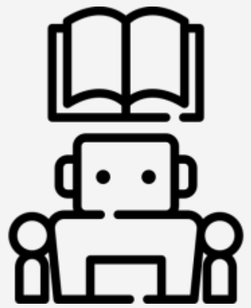
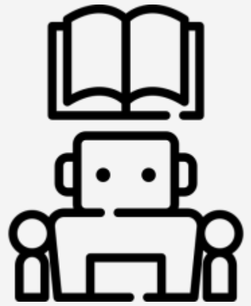
Machine Learning Term Project #5

비디오를 이용한 사람의 행동 분류기



20011844 안수경

Overview

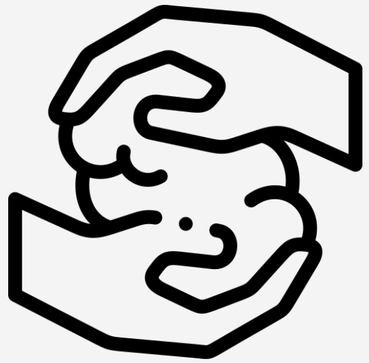


[Overview] - 프로젝트 방향성 이해하기

프로젝트의 목적

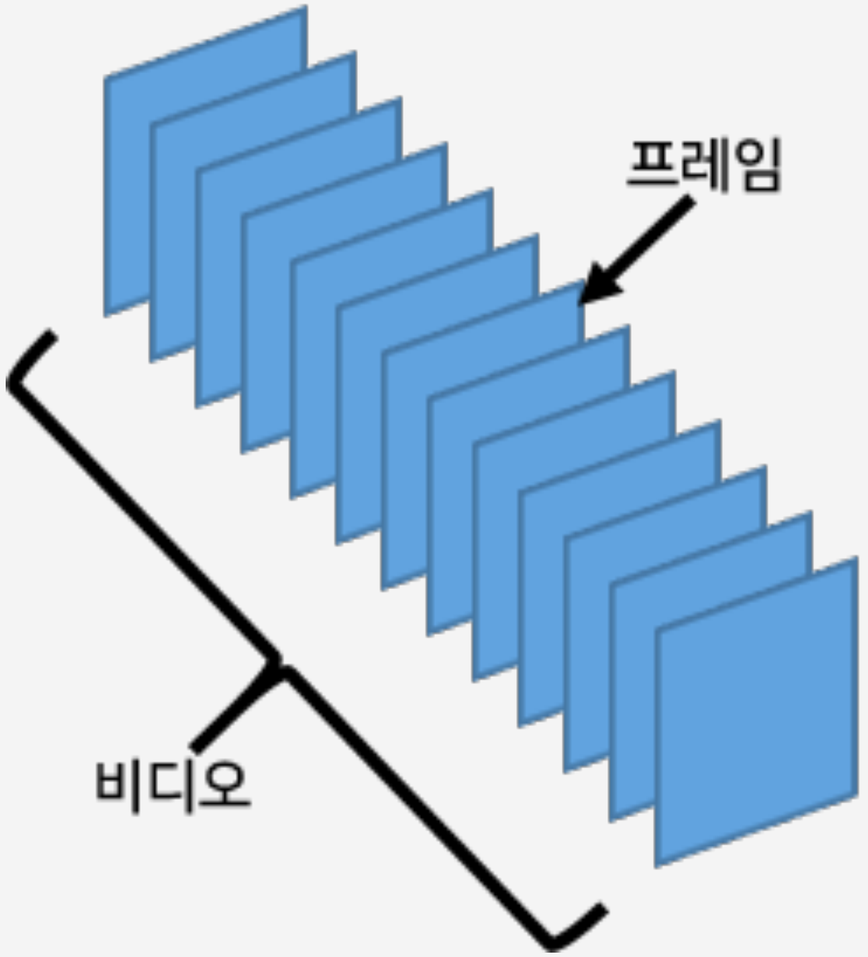
본 프로젝트를 통해 3D 비디오 데이터를

Handcrafted Feature 로 기술하는 법을 알 수 있다.



