An Empirical Study on Improving Loop Closure Detection through Autoencoder-based Dimension Reduction and PCA Integration

Deep Learning Assignment

Presented by **Neural Geeks**

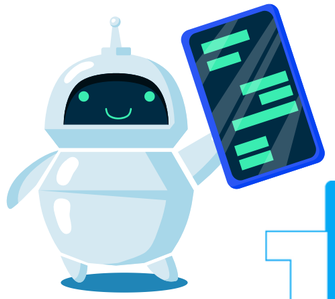
Sanjay Chhaba
200106058

Hima Varshitha
200104070

Sarthak Ray
200106059

Dakoori Kavya
200104029

Srijanya Durganala
200104034



1 Introduction

2 Abstract

3 Related Work

4 Proposed Work

5 Conclusion

SLAM

LCD

PCA

AUTOENCODER

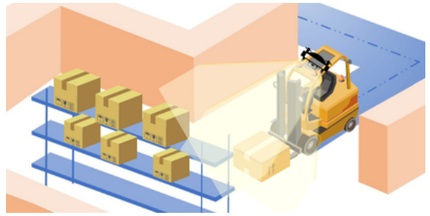
SLAM

Simultaneous Localization and Mapping

In SLAM a mobile robot can simultaneously perform its position estimation and map building tasks in an unknown environment.

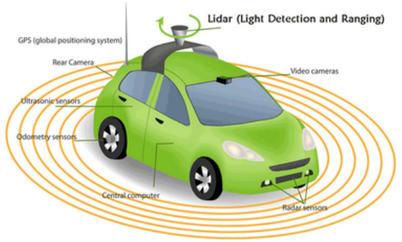
Visual Based

This method primarily relies on cameras or other vision sensors to perceive the environment.



LiDAR Based

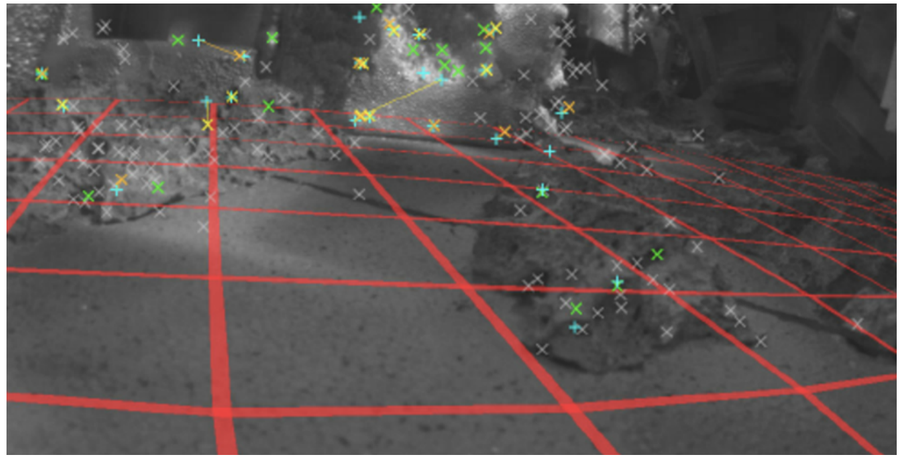
LiDAR (Light Detection and Ranging) sensors emit laser pulses and measure the time it takes for the pulses to return.



Key Points

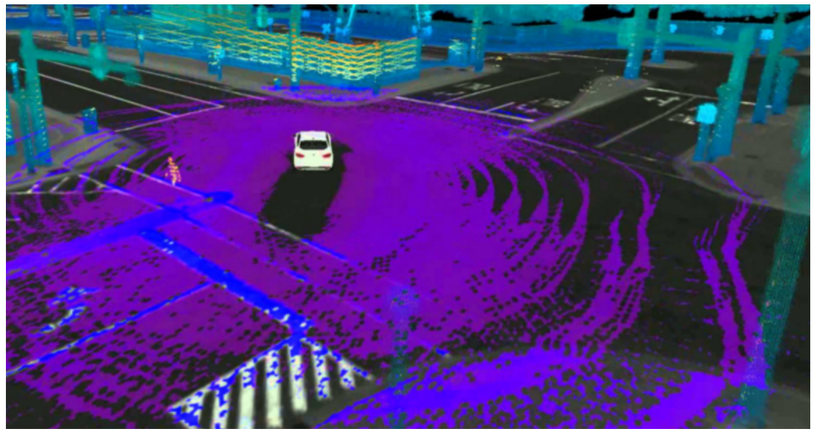
Visual SLAM

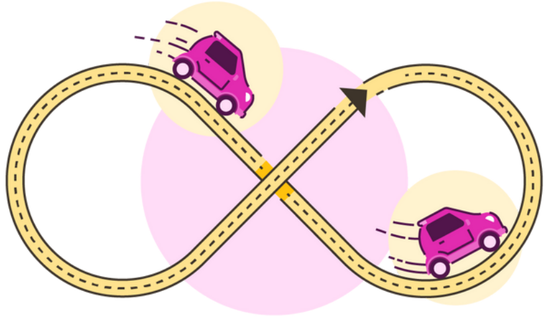
- Cost-effective,Works good in well-lit and structured environment
- Vulnerable to lighting changes
- Depth estimation may be challenging



LIDAR SLAM

- Uses 3D point cloud data
- Works well in various lighting conditions, including darkness
- Can be expensive,and affected by weather conditions, such as rain or fog





LCD

Loop CLosure Detection

Loop closure detection is the process of detecting whether an agent has returned to a previously visited location.

