老师您好，我刚刚读了一篇很有启发的文章，题为《Frozen CLIP Transformer Is an Efficient Point Cloud Encoder》，发表在 AAAI-24 上。本文提出了一种新颖的方式：直接利用冻结的 CLIP transformer 编码点云特征，适应多种 3D 任务。以下是它的核心步骤：

1. 首先，对点云进行预处理，使用 FPS（最远点采样）和 KNN（最近邻算法）提取局部邻域，再通过 MLP（多层感知机）对每个邻域编码，形成多个 patch。然后将这些 patch 与一个随机初始化的任务 token 一起传入 CLIP 的 transformer 编码器。
2. CLIP transformer 将这些 patch 和任务 token 一起编码，生成特征表示。
3. 根据任务需求，设计不同的任务头来生成特定任务的输出。
4. 在训练过程中，通过反向传播根据任务的复杂性或模型的收敛速度调整任务头和任务 token，使模型逐步适应不同的 3D 任务。

本文的创新之处在于，通过仅添加任务头和任务 token，便能使冻结的 CLIP 模型适应 3D 点云的多种任务需求。这让我想起了台湾大学李宏毅教授的一个观点，他曾提到，数学界对深度学习模型的批评在于，模型的参数作用难以解释，通常仅依靠大量训练就能获得显著效果。这篇文章似乎印证了这一点，因为 transformer 中每个 token 在计算过程中会通过 self-attention 和 cross-attention 与其他 patch 进行交互，而这些 patch 本身已经包含点云的空间结构和语义信息。所以，在更新任务 token 时，可以理解为 CLIP 模型通过反向传播来逐渐“理解”这些点云 patch 的特征。

如果这个理解是正确的，那么我们可以考虑使用其他点云特征提取模型（例如 PointNet、DGCNN、ReCON 等）替代原本的 patch 生成方式。然后，通过任务 token 的更新帮助 CLIP transformer 逐渐适应不同的点云特征，从而更灵活地适应各种 3D 任务。

Dear Professor,

I just read an inspiring paper titled *"Frozen CLIP Transformer Is an Efficient Point Cloud Encoder,"* published in AAAI-24. This paper introduces an innovative approach: directly using a frozen CLIP transformer to encode point cloud features, making it adaptable for various 3D tasks. Here’s a summary of the core steps:

1. First, the point cloud undergoes preprocessing using FPS (Farthest Point Sampling) and KNN (K-Nearest Neighbors) to extract local neighborhoods. Then, each neighborhood is encoded by an MLP (Multi-Layer Perceptron) to form multiple patches. These patches are combined with a randomly initialized task token and then passed into the CLIP transformer encoder.
2. The CLIP transformer encodes these patches along with the task token, generating a feature representation.
3. Different task heads are designed to generate task-specific outputs based on requirements.
4. During training, the task head and task token are adjusted through backpropagation according to the task complexity or model convergence rate, gradually adapting the model to various 3D tasks.

The key innovation here is that by merely adding a task head and task token, the frozen CLIP model can adapt to multiple 3D point cloud tasks. This reminds me of a point Professor Hung-yi Lee from National Taiwan University made: he mentioned that one critique mathematics has of deep learning models is that the role of parameters is difficult to interpret, and often impressive results are achieved simply through extensive training. This paper seems to illustrate that, as each token in the transformer interacts with others through self-attention and cross-attention, and these patches inherently contain the spatial and semantic information of the point cloud. Therefore, updating the task token could be seen as enabling the CLIP model to gradually “understand” the features of these point cloud patches through backpropagation.

If this understanding is correct, we could consider using other point cloud feature extraction models (such as PointNet, DGCNN, ReCON, etc.) to replace the original patch generation method. Then, by updating the task token, we could help the CLIP transformer gradually adapt to different point cloud features, making it more flexible for a variety of 3D tasks.

I apologize for reaching out at such a late hour, and thank you for your guidance and support.