**Lab- 8**

**CSET340- Advanced Computer Vision and Video analytics**

**Task-1:- Interest Point Detection**, **Feature Matching and Contour Detection,**

**Interest Point Detection:-** Apply **SIFT (Scale Invariant Feature Transform)** Detector function using cv.SIFT\_create()

**Feature Matching:-** Feature matching is a fundamental technique in computer vision and image processing that involves finding correspondences between features detected in different images.

Use methods namely **ORB (Oriented FAST and Rotated BRIEF) and BFMatcher (Brute-Force Matcher).**

**Contour Detection with Custom Seeds**:- **Contour** represents the outline or boundary of an object, connecting continuous points of similar intensity or color.

In image processing, **edges** represent abrupt changes in brightness or color, while **contours** are closed curves that outline the shape or form of an object, often derived from edges.

Functions like markers, watershed, with some additional color placement functions can be used.

**Task-2.1:- Image Classification using Resnet network on Cifar 100**

**Objective:**

* Compare the performance of **Resnet 18** and **Resnet 34** an image classification task.
* Use **CIFAR-100**, a common dataset for image classification.
* Analyse model accuracy, loss, and inference time on a dataset.

Step 1: Installation of necessary libraries

Step 2: Load the dataset.

Step 3: Load the pretrained Model

Step 4: Train the Models

Step 5: Evaluate the Performances

Step 6: Compare the results.

**Datasets-**

1. CIFAR-100 dataset:- The CIFAR100 (Canadian Institute For Advanced Research) dataset consists of 100 classes with 600 color images of 32×32 resolution for each class.

This dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 super classes. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

A collage of many squares

Description automatically generated

1. Load CIFAR 100 Dataset Training Subset in Python
   1. import deeplake
   2. ds = deeplake.load("hub://activeloop/cifar100-train")
2. Load CIFAR 100 Dataset Testing Subset in Python
   1. import deeplake
   2. ds = deeplake.load("hub://activeloop/cifar100-test")
3. CIFAR 100 Dataset Structure
   1. **CIFAR 100 Data Fields**
      1. images: tensor containing images of the dataset.
      2. labels: tensor containing labels for their respective image.
      3. coarse\_labels: tensor containing superclass for their respective image.

**Task-2.2:- Meta learning approaches for image classification on MNIST dataset**

Objective:

In this lab assignment, you will explore meta-learning techniques such as few-shot and one-shot learning for image classification using the MNIST dataset. You will perform data preprocessing, implement meta-learning models, and evaluate their performance.

Tasks Overview:

1. Dataset Preprocessing
   * Load the MNIST dataset.
   * Apply normalization (**apply min-max scaling**).
   * Implement data augmentation (**apply Elastic Deformations -** It involves applying random distortions to images that simulate realistic variations while retaining the essential features of the objects within them.)
   * Split the dataset into training, validation, and test sets.
2. Building a Meta-Learning Framework
   * Implement data sampling strategies for one-shot and few-shot learning.
   * Create a support set and a query set for meta-training.
   * Implement episodic training for meta-learning models.
3. Implementing Few-Shot Learning Models
   * Apply Prototypical Networks for few-shot classification.
   * Use Siamese Networks for similarity-based classification.
   * Train and evaluate the models using the meta-learning framework.
4. Implementing One-Shot Learning Models
   * Apply Matching Networks to perform one-shot classification.
   * Experiment with a modified Siamese Network for one-shot learning.
   * Evaluate model performance on unseen digits.
5. Performance Evaluation and Analysis
   * Compare accuracy across different meta models.
   * Analyse failure cases and suggest improvements.

**Detailed Task Breakdown:**

Task 1: Dataset Preprocessing

* Load the MNIST dataset using TensorFlow/Keras or PyTorch.
* Normalize pixel values to the range [0,1].
* Apply data augmentation techniques i.e. Elastic Deformations.
* Split the dataset into training (80%), and test (20%) sets.

Task 2: Building a Meta-Learning Framework

* Define episodic training by selecting N classes randomly.
* Create a support set (K examples per class) and a query set.
* Implement data pipeline for dynamically generating episodes.
* Set up evaluation metrics suitable for meta-learning.