Feature Extractor

Extract distinctive and robust features from the sensor data. Techniques include:

- SIFT (Scale-Invariant Feature Transform)
- ORB (Oriented FAST and Rotated BRIEF)
- SURF (Speeded-Up Robust Features)

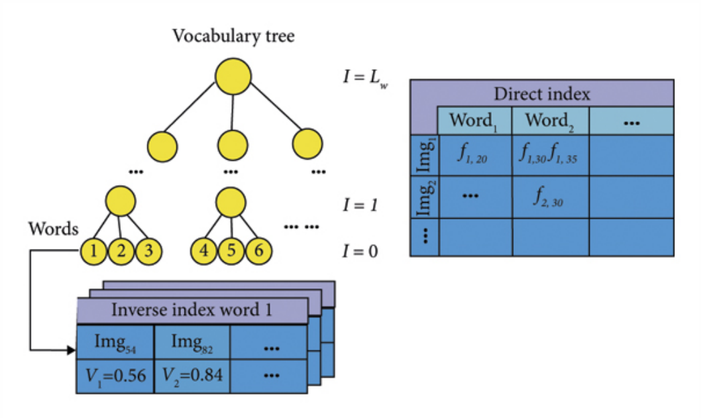
Feature Matching

Find correspondences between current and database features. Methods include:

- nearest-neighbor search
- geometric verification
- RANSAC (Random Sample Consensus)

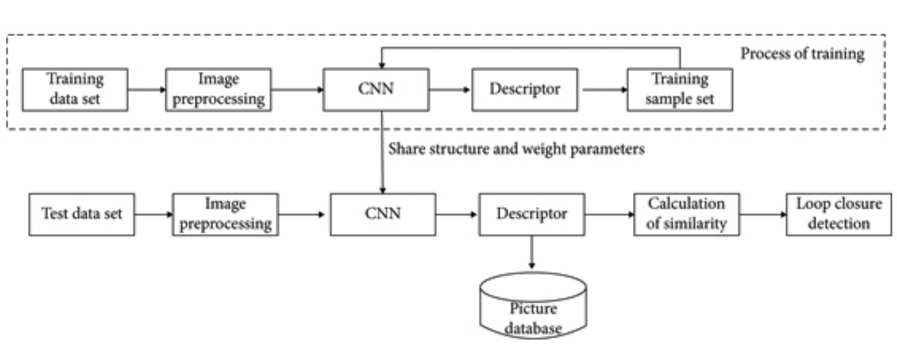
Loop closure decision

If enough correspondences meet a certain threshold a loop closure will be declared.



Abstract

Deep Learning-based Loop Closure Detection for Visual SLAM



State of the Art

- Traditional feature-based methods like bag-of-words and vocabulary trees are used for Visual SLAM.
- Herein, the model simply extracts features from the input images and matches them with the existing “bag” of features.

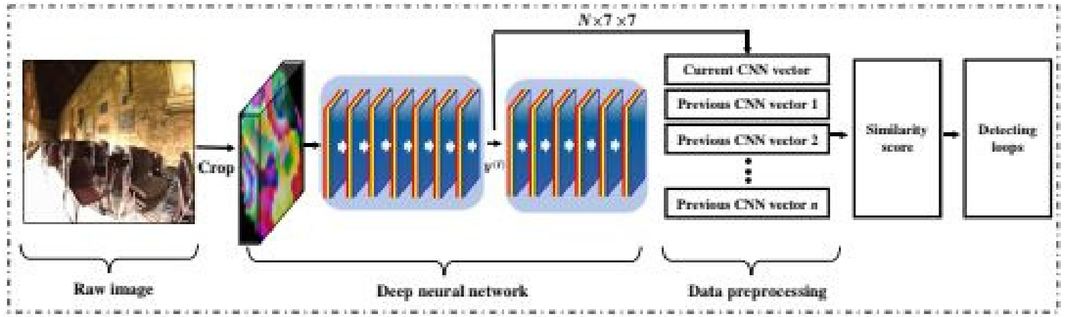
Research and Development

- Deep learning methods like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are being increasingly sought after to replace traditional methods.
- These networks can capture intricate patterns and relationships without relying on predetermined features.

Related Work

Fast and robust loop-closure detection using deep neural networks and matrix transformation for a visual SLAM system

Journal of Electronic Imaging, Nov/Dec 2022



Principal

The key innovation lies in combining a pre-trained CNN model with a convolutional autoencoder to obtain compact, low-dimensional image representations thereby improving the performance.

Overcame the Challenge

This approach significantly reduces computation time and memory usage while maintaining high detection accuracy by addressing Perceptual Aliasing.

Scope

There is room for further research, particularly in understanding the general principles governing the resolution of perceptual aliasing and improving the algorithm's ability to detect missed loops.

Deep Reinforcement Learning Based Loop Closure Detection

Journal of Intelligent & Robotic Systems, Oct 2022

Principal

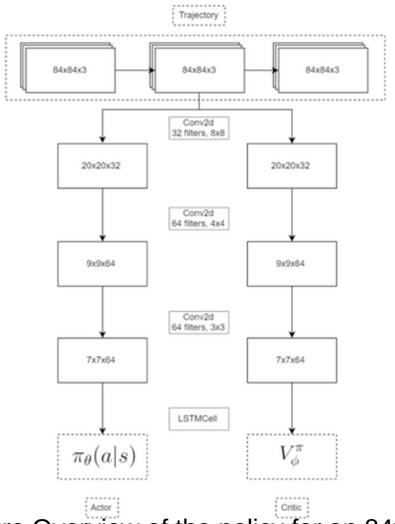
This offers a solution that uses a deep RL algorithm that interacts with a simulated grid environment for the loop closure problem. It uses entropy maximization for selecting a batch size to demonstrate the algorithm's training improvement and accuracy.

