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, I 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, which are more 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 octaves, 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 the Harris–Laplace method, its central idea lies in effectively combining the spatially accurate Harris corner detector with the scale-invariant properties of the scale-normalized Laplacian-of-Gaussian (LoG) operator. This ensures that detected interest points not only have high spatial precision but are also stable across scales.

Its brief workflow is as follows:

1. Spatial corner detection at multiple scales: The Harris corner detection algorithm is applied independently to the input image at several scales (constructed via Gaussian smoothing). This step provides precise spatial localization of corner-like structures across each individual scale.
2. Scale selection through scale-normalized LoG: For each corner candidate detected from the previous step, the scale-normalized Laplacian-of-Gaussian (LoG) response is computed across scales. The scale at which the candidate corner exhibits the strongest (local extrema) response is selected as its characteristic scale, ensuring robust scale invariance.

By this way, the Harris–Laplace method effectively emphasizes corner-type interest points, which typically convey richer structural information compared to edge points, while simultaneously preserving the desirable property of scale invariance inherent in LoG-based approaches.

In addition, the traditional Harris and LoG algorithms are also tested in this project, and their results are compared in the Experiments section.

2) SIFT Descriptor Extraction: After gaining the interest points in Section III-B1, the project manually implements the various steps of SIFT descriptor extraction. 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 region. Then, an orientation histogram with 36 bins (each covering 10°) is constructed. The peak direction from this histogram is selected as the keypoint' s main orientation, ensuring rotation invariance. (The project also experimented with fewer bins, such as 8, to observe effects on accuracy and computational complexity.)
2. After aligning the neighborhood region with the keypoint' s main orientation (rotating the local patch accordingly), the region is divided into a 4×4 grid of smaller cells. Within each of these 16 cells, gradient orientations are quantized into an 8-bin histogram covering 360°. These histograms are then concatenated, forming a $4 \times 4 \times 8 = 128$-dimensional vector.
3. Applying L_2 normalization to the 128-D vector to obtain a final descriptor that is robust to make them independent of illumination and contrast variations, thus enhancing robustness.

3) Build Bag of Visual Words: After extracting SIFT descriptors (the 128-D vector) 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. And the dimensionality of the final BoVW image coding vector is determined by the number of target clusters, presenting the expressive capacity of the BoVW.

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), and Softmax Regression. The performance of these classifiers will be compared with each other to explore the best classifier for the BoVW feature vectors.

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 - Self-Designed Simple CNN: 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-18
- ResNet-50

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

The following Figure 1 is the brief architecture of the self-designed simple CNN model. It contains 2 stage, each stage contains 1 convolutional layers, 1 max pooling layer, and 1 ReLU activation layer. Then, the output of the last stage is flattened and passed to a fully connected layer. Its main character is to play as the baseline model, and

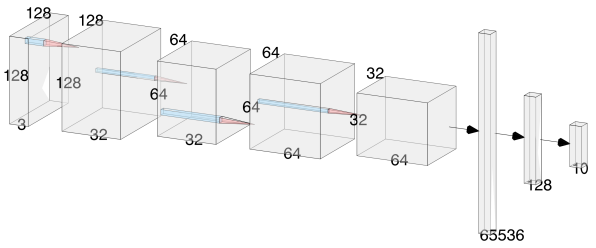

Fig. 1: Architecture of the Self-Designed Simple CNN

compare with the other two morden architectures (i.e. ResNet-18 and ResNet-50) to explore the performance of different network architectures.

2) Network Architecture - ResNet: The Table I shows the architecture of ResNet-18 and ResNet-50, which are two popular architectures in the field of computer vision [3]. Compared with the self-designed simple CNNs, ResNets are more complex and deeper, but their most notable feature is the introduction of a residual learning mechanism to alleviate the degradation problem that occurs in the training process of deep networks. For traditional deep CNN models (e.g., my self-designed CNN model), although theoretically I can enhance the model's representation ability by adding more layers and building

layer name	output size	18-layer	50-layer
conv1	112×112	7×7, 64, stride 2	
conv2_x	56×56	3×3 max pool, stride 2	
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax	
FLOPs		1.8×10^9	3.8×10^9

TABLE I: Architectures for ResNet (reproduced from [3])

a deeper network, they are prone to suffer from the gradient disappearance or model performance degradation in practice.

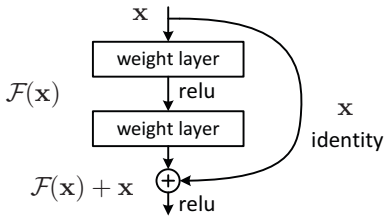

Fig. 2: Residual Block Architecture (reproduced from [3])

In order to solve this bottleneck, ResNet introduces skip connections in the network structure, as shown in Figure 2, which allows the network to learn a residual mapping, thus transforming the final objective function into the form of $H(x) = F(x) + x$. In addition, this structure also promotes the reuse of features, allowing shallow features to directly affect deeper outputs, effectively mitigating the problem of gradient attenuation in deeper networks during backpropagation.

