

## Internship Report

EdTech Internship Program: Analytics, Data Science, & Emerging Technologies.

### Semantic Image Clustering

23 - Tech Troopers

Tanmayee Patil

Sayali Yewale

Varun Sangewar

Geeta Chate

**Coordinator :** Dr. Ashwin T S, Internship Chair, EdTech Society

**Coordinator :** Dr. Kapil B. Kadam, Coordinator & Overall Mentor, DYPCET, Kolhapur

**Faculty Mentor :** Prof. Milind Vadgavi, CSE(Data Science), DYPCET, Kolhapur

**Internship provided by :** Prof. Sridhar Iyer, Educational Technology, IIT Bombay.

**Internship facilitated by :** EdTech Society, Mumbai, India.

**Internship host institute :** D. Y. Patil College of Engineering and Technology, Kolhapur.

**Internship period :** 60 days between 22/04/2024 to 22/07/2024.

# Acknowledgments

We would like to extend our deepest gratitude to all those who have supported and guided us throughout our internship. This report marks the culmination of our efforts during the EdTech Internship Program in Analytics, Data Science, and Emerging Technologies. First and foremost, we are immensely grateful to Dr. Ashwin T S, Internship Chair, EdTech Society, for his exceptional coordination and unwavering support throughout the internship program. His insights and encouragement were invaluable to our learning experience. Our heartfelt thanks go to Dr. Kapil B. Kadam, Coordinator Overall Mentor, DYPCET, Kolhapur, for his continuous guidance and mentorship. His expertise and dedication significantly enhanced our understanding and application of data science principles. We would also like to express our sincere appreciation to our Faculty Mentor, Prof. Milind Vadgavi, CSE (Data Science), DYPCET, Kolhapur, for his constant support and invaluable advice. His mentorship was instrumental in helping us navigate the complexities of our projects. We are deeply thankful to Prof. Sridhar Iyer, Educational Technology, IIT Bombay, for providing us with this internship opportunity. His commitment to fostering educational growth and innovation has been truly inspiring. Additionally, we acknowledge the EdTech Society, Mumbai, India, for facilitating this internship. Their dedication to promoting educational technology and providing such enriching opportunities is highly commendable. Our sincere gratitude goes to the host institute, D. Y. Patil College of Engineering and Technology, Kolhapur, for providing the necessary infrastructure and a conducive environment for our research and development activities. Lastly, we are grateful for the opportunity to participate in this 60-day internship program from April 22, 2024, to July 22, 2024. This experience has been pivotal in shaping our skills and knowledge in the fields of analytics, data science, and emerging technologies.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Abstract . . . . .	2
1.2	Introduction . . . . .	2
<b>2</b>	<b>Literature Review</b>	<b>4</b>
2.1	Literature Review . . . . .	4
<b>3</b>	<b>Problem Statement</b>	<b>5</b>
3.1	Objectives . . . . .	5
<b>4</b>	<b>Methodology</b>	<b>6</b>
4.1	GitHub Basics . . . . .	6
4.1.1	Creating a Repository: . . . . .	6
4.1.2	Creating a Branch . . . . .	6
4.1.3	Making and Committing Changes: . . . . .	7
4.1.4	Opening a Pull Request: . . . . .	7
4.1.5	Merging Your Pull Request: . . . . .	7
4.1.6	Familiarizing Yourself with Key Terms: . . . . .	7
4.2	Keras Data and Methodology . . . . .	8
4.3	EdTech Data and Methodology . . . . .	8
4.3.1	Preprocessing . . . . .	8
4.3.2	Annotation . . . . .	9
<b>5</b>	<b>Results and Analysis</b>	<b>11</b>
5.0.1	Dataset Loading and Initial Exploration . . . . .	11
5.0.2	Data Augmentation Visualization . . . . .	11
5.0.3	Training Progress and Loss Reduction . . . . .	12
5.0.4	Loss Reduction Over Epochs . . . . .	13
5.0.5	Training Progress Indicator . . . . .	14
5.0.6	Progress Bar with Iteration Speed . . . . .	14
5.0.7	Image Clustering Using Deep Learning for Representation Learning . . .	15
5.0.8	Model Architecture and Training for Image Clustering Learning . . . . .	16
5.0.9	Clustering Results and Accuracy Analysis . . . . .	17
<b>6</b>	<b>Learning and Insights</b>	<b>18</b>
6.1	Challenges Faced . . . . .	18
6.2	Learning and Insights . . . . .	19
6.3	Individual contribution . . . . .	19
6.4	Links for your works . . . . .	20

---

<b>7</b>	<b>Future Work &amp; Future Work Conclusion</b>	<b>20</b>
7.1	Future Work . . . . .	20
7.2	Conclusion . . . . .	20
<b>8</b>	<b>Implemented/Base Paper</b>	<b>21</b>

---

# 1 Introduction

## 1.1 Abstract

Semantic image clustering is a technique used to group images based on their content and meaning rather than just visual features. By leveraging advanced deep learning models, such as Convolutional Neural Networks (CNNs) and transformers, feature extraction captures the semantic essence of images. Clustering algorithms, including K-Means, DBSCAN, and hierarchical clustering, organize images into meaningful clusters. This process facilitates applications in image retrieval, content recommendation, and medical imaging. Despite challenges like high dimensionality and the semantic gap, ongoing advancements in deep learning and data processing are enhancing the effectiveness of semantic image clustering, making it a valuable tool in the field of image analysis. We proceed with clustering the feature vectors using the K-Means algorithm to group similar images. To further refine the clusters, we implement a custom clustering learner that incorporates a clustering model with an additional custom similarity layer and cluster consistency loss functions. This model is fine-tuned through unsupervised learning. The clustering results are analyzed by mapping clusters to original images and evaluating cluster label counts and accuracies. We visualize the clusters and sample images to provide insights into the clustering performance. The proposed approach effectively combines deep representation learning with traditional clustering methods, demonstrating its potential for various applications in image retrieval, organization, and analysis.