Overcame the Challenge

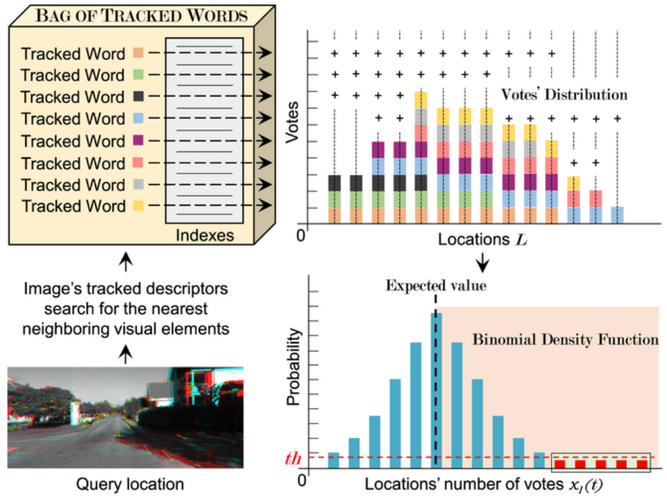
The approach reduces computational complexity, achieves high precision, and addresses an entropy issue affecting trajectory sampling, enhancing training efficiency.

Scope

Future directions include extending the approach to 3D motion, testing in outdoor environments, and improving the transition from simulation to full automation.



Architecture Overview of the policy for an 84x84x3 resolution



Modest-vocabulary loop-closure detection with incremental bag of tracked words

Robotics and Autonomous Systems ,July 2021

Principal

Introduces a more efficient and memory-friendly loop-closure detection system for autonomous robots by representing the environment with a compact visual vocabulary, using probabilistic methods, leveraging temporal information, and incorporating geometrical verification.

Overcame the Challenge

The proposed approach reduces computational demands by using a compact visual vocabulary and efficient matching techniques.

Scope

Future work could focus on semantic integration, dynamic environment handling, and integration with SLAM for a more comprehensive mapping and localization system.

A Semantic-Based Loop Closure Detection of 3D Point Cloud

International Conference on Robotics and Biomimetics, Dec 2021

Principal

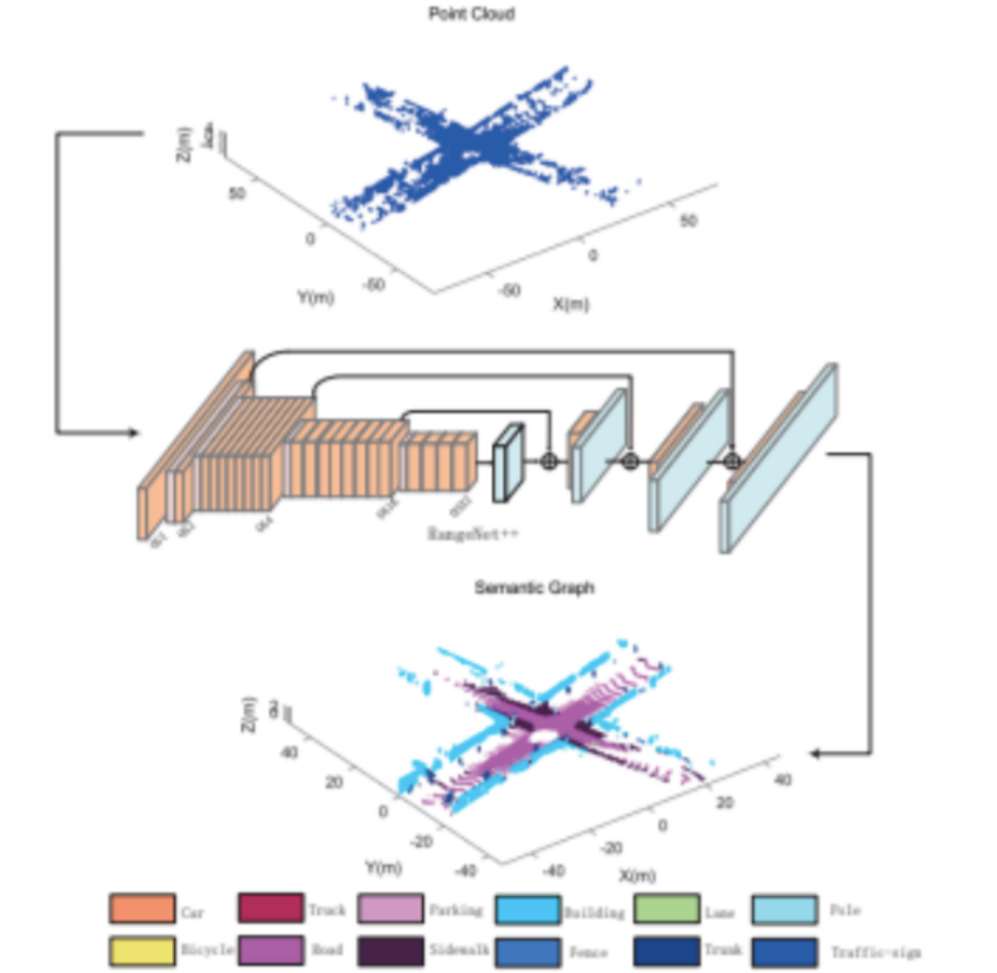
- A **semantic-based LCD** method, using **PCA algorithm**, and the **local column shift** applied to reduce the influence of noise points.
- **Similarity calculation** is performed on the semantic images.

Overcame the Challenge

- The algorithm uses semantic objects and their topological relationships to understand the scene, recognize surrounding environment like a human. The use of semantic objects **overcomes the natural defects** of point cloud data.

