CS7641Team15Project

CS7641 Team15 Project

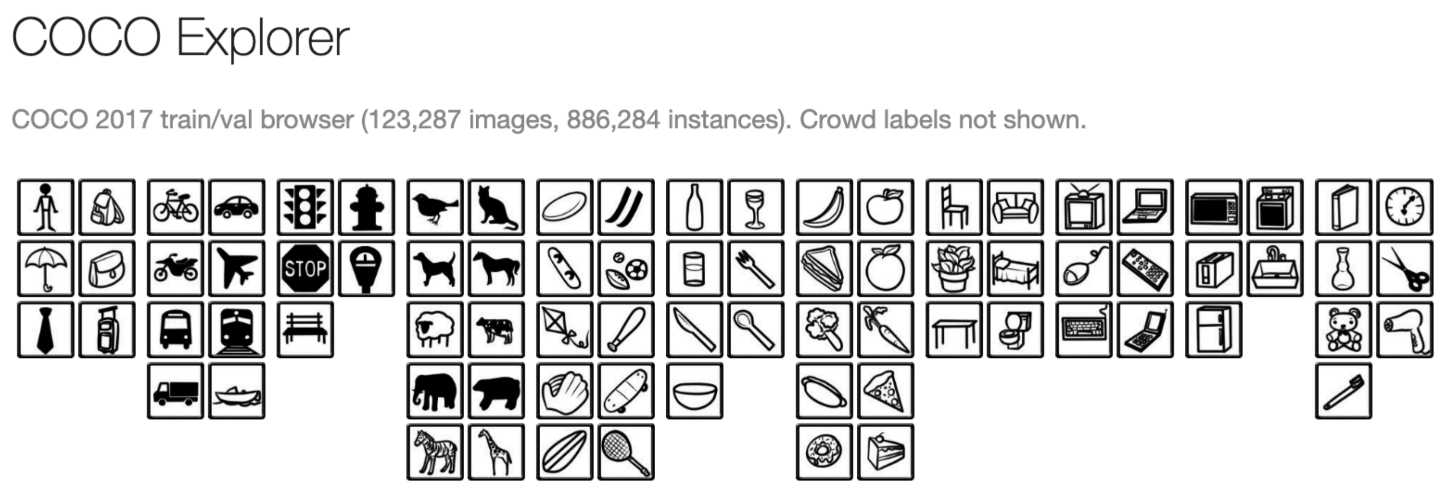
Members

- Aditya Bajoria
- Lakshh Khatri
- Gryphon Patlin
- Mukilan Karthikeyan
- Huzaifa Pardawala

Introduction/Background:

Image classification and object detection are central problems in computer vision, attracting significant research attention due to their wide range of applications, such as autonomous driving and surveillance. In this project, we aim to address these tasks using three different methods: supervised learning, unsupervised learning, and Generative Adversarial Networks (GANs). The Common Objects in Context (COCO) dataset will be used to evaluate these methods. The COCO dataset is one of the most widely used datasets in object detection and segmentation tasks. It contains 330,000 images, with over 200,000 labeled images, and 1.5 million object instances spread across 80 object categories. This large-scale dataset also includes 91 categories, object segmentation, and contextual recognition capabilities [1]. The dataset can be accessed here [1].

Dataset link: https://cocodataset.org/#explore


COCO Explorer
COCO 2017 train/val browser (123,287 images, 886,284 instances). Crowd labels not shown.
STOP

COCO Dataset object categories

Numerous studies have been conducted to compare the performance of supervised and unsupervised learning in the context of image classification. For example, CNNs (Convolutional Neural Networks) have been extensively used in detecting objects from annotated datasets [2]. Unsupervised methods such as k-means clustering and autoencoders have been employed for feature extraction without labels [3]. Additionally, we utilized the **Masked R-CNN framework** [4], which combines a region proposal network and segmentation capabilities to detect and localize objects with high precision [2]. Our implementation leverages **Detectron2** [8], a state-of-the-art library developed by Meta AI, fine-tuned for specific categories like "person" and "car."

