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

García Santa, Carlos – González Gallego, Miguel Ángel

1 Preguntas Tareas opcionales

En esta ocasión, para realizar los siguientes ejercicios correspondientes a la Práctica 3 de Introducción a la Visión Artificial, se ha tenido en cuenta las bases de reconocimiento de escenas mediante el uso de descriptores y clasificadores, utilizando para ello características **HOG**, **KNN** o **SVM**. Además se ha desarrollado la implementación del modelo **BOW** (Bag-Of-Words), reconocido modelo de texto que sirve para el procesamiento del lenguaje natural y recuperación de información. Todos los métodos realizados en las tareas 1 y 2, han sido utilizados para la recolección de resultados de las siguientes cuestiones.

1.1 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**

Como se indica en el enunciado del ejercicio, se ha realizado un script denominado p3_pregunta_3.1.py donde se han realizado diversas pruebas para la resolución de las preguntas propuestas. Se ha hecho uso de las funciones implementadas con anterioridad; tales como *construir_vocabulario()*, *obtener_bags_of_words()* y funciones proporcionadas por la librería sklearn [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.

Para ello, hemos realizado en primer lugar la obtención de las características **HOG** con un tamaño de 100, y de las características Tiny con tamaño de 64. Tras esto, se ha construido un diccionario de tamaño 50 y a partir de las predicciones realizadas con **KNN**, hemos obtenido los valores porcentuales de rendimiento tanto de **HOG** como de **Tiny** de entrenamiento y de test. En nuestro caso la salida obtenida es la siguiente:

```
HOG -> Train Rendimiento: 0.618, Test Rendimiento: 0.455
Tiny -> Train Rendimiento: 0.280, Test Rendimiento: 0.202
```

Fig. 1. Valores obtenidos de rendimientos HOG y Tiny

Podemos observar, que los valores obtenidos son razonablemente los esperados, según el ranking facilitado en el apartado de Autoevaluación. En primer lugar, en el caso de **HOG**, el valor obtenido del entrenamiento es de un **61.8%** y del test es de un **45.5%**. Por otro lado, en el caso de **Tiny**, el valor del rendimiento de datos de entrenamiento es de un **28.0%** y en el caso de datos de test es de un **20.2%**. En ambos casos, en los valores de datos de test utilizando como entrenamiento 200 imágenes por categoría, se asemejan a los valores esperados, por lo que podemos observar que el resultado es correcto. La precisión entre las características **HOG** y **Tiny** se relacionan a sus condiciones; **HOG** es un descriptor más robusto para capturar características locales en las imágenes, en cambio, las características **Tiny** no capturan información relevante para la clasificación de escenas, por lo que justifica su bajo porcentaje de precisión.

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.

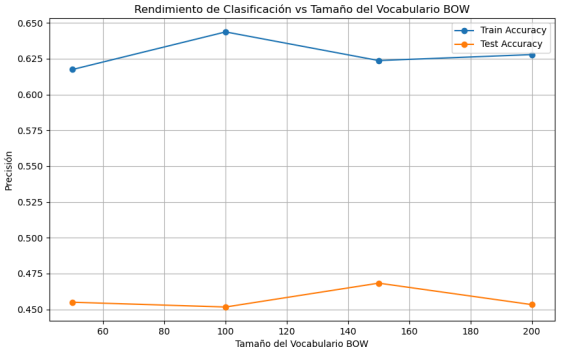

Fig. 2. Rendimientos de clasificación en función del tamaño del vocabulario

Como se puede apreciar, tras la realización de las pruebas pertinentes, se demuestra que dentro del rango de tamaño del vocabulario **BOW** entre 50 y 200, el valor máximo u óptimo es 150. Este valor coincide con el valor máximo de precisión en el test. El desarrollo de la precisión sobre los datos de entrenamiento y test son bastante estables. Se puede observar un ligero aumento del rendimiento en entrenamiento pero no algo significativo. En cambio, el rendimiento en test tiene fluctuaciones menores. Si nos fijamos en el caso de mayor tamaño (200), se empeora la precisión, que puede ser provocada por la aparición de ruido.

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.

