

# FYP Planning Report

## Title: Leverage CLIP to Impart the Capability to Understand Text Features to Lightweight Vision Models

### **Project Goal:**

The primary goal of this Final Year Project (FYP) is to develop and evaluate a method for transferring the robust, cross-modal (image-text) understanding capabilities of a large pre-trained model, CLIP (Contrastive Language-Image Pre-training), to a lightweight, efficient vision model (like YOLO). The project aims to enable these small models to understand and respond to semantic text queries for tasks like zero-shot image classification, image-text or text-image retrieval, and even open-vocabulary object detections, without the computational cost associated with running the full CLIP model.

### **Significance:**

What does this project aim to achieve:

This project aims to develop a lightweight, efficient vision model (e.g., YOLO-based) that inherits CLIP's state-of-the-art cross-modal (image-text) understanding—enabling it to perform zero-shot image classification, text-image retrieval, and open-vocabulary object detection—without relying on CLIP's prohibitive computational costs.

Who will benefit from this work:

Edge Device Developers and Manufacturers will benefit. Smart cameras, mobile phones, and IoT devices often rely on lightweight vision models for real-time processing. Our method equips these devices with text-understanding capabilities without requiring high-end hardware, unlocking applications like on-device "visual search by text".

Why is this project important:

As AI shifts toward edge deployment, efficient multi-modal models are critical. By making text-understanding accessible to lightweight vision models, we democratize advanced AI features for resource-constrained environments, bridging the gap between high-performance servers and everyday devices.

### **Problem statement:**

The rapid advancement of large pre-trained models like CLIP has demonstrated unprecedented capabilities in cross-modal (image-text) understanding, enabling zero-shot image classification, image-text retrieval, and semantic visual reasoning with minimal task-specific training. However, CLIP's computational demands—stemming from its massive architecture (e.g., ViT-B/16 backbone) and extensive pre-training—render it impractical for deployment on edge devices (smart cameras, mobile phones, IoT systems), where resource constraints (memory, latency, power) mandate lightweight solutions. Lightweight vision

models (e.g., YOLO series, MobileNet) excel in efficiency, processing images in milliseconds with minimal hardware requirements.

## Proposed Solution:

The core of this project is to design a novel, end-to-end trainable architecture that fuses text-guided semantics from CLIP with the efficient detection capabilities of a lightweight vision model like YOLO to create a low-cost, lightweight model capable of multiple open-vocabulary tasks. The key innovation is to use a *frozen*, pre-trained CLIP model as a "semantic teacher" to process text and images image features, and then distill this semantic understanding into a *single, efficient* model that can perform various tasks without the need to run the full CLIP model during inference.

## Proposed Methodology:

The solution involves a two-stage pipeline: Feature Alignment Pre-training followed by Task-Specific Fine-Tuning.

### Stage 1: Building a Unified, Lightweight Vision-Language Backbone

This is the core of the project, fulfilling the goal of a "low-cost lightweight model."

#### 1. Architecture Design:

- Text Pathway: 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.
- Image Pathway: 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.

#### 2. Knowledge Distillation and Fusion:

- The goal is to "impart" the text understanding to the visual model. We achieve this by aligning the features from the lightweight image encoder with the semantic space defined by CLIP.
- The image features from the YOLO backbone are projected into a shared embedding space via a small, trainable projection network.
- A distillation loss (e.g., Cosine Embedding Loss or L2 Loss) is computed between these projected image features and the corresponding image features from the frozen CLIP image encoder for the same input image. This forces the lightweight model to "see" the image in a way that is semantically aligned with CLIP.

- Crucially, we also introduce a cross-modal guidance mechanism. The text embedding from the CLIP text encoder is used to dynamically modulate or "guide" the image features within the YOLO backbone/neck, perhaps through a conditioning mechanism like Feature-wise Linear Modulation (FiLM) or an attention-based fusion module. This teaches the model to focus on visual features relevant to the text query.

## Stage 2: Task-Specific Heads and Fine-Tuning

After the lightweight model's features are aligned with CLIP's semantic space, we attach small, task-specific heads to perform the final objectives.

