Report 1

一、VLM

1. 基本流程

VLM 通过将图像和文本信息映射到同一个多模态联合嵌入空间来实现“理解”，我了解了 Llava 这一常用的框架，大致流程如下：

- 图像X_v, 先使用视觉编码器 $g(X_v)$ 将其编码为 Z_v，再通过线性层 W 转换成 H_v。
- 对于指令 X_q，使用 embedding 层将其转换为 H_q。
- 拼接 H_v 和 H_q，将其输入给语言模型 $f_\Phi(.)$ 并得到输出 X_a。

即：

$$X_a = f(\text{concat}([H_v, H_q])) = f(\text{concat}([WZ_v, H_q])) = f(\text{concat}([Wg(X_v), H_q]))$$

图示如下。

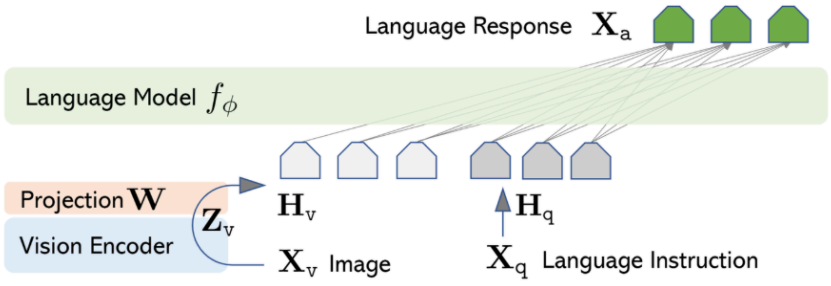

图 1　Llava

2. 微调

- 阶段一：冻结 LLM 权重和视觉编码器，只微调线性层 W，目的是对齐视觉特征 H_v 和 H_q。(让语言模型知道图片的意义)
- 阶段二：只冻结视觉编码器，微调 LLM 权重和线性层 W。(让 LLM 给出正确的回答)

3. Others

对于我们的面向电路的LLM，或许也可以将电路转化成某种H_q, 和自然语言拼接得到输出X_q。（但这么做也有问题，因为电路没法用自然语言精确描述）

二、Clip：图文对齐

1. 基本流程

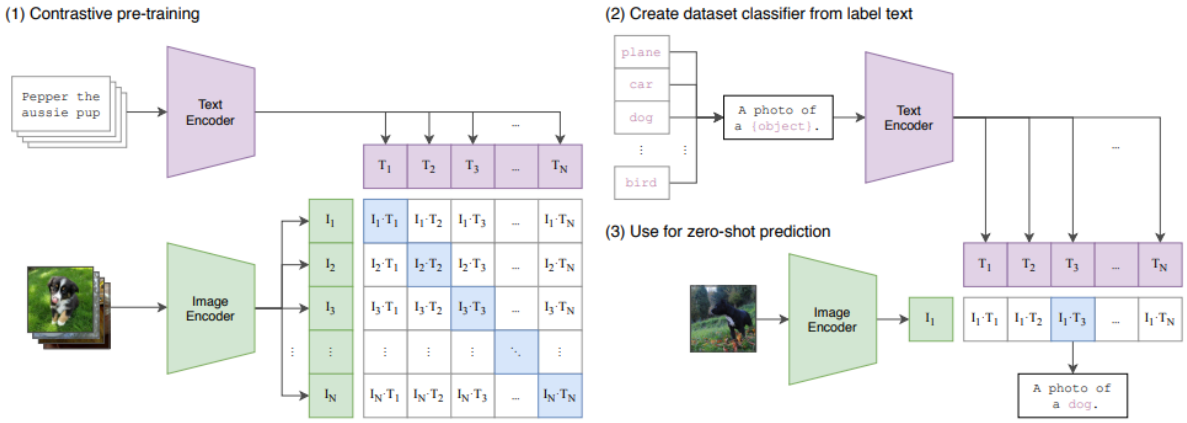

图 2　Clip

- 模型输入：图片、文字配对
  - 图片 → 编码器 → 特征
  - 文本 → 编码器 → 特征
- Clip 在这些特征上对比学习
  - 正样本、负样本的规定：图中矩阵里，对角线上都是正样本

2. Zero shot 的方法： Prompt Engineering

Clip 使用的办法是将 1000 个词嵌入特定的 Template 中形成特定的句子，再进入 Text Encoder。

由此分类头可以不局限于原先训练的 1000 类，可以进行更多样的调整。例如：图中有三轮车，那么可以在前面的单词库里加入三轮车，嵌入 template，则如果图里有三轮车，就可以识别出来。

Clip's Advantages

连续了视觉和句子的语义，可迁移性非常强，彻底摆脱了预训练集对于 categorical label 的约束。

3. 关于 Prompt Engineering

如果要训练电路领域的 LLM 的话，感觉对于 LLM，Q&A 对也可以这么训练。比如（以下问题来自 Gemini 生成）：

- 元件识别型 (Component Identification):

  Q: "Locate the component U1 in the image."

