Geometric Set Classification and Segmentation via Point Transformer V3

Hatem Feckry (120220264) Seifeldeen Mohamed Galal (120220252)
Abdelrahman Ashraf (120220292) Ahmed Elsherbeny (120220271)

May 8, 2025

1 Introduction

Last week we trained a YOLO classifier for the data we generated and tested it on a proportionally large amount of new data. However, we didn't clearly explain the classification problem and the rules of inference. This week we will clearly outline how we turned the Happy Ending problem into a classification problem and explore it in a segmentation setting. This new viewpoint allowed us to choose an existing transformer architecture, Point Transformer V3, which we attempted to use. Unfortunately, we didn't manage to get the transformer to work locally on any of our machines for various reasons and remote deployment was equally challenging.

2 Happy Ending as a Classification Problem

The Erdős–Szekeres conjecture states that any set of points larger than or equal to $2^{n-2}+1$ will always has at least one convex polygon of size n. However, in sets less than this threshold some, rare, sets can be void of convex polygons. We have called these sets empty sets for ease of reference.

The empty sets often have a unique and specific geometric reason that allow them to avoid convex polygons. For example, a set of 8 points void of pentagons is a nested quadrilateral with relative rotation and/or edge scaling. These geometric solutions are the main interest of our project, so we aim to develop a framework that can efficiently generate empty sets of a given size. This approach aims to benefit future research by giving multiple examples and a dataset, something we didn't find publicly on the internet. We have tried multiple empirical optimizations to set generation and felt we reached a good position to extend this dataset using AI.

2.1 CNN Classifier

We used a classification YOLO model to validate that with enough data and network depth even a CNN can pick on the complex geometric structures in this problem. The setup was as the following:

1. Convert the sets into images (the input domain for YOLO)
2. Set images into two classes: Empty and Non-empty
3. Train the model on 2800 images and test on 700 new ones

This way the model can judge if the underlying set used for image generation had a convex polygon or not. This is a global decision based on the position of all points in the image as we kept the number of points per set constant. Through this setup, the model was trained through only 40 epochs and the loss-epoch trends clearly showed the model didn't overfit. Despite this limited training, it was able to achieve 99.3% accuracy in classifying an equal split of empty and non-empty sets. We were surprised by the degree of the success this approach had, and decided to extend the problem to be point-wise decision, i.e. segmentation.

3 Segmentation and Point-wise Inference

A segmentation task in CV is similar to classification but instead of a global label to the whole scene, a label is assigned to each point or pixel in the input. This is a much harder task, but also much more helpful to our problem. Instead of classifying the sets, we can "extract" the points within the set that together make a set void of polygons of a given size. This not only better forces the model to learn the exact geometric relationship between the points, but also can serve as an intermediate output for further processing.

Formally, for a point set $\mathcal{P} = \{p_1, ..., p_n\}$ where each $p_i = (x_i, y_i)$, we define the segmentation target as:

$$s_i = \begin{cases} 1 & \text{if } p_i \in \mathcal{S} \text{ (critical for emptiness)} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

4 Point Transformer V3 Architecture and Point Serialization

Point Transformer V3 (PTV3) is a state-of-the-art neural network architecture designed for processing unordered point cloud data, such as LiDAR scans or 3D geometric sets. Unlike CNNs, which require structured grid inputs like images, PTV3 operates directly on raw point coordinates, making it inherently permutation-invariant—meaning the order of input points does not affect the output.

Each point is initially represented by its coordinates (x, y) and optionally additional features such as color or intensity if applicable. A shared Multi-Layer Perceptron (MLP) projects these raw coordinates into a higher-dimensional feature space. Instead of processing all points at once, which is computationally expensive, PTV3 uses local attention blocks where points are grouped into neighborhoods via k-Nearest Neighbors (k-NN) or ball query algorithms. Within each neighborhood, a self-attention mechanism dynamically computes relationships between points, allowing the model to weigh their importance. Global information is aggregated through downsampling and upsampling layers, similar to a U-Net, ensuring both fine-grained and high-level reasoning.

