

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación  
Tratamiento de Señales Visuales (TSV)

Grado en Ingeniería Informática  
Introducción a la Visión Artificial (IVA)

# Práctica 3

## Reconocimiento de Escenas con modelos Bag-of-Words



- Reconocimiento de escenas mediante el uso de descriptores y clasificadores basados en bag of words



entrenamiento

Características

BoW

KNN/SVM

...

Clasificador  
Entrenado

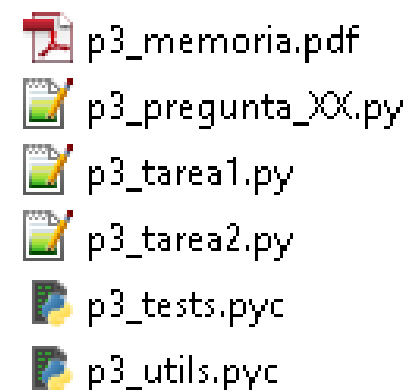
clasificación

¿Categoría?



Imagen test

- Descargue los ficheros de código Python disponibles en Moodle
  - Fichero `p3_bow_code.zip`
- Complete las tareas utilizando los ficheros `*.py` contenidos en el ZIP
- La entrega se **realiza por parejas** y constará de:
  - Memoria en PDF (1 fichero)  
+ código Python adicional
  - Ficheros código Python de las tareas realizadas  
(hasta 2 ficheros)
  - Fichero `p3_tests.pyc` y `p3_utils.pyc`
  - **No entregar carpetas 'dataset' y 'test\_data'**



No es necesario entregar todas las tareas

- Tarea 0 – Descargar material e incluir nombre en ficheros
- Tarea 1 – Implementar el modelo BOW
- Tarea 2 – Implementar descriptores Tiny/HOG
- Tarea 3 – Responder a las preguntas en memoria
- Funciones prohibidas:
  - Cualquier función que no sea del paquete de `numpy` y `scipy`
  - Se pueden utilizar las funciones recomendadas de `skimage`
  - Para normalización de imágenes, no se pueden utilizar las funciones `img_as_float64`, `img_as_float32`, y `img_as_float`.  
Se debe realizar la operación manualmente.

Aumento de complejidad con respecto a la práctica anterior y mayor número de preguntas en memoria

- En cada fichero debe poner la ruta donde se encuentra el dataset que ha descargado de Moodle

```
p3_tarea2.py 1 • p3_tarea1.py 1 X
p3_2122 > p3_tarea1.py > obtener_bags_of_words
33 # https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
34 """
35 vocabulario = np.empty(shape=[vocab_size,list_img_desc[0].shape[1]]) # iniciamos la variable de salida (
36
37 #...
38
39 return vocabulario
40
41 def obtener_bags_of_words(list_img_desc, vocab):
42 """
43 # Esta función obtiene el Histograma Bag of Words para cada imagen
44 #
45 # Argumentos de entrada:
46 # - list_img_desc: Lista 1xN con los descriptores de cada imagen. Cada posición de la lista
47 #   contiene (MxD) numpy arrays que representan UNO O VARIOS DESCRIPTORES
48 #   extraídos sobre la imagen
49 # - M es el número de vectores de características/features de cada imagen
50 # - D el número de dimensiones del vector de características/feature.
51 # - vocab: Numpy array de tamaño [vocab_size, D],
52 #   que contiene los centros de los clusters obtenidos por K-Means.
53 #
54 # Argumentos de salida:
55 # - list_img_bow: Array de Numpy [N x vocab_size], donde cada posición contiene
56 #   el histograma bag-of-words construido para cada imagen.
57 #
58 """
59 # iniciamos la variable de salida (numpy array)
60 list_img_bow = np.empty(shape=[len(list_img_desc),len(vocab)])
61
62 #...
63
64 return list_img_bow
65
66 if __name__ == "__main__":
67     dataset_path = './datasets/scenes15/'
68     print("Practica 3 - Tarea 1 - Test autoevaluación\n")
69     print("Tests completados = " + str(test_p3_tareal(dataset_path,stop_at_error=False,debug=False))) #analiz
70     #print("Tests completados = " + str(test_p3_tareal(dataset_path,stop_at_error=True,debug=True))) #analiz
```

