ANALYSIS OF TWO ACTION RECOGNITION METHODS WITH STILL IMAGES

GROUP 404

何義翔 王裕昕 邱冠崴 林悦揚

TABLE OF CONTENTS

INTRODUCTION

Motivation

Recognizing human actions allows machines to actively support the execution of human tasks and operations, resulting in tangible cost savings through reduced manpower requirements.

OUR GOAL

Goal: Develop an AI system to detect human positions and actions in images, annotate these actions, and effectively handle multiple subjects and user-provided images. Analyzed and compare with different method.

RELATED WORK

Method 1
Learning to Detect Human-Object Interactions
Conducted by Deeptha Girish, Vineeta Singh, and Anca Ralescu

- This paper studies the detection of human-object interactions (HOI) in static images. It introduces HICO-DET, a large benchmark for HOI detection, and proposes the Human-Object Region-based Convolutional Neural Network (HO-RCNN). The HO-RCNN framework uses Interaction Patterns to characterize the spatial relations between two bounding boxes. The experiments show that HO-RCNN significantly improves HOI detection performance over baseline approaches.

RELATED WORK

Method 2
Learning to Detect Human-Object Interactions
Conducted by Yu-Wei Chao, Yunfan Liu, Xieyang Liu, Huayi Zeng, and Jia Deng

- This paper addresses the problem of recognizing actions from static images, which is more challenging than video-based action recognition due to the absence of spatio-temporal features. The study focuses on actions involving object manipulation and breaks down complex actions into smaller semantic components. It explores the importance of these components in action recognition. The paper leverages YOLO for detecting regions of interest and uses pre-trained AlexNet features for classification.

RELATED WORK

the difference between our work and the existing ones

- Modern HAR methods typically use video data, which contains rich spatio-temporal information essential for accurate action recognition.
- Combines Convolutional Neural Networks and Long Short-Term Memory networks (LSTMs) to extract spatio-temporal features from videos, with 3D CNNs handling both spatial and temporal dimensions.

DATASET

Verbs in COCO (V-COCO) Dataset
This repository hosts the Verbs in COCO (V-COCO) dataset and associated code to evaluate models for the Visual Semantic Role Labeling (VSRL) task as described in *this technical report.*

- Following the GitHub to download the image.
- the image is JPG file
- Use the official 2014 Train/Val annotations to select some specific feature

https://github.com/s-gupta/v-coco

DATASET

crop_image.py

1. Loads and processes COCO and V-COCO annotations.
2. Filters actions with **one person** and **one object**.
3. Prints the count of positive samples.
4. Downloads images, handles errors, and processes bounding boxes.

DATASET IMPROVEMENT

The drawback of V-COCO:

- The differences in the number of photos vary greatly across each category.
- The quantity of testing data is significantly greater than that of training data.

Improvement:

- Delete the action whose picture numbers are less than 400
- Select the equal numbers of each action’s pictures
- Split the training and testing dataset on our own

DATASET

Split Dataset

1. Check the number of images files ignoring actions fewer than 400 images
2. Split the dataset with test size 10% train size 90% and every action is seleted 400 pictures.
3. the data is order by the number of labels.

```python
def split_dataset():
    print('\n ======== Splitting dataset ======== ')
    root_dir = "data/all"
    datas = []
    labels = []
    for i, action in enumerate(os.listdir(root_dir)):
        action_path = os.path.join(root_dir, action)
        image_ids = os.listdir(action_path)
        if len(image_ids) < 400:
            print(f'{action} has less than 400 images, ignore')
            continue
        datas += [os.path.join(action_path, image_id) for image_id in image_ids[:400]]
        labels += ([i] * 400)
        actions.append(action)
        print(f'{i} - {action}: {len(image_ids)} images')
    train_id_path, test_id_path, train_label, test_label = train_test_split(datas, labels,
    train_id_path, train_label = zip(*sorted(zip(train_id_path, train_label)))
    test_id_path, test_label = zip(*sorted(zip(test_id_path, test_label)))
    print('Train dataset:', len(train_id_path))
    print('Test dataset:', len(test_id_path))

    return train_id_path, test_id_path, train_label, test_label
```

BASELINE

Baseline Model: Multinomial Logistic Regression

- Purpose:
  - Logistic regression is a statistical method used for binary classification problems, where the goal is to predict one of two possible outcomes.
  - Multinomial logistic regression generize the idea into multiple classes.

- Mechanism:
  - It models the probability that a given input belongs to a particular class.
  - The logistic function (sigmoid function) is used to map predicted values to probabilities.

BASELINE

Baseline Model: Logistic Regression

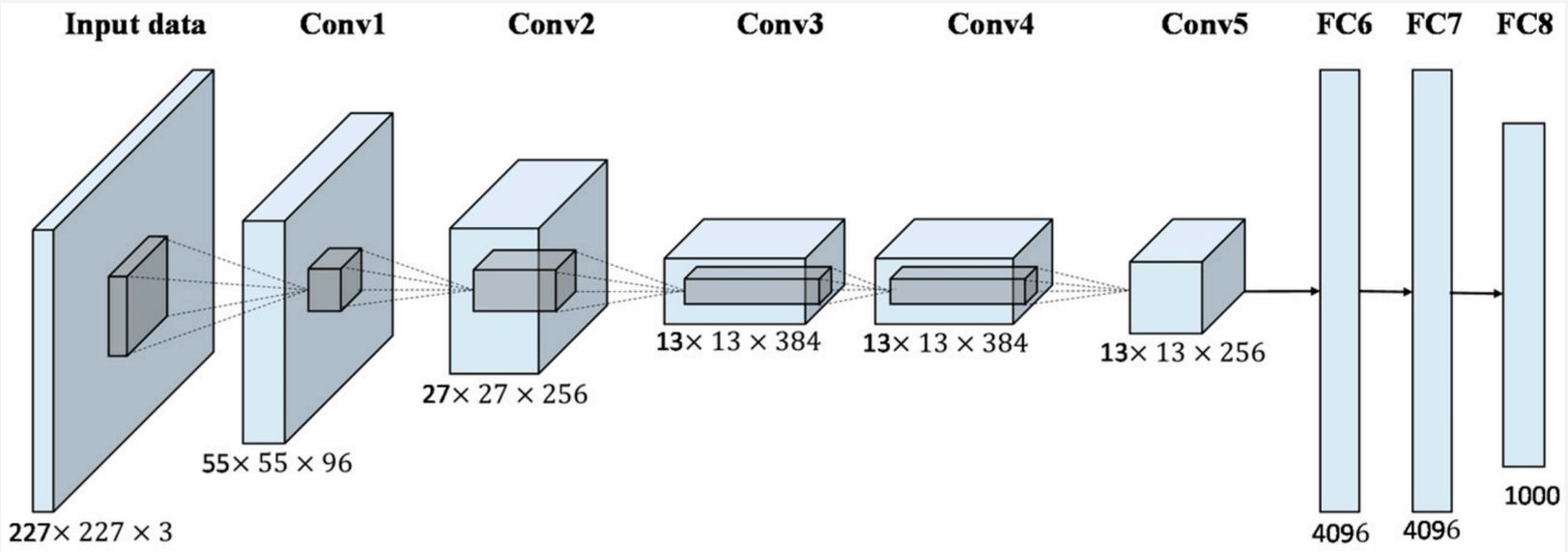
- Training:
  The model is trained using the maximum likelihood estimation approach. It adjusts weights to minimize the difference between the predicted probabilities and the actual class labels.

