Exploring Few-Shot Learning with EasyFSL

1. Introduction

In this report, I document my journey from knowing nothing about few-shot learning to successfully implementing a model using the EasyFSL library. The goal of this project was to develop a pipeline capable of detecting and counting shampoo sachets in images based on a single sample image. The approach leverages few-shot learning, a method that enables models to recognize new classes with very limited training examples.

Throughout this process, I followed a systematic learning path, which I also compiled into a YouTube playlist: Few-Shot Learning Playlist.

2. Understanding Few-Shot Learning

Few-shot learning is a subfield of machine learning focused on enabling models to learn new tasks rapidly with minimal training data. Traditional supervised learning requires extensive datasets for each class to achieve good performance. However, few-shot learning methods, like Prototypical Networks, address this limitation by leveraging a metric-based approach:

- **Support Set:** A small set of labeled examples.
- **Query Set:** A set of unlabeled examples for which labels are to be predicted.
- **Prototypical Networks:** Compute a prototype for each class in the support set as the mean of its embeddings and classify query samples based on their distance to these prototypes.

Key Benefits:

- Efficient learning with limited data.
- Effective for tasks where collecting large datasets is impractical.

3. Discovering EasyFSL

During my research, I discovered EasyFSL, a Python library that simplifies the implementation of few-shot learning methods:

- **Key Features:**
  - Predefined methods like Prototypical Networks.
  - Easy integration with PyTorch datasets and models.
  - Utilities for tasks like prototype computation and task sampling.

By following the official documentation and examples, I quickly grasped how to build a few-shot learning pipeline using this library.

4. The Shampoo Sachet Detection Project

Goal:

Detect and count shampoo sachets in an image based on a single sample image using few-shot learning.

Approach:

1. **Sample Image:**
   - Load and preprocess a single image of the sachet as a reference.
2. **Feature Extraction:**
   - Utilize a pre-trained ResNet18 model as the backbone for extracting embeddings.
   - Compute a prototype for the sachet class using EasyFSL.
3. **Sliding Window Detection:**
   - Scan an overall image using a sliding window technique.
   - Extract features for each window and compute distances to the prototype.
   - Mark windows as matches if the distance is below a threshold.
4. **Result Visualization:**
   - Highlight detected sachets in the image.
   - Display the count and positions of detections.

5. Key Challenges Faced

1. Setting Up EasyFSL:

- **Issue:** Import errors and package compatibility.
- **Solution:** Updated and installed the required packages manually, ensuring EasyFSL and PyTorch compatibility.

2. Device Utilization:

- **Issue:** Limited by CPU processing on my M1 MacBook Air.
- **Solution:** Configured the script to use Apple's Metal Performance Shaders (MPS) for accelerated processing.

3. Distance Threshold Tuning:

- **Issue:** False positives due to improper threshold settings.
- **Solution:** Implemented a distance distribution analysis to find an optimal threshold.

6. Results and Findings

- **Detection Accuracy:** Achieved reasonable accuracy in detecting sachets with a fine-tuned distance threshold.
- **Performance:** Significant speedup using MPS compared to CPU-only execution.
- **Key Insight:** The sliding window approach was effective but resource-intensive. Reducing step size improved accuracy but increased computation time.

7. Key Learnings

- **Prototypical Networks:** Simple yet powerful for few-shot learning tasks.
- **EasyFSL:** Simplifies few-shot learning implementations with PyTorch.
- **MPS Utilization:** Leveraging MPS on M1 MacBook Air provided a practical boost for deep learning tasks.

8. Next Steps

1. **Optimize Performance:**
    - Explore alternative backbones or reduce feature dimensions.
2. **Expand Dataset:**
    - Incorporate data augmentation to enhance detection robustness.

References:

- Few-Shot Learning Playlist
- EasyFSL Documentation: GitHub