Assignment-1

Deep Learning

Wen-Hai Tseng
RE6124027
Nation Cheng Kung University
w13281328@gmail.com

Abstract*—This assignment requires the use of three different models and feature extraction techniques to perform image classification tasks, as well as an analysis of the impact of different feature extraction methods and models on performance. Additionally, it involves comparing the results with the use of learning-based feature extraction methods.

I. Introduction

This assignment requires the use of three different machine learning models and feature extraction techniques to perform image classification tasks. I will utilize three common machine learning models: K-Nearest Neighbors, Support Vector Machine, and XGBoost, along with three feature extraction methods: BRISK, ORB, and HOG. Additionally, I will employ a learning-based feature extraction method using ResNet50. We will discuss the impact of these feature extraction methods and models on performance later on.

II. Method

This section will introduce the feature extraction methods used, including BRISK, ORB, ResNet50, and HOG. These feature extraction techniques enable our classification models to achieve better performance during training.

A. *feature extraction --BRISK*

BRISK (Binary Robust Invariant Scalable Keypoints) is a feature detection and description algorithm commonly used in applications such as object recognition, target tracking, and 3D reconstruction in computer vision. The algorithm consists of two main parts: feature detection and description.

Feature Detection:

BRISK employs the Harris corner detector to detect corners in the image, which are prominent features used for identification and matching.

Firstly, a Gaussian filter is applied to the image to reduce noise.

Then, the Harris corner detector is used to compute the corners in the image.

Finally, the final feature points are selected based on the values of the Harris response function and a certain threshold.

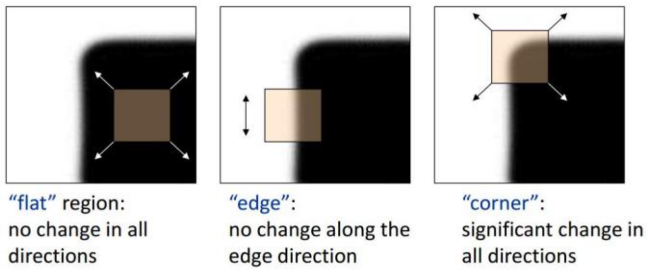
"flat" region:
no change in all directions
"edge":
no change along the edge direction
"corner":
significant change in all directions

Fig.1 Feature points in an image generally have specific coordinates and possess certain mathematical characteristics, such as local maximum or minimum grayscale values, and certain gradient features. Corner points can be simply regarded as the intersection points of two edges, or more strictly defined as feature points with two main directions in their neighborhood, meaning significant grayscale changes occur in two directions. As shown in Figure 1, by moving a small window in various directions, if grayscale changes occur in all directions, it is considered a corner point; if there is no change in any direction, it is a uniform region; and if grayscale changes occur only in one direction, it may be an image edge.

Feature Description:

For each detected feature point, BRISK describes the surrounding texture information using a fixed-size region.

BRISK utilizes binary descriptors to describe these regions, containing comparison results for each subregion and a summarized comparison result.

Binary descriptors provide high computational efficiency and robustness, making them particularly suitable for real-time applications.

For instance, suppose we have an image and wish to detect and describe its feature points using the BRISK algorithm.

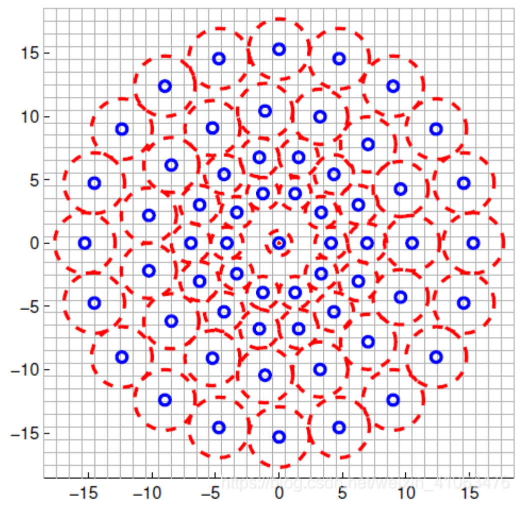

Fig.2 Construct concentric circles with different radii centered around feature points, and sample a certain number of equidistant points on each circle.

