# 二値分解法を用いた物体検出モデルの近似計算による軽量化

EP20022 大杉佳史　　指導教授：山下隆義

## 1. はじめに

Transformer モデルを用いた物体検出手法である DEtection TRansformer (DETR) は，高い性能を達成している．DETR は Transformer の Encoder と Decoder から構成されるため，そのモデルサイズは大きい．そのため，組み込み機器やモバイル端末等のリソースが限られた環境で DETR を実装するには，モデルの圧縮と推論の高速化が必要となる．本研究では，DCNN のモデル圧縮を再学習なしに実現したモデルである Binary-decomposed DCNN (BdDCNN) を参考に，DETR の畳み込み層と全結合層を二値分解することで，エッジデバイス上で動作可能にする．

## 2. DEtection TRansformer

DEtection TRansformer (DETR)[1] は，Transformer を採用した物体検出モデルである．物体検出を直接的な集合予測問題と捉えることで，End-to-End な学習を可能とするモデル設計になっている．DETR の構造を図 1 に示す．

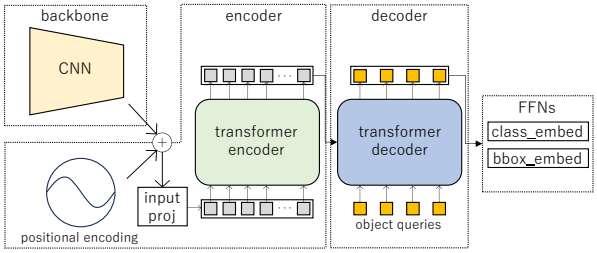


図 1：DETR の構造

## 3. 提案手法

DETR に二値分解法を適用したモデルである Binary-decomposed DETR (BdDETR) を提案する．

### 3.1. 二値分解法による近似

各層の重みと特徴マップを二値化し，各層の計算を論理演算とビットカウントによる近似内積計算にする．二値同士の内積計算は実数の重みと特徴マップの値を二値化する必要がある．そのために二値分解を用いる．二値分解は，重みベクトル $\mathbf{w} \in \mathbb{R}^D$ を二値基底行列 $\mathbf{M} \in \{-1, 1\}^{D \times k}$ とスケール係数ベクトル $\mathbf{c} \in \mathbb{R}^k$ に分解する．$k$ は基底数，$D$ は入力次元数を示している．特徴マップは入力データに依存する負値を含んだ実数値であるため，事前に二値化できない．そこで，特徴マップの量子化に Quantization sub-layer を導入する．これは，負値を含む実数値の特徴マップを高速かつ効果的に特徴マップの値を量子化し，二値の演算が可能なビット表現に変換する．

### 3.2. 畳み込み層と全結合層の計算

DETR は，backbone，encoder，decoder，bbox_embed，class_embed，input_proj で構成されており，これらは畳み込み層と全結合層からなる．畳み込み層と全結合層の計算は，入力特徴マップ $\mathbf{h}$ と重みフィルタ $\mathbf{w}$ による積和計算により計算される．そこで，この計算を式 (1) に示す二値演算に置き換える．

$$\begin{aligned} \mathbf{w}^{\mathrm{T}}\mathbf{h} &\approx \hat{\mathbf{c}}^{\mathrm{T}}\hat{\mathbf{M}}^{\mathrm{T}}(\mathbf{B}\mathbf{r} + \min(\mathbf{h})\mathbf{1}) \\ &\approx \hat{\mathbf{c}}^{\mathrm{T}}\hat{\mathbf{M}}^{\mathrm{T}}\mathbf{B}\mathbf{r} + \min(\mathbf{h})\hat{\mathbf{c}}^{\mathrm{T}}\hat{\mathbf{M}}^{\mathrm{T}}\mathbf{1} \end{aligned} \quad (1)$$

ここで，$\mathbf{B}$ と $\mathbf{r}$ は Quantization sub-layer により得られた二値行列と復元係数ベクトル，$\min(\mathbf{h})$ は量子化時に発生したオフセットである．このとき，$\hat{\mathbf{M}}$ と $\mathbf{B}$ は，二値であるため式 (2) を用いて論理演算と 64bit のポップカウント命令により計算できる．

$$\hat{\mathbf{M}}^{\mathrm{T}}\mathbf{B} = 2 \times \mathrm{POPCNT}(\mathrm{AND}(\hat{\mathbf{M}}^{\mathrm{T}}, \mathbf{B})) - ||\mathbf{h}|| \quad (2)$$