Scope

- Apply TF-IDF on the **bag-of-words** algorithm of the visual LCD to the point cloud semantic loop closure detection work to **obtain weights** for semantic objects.



Proposed Work

Base Model

The Siamese CNN

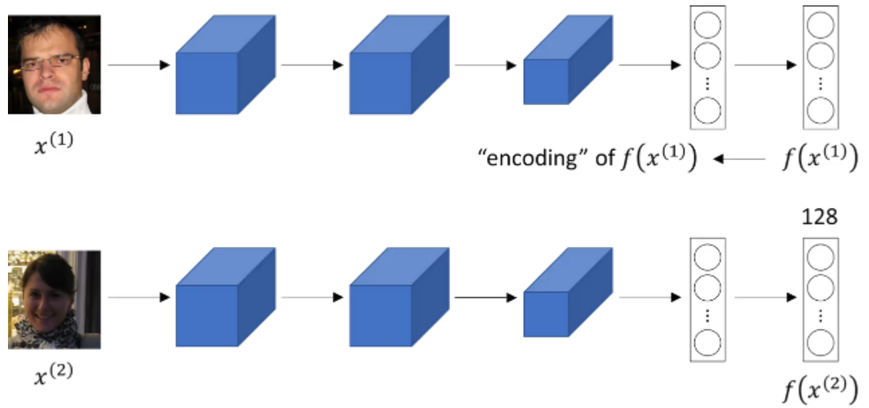
A Siamese Convolutional Neural Network (Siamese CNN) is a neural network architecture that consists of two identical subnetworks (twins) sharing weights. It's designed for similarity-based tasks like face recognition. By learning to distinguish between similar and dissimilar pairs, it excels at one-shot learning and binary classification. It's commonly used to determine whether two input samples, typically images, are the same or different, making it valuable for various recognition and matching tasks.

APPLICATION

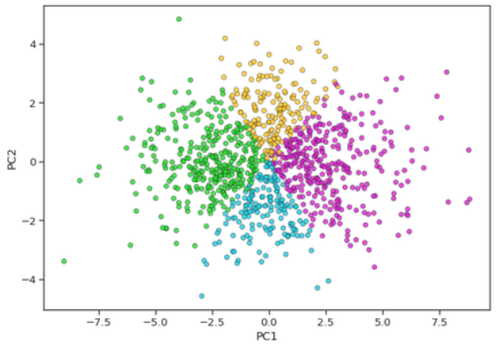
Face Recognition: Siamese CNNs can verify whether two facial images belong to the same person, making them valuable for biometric authentication.

Signature Verification: They are used to verify the authenticity of handwritten signatures, ensuring document security.

Object Tracking: Siamese networks can track objects by comparing their features in consecutive frames of a video.



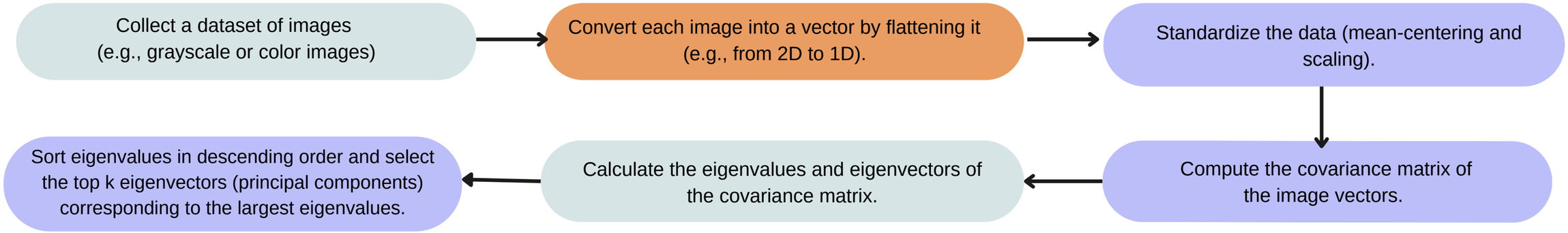
PCA



INTRODUCTION

Principal Component Analysis (PCA) is a dimensionality reduction technique used to transform a high-dimensional dataset into a lower-dimensional subspace while retaining as much of the original data's variance as possible. It finds applications in image compression, feature extraction, and noise reduction, contributing to various computer vision tasks.

HOW PCA WORKS



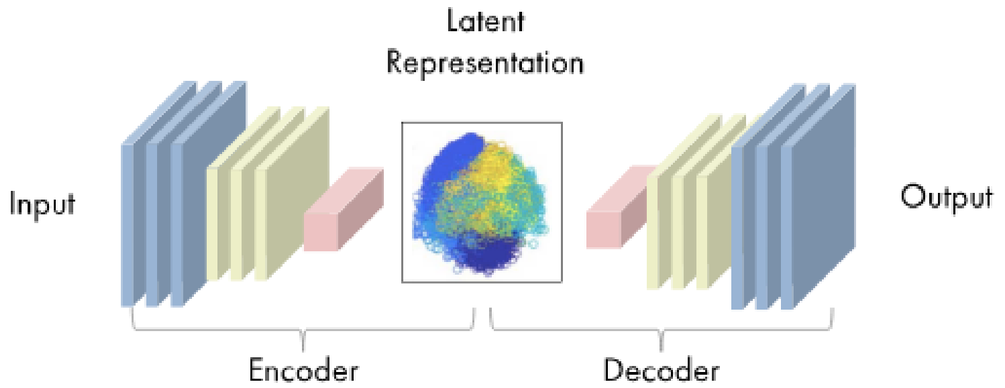
APPLICATION

Image Compression: Represent images using fewer principal components, reducing storage and transmission requirements.