MAIN APPROACH

MAIN APPROACH-METHOD1

Building Neural Network

1. Create 3 streams: human stream, object stream, and pair stream.
2. Utilize AlexNet as the fundamental structure of human stream and object stream.
3. Build another convolutional neural network for pair stream.



MAIN APPROACH-METHOD1

Training

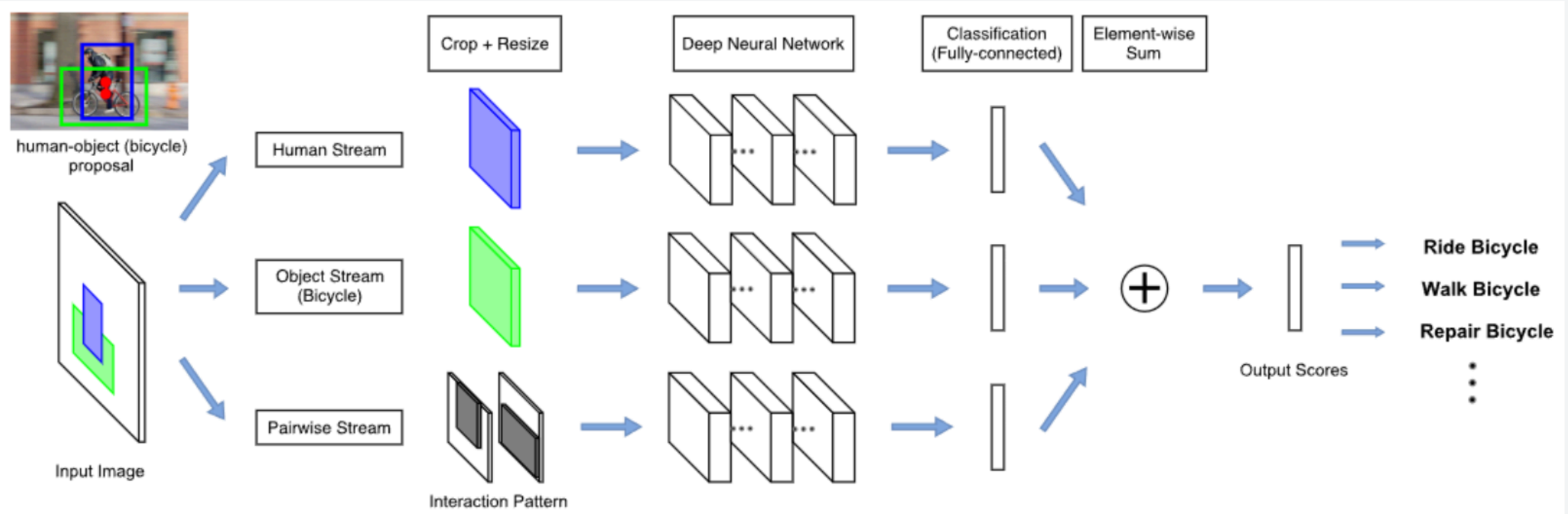
1. Train human stream model with human images, object stream model with object images, pair stream model with human&object union images.