Like traditional transformers, PTV3 employs residual connections and layer normalization to stabilize training and prevent vanishing gradients. For point-wise classification (segmentation), a final MLP maps the learned features to per-point logits, which can be interpreted as class probabilities—for example, distinguishing whether a point is part of an empty set or not.

Since transformers typically expect sequential inputs, point clouds must be serialized before processing. However, unlike language models where word order matters, point order should not influence the output. PTV3 handles this by treating points as an unordered set and using attention mechanisms that are invariant to permutations. Positional encodings, such as Fourier features or learned embeddings, help the model reason about spatial relationships without imposing an artificial order.

5 Why Point Transformer V3 is a Good Fit for the Happy Ending Problem

The Happy Ending problem deals with unordered geometric point configurations, making PTV3 a natural choice since it avoids the need for rasterization, unlike CNNs. By processing raw (x, y) coordinates directly, PTV3 preserves geometric relationships without distortion from grid-based approximations. The problem's output—identifying empty sets—should not depend on the order of points, a property inherently satisfied by PTV3 due to its permutation invariance.

Convexity constraints in the Happy Ending problem often depend on both local point arrangements, such as nested quadrilaterals, and global structure, such as overall point distribution. PTV3's hierarchical attention mechanism captures both scales effectively, allowing it to reason about fine-grained geometric patterns while maintaining an understanding of the entire set. Instead of just classifying entire sets, PTV3 can segment which points contribute to an empty set, providing interpretable insights into the underlying geometric constraints.

Additionally, PTV3's efficient attention mechanisms, such as grouped local attention, allow it to handle larger point sets (e.g., 28+ points) better than brute-force approaches or dense CNNs. This scalability is crucial for extending empirical verification of the

Erdős–Szekeres conjecture to higher polygon sizes, where manual analysis becomes impractical. By leveraging PTV3, we aim to automate the discovery of empty sets at previously intractable scales, advancing research on this long-standing geometric problem.

6 The results of running the model locally

the model is publicly available via a github repo that we downloaded. However, as none of us had experience with anaconda or pytorch, we couldn't get it to run. We first needed to work on linux which wasn't installed on the our main laptop. This led us to use an older laptop which we discovered didn't meet the minimum requirements to run a submodule of the model. We installed a fresh linux version on one our main laptop, however because of the mentioned lack of experience, we had many conflicts in the environment and linux crashed on us multiple times.

7 Conclusion and Future Work

While this week's attempt to implement Point Transformer V3 did not yield operational results due to technical deployment challenges, the process provided invaluable insights that will inform our next steps. The theoretical analysis confirmed PTV3's strong theoretical suitability for the Happy Ending problem, as demonstrated by our analysis above.

7.1 Implementation Challenges

Our implementation attempts revealed several technical hurdles:

- **Environment Setup**: Required CUDA version $\geq$ 11.7 but our hardware only supported 11.4
- **Memory Constraints**: The base PTV3 model requires $\sim$8GB VRAM for $n = 30$ point sets
- **Dependency Conflicts**: Between PyTorch $\geq$ 2.0 and legacy Anaconda packages

7.2 Solution Roadmap

The planned implementation strategy involves:

1. Containerized environment with Docker (CUDA 11.7+ support)
2. Validation on cloud platforms (AWS EC2 g4dn.xlarge instances)

3. Progressive testing with point set sizes: $8 \to 16 \to 24 \to 32$
4. Integration of attention visualization tools

7.3 Prospects for Success

The 99.3% classification accuracy achieved by our YOLO baseline (Section 2.1) suggests high probability of success:

$$P(\text{Success}|\text{PTV3 Implementation}) > 0.95 \quad \text{given proper implementation} \tag{2}$$

This motivates continued work on the segmentation approach to identify geometric configurations satisfying:

$$\mathcal{E}_k(\mathcal{P}) = 1 \quad \text{for} \quad k \geq 7 \tag{3}$$

The technical experience gained this week, while not producing immediate results, has significantly advanced our capacity to complete this implementation successfully.