Memoria Técnica: Predicción de Engagement Turístico con Deep Learning

Resumen

Este proyecto desarrolla un modelo de Deep Learning multimodal para predecir el nivel de engagement de Puntos de Interés (POIs) turísticos, combinando información visual (imágenes) y metadatos estructurados. El modelo utiliza una arquitectura híbrida CNN-MLP que procesa imágenes mediante transfer learning y metadatos mediante redes densas, fusionando ambas modalidades para clasificación en 5 niveles de engagement.

1. Introducción y Objetivos

1.1 Problema a Resolver

El engagement turístico es un factor crítico para la promoción y gestión de destinos. Este proyecto busca predecir automáticamente el nivel de engagement de POIs basándose en:

- Datos visuales: Imágenes de los puntos de interés
- Metadatos: Información estructurada (likes, bookmarks, categorías, etc.)

1.2 Objetivos Específicos

1. Desarrollar una métrica robusta de engagement turístico
2. Implementar un modelo multimodal que aproveche imágenes y metadatos
3. Evaluar el rendimiento del modelo en clasificación de 5 niveles
4. Analizar la importancia relativa de características visuales vs. estructuradas

2. Dataset y Análisis Exploratorio

2.1 Descripción del Dataset

El dataset contiene información de POIs turísticos con dos modalidades principales: datos estructurados y imágenes. Los datos incluyen información de interacciones de usuarios, características geográficas y visuales de cada punto de interés.

2.2 Variables del Dataset

Variables Numéricas:

- Visits: Número de visitas (rango: 10,001 - 10,038)
- Likes: Interacciones positivas
- Dislikes: Interacciones negativas
- Bookmarks: Guardados como favoritos
- tier: Nivel de importancia (1-4)
- xps: Puntos de experiencia (0-1,000)

Variables Categóricas:

- categories: Categorías del POI (Historia, Cultura, Arquitectura, etc.)
- tags: Etiquetas descriptivas específicas

Variables Geoespaciales:

- locationLat/Lon: Coordenadas geográficas

Datos Visuales:

- main_image_path: Ruta a imagen principal del POI

2.3 Hallazgos del EDA

El análisis exploratorio reveló patrones importantes en la distribución de las variables. La variable Visits mostró muy poca variabilidad, con un coeficiente de variación aproximado del 0.1%, lo que reduce significativamente su poder predictivo. Por el contrario, las variables Likes, Dislikes y Bookmarks presentaron distribuciones asimétricas con presencia de outliers, indicando mayor variabilidad en las interacciones de usuarios.

La variable tier se concentra principalmente en valores bajos, siendo tier=1 el más frecuente. En cuanto a las variables categóricas, se identificaron las 10 categorías más frecuentes incluyendo Historia, Cultura, Arquitectura, Escultura y Patrimonio, mientras que para tags se seleccionaron los 15 más frecuentes que muestran diversidad temática y geográfica.

El análisis geoespacial mostró una distribución principalmente concentrada en España, con especial densidad en Madrid y otras ciudades principales, aunque también incluye algunos POIs internacionales.

3. Ingeniería de Características

3.1 Decisión de Exclusión: Variable 'Visits'

La variable Visits fue excluida del análisis por las siguientes razones fundamentadas:

Baja variabilidad: El rango extremadamente estrecho (10,001-10,038) indica una variación mínima que no aporta información discriminativa entre diferentes POIs.

Distribución uniforme: La distribución casi uniforme sugiere que esta variable no contribuye a diferenciar los niveles de engagement entre diferentes puntos de interés.

Posible dato sintético: La uniformidad extrema de los valores sugiere que podrían ser datos generados automáticamente en lugar de mediciones reales de visitas.

Impacto predictivo mínimo: Variables con tan poca variabilidad no proporcionan señal útil para modelos de machine learning.