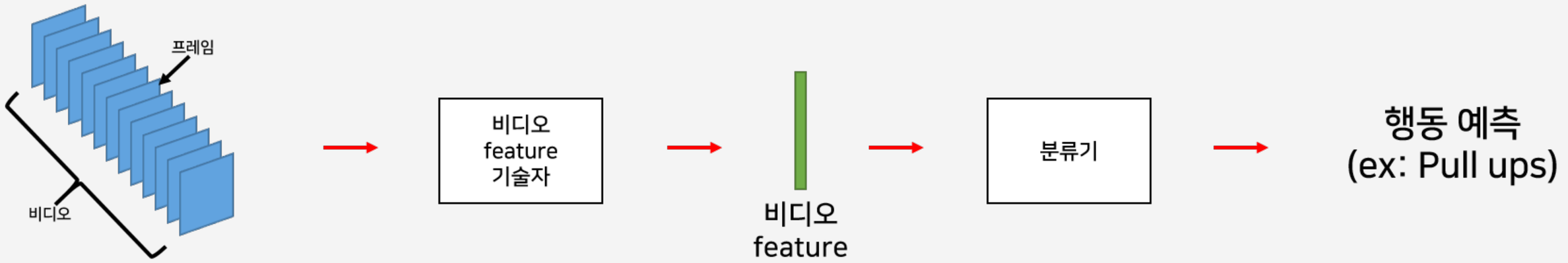
[Overview] - 3D 데이터 다루기 - 비디오를 이용한 사람의 행동 분류기

Video란?



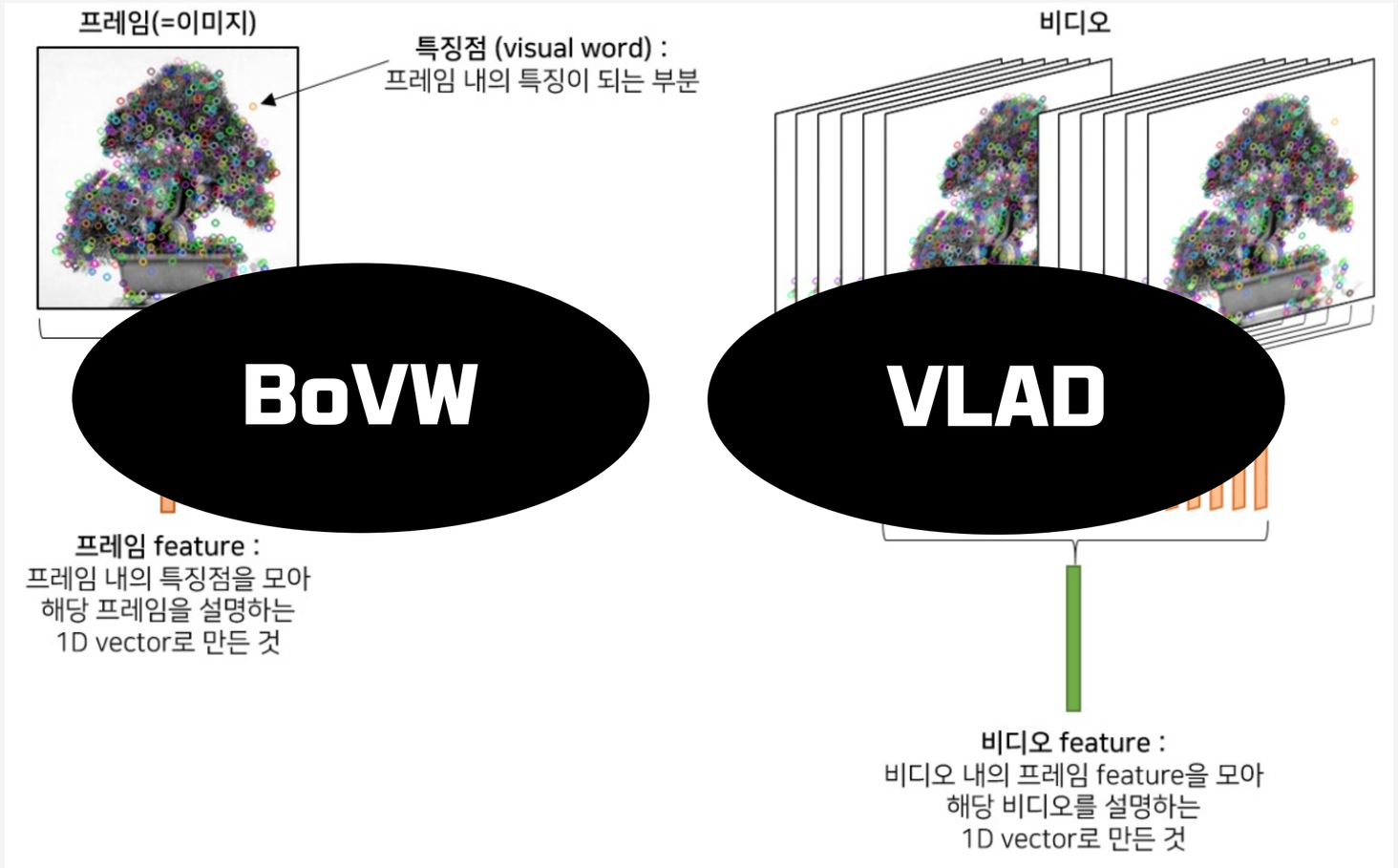
[Overview] - 3D 데이터 다루기 - 비디오를 이용한 사람의 행동 분류기