Task 3: Few-Shot Learning Models

3.1 Prototypical Networks:

* Compute class prototypes using the mean embedding of support examples.
* Use Euclidean distance to classify query samples.
* Implement training and testing loops.

3.2 Siamese Networks:

* Train a convolutional network to learn feature embeddings.
* Use contrastive loss to measure similarity between pairs of images.
* Perform evaluation using nearest neighbour classification.

Task 4: One-Shot Learning Models

4.1 Matching Networks:

* Implement attention-based feature matching.
* Use cosine similarity for classification.
* Train and evaluate the model on MNIST.

4.2 Siamese Networks for One-Shot:

* Modify the Siamese Network to work with one-shot pairs.
* Train with a focus on minimizing intra-class variance.
* Also, Evaluate using unseen digits.

Task 5: Performance Evaluation and Analysis

* Compute accuracy, precision, recall, and F1-score.
* Compare the performance of different meta-learning models.
* Discuss the impact of training data size on few-shot classification.
* Identify challenges in one-shot classification and propose solutions.

Submission Requirements:

1. Code Implementation: Submit Notebooks code for all tasks.
2. Report: Summarize your findings, including visualizations of training curves and accuracy comparisons.

Links for help:- <https://www.geeksforgeeks.org/contour-detection-with-custom-seeds-using-python-opencv/>

<https://www.geeksforgeeks.org/feature-matching-in-opencv/>

<https://www.geeksforgeeks.org/sift-interest-point-detector-using-python-opencv/>

<https://www.geeksforgeeks.org/check-if-image-contour-is-convex-or-not-in-opencv-python/>

Link for meta learning:- <https://colab.research.google.com/github/phlippe/uvadlc_notebooks/blob/master/docs/tutorial_notebooks/tutorial16/Meta_Learning.ipynb>

<https://github.com/shruti-jadon/Hands-on-One-Shot-Learning/blob/master/Ch02-MetricsBasedMethods/Matching%20Networks.ipynb>

<https://medium.com/@heyamit10/few-shot-object-detection-with-meta-learning-f47bd876661e>

Some libraries/ frameworks for use:-

**Higher** (for PyTorch)

* pip install higher
* Helps in **MAML** (Model-Agnostic Meta-Learning) by allowing differentiable optimization steps.

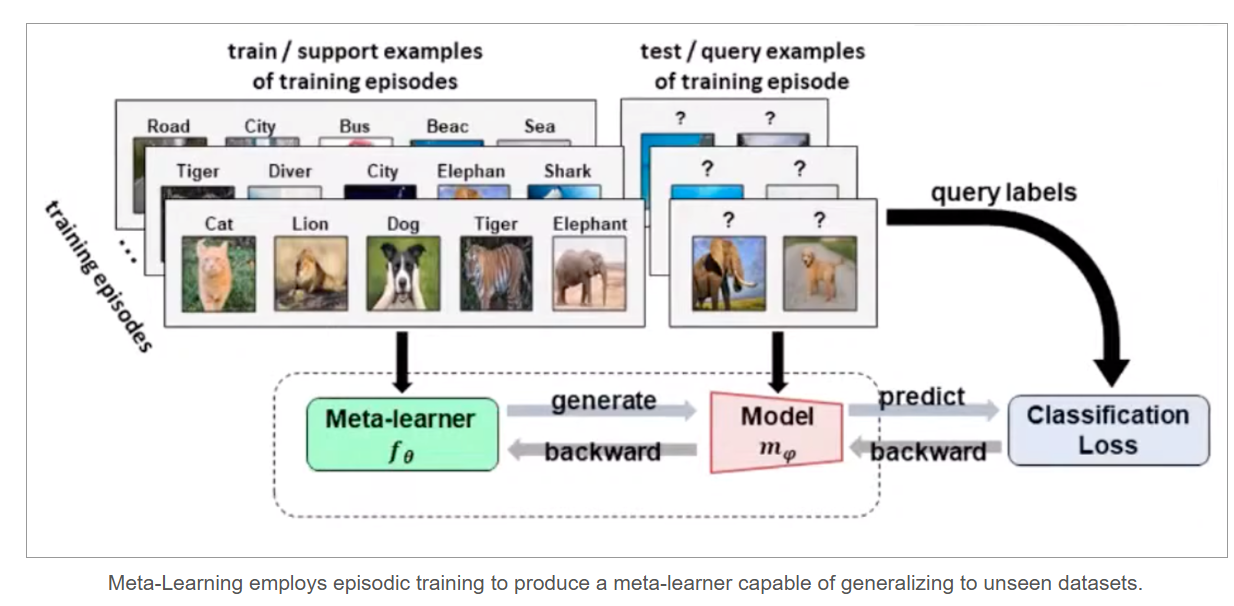
**Torchmeta**

* pip install torchmeta
* Provides prebuilt few-shot learning datasets and utilities for meta-learning in PyTorch.