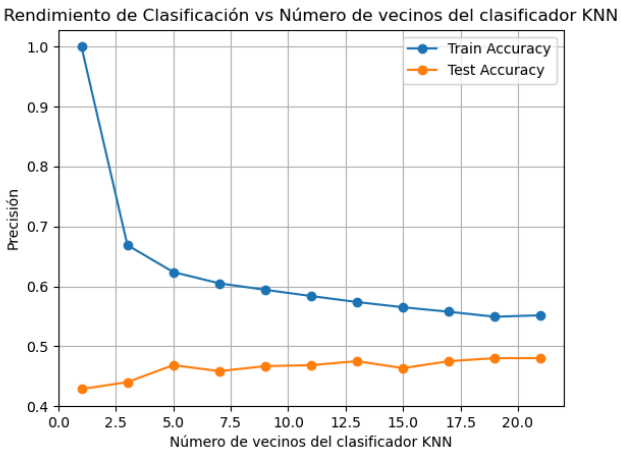

Fig. 3. Precisión en función del número de vecinos del clasificador KNN

En el caso inicial del experimento, el rendimiento de entrenamiento tiene un valor del 100 %, que puede ser debido al sobreajuste, el modelo podría estar realizando una memorización de los datos. Al aumentar el número de vecinos, la precisión de entrenamiento disminuye drásticamente, ya que el modelo es más general.

Si nos fijamos en el rendimiento de test, en el caso inicial, para K=1, la precisión toma el valor más bajo del experimento. Con el aumento del número de vecinos, se toman los valores máximos en K=19. Esto es debido a que en este caso, el modelo generaliza mejor, logrando así una mayor precisión en test.

Como se va a ir observando en el desarrollo de la práctica, estos modelos pueden ser vistos mejorados por los siguientes propuestos como **SVM** o **Random Forest**.

Finalmente, con el uso del método *create_results_webpage()*, se creará una matriz de confusión donde se mostrarán los resultados visuales de aciertos/errores [2]. Dentro del HTML generado se consigue observar una precisión media de **0.480**. Esto quiere decir que el modelo utilizado en este caso ha llegado a clasificar de manera correcta al **48%** de las imágenes en todas las categorías. Después se muestran los valores por categorías, pudiendo resaltar las siguientes categorías con un porcentaje:

- Suburb: **82.5 %**
- Highway: **72.5 %**

En cuanto a las categorías que tienen un peor rendimiento, podemos destacar:

- Industrial: **15.0 %**
- Store: **15.0 %**

Estos últimos pueden darse ya que las características visuales son menos distinguibles o pueden ser confundidas con otras.

Fig. 4. Matriz de confusión para KNN

El resultado final de la ejecución del método create_results_webpage() genera la matriz de confusión de la Fig. 4. Además en la página web, se muestran todos los elementos que se utilizan para la creación de la matriz. Se clasifican en:

- Verdaderos Positivos
- Falsos Positivos
- Falsos Negativos
- Verdaderos Negativos

1.2 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**

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.

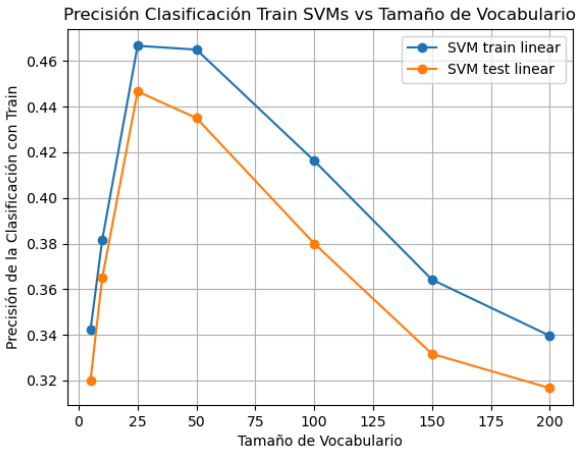

Fig. 5. Precisión de SVM lineal para conjuntos de datos Train y Test

Analizando el rendimiento del clasificador **SVM** (con kernel lineal) en función del tamaño del vocabulario **BOW (Fig. 5)** podemos ver como existe una clara relación entre la precisión de la clasificación y el tamaño de vocabulario, tanto en los datos de **entrenamiento** como en los de **validación**. En la gráfica, en ambos conjuntos podemos observar como inicialmente para un tamaño de **10 palabras** el rendimiento es de un **~33%**, este rendimiento progresa hasta llegar a un máximo para **25 palabras** donde se alcanza un **~45%** de precisión, posteriormente a este máximo el valor cae de forma continuada alcanzando la misma precisión de clasificación con **200 palabras** que con **10 palabras**.

