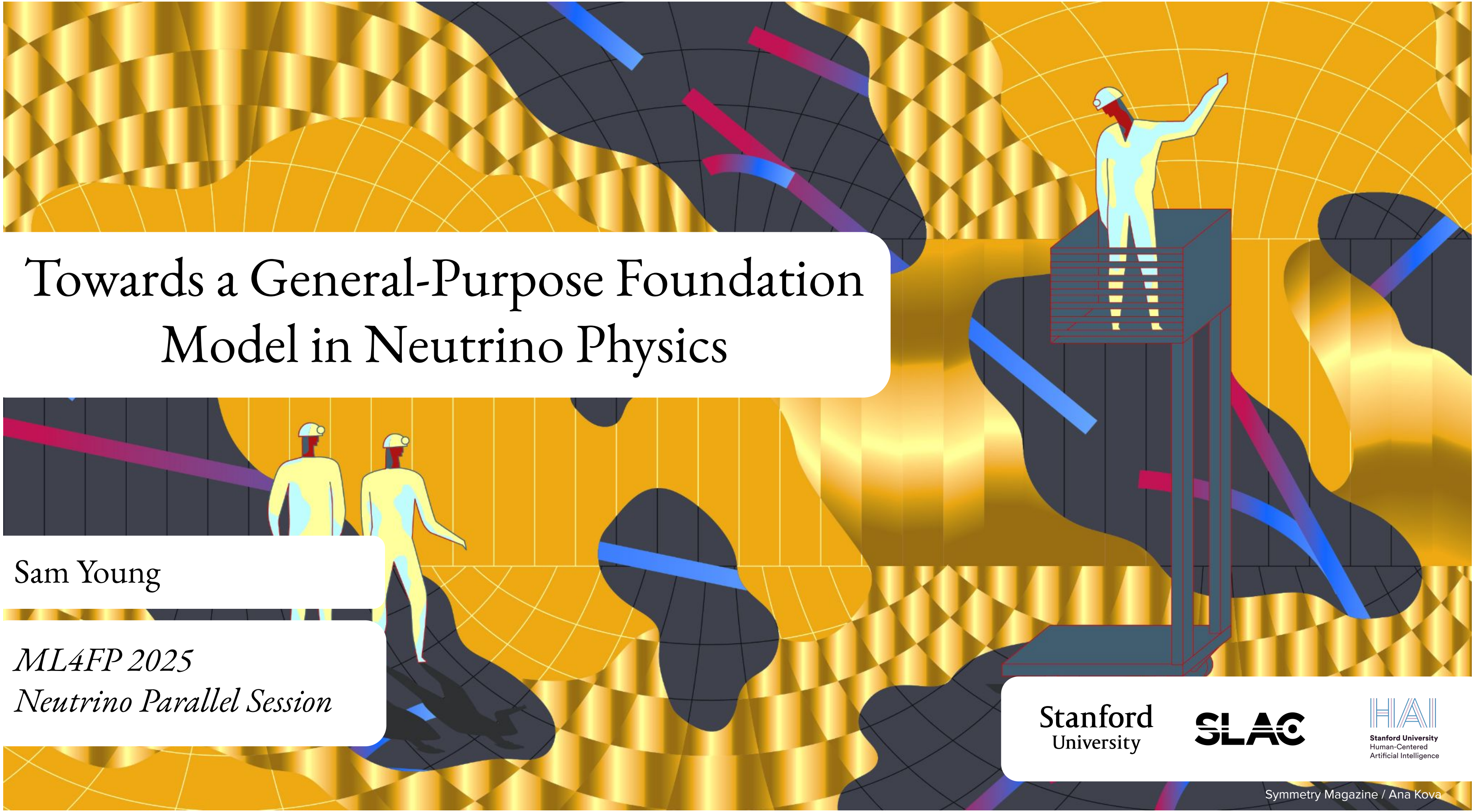
Towards a General-Purpose Foundation Model in Neutrino Physics

Sam Young

ML4FP 2025
Neutrino Parallel Session

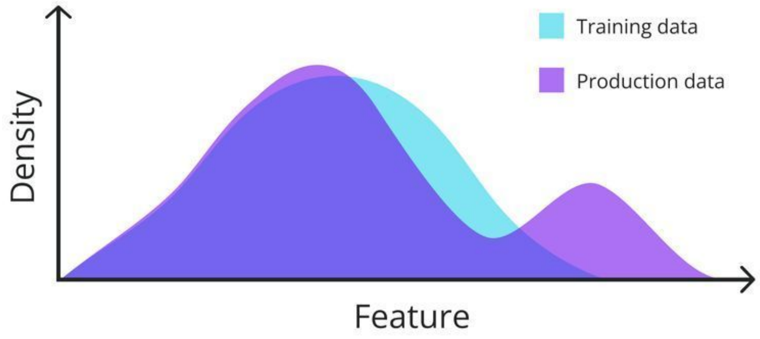
Stanford University
SLAC
Stanford University Human-Centered Artificial Intelligence



Symmetry Magazine / Ana Kova

Single-task supervised models are AMAZING, but...

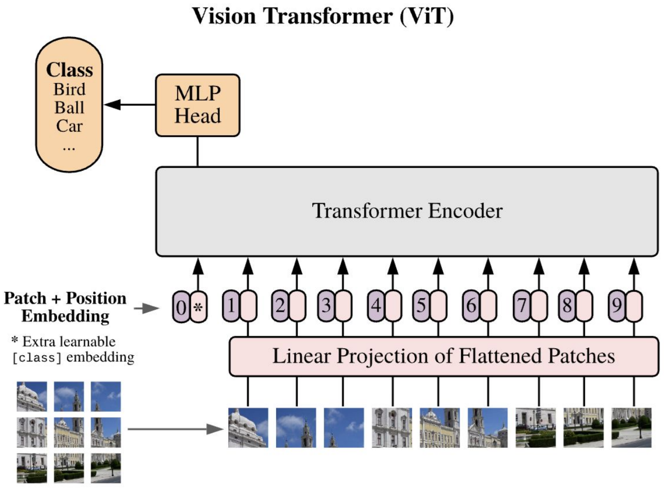
- Require large O(100,000 events) labeled datasets
  ⇒ heavy reliance on well-calibrated simulations.
- It can take a very long time to match reconstruction performance between simulation and real detector data.



"sim2real" gap

Single-task supervised models are AMAZING, but...

- They are domain experts. They only extract from data what they exactly need for their task.

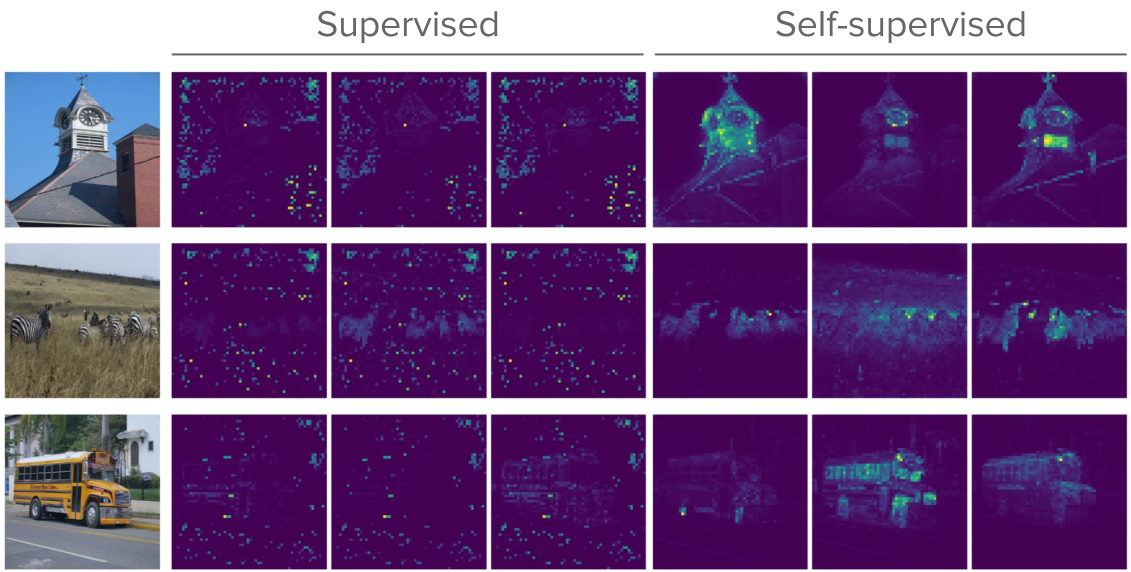


Attention maps

Attention maps from image classification in a vision transformer
[DINO (2104.14294)](#)

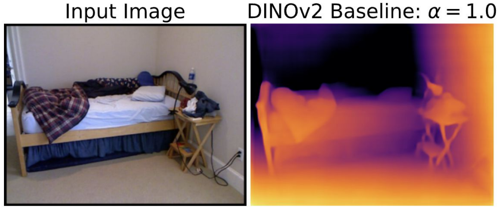
(Vision) foundation models are generalists

FM = learn more than the task requires so you can reuse it later



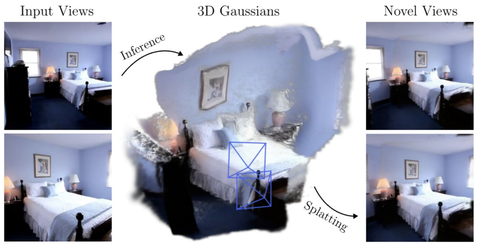
Attention maps from image classification and self-supervised tasks in a vision transformer
[DINO (2104.14294)](https://arxiv.org/abs/2104.14294)

(Vision) foundation models are generalists

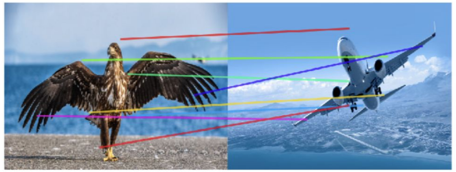


Monocular depth estimation [1]

DINO (SSL) [2]



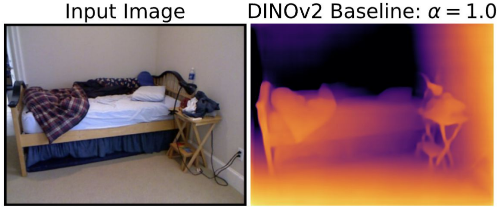
Multi-view Reconstruction [3]