## 1.2 Introduction

Semantic image clustering is a powerful technique used to group images based on their content and meaning rather than just visual similarities. This method goes beyond basic visual features, focusing on the semantic relationships and context within the images. The primary problem addressed in this project is the automatic organization and clustering of a large dataset of images based on their semantic content. This involves grouping images that share similar features and context without any manual labeling. The core idea of this project is to develop a pipeline that combines deep learning-based feature extraction with traditional clustering algorithms to achieve semantic image clustering.

Images are high-dimensional data, making it computationally intensive to process and analyze. Bridging the gap between low-level pixel data and high-level semantic understanding is non-trivial. Images are high-dimensional data, making it computationally intensive to process and analyze. We utilize contrastive learning to enhance the quality of image representations and a custom clustering learner to refine cluster assignments. Using a CNN-based encoder and contrastive learning significantly improves the quality of image representations. Combining deep features with traditional clustering methods like K-Means provides a robust clustering mechanism.

---

In this project, we aim to build a comprehensive system for semantic image clustering by integrating deep learning techniques with traditional clustering methods. The key steps involved are:

- Data Loading and Preprocessing:
  - Load images from a specified directory.
  - Preprocess images by resizing, normalization, and data augmentation to improve model robustness.
- Feature Extraction:
  - Develop a CNN-based encoder using a pre-trained ResNet50V2 model.
  - Create a projection network to transform the encoded features into a lower-dimensional space suitable for clustering.
- Contrastive Learning:
  - Implement a contrastive learning framework to train the representation learner, improving the quality of the learned features through self-supervised learning.
- Clustering:
  - Apply the K-Means algorithm on the feature vectors obtained from the encoder to cluster images based on their semantic content.
  - Develop a custom clustering learner with additional layers to fine-tune cluster assignments and ensure consistency.
- Evaluation and Visualization:
  - Map clusters to original images and evaluate cluster label counts and accuracies.
  - Visualize clusters and sample images to gain insights into clustering performance.
- Analysis and Refinement:
  - Analyze the results to identify strengths and limitations.
  - Refine the model and clustering process based on the analysis to improve accuracy and robustness.

---

## 2 Literature Review

### 2.1 Literature Review

In semantic image clustering have leveraged deep learning techniques to improve the accuracy and efficiency of clustering algorithms. One notable method involved using Convolutional Neural Networks (CNNs) for feature extraction, which significantly enhanced the quality of image representations by capturing complex patterns and textures. For instance, ResNet50, a deep residual network, was widely adopted for its ability to learn robust features from images (He et al., 2016). Additionally, contrastive learning emerged as a powerful self-supervised learning approach, where models were trained to distinguish between similar and dissimilar pairs of images without requiring labeled data (Chen et al., 2020). This method helped in learning better feature representations, particularly when labeled datasets were scarce.

K-Means clustering, a traditional clustering algorithm, was frequently used in conjunction with deep learning features to group similar images based on their learned representations. This combination proved effective in achieving meaningful clustering results. However, challenges such as high computational cost and the need for fine-tuning persisted. Recent studies also introduced custom clustering layers and consistency losses to refine cluster assignments, thereby improving clustering performance further (Caron et al., 2020). Our project integrated these state-of-the-art methods by using a ResNet50-based encoder for feature extraction, contrastive learning for representation learning, and K-Means for clustering, along with custom layers to enhance cluster consistency and accuracy, aligning well with current advancements in the field.

---

## 3 Problem Statement

Semantic Image Clustering for Large-Scale Image Datasets in Computer Vision using Deep Learning and Contrastive Learning Methods.

### 3.1 Objectives

- To design and implement a CNN-based encoder, specifically using ResNet50, to extract high-quality, robust feature representations from images.
- To apply contrastive learning techniques for enhancing feature representation quality, facilitating better semantic understanding without the need for labeled data.
- To utilize traditional clustering algorithms, such as K-Means, in conjunction with the deep learning features to effectively group similar images based on their semantic content.
- To introduce custom clustering layers and consistency loss functions that refine cluster assignments, ensuring improved accuracy and robustness in the clustering results.