B. *feature extraction --ORB*

FAST (Features from Accelerated Segment Test) is used for rapid identification of keypoints in images. It detects potential corner positions by comparing the grayscale values of pixels with their surrounding pixels. The FAST algorithm searches for pixels with significant grayscale changes, which often correspond to corners in the image. For each detected keypoint, the ORB algorithm computes its main orientation by calculating the gradient directions of surrounding pixels. This orientation assignment enables rotation of descriptors relative to the keypoints' orientation during descriptor computation, thereby increasing descriptor robustness. BRIEF descriptors are binary descriptors used to describe local image structures around keypoints. They are implemented by selecting a set of binary tests within a window around the keypoint. In ORB, to enhance descriptor robustness, BRIEF descriptors are rotated to align them with the keypoints' orientation.

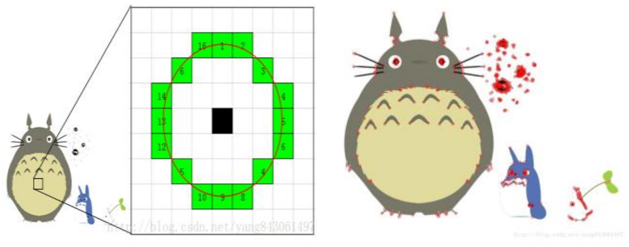

Fig.3 The FAST (Features from Accelerated Segment Test) a lgorithm is used for detecting keypoints. The core idea of FA ST is to find pixels with significant grayscale changes by co mparing a point with its surrounding points. If a point differs from most of its surrounding points, it can be considered a fe ature point.

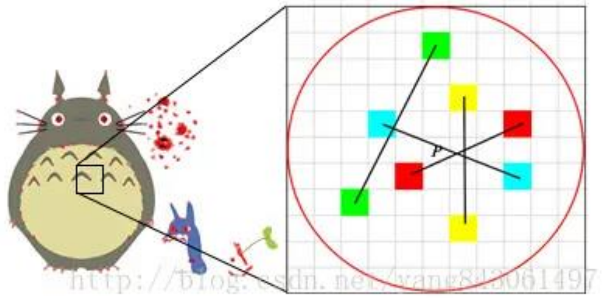

Fig.4 ORB uses the BRIEF algorithm to compute descriptors for keypoints. The core idea of the BRIEF algorithm is to select a certain pattern of N point pairs around the keypoint P and combine the comparison results of these N point pairs as the descriptor.

C. *feature extraction --HOG*

In images, the representation and shape of local objects can be well described by the density distribution of gradients or edges. Its essence lies in the statistical information of gradients, which primarily exist at the edges. The image is divided into small connected regions called cells. Then, histograms of gradients or edge orientations are collected for each pixel in these cells. Finally, these histograms are combined to form feature descriptors.

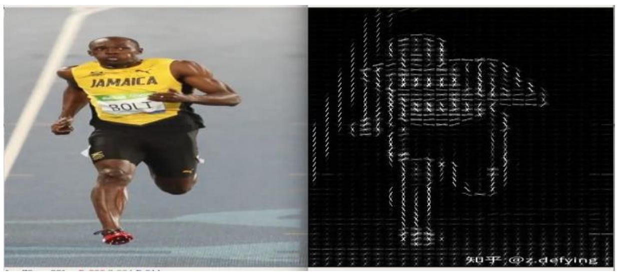

Fig.5 The visualization of the HOG feature descriptor is shown in the figure, where we can clearly observe the outline of a person.

D. *ResNet50*

ResNet50 is a classic deep residual neural network commonly used for image recognition and feature extraction tasks. The network consists of multiple residual blocks, each containing several convolutional layers and batch normalization layers. By incorporating skip connections, ResNet50 introduces the original input information into deeper layers of the network, which helps address the vanishing gradient problem. ResNet50 gradually extracts abstract features from images, such as edges, textures, shapes, and object categories, thereby performing exceptionally well in various image-related tasks.

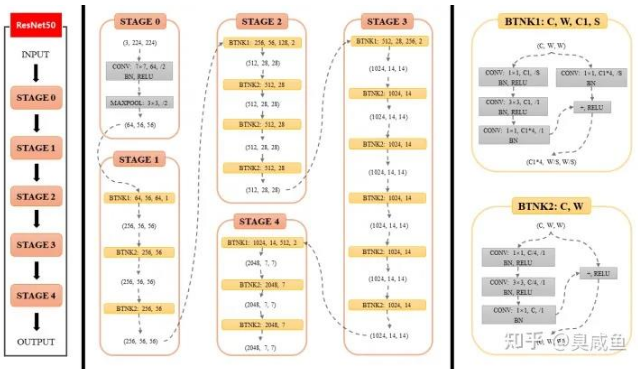

Fig.6 The above diagram illustrates the network architecture of ResNet.

