにKANでは学習可能な活性化関数をエッジ上に持つ。KANの活性化関数$\phi(x)$を式(5)に示す。

$$\phi(x)=\omega_b b(x)+\omega_s spline(x) \tag{5}$$

ここで、ω_bとω_sはそれぞれ学習可能な重みパラメータ、$b(x)$は残差接続と似た役割を持つ基底関数、$spline(x)$はスプライン関数となっており、KANにおける活性化関数は基底関数とスプライン関数の和として表現されている。

基底関数$b(x)$を式(6)に示す。

$$b(x)=silu(x)=x/(1+e^{-x}) \tag{6}$$

基底関数にはReLU系の関数であるSiLUを使用している。

スプライン関数$spline(x)$を式(7)に示す。スプライン関数では、制御点の位置によって形が決まるなめらかで局所的に調整可能な曲線であるBスプライン曲線を用いており、これにより柔軟な非線形性を実現している。

$$spline(x)=\sum_i c_i B_i(x) \tag{7}$$

ここで、c_iは学習可能な制御点、$B_i(x)$はBスプライン基底関数を表す。つまり、各制御点をBスプライン基底関数で重みづけして合成することでBスプライン曲線を形成しており、スプライン関数はBスプライン曲線の線形結合としてパラメータ化されている。

de Boor Coxの漸化式によって定義されるBスプライン基底関数$B_i(x)$を式(8)に示す。

$$\begin{aligned}&B_{i,0}(t)=\begin{cases}1 & if\ t_i\le t<t_{i+1}\\ 0 & otherwise\end{cases}\\&i=0,\dots,m-2\\&B_{i,k}(t)\\&=\frac{t-t_i}{t_{i+k}-t_i}B_{i,k-1}(t)\\&+\frac{t_{i+k+1}-t}{t_{i+k+1}-t_{i+1}}B_{i+1,k-1}(t)\\&i=0,\dots,m-k-2\end{aligned} \tag{8}$$

ここで、t_iはノット列、kは次数を表す。t_iは単調増加であり、Bスプライン基底関数はノット列に基づき、0次から順に再帰的に求まる。0次

基底関数では、非ゼロの値を持つ区間が決まっており、それ以外では0となる。この性質により、Bスプライン基底関数は局所的にしか値を持たず、その結果、制御点の影響も局所的に限定される。

Bスプライン基底関数とBスプライン曲線の関係をFigure 4に示す。

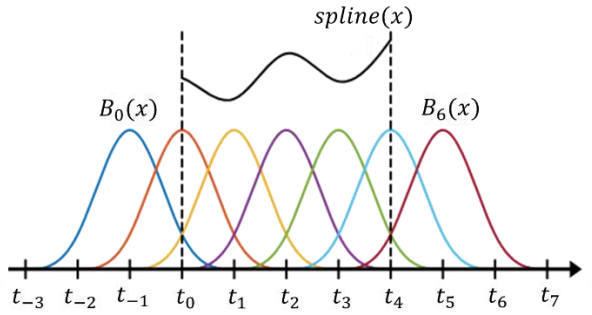

Figure 4　Bスプライン基底関数と曲線の関係

制御点とBスプライン基底関数は一対一対応しており、Bスプライン曲線上の点は、影響を受ける制御点の基底関数による重み付き合成によって位置が決まる。各Bスプライン基底関数は前述したように局所的にしか非ゼロでないため、曲線上の点はその点に影響する制御点だけによって局所的に決まる。また、基底関数は滑らかに重なっているため、曲線全体は連続的で滑らかになり、任意の点で微分可能である。この性質により、曲線の勾配を計算して勾配降下法などの最適化手法を用いることが可能となる。

2.3.KAN-Mixers[3]

KAN-MixersではMLP-Mixerにおける Mixer層の Token-Mixing MLP と Channel-Mixing MLPのMLP部分をKANへと置き換え、KAN Token-MixerとKAN Channel-Mixerとしている。KAN Token-MixerとKAN Channel-Mixer内でのKANでは活性化関数は学習可能であり、エッジごとに持つため、GELUは用いておらず、正則化のためにドロップアウトを行っている。KAN-MixersにおけるKANは、元論文では従来のKANより効率的なefficient-KANを用いているが、本研究ではefficient-KANよりさらに高速なFast KAN[4]を用いる。また、GAP層を、入力サイズに関わらず出力サイズを指定することが出来るAdaptive Average Pooling層へと変更している。

3.　提案手法

前述したように、KAN-Mixersには訓練時間が長いという課題が存在する。そこで計算量を削減するためのノード統合手法を提案する。

3.1.Nodes Merging

KAN-Mixers における KAN Token-Mixer と KAN Channel-Mixer 内の KAN の中間層の次元は、MLP-Mixerにおける Token-Mixing MLP と Channel-Mixing MLP 内の MLP の中間層同様任意となっている。そこで、Nodes Merging により中間層のノードを統合し次元を減らすことで計算量削減を目指す。Nodes Merging のイメージを Figure 5 に示す。

