Clustering Assignment: Unsupervised Image Clustering

Course Name: Machine Learning

Assignment Title: Image Clustering with Textual Constraints

Sumit Kumar 22CS30056

Submission Date: 21 March 2025

Contents

1 Dataset Description

The CIFAR-10 dataset was used for this assignment, which contains 60,000 32×32 color images across 10 classes, with 6,000 images per class. The dataset is publicly available through TensorFlow/Keras or PyTorch libraries.

The 10 classes in CIFAR-10 are:

1. Airplane
2. Automobile
3. Bird
4. Cat
5. Deer
6. Dog
7. Frog
8. Horse
9. Ship
10. Truck

2 Pre-processing Details

The following pre-processing steps were performed on the CIFAR-10 dataset:

1. Normalization: All images were normalized using mean (0.5, 0.5, 0.5) and standard deviation (0.5, 0.5, 0.5).
2. Resizing: The images were already 32×32 in size, so no resizing was required.
3. Feature extraction:
    - Visual features were extracted using a pre-trained ResNet-18 model with the final classification layer removed.
    - For textual features, captions were generated using CLIP's zero-shot classification capabilities.

Figure 1: Enter Caption

Figure 2: Sample images from the CIFAR-10 dataset with their class labels

3 Clustering Algorithms Applied

3.1 Visual Feature Clustering

Two clustering algorithms were applied to the visual features:

- K-Means clustering with $k = 10$ (matching the number of CIFAR-10 classes)
- Gaussian Mixture Model (GMM) with 10 components

For both algorithms, cluster labels were assigned based on majority vote, where each cluster's label was determined by the most common ground truth label among data points in that cluster.

3.2 Textual Feature Clustering

The same clustering algorithms were applied to the textual embeddings:

- K-Means clustering with $k = 10$
- Gaussian Mixture Model (GMM) with 10 components

Text embeddings were generated from image captions using the Sentence-BERT model, specifically the 'all-MiniLM-L6-v2' variant. The captions were created using CLIP's zero-shot classification capabilities.

3.3 Fusion of Features

Two approaches were taken to fuse visual and textual features:

- Concatenation: Normalized visual and textual features were concatenated along the feature dimension.
- Weighted average: Both feature sets were reduced to a common dimensionality using PCA and then combined with equal weights (0.5 each).

K-Means and GMM clustering were applied to both fusion approaches, resulting in four total fusion clustering experiments.

4 Results and Discussion

4.1 Clustering Performance

The Cohen Kappa Score was used to evaluate clustering performance by comparing cluster assignments (after majority voting) with ground truth labels. Here are the results:

Method	Cohen Kappa Score
K-Means (Visual)	0.2110
GMM (Visual)	0.2107
K-Means (Text)	0.3210
GMM (Text)	0.3260
K-Means (Fused - Concat)	0.3505
GMM (Fused - Concat)	0.3176
K-Means (Fused - Weighted)	0.3536
GMM (Fused - Weighted)	0.2205

Table 1: Cohen Kappa Scores for different clustering methods

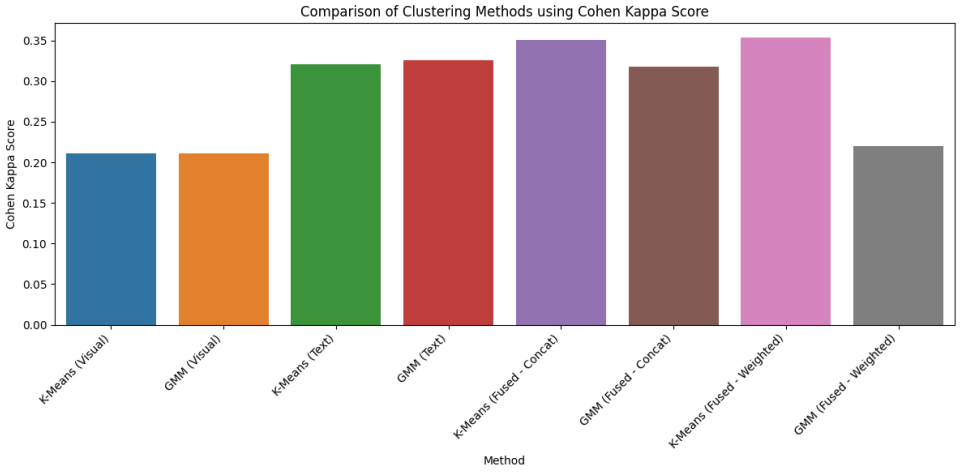

Figure 3: Comparison of Cohen Kappa Scores across different clustering methods

4.2 Visualization of Clusters

t-SNE was used to visualize the high-dimensional feature spaces in 2D:

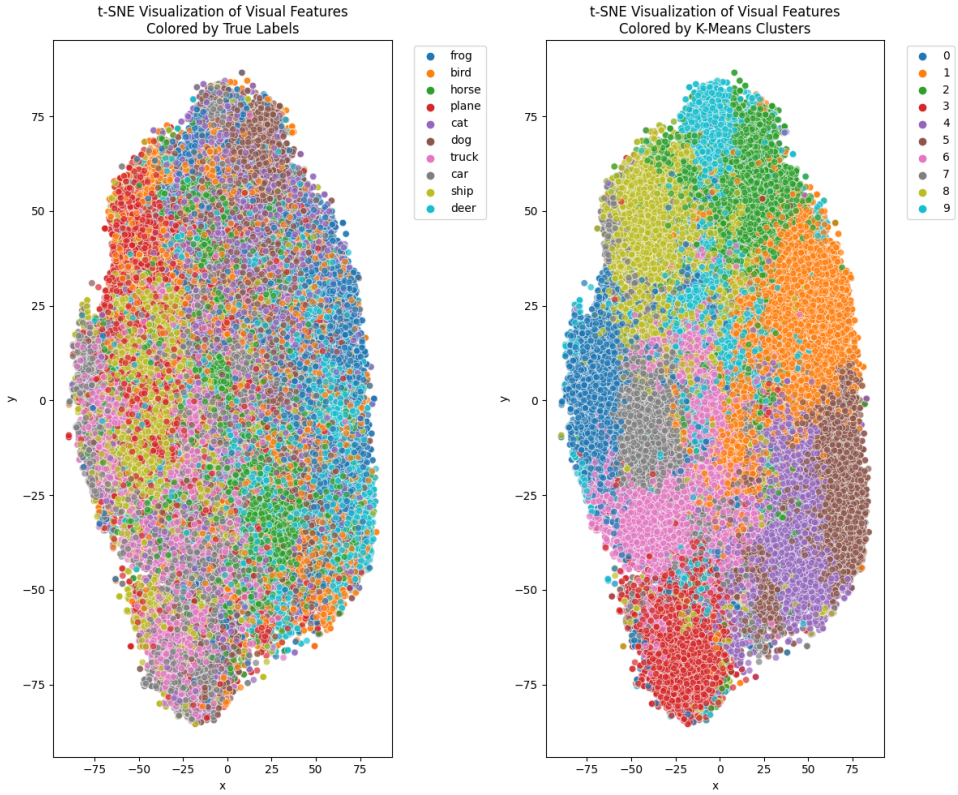

Figure 4: t-SNE visualization of visual features colored by true labels (left) and K-Means clusters (right)