2. Obtain an element-wise sum of the outputs from the three models and make predictions.

3. Calculate the loss of prediction and do backpropagation.

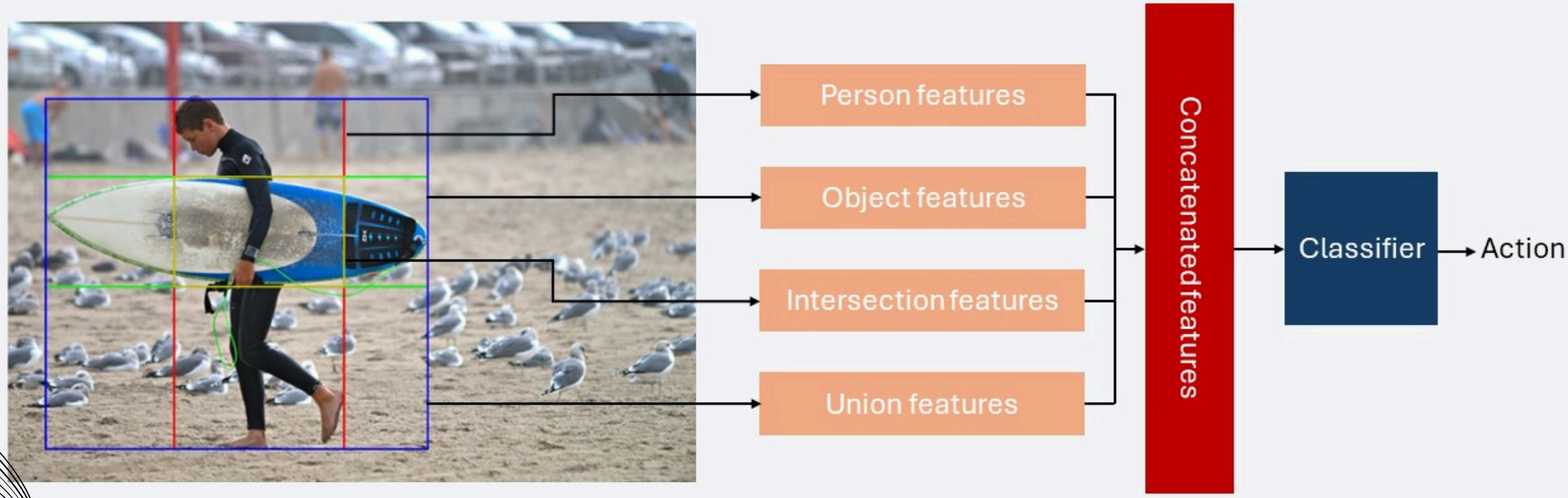
MAIN APPROACH-METHOD1

Overall concept



MAIN APPROACH-METHOD2

Workflow



MAIN APPROACH-METHOD2

Workflow

1. Crop the original image into four images. Each image contains only agent, object, union, intersection.
2. Using AlexNet to extract the feature (activated from FC7) for each image and concate them. We will get a 1x12192 array
3. Using the arrays above and apply multinomial logistic regression, the result is just the action.

MAIN APPROACH-METHOD2

extract_feature.py

1. Use AlexNet but activate from fc7 layer (the second fully connect layer)
2. Put four images (human, object, union, intersection) into AlexNet to extract sub-features
3. Concatenate those sub-features into a big feature, representing the original image

MAIN APPROACH-METHOD2

main.py-train

1. Use the method in Dataset Improvement to spilt training and testing datset
2. Extracts features from training images using a pre-trained AlexNet model.
3. Save those features to reuse.
4. Trains a multinomial logistic regression model with the extracted features.
5. Saves the trained model to a file.

MAIN APPROACH-METHOD2

main.py-test

1. Load the features and the classifier
2. Evaluates the model's accuracy on the test dataset and draw confusion matrix.

RESULTS AND ANALYSIS

Type of experiment

1. comparison between two methods
2. with/without dataset improvement
3. Method 2 with/without transferred weights from Method 1
4. with/without additional classifiers

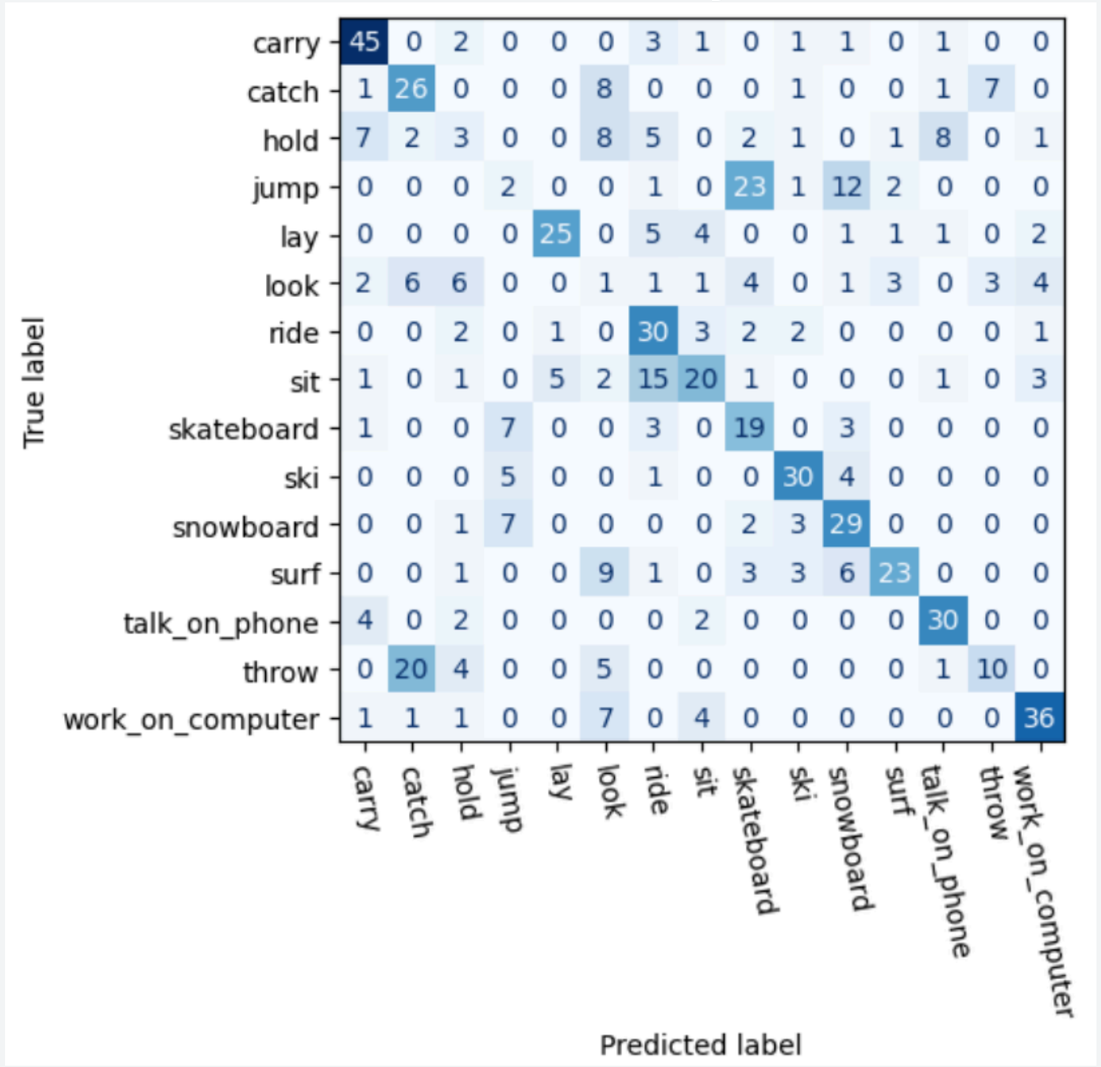
EVALUATION METRIC

- **PERFORMANCE** (ALL THE RESULT ACCURACY IS ROUNDED OFF TO THE 3ND DECIMAL PLACE)

