Práctica Deep Learning – Predicción de Engagement en POIs Turísticos

Leticia Cabañas Morales
KeepCoding – Junio 2025

---

1. Objetivo

El propósito de este proyecto es desarrollar un modelo avanzado de **Deep Learning** capaz de predecir con precisión el nivel de *engagement* que generarán distintos **puntos de interés turísticos (POIs)**. El modelo integra dos tipos de información complementaria:

- **Características visuales** extraídas mediante redes neuronales convolucionales (CNN), que analizan elementos visuales distintivos en las imágenes de los POIs.
- **Metadatos estructurados**, como coordenadas geográficas, categorías, etiquetas, popularidad y otros atributos contextuales.

El enfoque multimodal empleado permite capturar tanto la reacción estética del usuario como factores prácticos, lo que **mejora la capacidad predictiva** en comparación con modelos unidimensionales.

Entre los objetivos específicos destacan:

- Optimizar la selección de contenido para maximizar la interacción del usuario.
- Identificar patrones visuales asociados a mayor *engagement*.
- Mejorar la experiencia del usuario destacando contenido relevante.
- Generar recomendaciones y *insights* basados en datos para la toma de decisiones.

---

2. Descripción del Dataset

El conjunto de datos ha sido recopilado desde la plataforma **Artgonuts** e incluye información detallada sobre **1.569 POIs**, estructurada en:

Imágenes

- Cada POI dispone de una imagen principal representativa, cuya ruta se encuentra en la columna `main_image_path`.

Metadatos

- `id`: Identificador único del POI.
- `name`: Nombre del punto de interés.
- `shortDescription`: Breve descripción textual.
- `locationLat`, `locationLon`: Coordenadas geográficas.
- `barrio`, `distrito`: Información geográfica adicional.
- `categories`: Temática o tipo de atracción.
- `tier`: Nivel de popularidad o relevancia.

- `tags`: Etiquetas descriptivas.
- `xps`: Métrica de gamificación (puntos de experiencia).

Métricas de Engagement

- `Visits`: Número de visitas registradas.
- `Likes`, `Dislikes`: Valoraciones positivas y negativas.
- `Bookmarks`: Número de veces que el POI ha sido guardado como favorito.

El diseño del dataset permite flexibilidad en la selección de variables, fomentando la justificación analítica y el enfoque en aquellas más relevantes para el modelo predictivo. El dataset se encuentra organizado jerárquicamente, con imágenes almacenadas en carpetas por ID y archivos de metadatos estructurados asociados.

3. Metodología

Reproducibilidad y configuración inicial

Para garantizar la reproducibilidad de los experimentos, se establecieron semillas aleatorias para los distintos entornos (`random`, `numpy` y `torch`), además de fijar los parámetros de CUDNN para obtener resultados deterministas. También se comprobó automáticamente la disponibilidad de GPU para aprovechar la aceleración por hardware durante el entrenamiento del modelo.

Se han instalado las dependencias con `requirements.txt`. Se han importado las librerías clave, y se han cargado desde `module_utils.py` las funciones auxiliares utilizadas en la práctica.

3.1 Preparación y Análisis de Datos

Carga y exploración inicial del dataset

Se cargó el dataset `poi_dataset.csv`, compuesto por 1569 registros y 14 columnas. Se verificó que no existen valores nulos. Posteriormente, se eliminaron las columnas `id`, `name` y `shortDescription` por no aportar valor directo o requerir tratamiento NLP avanzado.

Se llevó a cabo un análisis estadístico y visual de las variables numéricas, identificando distribuciones sesgadas y escalas heterogéneas. Por este motivo, se aplicaron transformaciones logarítmicas a las métricas de engagement (`Visits`, `Likes`, `Dislikes` y `Bookmarks`) antes de definir la variable objetivo, y se normalizaron las variables numéricas.

También se observó que la variable `Visits` presenta muy poca dispersión en los datos. Por el contrario, `Likes`, `Dislikes` y `Bookmarks` sí presentan más variabilidad, lo cual nos hace pensar, a priori, que podrían tener más peso en nuestra variable objetivo.

Análisis y preprocesamiento de variables categóricas

Las columnas `categories` y `tags` contenían listas representadas como cadenas. Se transformaron a listas reales para su posterior procesamiento. El análisis mostró:

- **categories**: La mayoría de los POIs tiene exactamente 3 categorías. Se aprecia un desbalanceo evidente.