Feature Extraction: Extract meaningful image features for tasks like facial recognition, object detection, and image classification.

Noise Reduction: Remove noise and enhance signal-to-noise ratio in images.

AUTO ENCODER



INTRODUCTION

Autoencoder is a neural network designed to learn an identity function in an unsupervised way to reconstruct the original input while compressing the data in the process so as to discover a more efficient and compressed representation

HOW AUTOENCODER WORKS

Feed an image through the encoder to obtain an encoding.

Use the encoding to reconstruct the image with the decoder.

Compare the reconstructed image to the original (e.g., using mean squared error).

Adjust the encoder and decoder weights to minimize the reconstruction error.

APPLICATION

Eliminating Complexity : Autoencoders capture essential image features in a lower-dimensional space.

Anamoly Detection : Use the measured loss between the input and the reconstructed output. If error is too high autoencoder detects anomaly.

Unsupervised Learning: Problems By reducing your data down to a very small subset (i.e., 2 or 3 output neurons), and train the model to learn how the data can be effectively split into sub-classes.

LOAD THE DATASET

Training dataset

→ New College Dataset

Testing dataset

→ City College Dataset

```
print(f"The New College dataset has {len(newcollege)} images.")
print(f"The City Centre dataset has {len(citycentre)} images.")
```

```
The New College dataset has 2146 images.
The City Centre dataset has 2474 images.
```

IMAGE AUGUMENTATION FOR AE

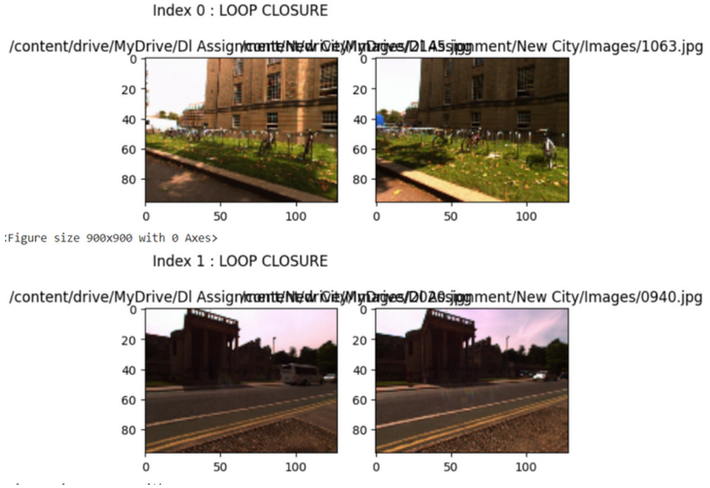
- **preprocess_image:** This function loads the specified file as a JPEG image and converts it to float32, corrects the fish-eye distortion, and resizes it to the target shape to work with it
- **unison_shuffle_triple:** Shuffles three arrays while preserving the indices
- **undistort_fish_eye:** Gets rid of the fish eye camera distortion
- **four_point_transform:** Applies a 4-point transformation to an image
- **augment:** Applies brightness change and necessary augmentations to the image

CREATING TRAINING DATASET

CreateInputOutput():

Created a large input dataset combining similar and dissimilar pairs, as well as corresponding outputs. Four empty lists are initialized: data_n1, data_n2, data_p1, and data_p2. These lists will be used to store indices of data pairs that are considered "non-closures" and "closures".

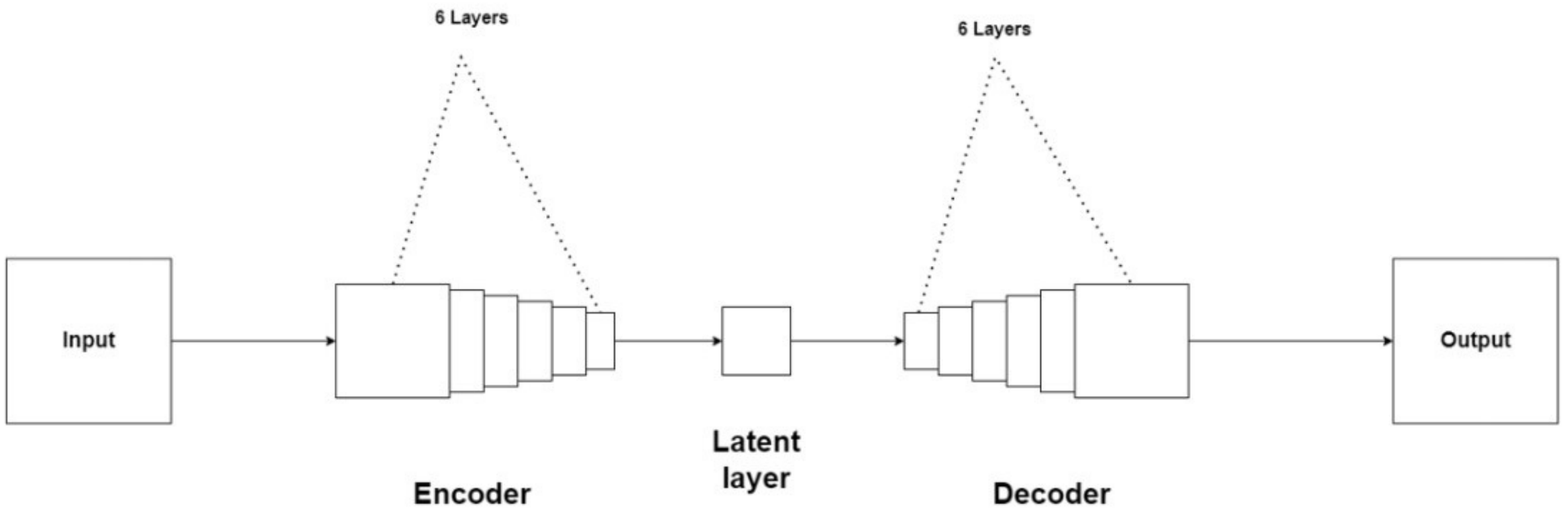
Assemble the train and test datasets from City Centre and New College respectively:
data1 - image 1 from the pair.
data2 - image 2 from the pair.
out - closure or non-closure.
train/val - training and validation sets respectively.



