Práctica 3 – Reconocimiento de escenas con modelos Bag-of-Words

Fernandez Moreno, Jose-Luis – Ramasco Gorria, Pedro

3 PREGUNTASTAREAS OPCIONALES

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 K=5
+ Otros: Ratio train_test=0.20, y Máx número datos por categoría: 200

Para construir este esquema puede basarse en:
+ Función sklearn.model_selection.train_test_split
+ Función sklearn.neighbors.KNeighborsClassifier
+ Función obtener_features_hog (fichero p3_tarea2.py)
+ Función construir_vocabulario y obtener_bags_of_words (fichero p3_tarea1.py)
+ Función load_image_dataset (fichero p3_utils.pyc)

Y aplíquelo sobre el dataset de escenas scene15 disponible en el material de la práctica para gene-rar una partición determinista. A continuación, responda a las dos 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)

Tras ejecutar el código, obteniendo los resultados mostrados en la Figura 1 y sabiendo que hemos de usar los parámetros establecidos por el enunciado, podemos sacar las siguientes conclusiones:

```
Tamaño Diccionario =  50
Rendimiento TEST con Hog=  0.405
Rendimiento TRAIN con Hog =  0.5791666666666667
Rendimiento TEST con Tiny=  0.13166666666666665
Rendimiento TRAIN con Tiny =  0.18208333333333335
```

Figura 1: Ejecución 3.1.1

Vemos que el valor de test de HOG será de 0.405 y el valor de test de Tiny será de 0.131666…
Con estos resultados podemos llegar a la conclusión de que, en términos de rendimiento de clasificación de datos, HOG será superior a Tiny, esto indicara que HOG será mucho más efectivo para este problema de clasificación
Es por esto por lo que creemos que HOG es una elección mucho más efectiva en este contexto, sin embargo, es importante tener en cuenta el overfitting y considerar la generalización del modelo apartir de la evaluación en datos de entrenamiento.

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)

Para facilitar la ejecución del código y el llegar a la conclusión, hemos establecido un bucle que recorrerá una lista que está conformado por varios tamaños de vocabulario (tam_diccionario = [5, 15, 25, 50, 75, 100, 125, 150, 175, 200]).
Una vez ejecutado código y analizando los resultados podemos observar que por norma general mejoraran los resultados. Podemos observar que debido a que, al expandir el vocabulario, contamos con un mayor número de palabras que mejoran nuestra precisión en el reconocimiento, proporcionándonos así un vocabulario más diverso y enriquecido. No obstante, la ampliación del diccionario presenta una notable desventaja, que radica en el tiempo necesario para el procesamiento o computo.
En las siguientes figuras mostramos los mejores resultados para los valores de diccionario 5, 75 y 100.

```
Tamaño Diccionario =  5
Rendimiento TEST con Hog=  0.20666666666666667
Rendimiento TRAIN con Hog =  0.4375
Rendimiento TEST con Tiny=  0.11
Rendimiento TRAIN con Tiny =  0.12875
```

Figura 2: Tamaño Diccionario igual a 5

```
Tamaño Diccionario =  75
Rendimiento TEST con Hog=  0.4116666666666667
Rendimiento TRAIN con Hog =  0.6095833333333334
Rendimiento TEST con Tiny=  0.16
Rendimiento TRAIN con Tiny =  0.20708333333333334
```

Figura 3: Tamaño Diccionario igual a 75

```
Tamaño Diccionario =  200
Rendimiento TEST con Hog=  0.4266666666666667
Rendimiento TRAIN con Hog =  0.5995833333333334
Rendimiento TEST con Tiny=  0.185
Rendimiento TRAIN con Tiny =  0.2425
```

Figura 4: Tamaño Diccionario igual a 75

En el caso de la figura 4, será el que consideremos como el más optimo (tamaño diccionario 200) y procedemos a mostrar el accuracy generado por créate_webpage_results para este caso