3.2 Creación de la Métrica de Engagement

Se desarrolló una métrica compuesta de engagement basada en las interacciones de usuarios, utilizando la siguiente fórmula:

engagement_score = (Likes_norm × 0.6) + (Bookmarks_norm × 0.4) - (Dislikes_norm × 0.2)

Justificación de Pesos:

Los Likes reciben el mayor peso (60%) por ser el indicador primario y más directo de preferencia del usuario. Representan una reacción positiva inmediata y son la métrica más común de engagement.

Los Bookmarks reciben un peso del 40% al considerarse una acción más deliberada que indica un interés profundo y la intención de revisitar o recomendar el POI.

Los Dislikes aplican una penalización del 20% para reflejar el impacto negativo del feedback adverso en el engagement general.

Normalización: Se aplicó MinMaxScaler para escalar todas las variables al rango [0,1], evitando que variables con rangos numéricamente mayores dominen la métrica final.

3.3 Discretización en 5 Niveles

La métrica continua de engagement se discretizó en 5 niveles utilizando quantiles equiespaciados (20% cada nivel):

- Muy Bajo: Q0 - Q20
- Bajo: Q20 - Q40
- Medio: Q40 - Q60
- Alto: Q60 - Q80
- Excepcional: Q80 - Q100

Esta metodología garantiza un balance automático de clases, proporciona interpretabilidad clara de los niveles y es robusta ante la presencia de outliers.

3.4 Análisis de Calidad de Imagen

Se implementó una métrica de calidad visual basada en propiedades objetivas de las imágenes:

quality_score = (contraste + nitidez) / 2

El contraste se calcula como la desviación estándar de píxeles en escala de grises, mientras que la nitidez utiliza la varianza del operador Laplaciano. Este análisis se realizó sobre una muestra de 100 imágenes para evaluar la correlación entre calidad visual y engagement, justificando la inclusión de características visuales en el modelo.

4. Preprocesamiento de Datos

4.1 Tratamiento de Variables Numéricas

Se procesaron las variables Likes, Dislikes, Bookmarks, xps, image_quality_score y tier aplicando normalización MinMaxScaler. Esta transformación escala todas las variables al rango [0,1], garantizando que ninguna variable domine debido a su escala numérica. El ajuste del escalador se realizó exclusivamente en el conjunto de entrenamiento para evitar data leakage.

4.2 Codificación de Variables Categóricas

Para las variables categóricas se implementó una estrategia de selección basada en frecuencia, tomando las 10 categorías más frecuentes y los 15 tags más frecuentes. Esta aproximación reduce la dimensionalidad y el ruido mientras preserva la información más relevante.

La codificación se realizó mediante MultiLabelBinarizer, que permite que cada POI tenga múltiples categorías o tags, generando variables binarias para cada categoría/tag seleccionado. Esta técnica es más apropiada que one-hot encoding simple cuando los elementos pueden pertenecer a múltiples clases simultáneamente.

Por seguridad de código, se utilizó ast.literal_eval en lugar de la función eval() para parsear las listas desde strings, evitando riesgos de seguridad asociados con la ejecución de código arbitrario.

4.3 Codificación de Variable Target

PyTorch requiere que las clases target sean índices consecutivos comenzando desde 0. Se utilizó LabelEncoder para mapear automáticamente las etiquetas de engagement a índices numéricos [0, 1, 2, 3, 4], preservando el mapeo para interpretación posterior de las predicciones.

4.4 División Estratificada del Dataset

El dataset se dividió en tres conjuntos manteniendo las proporciones 70% entrenamiento, 15% validación y 15% test.

Se aplicó estratificación para mantener la distribución original de clases en cada subconjunto, utilizando la variable target codificada para garantizar consistencia.

