SIFT And CNN in Objects Classcification

COMP61342&41342 Cognitive Robotics And Computer Vision Assignment Report

Yuyi Yang
yuyi.yang@postgrad.manchester.ac.uk
p68393yy

***Abstract*—This project aims to explore the use of SIFT (Scale-Invariant Feature Transform) and CNN (Convolutional Neural Network) in object classification tasks. By implementing different approaches on different datasets, we aim to compare their performance, and understand their strengths and weaknesses of in the context of object recognition.**

***Index Terms*—Local Feature, CNN, Object Classification**

I. Introduction

In recent years, equipping robots with reliable perception capabilities has become a foundation for autonomous manipulation, navigation, and human–robot interaction. Object recognition—identifying and localizing objects from camera data—remains especially challenging in unstructured, dynamic environments where lighting, occlusion, and viewpoint changes can drastically reduce accuracy.

Traditional computer-vision pipelines, based on handcrafted local features (i.e. SIFT) and classifiers (e.g. SVM, K-NN), offer interpretability and low inference latency; however, they may falter when faced with complex environment and large intra-class variability. Conversely, convolutional neural networks (CNNs) have revolutionized vision performance, automatically learning hierarchical features, but at the cost of greater computational demand and complex hyperparameter tuning.

This work systematically compares these two paradigms on the CIFAR-10 [1] and iCubWorld 1.0 [2] benchmark. This project researches how choice of feature extractor, classifier, network architecture, and training strategy affect performance of classification accuracy and inference speed, and model robustness to environmental changes.

The rest of the report is organised as follows: Section 2 describes our methodology; Section 3 reviews the experiments; Section 4 presents results and analysis; Section 5 offers a comparative discussion; and Section 6 concludes with future directions.

II. Literature Review

Robotics Vision is always an important but challenging task in the field of computer vision. Considering that the vision is an esstential way for both human begings and robots to perceive the world, it is important to make the robot understand the world through vision. In the past decades, scientists and researchers have been trying to seek an effective and stable way to achieve this goal. The mainstream methods can be divided into two categories:

- Traditional Local Feature Extraction Methods
- Deep Learning Methods

III. Methodology

A. Dataset

CIFAR-10 and iCubWorld 1.0 are two datasets used in this project for object classification tasks. CIFAR-10 is a well-known dataset containing 60,000 32x32 color images (10,000 for test) in 10 classes, with 6,000 images per class. While, iCubWorld 1.0 is a dataset collected by the iCub robot, including 10 categries, each categries contains 4 instances in the training set. Both dataset are famous benchmark in the field of computer vision, which is very suitable to compare the performance of local feature and deep learning methods. For each dataset, I applied corresponding data pre-processing methods separately, trying to improve the accuracy and robustness of the model, and compare the performance of different pre-processing methods.

1) Resizing: From the perspective of datasets, I applied different resizing approaches, since the image size variance. For CIFAR-10, since the original image size is just 32x32, which is definitely too small for SIFT method, whose basic patch size is 16x16. In order to apply the SIFT method on CIFAR-10, I resized the original image to 128x128 to ensure that it can get enough SIFT descriptors from the image. In this process, I choose the *Bicubic Interpolation* as the interpolation method, which can produce smoother results than other methods like nearest-neighbor or bilinear interpolation. For iCubWorld 1.0, the original image size is 640x480, and all of the training images are pre-cropped to 160x160. In this case, I also applied the *Bicubic Interpolation* method to resize the image to 224x224, which is the input size of my CNN model and allows more augmentation methods to be applied.

2) Normalization: In the SIFT+BoVW process, instead of doing a global "minus mean/divide variance" normalization on the whole image, we usually detect the keypoints on the original pixels and compute the SIFT descriptors, which perform L2 normalization and threshold truncation of the gradient histograms within each local patch, which itself confers partial invariance to lighting changes; histogram equalization or CLAHE can also be performed on the whole image beforehand to further enhance the contrast, if necessary. In contrast, CNNs are more sensitive to the distribution of the input data. We usually calculate the global mean and standard deviation of each channel (R/G/B) on the training set, and use the BatchNorm technique inside the network to stabilize the training process.

3) Data Augmentation: After that, in the following experiments pipeline, data augmentation is a common technique to artificially increase the generation of training data by applying various transformations to the original images. This helps to improve the generalization ability of the model and reduce overfitting. In this project, the data augmentation methods include: *Resizing*, *Cropping*, *Flipping*, *Color Jittering*, *Rotation*, *Affining*, *Perspective Changing*, *Erasing* and *Normalization*.