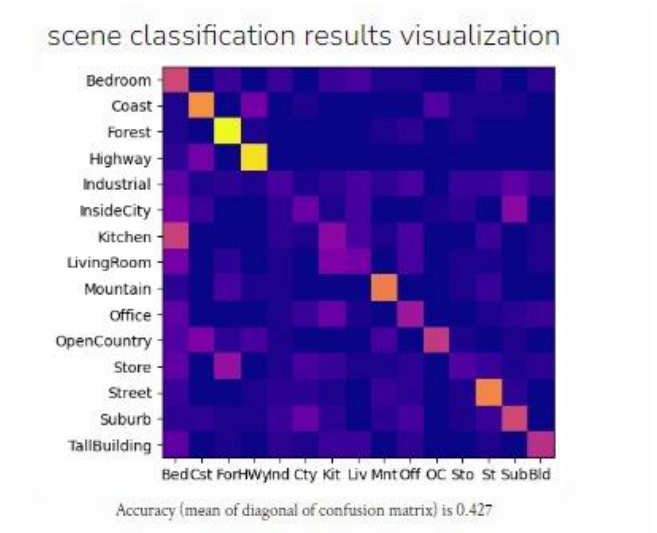

Figura 5: Matriz de confusión para tam_diccionario=200

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

Comenzamos fijando el tamaño de vocabulario a 200, pues ya hemos comprobado que es el que mejor nos funciona.
Tras esto modificaremos el número de vecinos con el que ejecutaremos KNN. Con un número de vecinos igual a 21 nos dará un accuracy de 43,5%.
Ademas apreciamos una mejora respecto al k=5 de los apartados anteriores.
Una vez tenemos todos los valores y numero de vecinos construimos y mostramos la matriz de confusión

```
Tamaño Diccionario =  200
Rendimiento TEST con Hog=  0.435
Rendimiento TRAIN con Hog =  0.525
Rendimiento TEST con Tiny=  0.21833333333333332
Rendimiento TRAIN con Tiny =  0.26708333333333334
```

Figura 6: Tamaño Diccionario igual a 200

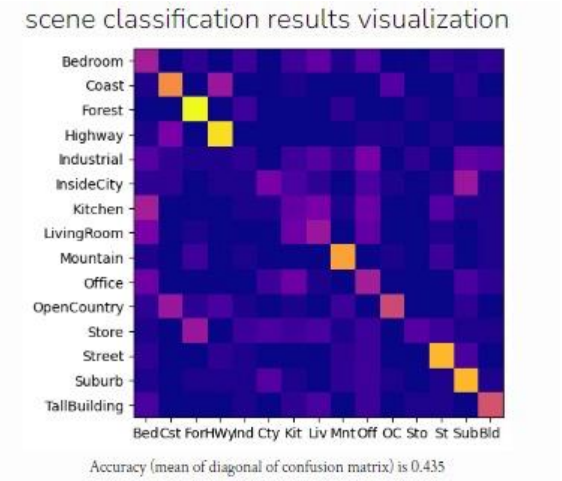

Figura 7: Matriz de confusión para tam_diccionario=200 y número de vecinos igual a 21

P3.2. Cree
un esquema de clasificación de imágenes completo con los siguientes detalles:
+Características: HOG con parámetro tam= 100
+ Modelo BOW sobre HOG con max_iter=10
+ Clasificador: SVM (lineal)
+ Ratio train_test: 0.20
+ Máx número de ejemplos por categoría: 2 00

Para construir este esquema puede basarse en
+ Función
sklearn.model_selection.train_test_split
+ Función
sklearn. svm
+ Función
obtener_features_hog (fichero p3_tarea 2)
+ Función
construir_vocabulario y obtener_bags_of_words
+Función load_image_dataset (fichero p3_utils.

Y aplíquelo sobre el dataset de escenas scene15
disponible en el material de la práctica. A continuación, responda a la siguiente pregunta:

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)

Hemos realizado distintas simulaciones al variar el tamaño del diccionario BOW donde obtendremos diferentes valores de rendimiento en la clasificación. Inicialmente, con un tamaño de 5 (Figura 8), alcanzamos una precisión del 38% en el test y de un 36% en training. A medida que vamos observan los resultados el mejor tamaño de diccionario será 50(figura 9) con un rendimiento 51% para el valor del test y de 51.5% para un valor de entrenamiento. Desde este valor el rendimiento de test y train va cayendo y creemos

que se deberá al sobreajuste, esto sugiere que hemos alcanzado el punto óptimo de rendimiento. Por el contrario, un valor menor indicaría una simplificación o sobreajuste, ya que no lograríamos la máxima precisión posible. Por lo tanto, determinamos que el tamaño óptimo del diccionario es de 50.

```
Tamaño Diccionario =  5
Rendimiento TEST con Hog=  0.38
Rendimiento TRAIN con Hog =  0.3641666666666667
Rendimiento TEST con Tiny=  0.12833333333333333
Rendimiento TRAIN con Tiny =  0.15
------------------------
```

Figura 8: Tamaño Diccionario igual a 5

```
------------------------
Tamaño Diccionario =  50
Rendimiento TEST con Hog=  0.51
Rendimiento TRAIN con Hog =  0.515
Rendimiento TEST con Tiny=  0.23
Rendimiento TRAIN con Tiny =  0.25958333333333333
------------------------
```

Figura 9: Tamaño Diccionario igual a 50

Por ultimo mostramos la matriz de confusión para el que hemos concluido como mejor caso:

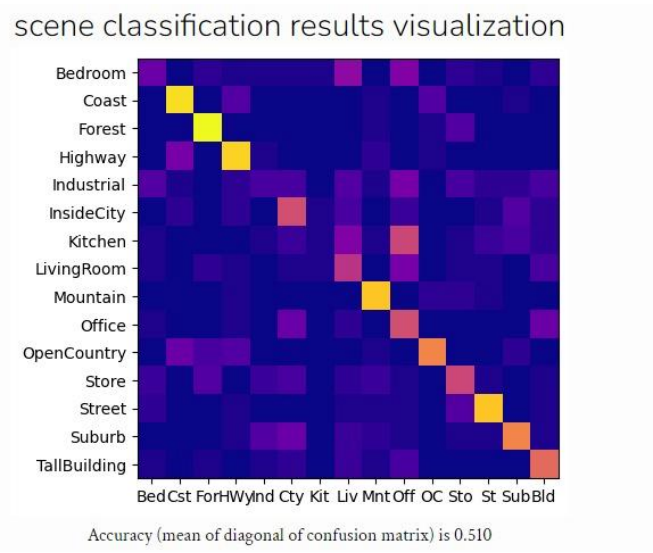

Figura 10: Matriz de confusion para tamaño diccionario igual a 5

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 crea-te_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)

Tras la conclusión obtenida de que 50 era nuestro mejor tamaño, para recapitular sabemos que el rendimiento que obteníamos con un kernel lineal será 51% para el valor del test y de 51.5% para un valor de entrenamiento.
Tras ejecutar los diferentes kernels no lineales obtendremos: un rendimiento de 52,33% de valor de test y 68,5% para valor de training, para "rbf"(figura11), para "poly"(figura 13) obtendremos un valor de test de 52.16% y un valor de training de 74.12%.

Si analizamos los rendimientos vemos como en rbf mejorara, pero será mas significativo en el caso de poly.
La variación en la calidad del rendimiento se debe a la ubicación de la línea de división con respecto a los datos. En función de esta división y clasificación de los datos, estos se asignarán a un grupo u otro, lo que resultará en un mayor o menor número de errores en la clasificación.

```
Tamaño Diccionario =  50
Rendimiento TEST con Hog=  0.5233333333333333
Rendimiento TRAIN con Hog =  0.6858333333333333
Rendimiento TEST con Tiny=  0.23
Rendimiento TRAIN con Tiny =  0.25958333333333333
```

Figura 11: Tamaño Diccionario igual a 50 con rbf

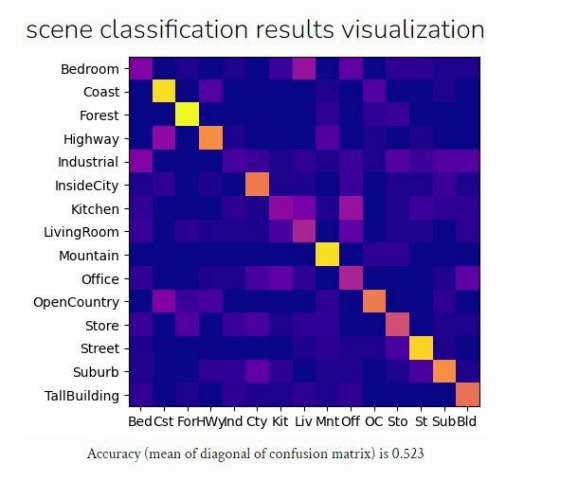

Figura 12: Matriz de confusión para rbf

```
Tamaño Diccionario =  50
Rendimiento TEST con Hog=  0.5216666666666666
Rendimiento TRAIN con Hog =  0.74125
Rendimiento TEST con Tiny=  0.23
Rendimiento TRAIN con Tiny =  0.25958333333333333
```

Figura 13: Tamaño Diccionario igual a 50 con poly

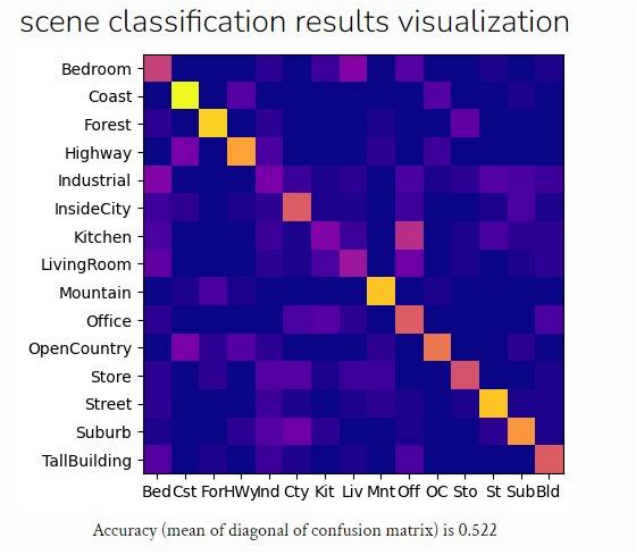

Figura 14: Matriz de confusion para poly

P3.3.

