Final Year Project Term1 Midterm Report:

# 1. Review of papers and some other related knowledge

## 1.1. Research Background and Motivation

### 1.1.1. Problem Context

Contemporary visual understanding systems excel under closed-set assumptions, where category vocabularies are fixed and fully supervised. However, practical applications increasingly demand open-vocabulary and long-tail recognition, where models must generalize to previously unseen concepts and adapt to evolving taxonomies without exhaustive relabeling. Simultaneously, edge and near-real-time deployment scenarios impose stringent constraints on latency, memory, and energy consumption, creating a dual requirement: broad semantic coverage with efficient inference.

### 1.1.2. Opportunity from Vision–Language Pretraining

Vision–language models such as CLIP demonstrate that contrastive pretraining on large-scale image–text pairs can produce a shared embedding space where images and natural language descriptions align semantically. This enables zero-shot classification, text-to-image retrieval, and phrase grounding by replacing closed-set classifiers with cosine similarity against textual prompts. The success of prompt ensembling and temperature scaling further indicates that language can serve as a flexible, compositional index into visual semantics.

### 1.1.3. Gaps in Current Practice

Despite their semantic generality, state-of-the-art vision–language models are computationally heavy, with significant memory footprints and latency that hinder edge deployment. Conversely, lightweight detectors (e.g., YOLO families) meet real-time constraints but remain closed-set and lack robust language grounding. Bridging this gap requires transferring or distilling the semantic alignment learned by large vision–language models into efficient, task-oriented architectures without sacrificing throughput.

### 1.1.4. Project Positioning

This project targets an efficient open-vocabulary pipeline by: (i) leveraging a frozen text encoder from a pretrained vision–language model to provide a stable semantic reference; (ii) employing a lightweight visual backbone to extract multi-scale image features under strict compute budgets; and (iii) introducing distillation and text-conditioned modulation to align visual features with the text embedding space. The resulting model aims to support zero-shot classification/retrieval and, subsequently, open-vocabulary object detection with practical inference speed.

## 1.2. Fundamental Theory, Key Techniques, and Representative Related Work

### 1.2.1. Cross-Modal Alignment via Contrastive Learning

Core idea. Vision–language pretraining aligns images and texts in a shared embedding space using contrastive objectives. Given batches of

image–text pairs, the model maximizes similarity for matched pairs while minimizing it for mismatches, typically with an InfoNCE/CLIP loss, L2-normalized features, and a learnable temperature to control logit scaling.

Encoders. Visual encoders are commonly ResNets or Vision Transformers (ViTs); text encoders are Transformer-based. Pretraining scale and data diversity directly affect coverage of long-tail concepts and compositional generalization.

Prompting and calibration. Prompt ensembling (multiple templates per Representative work. CLIP (Radford et al., 2021) established the modern baseline for zero-shot classification and retrieval. ALIGN (Jia et al., 2021) scaled noisy web data. LiT (Zhai et al., 2022) “locks” the image encoder and tunes the text side to improve transfer.

### 1.2.2. YOLO Family: Fundamentals, Techniques, and Evolution

Overview and principle. YOLO (You Only Look Once) formulates object detection as direct dense prediction from feature maps, removing the region proposal stage to achieve real-time performance. A single forward pass outputs bounding boxes and class confidences at multiple scales.

Canonical architecture:

Backbone: Efficient feature extraction (e.g., CSPDarknet, ELAN/RepVGG-style blocks, or lightweight MobileNet/EfficientNet variants in custom deployments).
Neck: Multi-scale feature aggregation via FPN/PAN to enhance small/large object localization.
Head: Decoupled classification and box regression. The family evolved from anchor-based (YOLOv3–v5) to anchor-free heads (YOLOv8), simplifying optimization and improving stability.

### 1.2.3 Knowledge Distillation and Semantic Alignment Strategies

Objectives. Transfer the semantic richness of large VLMs into compact, fast models while preserving alignment with the text space.
Distillation types.