Video를 이용한 행동 분류



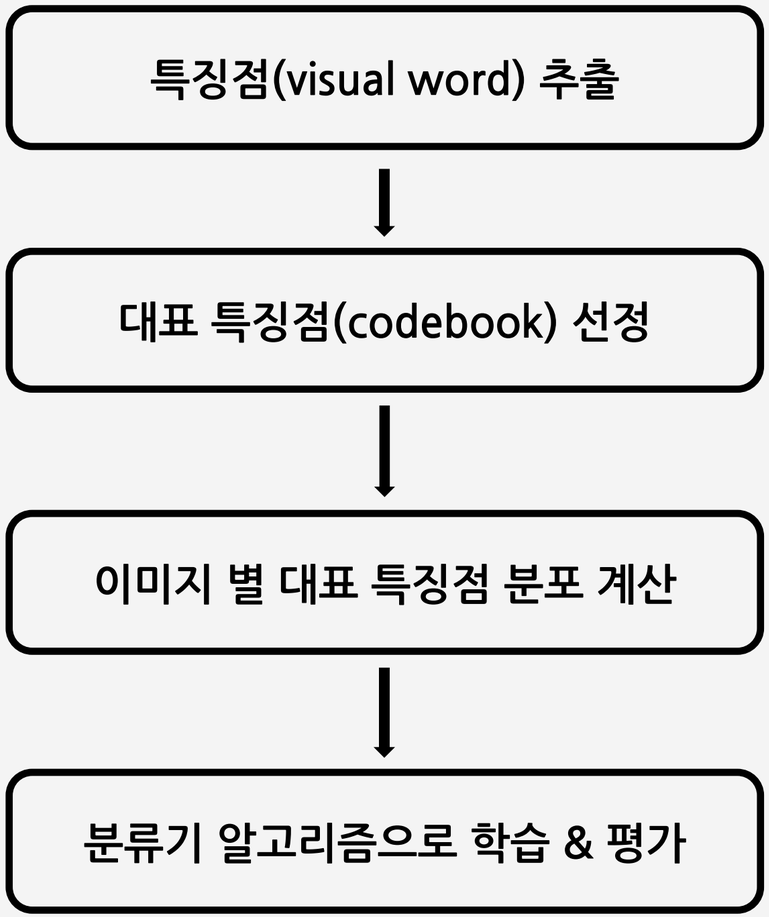
[Overview] - 3D video feature

비디오 내 단일 프레임 피쳐 추출 방법 (Frame Feature Extraction)