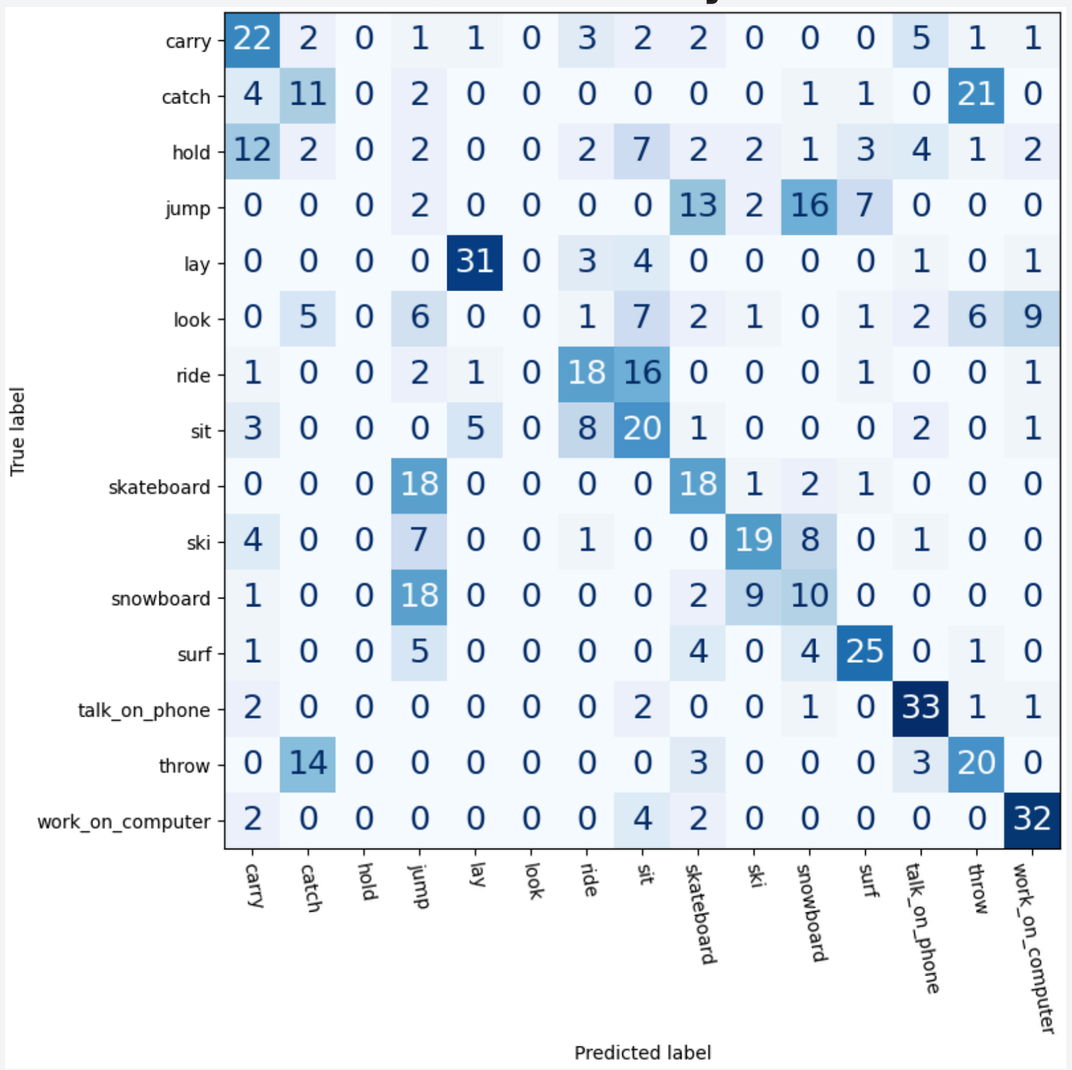
	Method 1	Method 2
Exp1	0.525	0.433
Exp2	0.679	0.677
Exp3	0.677	0.750
Exp4	0.733	0.738