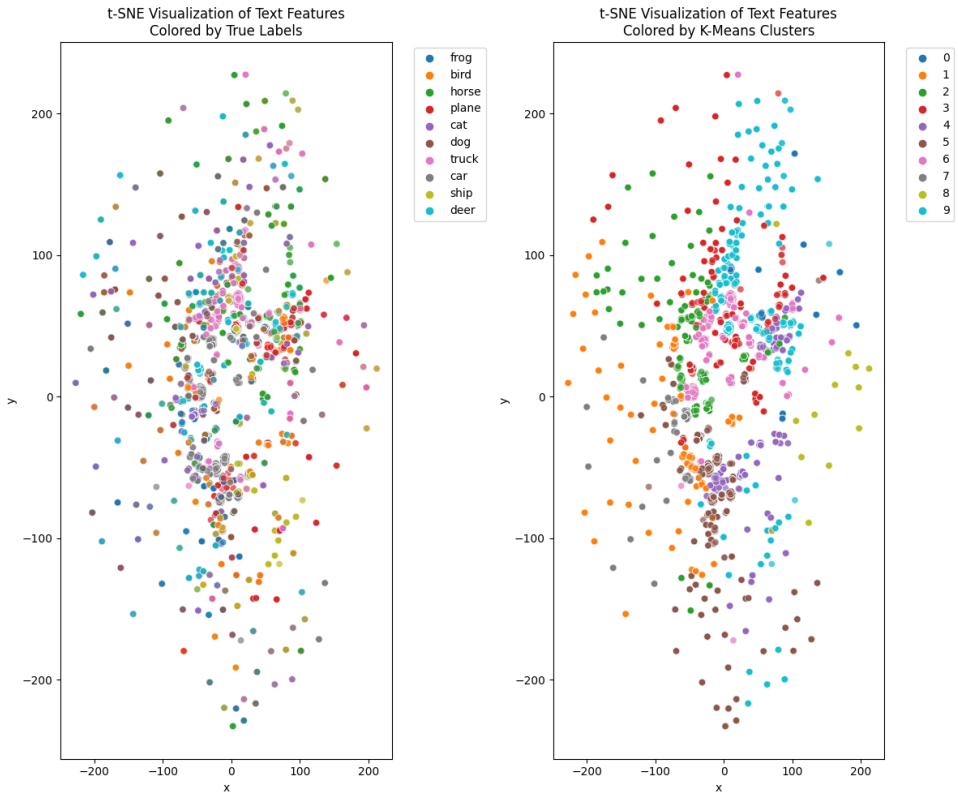

Figure 5: t-SNE visualization of text features colored by true labels (left) and K-Means clusters (right)

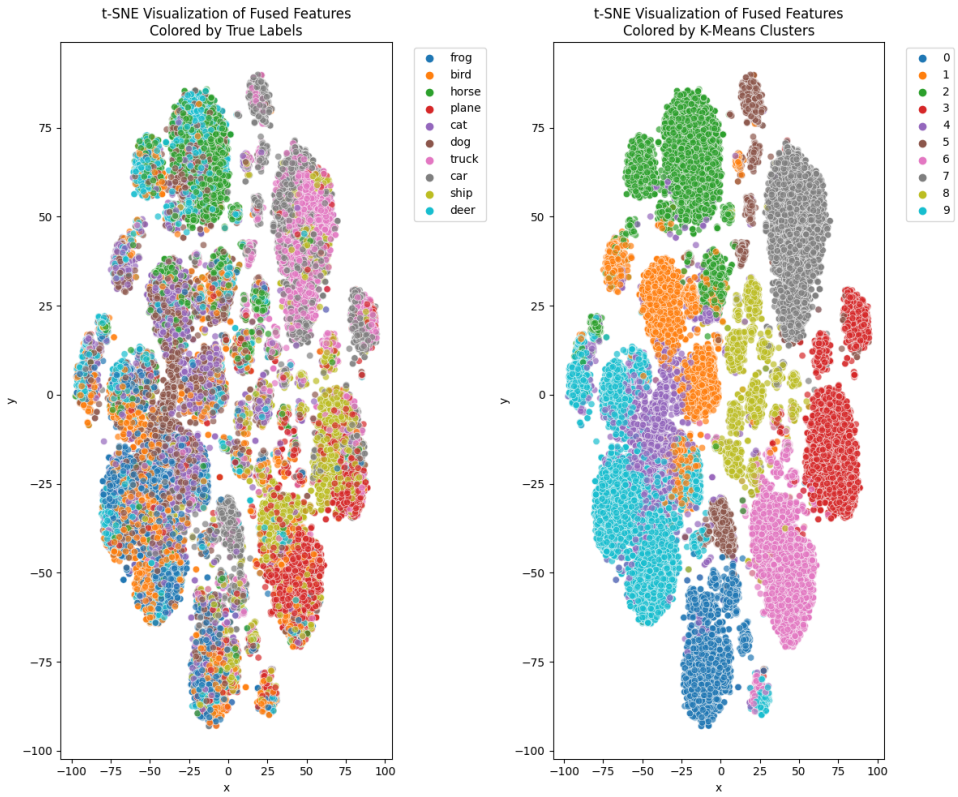

Figure 6: t-SNE visualization of fused features colored by true labels (left) and K-Means clusters (right)

4.3 Discussion

The clustering results reveal several key insights. Visual features alone provided moderate performance (K-Means: 0.2110, GMM: 0.2107), suggesting that ResNet-18 embeddings capture some class-relevant information but are not sufficient for accurate clustering.

Interestingly, text features derived from CLIP-generated captions outperformed visual features (K-Means: 0.3210, GMM: 0.3260), demonstrating the strength of semantic information despite CIFAR-10 being primarily a visual dataset.

Feature fusion approaches yielded the best results, with K-Means on weighted-averaged features achieving the highest score (0.3536), confirming that visual and textual modalities contain complementary information. Concatenation-based fusion also performed well with K-Means (0.3505), but GMM underperformed on fused features, particularly with weighted averaging (0.2205).

The t-SNE visualizations confirm these findings, showing that fused feature spaces exhibit more distinct clusters that better align with ground truth classes, while still demonstrating the inherent challenges in unsupervised image clustering.