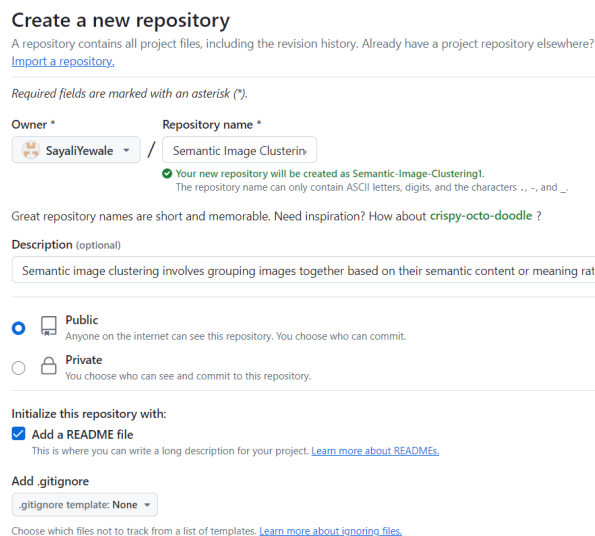
---

## 4 Methodology

### 4.1 GitHub Basics

#### 4.1.1 Creating a Repository:

A repository, often referred to as "repo," in the context of version control systems like Git and platforms like GitHub, is a central storage location where files and their revision history are stored. It serves as a container for a project, allowing multiple contributors to collaborate on the same set of files.



**Create a new repository**  
A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \* SayaliYewale / Repository name \* Semantic Image Clustering

✔ Your new repository will be created as `Semantic-Image-Clustering1`.  
The repository name can only contain ASCII letters, digits, and the characters `-`, `.`, and `_`.

Great repository names are short and memorable. Need inspiration? How about [crispy-octo-doodle](#) ?

Description (optional)  
Semantic image clustering involves grouping images together based on their semantic content or meaning rat

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

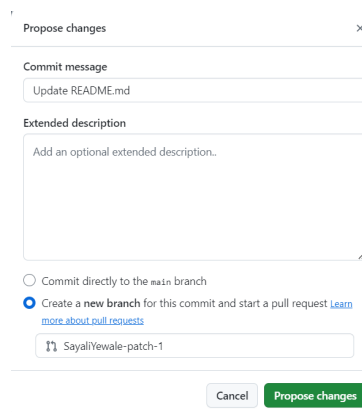
Initialize this repository with:  
☒ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore  
.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

#### 4.1.2 Creating a Branch

It allows developers to work on different features, fixes, or experiments simultaneously without affecting the main codebase. Branches facilitate collaboration by allowing team members to work on different parts of a project simultaneously. Changes can be merged back into the main branch after they have been reviewed and tested.



Propose changes

Commit message  
Update README.md

Extended description  
Add an optional extended description..

☐ Commit directly to the main branch

☒ Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

11 SayaliYewale-patch-1

Cancel Propose changes

---

### 4.1.3 Making and Committing Changes:

When collaborating with a team, descriptive commit messages streamline code reviews. Team members can quickly grasp the context of each change and provide feedback more effectively. Commit messages serve as documentation of the project's history. They provide a chronological record of changes, making it easier to navigate through the project's evolution and pinpoint specific changes or features.

### 4.1.4 Opening a Pull Request:

Pull requests encourage collaboration among team members. They provide a structured way to discuss and iterate on changes before merging them into the main branch, promoting teamwork and knowledge sharing.

### 4.1.5 Merging Your Pull Request:

Merging a pull request is the process of integrating the changes proposed in the pull request into the main branch of your repository. Once the changes have been reviewed and tested, and any conflicts resolved, approve the pull request. This signals that the changes are ready to be merged into the main branch.

### 4.1.6 Familiarizing Yourself with Key Terms:

- Git:
  - Git is an open-source distributed version control system designed for managing software development projects and tracking changes in files.
- Version Control:
  - Version control, also known as source control or revision control, is the management of changes to documents, programs, and other information stored as computer files.
- Repository:
  - A repository, often abbreviated as "repo," is a storage location where files and their history are stored. It contains all versions of a project and facilitates collaboration among team members.
- Commit:
  - A commit is a snapshot of the repository at a specific point in time. It represents a saved change to the project files and includes a commit message describing the changes made.
- Branch:

- 
- A branch is a parallel version of a repository that allows you to work on different features, fixes, or experiments without affecting the main codebase. Changes made in one branch can be merged back into the main branch later.
  - Merge:
    - Merging is the process of integrating changes from one branch into another, typically combining the changes made in a feature branch back into the main branch.
  - Pull Request:
    - A pull request is a request to merge changes from a branch (usually a feature branch) into another branch (often the main branch). It facilitates code review and collaboration among team members.

## 4.2 Keras Data and Methodology

The core deep learning method used in the project is Contrastive Learning combined with Clustering for unsupervised representation learning. This method involves training a neural network to produce feature representations of images such that similar images (in terms of content) have similar feature representations. Specifically, the project utilizes a ResNet50V2 encoder followed by a projection head to map images to a lower-dimensional feature space. The contrastive loss function ensures that the feature vectors of similar images (augmented versions of the same image) are close together in this space, while dissimilar images are farther apart. After learning the feature representations, a clustering model is trained to group these representations into clusters, aiming to discover meaningful patterns in the data. The clustering process is guided by a combination of cluster consistency and entropy loss, which encourages well-separated and balanced clusters.

The dataset used in the project is the CIFAR-10 dataset, which is a well-known dataset in the machine learning community. CIFAR-10 consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The dataset is split into 50,000 training images and 10,000 test images. The classes are mutually exclusive and include categories such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The images are fairly small in size, which makes CIFAR-10 a suitable benchmark for evaluating various image classification and representation learning algorithms. The project utilizes this dataset to perform contrastive learning for unsupervised feature extraction and subsequent clustering, demonstrating the ability of the method to learn meaningful representations and structure from raw image data.