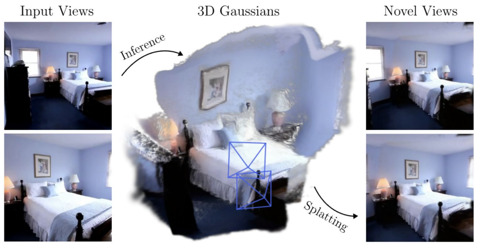
Point Correspondence [2]

Video Tracking [4]

(Vision) foundation models are generalists

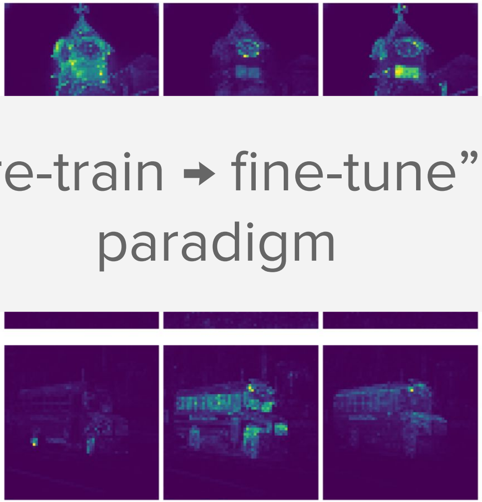


Monocular depth estimation [1]

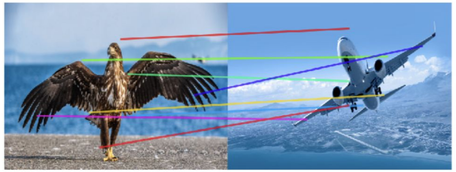


Multi-view Reconstruction [3]

“Pre-train ➔ fine-tune” paradigm



DINO (SSL) [2]



Point Correspondence [2]

Video Tracking [4]

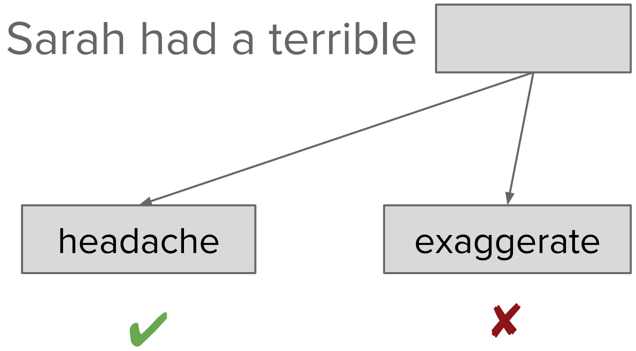
Ok… so how do you create one?

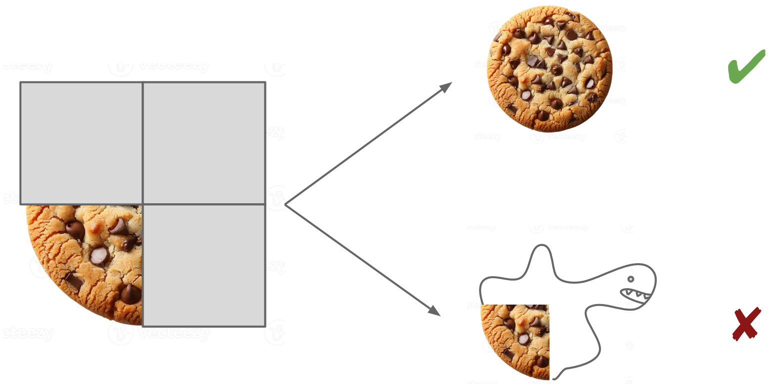
HTCAFM (how to create a foundation model)

- Make like things alike, unalike things unalike.
- **Create a hard task** that forces the model to understand the dataset you are giving to it.
  - The task should ideally sit on the phase transition of learning vs total collapse.

Simple Example: only give the model some of data, and ask it to tell you what is missing.

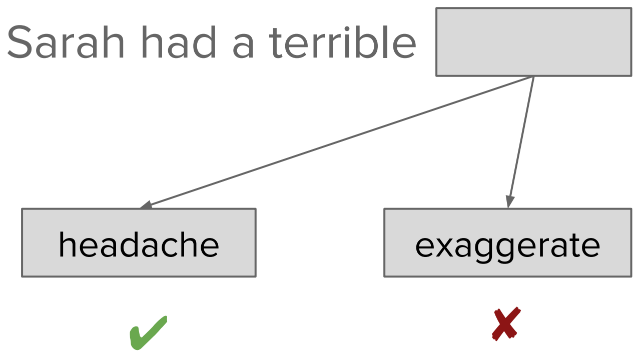
- The full data becomes your “truth” label ➔ no actual labels needed!

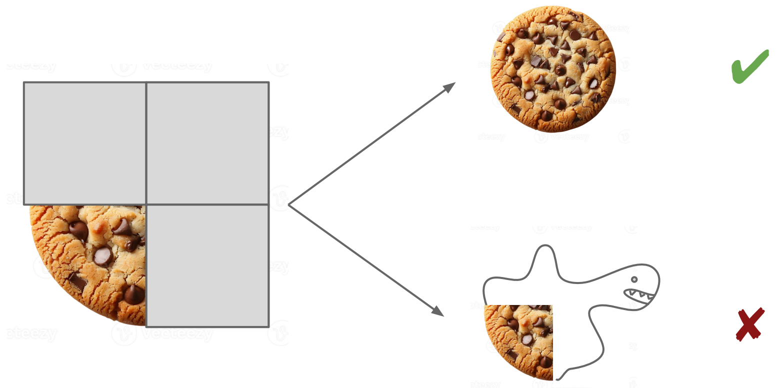




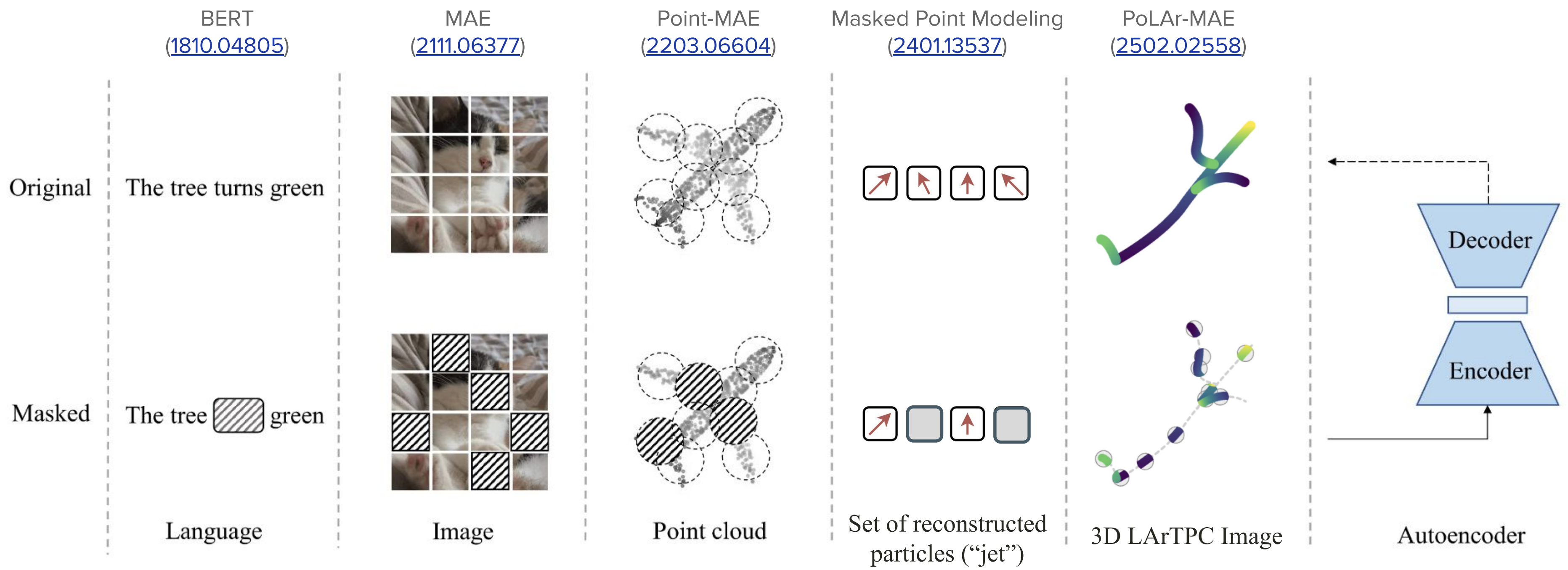
HTCAFM (how to create a foundation model)

- Note that in this scenario, the sentence and image are split into **chunks**, or **tokens**.
- An underlying assumption of these types of models, called **masked autoencoders** [1], is that the underlying data contains nuggets of information that contextually relate to one another.
    - E.g., words make grammatically correct sentences, quarters of a 2D cookie make a full 2D cookie. Different components of a par



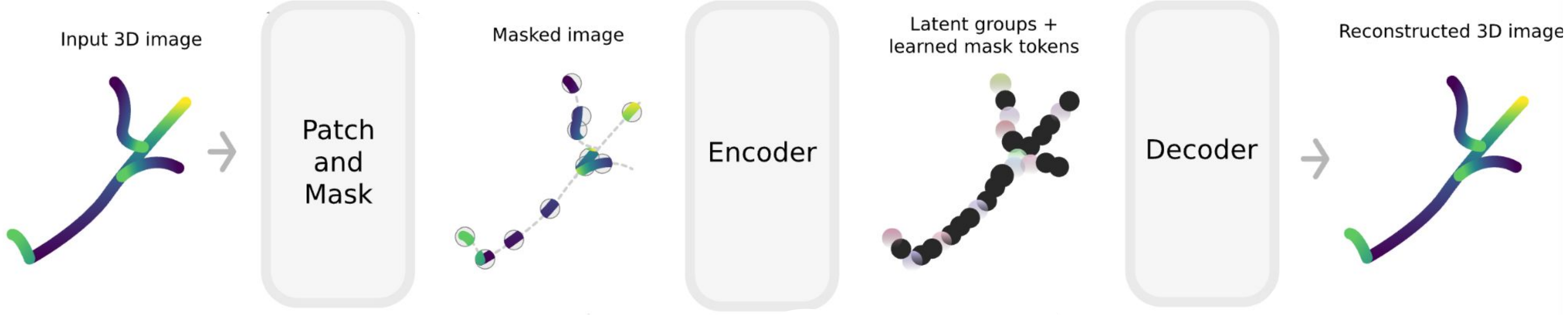


Masked Autoencoders Across Modalities



[adapted from PointMAE: arXiv:2203.06604]