- Deliverable 1: Zero-Shot Image Classification
  - Head: A simple cosine similarity calculator.
  - Inference: For a set of text prompts (e.g., "a photo of a [class]"), the frozen CLIP text encoder generates their embeddings. The lightweight image encoder processes the image to produce a single, global image embedding. The class is predicted by finding the text embedding with the highest cosine similarity to the image embedding.
- Deliverable 2: Text-Image Retrieval
  - Head: A cosine similarity calculator.
  - Inference: Given a database of images and a text query, the model generates an embedding for each image and the query text. Images are ranked by the similarity between their embeddings and the text query embedding.
- Deliverable 3: Open-Vocabulary Object Detection (The Most Ambitious Goal)
  - Head: A modified, lightweight detection head that outputs region proposals. Instead of predicting a fixed set of classes, it outputs a region-specific embedding for each proposed bounding box.
  - Inference: For each proposed region, its embedding is compared via cosine similarity to the embeddings of all target class names (provided as text prompts). The class with the highest similarity score above a certain threshold is assigned to the box. This directly leverages the aligned vision-language space created in Stage 1.

## Deliverables:

- D1: A Novel Model Architecture: The complete PyTorch implementation of the proposed lightweight model, featuring the CLIP-guided YOLO backbone and the projection/fusion modules.
- D2: Pre-trained Weights: A model pre-trained on a large dataset (e.g., COCO, a subset of LAION) using the proposed distillation and feature alignment framework.
- D3: Quantitative Evaluation Suite:

- Zero-Shot Classification: Evaluation on datasets like CIFAR-10/100, ImageNet.
- Image-Text Retrieval: Evaluation on Flickr30k or MS-COCO retrieval tasks.
- Open-Vocabulary Detection: Evaluation on a held-out set of COCO classes or the LVIS dataset.
- Efficiency Benchmarking: Comprehensive reports on model size (parameters), computational complexity (FLOPs), and inference speed (FPS) on both GPU and a resource-constrained device (e.g., Jetson Nano or mobile phone).
- D4: Final Project Report & Dissertation: A comprehensive thesis documenting the architecture, training methodology, results, and analysis, highlighting the trade-off between performance and efficiency compared to large models like CLIP and other lightweight baselines.

## Proposed Timelines (4-Month / 16-Week Schedule)

This condensed timeline is ambitious and requires a focused, iterative approach. The phases are designed to build upon each other, with the core architecture serving as the foundation for all three deliverables.

Phase	Key Activities & Milestones	Week
Phase 1: Foundation & Architecture Design	<ul style="list-style-type: none"> <li>- Literature Deep Dive: Finalize understanding of CLIP, YOLO architecture, and knowledge distillation techniques.</li> <li>- Environment Setup: Establish codebase (PyTorch), install dependencies, and prepare data loaders (e.g., COCO, Flickr30k).</li> <li>- Milestone 1: Finalize and document the high-level model architecture diagram (text guidance mechanism, projection network).</li> <li>-&gt; submit the FYP Term 1 Midterm Report: 1. review of papers and some other</li> </ul>	1-4

	related knowledge, 2. How to setup environment 3. design the model architecture	
Phase 2: Core Model Implementation & Training	<ul style="list-style-type: none"> <li>- Implement Base Model: Code the lightweight YOLO backbone, projection network, and the feature fusion mechanism (e.g., FiLM/Attention).</li> <li>- Implement Loss Functions: Code the distillation loss (L2/Cosine) and any other alignment losses.</li> <li>- Initial Training &amp; Debugging: Run initial training cycles on a small dataset to debug the pipeline and ensure gradients flow.</li> <li>- Milestone 2: Successfully train the core model; the output image embeddings show meaningful cosine similarity with CLIP's text embeddings.</li> </ul> <p>-&gt; FYP Term End Report</p>	5-9
Phase 3: Task-Specific Evaluation & Refinement	<ul style="list-style-type: none"> <li>- Implement &amp; Evaluate Task Heads: Sequentially implement and test the simple heads for Zero-Shot Classification and Image-Text Retrieval.</li> <li>- Benchmark Performance: Run quantitative evaluation on standard benchmarks for these two tasks.</li> <li>- Implement OVOD Head: Develop and integrate the more complex Open-Vocabulary Detection head.</li> <li>- Fine-tune for Detection: Fine-tune the entire model (or just the detection head) on a detection dataset like COCO.</li> <li>- Milestone 3: Obtain initial results for all three</li> </ul>	10-13

	promised tasks (Classification, Retrieval, OVOD).	
Phase 4: Final Benchmarking & Report Finalization	<ul style="list-style-type: none"> <li>- Comprehensive Efficiency Tests: Benchmark final model size (params), speed (FPS on GPU/edge device), and accuracy.</li> <li>- Final Analysis &amp; Comparison: Compare results against pre-defined baselines (full CLIP, vanilla YOLO).</li> <li>- Thesis Writing &amp; Consolidation: Compile all work into the final dissertation. Create presentation slides.</li> <li>- Milestone 4: Complete final project report and all associated code and model weights.</li> </ul>	14-16