ただし，64bit のポップカウント命令は，多くのエッジデバイスで使用されている ARM の CPU には存在しない．そのため，ポップカウント命令は vld1_u8，vcnt_u8，vpaddl_u8，vpaddl_u16，vpaddl_u32，vst1_u64 により代替することで実現する．

## 4. 評価実験

本実験では，DETR と BdDETR の性能を比較する．データセットは COCO2017，バッチサイズは 2，基底数と量子化ビット数は 8 である．

### 4.1. メモリ圧縮率

各層におけるメモリ圧縮率を図 2 に示す．モデル全体で，約 158MB のパラメータを重み分解により約 44MB に削減し，全体のメモリ圧縮率は約 72.2%となった．

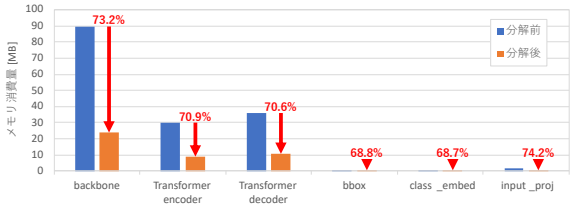


図 2：各層におけるメモリ圧縮率

### 4.2. 提案手法適用時の精度比較

提案手法の適用レイヤを変更した場合の実験結果を表 1 に示す．適用レイヤの ba は backbone，en は transformer.encoder，de は transformer.decoder の略称である．全ての層に二値分解法を適用した場合，精度が約 5pt 低下した．一方で，二値分解法をメモリ消費量が多いレイヤである backbone，encoder，decoder のみに限定すると精度低下を約 4pt に抑制できた．これより，適用レイヤをメモリ消費量の多いレイヤのみに絞ることで，精度低下を抑制しつつモデルサイズを約 70.9%削減できた．

表 1：提案手法の適用レイヤを変更した場合の実験結果

| モデル | 適用レイヤ | AP | $AP_S$ | $AP_M$ | $AP_L$ | モデルサイズ [MB] |
|---|---|---|---|---|---|---|
| DETR | - | 42.0 | 20.5 | 45.8 | 61.0 | 158.0 |
| BdDETR | 全て | 37.2 | 17.1 | 40.5 | 56.4 | 44.0 |
| BdDETR | ba, en, de | 38.4 | 18.6 | 41.9 | 57.1 | 45.9 |

### 4.3. エッジデバイスにおける評価

実行環境による DETR と BdDETR の比較を表 2 に示す．表 2 より，実行環境が Raspberry Pi においても PC と同等の精度を確認し，BdDETR はエッジデバイス上でも実行可能である．また，推論時間に関しては，PC では DETR より BdDETR が高速であるが，Raspberry Pi では逆に DETR が高速である．この原因は，PC のポップカウント命令が 4 サイクルで動作するのに対し，Raspberry Pi は 18 サイクルで動作するためと考える．

表 2：実行環境による DETR と BdDETR の比較

| モデル | 実行環境 | AP | $AP_S$ | $AP_M$ | $AP_L$ | 推論時間 [s] |
|---|---|---|---|---|---|---|
| DETR | PC | 44.9 | 23.7 | 51.9 | 66.0 | 444.6 |
| BdDETR | PC | 40.2 | 19.7 | 43.8 | 63.2 | 191.5 |
| DETR | Raspberry Pi | 44.9 | 23.7 | 51.9 | 66.0 | 3531.1 |
| BdDETR | Raspberry Pi | 40.2 | 20.5 | 43.5 | 62.7 | 4716.2 |

## 5. おわりに

本研究では，BdDETR の性能評価を行った．BdDETR は，エッジデバイス上でも PC と同様の精度を得られるが，DETR より高速に動作しないことが分かった．今後は，エッジデバイスにおいても高速で動作するように BdDETR の構造を変更することを検討する．

## 参考文献

[1] N. Carion, *et al.*, ”End-to-End Object Detection with Transformers”, ECCV, 2020.

[2] R. Kamiya, *et al.*, “Binary-decomposed DCNN foraccelerating computation and compressing model without retraining”, ICCV Workshop, 2017.