  A: 给出 U1 的坐标 bounding box

- 参数查询型 (Parameter Query):

Q: “What is the resistance of R5?”

A: “The resistance of R5 is 10k ohms.”

- 连接追溯型 (Connectivity Tracing):

  Q: “What is pin 3 of the operational amplifier U1 connected to?”

  A: “Pin 3 of U1 is connected to the junction of resistor R2 and capacitor C1.”

- 功能理解型 (Functional Understanding):

  Q: “What is the function of the components R1 and C1?” (这需要更高级的模板)

  A: “R1 and C1 form a low-pass filter.”

具体怎么分割感觉需要进一步讨论，借鉴例如 《AI2D: A Dataset for Continuous Diagrammatic Reasoning》.etc

4. 关于 Clip 实现电路 Q&A 的局限性：

1. 最大问题在于 Clip 是用**语义训练**的，没法回答电路这种推理/精确的问题
2. 电路领域的 Q&A 需要“精确细节”的逻辑推理（例如网表结构，netlist，元器件间是有联系的，并非靠整体感觉），Clip 完成不了。
3. 实际实验结果表明 GPT-4 还有 Clip 的效果不如基于网表的大模型（Netlistfy）.

三、电路图 - 网表/Graph 对齐

1. 网表 | Netlist

- **直接用 Netlist 构建数据集/训练：**不可行。Netlist 实际上是节点与边的连接，所以大模型搜索的时候会很费力。
- **用 Netlist 表示电路图：**可行，例如 Spice 这样的 EDA 工具可以基于 Netlist 生成电路图，因此可以用脚本先将我们生成的拓扑图转化成 Netlist, 再转化成电路图。

2. Graph

- **用 Graph 训练：**可行，可以使用 GNN。而且在实际电路中若干个较小的电路单位可以组合成一个相对较大的单位，和图里的子图等关系比较类似。
- **将实际电路图转换成 Graph：**可行。这篇论文用了以下步骤：
  - Component Detection (元件检测):用 YOLOv8 模型把图上所有的元件都框出来。—— 这就是在构建 Graph 的“节点(Nodes)”。
  - Connectivity Analysis (连接性分析): 用一个 Transformer 模型来检测所有的连接线段。—— 这就是在构建 Graph 的“边(Edges)”。
  - Connectivity Extraction (连接性提取): 将前面得到的所有节点和边组合起来
  - 可以用上细粒度对齐的思路，还没细想

- **输出**：可以用算法将 Graph 转化成 Netlist。

四、细粒度对齐

这部分我刚刚看懂 CktGNN 还有 Circuit Fusion，其他的文章都还没看。

1. CktGNN

> **Note**
> CktGNN 的典型应用场景是对所给的功能生成对应的模拟电路图，和输入电路图进行功能分析不太一样。

1. 人类专家预先定义了一个包含各种基础模块的“库” **Subgraph Basis**
2. 面向 OpenBCD:
   - 内部 GNN (Inner GNN)：分别理解每一个 Subgraph Basis 的特性
   - 外部 GNN (Outer GNN)：分析这些 Subgraph Basis 是如何组合在一起的，学习它们之间的连接关系。
3. 面向 prompt，利用 Optimizer 找到最好的电路组合，组合成向量，重建电路

（细节还没细看，因为之前不太懂 GNN，可能还得补补课）

2. Circuit Fusion

粒度：对于每一个寄存器，模型会向上回溯，找到在一个时钟周期内决定该寄存器下一个值的所有组合逻辑。这个“以寄存器为终点的逻辑锥 (Logic Cone)”就被定义为一个子电路 (Sub-circuit)

对齐思路

使用对比学习进行训练：

1. 模态内对齐：语义对齐

取一个子电路的结构图 G。然后用一个逻辑综合工具（如 Yosys）对它进行微小的、功能等价的变换，得到一个新的结构图 G’。G 和 G’ 在视觉和结构上可能完全不同，但它们的功能是 100% 相同的。G 和 G’ 就构成了一对**正样本**，其余功能不同的是负样本。

2. 跨模态对齐 - “多视角对齐”

让模型理解：对于同一个子电路，它的代码、图、和摘要这三种描述方式，指向的是同一个东西

3. 跨阶段对齐 - “实现感知对齐”

让模型理解，一个在设计早期（RTL 阶段）的子电路，与它在设计后期（物理实现阶段）的**最终形态（门级网表）**之间的关系。

经过这一步，模型学会了从高层次的 RTL 描述中预见其最终的**物理实现特性**。

3. 使用细粒度对齐的必要性

1. 电路图表现形式多：不同阶段的电路图（原理图、PCB | RTL 阶段、综合后）可能指向最终一样的物理特性
2. 电路功能分析需要严谨地推导，所以需要区分各个元件。

五、总体流程

1. 电路图功能分析

- image 形式：
  - 转化成 Graph
  - 多模态训练进行功能分析 → 特征向量
- 其他形式：
  - ...
- 细粒度对齐

2. LLM 部分

利用前面的特征向量，prompt Engineering

3. 输出

- 生成文字
- 生成电路图：CktGNN 类似思路，由小子图生成电路图

4. BenchMark

- OpenBCD
- 略

画了一张导图，时间有限，只能简单手绘了，很多地方还没搞清楚..