EXPERIMENT 1

Method 1 / Accuracy: 0.525

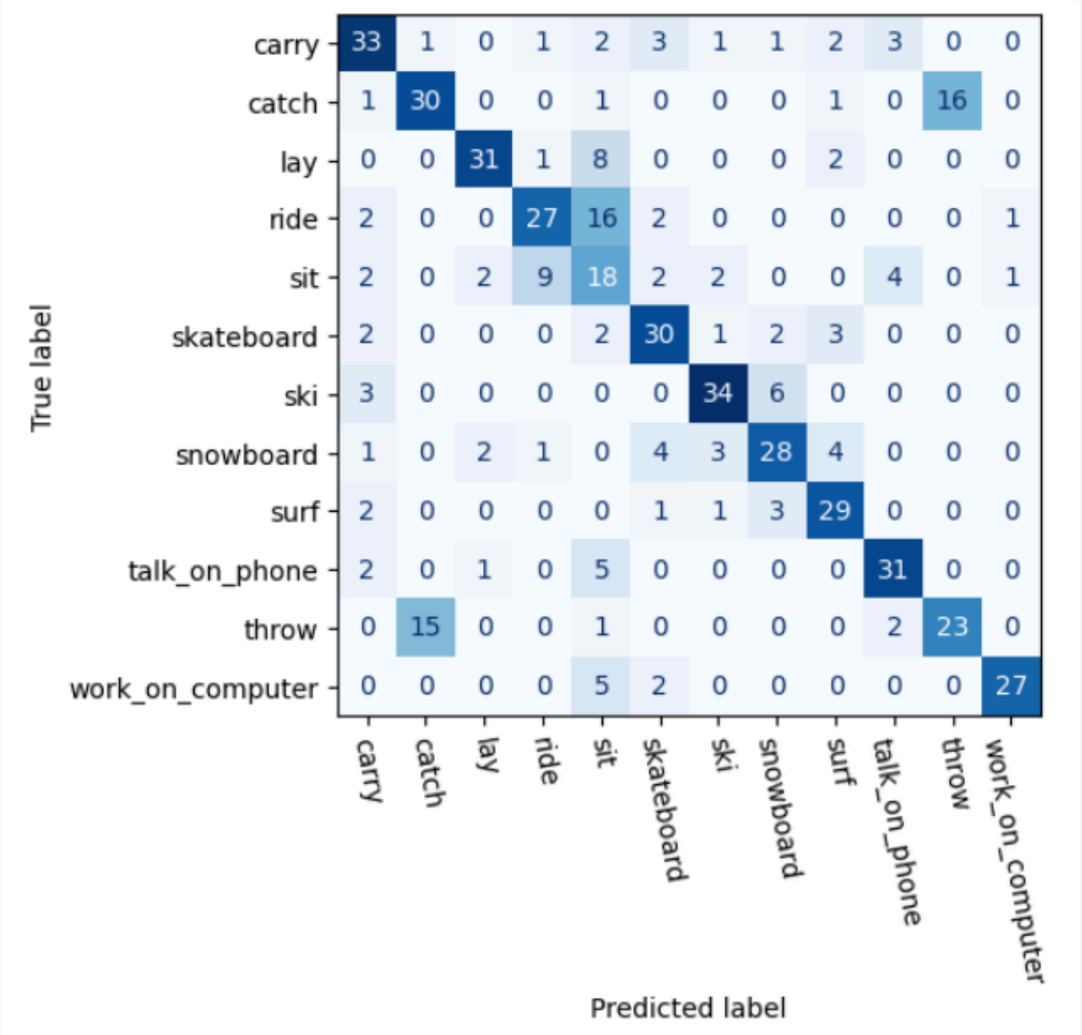


Method 2 / Accuracy: 0.435

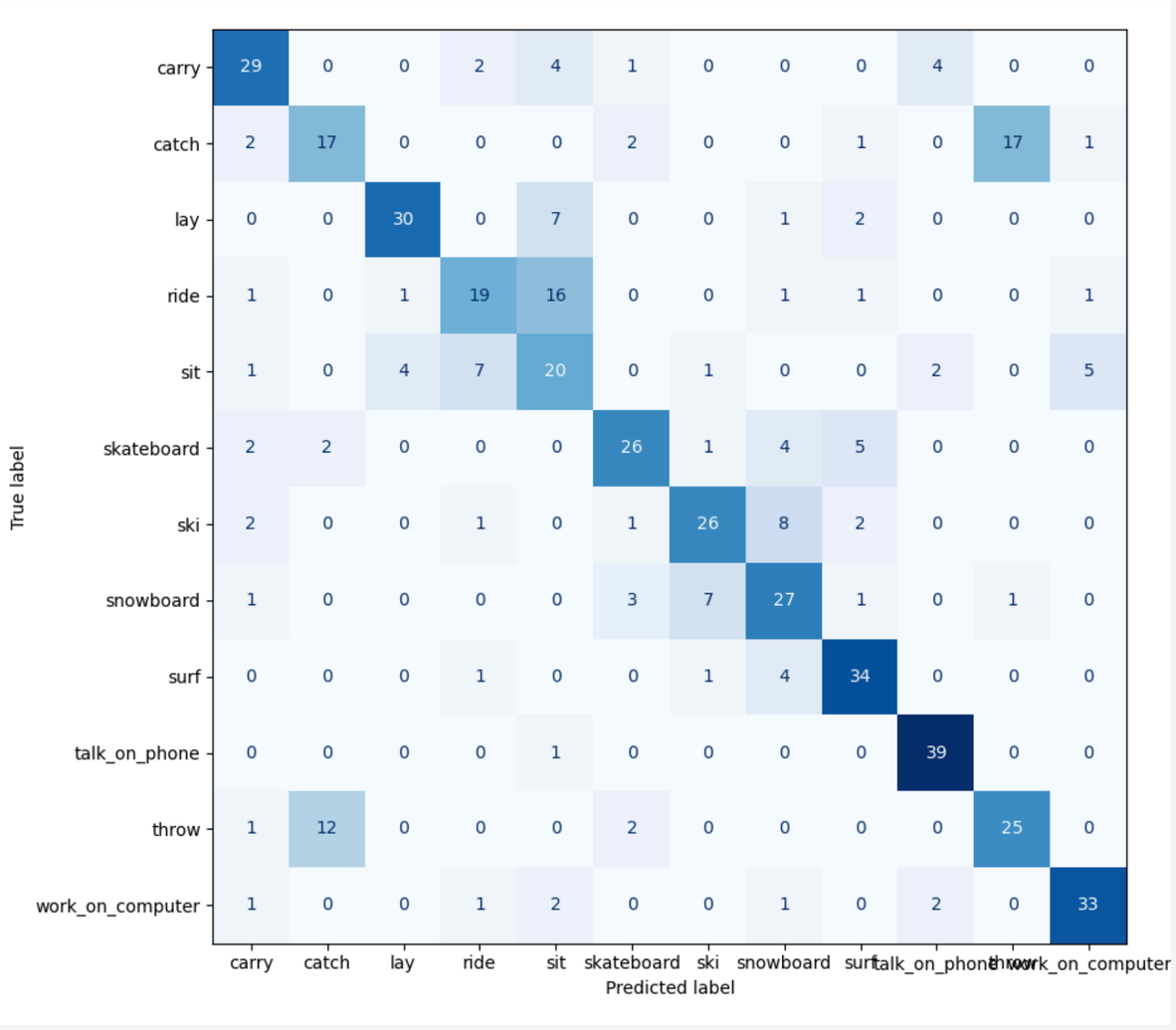


EXPERIMENT 2

Method 1 / Accuracy: 0.679

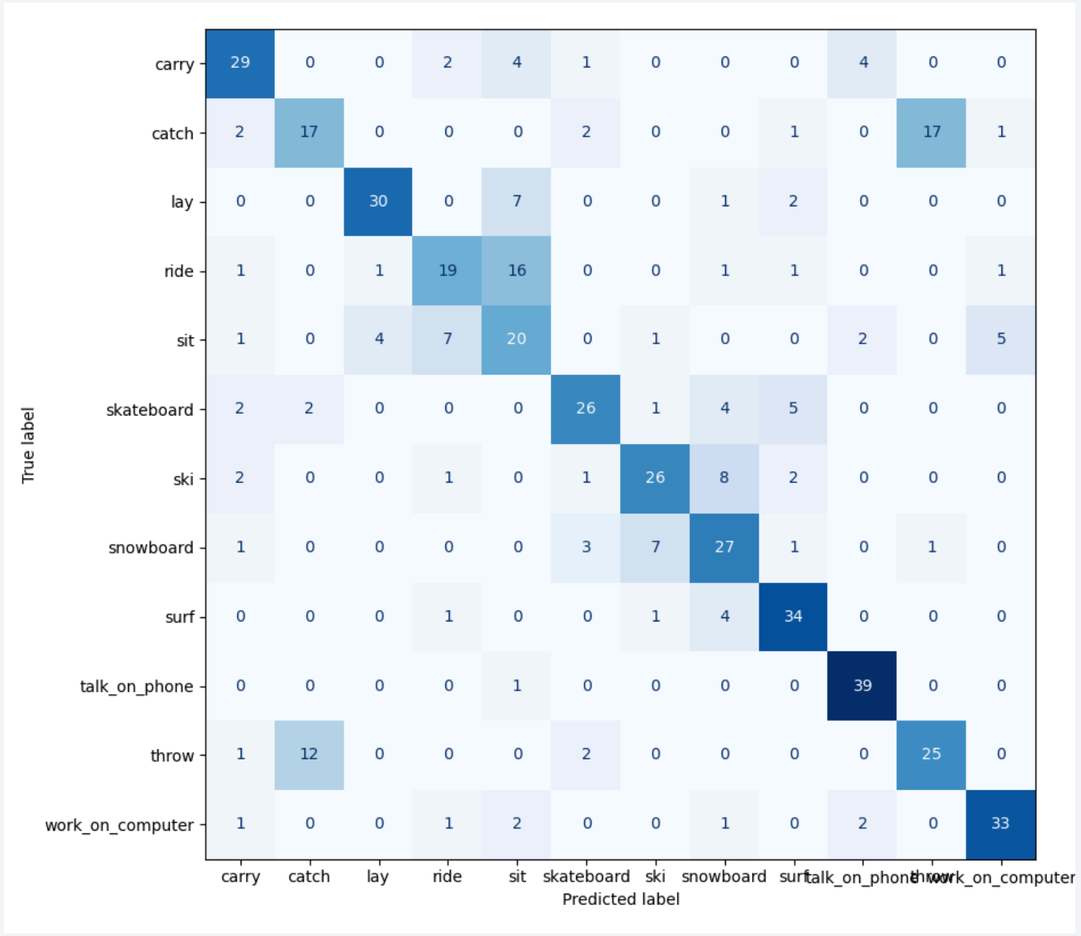


Method 2 / Accuracy: 0.677

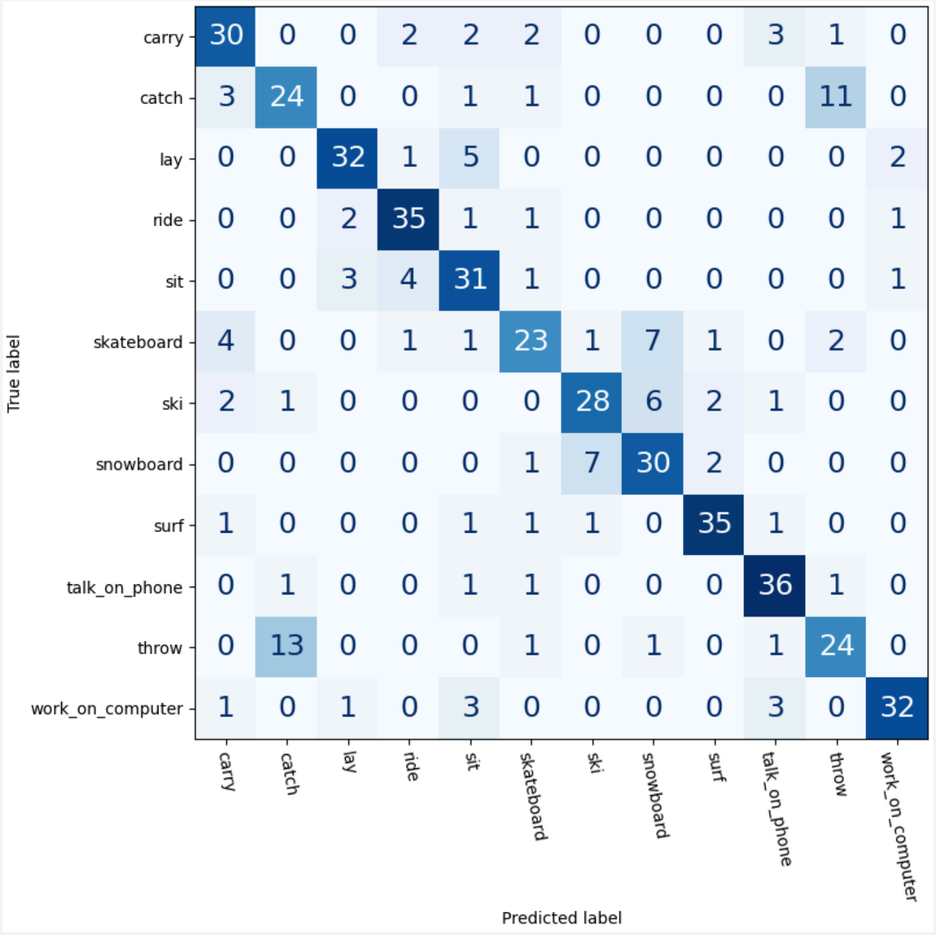


EXPERIMENT 3

Method 2 w/ default weight
Accuracy: 0.677