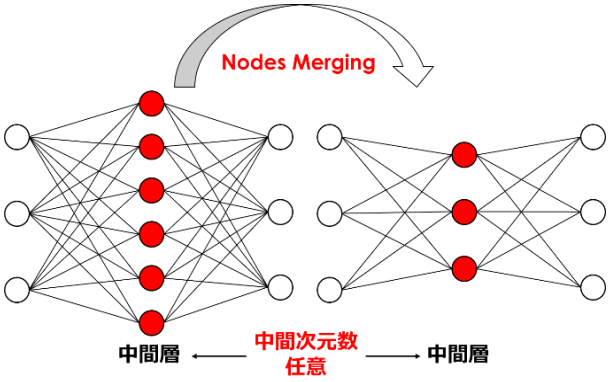

Figure 5　Nodes Merging のイメージ

Nodes Merging ではノードの重要度に基づいて中間層の閾値以下のノード同士を動的に統合し、ノード数を減らす。ノードの重要度には、KAN の重みの L1、L2 ノルム、コサイン類似度などを用いる予定である。

また、重要度に基づき閾値を設定する。Nodes Merging における閾値を式(9)に示す。

$$閾値 = \mu - k\sigma \tag{9}$$

ここでμは 1 つの MLP における重要度の平均、σは標準偏差、kは統合する強さを決めるパラメータであり、閾値はノードの重要度の平均からパラメータkをかけた重要度の標準偏差を引くことにより求める。固定値でなく平均と標準偏差を用いることで、データの散らばりも考慮した動的な閾値設定を行うことを目的としている。

現段階では Nodes Merging のみを提案しているが、今後新たな手法を着想した場合には、追加で提案を行う予定である。

4.　実験および検討

本研究では画像分類タスクで使用するデータセットを用いて、従来研究と提案手法の分類精度と実行時間の比較を行う。比較する手法はMLP-Mixer、KAN-Mixers、Nodes Mergingを導入したMLP-Mixer及びKAN-Mixersの4種類を比較する。KAN-Mixersが未実装のため、現段階ではMLP-MixerとNodes Mergingを導入したMLP-Mixerの比較を行い、提案手法の有効性を確認する。

実験で使用するデータセットは28×28サイズの0から9までの手書き数字画像となっているMNISTを使用し、60000枚の学習データと10000枚のテストデータを用いて実験を行う。

本研究の実験環境は、OSがWindows11 Pro、メモリ容量が16GB、CPUが12th Gen Intel® Core™ i7-12700F(2.10GHz)、GPUがNVIDIA GeForce RTX 3060、開発ツールはMicrosoft Visual Studio 2019である。

分類精度及び実行時間の比較実験の結果について示し、考察を行う。従来のMLP-MixerにおけるToken-mixing MLPの次元は(16, 49, 16)、Channel-Mixing MLPの次元は(49, 16, 49)としている。Nodes Mergingを導入したMLP-MixerにおけるToken-Mixing MLPの次元は(16, TM, 16)、Channel-Mixing MLPの次元は(49, CM, 49)であり、TMとCMの次元は任意でありそれぞれ初期値である49と16から動的に削減されていく。また、統合の強さを決める閾値のパラメータ k はToken-Mixing MLPでは0.55、Cannel Mixing MLPでは1.00としており、ノードの重要度には現段階ではL1ノルムを用いている。パッチの分割数は7、縦横のピクセル数は4としている。MLP-MixerとNodes Mergingを導入したMLP-Mixerの分類精度と実行時間の実験結果をTable 1に示す。

Table 1　分類精度と実行時間の比較

	Nodes Merging なし	Nodes Merging あり
TraAcc	0.970	0.961
TesAcc	0.969	0.963
Time	446	241

Table 1に示されているように、Nodes Mergingを導入したMLP-Mixerでは従来のMLP-Mixerと比較して、訓練データの分類精度は約0.9%、テストデータの分類精度は約0.6%の精度劣化であり、実行時間では約46%の速度向上となるという結果が得られた。

このことから、Nodes Mergingは精度の維持と計算時間の向上を達成できており、本研究の目的に対して有効な手法であることが確認できる。KAN-Mixersでは、エッジ上で活性化関数を計算するため、その処理に時間を要すると考えられ、本手法の導入によりこの計算が削減されることで、MLP-Mixer以上の計算量削減効果が期待できる。また、KANはMLPに比べて少ないパラメータ数で同等の表現力を有することが知られており、精度を維持しながら効率化を図ることがMLP-Mixerに比べ容易であると考えられる。

5. まとめ

本研究では、KAN-Mixersにおける訓練時間が長くなるという問題に対し、訓練時間を向上させるため、Nodes Mergingによるノード統合手法を提案し有効性を検証した。今後は従来手法であるKAN-Mixers及び提案手法であるNodes Mergingを導入したKAN-Mixersを実装し、実験によりデータを取得したうえで、有効性について検証、考察をする。

参考文献

1) Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic and Alexey Dosovitskiy, ”MLP-Mixer: An all-MLP Architecture for Vision”, arXiv:2105.01601(2021).

2) Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić and Thomas Y. Hou, Max Tegmark, “KAN: Kolmogorov-Arnold Networks”, arXiv:2404.19756(2024).

3) Jorge Luiz dos Santos Canuto, Linnyer Beatrys Ruiz Aylon and Rodrigo Clemente Thom de Souza, “KAN-Mixers: a new deep learning architecture for image classification”, arXiv:2503.08939(2025)

4) Ziyao Li, “Kolmogorov-Arnold Networks are Radial Basis Function Networks”, arXiv:2405.06721(2024)