Este comportamiento ocurre debido a la capacidad de **representación** del modelo **BOW** y su interacción con la clasificación de la **SVM** con kernel **lineal**. Para un tamaño de vocabulario reducido, tenemos una cantidad insuficiente de características dentro del **BOW** y por ende no se captura toda la información necesaria para discriminar entre categorías de imágen, resultando en un rendimiento peor del clasificador. Por otro lado, a medida que se incrementa la cantidad de palabras la precisión en la clasificación aumenta significando que la representación de las imágenes con el vocabulario va mejorando, llegando al punto óptimo de 25 palabras. Sin embargo, cuando el tamaño del vocabulario va incrementando más allá de este punto óptimo, el modelo emplea un vector de parámetros con una dimensionalidad muy grande, lo que genera problemas de ruido en el conjunto de características del **BOW**. En espacios de alta dimensión, aunque el kernel lineal sigue buscando una frontera de decisión en un espacio de características lineal, el número elevado de características dificulta encontrar una separación efectiva entre las clases. Como consecuencia, las fronteras de decisión se vuelven menos precisas, y el rendimiento empeora tanto en el entrenamiento como en el conjunto de pruebas.

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 qué se obtienen 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

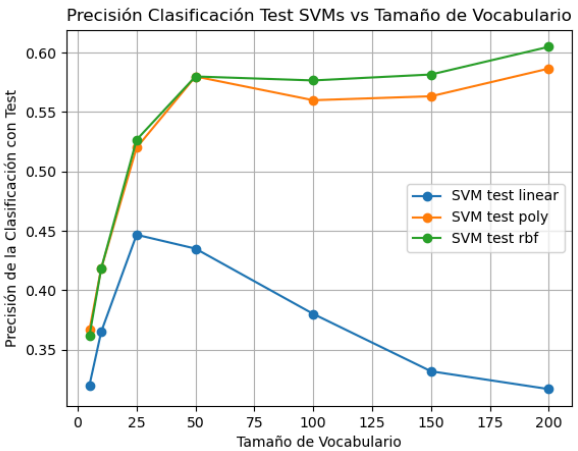

Fig. 6. Precisión de SVMs para conjuntos de datos Test

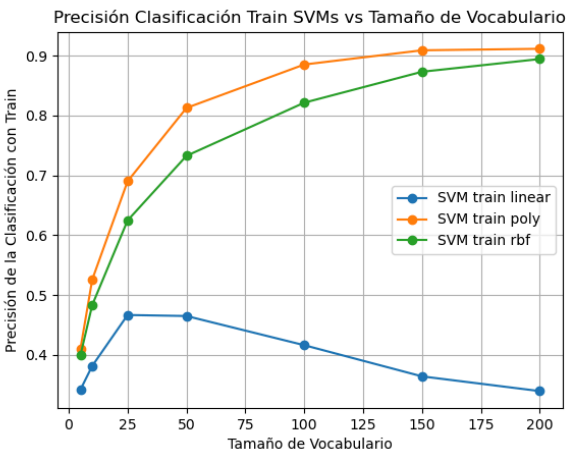

Fig. 7. Precisión de SVMs para conjuntos de datos Train

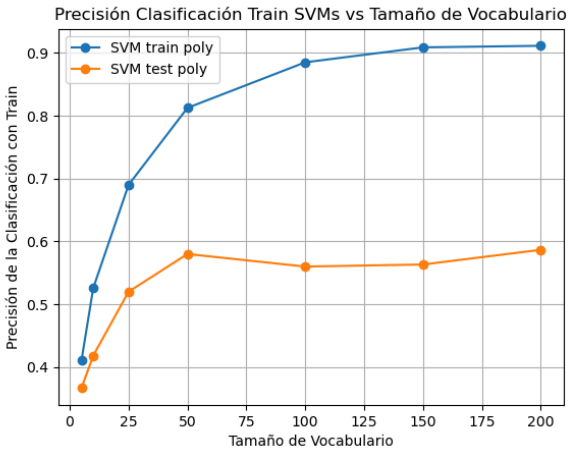

Fig. 8. Precisión de SVM polinomial para conjuntos de datos Train y Test

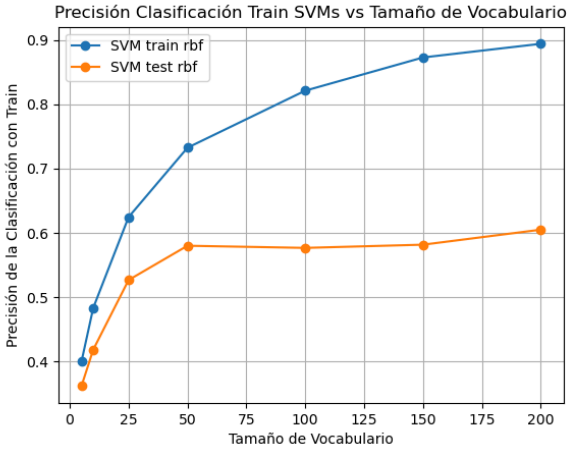

Fig. 9. Precisión de SVM radial para conjuntos de datos Train y Test

Evaluando las gráficas **(Fig. 6-9)**, podemos ver como el rendimiento de los clasificadores **SVM** con distintos kernels no lineales **(polinómico y radial)** también varía en función del tamaño del vocabulario **BOW** utilizado para representar las imágenes.

