In this assignment, I used the same dataset as project one. This dataset was scraped from CityScrape [1]. This is an image segmentation dataset, where each image can have multiple labels, i.e., one random image from a road can have sky, road, car, pedestrian, tree, traffic light, and so on. For this assignment, I have used Principal Component Analysis (PCA) [2] to find out if these labels can be clustered to find out if they have an underlying pattern. My code was done on Jupyter Notebook [3]. The code has been added at the end of this document and also on GitHub. I have used PyTorch [4], and SkLearn [5] to fit the data using PCA, and for plotting, I have used Matplotlib [6].

Purpose of this analysis:

My main goal was to find underlying patterns among the labels. Representing different parts of daily life, they should have an underlying pattern like the sky and road have a distinct difference; the sky is blue or white, whereas roads are primarily black or grey. This assumption has led to the inspiration for my analysis. Being a machine learning-based visualization, our project can be highly helped by this analysis. It can give us an idea that there is a pattern that Machine Learning models can learn. In the end, I found out that even if the dimension was reduced to 2, the labels could be somewhat clustered. That proved that machine learning models would learn those patterns with many more (500+) dimensions available.

Parameters and Options:

The main challenge behind using PCA on this dataset is the initial dimension. PCA reduces the dimension of a dataset by computing eigenvectors and eigenvalues. It reduces the dimensionality such that most of the information of all dimensions is stored. But the problem with the reduction is that the initial dataset needs to have the same dimension per label. But if we think about our dataset, the area of the sky varies from image to image; hence the dimension also varies. We need to make the dimension the same for PCA to work. To facilitate this work, we have explored three ways to do that, which are briefly explained below. Also, as the dataset is huge with a high data dimension, I have used item filtering to make the visualization more straightforward. I have extracted **200** random images from the dataset and taken only **four** labels (car, sky, person, traffic light). As the data is reduced to 2 dimensions, even if they have a pattern, they tend to be clustered together. To show the visualization clearly, we have added noise for each label. I have provided both of the visualizations below to indicate if that noise has a positive impact on visualization.

1. Truncation

The first option we have tried is the most simple one. Truncate each label's dimensions to the minimum one for that label. For example, if image 1 has an area of 126 and image 2 has an area of 64 for the sky, I have made both 64 by truncating. To be clear, this truncation is done on per label basis to keep most of the information. So the minimum dimension for the sky won't affect the minimum dimension of the road. Later, PCA is used on the reduced dimension matrix to find two principal components for visualization. For the truncation, I have explored both truncation at start and truncation at end.

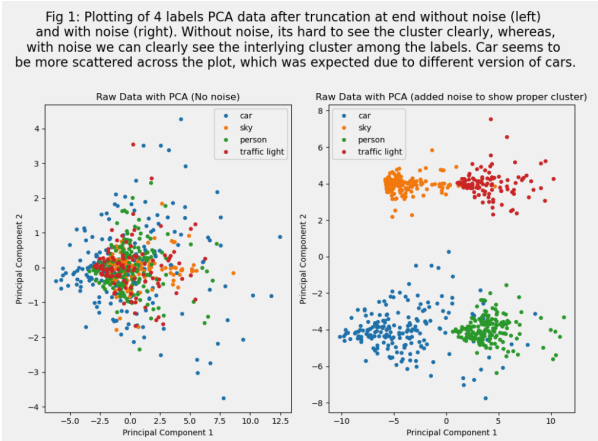


Fig 1: Plotting of 4 labels PCA data after truncation at end without noise (left) and with noise (right). Without noise, its hard to see the cluster clearly, whereas, with noise we can clearly see the interlying cluster among the labels. Car seems to be more scattered across the plot, which was expected due to different version of cars.

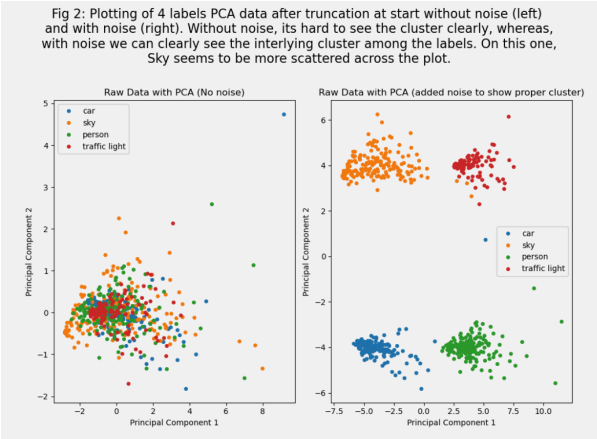


Fig 2: Plotting of 4 labels PCA data after truncation at start without noise (left) and with noise (right). Without noise, its hard to see the cluster clearly, whereas, with noise we can clearly see the interlying cluster among the labels. On this one, Sky seems to be more scattered across the plot.

