Imperial College London

2 February 2024

Weekly Meeting

Week 11

Ahmad Agus Widodo (02546516) , PhD student, Department Civil and Environmental Engineering, Geotechnics Section

Updating the previous work

- Waiting the email from Andrew about the availability of RGB dataset
- Migration from Colab to Jupyter notebook (some code do not work)

Convolutional

$$h_0 = \left(\frac{h - f + p}{s} + 1\right) \tag{1}$$

$$w_0 = \left(\frac{w - f + p}{s} + 1\right) \tag{2}$$

The height h and width w of the input image define the size of the data being processed. The convolution operation uses a filter or kernel of size f, which scans over the input image. Padding p is applied around the input image to control the size of the output feature map and to allow the kernel to operate at the edges of the input image. The stride s, which defines the step size with which the filter moves across the input image, influences the downsampling factor and thus the size of the output feature map.

Kernel Function

$$(F * g)(x) = \sum_{x_i \in N_x} g(x_i - x) f_i \qquad (3)$$

The equation is expressed as a sum over x_i in the neighborhood N_x, with each term being the product of $g(x_i - x)$ and f_i. Here, $(F * g)(x)$ denotes the convolution of F with g evaluated at x, where f_i represents the value of F at x_i, and $g(x_i - x)$ represents the value of g at $x_i - x$.

Let $\{x_{ek}|k < K\} \subset B_r^3$ be the kernel points and $\{W_k|k < K\} \subset \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$ be the associated weight matrices that map features from dimension D_{in} to D_{out}. We define the kernel function g for any point $y_i \in B_r^3$ as

$$g(y_i) = \sum_{k<K} h(y_i, x_{ek}) W_k \tag{4}$$

where h is the correlation between x_{ek} and y_i, designed such that it should be higher when x_{ek} is closer to y_i. This design implies that h evaluates the degree of closeness or similarity between points, assigning greater values as the distance between x_{ek} and y_i diminishes.

Imperial College London

Deformable KPConv

- Deformable KPConv is particularly for tasks involving 3D shapes and structures.
- Flexibile by allowing the kernel points to adapt dynamically to the local geometry of the point cloud.

$$g_{\text{deform}}(y_i, \Delta(x)) = \sum_{k<K} h(y_i, x_{ek} + \Delta_k(x))W_k \tag{5}$$

Imperial College London

Loss Function

- a "repulsive" regularization loss between all pair off kernel points when their influence area overlap, so that they do not collapse together.
- the network generates shifts that fit the local geometry of the input point cloud.

$$L_{\mathsf{reg}} = \sum_{x} \left(L_{\mathsf{fit}}(x) + L_{\mathsf{rep}}(x) \right) \tag{6}$$

$$L_{\mathsf{fit}}(x) = \sum_{k<K} \min_{y_i} \left(\frac{\|y_i - (x_{ek} + \Delta_k(x))\|}{\sigma} \right)^2 \tag{7}$$

$$L_{\mathsf{rep}}(x) = \sum_{k<K} \sum_{\substack{l<K \\ l\neq k}} h(x_{ek} + \Delta_k(x), x_{el} + \Delta_l(x))^2 \tag{8}$$

Backpropagation

$$R_j = \sum_k \frac{|z_{jk}|}{\sum_j |z_{jk}|} R_k \tag{9}$$

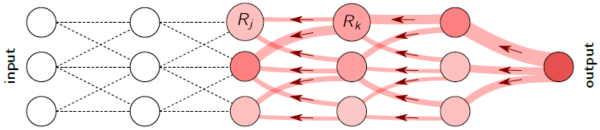

Figure 1: Back propagation

Forward : Input → Hidden Layer Calculation → Output Backward : Output → Loss Function → Differential Calculation → Weighted Update → Hidden Layer Calculation → Output

Imperial College London

Follow Up

- Issue: point clouds with color information usually have 3 scalar fields representing the Red, Green, and Blue values for each point
- Waiting email from Andrew related data availability of RGB type.
- Exploring how to use PyntCloud : library that offers functionalities to work with scalar fields in point clouds and to manipulate these scalar fields to achieve the desired RGB representation.
- link: $https : //pyntcloud.readthedocs.io/en/latest/scalar_fields.html$

Reference

[1] Martin Kada and Dmitry Kuramin. "ALS point cloud classification using Pointnet++ and KPConv with prior knowledge". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 46 (2021), pp. 91–96.

[2] Hugues Thomas et al. "Kpconv: Flexible and deformable convolution for point clouds". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6411–6420.

[1] [2]