- **tags**: Hay un número muy elevado de etiquetas diferentes. Presentan una distribución bastante desbalanceada. La gran mayoría de los POIs tienen varios tags, especialmente 10 y 13. Se observaron algunas listas vacías en ambas variables, pero no fueron tratadas de ninguna manera.

Dado que muchos POIs presentan múltiples categorías, se optó por una representación mediante **multi-hot encoding**, generando vectores binarios por muestra donde cada posición indica la presencia o ausencia de una categoría o etiqueta específica. Esta codificación resulta especialmente útil en tareas de clasificación donde la semántica contextual juega un papel importante.

En el caso de los `tags`, debido a su alta cardinalidad y distribución desbalanceada, se seleccionaron únicamente los 10 más frecuentes para su representación individual. El resto de etiquetas se agruparon en una columna adicional `tag_other`. Esta estrategia permite capturar las etiquetas más representativas sin incrementar excesivamente la dimensionalidad del modelo, mejorando así la eficiencia y capacidad de generalización.

Distribución y correlaciones de variables numéricas

La variable `tier` se decidió utilizar como una variable ordinal, ya que sus valores reflejan distintos niveles de popularidad. Se verificó que el valor 1 se asociaba con un mayor número de `Likes`, indicando una mayor relevancia, mientras que el valor 4 correspondía a los POIs menos populares.

La variable `xps`, aunque es discreta, tiene un valor cuantitativo que el modelo puede explotar directamente, por lo que se decidió tratarla como numérica.

Se evaluaron la distribución y la relación entre las métricas de engagement: `Visits`, `Likes`, `Dislikes` y `Bookmarks`. Se observó lo siguiente:

- `Likes` y `Bookmarks` están fuertemente correlacionadas, lo cual era de esperar. No obstante, dado que los `Bookmarks` son mucho menos frecuentes, se interpretan como acciones que implican una mayor intención.
- `Dislikes` muestra una correlación negativa moderada con `Likes` y `Bookmarks`, por lo que se decidió penalizar su presencia en la métrica de engagement.
- `Visits` prácticamente no presenta correlación con ninguna de las otras métricas, lo que sugiere que el número de visitas no está directamente relacionado con que el POI guste o se guarde.

Se aplicó una transformación logarítmica a estas variables para reducir su asimetría. A continuación, se definió una métrica compuesta llamada `engagement_score`, en la que se asignó mayor peso a los `Bookmarks` (por ser una acción más comprometida) y se penalizó con fuerza la presencia de `Dislikes`, además de considerar `Likes` y `Visits`.

Dado que la distribución de `engagement_score` era multimodal, se planteó el problema como una clasificación multiclase (`engagement_level`), dividiendo el score en tres niveles:

- Bajo (`0`)
- Medio (`1`)
- Alto (`2`)

La división se realizó utilizando los percentiles Q1 y Q3, lo que facilita la interpretación y mejora la capacidad del modelo para aprender. Como se detectó cierto desbalance entre las clases, se optó por realizar una división estratificada de los datos.

Preprocesamiento de imágenes y metadatos

Las imágenes fueron redimensionadas a 224×224 píxeles para ser compatibles con arquitecturas preentrenadas como **ResNet18**, que fue la red neuronal convolucional finalmente utilizada. Además, se normalizaron utilizando las medias y desviaciones estándar de **ImageNet**.

Durante el entrenamiento, se aplicó **data augmentation** sobre las imágenes, incluyendo rotación aleatoria y alteración de color, con el objetivo de mejorar la robustez y capacidad de generalización del modelo.

Las variables numéricas `xps`, `locationLat` y `locationLon` se estandarizaron utilizando `StandardScaler` para facilitar el aprendizaje del modelo y prevenir que variables con escalas mayores dominen durante el entrenamiento.

División del dataset y creación de los loaders

Se realizó una **división estratificada** del dataset en tres conjuntos:

- Entrenamiento: 70%
- Validación: 15%
- Test: 15%

Esta división se llevó a cabo con el objetivo de preservar la distribución de clases de la variable objetivo (`engagement_level`).

Se definió una clase personalizada llamada `POIDataset` para integrar de forma conjunta tanto las imágenes como los metadatos en un único flujo de entrenamiento. Su correcto funcionamiento fue validado mediante ejemplos individuales.