From both of the visualizations, we can see that truncation at start helps cluster more than the other.

2. Padding

Instead of truncation, I have also used padding to keep not most but all of the information for each label. For this method, I have used zero padding to make all the dimensions to same size. For example, if image 1 has an area of 126 and image 2 has an area of 64 for the sky, I have made them 126 by padding. Later PCA is used on this updated dimension to reduce the dimension to 2 for visualization. For the padding also, I have explored both at start and at end.

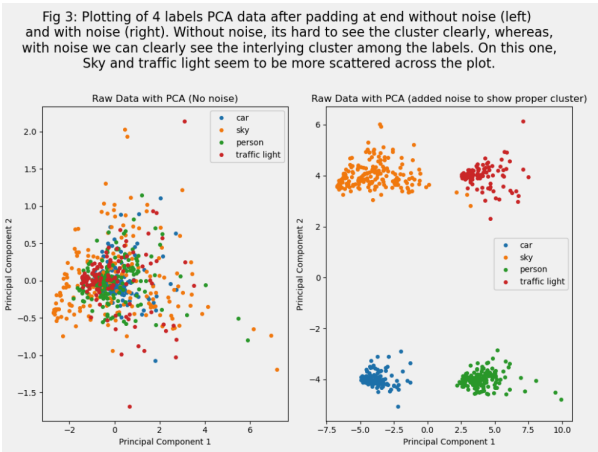


Fig 3: Plotting of 4 labels PCA data after padding at end without noise (left) and with noise (right). Without noise, its hard to see the cluster clearly, whereas, with noise we can clearly see the interlying cluster among the labels. On this one, Sky and traffic light seem to be more scattered across the plot.