**Learn2Learn (L2L)**

* pip install learn2learn
* A meta-learning framework for PyTorch supporting **MAML, Reptile, Prototypical Networks, and Meta-SGD**.

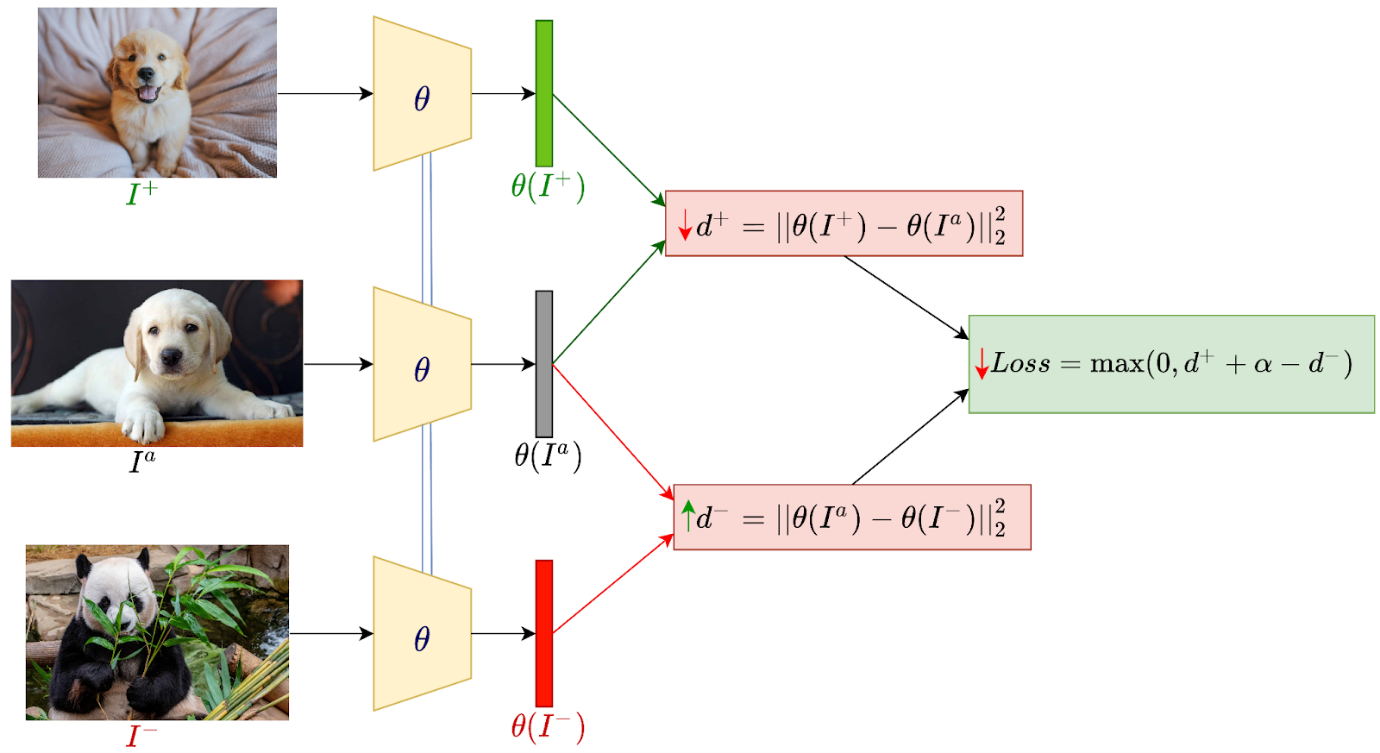
Prototypical Networks – few shot learning:-



Siamese Network architecture:-

A diagram of a network

Description automatically generated



Matching Network:- one shot learning:

