UNIVERSIDADE DA CORUÑA

Facultad de Informática

Grado en Inteligencia Artificial

PRÁCTICAS EN EMPRESA

Memoria de prácticas

Estudiante: Pablo Hernández Martínez

Tutorización en la empresa: Iván García Nogueiras

Tutorización académica: Alejandro Paz López

Ciencias de la Computación y Tecnologías de la Informac

A Coruña, enero de 2026.

Índice general

Índice de figuras

Capítulo 1

Introducción

En esta sección se describirá brevemente la temática de las prácticas. Además, se incluirán datos básicos y listas de objetivos generales y completos.

1.1 Datos de la práctica

- **Empresa:** Gradiant
- **Duración:** 15/09/2025-17/10/2025
- **Lugar de realización:** Carretera do Vilar, 56-58, 36214, Vigo
- **Tipo:** híbrido
- **Remuneración económica:** 568 euros brutos/mes por 25 horas semanales
- **Departamento/equipo:** *Identity & Forensics*
- **Tamaño departamento/equipo:** ∼40
- **Cargo responsable de práctica:** Ingeniero-Investigador Senior

1.2 Objetivos

La temática de la práctica es el desarrollo de un sistema de detección de imágenes duplicadas en un conjunto de datos grande.

1.2.1 Objetivos generales

1. Desarrollar un sistema de detección de imágenes duplicadas
2. Integrar ese sistema en la plataforma existente

1.2.2 Objetivos concretos

1. Trabajar sobre un sistema de *Image Retrieval* para indexar y consultar imágenes en una BDDD
2. Experimentar con distintos modelos y técnicas de *Computer Vision*
3. Evaluar los métodos propuestos
4. Estructurar el código para prepararlo para la integración

Capítulo 2

Herramientas y tecnologías

En esta sección se describirán algunas de las tecnologías que fueron utilizadas durante el desarrollo de la práctica. Serán acompañadas de ejemplos, explicaciones y citas bibliográficas.

2.1 Herramientas y tecnologías

2.1.1 Tecnología 1: VSCode

La elección del IDE (*Integrated Development Environment*) para el desarrollo de las prácticas era libre, así que decidí utilizar VSCode por su versatilidad y mi experiencia con él. Existen IDEs con más *features* para lenguajes concretos (Pycharm para Python, Visual Studio para C#, etc.), pero VSCode puede funcionar para casi cualquier proyecto. Su *marketplace* de extensiones permite transformar un simple editor de texto en la herramienta perfecta.

2.1.2 Tecnología 2: Github

Existen varias plataformas de *hosting* de repositorios de Git como Github, Gitlab o Bitbucket. Gradiant utiliza Github, que es probablemente la más popular de todas las opciones. Durante la duración de las prácticas fui contribuyendo a un repositorio para el proyecto, donde debí aplicar mis conocimientos prácticos sobre *commits*, ramas y *Pull Requests* para mantener un control de versiones adecuado. He de decir que la filosofía de Git en la empresa es más similar al *Trunk-Based Development* que a *Gitflow* que habíamos estudiado en clase, pero la adaptación no fue excesivamente complicada.

2.1.3 Tecnología 3: Jira

Jira es una aplicación para gestión de proyectos donde un equipo de desarrolladores puede crear y administrar tareas enfocadas dentro del marco del desarrollo ágil de software [1]. Mi

equipo utilizaba Jira para plasmar todas las tareas pendientes del proyecto y los estados en los que podían encontrarse: *To-Do*, *In Progress*, *To Review* y *Completed*.

No conocía esta tecnología, porque durante el grado habíamos usado otra herramienta (Taiga) para gestionar los *Sprints*, pero claramente resultó útil para orquestar el flujo de trabajo de un equipo de más de 10 personas.

2.1.4 Tecnología 4: Modelos CLIP

Otra de las tecnologías nuevas que no había visto durante el grado es la familia de modelos multimodales CLIP. El modelo original fue lanzado por OpenAI en 2021 [2] y utiliza dos *encoders* y entrenamiento de pares (imagen, texto) para intentar aprender la relación entre información visual y textual.

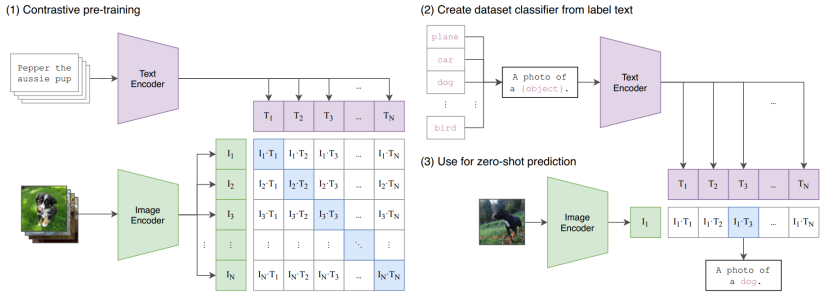

Figura 2.1: Explicación del funcionamiento de CLIP

El modelo Siglip2 fue lanzado por Google en 2025 [3] y es una versión mejorada con capacidades plurilingües y mejor comprensión semántica. También experimenté con Siglip [4] y *Perceptual Encoder* [5].

