Solution to Research Engineer Test ANGELSWING DEVELOPMENT TEST (Written)

Submitted By : Anish Manandhar

Email : luckyanish47@gmail.com

Cell: +977-9841510668

Q. 1 After training a deep learning model for multiple-class object detection, how would you approach fine-tuning to improve its performance? Discuss the strategies you would use for adjusting hyperparameters and optimizing the model's accuracy on a validation set (Max 300 words)

Fine tuning is making little changes to a pre-trained model to enhance its performance on a particular task. There are various approaches among which Genetic algorithm can be the most suitable in hyperparameter optimization and optimizing the model's accuracy on the validation set. In multiple class object detection, hyperparameters play a crucial role in the training process, but determining the best values for them can be difficult. Genetic algorithm is one of the most famous metaheuristic algorithms unlike traditional methods like grid searches, random search the large number of hyperparameters, unknown relationships between them, and the time-consuming nature of evaluating each set of hyperparameters can be sorted out. Genetic algorithms are a viable solution for optimum hyperparameter searches. Hyperparameter visualization with the genetic algorithm helps us to select appropriate learning rate, momentum, weight decay, translate , hsv and other hyperparameters.

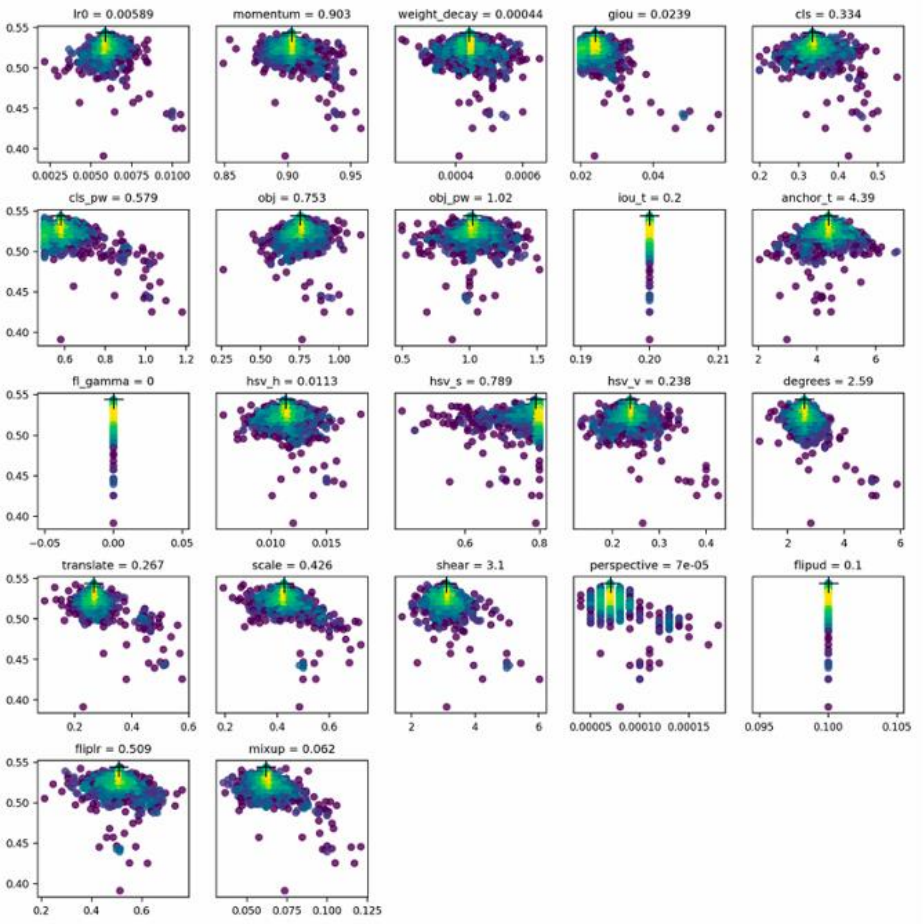


Fig: Hyperparameter Optimization using genetic algorithm adapted [Hyperparameter evolution - Ultralytics YOLO Docs]

