Fully Convolutional Geometric Features 논문 리뷰

Problem to solve (generally for Semantic Segmantation)

1. 기존의 Learning based models : low-level geometric 특성을 인풋으로 (e.g angular deviation, point distributions, volumetric distance functions), 3D pathch를 extract하여 MLP, 3D convolution으로 low-dimensional feature space에 mapping

   - Cons : Computationally expensive, features are extracted only at downsampled points -> should lower spatial resolution for subsequent step, Patch based process inefficient

2. Fully-convolutional network는 non-fully에 비해 광범위한 정보를 빠르고, 효율적으로 얻을 수 있지만(activation의 재사용 덕분에), 3D data의 특성 때문에 3D에서는 잘 쓰이지 못하였음. (인풋이 4D tensor이므로 매우 큰 용량)

   ➔ Sparse tensor representation을 적용해서 FCN을 3D semantic segmentation에 적용

Prior Knowledge

1. Fully Connected Network (FCN) 과 metric learning

   https://medium.com/@msmapark2/fcn-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-fully-convolutional-networks-for-semantic-segmentation-81f016d76204

   - 왜 FCN을 통과한 feature는 batch size에서 independent identically distributed(i.i.d) 가 왜 깨지는가? 4.1 -> Feature embedding으로

   - Contrastive Loss and Triplet loss (4)

$$L(\mathbf{f}_i, \mathbf{f}_j) = I_{ij} \left[D(\mathbf{f}_i, \mathbf{f}_j) - m_p\right]_+^2 + \bar{I}_{ij} \left[m_n - D(\mathbf{f}_i, \mathbf{f}_j)\right]_+^2$$

where $I_{ij} = 1$ if $(i, j) \in \mathcal{P}$ and 0 otherwise, and $\bar{\cdot}$ is the NOT operator. m_p and m_n are margins for positive and negative pairs. Similarly, we can convert the ranking constraint $m + D(\mathbf{f}, \mathbf{f}_+) < D(\mathbf{f}, \mathbf{f}_-)$ into a triplet loss:

$$L(\mathbf{f}, \mathbf{f}_+, \mathbf{f}_-) = \left[m + D(\mathbf{f}, \mathbf{f}_+) - D(\mathbf{f}, \mathbf{f}_-)\right]_+^2 \qquad (4)$$

2. Hand-Crafted 3D features

   - Review paper :A comprehensive performance evaluation of 3d local feature descriptors

3. Sparse Tensors and Convolutions

$$C = \begin{bmatrix} x_1 & y_1 & z_1 & b_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & z_N & b_N \end{bmatrix}, \; F = \begin{bmatrix} \mathbf{f}_1^T \\ \vdots \\ \mathbf{f}_N^T \end{bmatrix} \qquad (1)$$

where $x_i, y_i, z_i \in \mathbb{Z}$ is the i-th 3D coordinate and b_i is the batch index which provides an additional dimension for batch processing. $\mathbf{f}_i$ is the feature associated with the i-th coordinate.

- Comparing between Dense Convolution (2) and Sparse Convolution (3)

For example, in 1D, $\mathcal{V}^1(3) = \{-1, 0, 1\}$. The dense convolution can be defined as in Eq. 2, where $W_\mathbf{i}$ denotes the kernel value at offset $\mathbf{i}$:

$$\mathbf{x}_\mathbf{u}^{\text{out}} = \sum_{\mathbf{i} \in \mathcal{V}^3(K)} W_\mathbf{i} \mathbf{x}_{\mathbf{u}+\mathbf{i}}^{\text{in}} \text{ for } \mathbf{u} \in \mathbb{Z}^3. \qquad (2)$$

In contrast, a sparse tensor has a feature at location $\mathbf{u}$ only if the corresponding coordinates are present in the set C. Thus it suffices to evaluate the convolution terms only over a subset $\mathcal{N}^n(\mathbf{u}, K, C) = \{\mathbf{i} | \mathbf{i} \in \mathcal{V}^n(K), \mathbf{i} + \mathbf{u} \in C\}$. (I.e., the set of offsets $\mathbf{i}$ for which $\mathbf{i} + \mathbf{u}$ is present in the set of coordinates C.) If we make the sets of input and output coordinates different (C^{in} and C^{out}, respectively), we arrive at the generalized sparse convolution [2], summarized in Eq. 3:

$$\mathbf{x}'^{\text{out}}_\mathbf{u} = \sum_{\mathbf{i} \in \mathcal{N}^3(\mathbf{u}, K, C^{\text{in}})} W_\mathbf{i} \mathbf{x}'^{\text{in}}_{\mathbf{u}+\mathbf{i}} \text{ for } \mathbf{u} \in C^{\text{out}}. \qquad (3)$$

➔ Sparse convolution 은 offset에 대해 kernel 방향으로의 성분이 있을 때만 weight를 계산한다. E.g) kernel이 3x3x3이라고 하면, offset point(0,0,0)에 대해 (-1~1,-1~1, -1~1)인 point에 대해서만 convolution 계산.