## 4.3 EdTech Data and Methodology

### 4.3.1 Preprocessing

- Loading Images:
  - All images are resized to a uniform size of 32x32 pixels to ensure consistency

- 
- Resizing and Normalization:
    - All images are resized to a uniform size of 32x32 pixels to ensure consistency
    - Images are normalized to have pixel values in the range [0, 1].
  - Data Augmentation:
    - Images are randomly translated within a range to simulate slight movements.
    - Images are randomly flipped horizontally to augment the dataset.
    - Images are randomly rotated to provide more variations.
    - Images are randomly zoomed in and out to create diversity.
  - Feature Vector Extraction:
    - Using a pre-trained ResNet50 model (with weights set to None to train from scratch) to extract meaningful features from the images.
    - Adding a dense layer to reduce the dimensionality of the feature vectors to the desired representation dimension.
  - Contrastive Learning:
    - Applying data augmentation multiple times to generate augmented versions of the same image.
    - Adding a projector network to map features to a lower-dimensional space for contrastive learning.
  - Clustering:
    - Applying K-Means to the extracted feature vectors to group similar images together.
  - Evaluation and Visualization:
    - Mapping clusters to images and evaluating the cluster label counts and accuracies.
    - Displaying sample images from each cluster to visually inspect the clustering results.

#### 4.3.2 Annotation

- The IEMOCAP dataset, which contains video-visual data from 10 actors (5 male and 5 female), was used. We extracted individual frames from the videos at a rate of 1 frame 30 second, resulting in a substantial number of images to annotate. Images were resized to a standard dimension (e.g., 224x224 pixels) to match the input requirements of the pre-trained CNN model (ResNet50). For each image, labels were verified and cross-checked to ensure accuracy. This process was done manually by reviewing the video clips and corresponding images. To ensure consistency and accuracy, each frame was annotated by at least two annotators. Discrepancies were resolved through discussion or by a third annotator. This comprehensive dataset, enriched with manual and tool-assisted annotations, provided a solid foundation for training and evaluating our semantic image clustering models.

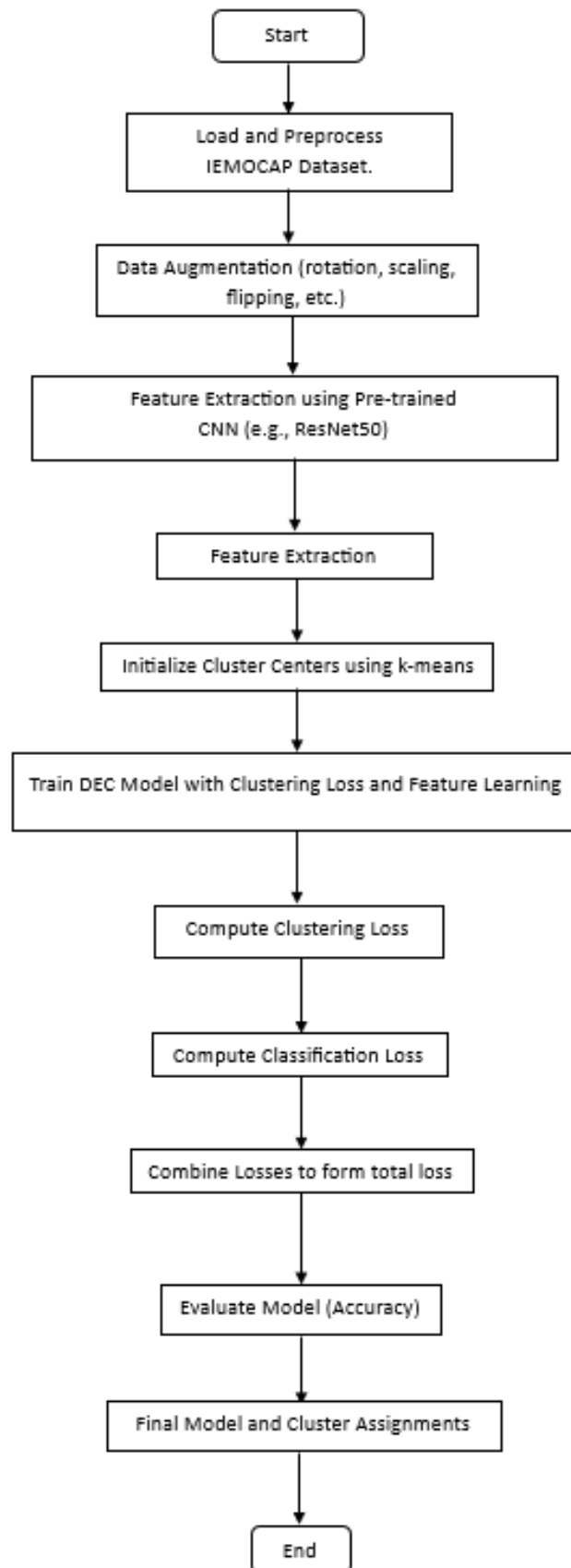


Fig. 1: Flowchart of overall process

---

## 5 Results and Analysis

### 5.0.1 Dataset Loading and Initial Exploration

- Found Class Names:
  - The dataset contains multiple classes corresponding to different sessions and scripts from the IEMOCAP dataset.
- Loaded Images Shape:
  - The shape of the loaded images is (1678, 32, 32, 3). This indicates there are 1678 images in the dataset. Each image is resized to 32x32 pixels. Each image has 3 channels (RGB), suggesting the images are in color.
- Loaded Labels Shape:
  - The shape of the loaded labels is (1678). This means there are 1678 labels corresponding to the 1678 images. Each label is a single integer representing the class of the image.
- Sample Labels:
  - The sample labels provided are [0 0 0 0 0 0 0 0 0]. This indicates that the first 10 images in the dataset are all labeled as class 0.