Applying transfer learning to move the knowledge gained from the source dataset to the target dataset is another way to solve the problem. For instance, the model trained on the ImageNet dataset may extract more generic image properties, such as edges, textures, forms, and object composition, even though most of the images in the dataset have nothing to do with aerial

images. Aerial images recognition may also benefit from these comparable characteristics. Optimizing the model can also be achieved by freezing the convolutional and max pooling layers to prevent them from overlapping and by adjusting the sole classifier component.

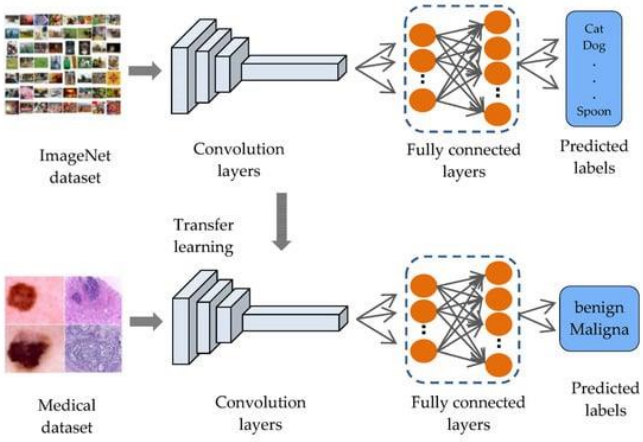


Fig: Fine tuning using transfer learning adapted from [Sensors | Free Full-Text | Incorporating a Novel Dual Transfer Learning Approach for Medical Images (mdpi.com)]

However, to enable fine-tuning of the full model without freezing specific layers, we can use LoRA, a technique that reduces the number of trainable parameters by adapting only a small subset of the model's weights , LoRA is greatly used for finetuning LLMs and can be used to fine tune multiple -class object detection as explained in the article :[How to Fine-tune Florence-2 for Object Detection Tasks (roboflow.com)]

2. You have a dense point cloud from a laser scan. Imagine this as a bunch of points on a 3D surface. Each point would have a normal vector on a smooth surface. Develop a basic algorithm to estimate the normal vector at each point in the point cloud