C:\practicatsv\datasets\scenes15

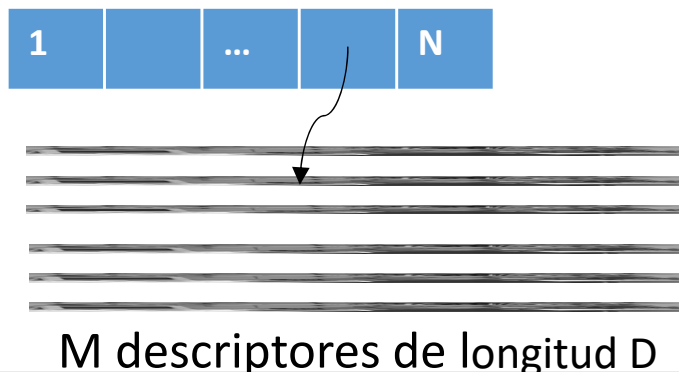
Nombre

Bedroom  
Coast  
Forest  
Highway  
Industrial  
InsideCity  
Kitchen  
LivingRoom  
Mountain  
Office  
OpenCountry  
Store  
Street  
Suburb  
TallBuilding

dataset\_path = 'C:/practicatsv/datasets/scenes15/'

- Implementar el modelo Bag-Of-Words (`p3_tarea1.py`)
  - Función `construir_vocabulario(list_img_desc, vocab_size=10, max_iter=300)`
  - Consideraciones:
    - Es resultado es el vocabulario en un array `numpy` con dimensión `[vocab_size, D]`
  - Funciones recomendadas:
    - Para agrupar, utilice la función `sklearn.cluster.Kmeans` con el parámetro `random_state=0`. En esta función utilice los parámetros `vocab_size` y `max_iter` de la función `construir_vocabulario`
    - Para convertir la lista en arrays `numpy`, utilice `numpy.concatenate`

Lista con descriptores para N imágenes



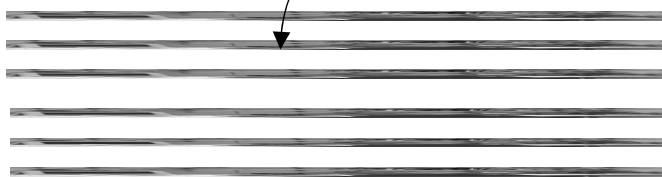
Construir  
vocabulario



=====  
=====  
=====  
vocab\_size centroides  
de longitud D  
(representantes grupos de descriptores)

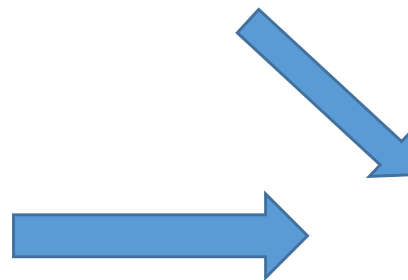
- Implementar el modelo Bag-Of-Words (`p3_tarea1.py`)
  - Función `obtener_bag_of_words(list_img_desc, vocab)`
  - Consideraciones:
    - El resultado es un array `numpy` con dimensión  $[N, \text{vocab\_size}]$  donde cada fila es el histograma BOW (normalizado) de cada imagen
  - Funciones recomendadas:
    - Para calcular distancias utilice `scipy.spatial.distance.cdist` con valores por defecto.

Lista con descriptores  
para N imágenes

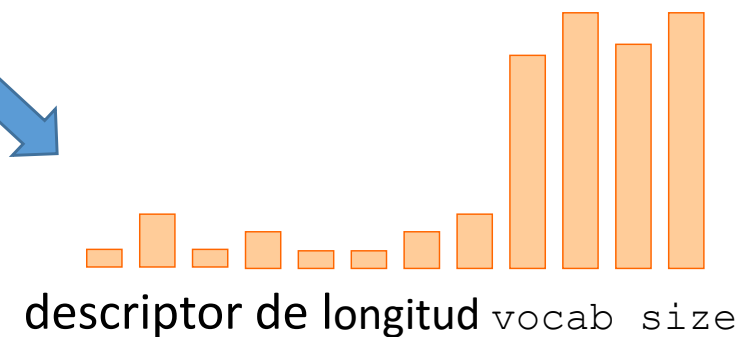


Cada imagen se define como  
M descriptores de longitud D

Vocabulario  
BOW  
(`vocab_size` x D)



Repetir proceso para todas  
las imágenes de la lista



- Implementar extracción de descriptores (p3\_tarea2.py)