Representation regression: Match student visual embeddings to teacher image embeddings via L2 or cosine regression with feature normalization.

Contrastive distillation: Place the student in a contrastive setup using the teacher as a target or teacher-generated pseudo-pairs; improves discriminability.

Multi-level/multi-scale alignment: Align features at several depths and scales to stabilize training and benefit detection heads.Text conditioning.

### 1.2.4 Method Comparison and Key Differences

We compare four paradigms along supervision, architecture,

efficiency, and deployability: (1) CLIP-like zero-shot classifiers, (2) region-level grounded pretraining (e.g., GLIP/RegionCLIP), (3) open-vocabulary ViT detectors (e.g., OWL-ViT), and (4) lightweight YOLO-based OVD with text alignment (ours). CLIP-style models offer broad semantic coverage and strong zero-shot classification/retrieval but lack explicit region localization and are compute-heavy at high resolutions. Grounded pretraining aligns region–text pairs at scale, yielding robust localization and transfer, yet the training pipelines are complex and teacher-dependent. Open-vocabulary ViT detectors provide clean end-to-end text–region similarity heads and strong accuracy on unseen categories, but their transformer backbones incur higher latency and memory, limiting edge deployment. In contrast, our lightweight YOLO-based design replaces fixed softmax heads with normalized feature projections and cosine similarity to cached text embeddings, adds multi-scale alignment/distillation for region-level semantics, and uses FiLM-based conditioning for query-aware inference. This yields a favourable speed–accuracy trade-off: competitive open-vocabulary detection with substantially lower latency, smaller memory footprint, and streamlined export (ONNX/TensorRT), while preserving zero-shot flexibility.

2. Specifications: Environment Setup and Usage
   - 2.1. Preparation for the Model's Coding
     - 2.1.1. Basic programming environments

       Linux, Windows, MacOS all should support our programming. C++/C, Python, Java and other programming language packages should be downloaded in advance.
     - 2.1.2. Pytorch and other required packages

       First, according to our plan, users should install PyTorch and torchvision, as well as small additional dependencies. Plus, it needs a CUDA GPU machine. A dependency list(requirements.txt) will be created when coding. The following then will do the trick: pip install -r requirements.txt
   - 2.2. Usage of the Model
     - 2.2.1. API

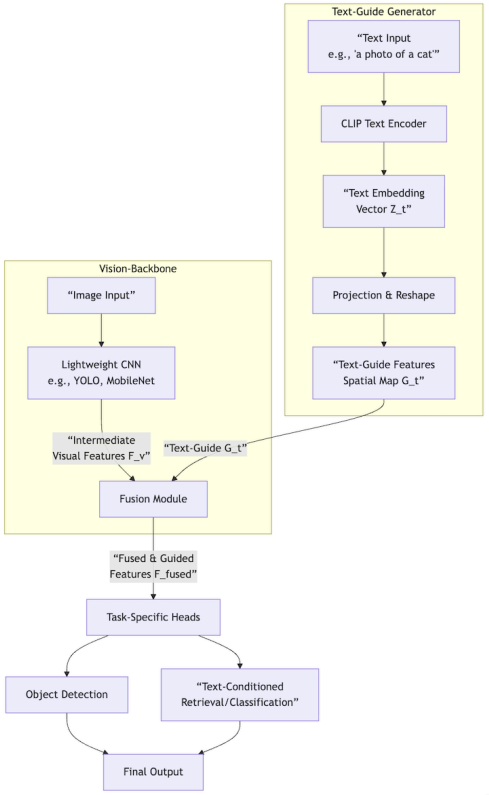
       In final report, the model module should provide some APIs like available_models() for listing all available ViT and other models, load() for loading model checkpoint files, encode_image() and encode_text() for returning the image/text features encoded by the model, model() for returning the cosine similarity between text and image features, and other useful methods.
     - 2.2.2. Datasets and Demonstrations

       In the later report, it will offer a list that shows the model's performance (accuracy, recall, and reference speed) over a wide range of datasets, corresponding datasets and instructions for verifications and demonstrations.
     - 2.2.3. Tasks

       This model will be able to do some cross-modal tasks at a low cost, including text-image retrieval, zero-shot image classification, and even open-vocabulary object detection.
3. Model Architecture and Programming Implementations
   - 3.1. A simple model architecture diagram

     We propose a dual-stream, lightweight vision-language model that

leverages a frozen CLIP model as a semantic teacher and a lightweight YOLO-based backbone as the student. The model is designed to be end-to-end trainable and supports multiple open-vocabulary tasks without requiring the full CLIP model during inference.