visualize:

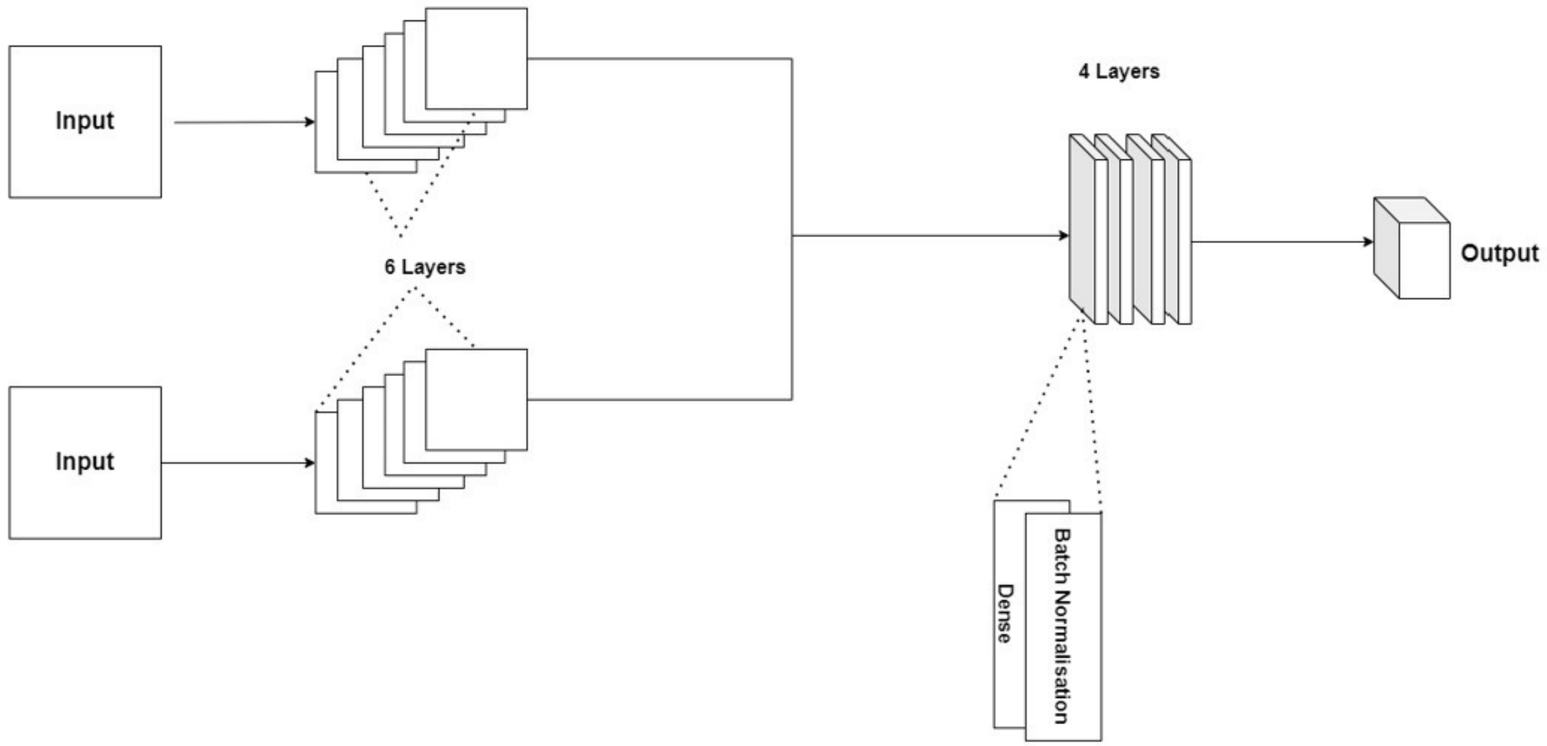
The visualize function is used to display pairs of images from given datasets and their corresponding verdict (LOOP CLOSURE or NOT LOOP CLOSURE). It first determines the images and verdict based on the input, and then it displays them using matplotlib. This function is used to visualize sample image pairs from two datasets (citycentre and newcollege) in two separate loops.