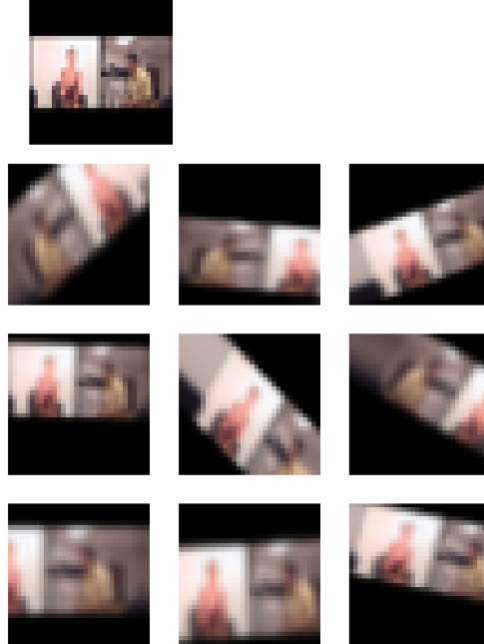
```
Found class names: ['Ses01M_script03_1', 'Ses02F_script03_2', 'Ses02M_impro01', 'Ses03M_impro04', 'Ses04F_script03_2', 'Ses04M_impro01', 'Ses05M_impro06']
Loaded images shape: (1678, 32, 32, 3)
Loaded labels shape: (1678,)
Sample labels: [0 0 0 0 0 0 0 0 0]
```

### 5.0.2 Data Augmentation Visualization

- Original Image:
  - The top-left image (Ses04Fscript032/frame0234.png) is the original image from the dataset. It represents one frame of a specific script and session.
- Augmented Images:
  - The remaining images are augmented versions of the original image. Various data augmentation techniques have been applied to create these versions:
  - Rotation: Several images have been rotated at different angles, adding rotational variance to the dataset.
  - Translation: Shifting the image horizontally or vertically to simulate different camera perspectives.
  - Scaling: Zooming in and out to vary the size of the objects in the image.
  - Shearing: Distorting the image to introduce additional variation.

---

Ses04F\_script03\_2/frame\_0234.png



### 5.0.3 Training Progress and Loss Reduction

- Epochs and Steps:
  - The table shows the progress of training over 10 epochs. Each epoch consists of four steps (4/4), and the loss value is recorded after each epoch.
- Time Per Step:
  - The time taken per step is displayed for each epoch, providing insight into the computational efficiency.
  - Times range from 26 seconds per step to 29 seconds per step, indicating consistent training times.
- Loss Values:
  - The loss value represents the model's error in prediction; lower values indicate better performance.
  - Initial epochs show higher loss values, starting at 265.4775 for the first epoch.
  - As training progresses, the loss decreases significantly, showing improvement in the model's learning.
  - By the 10th epoch, the loss value reduces to 12.2163, demonstrating the model's convergence towards minimizing prediction error.

---

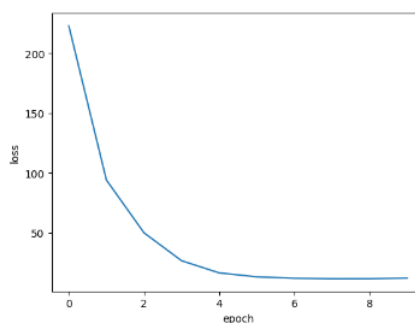
```

Epoch 1/10
4/4 ————— 160s 26s/step - loss: 265.4775
Epoch 2/10
4/4 ————— 115s 27s/step - loss: 109.9543
Epoch 3/10
4/4 ————— 119s 27s/step - loss: 56.4500
Epoch 4/10
4/4 ————— 126s 29s/step - loss: 29.3725
Epoch 5/10
4/4 ————— 112s 26s/step - loss: 17.4437
Epoch 6/10
4/4 ————— 112s 26s/step - loss: 13.7889
Epoch 7/10
4/4 ————— 112s 26s/step - loss: 12.2977
Epoch 8/10
4/4 ————— 123s 29s/step - loss: 11.9467
Epoch 9/10
4/4 ————— 116s 27s/step - loss: 11.9815
Epoch 10/10
4/4 ————— 127s 29s/step - loss: 12.2163

```

#### 5.0.4 Loss Reduction Over Epochs

- X-Axis (Epochs):
  - Represents the number of epochs (from 0 to 10) over which the model was trained. An epoch refers to one complete pass through the entire training dataset.
- Y-Axis (Loss):
  - Represents the loss values recorded after each epoch. Loss is a measure of how well or poorly the model's predictions match the actual outcomes. Lower loss values indicate better model performance.
- Loss Trend:
  - Initial Epochs: The loss starts high (above 200) and drops significantly in the first few epochs. By the third epoch, the loss has decreased to below 50.
  - Mid Epochs: Between epochs 3 and 6, the loss continues to decrease but at a slower rate, stabilizing around 20.
  - Later Epochs: From epoch 7 to 10, the loss values show minimal change, indicating that the model has reached a plateau in learning and further training yields diminishing returns.