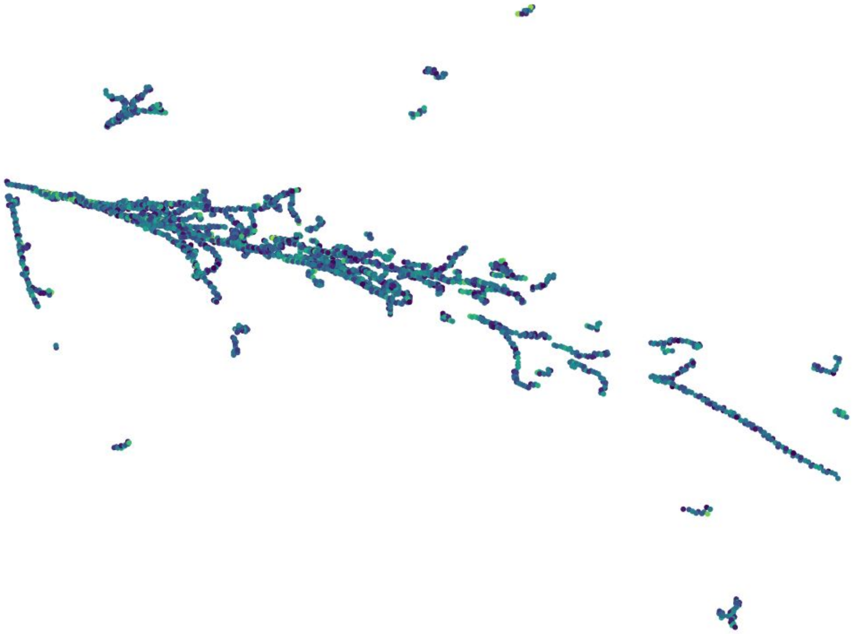
Point-based LAr Masked Autoencoder (PoLAr-MAE) [1]



Technical notes:

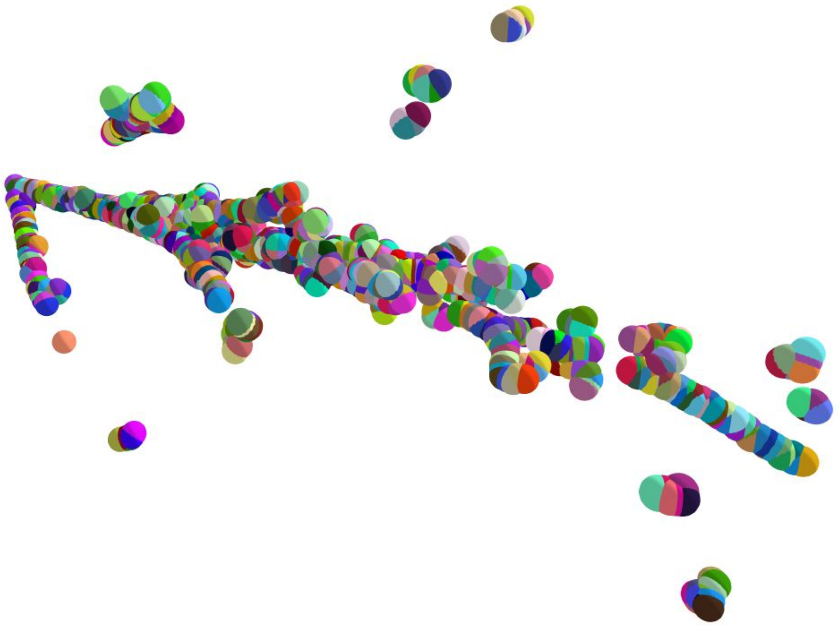
- We must find a way to patchify points?
- Encoder-decoder is **asymmetric**, i.e. encoder params $\gg$ decoder params.
- Masked tokens are not fed into encoder.

Patchification



3D charge deposited

Patchification



3D charge deposited

➜ Treat each point as sphere

Patchification

3D charge deposited

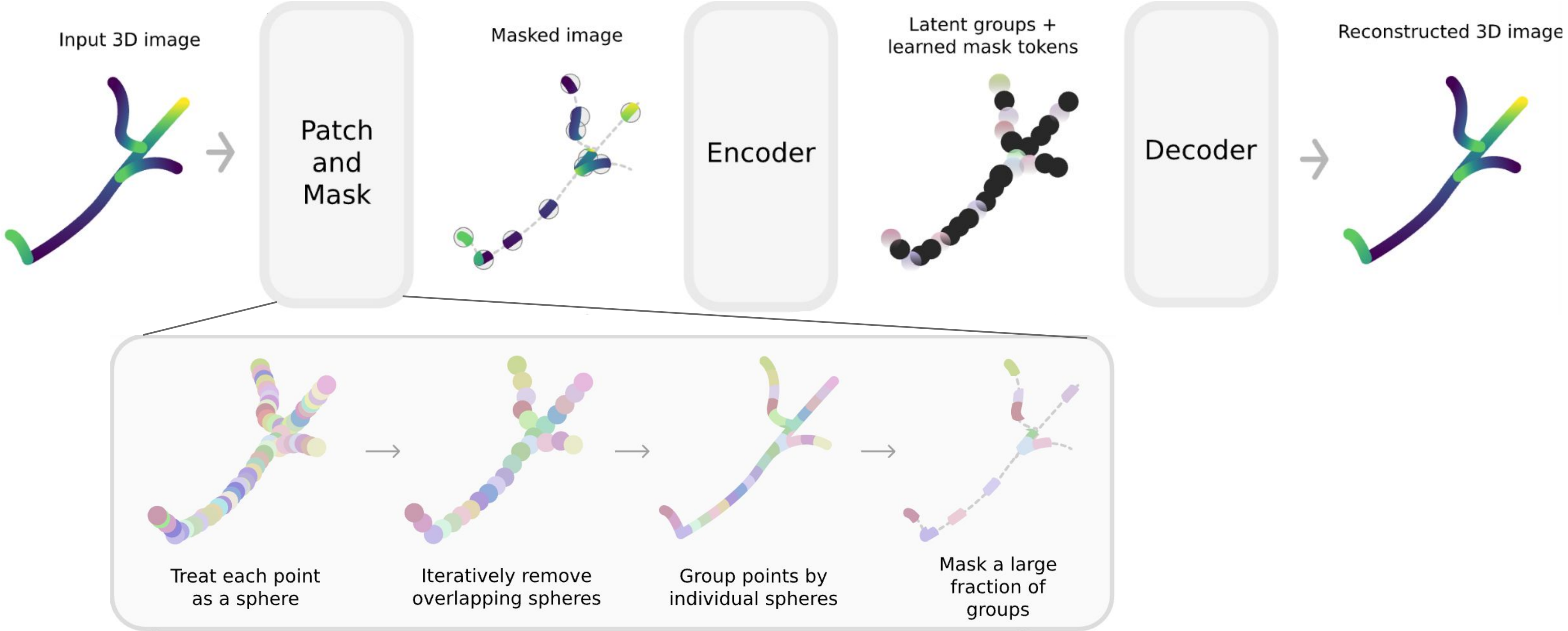
- ➜ Treat each point as sphere
- ➜ Remove overlapping spheres

Patchification

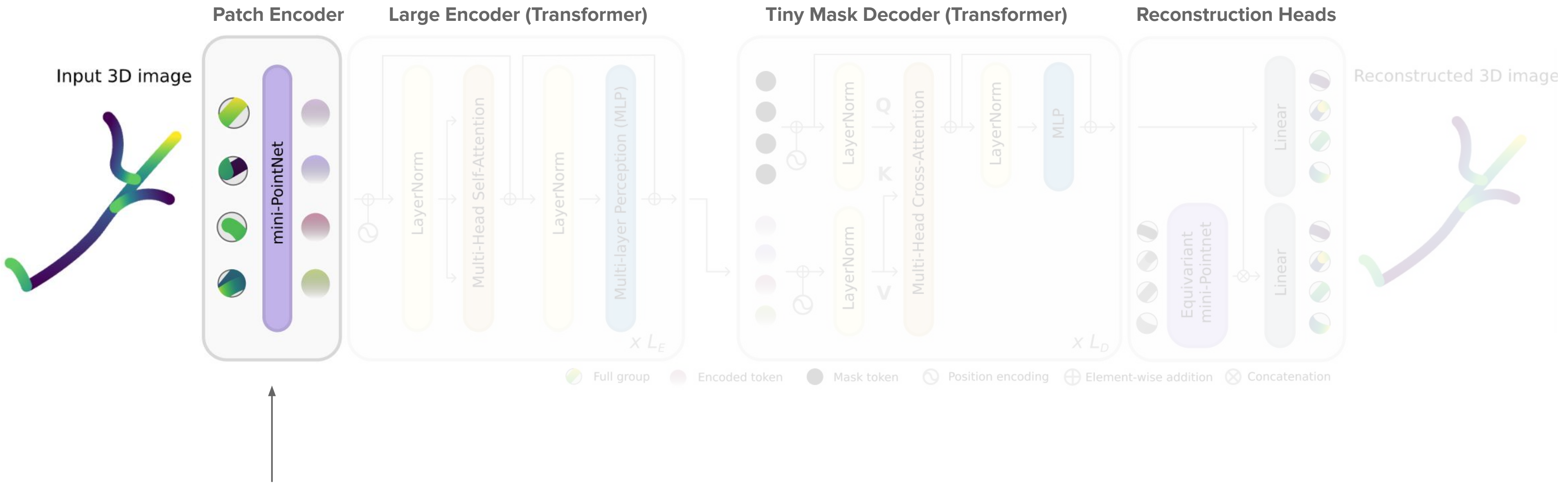
3D charge deposited

- ➜ Treat each point as sphere
- ➜ Remove overlapping spheres
- ➜ Ball query to get patches

Point-based LAr Masked Autoencoder (PoLAr-MAE) [1]

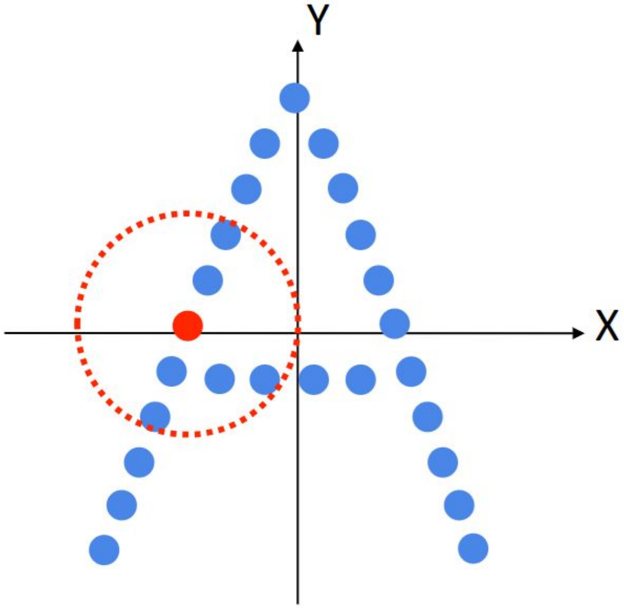


Point-based LAr Masked Autoencoder (PoLAr-MAE) [1]



Ok... how do you encode a variable number of points into a single feature vector?