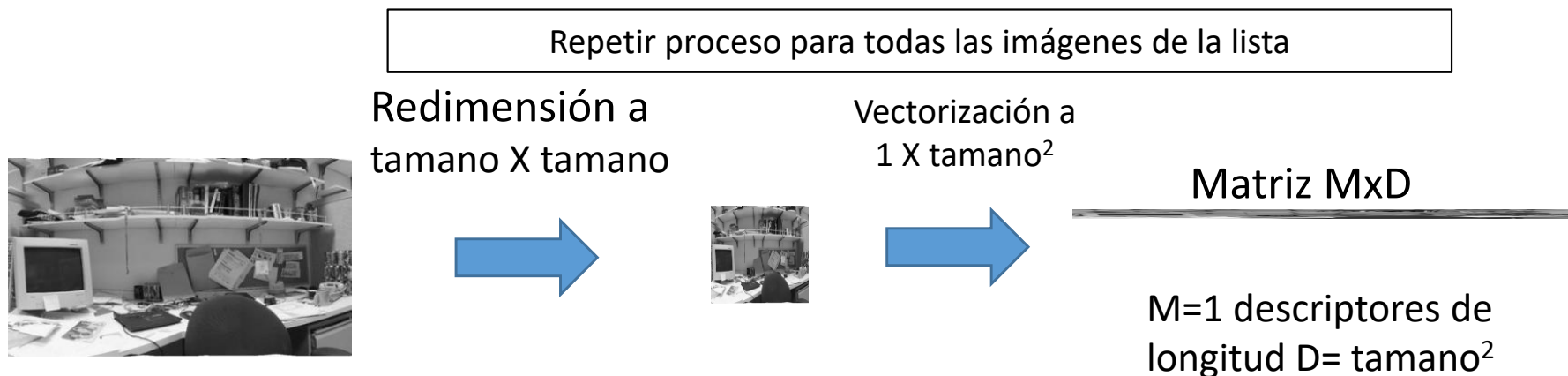
- Función `obtener_features_tiny(path_imagenes, tamaño=16)`

- Consideraciones:

- Convierta cada imagen a gris, formato `float` y en el rango `[0,1]`
    - La salida de esta función es una lista 1xN, donde cada posición es un `numpy array` con los descriptores calculados para cada imagen

- Funciones

- Recomendadas: `numpy.reshape`, `skimage.io.imread`, `skimage.color.rgb2gray`, `skimage.transform.resize`
    - Prohibidas: Cualquiera que no sea de `numpy` ó `scipy`





- Implementar extracción de descriptores (p3\_tarea2.py)

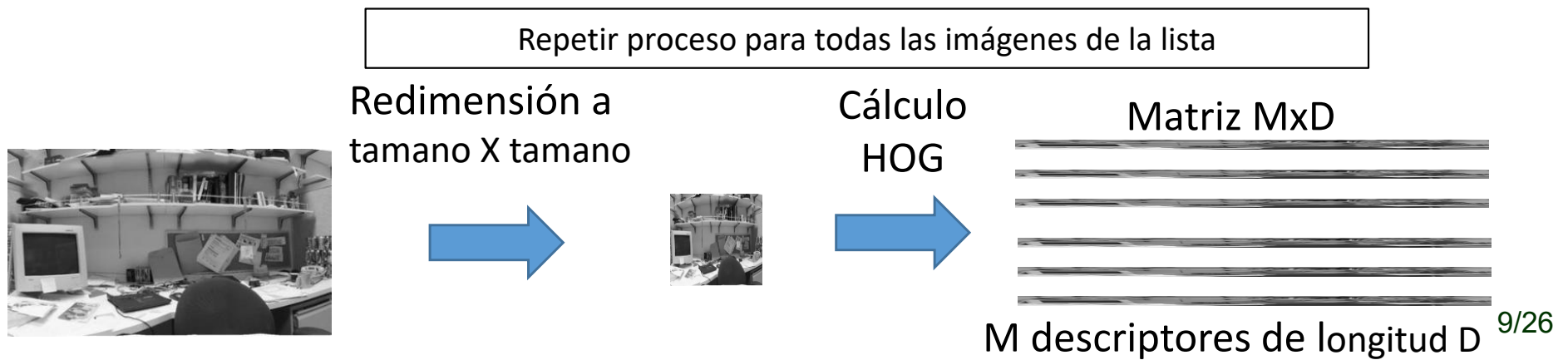
- Función `obtener_features_hog(path_imagenes, tamaño=100, orientaciones=9, pixeles_por_celda=(8, 8), celdas_bloque=(4, 4))`

- Consideraciones:

- Convierta cada imagen a gris, formato `float` y en el rango `[0,1]`
    - La salida de esta función es una lista 1xN, donde cada posición es un `numpy array` con los descriptores calculados para cada imagen

- Funciones

- Para calcular HOG, utilice `skimage.feature.hog` con los parámetros `orientations=orientaciones`, `pixels_per_cell=pixeles_por_celda`, `cells_per_block=celdas_bloque`, `feature_vector=False` (resto por defecto)
    - Recomendadas (Skimage): `io.imread`, `color.rgb2gray`, `transform.resize`



- Responda a las preguntas en un fichero “preguntas\_P3.docx”
  - No necesita realizar funciones nuevas, el trabajo se basa en realizar experimentos con la funcionalidad que ha generado
  - Se sugiere emplear búsqueda en grid para encontrar parámetros óptimos en cada pregunta.
  - Puede necesitar repasar algunos conceptos teóricos
  - Utilice figuras y tablas para resumir experimentos
- Ficheros a generar
  - Fichero “preguntas\_P3.pdf” en formato PDF con las respuestas
  - Genere tantos ficheros Python como necesite con el formato `p3_pregunta_XX.py` donde `XX` es el número de pregunta
- Criterios generales de evaluación: texto conciso y claro; relación con teoría; numeración de diagramas, figuras y tablas; referencias a figuras y tablas en texto; referencias a fuentes externas (si procede); errores ortográficos.
- Seis (6) preguntas en total agrupadas por temática.