En el conjunto de prueba, los clasificadores no lineales muestran un rendimiento similar entre ellos, pero con una diferencia muy notable con el lineal **(Fig. 6)**. En concreto, vemos como la **SVM** con **kernel lineal** muestra el mismo rendimiento que se ha obtenido en la anterior pregunta, el cual es significativamente peor que el de los otros kernels.

Atendiendo a los kernels no lineales, podemos ver un rendimiento muy similar entre ellos. En la **SVM** con **kernel polinómico** observamos un inicio con una precisión de **~0.38** para **10 palabras**, que progresa hasta alcanzar **~0.58** en **200 palabras**. En particular, la precisión rápidamente alcanza un pico local en **50 palabras** con una precisión de **~0.57** , después se estabiliza con una ligera disminución y vuelve a crecer al llegar a las **200 palabras** alcanzando la precisión de **~0.58**. Este comportamiento general nos indica que el **kernel polinómico** es más capaz de adaptarse a la complejidad de los datos que el **kernel lineal**, aprendiendo patrones no lineales de manera más eficiente. A pesar de esto, cabe destacar la leve disminución posterior al pico en 50 palabras seguida de un nuevo aumento donde se vuelve a alcanzar el máximo anterior, esto puede indicar que aunque el modelo mejora con el aumento del vocabulario, puede estar alcanzando un punto donde la información adicional aporta menos valor lo cual afecta a la precisión.

El **kernel rbf** presenta una progresión casi idéntica al **polinómico**, con una precisión de **~0.38** para **10 palabras,** un aumento rápido hasta **~0.57** en **50 palabras**, y luego una subida más suave hasta **~0.62** en **200 palabras**. Aunque el comportamiento es muy similar al del kernel polinómico, se observa una ligera diferencia en el intervalo de **50 a 200 palabras**, donde la precisión del **rbf** es ligeramente mayor que la del polinómico y no decrementa en ningún punto, sino que se mantiene lineal y vuelve a incrementar al llegar a **200 palabras**. A pesar de esta diferencia, podemos concluir que ambos kernels no lineales logran un rendimiento significativamente superior al del kernel lineal en el conjunto de datos de prueba para tamaños de vocabulario grandes, lo que representa una mayor efectividad de estos kernels al abordar la complejidad de la clasificación con dimensionalidades grandes.