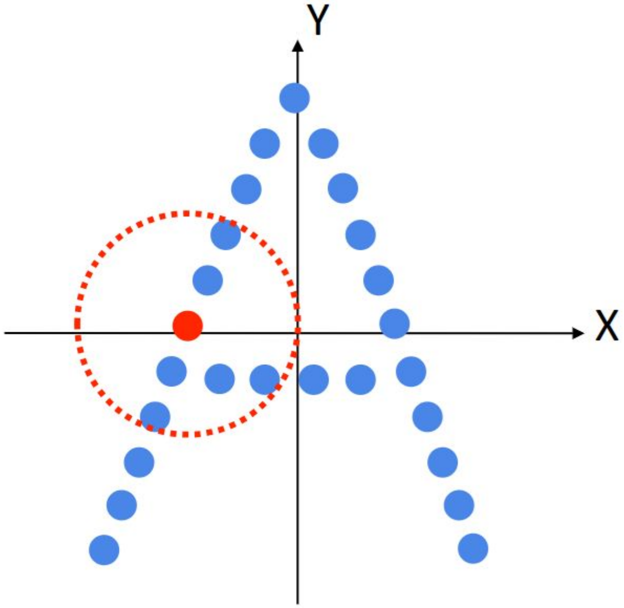
Point feature learning



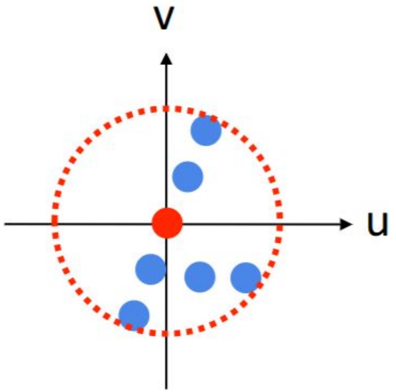
N points in (X,Y)

Adapted from Leo Guibas' slides in Stanford CS468

Point feature learning



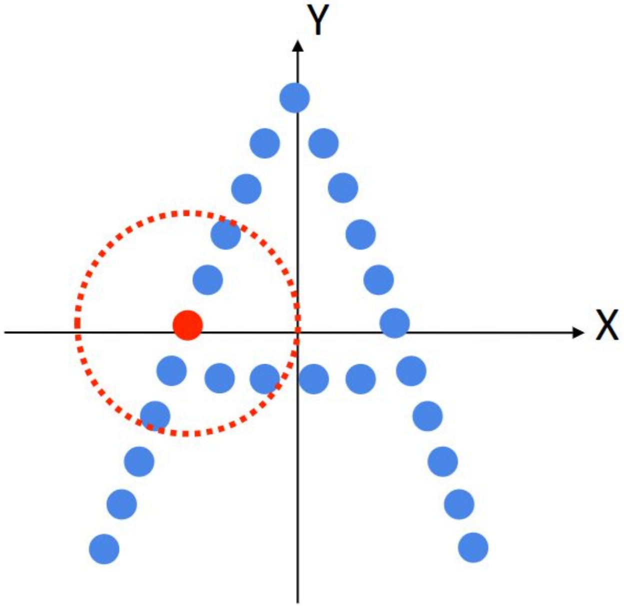
N points in (X,Y)



k points in local coordinates (u,v)

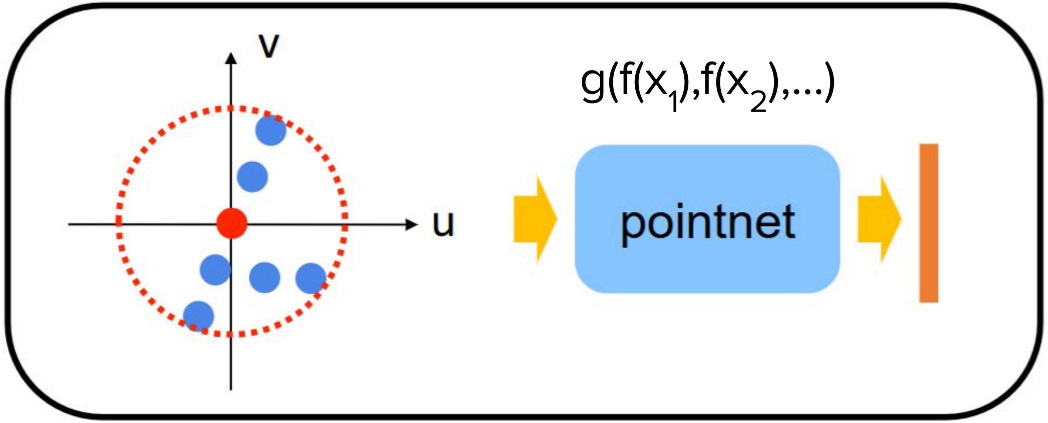
Adapted from Leo Guibas' slides in Stanford CS468

Point feature learning



N points in (X,Y)

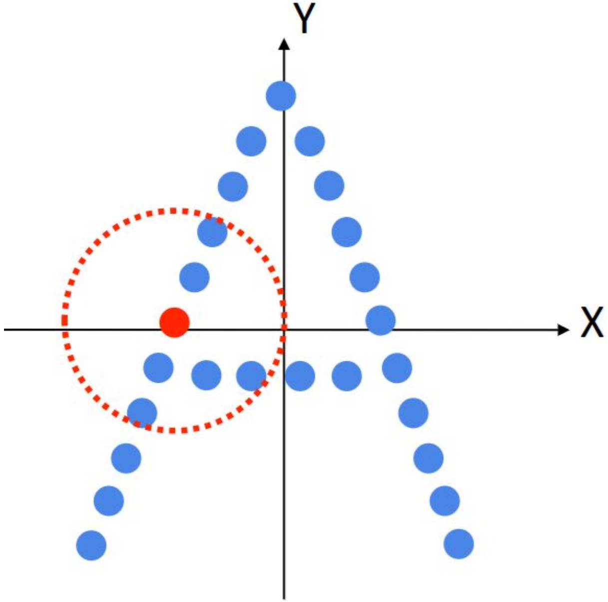
Apply pointnet at a local region



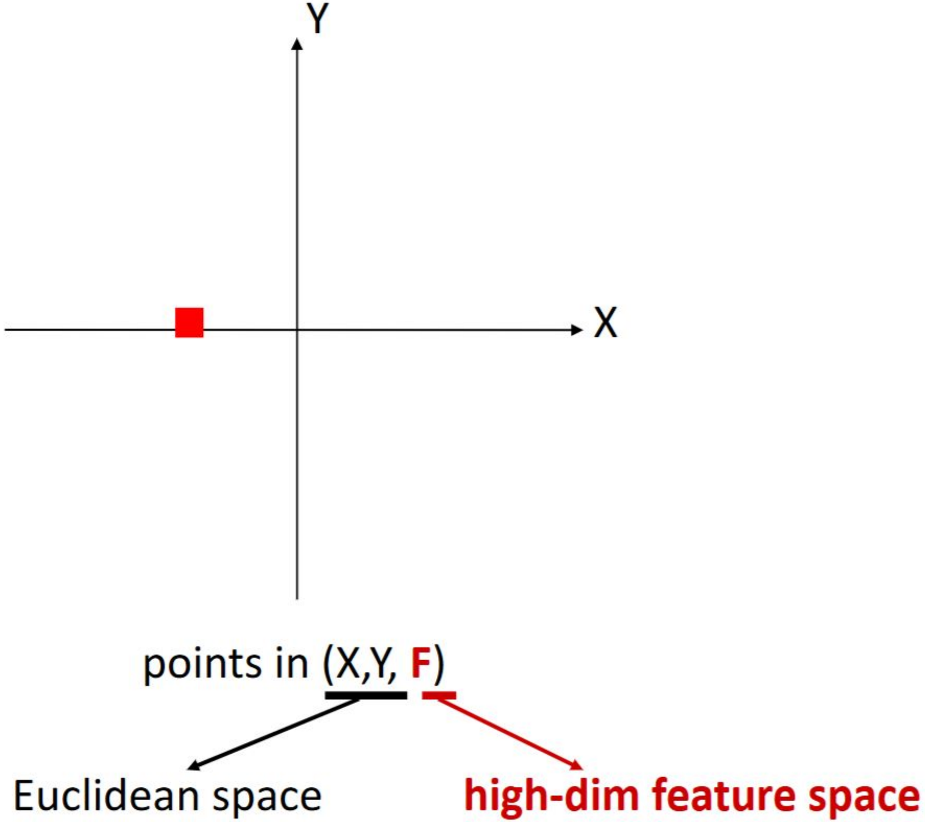
k points in local coordinates (u,v)