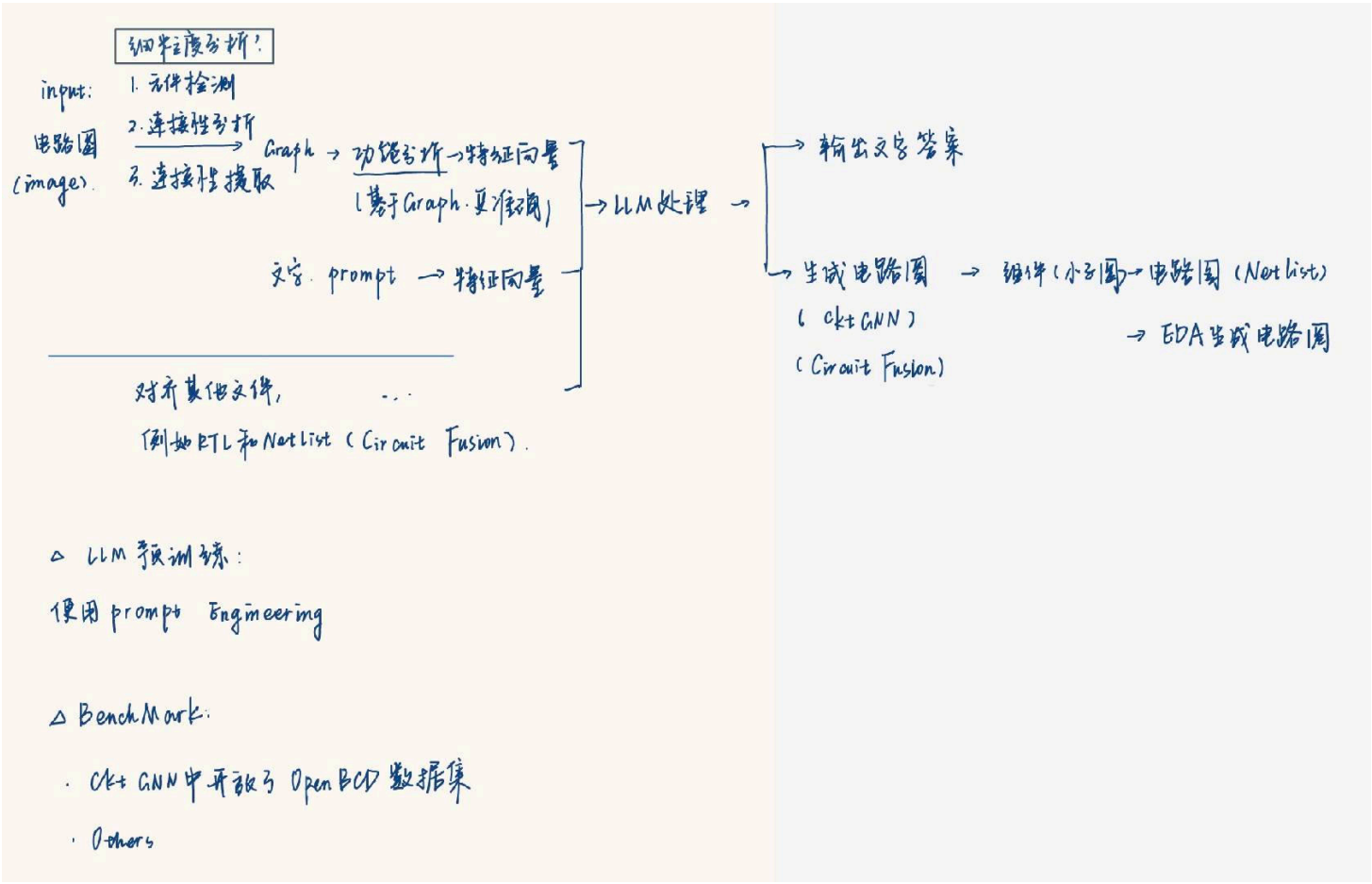
细粒度分析:
input:
电路图
(image)
1. 元件检测
2. 连接性分析
3. 连接性提取
Graph → 功能分析 → 特征向量
(基于Graph. 算法)
文字. prompt → 特征向量
→ LLM处理 →
→ 输出文字答案
→ 生成电路图 → 组件(小子图) → 电路图 (Net list)
(ckt GNN)
(Circuit Fusion)
→ EDA生成电路图
对齐其他文件, ...
例如RTL和Net list (Circuit Fusion).
△ LLM预训练:
使用 prompt Engineering
△ BenchMark:
· ckt GNN中开放了 OpenBCD 数据集
· Others

图 3　流程图