In the following Experiments section, this project will present performance comparison of different data preprocessing combinations (i.e. resizing, normalization, and augmentation), and explore the fittest strategy in each dataset and model.

B. SIFT

In this project, we use the SIFT algorithm to extract local features from images. SIFT is a feature detection algorithm that identifies and describes local features in images. It is invariant to scale and rotation, making it suitable for object recognition tasks where the objects may appear at different scales or orientations.

1) Keypoints Detection: The first step in the SIFT algorithm is to detect interest points in the image. These are points that are likely to be stable and distinctive, making them suitable for matching across different images. In order to detect these interest points for the following SIFT descriptor extraction, in this preject, I tried two different methods:

- Difference of Gaussian (DoG)
- Harris-Laplace

In computer vision, scale invariance is one of the key requirements for feature detection. And DoG is an efficient scale-space keypoint detection method, which is commonly used in local feature extraction processes such as SIFT. It approximates the Laplace-Gaussian (LoG) operator by making a subtraction between different Gaussian smoothing scales, and searches for spatial-scale extreme points in the generated DoG pyramid, obtaining points of interest with invariance to scale, which in turn significantly reduces the amount of computation.

While, for Harris–Laplace, its main idea of Harris–Laplace is combining multi-scale Harris corner detection with a scale-normalized LoG operator to selects keypoints that are both spatially precise and stable across scales. Its brief workflow is as follows:

1. Apply the Harris corner detection algorithm to the image at multiple scales to identify precise spatial localization of corner-like structures at each scale.
2. For each detected corner point, compute the LoG to determine the scale where the detected corner yields the strongest response, achieving scale invariance.

2) SIFT Descriptor Extraction: In order to practice SIFT, the project manually implements the various steps of SIFT descriptor extraction instead of calling the encapsulated SIFT methods directly through OpenCV. The following will briefly describe the SIFT workflow in the project:

1. Gradient magnitude and direction are calculated at each keypoint neighborhood through a 16*16 sized patch, and direction histograms are generated to pick the main direction (the project tried both 8-bins histgram and 36-bins histogram solutions)
2. The neighborhood of keypoints aligned with the main direction is divided into 4×4 subzones, and the histograms of gradients in 8 directions are counted in each zone and concatenated into a 128-dimensional vector
3. Applying L_2 normalization to obtain a final descriptor that is robust to illumination, scale, and rotation

3) Build Bag of Visual Words: After extracting SIFT descriptors for all the training images, I first clustered all the descriptors by kMeans method to get the cluster center (i.e. visual word) of each cluster.

For each image descriptor, call *kmeans.predict()* to assign each descriptor to the nearest visual word; then use *np.histogram* to count the number of descriptors that fall in each cluster, and obtain a one-dimensional histogram.

Finally, the histogram is normalized to convert the frequency into a probability distribution to form the final BoVW feature vector.

4) Classifier: This preject test diffetent classifiers on the BoVW feature vectors, including:

- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Softmax Regression

All of them will be applied to the BoVW feature vectors to classify the images, and their performance will be compared.

C. CNN

CNN is selected as another approach to achieve object classification, which is more state of the art and widely used in the field of computer vision.

1) Network Architecture: CNN Network Architecture is a key factor that affects the performance of CNN, which may influence the accuracy, speed, and robustness of the model. In this project, three different network architectures are tested:

- Self-Designed Simple CNN
- ResNet-50
- ResNet-50 with Coordinated Attention

All of them will implemented by self-coding, and the performance of them will be compared with each other.

2) Training Strategy: In the training process of CNN, the choice of training strategy is also a key factor that affects the performance of the model. In this project, different training strategies are tested, including:

- Different Optimizers (RMSprop, AdamW, Ranger)
- Different Learning Rates
- Different Batch Sizes
- Different Epochs

The performance of different training strategies will be recirded and compared with each other, and the best training strategy will be selected for the final model.

IV. Experiments

V. Results And Analysis

VI. Discussion

VII. Conclusion

References

[1] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images.(2009),” 2009.

[2] S. Fanello, C. Ciliberto, M. Santoro, L. Natale, G. Metta, L. Rosasco, and F. Odone, “icub world: Friendly robots help building good vision data-sets,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2013, pp. 700–705.