Adapted from Leo Guibas' slides in Stanford CS468

Point feature learning

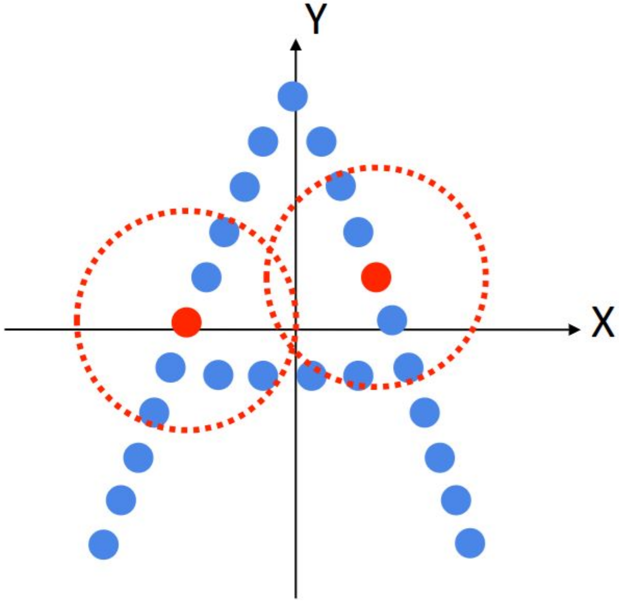


N points in (X,Y)

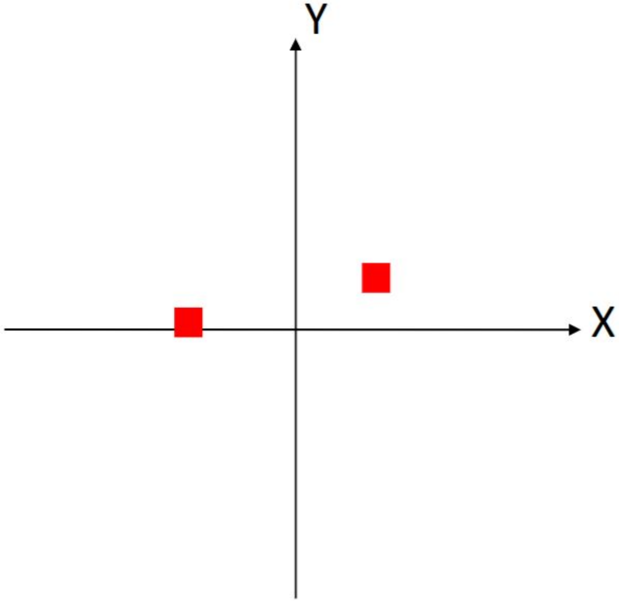


Adapted from Leo Guibas' [slides](#) in Stanford CS468

Point feature learning



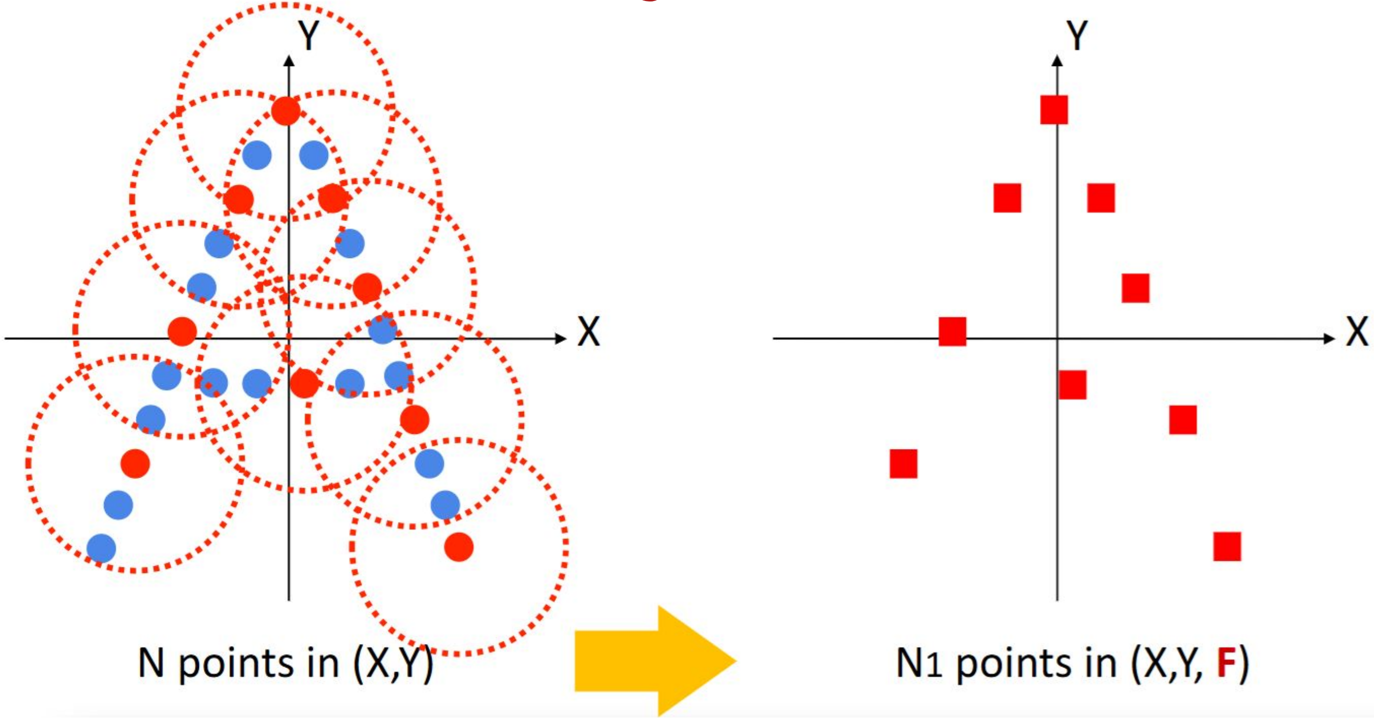
N points in (X,Y)



points in (X,Y, **F**)

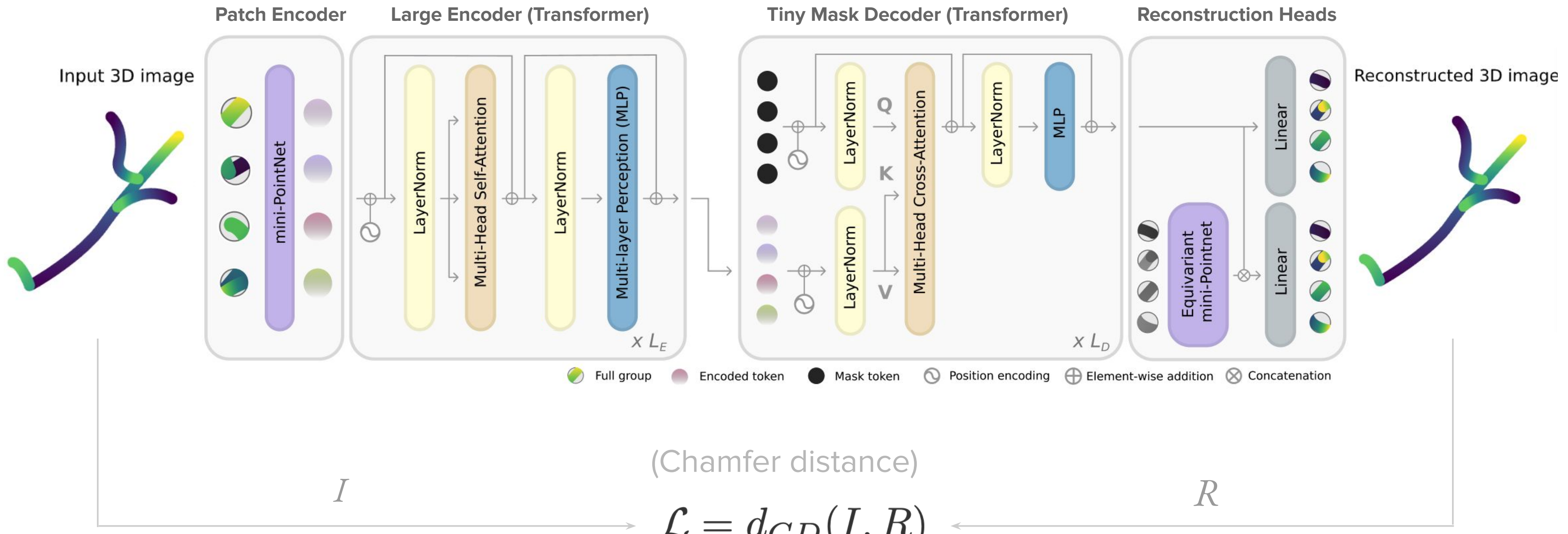
Adapted from Leo Guibas' [slides]() in Stanford CS468

Point feature learning



Adapted from Leo Guibas' slides in Stanford CS468

Point-based LAr Masked Autoencoder (PoLAr-MAE) [1]



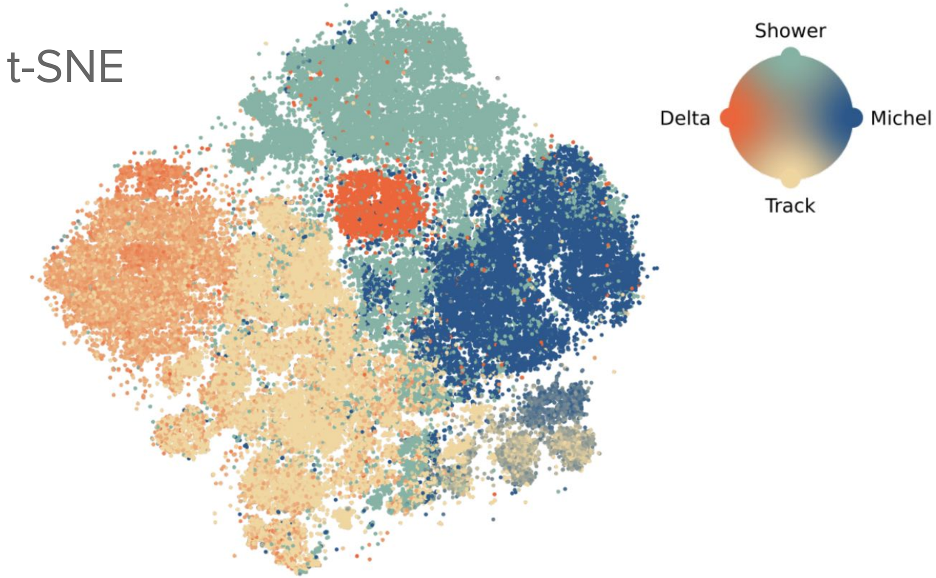
$$\mathcal{L} = d_{CD}(I, R)$$

$$= \sum_{x \in I} \min_{y \in R} ||x - y||_2^2 + \sum_{x \in R} \min_{y \in I} ||x - y||_2^2$$

Patch Representations

A look at patch representations.