2.1.5 Tecnología 5: uv

uv [1] es una herramienta ligera escrita en Rust para gestionar proyectos de Python. En el proyecto utilizamos esta tecnología para gestionar las dependencias y compilar el paquete.

```
# crear .venv (si no existe) y sincronizar dependencias existentes
uv sync

# ejecutar script dentro de venv
uv run script.py

# compilar proyecto
uv build
```

[1] https://docs.astral.sh/uv/

Capítulo 3

Desarrollo de la práctica

En esta sección se describirá detalladamente el desarrollo de las actividades realizadas durante las prácticas. Se incluirán referencias a las tecnologías descritas anteriormente.

3.1 Actividades realizadas

3.1.1 Actividad 1: Familiarización con el problema

Código inicial (5 días)

La primera tarea que tuve que realizar fue enterarme de qué hacía el pequeño repositorio que me proporcionaron la primera semana en la empresa. Otros miembros del equipo habían preparado hace varios meses un pequeño prototipo del sistema para una demo, y ese sería mi punto de partida.

Dediqué mis primeros días a leer, ejecutar y tratar de entender el código (además de completar el procedimiento de *onboarding* de la empresa). Me di cuenta de que se podía usar una multitud de modelos, y decidí evaluarlos para ver cuáles serían utilizados en la versión definitiva.

Papers (3 días)

También dediqué varíos días a leer *papers* sobre el tema, porque no conocía las técnicas *state-of-the-art* de *image retrieval*. Aquí descubrí multitud de modelos, técnicas y conceptos que no conocía de antemano.

3.1.2 Actividad 2: *Framework* de evaluación

Datasets (4 días)

Para evaluar los distintos modelos necesitaba un dataset de imágenes duplicadas/casi duplicadas. Lo más cómodo era que estuviese etiquetado, pero también me servía un dataset no etiquetado pequeño donde fuese factible etiquetarlo a mano yo mismo.

El primer dataset que utilicé aparece en multitud de *papers*: UKBENCH [6]. Lo descargué de una web de archivo[1] porque la fuente original ya no está disponible. Está compuesto de 2550 grupos de 4 imágenes, que son fotografías de objetos cotidianos tomadas desde ángulos ligeramente distintos.

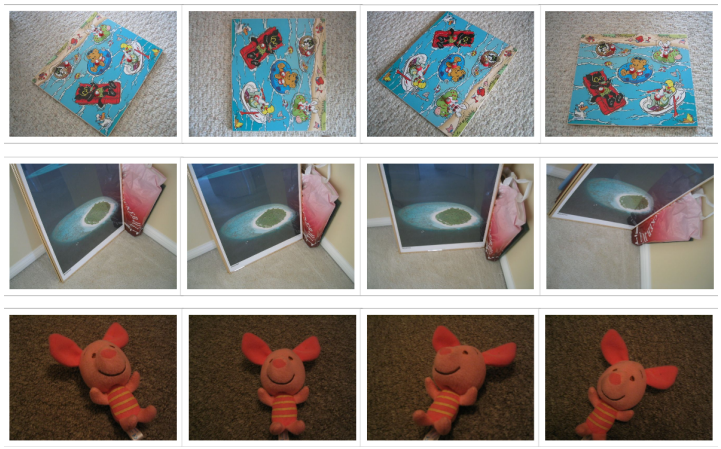

Figura 3.1: Ejemplo de muestras del dataset UKBENCH

El único inconveniente de este dataset es que es demasiado genérico para el caso de uso del sistema. Las imágenes caen en todo tipo de categorías: muebles, juguetes, plantas, personas, etc. Es por esto que decidí buscar otro dataset más ajustado a la temática de la práctica (que no revelaré por el acuerdo de confidencialidad). Después de buscar largo y tendido, encontré uno de 4000 imágenes, con el único inconveniente de que tuve que etiquetar manualmente en un subconjunto de 1000 cuáles eran duplicadas y cuáles no.

Evaluación (3 días)

En ambos datasets evalué los modelos que se listaban en el repositorio inicial. Para cada modelo, para cada imagen, buscaba las k más similares en la BBDD (todo el dataset) y el modelo debía encontrar las otras imágenes que tenemos etiquetadas como duplicadas.

[1] https://archive.org/details/ukbench

Interpretación de resultados (3 días)

Antes siquiera de buscar el segundo dataset, podía apreciar fácilmente cómo algunos modelos eran mucho mejores que otros. Alexnet destacaba por encima de GoogleNet, Densenet-121 y demás a pesar de ser el más antiguo y el más ligero de todos ([7]). CLIP 2.1.4 era ligeramente mejor que Alexnet, pero también era considerablemente más lento.

