特徴表現の維持に着目した構造化枝刈りとトークン枝刈りの併用

ER22034 篠田昂希　　指導教授：藤吉弘亘

1. はじめに

Vision Transformer（ViT）による物体認識モデルは高精度化に伴いパラメータ数が大規模化している．そのため，エッジデバイス適用やリアルタイム推論には，計算量とメモリ使用量が課題である．この課題に対して，入力トークン数を削減し，モデル全体の計算量を抑えるトークン枝刈りが注目されている．トークン枝刈りは入力を削減することで計算量を減らせるため，推論速度向上に有効である．一方で，トークンを過度に削減すると特徴表現が不足し，精度が著しく低下する．また，モデルのパラメータ数は変化しないため，モデル軽量化が不可能である．そこで本研究では，トークン枝刈りと構造化枝刈りを併用し，特徴表現と精度の維持を図りながら推論速度向上とモデル軽量化の両立を目指すハイブリッド手法を提案する．

2. トークン枝刈り

トークン枝刈りの代表的手法として DynamicViT[1] がある．DynamicViT は ViT のエンコーダ間にトークン選択モジュールを挿入し，各トークンの重要度に基づいて残すトークンを決定する．学習時は Gumbel-Softmax により選択をスコア化して学習可能にし，推論時は重要度の低いトークンを破棄して計算量を削減する．DynamicViT では，各層の保持率が目標値に近づくよう制約を与える損失 L_{ratio} を導入し，式 (1) で定義する．ここで $\mathcal{S}$ はトークン選択モジュールを挿入した層の集合，l はその層の添字である．また，r_l は目標保持率，$\hat{r}_l$ は実際の保持率である．

$$L_{\text{ratio}} = \frac{1}{|\mathcal{S}|} \sum_{l \in \mathcal{S}} |\hat{r}_l - r_l| \tag{1}$$

また，教師モデルの出力分布 $\mathbf{q}$ と生徒モデルの出力分布 $\mathbf{p}$ の差を抑える蒸留損失 L_{KL} を導入し，式 (2) で定義する．ここで k はクラスの添字であり，q_k および p_k はクラス k に対応する確率である．

$$L_{\text{KL}} = \sum_{k} q_k \log \frac{q_k}{p_k} \tag{2}$$

3. 提案手法

提案手法は図 1 に示すように，2 段階で枝刈りを行う．Step1 では cls token の特徴表現整合を導入したトークン枝刈りを行い，Step2 では勾配に基づいて MLP チャンネルを枝刈りする．

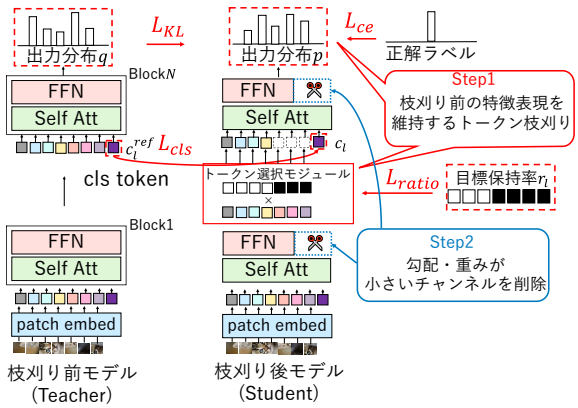

図 1：2 段階の枝刈り

3.1. cls token のコサイン類似度に基づく損失導入

Step1 では，DynamicViT に枝刈り前後の cls token のコサイン類似度に基づく損失 L_{cls} を追加する．

$$L_{\text{cls}} = \frac{1}{|\mathcal{S}|} \sum_{l \in \mathcal{S}} \left(1 - \cos(\mathbf{c}_l, \mathbf{c}_l^{\text{ref}})\right) \tag{3}$$

式 (3) に cls token の特徴表現に基づく損失項を表す．ここで $\mathbf{c}_l$ は Student の cls token，$\mathbf{c}_l^{\text{ref}}$ は Teacher の cls token である．総損失 L を式 (4) に示す．

$$L = L_{\text{ce}} + \lambda_{\text{ratio}} L_{\text{ratio}} + \lambda_{\text{KL}} L_{\text{KL}} + \lambda_{\text{cls}} L_{\text{cls}} \tag{4}$$