Remember: one patch contains a group of pixels, so can contain >1 particle type.

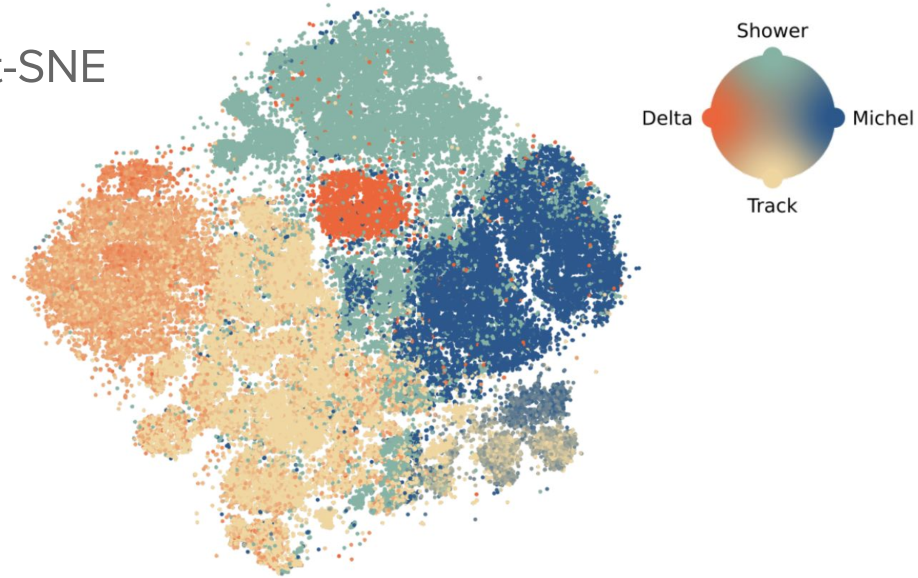


Patch Representations

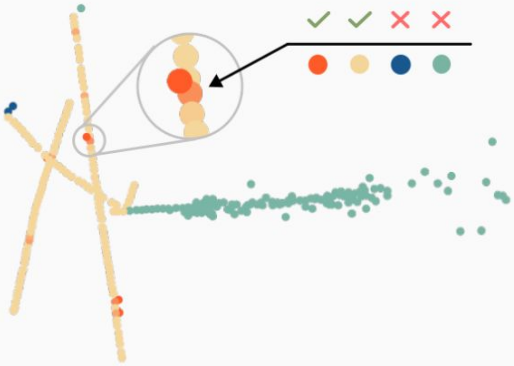
A look at patch representations.

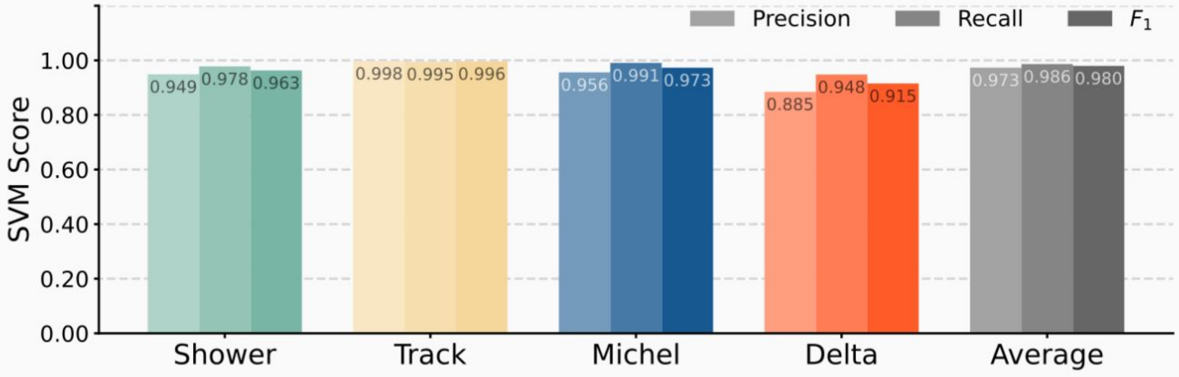
Remember: one patch contains a group of pixels, so can contain >1 particle type.

t-SNE



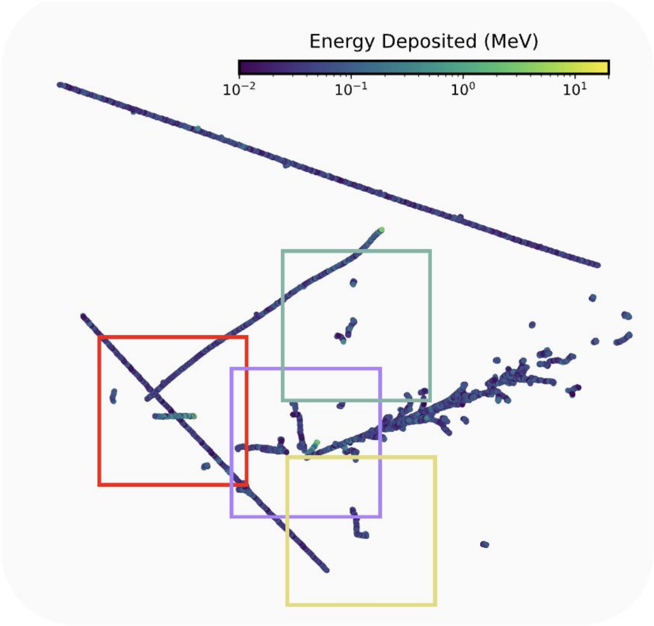
Patch makeup semantic segmentation

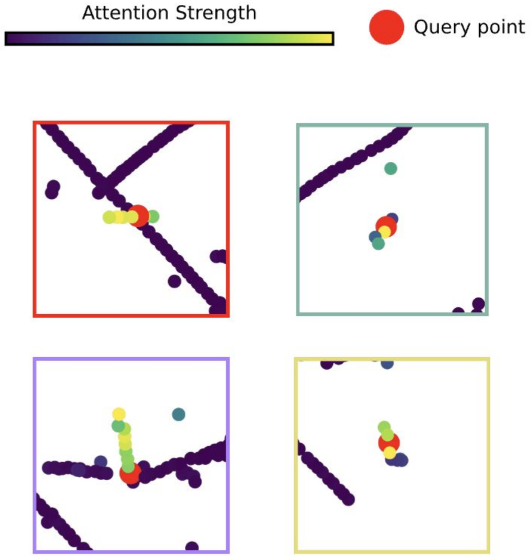




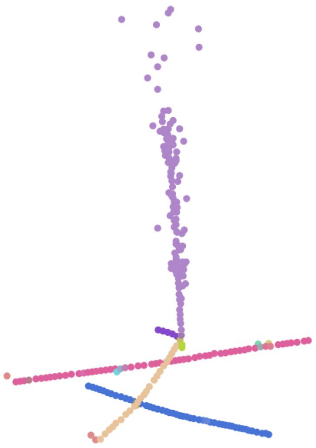
A Hint at Emergence: Attention Scores

$$Attention(Q,K,V) = \underbrace{\text{softmax}(\frac{QK^T}{\sqrt{d_k}})}_{\text{scores}} V$$





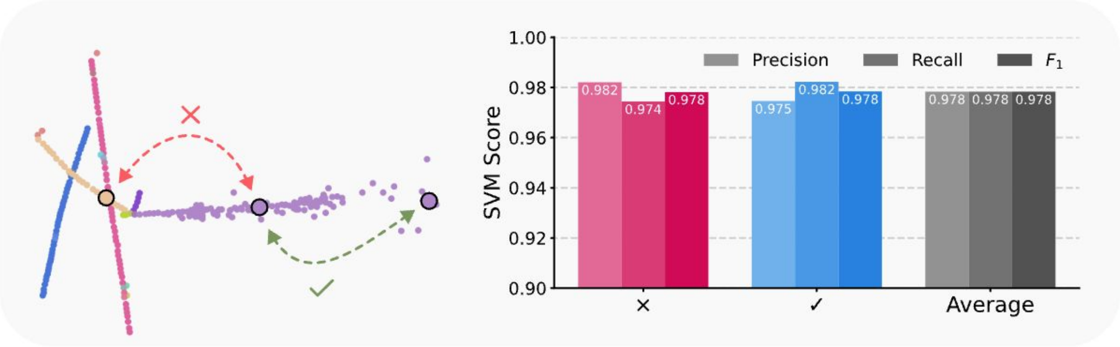
Recall:
common task in LArTPC image reconstruction is <u>instance segmentation</u>