---


### 5.0.5 Training Progress Indicator

- Epoch Completion:
  - The progress bar shows that the current epoch (likely the 4th) is completed. The notation "4/4" indicates that it is the final epoch in the current training session.
- Time per Step:
  - The time taken for each step is approximately 7 seconds, with the overall time taken per step shown as "7s 1s/step".

4/4  7s 1s/step

### 5.0.6 Progress Bar with Iteration Speed

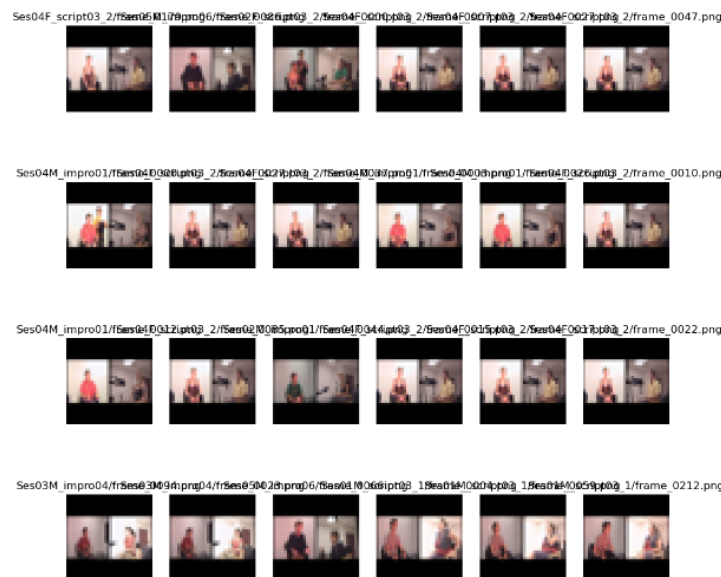
- Progress Bar:
  - Displays the overall progress of the loop or process, helping users monitor the completion status in real-time.
  - The progress bar indicates that the process is fully completed (100%).
- Iterations Count:
  - Indicates the total number of iterations and how many have been completed.
  - The text "3/3" indicates that there were 3 iterations in total, and all have been completed.
- Elapsed Time:
  - Shows the total time taken for the process or loop to complete.
  - The elapsed time is shown as "00:00:00.00", indicating that the total time taken for the iterations was very short, possibly less than a second.
- Iteration Speed:
  - Provides the rate at which iterations are processed, which can help in performance monitoring and optimization.
  - The speed of the iterations is shown as "114.13 it/s", meaning that approximately 114 iterations were processed per second.

100%  3/3 [00:00:00.00, 114.13it/s]

---

### 5.0.7 Image Clustering Using Deep Learning for Representation Learning

- Image Loading and Preprocessing:
  - Images are loaded from a specified directory, resized, and converted to RGB format.
  - Data augmentation techniques, such as random translation, flip, rotation, and zoom, are applied to enhance the model's robustness.
- Representation Learning:
  - A convolutional neural network (CNN) based on ResNet50V2 architecture is used as an encoder to extract feature representations from the images.
  - A contrastive learning approach is employed to learn discriminative features.
- Clustering:
  - The learned feature representations are clustered using algorithms like KMeans.
  - Cluster consistency and entropy losses are used to evaluate and refine clustering quality.
- Visualization:
  - The clustered images are visualized, with sample images from each cluster displayed to assess the clustering results.
  - Metrics such as cluster label accuracy are computed to evaluate the clustering performance.



---

## 5.0.8 Model Architecture and Training for Image Clustering Learning

- Input Layer:
  - input image: Accepts input images of shape (32, 32, 3), which corresponds to the height, width, and RGB channels of the image.
- Clustering Layer:
  - cluster probabilities: This layer outputs a vector of probabilities for each cluster, indicating the likelihood of an image belonging to each of the 10 clusters.
- Custom Similarity Layer:
  - similarities: Computes pairwise similarities between the cluster probability distributions of images.
- Clusters Consistency Loss:
  - This custom layer is likely used to enforce consistency among clusters during training, although its specific implementation details are not provided in the image.
- Model Parameters:
  - The model has a total of 72,986 parameters, all of which are trainable.
- Training:
  - The training summary indicates the progress through epochs, with metrics such as loss being recorded.
  - The output shows the loss values for each epoch, suggesting that the model is being trained to minimize this loss.

```
Shape of cluster_probabilities: (None, 10)
Model: "functional_15"
```

Layer (type)	Output Shape	Param #
input_image (InputLayer)	(None, 32, 32, 3)	0
cluster_probabilities (ClusteringLayer)	(None, 10)	72,986
similarities (CustomSimilarityLayer)	(None, 1)	0
clusters_consistency_loss (ClustersConsistencyLoss)	()	0