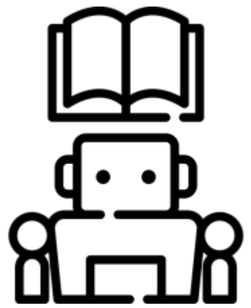
[Overview] - 3D video feature

비디오 내 단일 프레임 피쳐 추출 방법 (Frame Feature Extraction)



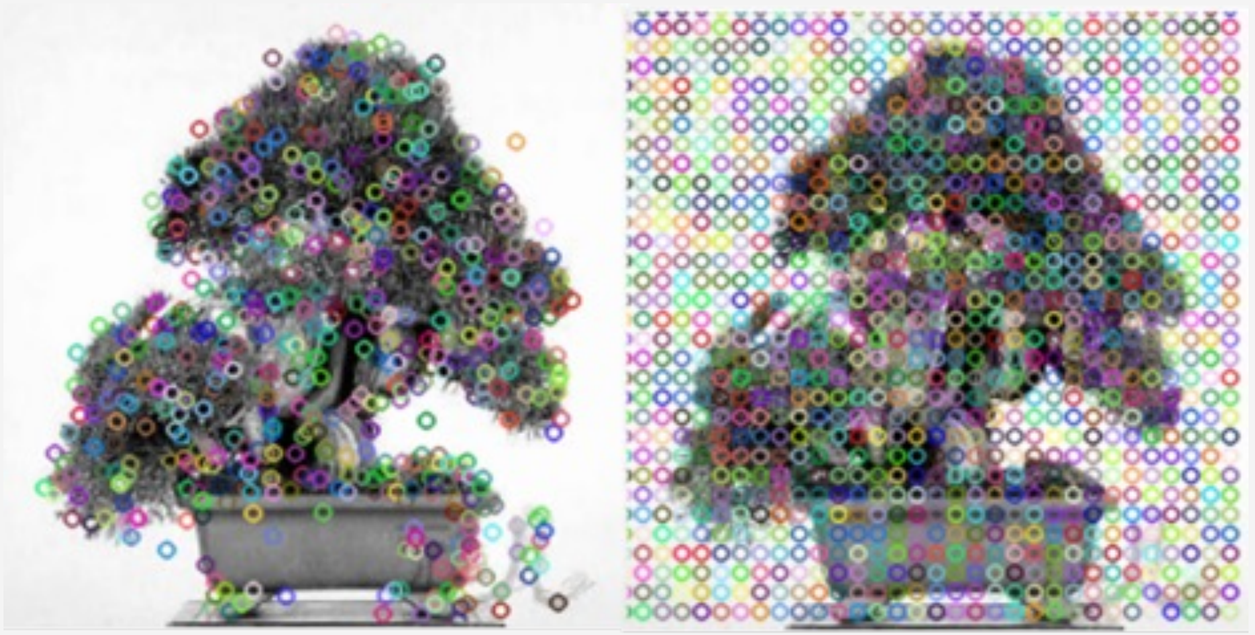
Empty Module #1

특징점(visual word) 추출



[Empty Module #1] - 3D video feature

특징점(visual word) 추출



SIFT

DenseSIFT

Memory ↑

성능 ↑

이미지 내의 픽셀 변화가 큰 부분을 추출

일정 픽셀 간격으로 얻은 위치를 모두 사용하여 추출

[Empty Module #1] - 3D video feature

특징점(visual word) 추출

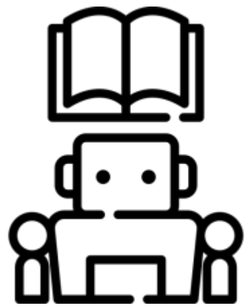
```
def computeSIFT(data, dense=False):
    x = {}
    for i in range(0, len(data)):
        if dense: # DenseSIFT 추출
            img = data[i]
            step_size = 8
            kp = [cv2.KeyPoint(x, y, step_size) for x in range(0, img.shape[0], step_size) for y in range(0, img.shape[1], step_size)] # keypoint 생성
            sift = cv2.SIFT_create()
            kp, desc = sift.compute(img, kp) # Dense SIFT는 위에서 생성한 keypoint를 사용해 compute 만을 진행

        else: # 기본 SIFT 추출
            sift = cv2.SIFT_create()
            img = data[i]
            kp, desc = sift.detectAndCompute(img, None)
        x.update({i : desc})

    return x
```