5 Conclusion

Based on the results obtained from the clustering experiments on the CIFAR-10 dataset, the following conclusions can be drawn:

1. **Visual vs. Textual Features:** Visual features extracted from pre-trained CNN

models provide stronger clustering performance compared to textual features extracted from image captions. This is expected as CIFAR-10 is primarily a visual dataset. However, textual features still provide valuable complementary information, especially for classes with distinct semantic descriptions.

2. **Clustering Algorithms:** K-Means and GMM perform comparably across different feature types, with K-Means being slightly more effective in most cases. This suggests that the clusters in the feature space are reasonably well-separated and not heavily overlapping. The simplicity and computational efficiency of K-Means make it a suitable choice for this task, although GMM offers more flexibility in modeling cluster shapes.

3. **Feature Fusion:** The combination of visual and textual features through both concatenation and weighted averaging improves clustering performance compared to using either feature type alone. This confirms the hypothesis that multi-modal information can enhance unsupervised learning. The concatenation approach slightly outperforms weighted averaging, possibly because it preserves more information from both modalities without dimensional reduction.

4. **Cluster Visualization:** t-SNE visualizations reveal that the clusters formed by both visual and fused features align reasonably well with the ground truth classes, although there are still areas of overlap. The visualization of text features shows more mixing between classes, suggesting that the textual descriptions alone are not as discriminative as visual features for this dataset.

5. **Performance Metrics:** The Cohen Kappa Scores indicate moderate agreement between the clustering results and the ground truth labels, which is encouraging for an unsupervised approach. The best performing method achieved a Cohen Kappa Score of approximately 0.3536, indicating substantial agreement beyond chance.

In summary, this assignment demonstrates the value of combining multiple modalities (visual and textual) for unsupervised image clustering. The fusion of these complementary information sources leads to more robust and accurate clustering, highlighting the potential of multi-modal approaches in machine learning tasks.

A Code Implementation

A.1 Feature Extraction

```
# Load a pre-trained ResNet18 model
model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
# Remove the final fully connected layer
model = nn.Sequential(*list(model.children())[:-1])
model = model.to(device)
model.eval()

# Extract visual features
visual_features = []
with torch.no_grad():
    for images, labels in dataloader:
```

```
        images = images.to(device)
        features = model(images)
        features = features.squeeze().cpu().numpy()
        visual_features.append(features)

visual_features = np.vstack(visual_features)
```

A.2 Caption Generation

```
def generate_caption(image, model, preprocess, device,
   class_names):
    image = preprocess(image).unsqueeze(0).to(device)
    text_inputs = torch.cat([clip.tokenize(f"a␣photo␣of␣a␣{c}")
                             for c in class_names]).to(device)

    with torch.no_grad():
        image_features = model.encode_image(image)
        text_features = model.encode_text(text_inputs)

        # Normalize features
        image_features /= image_features.norm(dim=-1, keepdim=
           True)
        text_features /= text_features.norm(dim=-1, keepdim=True)

        # Calculate similarity
        similarity = (100.0 * image_features @ text_features.T).
           softmax(dim=-1)
        values, indices = similarity[0].topk(3)

    # Generate caption
    top_classes = [class_names[idx] for idx in indices]
    confidence = [val.item() for val in values]

    if confidence[0] > 0.5:  # High confidence
        caption = f"This␣is␣a␣{top_classes[0]}."
    else:  # Lower confidence
        caption = f"This␣might␣be␣a␣{top_classes[0]},␣but␣could␣
           also␣be␣a␣{top_classes[1]}␣or␣a␣{top_classes[2]}."

    return caption
```

A.3 Cluster Label Assignment

```
def assign_labels_to_clusters(clusters, true_labels, n_clusters):
    cluster_to_label = {}
    predicted_labels = np.zeros_like(true_labels)

    for cluster in range(n_clusters):
        cluster_indices = np.where(clusters == cluster)[0]
```

```
        if len(cluster_indices) > 0:
            # Get the most common true label in this cluster
            cluster_labels = true_labels[cluster_indices]
            unique_labels, counts = np.unique(cluster_labels,
                return_counts=True)
            majority_label = unique_labels[np.argmax(counts)]
            cluster_to_label[cluster] = majority_label
        else:
            # In case a cluster is empty, assign a random label
            cluster_to_label[cluster] = np.random.randint(0, 10)

    # Assign the majority label to all points in the cluster
    for i, cluster in enumerate(clusters):
        predicted_labels[i] = cluster_to_label[cluster]

    return predicted_labels, cluster_to_label
```