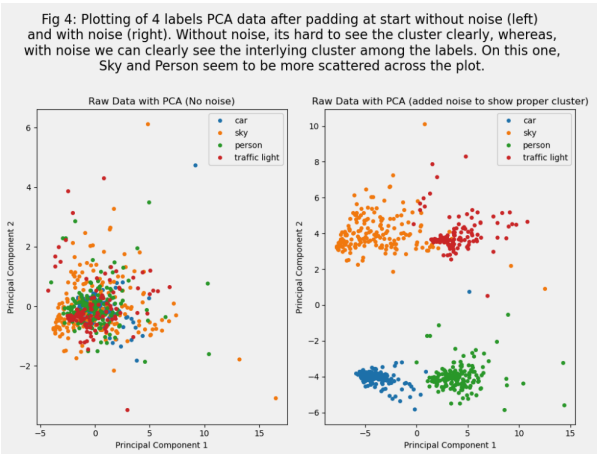


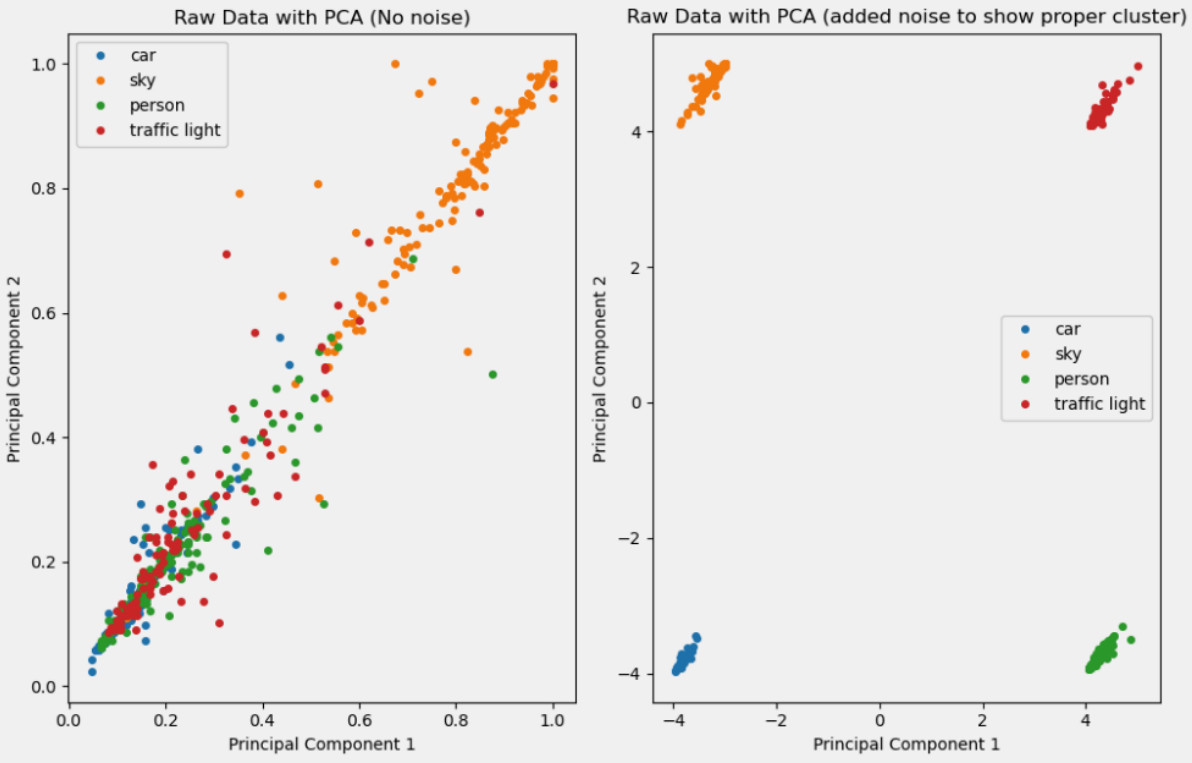
Fig 4: Plotting of 4 labels PCA data after padding at start without noise (left) and with noise (right). Without noise, its hard to see the cluster clearly, whereas, with noise we can clearly see the interlying cluster among the labels. On this one, Sky and Person seem to be more scattered across the plot.

From both of the visualizations, we can see they mainly performed similarly. It was expected as the data is not truncated on any of them, and the data distribution would be mostly the same actually, with some extra zeros on either side. PCA is intelligent enough to disregard the zero padding.

3. Feature Extraction

The last and more complex method I tried was to use a simple feature extractor. I have used *SelectKBest* [7] from sklearn to find the best *K* features from the dataset based on their label. This method worked in three steps: (1) the Whole dataset was zero-padded (at the end) to the same size, (2) Using a feature extractor, extracted *K* features from the dataset that is most crucial based on their label, (3) PCA was used on that extracted dataset to reduce the dimension to 2 for visualization. Unlike before, step 1 was done on the whole dataset instead of per label basis. The value of *K* (**141**) was determined by the minimum dimension across all labels and all images.

Fig 5: Plotting of 4 labels PCA data after feature extraction without noise (left) and with noise (right). Even without noise, we can observe a good view of the clusters. And with noise, they are perfectly clustered.



From the visualization, we can see they performed better than all the other visualizations before. Even without the noise, we can see the clustering pattern. It was expected as a feature extractor has been used to find out the most crucial features first, so PCA has to work on fewer dimensions than before, which made the clustering better.

Difficulties:

The main difficulty I faced on this assignment was the initial dimension. Being a complex dataset, using PCA was a challenging task. Also, another difficulty I faced was plotting the

scatterplot; as there were many overlaps between points, I had to add noise to make the clustering clear.

Limitations:
The main limitation of this work is the filtering of labels. Because of this filtration, we have lost the capability to see the other clusters. One way to solve that was to use details-on-demand or overview/detail to show only the user-selected labels. This work was left as a future scope of this work.

Integration to the group project:
This analysis facilitates the motivation of our group project to use a machine learning model to get the segmentation result. Also, when we are finally done with our group project if some of the labels are less accurate than the others, we can use this analysis to find out if the data distribution is the cause or if there are issues elsewhere.

References:

[1] “Cityscapes Dataset – Semantic Understanding of Urban Street Scenes,” Oct. 17, 2020. https://www.cityscapes-dataset.com/ (accessed Apr. 02, 2023).

[2] “sklearn.decomposition.PCA,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.decomposition.PCA.html (accessed Apr. 04, 2023).

[3] “Project Jupyter.” https://jupyter.org (accessed Apr. 04, 2023).

[4] “PyTorch.” https://www.pytorch.org (accessed Apr. 04, 2023).

[5] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *MACHINE LEARNING IN PYTHON*.

[6] “Matplotlib — Visualization with Python.” https://matplotlib.org/ (accessed Apr. 04, 2023).

[7] “sklearn.feature_selection.SelectKBest,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.feature_selection.SelectKBest.html (accessed Apr. 04, 2023).