3.2. Architecture:

3.2.1. Vision Backbone

The input image is processed not by the heavy CLIP image encoder, but by a lightweight vision model. A prime candidate is YOLO (e.g., YOLOv5s, YOLOv8n) due to its excellent speed-accuracy trade-off and inherent capability for object-level feature extraction. We will use its backbone and neck for feature extraction, but replace its detection head with a custom projection module.

3.2.2. Text Backbone

The input text prompt (e.g., "a photo of a dog") is processed by the frozen CLIP text encoder (a Vision Transformer, ViT, is typically used in CLIP's *image* encoder; the text encoder is a transformer-based model). Its output is a semantic text embedding.

The detection head is replaced with a custom projection module (e.g., a 2-layer MLP) to map image features into the same embedding space as CLIP.

3.2.3. Fusion Module

Implements Feature-wise Linear Modulation (FiLM) or a cross-attention mechanism to fuse text embeddings with image features.

Text embeddings are used to modulate the intermediate feature

maps of the YOLO backbone, enabling text-guided visual processing.
Therefore, the fusion module will consider both distillation loss, and text-image feature alignment. Plus, it will calculate errors between predictions and data for regularization and model fine-tuning. Further research and experiments will be conducted.

### 3.3. Loss Functions

Distillation Loss: Cosine Embedding Loss or L2 Loss between the projected image embeddings (from YOLO) and the image embeddings from the frozen CLIP image encoder.
Cross-Modal Alignment Loss: Contrastive loss (e.g., InfoNCE) to ensure image and text embeddings are semantically aligned.

### 3.4. Implementations Strategy

In order to increase work efficiency, it is important to use techniques like de-coupling - reduce interdependence among components.

### 3.5. Scripts for specific tasks

Several .py files will be provided for specific tasks mentioned before, including cross_modal_retrieval.py, zero_shot_classfication.py, open_val_obj_detection.py.

Zero-Shot Classification:

Head: Cosine similarity between image embedding and text class embeddings.

No additional loss; uses similarity scoring.

Text-Image Retrieval:
Same as above, but applied to a database of images and text queries.

Open-Vocabulary Object Detection (OVOD):
Head: Lightweight detection head that outputs region proposals and region-specific embeddings.
Loss: Combination of localization loss (e.g., GIoU) and embedding alignment loss (cosine similarity between region embeddings and text embeddings of class names).

References

[1] Radford, A., et al. Learning Transferable Visual Models From Natural Language Supervision (CLIP). ICML, 2021.

[2] Jia, C., et al. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision (ALIGN). ICML, 2021.

[3] Zhai, X., et al. LiT: Zero-Shot Transfer with Locked-image Tuning. CVPR, 2022.

[4] Redmon, J., Farhadi, A. YOLOv3: An Incremental Improvement. arXiv:1804.02767, 2018.

[5] Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of

Object Detection. CVPR, 2020.

[6] Hinton, G., Vinyals, O., Dean, J. Distilling the Knowledge in a Neural Network. arXiv:1503.02531, 2015.

[7] Tian, Y., Krishnan, D., Isola, P. Contrastive Representation Distillation. ICLR, 2020.

[8] Li, J., et al. GLIP: Grounded Language-Image Pre-training. CVPR, 2022

[9] Minderer, M., et al. Simple Open-Vocabulary Object Detection with Vision Transformers (OWL-ViT). ECCV, 2022; TPAMI, 2023.