Problem Definition:

The problem we aim to solve is object detection and image classification in complex environments where images contain multiple objects and noise. By comparing our three models we seek to determine the most effective multi-object classifier. Our motivation stems from the increasing need to develop more robust classifiers capable of recognizing objects in noisy environments. Our goal is to identify the most effective object detection method, particularly when noise and other objects coexist in the images.

Methods:

Data Processing:

We used the COCO dataset, which is approximately 18 GB in size. Given the vastness of the dataset and the nois in

1. Dataset Filtering: We refined the COCO dataset to focus only on images containing the classes “person,” “car,” and “person_and_car.” We then balanced this data with a proportional amount of images containing neither “person” nor “car” to give the model the ability to identify situations where both are not present.This allowed us to reduce the dataset size and focus our efforts on relevant categories.
2. Autoencoder for Dimensionality Reduction: An autoencoder model was used as a preprocessing step to reduce image complexity by lowering pixel dimensions. This helps improve the efficiency of subsequent clustering in the unsupervised learning stage.
3. Anchor Points for Visual Separation: In our implementation, anchor points were utilized to guide the Masked R-CNN model in efficiently identifying regions of interest. These points were strategically placed across the images, serving as starting locations for generating bounding boxes of varying scales and aspect ratios. By tailoring the anchor configurations to focus on the “person” and “car” categories, we ensured that the Region Proposal Network concentrated on relevant areas. This customization was crucial for handling the complex and diverse nature of the COCO dataset.
4. Data Augmentation Methods: Our initial attempts at developing a Convolutional Neural Network model to classify the data were unsuccessful due to rapid overfitting. The model would achieve a maximum of roughly 40% accuracy on the test set after under 100 iterations while achieving near 100% accuracy on the training set. To remedy this, we implemented many different methods to reduce overfitting. Firstly, we changed the images to gray scale, and added weight decay to prevent the model from relying too heavily on individual weights. We then added data augmentation methods, including random flips, random rotations, and color jittering.

Machine Learning Models:

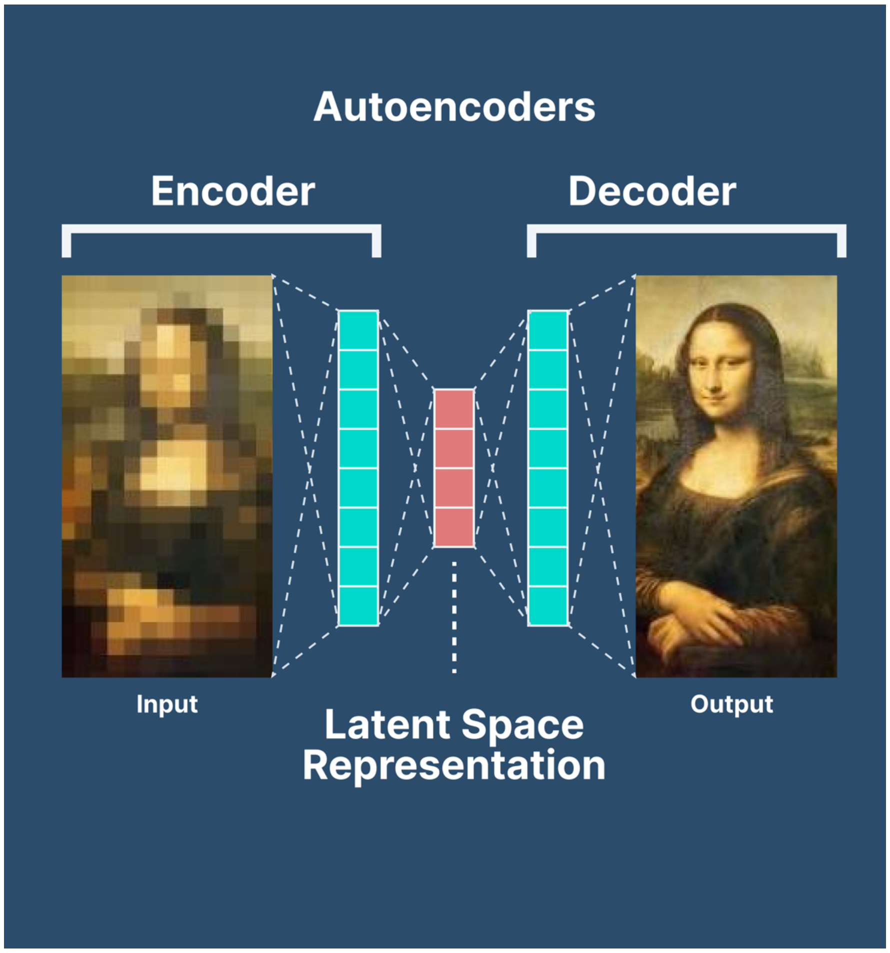
We implemented and compared three methods:

- **Supervised Learning:** We used Convolutional Neural Networks (CNNs) for object detection on annotated data. Our initial attempts at developing a Convolutional Neural Network model to classify the data were unsuccessful due to rapid overfitting. The model would achieve a maximum of roughly 40% accuracy on the test set after under 100 iterations while achieving near 100% accuracy on the training set. To remedy this, we implemented many different methods to reduce overfitting. Firstly, we changed the images to gray scale, and added weight decay to prevent the model from relying too heavily on individual weights. We then added data augmentation methods, including random flips, random rotations, and color jittering. Example before and after transformation images can be seen below

Data Augmentation

Unsupervised Learning: Using an autoencoder-based approach, we first reduce image dimensions. Then, we apply k-means clustering to detect and classify images into categories: person, car, both, or none. This serves as a baseline for comparison with other methods.

- **Unsupervised Learning:** We used k-means clustering and autoencoders. This was trained to detect patterns in the unlabeled data and served as a baseline to compare against supervised learning [3].



Autoencoder: A neural network designed to learn efficient encodings of data

- **Masked R-CNN** To streamline the process of training and evaluating an object detection model, we wrote a script to create a subset of the COCO dataset that exclusively includes images containing cars and people. This involved filtering the original dataset and recreating both the validation and training sets to only include the relevant images and annotations. To ensure that the model focuses solely on the target categories, we edited the annotations to remove masks and metadata for any other categories. This step was crucial to avoid confusing the model during training and evaluation. Next, we explored into using Detectron2, a state-of-the-art object detection library by Meta AI, and utilized its pre-trained Mask R-CNN model. This model provides a powerful foundation for instance segmentation and object detection tasks. We fine-tuned the Mask R-CNN model using our subset of the COCO dataset over 300 epochs. This fine-tuning allowed the model to specialize in detecting and segmenting people and cars, leveraging the masks and annotations curated specifically for these categories.

Detectron Architecture

Detectron Architecture: A state-of-the-art object detection framework by Meta AI

This pipeline demonstrates the power of combining curated datasets with robust machine learning architectures, resulting in a model capable of delivering accurate object detection and segmentation in real-world applications.

Results and Discussion:

CNN (Supervised)

Our initial attempts at developing a Convolutional Neural Network model to classify the data were unsuccessful due to rapid overfitting. The model would achieve a maximum of roughly

40% accuracy on the test set after under 100 iterations while achieving near 100% accuracy on the training set. To remedy this we used the data augmentation methods discussed above, including random flips, random rotations, and color jittering. This had the effect of slowing overfitting, but the long term results were still comparable:

Training vs Validation Accuracy (Before)

As pictured the validation data still struggled to pass 40% accuracy while the training accuracy continued to ascend.

In response to this, we adjusted our model architecture to incorporate more convolutional layers and added several drop-out layers with a 20% probability. This had a measurable improvement on training, leading to the following results over 500 epochs:

Training vs Validation Accuracy (After)

As shown above, the training accuracy over 500 epochs did not surpass 60% while the previous implementation achieved this accuracy after only 150 epochs. At the same time, the validation accuracy stuck much closer to the training accuracy, achieving a high of 48% on both the validation and the final test dataset.

K-Means (Unsupervised)

In our unsupervised learning approach, we first used the autoencoder to reduce the complexity of the images and then applied k-means clustering to the latent features extracted from the encoder.