### P3.1 Cree un esquema de clasificación de imágenes con los siguientes detalles:

- + Características: HOG con parámetro tam=100
- + Modelo: BOW con max\_iter=10
- + Clasificador: KNN con  $\bar{K}=5$
- + Otros: Ratio train\_test=0.20, y Máx número datos por categoría: 200

Debe utilizar las funciones para obtener features en `p3_tarea2.py`, funciones `construir_vocabulario/obtener_bags_of_words` (fichero `p3_tarea1.py`), y función `load_image_dataset` (fichero `p3_utils.py`). Para construir este esquema puede basarse en `sklearn.model_selection.train_test_split`, `sklearn.neighbors.KNeighborsClassifier`

Y aplíquelo sobre el dataset `scene15` disponible en el material de la práctica con una partición determinista. A continuación, responda a las siguientes preguntas:

3.1.1 Compare los resultados obtenidos al utilizar las características HOG y Tiny (con parámetro tam=64), en términos de rendimiento de clasificación sobre los datos de entrenamiento y test. Utilice valores por defecto indicados en el enunciado con un tamaño de diccionario de 50. **(0.75 puntos)**

3.1.2 Varíe el tamaño del diccionario BOW (hasta un valor máximo de 200) y estudie el rendimiento de clasificación sobre los datos de entrenamiento y test. Utilice valores por defecto indicados en el enunciado. **(0.75 puntos)**

3.1.3 Varíe el número de vecinos del clasificador KNN (hasta 21) con HOG y estudie el rendimiento de clasificación sobre los datos de entrenamiento y test. Muestre resultados visuales de aciertos/errores (se recomienda la función `create_webpage_results`). Utilice valores por defecto indicados en el enunciado con el tamaño de diccionario BOW óptimo obtenido en la pregunta 3.1.2. **(1.0 puntos)**

### P3.2 Cree un esquema de clasificación de imágenes con los siguientes detalles:

- + Características: HOG con parámetro tam=100
- + Modelo: BOW con max\_iter=10
- + Clasificador: SVM (lineal)
- + Otros: Ratio train\_test=0.20, y Máx número datos por categoría: 200

Debe utilizar las funciones para obtener features en `p3_tarea2.py`, funciones `construir_vocabulario/obtener_bags_of_words` (fichero `p3_tarea1.py`), y función `load_image_dataset` (fichero `p3_utils.py`). Para construir este esquema puede basarse en `sklearn.model_selection.train_test_split`, `sklearn.svm`

Y aplíquelo sobre el dataset `scene15` disponible en el material de la práctica con una partición determinista. A continuación, responda a las dos siguientes preguntas:

3.2.1 Varíe el tamaño del diccionario BOW (hasta un valor máximo de 200) y estudie como varía el rendimiento de clasificación sobre los datos de entrenamiento y test. Utilice valores por defecto indicados en el enunciado. **(1.0 puntos)**

3.2.2 Investigue y compare los distintos tipos de kernels no lineales para clasificadores SVM (e.g., rbf y poly) sobre los datos de entrenamiento y test. Razone por-que se obtiene unos resultados distintos a la pregunta anterior con una SVM lineal. Posteriormente, analice el rendimiento de la configuración con el mejor resultado y muestre resultados visuales de aciertos/errores (se recomienda la función `create_webpage_results`). Utilice valores por defecto indicados en el enunciado con el tamaño de diccionario BOW óptimo obtenido en la pregunta 3.2.1. **(1.25 puntos)**

### P3.3 Cree un esquema de clasificación de imágenes con los siguientes detalles:

- + **Características:** HOG con parámetro tam=100
- + **Modelo:** BOW con max\_iter=10
- + **Clasificador:** Random Forest
- + **Otros:** Ratio train\_test=0.20, y Máx número datos por categoría: 200