Key Architecture

1. 3D fully-convolutional network

   - Fully-Convolutional processing : MLP 의 fully connected layer 를 series of kernel size 1x1x1 convolutional layer 로 변환

2. New metric learning losses : Fully convolutional feature 가 기존의 iid 와 다르기 때문

Positive pair 에 대해 각각 hard negative 를 구한 후, false negative 를 제거하여 quadruplet 을 mining 한다.

convolutional contrastive loss:

$$L_C = \sum_{(i,j)\in\mathcal{P}} \Big\{ [D(\mathbf{f}_i,\mathbf{f}_j) - m_p]_+^2 / |\mathcal{P}| + \lambda_n I_i \Big[m_n - \min_{k\in\mathcal{N}} D(\mathbf{f}_i,\mathbf{f}_k) \Big]_+^2 / |\mathcal{P}_i| + \lambda_n I_j \Big[m_n - \min_{k\in\mathcal{N}} D(\mathbf{f}_j,\mathbf{f}_k) \Big]_+^2 / |\mathcal{P}_j| \Big\} \quad (5)$$

where $\mathcal{P}$ is a set of all positive pairs in fully-convolutionally extracted features in a minibatch and $\mathcal{N}$ is a random subset of fully-convolutional features in a minibatch that will be used for negative mining. I_i is short for $I(i,k_i,d_t)$, which is an indicator function that returns 1 if the feature k_i is located outside a sphere with diameter d_t centered at feature i and 0 otherwise, where $k_i = \text{argmin}_{k\in\mathcal{N}} D(\mathbf{f}_i,\mathbf{f}_k)$. $|\mathcal{P}_i| = \sum_{(i,j)\in\mathcal{P}} I(i,k_i,d_t)$ is the number of valid mined negatives for the first item ($|\mathcal{P}_j|$ for the second item) in a pair. The normalization term for negative pairs is simply averaging all valid negative pairs equally. λ_n is a weight for negative losses and we simply used 0.5 to weight positives and negatives equally. Similarly, we can form a triplet loss with hard negatives mined within a batch:

$$L_T = \frac{1}{Z} \sum_{(i,j)\in\mathcal{P}} \Big(I(i,k_i) \Big[m + D(\mathbf{f}_i,\mathbf{f}_j) - \min_{k\in\mathcal{N}} D(\mathbf{f}_i,\mathbf{f}_k) \Big]_+ + I(j,k_j) \Big[m + D(\mathbf{f}_i,\mathbf{f}_j) - \min_{k\in\mathcal{N}} D(\mathbf{f}_j,\mathbf{f}_k) \Big]_+ \Big) \quad (6)$$

where $Z = \sum_{(i,j)\in\mathcal{P}} (I(i,k_i) + I(j,k_j))$, a normalization constant. The above equation finds the hardest negatives for both $(i,j) \in \mathcal{P}$ (Fig. 3). Here $\mathcal{P}$ is the set of all positive pairs in the batch. Note that we followed Hermans et al. [15] and used non-squared loss to mitigate features from collapsing into a single point. Experimentally, we still observed that the fully-convolutional hardest triplet loss is prone to collapse (all features converge to a single point). Instead, we mix the above triplet loss with randomly sampled triplets to mitigate the collapse. Similar to Eq. 6, we weigh both randomly subsampled triplets and hard-negative triplets equally.

Quadruplet을 형성함으로 얻는 이득은, 기존의 hard triplet mining에서 anchor에 대한 hard negative만을 고려하는 것에서 anchor에 대한 positive가 그 hard negative와 postive일 수 있는 지점을 고려할 수 있다. (false negative) 즉 anchor에 대해서 positive인 지점이 negative인 지점과 서로 positive pair임에도 학습을 통해 멀어지는 것을 피할 수 있다.

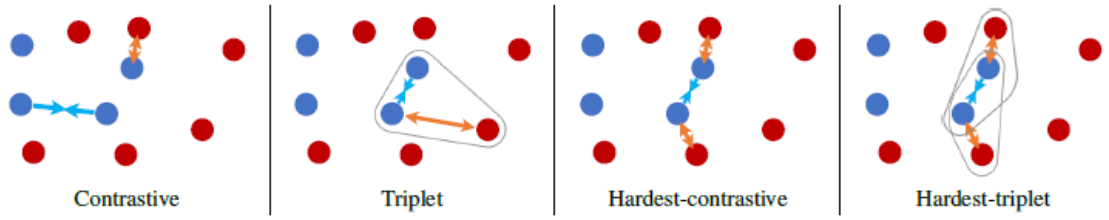

Figure 3: Sampling and negative-mining strategy for each method. Traditional contrastive and triplet losses use random sampling. Our hardest-contrastive and hardest-triplet losses use the hardest negatives.

3. Hash-based negative filtering -> Eq5 와 , Eq 6 의 I(I, j_i, d_t) Filtering 하는 부분이 품이 많이 든다. 이 부분을 hase-based filtering 하여 빠르게 filtering 할 수 있도록 한다.

Achievement

- Low-level preprocessing이나 3D patches의 활용 없이 faster, memory efficient하게 높은 성능으로 고화질의 feature를 판별해낼 수 있었다. (geometric correspondence의 추출.)

Limit

-

아이디어 및 논문에 관한 생각

- Minkloc는 Embedding Sapce로 보내므로 Fully Convolutional이 아니다. 따라서 metric lerning에서의 i.i.d 조건은 지켜질 것으로 생각된다. 또한 이미 overlap을 통해 negative pair를 뽑은 후 가장 가까운 Hard negative를 mining하는 것이기에, False Negative는 없을 것으로 보인다.
- 그러나 마찬가지로 55k의 features이기에 모든 pairwise distance는