Empty Module #2

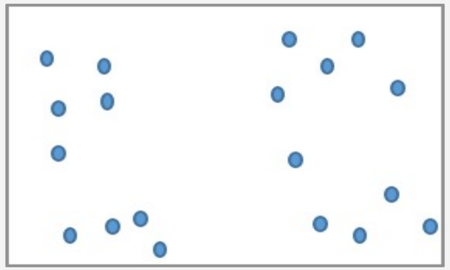
대표 특징점(codebook) 선정



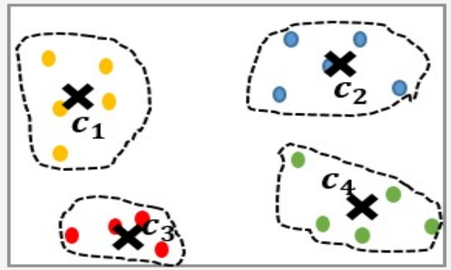
[Empty Module #2] - 3D video feature

대표 특징점(codebook) 선정

모든 프레임에 대해
특징점 추출



K-means
clustering



! 학습 비디오의 프레임에서 얻은
특징점만을 활용하여 선정

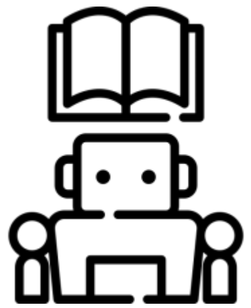
[Empty Module #2] - 3D video feature

```python
def clustering(train_desc, test_desc=None, n_clusters=200):
    from sklearn.cluster import MiniBatchKMeans
    kmeans = MiniBatchKMeans(n_clusters=n_clusters, random_state = 0).fit(train_desc) # 군집화 알고리즘인 MiniBatchKMeans를 사용하여 n_clusters개의 대표 특징점 생성
    clusters = kmeans.cluster_centers_ # 대표 특징점 선정
    train_pred = kmeans.predict(train_desc)

    if test_desc is not None:
        test_pred = kmeans.predict(test_desc)
    else:
        test_pred = None
    return train_pred, test_pred, clusters, kmeans
```