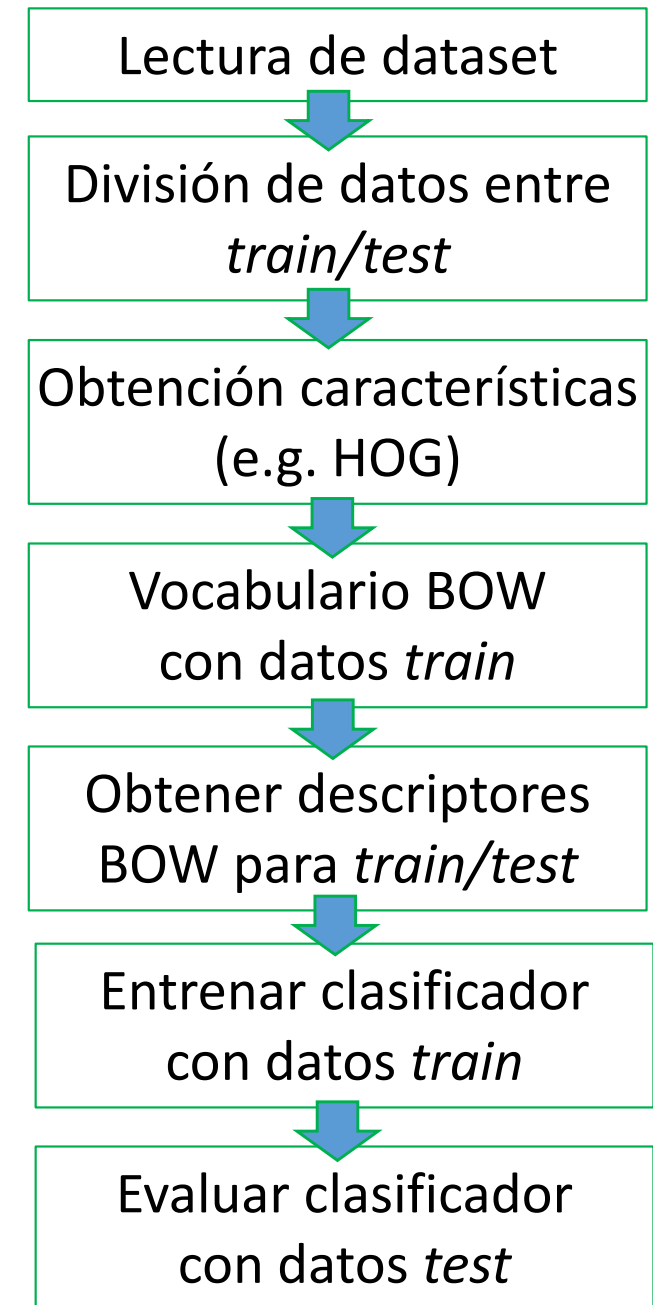
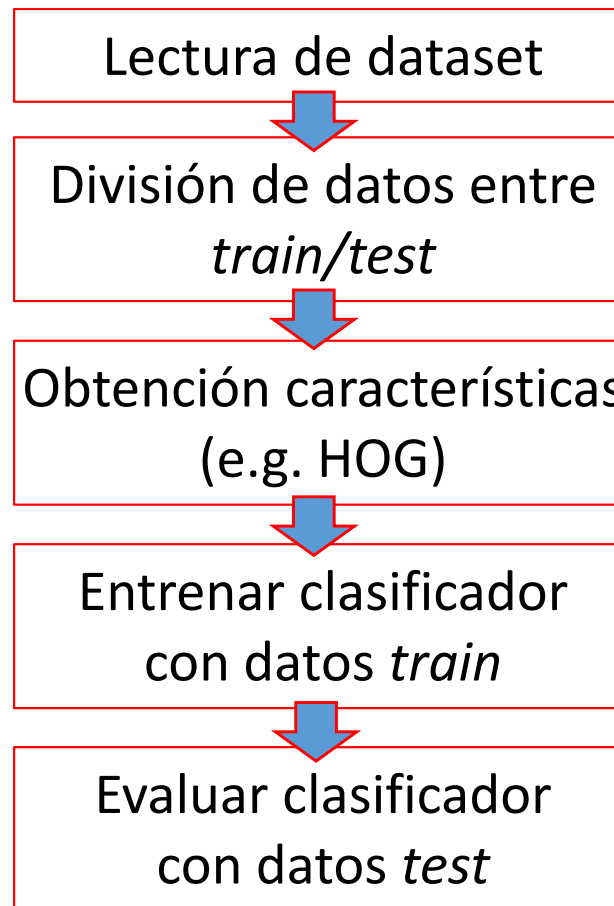
Debe utilizar las funciones para obtener features en `p3_tarea2.py`, funciones `construir_vocabulario/obtener_bags_of_words` (fichero `p3_tarea1.py`), y función `load_image_dataset` (fichero `p3_utils.pyc`). Para construir este esquema puede basarse en `sklearn.model_selection.train_test_Split`, `sklearn.ensemble.RandomForestClassifier`

Y aplíquelo sobre el dataset `scene15` disponible en el material de la práctica con una partición determinista. A continuación, responda a las dos siguientes preguntas:

3.3.1 Investigue la función `sklearn.ensemble.RandomForestClassifier` y seleccione solo uno de los parámetros disponibles. Compare y razone los resultados para distintos valores del parámetro seleccionado. Posteriormente, analice el rendimiento de la configuración con el mejor resultado y muestre resultados visuales de aciertos/errores (se recomienda la función `create_webpage_results`). Utilice valores por defecto indicados en el enunciado con el tamaño de diccionario BOW óptimo obtenido en la pregunta 3.2.1. **(1.25 puntos)**

- Las preguntas requieren construir varios sistemas de clasificación del dataset con tareas 1/2, clasificadores KNN/SVM/RF y características

- Opción 1:  
Raw/descriptores
- Opción 2:  
BOW descriptores



- Funciones útiles: lectura del dataset (fichero `p3_utils.pyc`) 1/2

```
data = load_image_dataset(container_path, *, description=None,
                           categories=None, load_content=True, shuffle=True, random_state=0,
                           resize_shape=None, max_per_category=None, debug=False):
```

Datos de imágenes

Etiquetas

Etiquetas (nombres)

Rutas de imágenes

-----

```
data : :class:`~sklearn.utils.Bunch`
       Dictionary-like object, with the following attributes.
```

```
data : list of str
       Only present when `load_content=True`.
       The raw image data to learn.
```

```
target : ndarray
        The target labels (integer index).
```

```
target_names : list
              The names of target classes.
```

```
DESCR : str
        The full description of the dataset.
```

```
filenames: ndarray
           The filenames holding the dataset
```

- Funciones útiles: lectura del dataset (fichero `p3_utils.pyc`) 2/2

```
data = load_image_dataset(container_path, *, description=None,
                           categories=None, load_content=True, shuffle=True, random_state=0,
                           resize_shape=None, max_per_category=None, debug=False):
```

```
#example 1 - Load dataset
# 100 images per category
# resize all images to (200,200))
dataset_path = './datasets/scenes15'
data = load_image_dataset(container_path=dataset_path, shuffle=True
                           load_content=True, resize_shape=(200,200),max_per_category=100)
```

```
#example 2 - Load dataset
# 200 images per category
# does not load images (only paths) - useful for computing features later
dataset_path = './datasets/scenes15'
data = load_image_dataset(container_path=dataset_path, shuffle=True,
                           load_content=False, max_per_category=200)
```

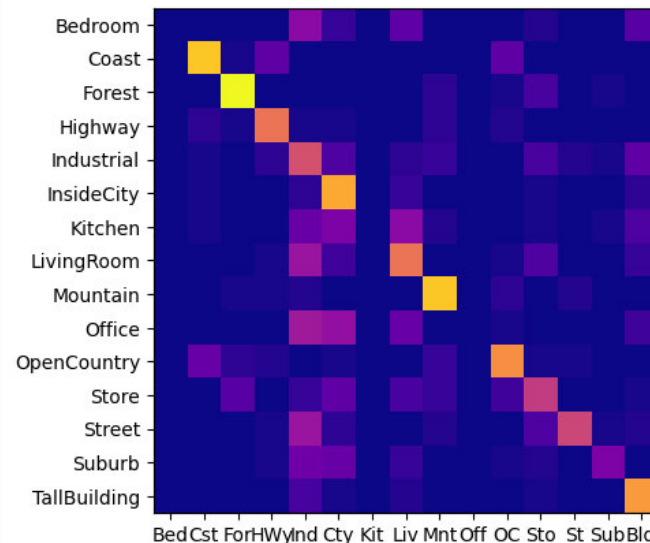


- Funciones útiles: visualización resultados (fichero `p3_utils.py`) 1/3


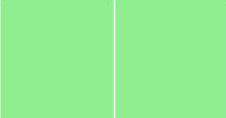
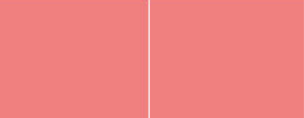





**`create_results_webpage(...)`:**

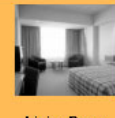
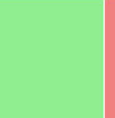
La función genera una página web html con resultados cualitativos y cuantitativos

scene classification results visualization



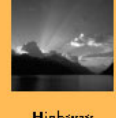
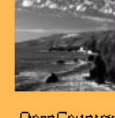
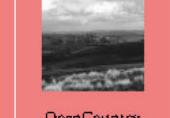
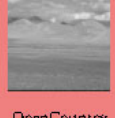
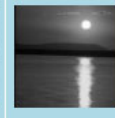
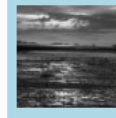
Accuracy (mean of diagonal of confusion matrix) is 0.513

Category name	Accuracy	Sample training images	Sample true positives	False positives with true label	False negatives with wrong predicted label
Bedroom	0.000				
Coast	0.836				