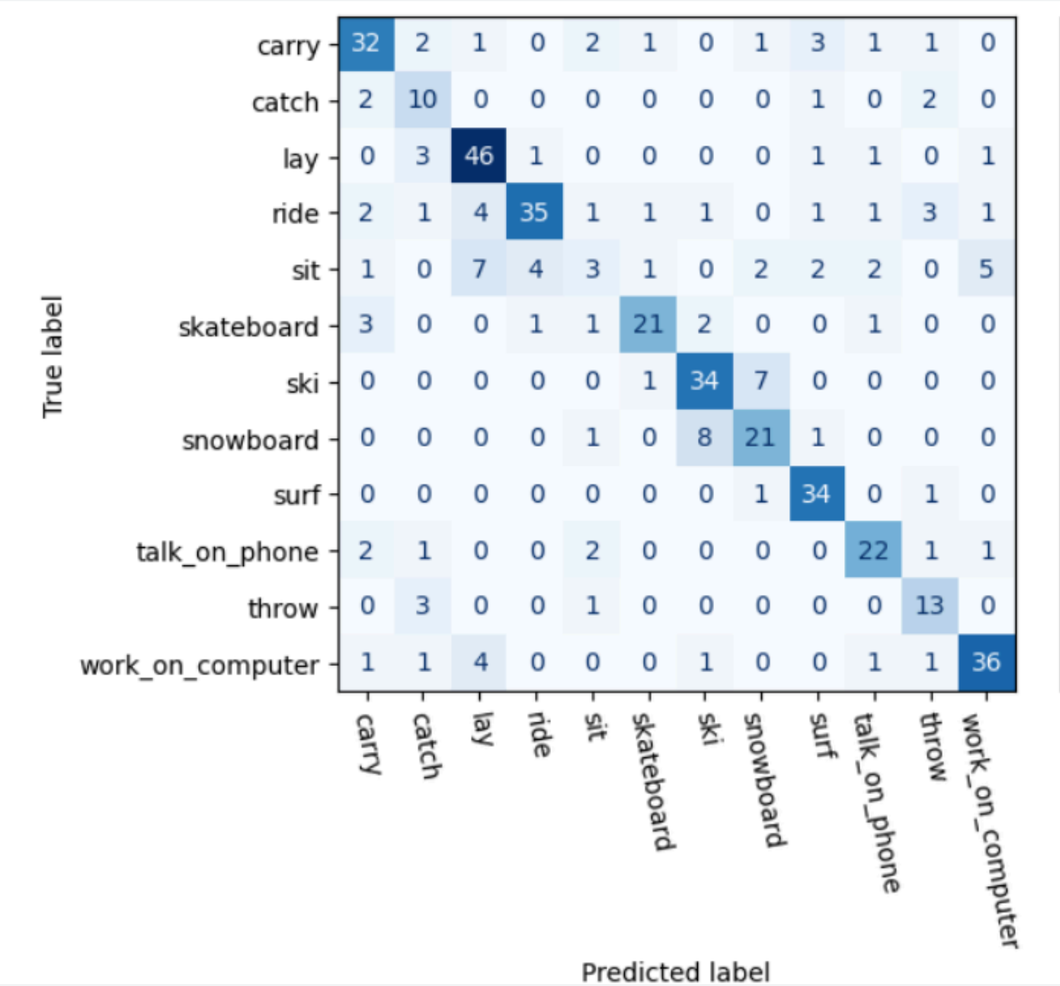


Method 2 w/ weights after transfer learning
Accuracy: 0.750

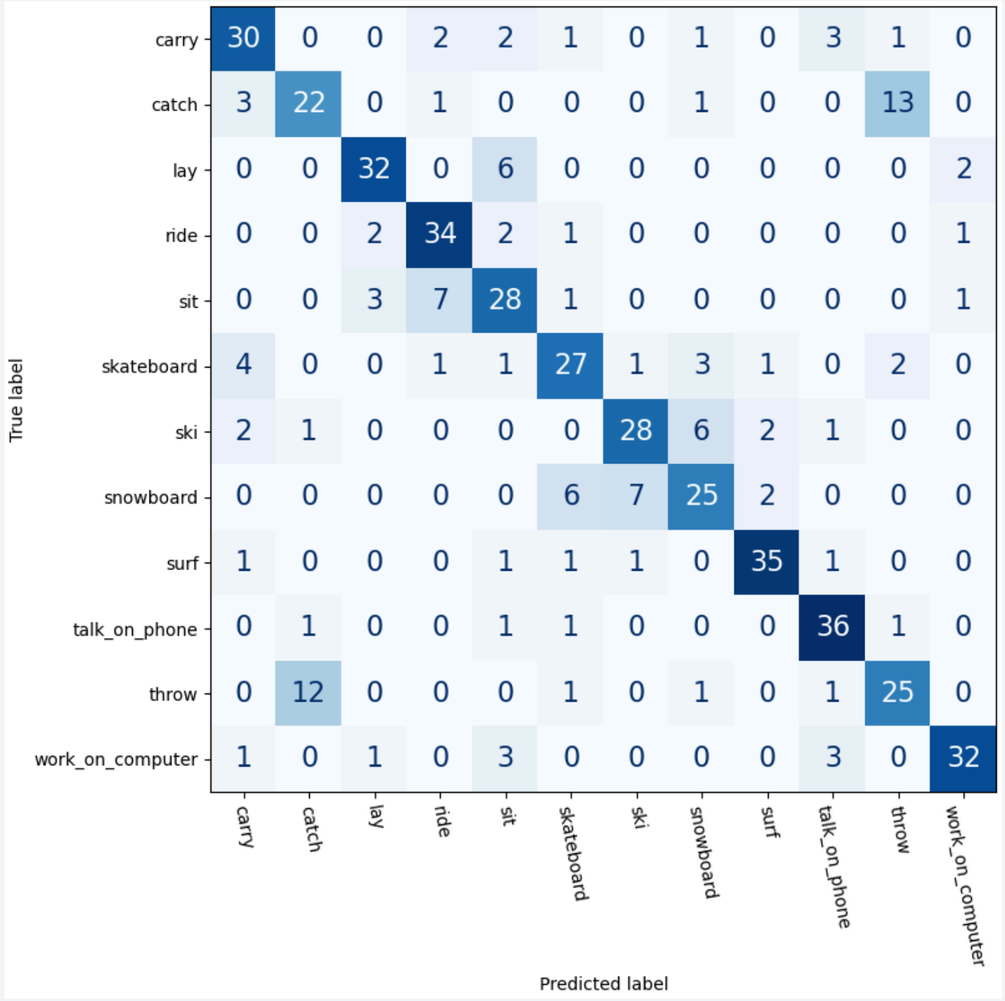


EXPRERIMENT 4

Method 1 / Accuracy: 0.733



Method 2 / Accuracy: 0.738



RESULTS AND ANALYSIS

Discussion and analysis

1. In experiment 1, it is obvious to find that both methods perform poorly with low accuracy. → label dependency
2. After modifying the dataset, the accuracy of both methods have increased noticeably. → Method 1 slightly outperforms Method 2.
3. Utilize pretrained weights in Method 1 to perform logistic regression. → In comparison to original one, the accuracy increases from 0.677 to 0.750.
4. Utilize additional classifiers to classify similar actions. However, the result of the accuracy is below our expectation.