En el conjunto de entrenamiento, todos los kernels, excepto el lineal, muestran un patrón de crecimiento constante, lo que nos indica que los modelos siguen aprendiendo de manera más efectiva a medida que el vocabulario aumenta **(Fig. 7)**. En concreto, el **kernel polinómico** alcanza una precisión de **~0.92** en **150 palabras** y **~0.93** en **200 palabras**, evidenciando una alta capacidad de aprendizaje y ajuste a los datos de entrenamiento. Sin embargo, esta alta precisión puede estar asociada a un sobreajuste **(Fig. 8)**, donde el modelo se adapta demasiado bien a los datos de entrenamiento, capturando tanto patrones subyacentes como ruido. El **kernel rbf**, aunque también muestra una mejora constante, presenta una precisión ligeramente inferior en cada punto de comparación, por ejemplo, con **10 palabras**, **rbf** logra **~0.4** de precisión frente a **~0.5** del **polinómico**, no obstante, su rendimiento final con **200 palabras** alcanza **~0.9**, lo que sigue siendo alto y sugiere una buena capacidad de aprendizaje, pero un sobreajuste como ocurre con el **kernel polinómico** a lo largo de distintos tamaños de vocabulario **(Fig. 9)**.

En contraste, el kernel lineal no muestra mejoras en el conjunto de entrenamiento como se ha discutido en la pregunta anterior.

Analizando la mejor configuración posible según el tamaño de vocabulario de **25 palabras** (que corresponde con el tamaño óptimo para el kernel lineal), el kernel **rbf** es el que genera la clasificación con mayor precisión para el conjunto de datos de test. Empleando esta configuración obtenemos la matriz de confusión de la **Fig. 9**.

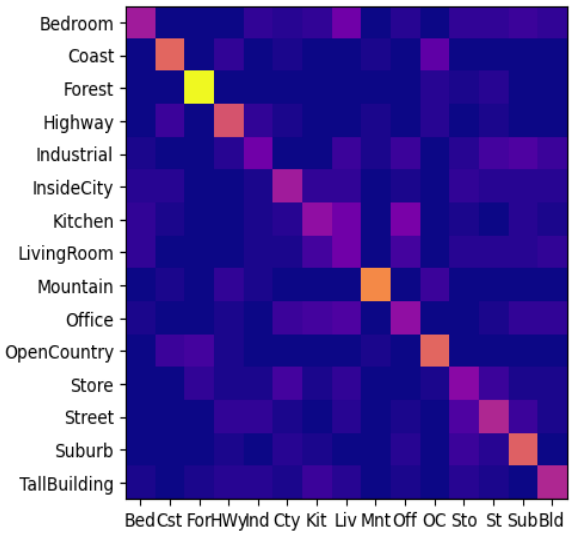

Fig. 9. Matriz de confusión para SVM rbf con 25 de tamaño de vocabulario BOW

La matriz de confusión muestra una precisión media de **0.527**, con un rendimiento bajo en clases como **"Industrial", "LivingRoom", "Kitchen", "Office" y "Store"**. Esto sugiere que el clasificador tiene dificultades para distinguir entre estas categorías debido a su similitud visual. Por otro lado, **"Forest"** muestra una precisión mucho más alta, lo que indica que tiene características más distintivas. Un vocabulario más grande podría mejorar este desempeño, ya que el kernel **rbf** podría aprovechar un espacio de características más amplio para capturar variaciones más sutiles y detalladas, mejorando la separación entre las clases problemáticas. De esta manera, con un vocabulario más extenso, el rendimiento del modelo podría aumentar al reducir el ruido y permitir una representación más precisa de las imágenes.

1.3 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**

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

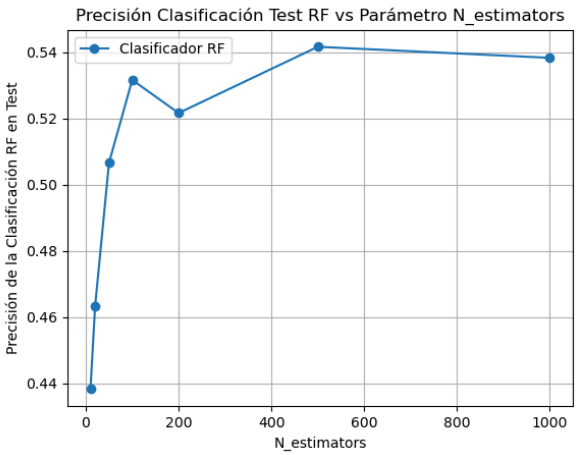

Fig. 10. Precisión de RF para conjuntos de datos Test

Se ha elegido el parámetro **n_estimators**, este parámetro controla el número de árboles en el **RF**, es clave para determinar la capacidad del modelo de capturar patrones complejos y garantizar la estabilidad de sus predicciones.

Realizamos evaluaciones utilizando diferentes valores de **n_estimators**, basándonos en un tamaño de vocabulario **BOW** óptimo de **25 palabras** obtenido previamente en la pregunta **3.2.1**. Los resultados de la **Fig. 10**, muestran que con **10 árboles** la precisión que se obtiene es de aproximadamente **~0.42**, indicando un **subajuste** debido a la limitada capacidad del modelo para capturar patrones. Al incrementar el número de **árboles a 100**, observamos una mejora significativa de la precisión hasta **~0.53**, reflejando una mayor capacidad de generalización. Con **500 árboles**, la precisión se estabiliza y alcanza su valor máximo de **~0.56**, lo que sugiere un equilibrio adecuado entre complejidad y robustez del modelo. Sin embargo, al aumentar el número de **árboles a 1000**, la precisión disminuye ligeramente **~0.54**, probablemente debido a redundancia o posibles efectos de sobreajuste.

Estos resultados nos llevan a concluir que un valor de **n_estimators** de **500** proporciona un balance óptimo entre generalización y capacidad de modelado. Por tanto, seleccionamos esta configuración como la más adecuada.

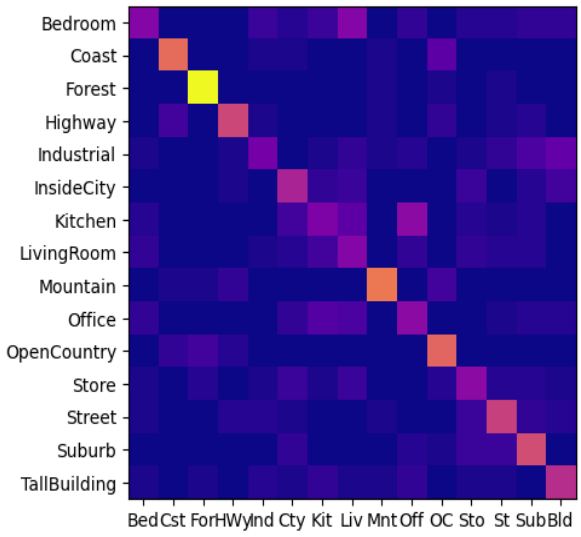

Fig. 11. Matriz de confusión para RF con 500 árboles y 25 de tamaño de vocabulario BOW

Con la configuración óptima de **500 árboles**, el modelo RF muestra una matriz de confusión **(Fig. 11)** con una precisión media de **0.542**. Aunque el rendimiento sigue siendo moderado, se observa una mejora en las clases previamente más difíciles de clasificar, como **"Industrial", "LivingRoom", "Kitchen", "Office" y "Store"**. Esto sugiere que el **RF** ha logrado una mejor capacidad para diferenciar entre estas clases en comparación con el **kernel rbf**, aprovechando la capacidad de los árboles de decisión para capturar relaciones no lineales.

Además, esta mejora puede ser vista como un indicativo de que **RF** es más capaz de manejar la variabilidad en las características de las imágenes. A diferencia de los métodos lineales, los árboles de decisión en un **RF** pueden segmentar el espacio de características en múltiples particiones, lo que les permite encontrar fronteras de decisión más flexibles. Esto explica por qué las clases que antes presentaban una confusión más alta ahora tienen un rendimiento ligeramente mejor. Por otra parte, el aumento en la precisión para ciertas categorías también refleja la capacidad del modelo para mejorar su generalización.

4 Carga de trabajo

Tarea	Horas dedicadas (Carlos García Santa)	Horas dedicadas (Miguel Ángel Gonzalez Gallego)
Explicación	0.5	0.25
Tarea 1	1	2
Tarea 2	2	1.5
Memoria	6	6

Tabla 1. Carga de trabajo en horas

Referencias

[1] Machine Learning en Python. *Scikit-Learn*. https://scikit-learn.org/stable/

[2] Juan Carlos San Miguel Avedillo, "Tema 3: Procesamiento de imágenes basado en Aprendizaje automático (Machine Learning"