Al evaluar en el dataset de la temática específica, los resultados eran parecidos. Alexnet y CLIP eran los claros ganadores. Fue durante esta fase cuando me planteé buscar un modelo más cercano al *SOTA* para ver si nos podía aportar mejor rendimiento, y fue aquí donde encontré Siglip2. Siglip2 resultó ser más lento aún que CLIP, pero también presentaba métricas superiores.

Soluciones alternativas (4 días)

Mi equipo había desarrollado un prototipo alternativo para aquella demo en el pasado, una solución basada en *deep learning* diferente a la que estaba desarrollando yo. También evaluamos esta opción en los mismos datasets, pero resultó ser inferior y menos escalable por lo que terminamos descartándola.

Además, durante mi lectura de *papers* inicial 3.1.1 había aprendido sobre varias técnicas sin *deep learning* para búsqueda de imágenes duplicadas. Estos sistemas empleaban *hashing* u otras técnicas de visión artificial como FFT, y efectivamente presentaban resultados muy competentes y operaban a velocidades récord. El factor que nos hizo descartar estos métodos fue que que solo sirven para duplicados exactos. En nuestro caso de uso sabríamos que tendríamos duplicados parciales (por ejemplo, fotos de un objeto desde ángulos distintos como en UKBENCH), y ahí estas técnicas dejarían de funcionar.

3.1.3 Actividad 3: Integración

Documentación (2 días)

La mayoría del código ya se fue documentando según se añadía, tanto *docstrings* como comentarios de línea para aclaraciones puntuales. Aún así, al final terminé de documentar algún segmento que faltaba. Adicionalmente, modifiqué el README para incluir todo tipo de detalles sobre configuración, información y ejecución del proyecto, además de imágenes con ejemplos. Todo esto asegura que el proyecto está en condiciones para cualquier desarrollador que contribuya en el futuro.

Preparar y construir el paquete (1 día)

Una vez terminada una versión inicial del sistema, empezamos a integrarlo con el resto de código de la empresa. Lo único que tuve que hacer fue estructurar adecuadamente el código como un módulo con un `__init__.py` que recoja los nombres de los métodos principales y ejecutar `uv build`.

Capítulo 4

Conclusiones

En esta sección se aportarán unas conclusiones generales. Se describirán las relaciones con la titulación, y daré mi opinión general sobre del desarrollo de la práctica.

4.1 Relaciones con la titulación

En la fase inicial 3.1.1 de familiarización con el código y lectura de *papers*, traté con bastante material relacionado con la asignatura de Recuperación de Información y Minería Web. La principal diferencia con los sistemas que habíamos tratado en clase es que aquí los documentos eran imágenes, no texto, y podría haber estado bien estudiar algo de *image retrieval* durante la asignatura. También aprendí sobre modelos basados en *transformers* como los que habíamos visto en Aprendizaje Profundo.

En la fase de evaluación del sistema 3.1.2, recordé lo estudiado en Fundamentos de Aprendizaje Automático y Modelos Avanzados de Aprendizaje Automático I y II al buscar datasets y métricas pertinentes para la tarea. Algunas de las soluciones alternativas también tocaban técnicas de Principios de Visión por Computador y Visión por Computador Avanzada, como SIFT, FFT y *Haar Wavelets*. En la asignatura de VCA podría haber sido interesante revisar más en profundidad el *SOTA* de *Visual Transformers*, por ejemplo.

Para la fase final de integración 3.1.3 empleé conceptos vistos en Programación II, Ingeniería de Software y Herramientas de Desarrollo y Despliegue, como documentación adecuada y gestión correcta de git.

4.2 Valoraciones de la práctica

En general, puedo decir que estoy muy satisfecho con las prácticas. He descubierto y he aprendido a usar multitud de tecnologías que usaré en el futuro, y me he adentrado temporalmente en el mundo empresarial, lo cual considero beneficioso.

Bibliografía

[1] TheAgileAlliance, "Manifesto polo desenvolvemento Áxil de software," 2001, consultado el 2026-01-01. [En línea]. Disponible en: https://agilemanifesto.org/iso/gl/manifesto.html

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021. [En línea]. Disponible en: https://arxiv.org/abs/2103.00020

[3] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, O. Hénaff, J. Harmsen, A. Steiner, and X. Zhai, "Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features," 2025. [En línea]. Disponible en: https://arxiv.org/abs/2502.14786

[4] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," 2023. [En línea]. Disponible en: https://arxiv.org/abs/2303.15343

[5] D. Bolya, P.-Y. Huang, P. Sun, J. H. Cho, A. Madotto, C. Wei, T. Ma, J. Zhi, J. Rajasegaran, H. Rasheed, J. Wang, M. Monteiro, H. Xu, S. Dong, N. Ravi, D. Li, P. Dollár, and C. Feichtenhofer, "Perception encoder: The best visual embeddings are not at the output of the network," 2025. [En línea]. Disponible en: https://arxiv.org/abs/2504.13181

[6] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 2161–2168.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.