Posteriormente, se crearon los `DataLoaders` correspondientes para cada conjunto, utilizando un `batch size` de 256. Para mitigar el desbalance observado en la variable `engagement_level`, se incorporaron **pesos por clase (`class_weights`)** en la función de pérdida `CrossEntropyLoss`.

3.2 Arquitectura del Modelo

Diseño modular y enfoque multimodal

El modelo final está compuesto por dos ramas principales:

- **Rama visual**: Se utilizó una CNN preentrenada (**ResNet18**), eliminando su capa final de clasificación. Esta red permite extraer representaciones visuales profundas de las imágenes sin necesidad de entrenar desde cero.
- **Rama de metadatos**: Para procesar los metadatos estructurados, se definió un bloque **MLP (Multilayer Perceptron)** compuesto por dos capas *fully connected*, activación **ReLU**, normalización **BatchNorm1d** para estabilizar y acelerar el entrenamiento, y **Dropout** para evitar el sobreajuste.

Las salidas de ambas ramas se **concatenan** y se pasan por un **bloque adicional de clasificación** (`fusion_branch`), compuesto por otra capa oculta con normalización y dropout, finalizando en una **capa de salida de 3 unidades**, correspondiente a las clases del engagement.

Prevención de *data leakage*

Se tuvo especial cuidado en evitar fuga de información entre los conjuntos de entrenamiento, validación y prueba. Las transformaciones aplicadas a los metadatos (como el escalado) se ajustaron **exclusivamente sobre el conjunto de entrenamiento**, y posteriormente se aplicaron a los conjuntos de validación y prueba.

El modelo fue probado utilizando un *batch* del conjunto de entrenamiento para verificar la correcta integración de las entradas y el funcionamiento de la red. La salida fue una **predicción en forma de tensor de 3 clases**, con un *batch size* de **256**, como se esperaba.

3.3 Entrenamiento y Optimización

Se implementó una función `POI_train_model` que encapsula el *pipeline* completo de entrenamiento, evaluación y aplicación de *early stopping*.

Durante el proceso, se entrenaron múltiples configuraciones del modelo, variando **hiperparámetros clave** como:

- *Learning rate*
- *Dropout rate*
- Número de épocas
- Función de activación
- *Weight decay*

También se aplicó la técnica de *early stopping* con una paciencia de **3 a 5 épocas**, deteniendo el entrenamiento cuando no se observaban mejoras en la métrica de validación.

Para manejar el **desbalance de clases**, se utilizó la función de pérdida `CrossEntropyLoss` ponderada con `class_weights`. El optimizador empleado en todos los entrenamientos fue **Adam**, dada su eficiencia y capacidad de adaptación.

A continuación se detallan en una tabla los resultados y métricas obtenidas en cada uno de los entrenamientos realizados, de manera que podemos observar el impacto de cada cambio. Finalmente, se seleccionó el modelo que mostró la **mayor *accuracy* en validación** y curvas de aprendizaje más estables.

Resumen de experimentos y configuración

ID	Dropout	Activación	Épocas	LR	WD	Mejor Val Acc	Comentario breve
1	0.3	ReLU	10	1e-3	1e-3	79.00%	Inicio base, picos en validación
2	0.3	ReLU	10	5e-4	1e-3	83.00%	Mejora por menor learning rate
3	0.3	ReLU	15	5e-4	1e-3	85.50%	Más épocas mejora estabilidad
4	0.4	ReLU	20	5e-4	1e-3	86.00%	Mayor regularización (dropout)

ID	Dropout	Activación	Épocas	LR	WD	Mejor Val Acc	Comentario breve
5	0.4	ELU	25	5e-4	1e-3	87.50%	Cambio de activación mejora rendimiento
6	0.5	ELU	25	5e-4	1e-3	88.00%	Mejor resultado, balance entre estabilidad y precisión

Análisis de los entrenamientos realizados

A continuación, se presenta un análisis secuencial de los experimentos llevados a cabo, explicando la motivación detrás de cada ajuste de hiperparámetros y observaciones clave a partir de los resultados obtenidos. En todos ellos se usó un *batch size* de 256, una semilla de 42 y *hidden_dim* de 128.

1. **Entrenamiento 1**
    - **Configuración**: Dropout 0.3, Activación ReLU, 10 épocas, LR = 1e-3, WD = 1e-3
    - **Accuracy validación**: 79.00%
    - **Observaciones**: Este experimento sirvió como punto de partida. Se observaron picos irregulares en la curva de validación, lo que sugiere posibles problemas de sobreajuste o una tasa de aprendizaje demasiado alta.
2. **Entrenamiento 2**
    - **Cambio aplicado**: Se redujo el *learning rate* a 5e-4
    - **Accuracy validación**: 83.00%
    - **Justificación**: Una tasa de aprendizaje menor permitió un entrenamiento más estable. La mejora del 4% en precisión valida que el modelo se beneficiaba de una menor agresividad en la actualización de pesos.
3. **Entrenamiento 3**
    - **Cambio aplicado**: Se aumentaron las épocas a 15
    - **Accuracy validación**: 85.50%
    - **Justificación**: El incremento de épocas permitió al modelo aprovechar mejor la tasa de aprendizaje ajustada. La estabilidad en las curvas de aprendizaje y la mejora del rendimiento sugieren que el modelo no estaba aún saturado.
4. **Entrenamiento 4**
    - **Cambio aplicado**: Aumento del dropout a 0.4
    - **Accuracy validación**: 86.00%
    - **Justificación**: Se introdujo mayor regularización para combatir un posible sobreajuste, especialmente tras más épocas. El resultado muestra una mejora marginal, lo que indica que la regularización fue beneficiosa sin perjudicar la capacidad de aprendizaje.
5. **Entrenamiento 5**
    - **Cambio aplicado**: Cambio de función de activación a ELU
    - **Accuracy validación**: 87.50%

- **Justificación**: ELU permite una mejor propagación de gradientes frente a ReLU, especialmente con datos normalizados. La mejora confirma que la red se benefició de funciones de activación más suaves.

6. **Entrenamiento 6**

   - **Cambio aplicado**: Dropout aumentado a 0.5
   - **Accuracy validación**: 88.00%
   - **Justificación**: El objetivo fue alcanzar un equilibrio óptimo entre regularización y capacidad de aprendizaje del modelo. Esta configuración logró el rendimiento más alto, con curvas de aprendizaje consistentes, lo que la consolidó como la opción final seleccionada.

3.4 Evaluación y Análisis

El modelo seleccionado fue evaluado sobre el **conjunto de test**, obteniendo un **accuracy del 87.29%**.

El análisis de métricas muestra un **buen equilibrio entre precisión y recall en todas las clases**, destacando especialmente la robustez en la **clase 0** (bajo engagement). La **clase 1** presentó una leve menor sensibilidad, lo que podría deberse a su naturaleza más ambigua o a una menor cantidad de ejemplos representativos.

Informe de clasificación

Clase	Precisión	Recall	F1-score	Soporte
0 (Bajo)	0.855	0.946	0.898	112
1 (Medio)	0.879	0.785	0.829	65
2 (Alto)	0.907	0.831	0.867	59
Accuracy			**0.873**	**236**
Macro avg	0.881	0.854	0.865	236
Weighted avg	0.875	0.873	0.872	236

Desde el punto de vista práctico, estos resultados implican que el modelo es capaz de **discriminar entre distintos niveles de engagement con bastante precisión**.

La **clase de mayor compromiso (2)** se predice de forma especialmente precisa, lo cual resulta relevante para la **toma de decisiones estratégicas en el ámbito turístico**, donde identificar contenido de alto impacto es crucial.

Adicionalmente, se **visualizaron predicciones reales del modelo** sobre imágenes del conjunto de test, observándose **7 aciertos y 2 errores entre 9 ejemplos**. Esto confirma que el modelo no solo rinde bien a nivel cuantitativo, sino que también mantiene una **coherencia cualitativa** en sus predicciones visuales.

Para **trabajos futuros**, se propone:

- **Explorar la interpretabilidad del modelo**, por ejemplo utilizando técnicas como **Grad-CAM** sobre imágenes o **SHAP** para metadatos, lo cual permitiría entender mejor las decisiones internas de la red.
- **Realizar un análisis más exhaustivo de errores**, identificando patrones o características comunes en los ejemplos mal clasificados, con el objetivo de afinar el modelo y mejorar su capacidad de generalización.