ここで L_{ce} はクロスエントロピー損失であり，$\lambda_{\text{ratio}}, \lambda_{\text{KL}}, \lambda_{\text{cls}}$ はハイパーパラメータである．

3.2. MLP 構造化枝刈り（勾配ベースの重要度推定）

Step2 では，式 (5) のようにトークン枝刈り学習の損失に対する重みの勾配を用いて MLP チャンネルの重要度を推定する．重要度の小さい順に枝刈りする．重要度は一次テイラー近似に基づき，重みと勾配をチャンネルごとに集約して算出する．ブロック N の MLP において，チャンネル c の重要度 $s_{n,c}$ を式 (5) により定義する．

$$s_{n,c} = \sum_{j} \left| \frac{\partial L}{\partial W_{c,j}^{(n)}} \cdot W_{c,j}^{(n)} \right| \tag{5}$$

ここで $W_{c,j}^{(n)}$ は MLP の重みであり，$s_{n,c}$ が小さいチャンネルほど出力への寄与が小さいとみなして枝刈りする．

4. 評価実験

計算量を揃えた条件で，枝刈り手法ごとの正解率，Throughput，および cls token の変化を比較する．

4.1. 実験概要

ImageNet-1k で事前学習済みの DeiT-Base/16 を用い，下流タスクとして CIFAR100 および StanfordDogs で枝刈り・評価する．比較手法は，ベースライン（枝刈り前モデル），Magnitude（MLP の Magnitude 構造化枝刈り），DynamicViT，単純併用（DynamicViT + Magnitude）である．推論時の計算量が約 8.0 GFLOPs となるように枝刈り率を調整する．トークン枝刈りはエンコーダ層 $\{3, 6, 9\}$ に適用する．MLP 構造化枝刈りは MLP チャンネルを削減する．評価指標は正解率，Throughput，および枝刈り前モデルを参照した cls token の特徴表現の変化とする．

4.2. 実験結果

表 1 に，各手法を約 8.0 GFLOPs に揃えた条件での性能を示す．ベースラインと比較すると，提案手法は計算量を抑えつつ推論速度を向上させ，精度低下も小さい．単純併用は高速化できる一方で精度低下が大きく，Magnitude 基準ではトークン枝刈り後の中間層の出力の分布変化を反映できないため，特徴表現が劣化すると考えられる．

表 1：CIFAR100 および StanfordDogs における性能比較

Method	CIFAR100		StanfordDogs	
	Acc↑	Thr.↑	Acc↑	Thr.↑
ベースライン	89.70	318.00	95.08	291.21
Magnitude	70.21	683.79	53.81	558.31
DynamicViT	80.58	**843.02**	80.67	**783.76**
単純併用	80.06	807.54	80.41	758.58
提案手法	**87.19**	808.86	**93.67**	761.27

表 2 に cls token の特徴表現の変化を示す．提案手法は単純併用と比べて特徴表現の変化が小さい傾向が確認でき，枝刈り前に近い特徴表現を維持できたといえる．

表 2：枝刈り前後の cls token の特徴表現の変化

Method	CIFAR100		StanfordDogs	
	cos sim↑	L1Norm↓	cos sim↑	L1Norm↓
Magnitude	0.699	1693.5	0.583	1773.53
DynamicViT	0.760	1554.0	0.753	1545.76
単純併用	0.403	2859.8	0.742	1631.72
提案手法	**0.980**	**697.33**	**0.869**	**777.16**

5. おわりに

本研究では，トークン枝刈りと MLP 構造化枝刈りを 2 段階で適用し，特徴表現を維持しながら高速化する手法を提案した．CIFAR100 および StanfordDogs において，提案手法は単純併用より特徴表現の変化が小さく，精度低下を抑えられることを確認した．

参考文献

[1] Y. Rao, *et al.*, “DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification”, NeurIPS, 2021.