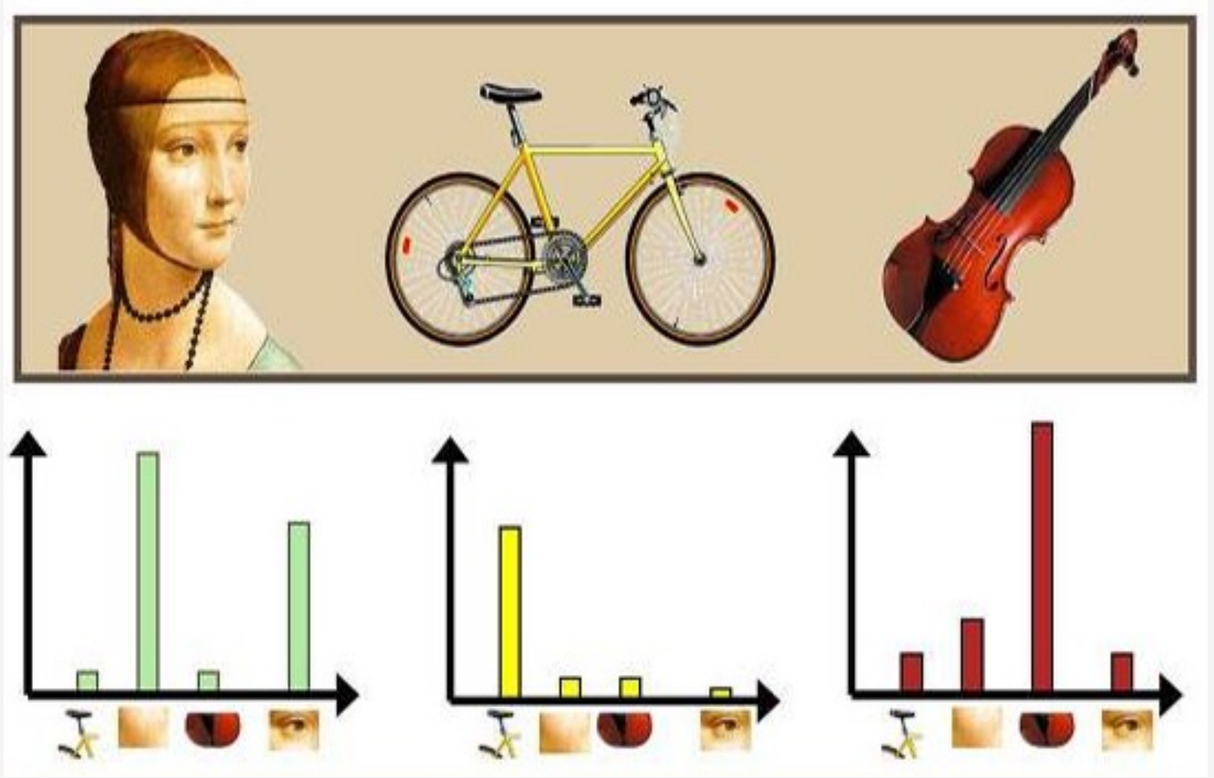
Empty Module #3. 4

이미지별 대표 특징점(codebook) 분포 계산



[Empty Module #4] - 이미지 별 대표 특징점(codebook) 분포 계산

BoVW (Bag of Visual Word) 피쳐 추출 방법

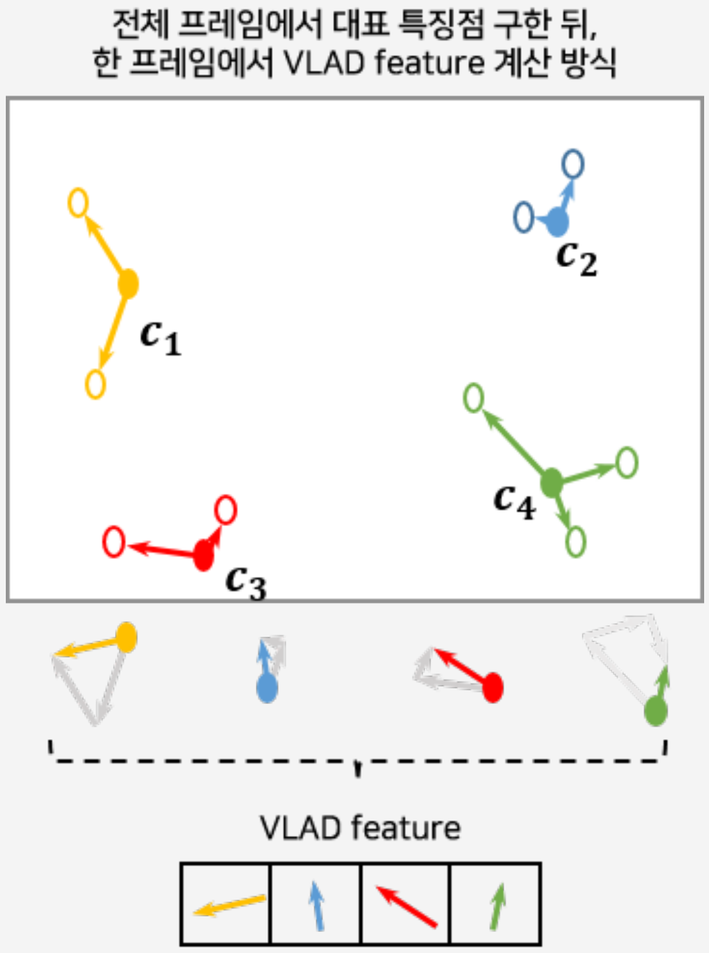


! frame 하나 당
1개의 히스토그램 feature

```
def BoW(alloc, n_cluster):
    # VLAD feature를 담기 위한 변수
    V, edge = np.histogram(alloc,bins = range(n_cluster+1), normed = True) # 이미지에서 추출된 특징점(visual word)이 대표 특징점(codebook)으로 할당된 정보 alloc의 histogram을 계산
    return V
```