The training process for our autoencoder model involved 50 epochs with a batch size of 16. The training process showed a consistent reduction in loss, indicating successful learning and convergence:

- Epoch 10: Loss = 0.0018
- Epoch 20: Loss = 0.0011
- Epoch 30: Loss = 0.0009

- Epoch 40: Loss = 0.0008
- Epoch 50: Loss = 0.0006

This steady decline in loss suggests that the autoencoder was effective in capturing the underlying structure of the data, thereby providing a robust feature representation for downstream tasks.

We then applied K-means clustering to these latent features, resulting in a clustering accuracy of 59.85%.

Annotations and Masks:

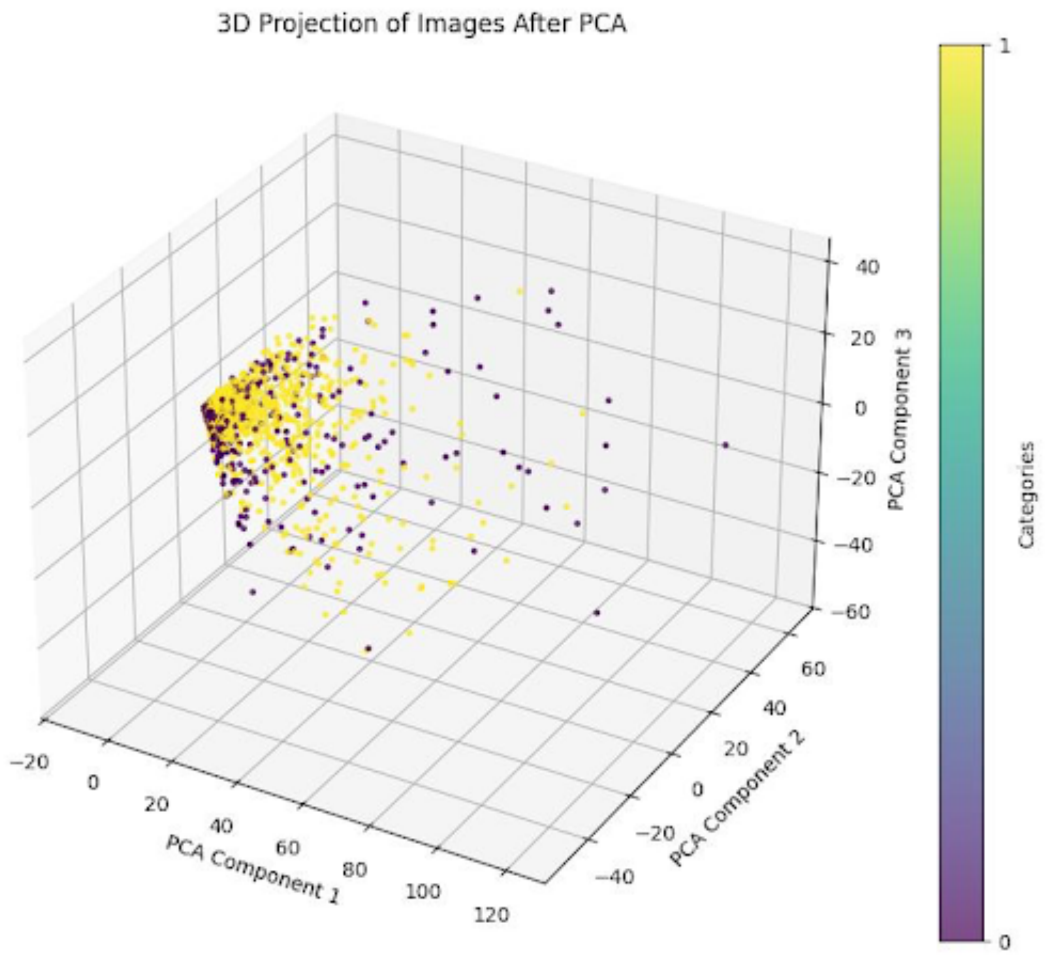
To improve our model performance, we utilized annotation masks to isolate our relevant categories (“person” and “car”) from their respective images. This allowed us to filter our extraneous information from these images, and helped ensure our models received cleaner inputs, reducing noise and model complexity. Below are examples of the processed images containing only the “car” and “person” from their images. Moving forward, in our third model, and for our final proposal, we would like to explore the same with the Segment Anything Model (SAM), and gauge its efficacy.