RESULTS AND ANALYSIS

Limitation of your work

1. Bounding boxes are essential for the model to perform further recognition.
2. It can only recognize one action per image. In other words, our model can only apply to pictures containing only one person and one object.
3. AlexNet might be too simple for function approximation and not able to extract enough useful features.

RESULTS AND ANALYSIS

apply the model/method to practical use.

1. Obtain agent, object, union, and intersection pictures.
2. Input action and picture id
3. Loads a single image and extracts features.
4. Loads the trained logistic regression model.
5. Predicts the action category of the image and show the image.

```
def test_one_image(action, image_id, model_name):
    # Load the model
    model_path = f'model/{model_name}'
    if not os.path.exists(model_path):
        raise FileNotFoundError(f"Model file not found: {model_path}")

    with open(model_path, 'rb') as f:
        classifier = pickle.load(f)

    image_id_path = f'data/test/{action}/{image_id}'
    if not os.path.exists(image_id_path):
        raise FileNotFoundError(f"Image path not found: {image_id_path}")

    image_types = ['agent.jpg', 'object.jpg', 'intersection.jpg', 'union.jpg']
    real_image_type = os.listdir(image_id_path)
    images = []
    for image_type in image_types:
        # If there is no intersection, use white image
        if image_type not in real_image_type:
            image_path = 'white.jpeg'
        else:
            image_path = os.path.join(image_id_path, image_type)
        image = Image.open(image_path)
        images.append(image)

    show_image_path = os.path.join(image_id_path, 'union.jpg')
    if os.path.exists(show_image_path):
        show_image = Image.open(show_image_path)
        plt.imshow(show_image)
        plt.title('Picture')
        plt.axis('off')
        plt.show()
    components = extract_feature(images).reshape(1, -1)
    probabilities = classifier.predict_proba(components)
    predicted_action = actions[classifier.predict(components)[0]]
    print(f"Predicted action: {predicted_action}")
```

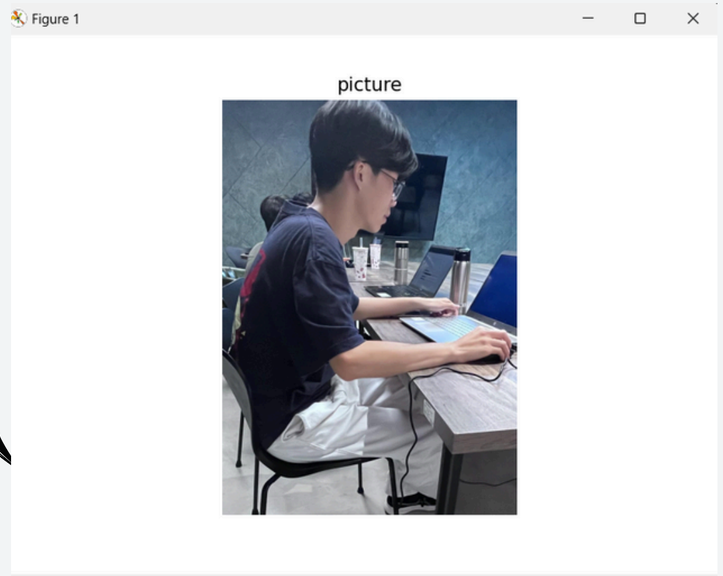
RESULTS AND ANALYSIS

apply the model/method to practical use.

input the action and the image_id(save the image as expected, agent, object, intersection, union)

```
PS C:\Users\user\Downloads\AI-Final-Project-main\AI-Final-Project-main> python.exe .\main.py
請輸入 action: sit
請輸入 image_id: 001
```

show the picture



show the action of the picture

```
C:\Users\user\Downloads\AI-Final-Project-
cRegression from version 1.3.2 when using
https://scikit-learn.org/stable/model_per
  warnings.warn(
sit
```

CONCLUSION

Future Prospect

1. Utilize YOLO to detect human and objects respectively in images.
2. Establish pair-wise connections between human and multiple objects and crop corresponding union regions as inputs.
3. Replace with deeper networks to enhance generalizability.

Experiment Conclusion

1. Data observation and preprocessing is important and may greatly influence the accuracy
2. Deep neural networks combined with logistic regression can obtain the best results on the task

CONTRIBUTION OF EACH MEMBER

何義翔(25%):method2、ppt、github
王裕昕(25%):method1、ppt、github
邱冠崴(25%):method1、ppt、github
林悦揚(25%):method2、ppt、video

GITHUB LINK

https://github.com/wonderer-kale/Analysis-of-Two-Action-Recognition-Methods-with-Still-Images

REFERENCE

1. Y. -W. Chao, Y. Liu, X. Liu, H. Zeng and J. Deng, "Learning to Detect Human-Object Interactions," 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 2018, pp. 381-389, doi: 10.1109/WACV.2018.00048.
2. D. Girish, V. Singh and A. Ralescu, "Understanding action recognition in still images," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1523-1529, 2020, doi: 10.1109/CVPRW50498.2020.00193.
3. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan,P. Doll´ar, and C. L. Zitnick, “Microsoft coco: Common objects incontext,” in Proceedings of ECCV, pp. 740–755, 2014.
4. COCO dataset website: https://cocodataset.org/#home

THANK'S FOR WATCHING