[Empty Module #3] - 이미지 별 대표 특징점(codebook) 분포 계산

VLAD (Vector of Locally Aggregated Descriptors) 피쳐 추출 방법




```python
def VLAD(X, alloc, centers):
    m,d = X.shape
    k = centers.shape[0]
    # VLAD feature를 담기 위한 변수
    V=np.zeros([k,d])
    # 이미지에서 추출된 특징점(visual word) X와 이들이 대표 특징점(codebook)으로 할당된 정보 alloc을 이용해
    # 동일한 대표 특징점(codebook)으로 할당된 특징점(visual word)들의 벡터 합 계산해 V[i]에 저장
    for i in range(k):
        if np.sum(alloc==i)>0:
            V[i]=np.sum(X[alloc==i,:]-centers[i], axis=0)
    # 후처리 과정
    V = V.flatten()
    V = np.sign(V)*np.sqrt(np.abs(V))
    if np.sqrt(np.dot(V,V))!=0:
        V = V/np.sqrt(np.dot(V,V))
    return V
```

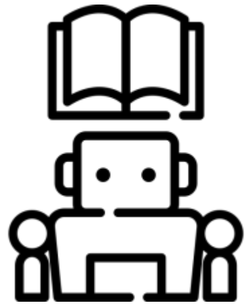
① 거리 순으로 할당

② 대표 특징점, 특징점 간의 벡터 차이 계산

③ 벡터 차이 값을 모두 합산

Empty Module #5. 6. 7

비디오 피쳐 추출 방법 (Video Feature Extraction)



[Empty Module #5, 6, 7] - 비디오 피쳐 추출 방법 (Video Feature Extraction)

Global averaging

Global voting

Video

[Empty Module #5] - 비디오 피쳐 추출 방법 (Video Feature Extraction)

Global averaging

```
print("\n\nSVM global averaging in frame")
start = time.time()

train_avg_video_desc = []
train_avg_video_label = []

for i in np.unique(train_global_id):
    tind = np.where(train_global_id==i)[0]
    train_avg_video_desc.append(np.mean(train_global_desc[tind], axis = 0))
    train_avg_video_label.append(np.mean(train_global_label[tind]))
train_avg_video_desc = np.asarray(train_avg_video_desc)
train_avg_video_label = np.asarray(train_avg_video_label)

test_avg_video_desc = []
for i in np.unique(test_global_id):
    tind = np.where(test_global_id==i)[0]
    test_avg_video_desc.append(np.mean(test_global_desc[tind], axis = 0))
test_avg_video_desc = np.asarray(test_avg_video_desc)

from sklearn.svm import SVC
clf = SVC(random_state = 0) # baseline : 0.27920
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100) # 0.29108
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.5) # 0.29702
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.55) # 0.29108
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.45) # 0.29900
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 1, tol = 0.45) # 0.27524
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.3) # 0.29702
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.35) # 0.29108
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.47) # 0.29900
clf = SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.445) # 0.29900 (best)

clf.fit(train_avg_video_desc, train_avg_video_label)
svm_predict = clf.predict(test_avg_video_desc).astype('int64')

saveFile(classinfo_arr[svm_predict][:,1], np.arange(len(test_list)), "svm_global_averaging")

print("\t{:3.2f}s\n\n".format(time.time()-start))
```

[Empty Module #5] - 비디오 피쳐 추출 방법 (Video Feature Extraction)