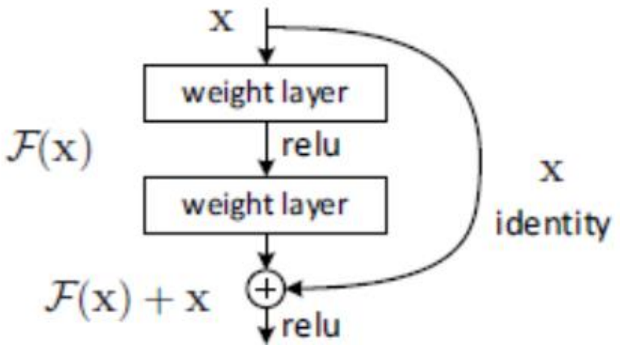

Fig.7 As the depth of a network model increases, one might expect the performance to improve. However, in practice, as more layers are stacked, the performance may start to degrade due to the common issue of vanishing gradients in older-style networks. Residual networks can be understood as incorporating shortcut connections in the forward pass of the

network. These connections skip certain layers and directly pass the original data to subsequent layers. The additional shortcut connections do not increase the number of parameters or complexity of the model, effectively addressing the problem of vanishing gradients in excessively deep networks.

III. EXPERMENTS AND RESULT

In this experiment, we utilized three models: SVM, KNN, and XGBoost, along with four feature extraction methods: BRISK, ORB, HOG, and ResNet50. After applying each feature extraction method, we compressed the extracted features' structures into the same format using k-means clustering. Additionally, in the program implementation, we only used the first 5 classes as examples to obtain results more quickly. The number of clusters in k-means was set to 50 centroids.

	BRISK	ORB	HOG	ResNet50
KNearest Neighbors	40%	40%	0%	80%
SVM	20%	60%	20%	80%
XGBoost	60%	80%	0%	80%

Table.1 showing the accuracy of three models and four feature extraction methods for the first 5 classes on test data.

	BRISK	ORB	HOG	ResNet50
KNearest Neighbors	33%	33%	0%	73%
SVM	20%	53%	7%	73%
XGBoost	53%	73%	0%	73%

Table.2 presents the F1-Score of three models and four feature extraction methods for the first 5 classes on test data.

From Tables 1 and 2, it can be observed that excluding learning-based feature extraction, HOG does not provide good features for the models to learn. Among ORB and BRISK, ORB yields better results. If considering learning-based feature extraction, it is found to yield similar results to ORB. Finally, it is observed that the best-performing combination of feature extraction and model in this test is ResNet50 or ORB combined with the XGBoost model. To improve accuracy, you may consider increasing the number of clusters in k-means and retaining a higher number of features.

Analyze the computational complexity of each classification model

K-Nearest Neighbors: K-Nearest Neighbors (KNN) does not have a visible training process, thus its training time is O(1). During prediction, it needs to iterate through the entire training dataset to find the nearest neighbors. The time complexity of this process is O(N), where N is the number of training samples. KNN requires storing the entire training dataset, resulting in a space complexity of O(N).

Support Vector Machine: The training time of SVM primarily depends on the size and complexity of the training dataset. For linear SVM, the training time is

:

:

typically O(N), where N is the number of training samples. For non-linear SVM, it usually ranges from O(N^2) to O(N^3), and the prediction time ranges from O(d) to O(Nd), where d is the number of features and N is the number of support vectors. SVM requires storing support vectors, resulting in a space complexity of O(N), where N is the number of support vectors.

XGBoost: The training time of XGBoost mainly depends on the number of trees, the depth of trees, and the size of the data. Typically, the training time of XGBoost ranges from O(ndlog(n)) to O(nd^2log(n)), where n is the number of samples and d is the number of features. The prediction time of XGBoost is O(k*d), where k is the number of trees and d is the number of features. XGBoost needs to store the structure of trees and the values of leaf nodes, resulting in a space complexity of O(k*d), where k is the number of trees and d is the number of features.

Based on the above information, you can choose an appropriate classifier according to the requirements of the problem and the scale of the data. If the data size is small, KNN might be a good choice; however, if the data size is large, SVM and XGBoost may be more suitable options.

.

REFERENCES

[1] https://blog.csdn.net/weixin_41063476/article/details/90407916

[2] https://blog.csdn.net/qq_30263737/article/details/94630517

[3] https://senitco.github.io/2017/06/18/image-feature-harris/

[4] Chatgpt