3) Coordinate Attention: In the field of deep learning, attention mechanisms [4] have been widely used to enhance the model's performance. In this project, I also applied the Coordinate Attention (CA) mechanism [5] to the ResNets. CA is a lightweight attention mechanism designed to enhance the ability of CNNs to model spatial information while maintaining high efficiency. Unlike traditional channel attention mechanisms (e.g., SE, CBAM) that focus on "channel dimensions", Coordinate Attention encodes both "spatial location information (coordinates)" and "channel dependencies" . "channel dependencies" . The specific schematic comparison in shown in Figure 3

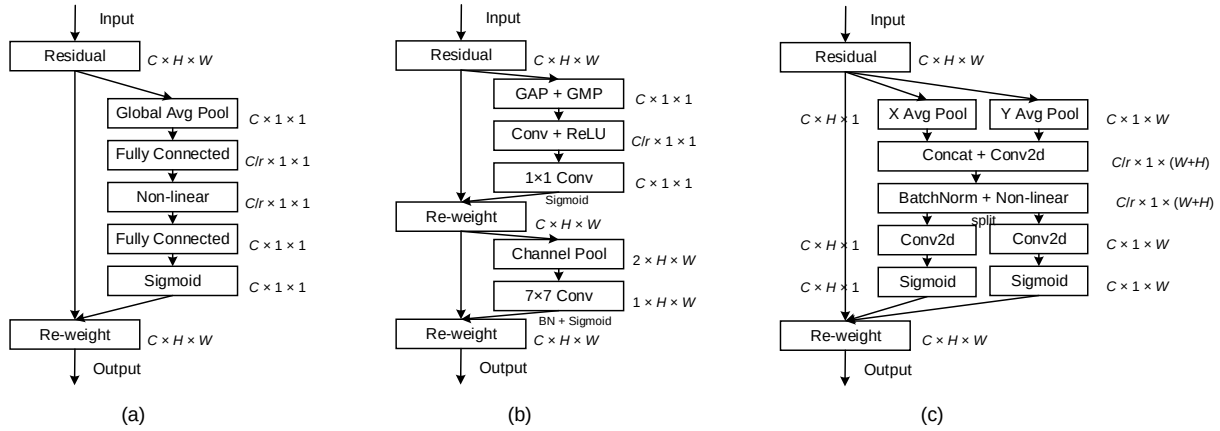

Fig. 3: Schematic comparison of the proposed Coordinate Attention block (c) to the classic SE channel attention block [6] (a) and CBAM [7] (b). (Reproduced from [5])

IV. Experiments

In the experimental part, this project focuses on designing the final model that meets the project task requirements by making a lot of comparisons with different hyperparameters as well as modeling mechanisms. For the SIFT method, the main hyperparameters to be adjusted are the BoVW size and the maximum number of key points to be included in each graph. For CNN, the main hyperparameters to be adjusted are batch size, learning rate, and epoch number, and these basic hyperparameters will be used to set the base for grid searching on each dataset at the beginning of the experiments, and further hyperparameters will be selected based on these basic parameters for more in-depth explorations (e.g., patch size in SIFT, or the weight delay of the optimizer in CNN). In addition, the experimental section will also test each of the specific mechanisms discussed in the methodology section.

A. SIFT

B. CNN

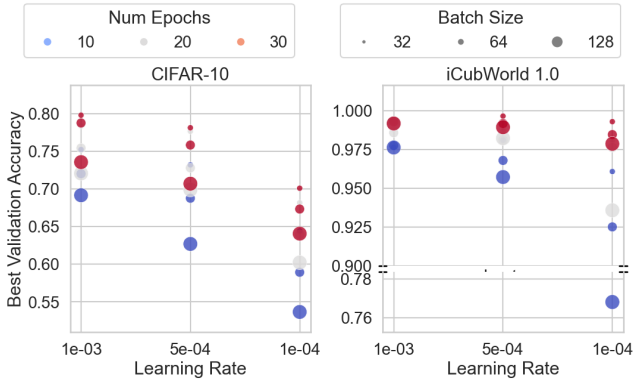

Fig. 4: Grid Search Results of CNN on CIFAR-10 and iCubWorld 1.0 (performed on Ranger optimizer)

1) Grid Search: In the grid search process on CNNs, due to the large amount of data (especially for CIFAR-10), it is inefficient to use the entire dataset. Thus, I applied the downsampling method to randomly select 30% of CIFAR-10 and 70% of iCubWorld 1.0 for the grid searching. In this case, there are 12000 images in CIFAR-10 and 4200 images in iCubWorld 1.0 being used for the grid search, which is representative. And the final results are shown in Figure 4. It is clear that the best performance is achieved with a batch size of 32, a learning rate of 1e-3, and 30 epochs in both datasets. Therefore, these hyperparameters are selected as the base for the further experience.

2) Model Architecture Comparison: As mentioned before, this project tests three different network architectures: *self-designed simple CNN*, *ResNet-18*, and *ResNet-50*.

3) Optimizer Comparison: In the training process of CNN, the choice of optimizer is also a key factor that affects the performance of the model. This project mainly tests two different optimizers: *AdamW* [8], and *Ranger* [9].

4) Attention Mechanism Comparison:

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.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 713–13 722.

[6] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.
[7] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.
[8] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
[9] L. Wright, "Ranger - a synergistic optimizer." https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer, 2019.