LivingRoom

TallBuilding



OpenCountry

OpenCountry

OpenCountry

Highway

- Funciones útiles: visualización resultados (fichero `p3_utils.pyc`) 2/3

```
confusion_matrix = create_results_webpage  
(train_image_paths, test_image_paths,  
train_labels, test_labels, categories, abbr_categories, predic  
ted_categories, name_experiment):
```

- `confusion_matrix` → matriz de confusión obtenida
- `train_image_paths` → Lista de Python con las rutas (relativas o absolutas) de las imágenes de entrenamiento
- `test_image_paths` → Lista de Python con las rutas (relativas o absolutas) de las imágenes de entrenamiento

- `train_labels`,
- `test_labels`
- `predicted_categories`

} Lista de Python con las categorías reales para los datos de train/test y predichas solamente para los datos de test

```
[ 'TallBuilding', 'TallBuilding', 'Kitchen', 'Bedro...  
> special variables  
> function variables  
0000: 'TallBuilding'  
0001: 'TallBuilding'  
0002: 'Kitchen'  
0003: 'Bedroom'  
0004: 'InsideCity'  
0005: 'Kitchen'  
0006: 'Kitchen'  
0007: 'InsideCity'  
0008: 'Kitchen'  
0009: 'TallBuilding'  
0010: 'Industrial'  
0011: 'Highway'  
0012: 'Kitchen'  
0013: 'Office'  
0014: 'Kitchen'  
0015: 'Kitchen'
```

- Funciones útiles: visualización resultados (fichero `p3_utils.pyc`) 3/3

```
confusion_matrix = create_results_webpage  
(train_image_paths, test_image_paths,  
train_labels, test_labels, categories, abbr_categories, predicted  
_categories, name_experiment):
```

```
# This is the list of categories / directories to use.
```

```
# The categories are sorted by alphabetical order
```

```
categories = ['Bedroom', 'Coast', 'Forest', 'Highway', 'Industrial',  
              'InsideCity', 'Kitchen', 'LivingRoom', 'Mountain', 'Office',  
              'OpenCountry', 'Store', 'Street', 'Suburb', 'TallBuilding']
```

```
# This list of shortened category names is used later for visualization.
```

```
abbr_categories = ['Bed', 'Cst', 'For', 'Hwy', 'Ind', 'Cty', 'Kit', 'Liv', 'Mnt',  
                  'Off', 'OC', 'Sto', 'St', 'Sub', 'Bld']
```

**name\_experiment** → string que describe el experimento realizado. Los resultados se guardan en la carpeta `./p3_results_webpage/name_experiment` (e.g. `name_experiment='HOG_BOW_SVM'`, genera la página html de resultados en `./p3_results_webpage/HOG_BOW_SVM/index.html`)

- Funciones útiles:

- Para dividir el dataset entre train/test (paquete sklearn)

- `sklearn.model_selection.train_test_split`

- Para clasificar, KNN y SVM del paquete sklearn

- KNN `sklearn.neighbors.KNeighborsClassifier`

- SVM `sklearn.svm`

- RF `sklearn.ensemble.RandomForestClassifier`

- Para evaluar el rendimiento el dataset

- `sklearn.neighbors.KNeighborsClassifier.score`

- `sklearn.svm.score`

- `sklearn.ensemble.RandomForestClassifier.score`

- Documentación:

- <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC>

- <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

- ¿Cómo comprobar si la implementación es correcta?
  - Se proporciona funciones que se pueden ejecutar cuando se desee
    - **test\_p3\_tarea1(...)** en el fichero `p3_tarea1.py`

```
Practica 3 - Tarea 1
Realizando tests para las funciones 'construir_vocabulario' y 'obtener_bags_of_words' de la tarea 1
La funcion es correcta si los resultados obtenidos tienen una tolerancia de 2 decimales con respecto a la salida correcta.

* Utilizando datos en fichero './p3_tarea1.data'
* Tests para 'construir_vocabulario':
  Testeando con tamano de vocabulario 5 (20 descriptores Tiny de dimension 256)... OK
  Testeando con tamano de vocabulario 5 (1620 descriptores HOG de dimension 144)... OK
  Testeando con tamano de vocabulario 10 (20 descriptores Tiny de dimension 256)... OK
  Testeando con tamano de vocabulario 10 (1620 descriptores HOG de dimension 144)... OK
* Tests para 'obtener_bags_of_words':
  Testeando con tamano de vocabulario 5 (20 descriptores tipo Tiny de dimension 256)... OK
  Testeando con tamano de vocabulario 5 (1620 descriptores tipo HOG de dimension 144)... OK
  Testeando con tamano de vocabulario 10 (20 descriptores tipo Tiny de dimension 256)... OK
  Testeando con tamano de vocabulario 10 (1620 descriptores tipo HOG de dimension 144)... OK
* Finalizado en 2.435 secs
* RESULTADO 'construir_vocabulario' con descriptor Tiny/HOG: 4/4 CORRECTOS ( 100.00% )
* RESULTADO 'obtener_bags_of_words' con descriptor Tiny/HOG: 4/4 CORRECTOS ( 100.00% )
Tests completados = True
```