Cree un esquema de clasificación de imágenes completo con los siguientes detalles:

Características: HOG con parámetro tam= 100
+ Modelo BOW sobre HOG con max_iter=10
+ Clasificador: Random Forest
+ Ratio train_test: 0.20
+ Máx número de ejemplos por categoría: 2 00

Para construir este esquema puede basarse en:
+ Función
sklearn.model_selection.train_test_split
+ Función
sklearn. ensemble.RandomForestClassifier
(con valores por
+ Función
obtener_features_hog (fichero p3_tarea2.py)
+ Función
construir_vocabulario y obtener_bags_of_words
(fichero p3_tarea 1 .
+ Función
load_image_dataset (fichero p3_utils.pyc)

Y aplíquelo sobre el dataset de escenas scene15
disponible en el material de la práctica. A continuación, responda a la siguiente pregunta:

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)

Como ya sabemos que 50 es el tamaño que hemos seleccionado, ejecutando random forest obtendríamos un 54.3% para el valor de test y 99.99% (prácticamente 100%) en el valor de training.

```
Tamaño Diccionario =  50
Rendimiento TEST con Hog=  0.5433333333333333
Rendimiento TRAIN con Hog =  0.9995833333333334
Rendimiento TEST con Tiny=  0.23
Rendimiento TRAIN con Tiny =  0.25916666666666666
```

Figura 15: valores de randon forest

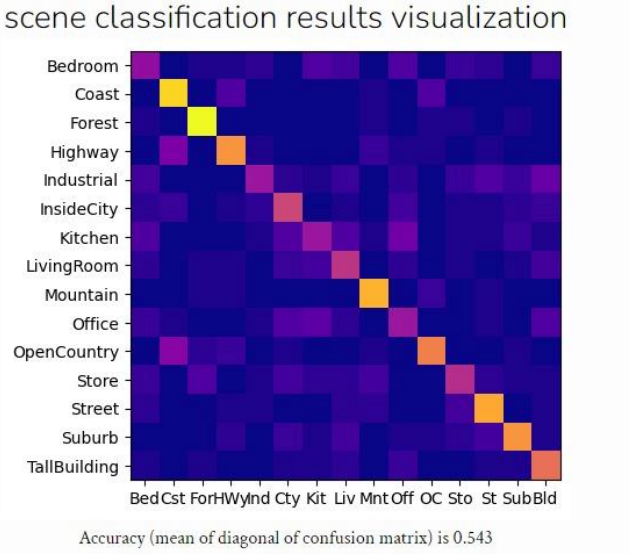

Figura 16: Matriz de confusion de random forest

Cabe destacar que hemos utilizado el parámetro random state igual a 42. Como vimos en teoria los árboles de decisión son propensos al overfitting, cuando estos son los suficientemente profundos. Esto puede llegar a dar valores altos de rendimiento en entrenamiento y bajo en test.

Los colores azules son valores bajos y los amarillos altos. Los más altos se encuentran en la diagonal principal que es el lugar donde tendremos los verdaderos positivos

4 CARGA DE TRABAJO

Tarea	Jose-Luis Fernandez Moreno	Pedro Ramasco Gorria
Tarea 1	2.5	2.5
Tarea 2	3	3
Tarea 3	3	3
Tarea 4	3	3
Memoria	5	5

REFERENCIAS

Incluya en esta seccion todas las referencias utilizadas para desarrollar el prototipo. Estas referencias deben seguir un formato estructurado como los ejemplos.

[1] J.S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," *Neurocomputing – Algorithms, Architectures and Applications,* F. Fogelman-Soulie and J. Herault, eds., NATO ASI Series F68, Berlin: Springer-Verlag, pp. 227-236, 1989. (Book style with paper title and editor)

[2] W.-K. Chen, *Linear Networks and Systems*. Belmont, Calif.: Wadsworth, pp. 123-135, 1993. (Book style)

[3] H. Poor, "A Hypertext History of Multiuser Dimensions," *MUD History,* http://www.ccs.neu.edu/home/pb/mud-history.html. 1986. (URL link *include year)

[4] K. Elissa, "An Overview of Decision Theory," unpublished. (Unpublished manuscript)

[5] 2001_TCSVT_Color_and_texture_descriptors

[6] A Comparative Study of Image Descriptors in Recognizin Human Faces Supported by Distributed Platforms

[7] harris-88-023

[8] Lowe2004_ijcv04

[9] Szeliski_Book_sec4.1

[10] T3.1_ML_ClasificacionImagenes.pdf

[11] T3.2_ML_DeteccionObjetos.pdf

[12] T3.3_ML_SegmentacionRegiones.pdf

[13] Material de Moodle