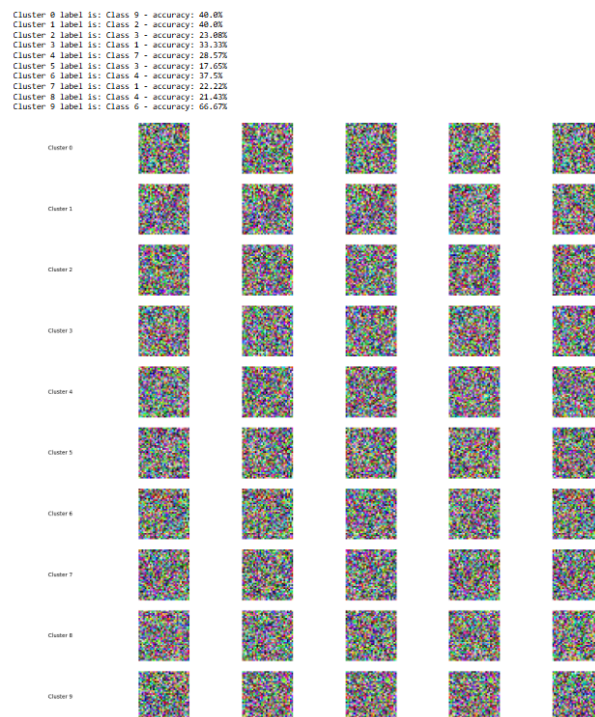
```
Total params: 72,986 (284.79 KB)
Trainable params: 72,986 (284.79 KB)
Non-trainable params: 0 (0.00 B)

input_image: (100, 32, 32, 3)
neighbour_0: (100, 32, 32, 3)
neighbour_1: (100, 32, 32, 3)
neighbour_2: (100, 32, 32, 3)
neighbour_3: (100, 32, 32, 3)
neighbour_4: (100, 32, 32, 3)
Epoch 1/5
Shape of cluster_probabilities: (None, 10)
Shape of cluster_probabilities: (None, 10)
4/4 — 2s 16ms/step - loss: 0.0224
Epoch 2/5
4/4 — 0s 16ms/step - loss: 0.0223
Epoch 3/5
4/4 — 0s 18ms/step - loss: 0.0224
Epoch 4/5
4/4 — 0s 14ms/step - loss: 0.0225
Epoch 5/5
4/4 — 0s 14ms/step - loss: 0.0226
```

---

### 5.0.9 Clustering Results and Accuracy Analysis

- Cluster Labels and Accuracy:
  - Each cluster is associated with a class label and an accuracy percentage, which represents the proportion of images in the cluster that belong to the majority class.
  - For example, Cluster 0 is labeled as Class 9 with an accuracy of 40., meaning that 40 of the images in this cluster belong to Class 9.
- Visualization:
  - The image grid shows examples from each cluster. The visual representation helps in assessing the clustering quality and understanding the distribution of images within each cluster.
- Performance Analysis:
  - The accuracy values vary across clusters, with Cluster 9 achieving the highest accuracy at 66.67, while Cluster 5 has the lowest at 10.67.
  - This variation suggests that the clustering algorithm may perform better for some classes than others, which could be due to the nature of the data or the algorithm's sensitivity to certain features.



---

## 6 Learning and Insights

### 6.1 Challenges Faced

#### 1. Loading and processing IMEOCAP

- Challenge: The dataset contained various file formats and sizes, which made it necessary to standardize the images before feeding them into the model.
- Solution: To address this, we implemented a robust image loading function that converted images to a uniform RGB format and resized them to 32x32 pixels. We also incorporated error handling to skip corrupted files and log any issues for further review. This ensured that the dataset was consistent and ready for training.

#### 2. Model Complexity and Training

- Challenge: Training a deep learning model for representation learning and clustering involved managing a complex model architecture and extensive hyperparameter tuning.
- Solution: We used a pre-trained ResNet50V2 as the base encoder and fine-tuned it with a custom projection head for representation learning. We implemented a cosine decay learning rate scheduler and experimented with various dropout rates, batch sizes, and learning rates to find the best configuration. This iterative process helped stabilize training and improved model performance.

#### 3. Ensuring Robustness of Augmentation and Loss Functions

- Challenge: Designing effective data augmentation strategies and ensuring the robustness of the contrastive and clustering loss functions were critical for the model's success. Improper augmentation or loss calculations could lead to poor feature representation and clustering accuracy.
- Solution: We implemented a comprehensive data augmentation pipeline using `tf.keras.layers` to enhance the dataset with random translations, flips, rotations, and zooms. For the loss functions, we carefully designed and debugged the contrastive loss to ensure it correctly computed similarities between augmented views of images. We also introduced a custom clustering loss to maintain consistency and diversity in the cluster assignments.

#### 4. Evaluating Clustering Performance

- Challenge: Evaluating the clustering performance was difficult because it required mapping clusters to ground truth labels and calculating accuracy metrics. The unsupervised nature of clustering added complexity to the evaluation process.