- **test\_p3\_tarea2(...)** en el fichero `p3_tarea2.py`

```
Practica 3 - Tarea 2
Realizando tests para las funciones 'obtener_features_tiny' y 'obtener_features_hog' de la tarea 2
La funcion es correcta si los resultados obtenidos tienen una tolerancia de 2 decimales con respecto a la salida correcta.

* Utilizando datos en fichero './p3_tarea2.data'

* Tests para 'obtener_features_hog':
  Testeando con tamano 50... Imagen #0 #1 #2 #3 #4 #5 #6 #7 #8 #9
  Testeando con tamano 100... Imagen #0 #1 #2 #3 #4 #5 #6 #7 #8 #9
* Finalizado en 1.469 secs
* RESULTADO 'obtener_features_hog': 20/20 DESCRIPTORES CORRECTOS ( 100.00% )
Tests completados = True
```

No hay funciones de auto-evaluación para preguntas en la memoria pero...

- Accuracy/rendimiento de clasificación sobre datos de test (utilizando como entrenamiento 200 imágenes por categoría)
  - ~0% → funcionamiento erróneo
  - ~7% → rendimiento similar a un clasificador aleatorio
  - ~18% → rendimiento para características Tiny con KNN ( $k=5$ )  
Se puede aumentar ligeramente ajustando parámetro  $k$
  - ~40% → rendimiento para características HOG-BOW y KNN ( $k=5$ )  
Se puede aumentar ligeramente ajustando parámetro  $k$
  - ~50% → rendimiento para características HOG-BOW y SVM (lineal)
  - ~55% → rendimiento para características HOG-BOW y RF
  - ~70% → rendimiento para características HOG-BOW y SVM

Ajuste de parámetros SVM, tamaño de la imagen, extracción HOG y diccionario BOW.

- 2 sesiones de prácticas (20 horas): 4h presenciales + 16h no presenciales
- Sugerencia de realización:

TAREA	Horas presenciales	Horas no presenciales	Horas TOTAL	Semana de prácticas
Explicación	0.5 h	0 h	0.5 h	1
Tarea 1	1.5 h	3 h	4.5 h	1,2
Tarea 2	2 h	1 h	3 h	2
Memoria	0 h	12 h	12 h	2,3
TOTAL	4h	16h	20h	

TAREA	Max nota	Criterio evaluado (ver rúbrica en Moodle)
Tarea 1	2	Código: Ejecución (60%) + Diseño(40%)
Tarea 2	2	Código: Ejecución (60%) + Diseño(40%)
Memoria	6	Memoria: Claridad, exactitud de la respuesta y experimentos realizados por cada pregunta (100%)
P3.1	2.5	
P3.2	2.25	
P3.3	1.25	
<b>TOTAL</b>	<b>10</b>	

- Penalizaciones:
  - Por entrega de ficheros no acorde a las especificaciones: -0.5 puntos
  - Por entrega de memoria con formato incorrecto: -0.5 puntos
  - Por uso de funciones prohibidas: -50% tarea
  - Por entrega tardía (tras considerar los 4 días disponibles para cada pareja):
    - -25% (un día), -50% (dos días), -75% (tres días), -100%(>= días)



- Wikipedia  
[https://en.wikipedia.org/wiki/Bag-of-words\\_model](https://en.wikipedia.org/wiki/Bag-of-words_model)
- **Descriptor Histograma de Gradientes Orientados**
  - Richard Szeliski, “Computer Vision: Algorithms and Applications” 2020, **Sección 7.1.2.** <https://szeliski.org/Book/>
  - Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91-110
- **Modelo Bag-of-words**
  - Richard Szeliski, “Computer Vision: Algorithms and Applications” 2020, **Sección 6.2.** <https://szeliski.org/Book/>

- Esta práctica presenta un **incremento sustancial** en complejidad respecto a **prácticas anteriores**.
- El **trabajo efectivo y coordinado de la pareja** es crítico para poder realizar todos los ejercicios satisfactoriamente.
- Sugerencia:
  - Tareas 1 y 2: realizar conjuntamente
  - Preguntas memoria:
    - Analizar que tareas son comunes a todas las preguntas (leer dataset, extraer características,...)
    - Repartir las tareas entre la pareja y realizar individualmente
    - Escribir los resultados en la memoria individualmente
    - Realizar un análisis y revisión conjunto de los resultados en memoria