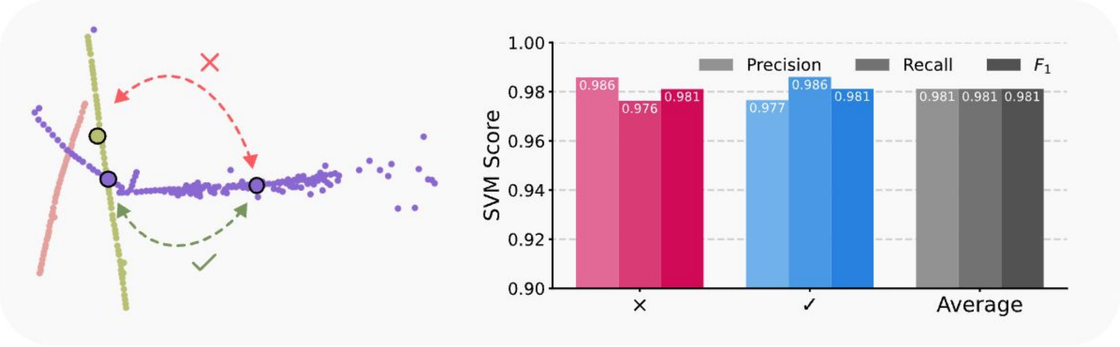
Interactive visualization here

Instance and Vertex Classification

Instance Sharing



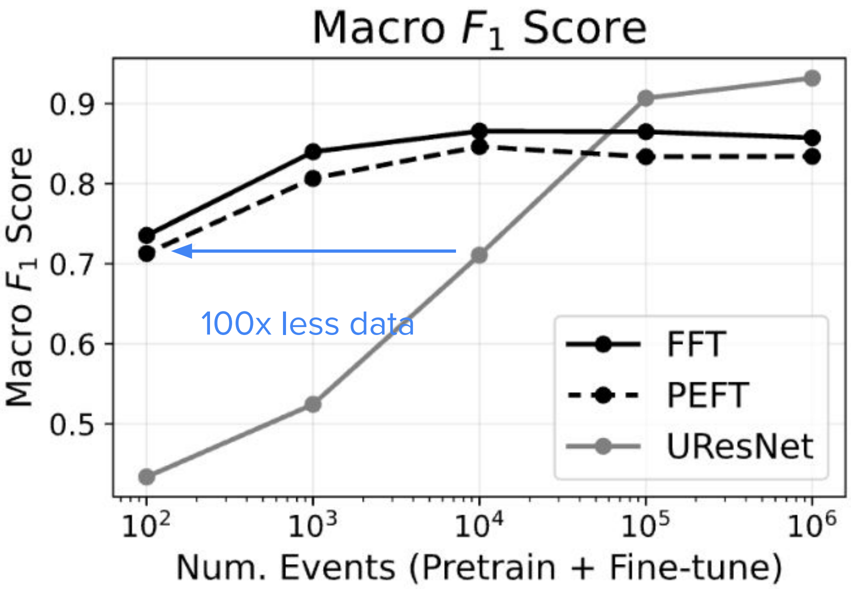
Vertex Sharing



Comparison to State of the Art (UResNet): Semantic Segmentation

What we care about: per-pixel classification

- Beats state-of-the-art in data-constrained environment, but not in the limit of many events.

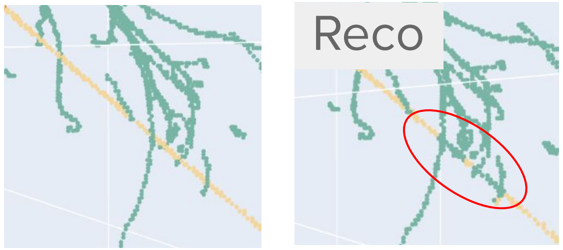


does not beat UResNet at high event counts.

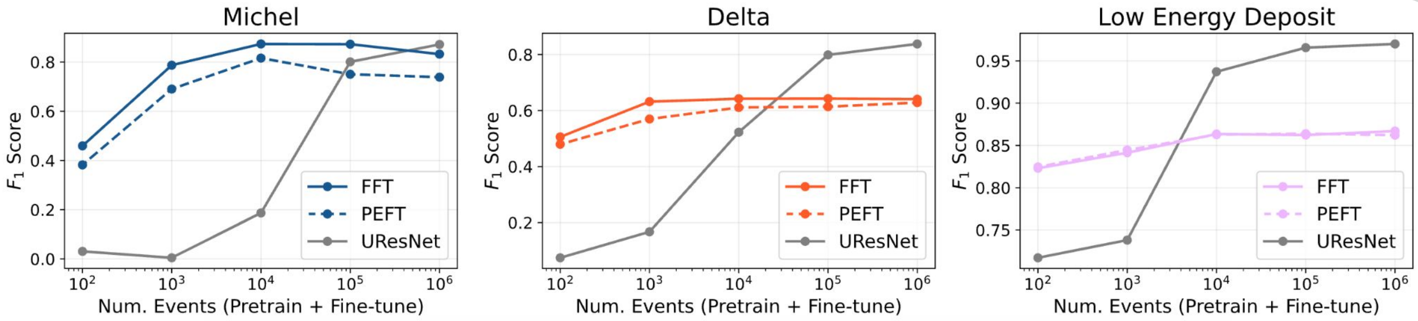
➔ fundamental limit in PoLAr-MAE architecture.

Comparison to State of the Art (UResNet): Semantic Segmentation

- Small features poorly modeled, i.e. “paint brush” classification.
- This is due to **single-scale** patches being used, which smears tiny structures.



“Small” classes



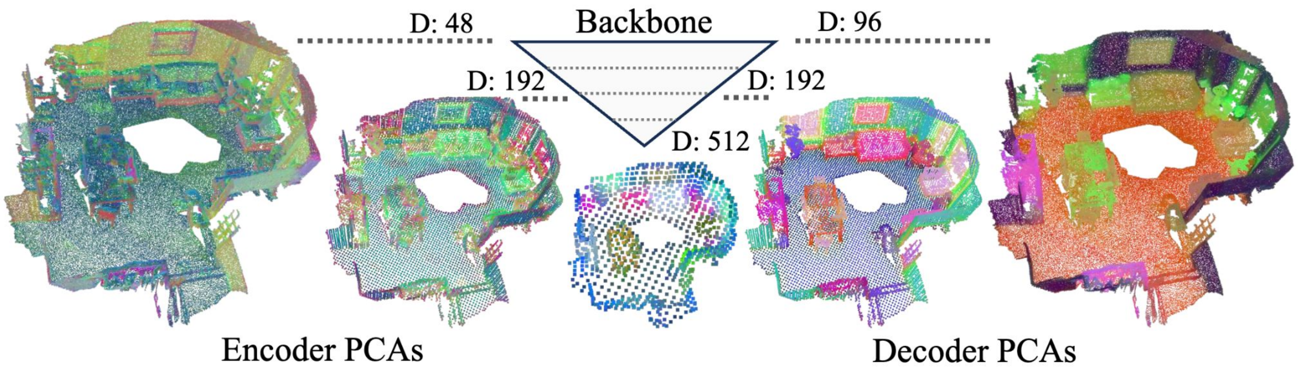
Ok… how can we do better?

Next step: Hierarchical models

New architecture: Point Transformer [1] – hierarchical features with efficient transformer implementation.

- Many fancy tricks to keep efficient and scalable… but will not go over.

Native per-point features are possible → fine-grained understanding



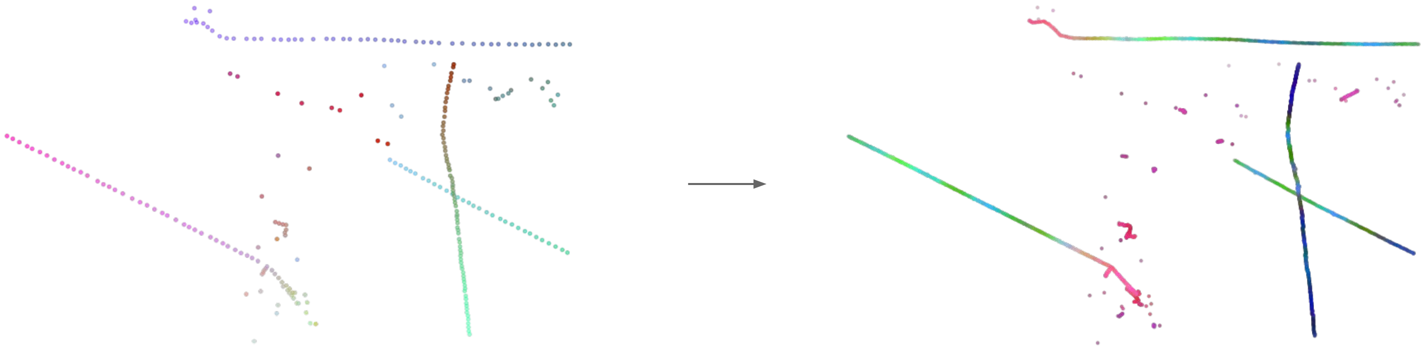
From [2]

Next step: Hierarchical models

New architecture: Point Transformer [1] – hierarchical features with efficient transformer implementation.

- Many fancy tricks to keep efficient and scalable… but will not go over.

Native per-point features are possible → fine-grained understanding



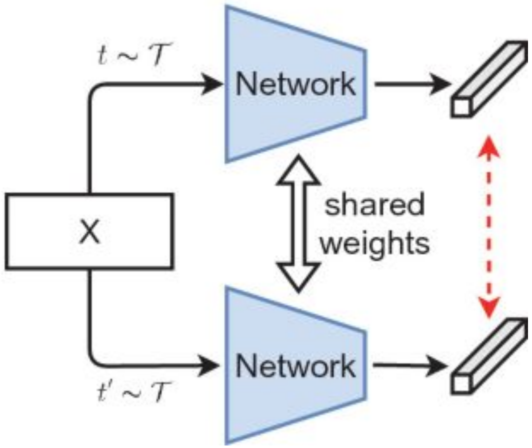
Single-scale **Multi-scale**

Next step: Hierarchical models + **self-distillation** = **SONATA** [1]

In the computer vision world, self-**di**stillation with **no** labels (**DINO**) [2] is changing the way research is being done.

If you use natural images, your feature extractor should probably be DINO.

Self-distillation [2,3] consists of forcing a model to agree across different augmented (jitter, crop, rotate) views of the same image.