AUTOENCODER



Autoencoder

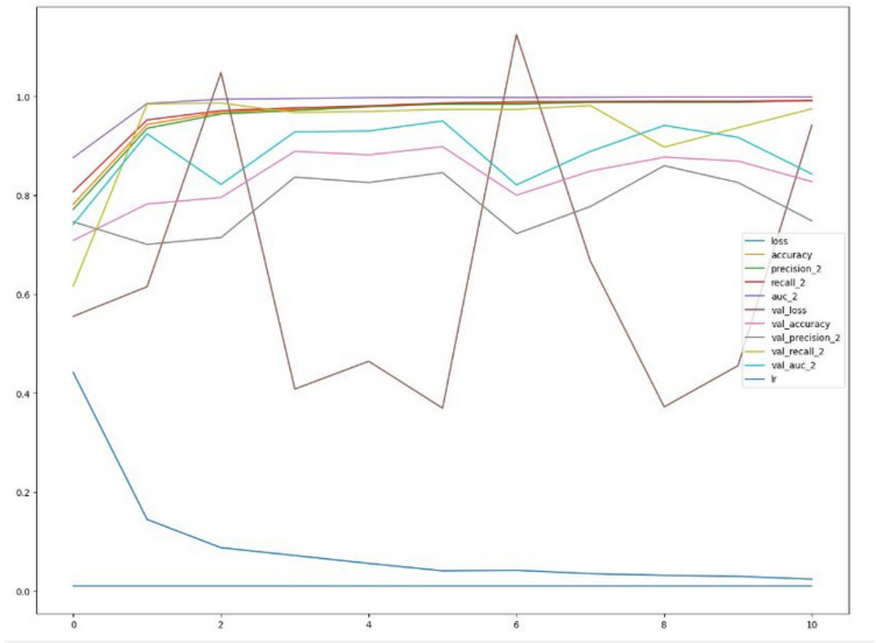
BASE MODEL : Siamese Convolutional Network



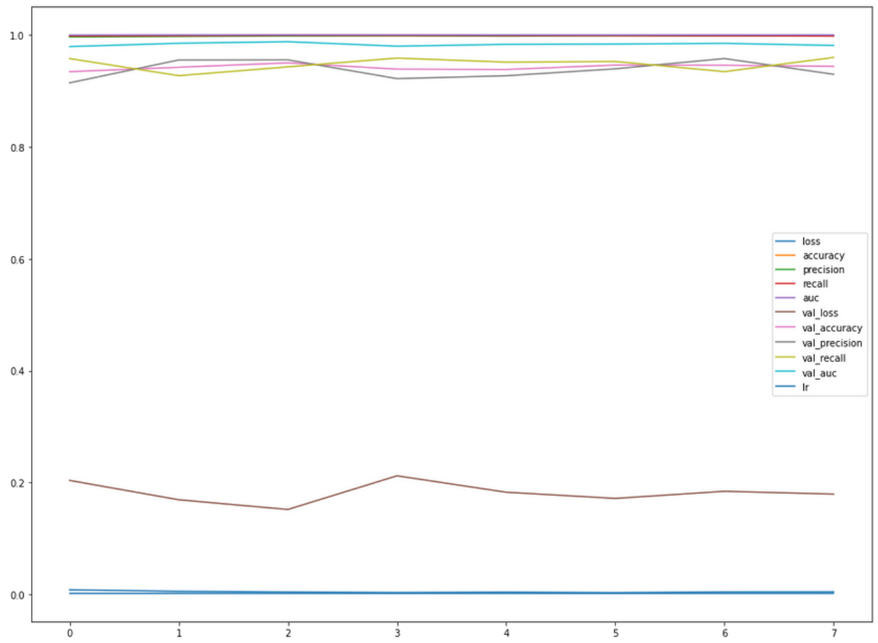
Siamese Convolutional Network

RESULT AND COMPARISION

Without Autoencoder



With Autoencoder

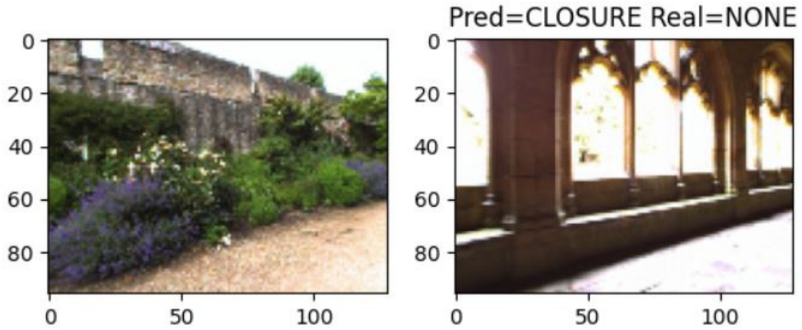


Observe how the training process is much smoother (less noise) after employing autoencdoer.

RESULT AND COMPARISION

Without Autoencoder

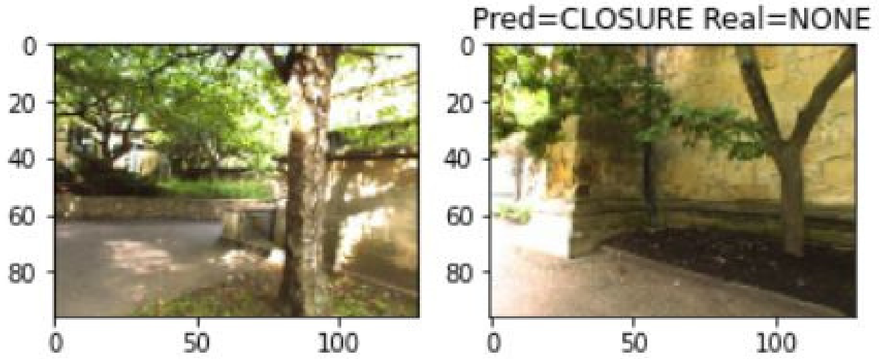
```
  0%|          | 0/1 [00:00<?, ?it/s]4/4 [==============================] - 1s 91ms/step
100%|██████████| 1/1 [00:01<00:00,  1.66s/it]
Total = 100     Mispredicted = 7
Accuracy = 0.9299999999999999
<Figure size 1600x1600 with 0 Axes>
```




```
<Figure size 1600x1600 with 0 Axes>
```

With Autoencoder

```
100%|██████████| 1/1 [00:00<00:00,  4.45it/s]
Total = 100     Mispredicted = 4
Accuracy = 0.96
<Figure size 1152x1152 with 0 Axes>
```