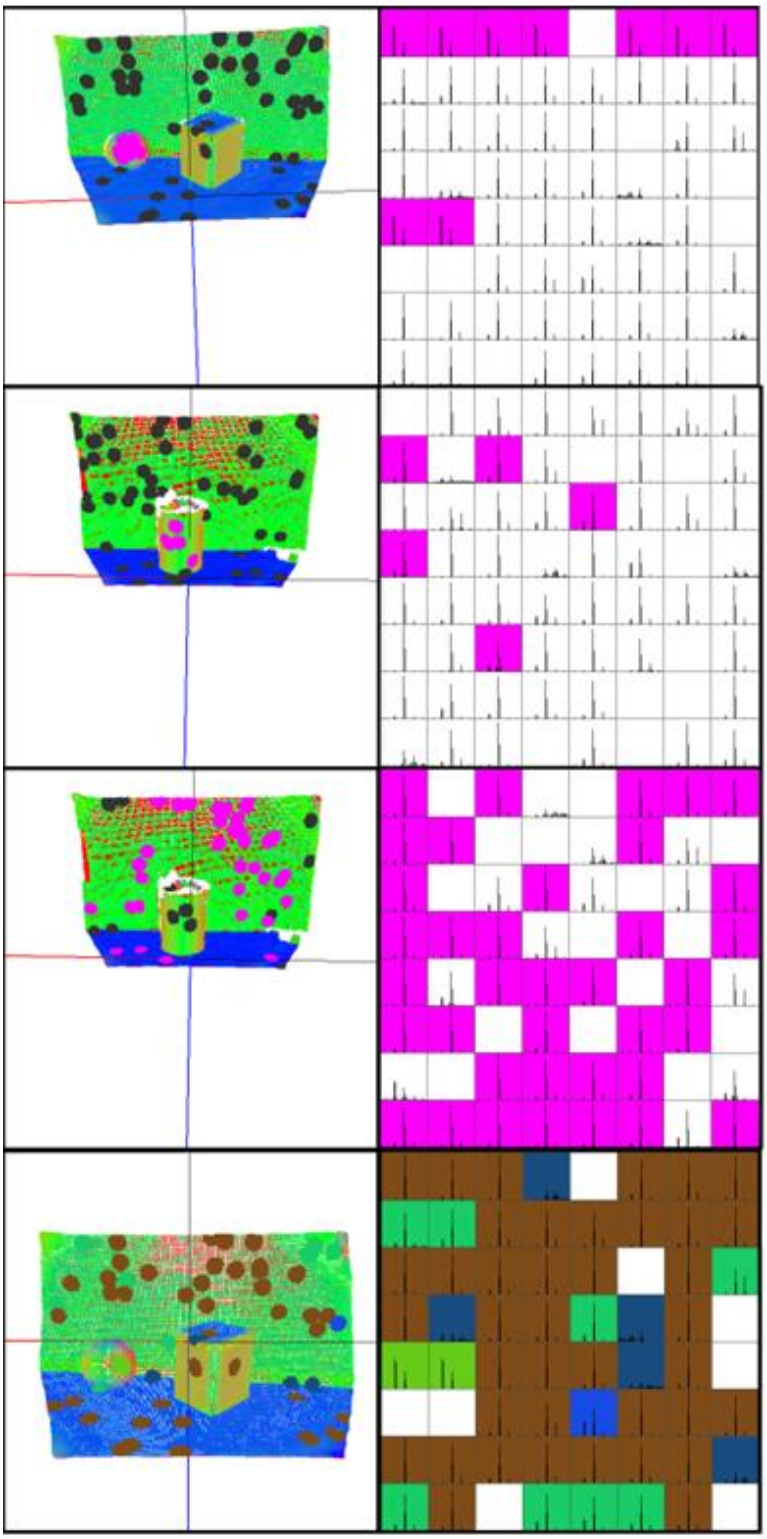


Fig: Normal Vector Estimation at each point in the point cloud, adapted from (PDF) Understanding 3D shapes being in motion (researchgate.net)

Normal vector computation using PCA Principal Component Analysis and SVD Singular Value Decomposition, PFH– Point Feature Histogram can be a wonderful technique to estimate the normal vector at each point in the point cloud as suggested in the paper.

Algorithm for Estimating Normal Vectors in a Dense Point Cloud:

Input: Dense point cloud data with 3D coordinates.

For each point in the point cloud:

Step 1: Identify the nearest neighbours of the cloud point.

Estimate normals in Point Cloud. In point cloud processing, “estimate… | by Simsangcheol | Medium suggests finding the neighbours using the KNN