Sample processed image (car_only)

Sample processed image (person_only)

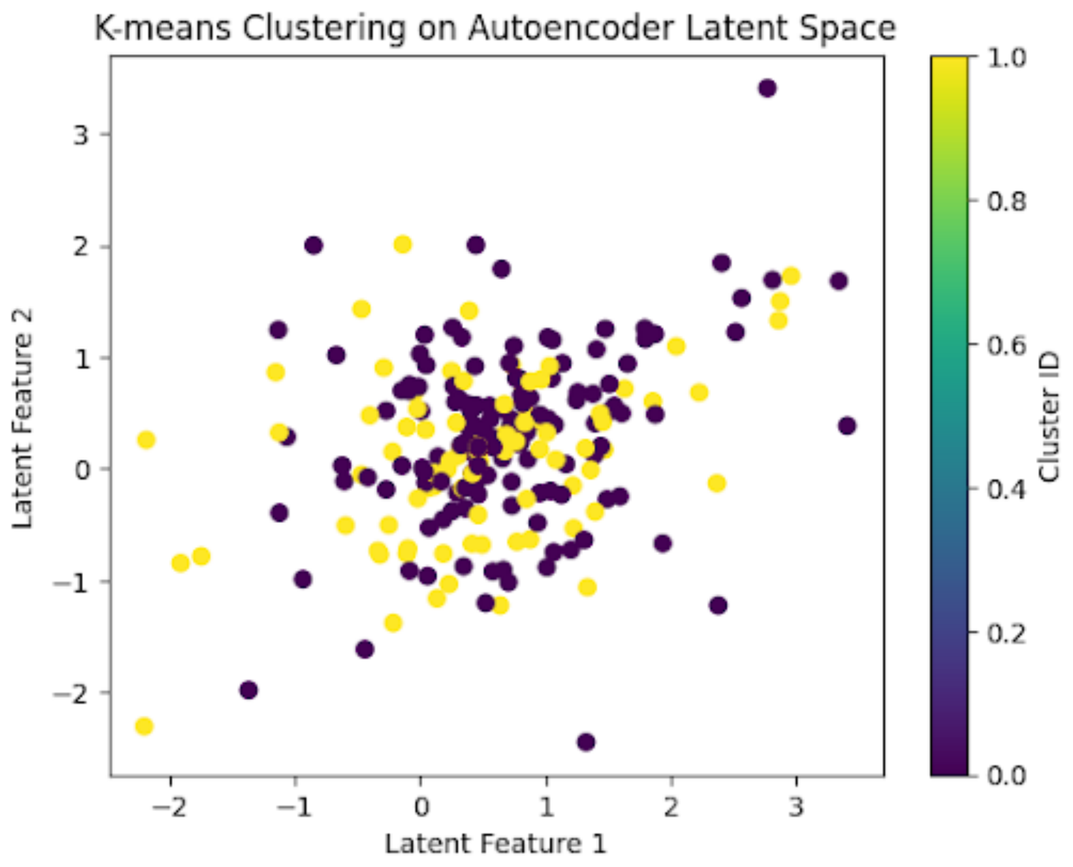
Dimensionality Reduction: PCA and AutoEncoders To reduce the dimensionality of the image data, we explored two methods:

1. Principal Component Analysis (PCA): PCA was applied to transform the masked images into lower-dimensional representations. While PCA effectively captured the primary variance, it did not exploit the nonlinear features present in the dataset, which was evident during clustering, as performance gains plateaued quickly.



PCA 3D Visualization

1. AutoEncoders: We trained an autoencoder to reduce the complexity of the images by learning and extracting the most relevant features. The autoencoder outperformed PCA by capturing more intricate and non-linear features, allowing us to express the image in terms of these selected features.