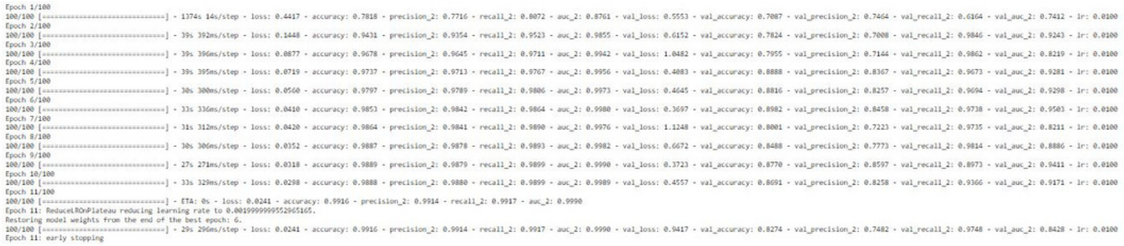



```
<Figure size 1152x1152 with 0 Axes>
```

Improvement of accuracy from 92.9% to 96%.

RESULT AND COMPARISION

Without Autoencoder



With Autoencoder

```
Epoch 1/100
100/100 [==============================] - 39s 300ms/step - loss: 0.5494 - accuracy: 0.7193 - precision: 0.7172 - recall: 0.72
26 - auc: 0.7812 - val_loss: 0.3307 - val_accuracy: 0.8605 - val_precision: 0.8291 - val_recall: 0.9063 - val_auc: 0.9326
Epoch 2/100
100/100 [==============================] - 29s 288ms/step - loss: 0.1114 - accuracy: 0.9600 - precision: 0.9542 - recall: 0.96
62 - auc: 0.9905 - val_loss: 0.4458 - val_accuracy: 0.8615 - val_precision: 0.8570 - val_recall: 0.8675 - val_auc: 0.8844
Epoch 3/100
100/100 [==============================] - 29s 289ms/step - loss: 0.0656 - accuracy: 0.9760 - precision: 0.9730 - recall: 0.97
91 - auc: 0.9968 - val_loss: 0.4302 - val_accuracy: 0.8805 - val_precision: 0.8231 - val_recall: 0.9685 - val_auc: 0.9371
Epoch 4/100
100/100 [==============================] - 31s 309ms/step - loss: 0.0515 - accuracy: 0.9812 - precision: 0.9806 - recall: 0.98
13 - auc: 0.9978 - val_loss: 0.9082 - val_accuracy: 0.8786 - val_precision: 0.8305 - val_recall: 0.9471 - val_auc: 0.9217
Epoch 5/100
100/100 [==============================] - 34s 346ms/step - loss: 0.0532 - accuracy: 0.9816 - precision: 0.9814 - recall: 0.98
18 - auc: 0.9973 - val_loss: 0.4362 - val_accuracy: 0.8876 - val_precision: 0.8312 - val_recall: 0.9710 - val_auc: 0.9386
Epoch 6/100
100/100 [==============================] - 39s 391ms/step - loss: 0.0425 - accuracy: 0.9855 - precision: 0.9844 - recall: 0.98
67 - auc: 0.9984 - val_loss: 0.2307 - val_accuracy: 0.9287 - val_precision: 0.9229 - val_recall: 0.9356 - val_auc: 0.9725
Epoch 7/100
100/100 [==============================] - 29s 291ms/step - loss: 0.0328 - accuracy: 0.9899 - precision: 0.9876 - recall: 0.99
21 - auc: 0.9980 - val_loss: 0.4056 - val_accuracy: 0.9323 - val_precision: 0.9500 - val_recall: 0.9145 - val_auc: 0.9773
Epoch 8/100
100/100 [==============================] - 34s 342ms/step - loss: 0.0395 - accuracy: 0.9868 - precision: 0.9829 - recall: 0.99
06 - auc: 0.9976 - val_loss: 0.4118 - val_accuracy: 0.8925 - val_precision: 0.8447 - val_recall: 0.9657 - val_auc: 0.9341
Epoch 9/100
100/100 [==============================] - 42s 427ms/step - loss: 0.0335 - accuracy: 0.9876 - precision: 0.9869 - recall: 0.98
83 - auc: 0.9989 - val_loss: 0.4900 - val_accuracy: 0.8717 - val_precision: 0.8139 - val_recall: 0.9656 - val_auc: 0.9298
Epoch 10/100
100/100 [==============================] - 29s 287ms/step - loss: 0.0274 - accuracy: 0.9915 - precision: 0.9921 - recall: 0.99
10 - auc: 0.9987 - val_loss: 0.8033 - val_accuracy: 0.8206 - val_precision: 0.7366 - val_recall: 0.9940 - val_auc: 0.8753
Epoch 11/100
100/100 [==============================] - 34s 337ms/step - loss: 0.0209 - accuracy: 0.9934 - precision: 0.9926 - recall: 0.99
38 - auc: 0.9996 - val_loss: 0.2971 - val_accuracy: 0.9014 - val_precision: 0.8577 - val_recall: 0.9597 - val_auc: 0.9587

Epoch 00011: ReduceLROnPlateau reducing learning rate to 0.0019999999552965165.
Restoring model weights from the end of the best epoch.
Epoch 00011: early stopping
```

CONCLUSION

1. **Autoencoder:** Learns essential image features and creates a lower-dimensional representation for pretraining future tasks.
2. **Siamese CNN:** Designed for similarity metric learning using shared branches for feature extraction.
3. **Synergy:** Combines autoencoder's feature learning with Siamese CNN for improved feature quality and similarity assessments.
4. **Loop Closure Detection:** Siamese CNN predicts loop closures efficiently, enhancing accuracy in robotics and computer vision.

Thank You!