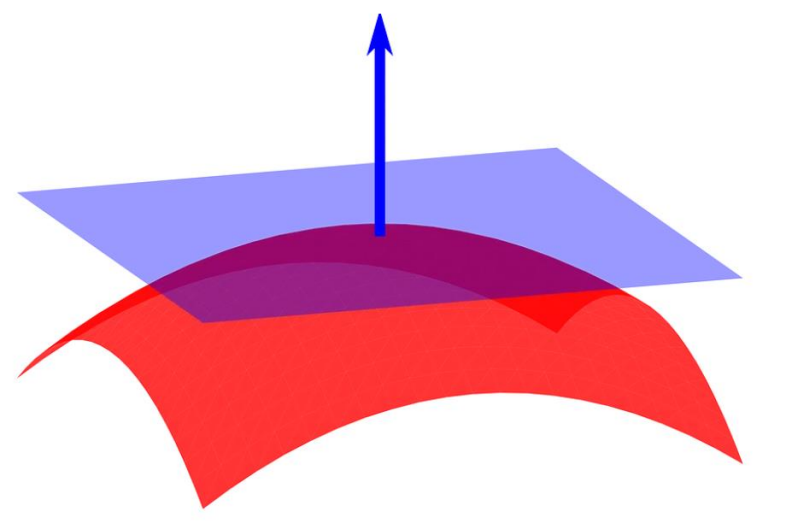
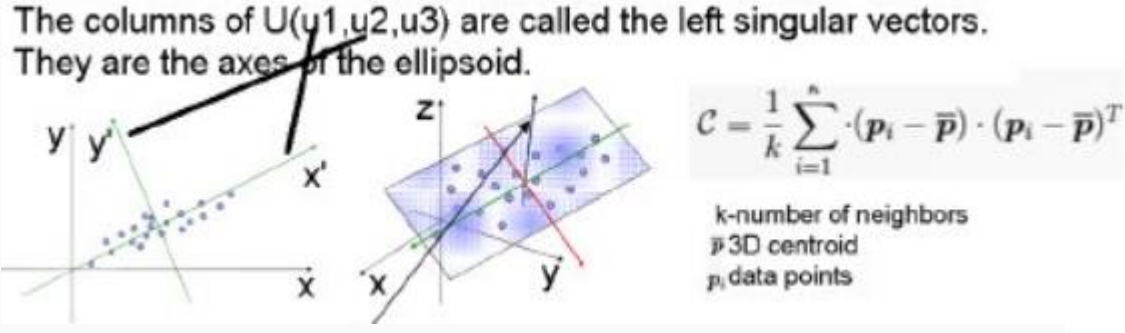


Fig: Nearest Neighbours

Step 2: Create a covariance matrix 'C' from the nearest neighbours.



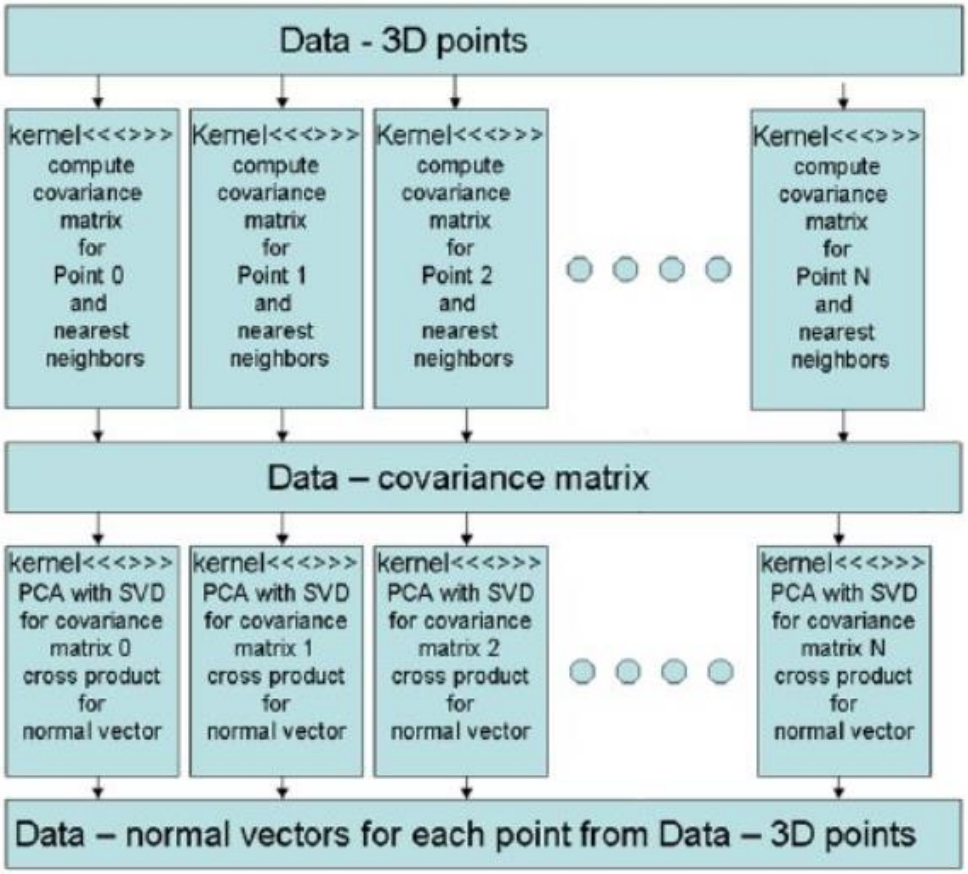
Step 3: Perform Principal Component Analysis (PCA) on 'C' to estimate the surface normal.