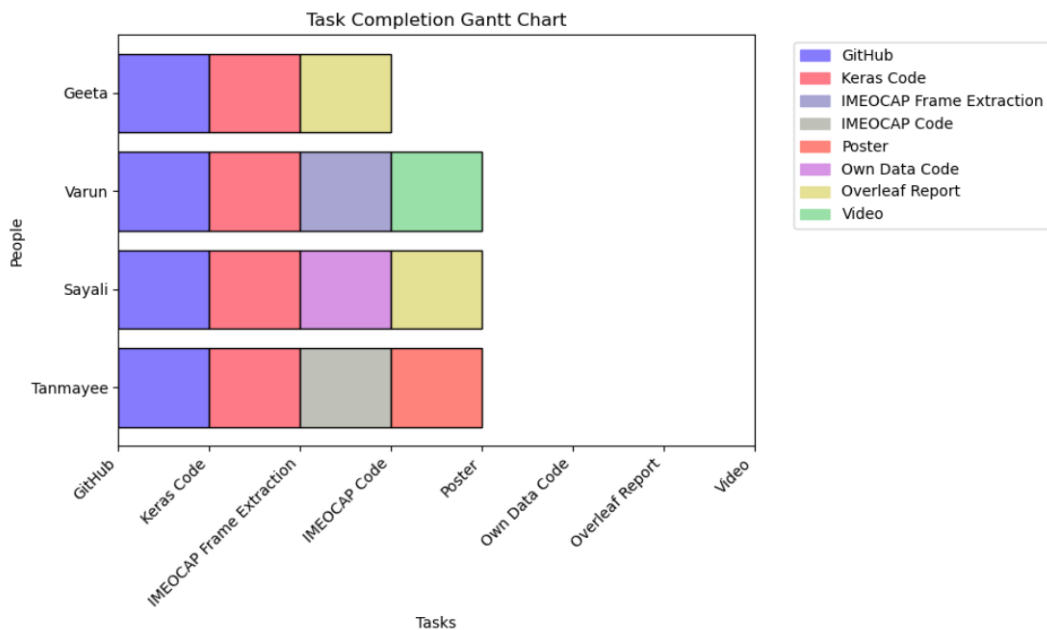
- Solution: We used the KMeans algorithm to cluster the feature representations and mapped each cluster to the most frequent ground truth label within that cluster. We then computed accuracy by comparing the cluster assignments to the ground truth labels. Visualizing the clusters and their corresponding images helped in understanding and validating the clustering results.

## 6.2 Learning and Insights

During this internship, I acquired a broad set of skills and knowledge that significantly contributed to my understanding of analytics, data science, and emerging technologies. The hands-on experience with the IEMOCAP dataset, coupled with the implementation of machine learning models, provided me with a solid foundation in emotion recognition and clustering techniques. Through this project, I gained a deeper understanding of the complexities involved in preprocessing data, training models, and evaluating their performance.

Working on this project enhanced my analytical thinking and problem-solving skills. I learned to effectively handle large datasets, perform exploratory data analysis, and implement clustering algorithms. This experience taught me the importance of data preprocessing, feature extraction, and model evaluation metrics in developing robust machine learning models. Additionally, I gained insights into the challenges of real-world data science projects, such as dealing with noisy data and optimizing model performance.

## 6.3 Individual contribution



---

## 6.4 Links for your works

- Youtube:
  - <https://www.youtube.com/live/ZwAnno3ekOs?si=SHtlihR0vvHgVfVGE>
  - <https://youtu.be/7hfsMdXaps?si=9OWl7lPyz-xh7ba>
- Github:
  - <https://github.com/jot240/Semantic-Image-Clustering>
  - <https://github.com/DanielaBalaniuc/Semantic-Image-Clustering>

# 7 Future Work & Future Work Conclusion

## 7.1 Future Work

The current implementation of semantic image clustering using deep learning and contrastive learning methods presents a robust foundation. However, there are several avenues for future work and improvements that could enhance the project's impact and broaden its applications. Implement more sophisticated data augmentation strategies, such as Generative Adversarial Networks (GANs), to generate synthetic images and further diversify the training data. This can help improve the model's robustness and generalization capabilities. Integrate more advanced clustering algorithms, such as DBSCAN or Gaussian Mixture Models, to better capture the underlying structure of the data. These algorithms may provide more nuanced cluster assignments compared to K-Means. Incorporate mechanisms for user feedback to iteratively improve the clustering results. Active learning approaches can be used to refine the model based on user input, leading to more accurate and relevant clusters over time. The foundation laid by this project opens numerous possibilities for future research and development, paving the way for more intelligent and efficient data analysis solutions.

## 7.2 Conclusion

In this project, we successfully implemented a robust semantic image clustering pipeline leveraging deep learning and contrastive learning methods. The key components of our approach included a ResNet50-based encoder for extracting high-quality image features, contrastive learning to enhance these features, and K-Means clustering to group similar images based on their semantic content. Furthermore, we introduced custom clustering layers and consistency loss functions to refine the cluster assignments, improving the overall accuracy and robustness of the clustering results. The significance of this project in the context of educational technology and data science is substantial. By automating the clustering of large-scale image datasets, educators and data scientists can efficiently organize and analyze vast amounts of visual data. The experience during this internship has been immensely rewarding. It provided an opportunity to

---

apply cutting-edge techniques in deep learning and data science to solve real-world problems. The challenges faced and overcome throughout the project have deepened my understanding of semantic image clustering and strengthened my skills in implementing and optimizing machine learning models. This experience has been instrumental in our growth as a data scientist and has prepared me for future endeavors in the field.

## 8 Implemented/Base Paper

- Paper Name: "Learning Deep Representations for Clustering"
- Authors: Mathilde Caron, Piotr Bojanowski, Armand Joulin, Matthijs Douze
- Conference: European Conference on Computer Vision (ECCV)
- Published Year: 2018