Case Study Assignment: Image Fragment Reconstruction via Machine Learning

Objective

You will develop a self-supervised machine learning model to group a large set of mixed image fragments back to their original source images. This task will test your ability to design, implement, train, and evaluate a model that can learn meaningful representations from fragmented images.

Dataset Overview

You will be provided with a dataset containing a set of images (**training** and **validation** split). Each image will be in a **square format 64x64 pixels, 3 color channels**. The content/label of these images is not important for this assignment.

A separate testing dataset will be used to evaluate the performance (after submission).

data.py: You will be provided with minimal code that loads **training** and **validation** data with generator that yields **N images** and applies **augmentation**

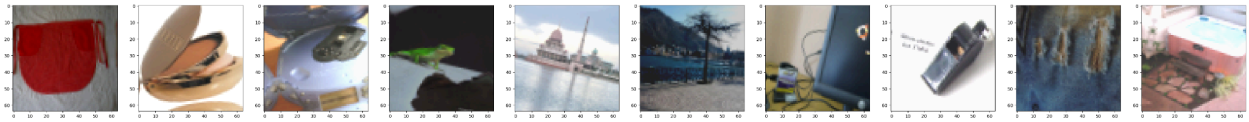
- Use generator output as a single **sample** in a training **batch**
- keep this code without changes if possible

Task Breakdown

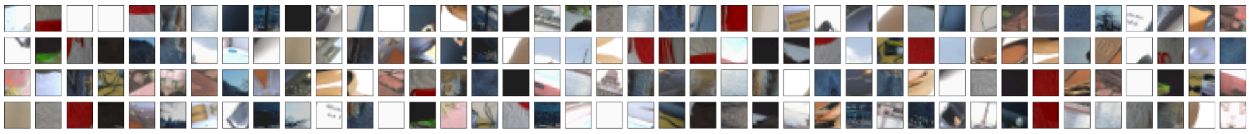
1. Data Preprocessing & Fragmentation

- Generate sample with 10 images
- Fragment each image in the **sample**: 4x4 non-overlapping fragments from a single image
- Collect all fragments into a single **unordered collection**.

- Original images



- Shuffled fragments



2. Model Development & Training

- Develop a self-supervised learning model that maps fragments back to their original images.
- Justify your choice of architecture (e.g., CNN, Autoencoder, etc.).
- Train the model using the prepared training dataset.
- Ensure the model can scale with increasing numbers of images and fragments.

3. Evaluation & Performance Metrics

- Define an appropriate performance metric to measure how well the model reconstructs fragment-source associations.
- Justify the chosen metric(s) (e.g., accuracy, clustering scores, recall, precision).
- The model does **not need to be perfect**, but performance should be clearly quantified and discussed.

4. (Optional) Advanced Challenge 1

- If you complete the core task, introduce **random noise and/or rotation** to fragments to evaluate model robustness.
- Describe how these augmentations affect model performance.

5. (Optional) Advanced Challenge 2

- Analyze the effects of fragment-size/number-of-fragments-per-image on the accuracy of the results.
- Describe the limitations and the scope of the model.

Technical Constraints & Expectations

- **Implementation**: You must write the code from scratch (i.e., no direct pre-trained model fine-tuning).
- **Resource Limitations**: The model should be designed to run on a local machine (no cloud or GPU access).
- **Code Deliverables**: Provide a structured codebase + training script and evaluation scripts (Python for src preferred, Python or Jupyter Notebook for training and evaluation scripts preferred).
- **Report Deliverable**: Prepare a concise report (max 3 pages) explaining:
    - Model architecture choice and justification
    - Training strategy
    - Evaluation results and performance analysis
    - Challenges and improvements

Evaluation Criteria

1. **Model Accuracy** – How well does the model reconstruct the original grouping?
2. **Engineering Rigor** – Is the implementation well-structured and efficient?
3. **Justification of Decisions** – Are architectural choices and metrics well-argued?
4. **Clarity of Report** – Are results and explanations clearly presented?

Submission Requirements

- Codebase in a GitHub repo or a well-organized folder.
- Model checkpoint.
- Evaluation scripts (executable with **validation data** and **model checkpoint**)
    - Script 1: metrics collection, loop over 1000? samples
    - Script 2: visualization of the model output and clustering on a single sample
    - Path to **validation data** and **model checkpoint** as the only 2 parameters to edit when scripts are executed
- Report (PDF format).
- A brief (~5 min) presentation explaining your approach.

This case study will test your ability to work with **image data, model selection, training strategies, and performance evaluation under computational constraints**.

Good luck!