Step 4: Check if the normal vectors are consistently oriented towards the viewpoint.

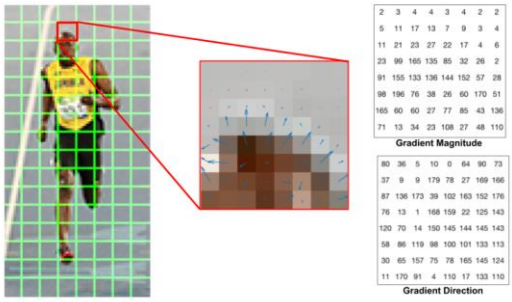
Step 5: If necessary, flip the normal vectors to ensure correct orientation.

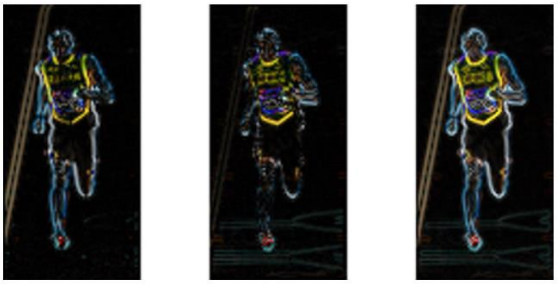
Output: Normal vectors estimated for each point in the dense point cloud.

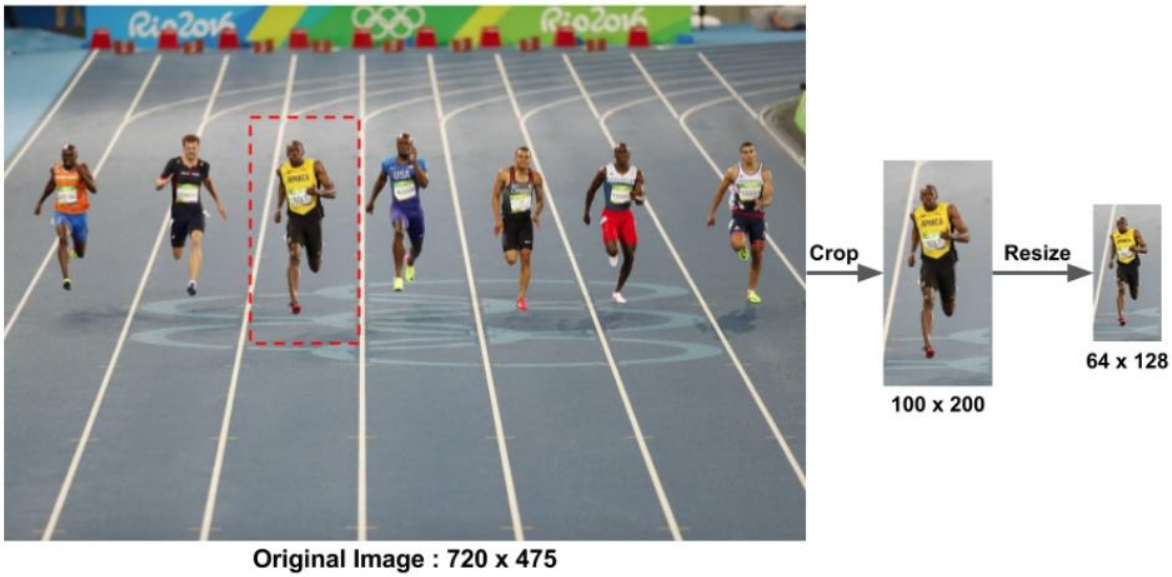
Modification:



The plan is to estimate the normal vector in two stages, assuming concurrent computation for every query point. Compute the covariance matrix first. In the second step, normal vector estimate is carried out for each query point in parallel using the SVD (Singular Value Decomposition) method. The implementation of CUDA Kernels for SVD solver and covariance matrix computing was the primary contribution. To sum up PFH descriptors can also be used to identify and match similar shapes and surfaces in different point clouds, which is useful for tasks like object recognition and alignment in 3D space.







Adapted from [Histogram of Oriented Gradients explained using OpenCV (learnopencv.com)]

3. You are given points that define a complex polygon, such as a detailed coastline. Your task is to simplify this polygon by reducing the number of points while maintaining its general shape and characteristics. Describe two different algorithms that can accomplish this polygon simplification. Your description should include: The main idea behind each algorithm. How each algorithm decides which points to keep or remove. The advantages and potential drawbacks of each approach For example, Consider simplifying a coastline polygon with hundreds of points into a simpler representation with fewer points, while still preserving its recognizable shape. (You may include pseudocode, basic implementations of one or both algorithms)

Two algorithms can be used to simplify the polygon by reducing the number of points while maintaining the general shape and they are explained below:

- Visvalingam's algorithm algorithm that estimates a curve composed of line segments to a similar curve with fewer points

Main idea

The program looks for a chain that is comparable and has less points. Points are eliminated from the least important to the most significant order after being assigned an importance depending on local variables. The relevance of each point in Visvalingam's method is determined by the triangular area it adds.

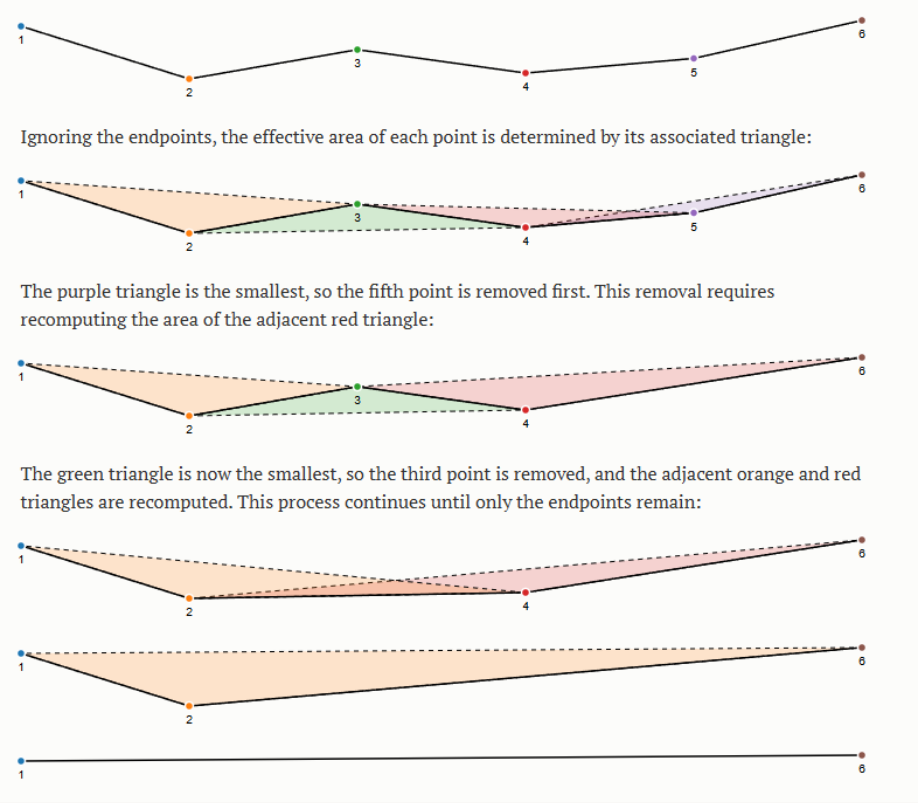


Fig: Explanation of the algorithm adapted from Line Simplification (ocks.org)

- Douglas Pecker algorithm is the simplified algorithm to make the complicated line segments (such as polygonal points) into the simplified format where it removes the points it can get away with.

Main idea:

The line is divided recursively using the method. It is initially provided every point from the first to the last. The first and last points to be kept are automatically marked. Next, it determines which point on the curve is the furthest from the approximating line segment between the end points, which is the line segment that has the first and last points as end points.

The point that is farthest from the line segment must be retained if it deviates more than ε from the approximation. The farthest point and the first point, followed by the farthest point and the last point—which includes the farthest point being marked as kept—are called upon by the algorithm iteratively. After the recursion is finished, a new output curve that includes only the points that have been designated as kept can be produced.

Visvalingam algorithm	Douglas-Peucker Algorithm
focuses on retaining significant points that define the shape of the line.	aims to reduce the number of points in a polyline while preserving its basic shape.
tends to produce more visually pleasing results in terms of retaining important details, such as bends and corners, in the line.	computationally efficient and widely used due to its simplicity and effectiveness in preserving the general appearance of the line.
Preferred when maintaining finer details, such as in maps where geographical features like coastlines or rivers require preservation of intricate shapes.	Commonly used in GIS and mapping applications where preserving the overall shape of the line is crucial, and a balance between simplification and detail is needed.
can sometimes remove important details, especially small but significant features of the line	does not differentiate between sharp spikes and shallow features, meaning that it will clean up sharp spikes that may be important.

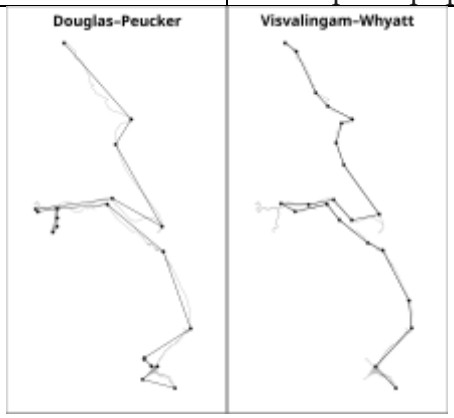


Fig: Comparison of two algorithms adapted from Visvalingam–Whyatt algorithm - Wikipedia

```
 # source: https://karthaus.nl/rdp/
function DouglasPeucker(PointList[], epsilon)
    # Find the point with the maximum distance
    dmax = 0
    index = 0
    end = length(PointList)
    for i = 2 to (end - 1) {
        d = perpendicularDistance(PointList[i], Line(PointList[1], PointList[end]))
        if (d > dmax) {
            index = i
            dmax = d
        }
    }

    ResultList[] = empty;

    # If max distance is greater than epsilon, recursively simplify
    if (dmax > epsilon) {
        # Recursive call
        recResults1[] = DouglasPeucker(PointList[1...index], epsilon)
        recResults2[] = DouglasPeucker(PointList[index...end], epsilon)

        # Build the result list
        ResultList[] = {recResults1[1...length(recResults1) - 1], recResults2[1...length(recResults2)]}
    } else {
        ResultList[] = {PointList[1], PointList[end]}
    }
    # Return the result
    return ResultList[]
```

Fig: Pseudo Code of Douglas Pecker Algorithm adapted from [Ramer–Douglas–Peucker algorithm - Wikipedia]