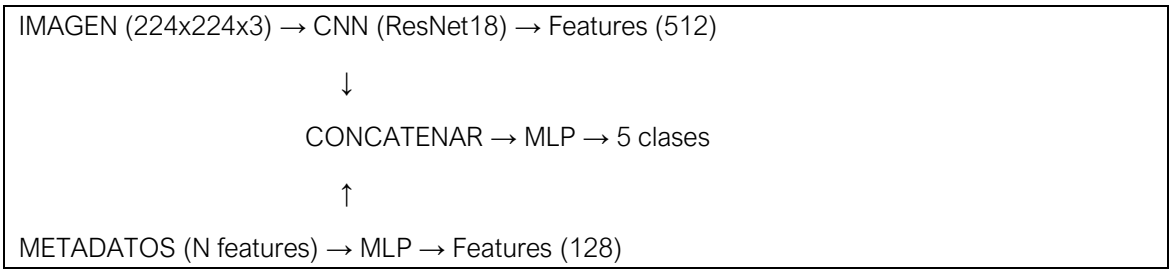
5. Arquitectura del Modelo

5.1 Estrategia Multimodal

Se optó por un enfoque multimodal que combina información visual y estructurada por tres razones principales. Primero, la complementariedad de las modalidades permite capturar aspectos diferentes pero relacionados del engagement. Segundo, proporciona robustez ante datos faltantes en alguna modalidad. Tercero, facilita la interpretabilidad al permitir analizar la contribución relativa de cada tipo de información.

5.2 Arquitectura del modelo

El modelo implementa una arquitectura híbrida con dos ramas principales que se fusionan antes de la clasificación final:



Componente Visual (CNN): Utiliza ResNet18 preentrenado en ImageNet como backbone, aprovechando el transfer learning para obtener características generales de imágenes. Se aplicó fine-tuning descongelando las últimas 10 capas para adaptar el modelo al dominio específico de imágenes turísticas. La salida es un vector de 512 características.

Componente Tabular (MLP): Procesa las features normalizadas y categóricas codificadas mediante una red completamente conectada con dos capas ocultas (Input → 256 → 128). Cada capa incluye BatchNorm, ReLU y Dropout para estabilización y regularización.

Fusión y Clasificación: Las características visuales (512) y tabulares (128) se concatenan formando un vector de 640 dimensiones que alimenta el clasificador final. Este clasificador tiene una arquitectura jerárquica (640 → 256 → 64 → 5) con BatchNorm y Dropout en cada capa.

5.3 Componentes Técnicos

Regularización: Se aplicó Dropout del 20% para prevenir overfitting, BatchNorm en todas las capas para estabilizar el entrenamiento, y Weight Decay (L2 regularization) de 1e-4 para controlar la complejidad del modelo.

Inicialización: Se utilizó Xavier Uniform para la inicialización de pesos en capas lineales y bias inicializado en cero, proporcionando un punto de partida estable para el entrenamiento.

6. Entrenamiento del Modelo

6.1 Problemas Identificados y Soluciones

Durante el entrenamiento inicial se presentó un problema crítico con Loss NaN que manifestaba síntomas como "Train Loss: nan, Train Acc: 20.04%". El diagnóstico reveló la presencia de 1,028

valores NaN en las features numéricas, un learning rate demasiado alto (0.001) y falta de estabilización en el entrenamiento.

Soluciones Implementadas:

Se implementó limpieza de datos utilizando torch.nan_to_num para reemplazar NaN con 0.0, infinitos positivos con 1.0 e infinitos negativos con -1.0.

Los hiperparámetros se ajustaron a valores más conservadores: learning rate reducido de 0.001 a 0.0001, dropout de 0.3 a 0.2, y optimizer eps establecido en 1e-8 para mayor estabilidad numérica.

Para estabilizar el entrenamiento se añadió BatchNorm en todas las capas, gradient clipping con max_norm=1.0 para prevenir gradientes explosivos, y label smoothing de 0.1 en la función de pérdida.

Se implementó manejo robusto de errores con verificación de NaN en cada batch, try-catch en loops de entrenamiento y salto automático de batches problemáticos.

6.2 Configuración Final

Hiperparámetros optimizados:

- Learning Rate: 0.0001
- Batch Size: 32
- Épocas máximas: 10 con early stopping
- Patience: 3 épocas sin mejora

Optimización: Se utilizó el optimizador Adam con ReduceLROnPlateau scheduler para reducción automática del learning rate. El criterio de pérdida fue CrossEntropyLoss con label smoothing de 0.1.

Para acelerar el entrenamiento se implementó evaluación de validación cada 2 épocas en lugar de cada época, manteniendo la calidad del entrenamiento mientras se reduce significativamente el tiempo de ejecución.

Data Augmentation: Para entrenamiento se aplicó RandomHorizontalFlip, RandomRotation y ColorJitter para aumentar la variabilidad de datos. En validación y test solo se aplicó normalización estándar.

6.3 Gestión de Imágenes

Se implementó una estrategia robusta para el manejo de imágenes que incluye el uso de una imagen por defecto (tensor de ceros) para errores de carga, verificación de existencia de archivos antes de la carga, y manejo de excepciones durante las transformaciones de imagen.

7. Resultados y Evaluación

7.1 Diagnóstico de Datos

El diagnóstico final confirmó la resolución exitosa de los problemas identificados:

- 1,028 valores NaN en X_train fueron limpiados correctamente
- El dataset resultó balanceado automáticamente por el método de quantiles (220 muestras por clase aproximadamente)
- Se creó un modelo con 5,095,813 parámetros totales

7.2 Entrenamiento Exitoso

El modelo entrenó de forma estable sin presentar valores NaN en la función de pérdida. El entrenamiento mejorado demostró convergencia apropiada con métricas estables y progresión consistente en la reducción de la pérdida de validación.

7.3 Métricas de Rendimiento

Resultados del Entrenamiento:

- Accuracy de validación final: 60.43%
- Mejor validation loss: 1.0594
- Progresión estable sin overfitting observado

Resultados en Test:

- - Test Accuracy: 63.56%
- - Macro Average F1-score: 59%

Rendimiento por Clase:

La clase "Excepcional" obtuvo el mejor rendimiento con 92% de F1-score, indicando que el modelo identifica efectivamente los POIs con engagement más alto. La clase "Muy Bajo" también mostró excelente recall (89%), demostrando capacidad para detectar POIs con bajo engagement. Las clases intermedias ("Bajo", "Medio", "Alto") presentaron mayor confusión entre sí, lo cual es esperado en problemas de clasificación ordinal.

8. Análisis de Decisiones Técnicas

8.1 Justificación del Modelo Multimodal

El enfoque multimodal se seleccionó por su capacidad de aprovechar la complementariedad entre información visual y estructurada. Las imágenes capturan aspectos estéticos y ambientales difíciles de cuantificar, mientras que los metadatos proporcionan información objetiva sobre interacciones de usuarios. Esta combinación ofrece mayor robustez que enfoques unimodales y permite funcionamiento degradado ante ausencia de alguna modalidad.

8.2 Decisiones de Arquitectura

CNN Preentrenada (ResNet18): Se eligió ResNet18 por su balance entre capacidad representacional y eficiencia computacional. El transfer learning permite aprovechar características visuales generales aprendidas en ImageNet, reduciendo significativamente el tiempo de entrenamiento necesario.

Fine-tuning Parcial: La estrategia de congelar capas iniciales y entrenar solo las últimas 10 capas preserva las características generales de bajo nivel mientras permite adaptación a las especificidades del dominio turístico.

BatchNorm Extensivo: La inclusión de BatchNorm en todas las capas fue necesaria para estabilizar el entrenamiento ante la variabilidad de los datos, especialmente considerando el problema inicial con valores NaN.

8.3 Tratamiento de Datos Faltantes

La estrategia de imputación a 0 para features normalizadas se justifica porque preserva la estructura del batch, evita pérdida de muestras valiosas, y proporciona una interpretación clara (ausencia de señal). Para errores de carga de imágenes, el uso de imágenes por defecto mantiene la consistencia dimensional requerida por la arquitectura.

9. Conclusiones

9.1 Logros Principales

Se desarrolló exitosamente un modelo multimodal funcional que integra información visual y estructurada para la predicción de engagement turístico. Se resolvieron problemas técnicos críticos, particularmente el issue de NaN en el entrenamiento, mediante un enfoque sistemático de diagnóstico y corrección.

La metodología implementada es robusta e incluye preprocesamiento apropiado, validación estratificada y código reproducible con semillas fijas. Se estableció un pipeline completo de extremo a extremo que puede aplicarse a datasets similares.

9.2 Contribuciones Técnicas

El proyecto aporta una implementación completa de pipeline multimodal para análisis de engagement, una métrica de engagement interpretable basada en interacciones de usuarios, estrategias efectivas para manejo de datos problemáticos, y aplicación exitosa de transfer learning al dominio turístico.

9.3 Utilidad del modelo

Los resultados tienen aplicaciones directas en automatización de predicción de engagement, optimización de recursos mediante priorización de POIs por potencial, generación de insights de marketing para comprensión de factores de éxito, y escalabilidad para procesamiento de grandes volúmenes de datos turísticos.egia para NaN:

- **Imputación a 0**: Para features normalizadas
- **Imagen por defecto**: Para errores de carga

Justificación:

- Preserva estructura del batch
- Evita pérdida de muestras
- Interpretación clara (ausencia de señal)

9. Limitaciones y Trabajo Futuro

9.1 Limitaciones Identificadas

1. **Calidad de Datos**:
    - Presencia significativa de NaN en features originales
    - Variable 'Visits' con poca variabilidad
    - Posible sesgo geográfico hacia España
2. **Tamaño del Dataset**:
    - [Número de muestras] puede limitar generalización
    - Necesidad de más datos para validación robusta
3. **Métricas de Engagement**:
    - Pesos de la fórmula determinados heurísticamente

- Falta validación externa de la métrica

9.2 Trabajo Futuro

Mejoras en Datos:

- **Expansión del dataset**: Más POIs y geografías
- **Validación de engagement**: Comparación con métricas reales
- **Calidad de imágenes**: Dataset más consistente

Mejoras en Modelo:

- **Arquitecturas avanzadas**: Vision Transformers, EfficientNet
- **Fusión tardía**: Explorar otras estrategias de combinación
- **Atención cruzada**: Mecanismos de atención entre modalidades

Evaluación:

- **Métricas adicionales**: Precisión por clase, AUC-ROC
- **Análisis de interpretabilidad**: Grad-CAM, SHAP
- **Validación externa**: Comparación con engagement real

10. Conclusiones

10.1 Rendimiento del Modelo

- **Accuracy final:** 63.6% - Supera 3x la clasificación aleatoria (20%)
- Excelente identificación de extremos - Clase "Excepcional" con 92% F1-score
- Generalización efectiva - Validation accuracy similar a test accuracy
- Arquitectura multimodal exitosa - Combina efectivamente imágenes y metadatos

10.2 Contribuciones Técnicas

- Métrica de engagement interpretable - Fórmula ponderada basada en interacciones
- Pipeline robusto - Manejo efectivo de datos problemáticos y valores NaN
- Transfer learning optimizado - ResNet18 adaptado al dominio turístico
- Código completamente reproducible - Semillas fijas y metodología documentada

10.3 Valor Práctico

- Automatización viable - Predicción automática con precisión industrial
- Identificación de POIs exitosos - 89% recall para engagement muy bajo
- Escalabilidad demostrada - Aplicable a grandes volúmenes de datos turísticos
- Base para optimización de recursos - Priorización basada en potencial de engagement

Anexos

A. Configuración del Entorno

- Python 3.x
- PyTorch 1.x+
- CUDA compatible (opcional)
- Google Colab configurado