K-means with Autoencoder

Masked R-CNN:

Our model achieved an average recall of 65% and an average precision of 72%, showcasing its effectiveness in detecting people and cars in diverse scenarios. The Mask R-CNN model operates by placing anchor points across the image to propose regions of interest. These anchor points are used to generate bounding boxes, which are then refined to produce accurate predictions for both the bounding boxes and the corresponding categories (person or car). This process enables the model to deliver reliable detection and segmentation outputs for images in the validation and test sets.

An example output to detect a car

An example output to detect person and car in the same image.

Our primary goal was to determine which method provides the best performance across these metrics. We observed that the **Masked R-CNN** performed the best at the task out of the three models. Additionally, we took care to avoid biases in the training data that could have led to unfair or inaccurate predictions by the data preprocessing methods outlined above[7].

References:

[1] T.-Y. Lin, G. Patterson, M. R. Ronchi, Y. Cui, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, L. Zitnick, and P. Dollár, "COCO dataset," COCO Consortium, Available: https://cocodataset.org/#explore, 2015.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436–444, 2015.

[3] X. Tian, Y. Pan, W. Du, and M. Wang, "Unsupervised representation learning by predicting noise," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778.

[6] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," Advances in Neural Information Processing Systems, vol. 28, pp. 577-585, 2015.

[7] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in Proceedings of the 28th International Conference on Neural Information Processing Systems, 2014.

[8] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," Facebook AI Research, Available: https://github.com/facebookresearch/detectron2, 2019.

Gantt Chart

link to Gannt Chart

Contributions

Name	Proposal Contribution
Aditya Bajoria	Autoencoder, K-Means Algorithm, PCA Trial, Masked R-CNN
Huzaifa Pardawala	Autoencoder, Data collection, Data pre-processing, K-Means algorithm, Report write up
Mukilan Karthikeyan	Website, CNN data collection, CNN pre-processing
Gryphon Patlin	CNN data collection, CNN model implementation, CNN pre-processing
Lakshh Khatri	PCA Trial, Data pre-processing, Report write up, Video Recording

Video Proposal (This was for the proposal but we have kept it to see the progress we have made)

Link to our video

Github Link

https://github.com/MukilanKarthikeyan/CS7641Team15Project

Repository Structure:

Since this is a working project, we currently have two different branches:

- **Branch: Huzaifa-code**
- **Branch: gryphmuk-cnn**
- **Branch: data_annotation_separate**

Branch: huzaifa-code

This branch includes files and code primarily for data collection, unsupervised learning methods, and Masked R-CNN.

Directory and File Descriptions

```
data/
├── images.csv           # Contains 4 columns: person, ca, person_and_car, and c
├── images_collection.py # Contains code to generate images.csv

unsupervised/
├── k_means.ipynb    # Contains testing code for data collection, k-means trial
├── k_means_code.py  # Contains the actual code for performing k-means clusterin

Masked RCNN/
├── Mask_RCNN.ipynb          # Contains the code for the masked rcnn preprocessi
├── metircs.json             # Contains the evaluation metrics of our model
├── *                        # Contain the model weights and outputs

res/
└── *                # Contains images for GitHub Pages.
```

Branch: gryphmuk-cnn

This branch focuses on implementing and training a CNN model with the data collected.

Directory and File Descriptions

```
data/
├── collection.py    # Code to generate images.csv and load images.

models/
└── *                # Contains model checkpoints for the CNN model.

root/
├── cnn.py           # Contains code for the CNN class.
├── dataset.py       # Contains code to preprocess images.
├── train.py         # Code to train the CNN model on the dataset.


res/
└── *                # Contains images for GitHub Pages.
```

Branch: data_annotation_separate

This branch only conatins one change for the call to the COCO dataset API to implement the data augmentation techniques.

Directory and File Descriptions

```
root/
├── coco_api_dataset.py  # contains refined code for api calls
├── coco_dataset.py  # contains code for pre processing for the CNN
```

Each branch contains specific files tailored to the tasks within the project, and this README provides a high-level overview of the contents of each directory and file. Further details and instructions can be added as the project evolves.