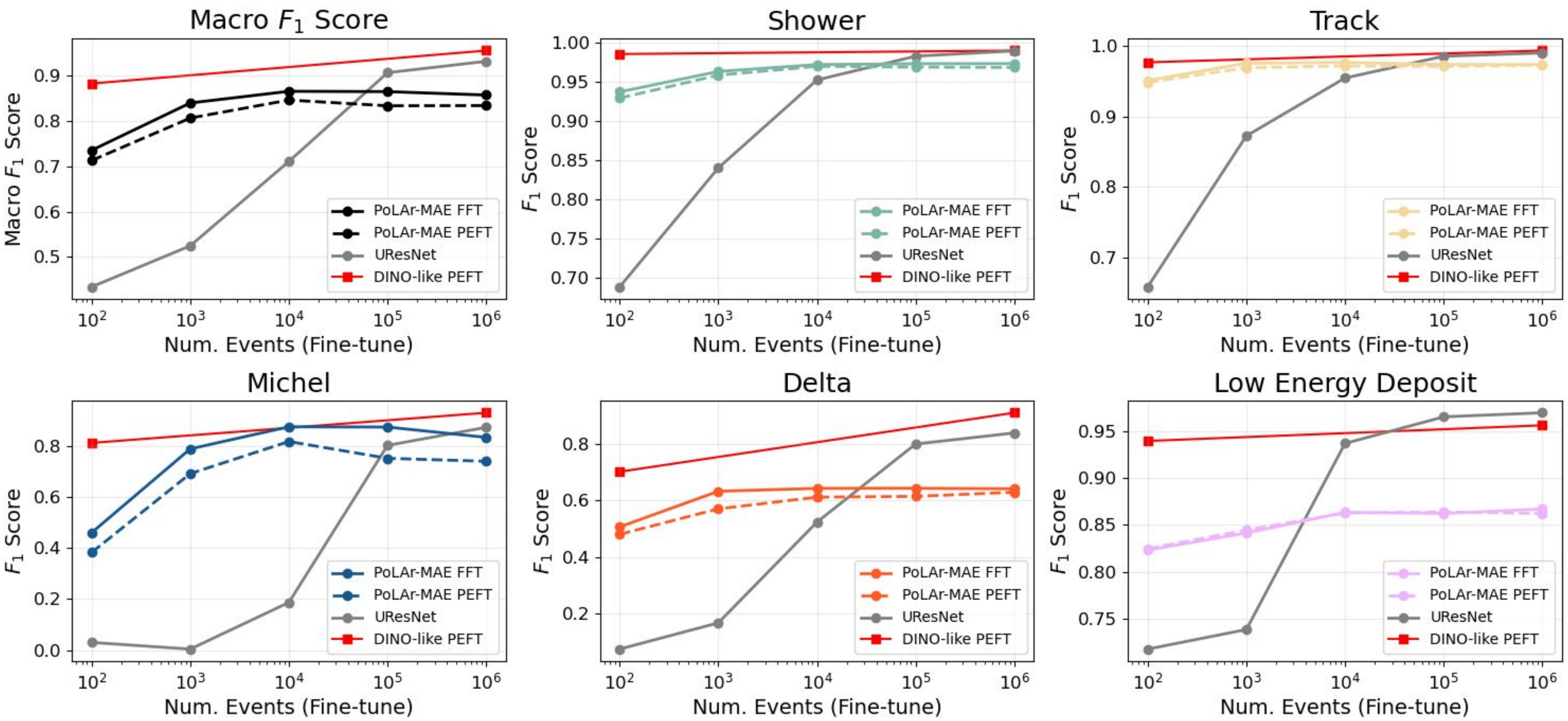


Taken from [4]

So how do we do?

First look at results: $[100, 10^6]$ events

■ DINO-like SSL, PEFT



Takeaways

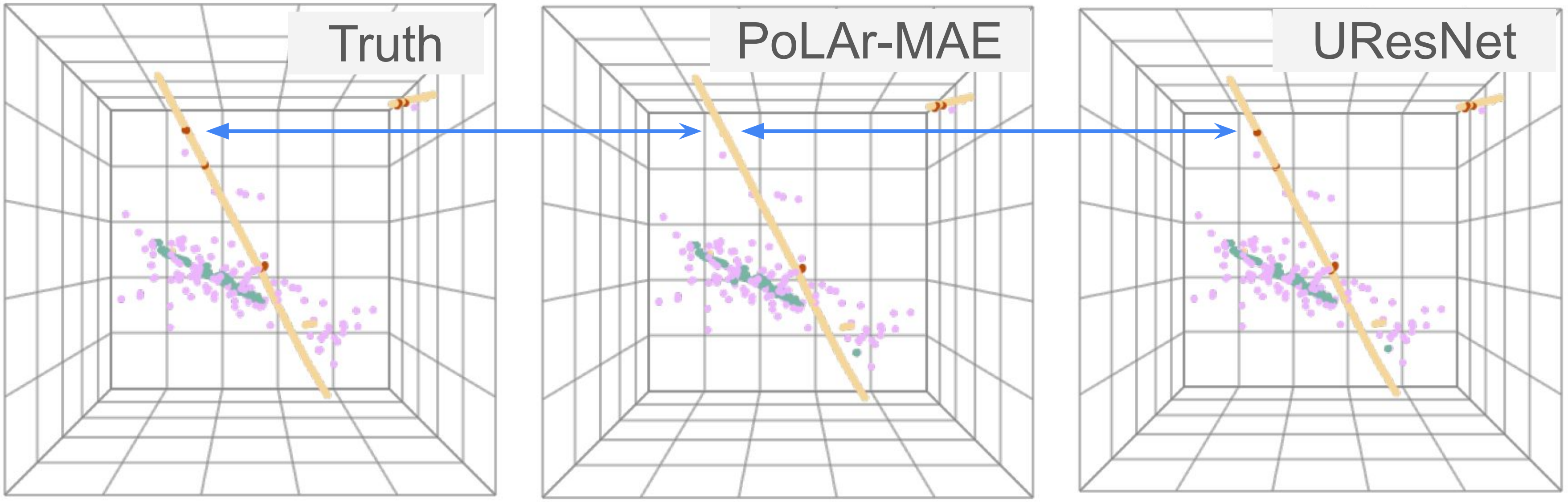
- Hopefully we understand a bit more about how self-supervised training works, and how you might attempt a foundation model for LArTPC images. But there are many other ways! (See this slide in Kazu's talk)
- A generic feature extractor unlocks new possibilities that were simply not possible before:
    - **Few-shot learning w/o well-calibrated sim**: track/shower, Michel tagging, particle ID, ...
    - **Reasoning over images/captioning** with language (human-in-the-loop)
    - **Content-retrieval at scale**: "find events like this" in this dataset.
    - **Cross-experiment datasets** ➔ invariant embeddings across detector conditions, easy adaptation.
    - **Anomaly detection**: flag data as detector conditions degrade.
    - Faster prototyping, quicker progress.

The future is exciting!

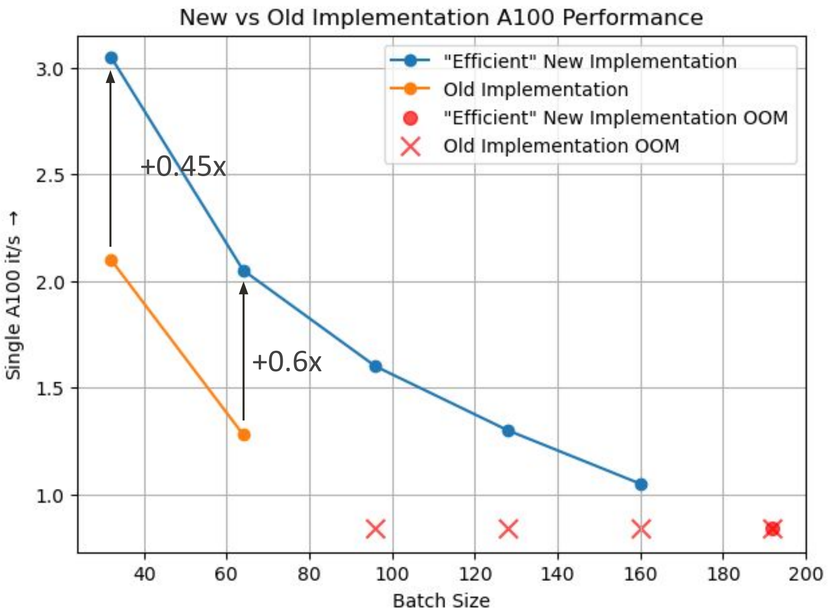
Extras

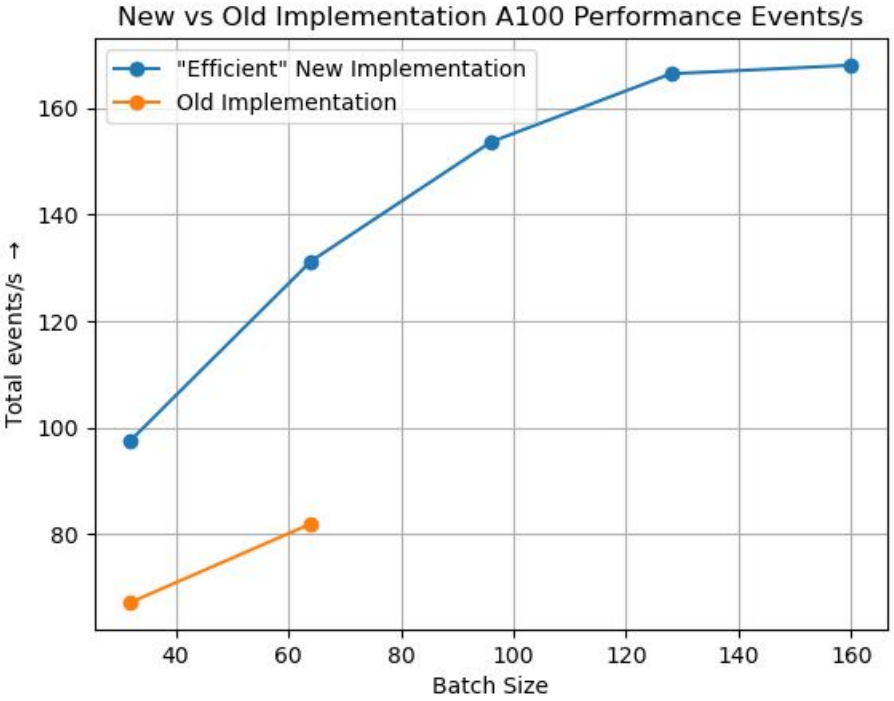
Semantic Segmentation Example

1M dataset, PoLAr-MAE FFT



Perf


New vs Old Implementation A100 Performance
"Efficient" New Implementation
Old Implementation
"Efficient" New Implementation OOM
Old Implementation OOM
+0.45x
+0.6x
Single A100 it/s →
Batch Size
3.0
2.5
2.0
1.5
1.0
40
60
80
100
120
140
160
180
200


New vs Old Implementation A100 Performance Events/s
"Efficient" New Implementation
Old Implementation
Total events/s →
Batch Size
160
140
120
100
80
40
60
80
100
120
140
160