Global voting

```
print("\n\nSVM global voting in frame")
start = time.time()

from sklearn.svm import SVC
svm = SVC(random_state = 0)
svm.fit(train_global_desc, train_global_label)
svm_predict = svm.predict(test_global_desc)
svm_predict_vote = []
for i in np.unique(test_global_id):
    tind = np.where(test_global_id == i)[0]
    voted = mode(svm_predict[tind]).mode.item()
    svm_predict_vote.append(voted)
svm_predict_vote = np.asarray(svm_predict_vote)

saveFile(classinfo_arr[svm_predict_vote][:,1], np.arange(len(test_list)), "svm_global_voting")
print("\t{:3.2f}s\n\n".format(time.time()-start))
```

[Empty Module #5] - 비디오 피쳐 추출 방법 (Video Feature Extraction)

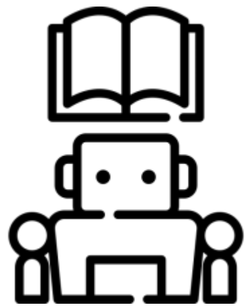
Video

```
print("\n\nSVM video descriptor")
start = time.time()

svm = SVC(random_state=0, probability = True, class_weight = 'balanced')
svm.fit(train_video_desc, train_video_label)
svm_predict = svm.predict(test_video_desc)

saveFile(classinfo_arr[svm_predict][:,1], test_video_id, "svm_video")
print("\t{:3.2f}s\n\n".format(time.time()-start))
```

Hyperparameter Tuning



Hyperparameter Tuning

Baseline

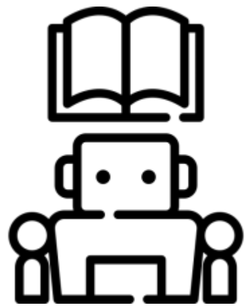
: SVC(random_state = 0)

Method	Score(baseline)
BoVW	0.21584
✓ VLAD	0.27920

Hyperparameter Tuning

Hyperparameter	score
SVC(random_state = 0)	0.27920(Baseline)
SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100)	0.29108
SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.5)	0.29702
SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.55)	0.29108
SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.45)	0.29900
SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 1, tol = 0.45)	0.27524
SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.3)	0.29702
SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.35)	0.29108
SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.47)	0.29900
⭐ SVC(random_state = 0, class_weight = 'balanced', probability = True, C = 100, tol = 0.445)	0.29900 (best)

Final Score



Final Score

svm_global_averaging (1).csv 10 days ago by Ahn Su-gyeong VLAD	0.27920	0.27920
svm_global_voting.csv 10 days ago by Ahn Su-gyeong BoVW	0.21584	0.21584

⬅ VLAD (0.27920)

⬅ BoVW (0.21584)

Final Score

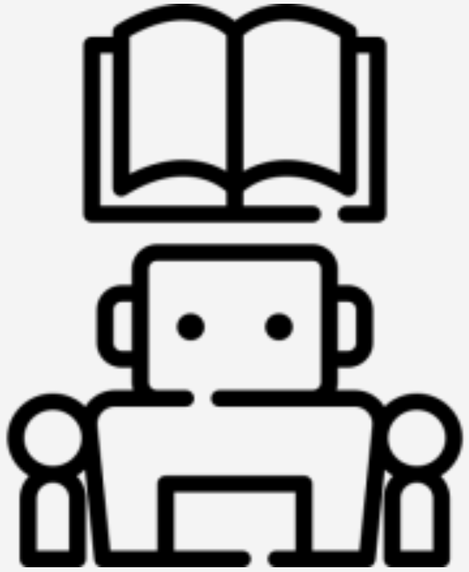
[2022-ML][P5]20011844_안수경

Python · 비디오를 이용한 사람의 행동 분류기

Notebook | Data | Logs | Comments (0) | Settings

Competition Notebook	Run	Private Score	Public Score	Best Score
비디오를 이용한 사람의 행동 분류기	420.1s	0.29900	0.29900	0.299 V13

← best score (0.29900)



THANK YOU

Machine learning term project #5