Apunte clase 03/06/2025

Autor: Daniel Alonso Garbanzo Carvajal
Ingeniería en Computación
Tecnológico de Costa Rica
San José, Costa Rica
dgarbanzo@estudiantec.cr

***Abstract*—Este documento resume conceptos clave acerca de la evaluación de modelos predictivos, técnicas de escalado de características (normalización y estandarización) y fundamentos en redes neuronales. Se discuten métricas de clasificación, la importancia de un adecuado escalado de *features* y se presentan casos de estudio que ilustran aplicaciones prácticas en problemas de clasificación.**

I. Introducción

Esta clase abordó la Tarea 3, el Proyecto 3 y el concepto de Análisis de Componentes Principales (PCA). La Tarea 3 consiste en desarrollar un asistente conversacional utilizando **Retrieval Augmented Generation (RAG)** y **agentes con herramientas** para consultar apuntes de clase, enfatizando el preprocesamiento de datos, el uso de embeddings y bases de datos vectoriales. El Proyecto 3 se enfoca en la **clasificación de mariposas** mediante **transfer learning** con **autoencoders** (incluyendo el *denoising autoencoder* y la posible extensión a *Variational Autoencoders*), requiriendo el uso de **PyTorch Lightning**, **Hydra** y **Wandb** para la experimentación y cuantización de modelos. Finalmente, se revisó **PCA** como una técnica de **reducción de dimensionalidad** que identifica las direcciones de mayor varianza en los datos mediante el cálculo de *eigenvalues* y *eigenvectors* de la matriz de covarianza.

II. Tarea 3: Asistente Conversacional con RAG y Agentes

A. Objetivo

Desarrollar un asistente conversacional que pueda responder preguntas basándose en los apuntes de los estudiantes (formato CT Play) usando RAG y agentes con herramientas.

A1. Fuente de Datos: Los apuntes de clase de los estudiantes en formato CT Play. Será el conjunto de datos utilizado.

B. Proceso RAG (Retrieval Augmented Generation)

A. Tomar toda la información de los documentos de apuntes (texto).
B. Agregar **metadata oportuna** a los documentos para mejorar la recuperación (fecha del documento, nombre del archivo, autor del apunte). Esto ayuda a responder preguntas sobre *quién* o *cuándo*.
C. Procesar el contenido en **chunks** (pedacitos de texto).
   - Decidir tamaño de chunk y *overlapping* (solapamiento) entre chunks adyacentes. Un *overlapping* pequeño generará muchos vectores.
D. Cada chunk se pasa por un **modelo de embeddings de texto** para obtener una representación vectorial (ej. vector de 768 elementos) que representa la semántica del texto.
E. Estos vectores (embeddings) se almacenan en una **base de datos vectorial** (ej. Faiss, se puede usar localmente generando un archivo).
F. Cuando el usuario hace una pregunta (*query*), esta también se pasa por el mismo modelo de embeddings para obtener su vector.
G. Se realiza una **búsqueda de semejanza** entre el vector de la pregunta y todos los vectores almacenados en la base de datos vectorial.
   - Se recomienda usar **comparación de coseno** para encontrar los chunks más semejantes.
H. La información recuperada (los chunks más relevantes) se utiliza como **contexto** para un **modelo de lenguaje grande (LLM)**.
I. El LLM utiliza este contexto y la pregunta del usuario para generar una respuesta coherente.

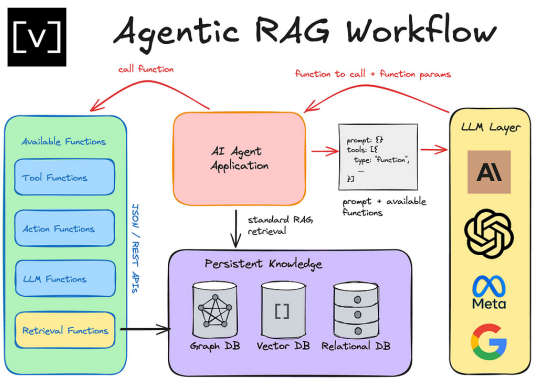

Figura 1. Diagrama de flujo del proceso RAG (Retrieval Augmented Generation).

C. Uso de Agentes y Herramientas (Tools)

A. Crear **dos herramientas (tools)**:

- Una para realizar el proceso RAG (extracción de la base de datos vectorial). Se recomienda/obliga a utilizar **LangChain** para esta herramienta.
- Otra para realizar búsquedas en internet.

B. Un **agente (orchestrator)**, que será un LLM, tomará las decisiones sobre cuándo usar la herramienta RAG o la herramienta de búsqueda en internet.

C. El **comportamiento del agente** se define mediante un **prompt**.
- Este *prompt* instruye al LLM cómo debe comportarse (ej. amigable) y qué herramientas debe usar.
- Ejemplo: Si la pregunta es sobre los apuntes, usar RAG Tool; si el usuario pide explícitamente buscar en internet, usar Internet Tool.

D. El agente debe **guardar el contexto** de las interacciones previas dentro de la **misma sesión** para facilitar la conversación. No es un histórico permanente, sino por sesión. Debe recordar preguntas anteriores.

D. Consideraciones Técnicas y Económicas

- Las búsquedas en internet solo se deben realizar cuando el usuario lo solicite explícitamente.
- Se recomienda usar el modelo `gpt-3.5-turbo` de OpenAI para la generación de texto (eficiencia de costo). Otros modelos (Claude, etc.) aceptables.
- Se recomienda usar un LLM (ej. `gpt-3.5-turbo`) como orquestador.
- El asistente debe ser lo **más preciso posible** y evitar respuestas como "no he encontrado la información"si la información está en los documentos.
- Crucial un **buen preprocesamiento** del texto de los PDFs para manejar problemas como tildes, caracteres especiales o errores de extracción.
- La interfaz de usuario debe ejecutarse en el navegador. Se sugiere usar **Streamlit**.

E. Rúbrica de la Tarea 3

- Indexación correcta de documentos y almacenamiento de metadata.
- Herramienta de búsqueda RAG funcional.
- Diseño efectivo del prompt del agente.
- Agente con tools incorporados y capacidad de orquestación.
- Interfaz de usuario funcional.
- Respuestas precisas a preguntas sobre metadata (autores, fechas, resúmenes) y contenido indexado.

F. Realización

La Tarea 3 se realiza en los **grupos de proyecto**.

III. Proyecto 3: Clasificación de Mariposas con Transfer Learning y Autoencoders

A. Objetivo General

Aplicar técnicas de **transfer learning** para crear **clasificadores multiclase de especies de mariposas** utilizando arquitecturas **autoencoders**. Explorar si el preentrenamiento de un autoencoder (especialmente *denoising*) puede mejorar el rendimiento del clasificador con datos etiquetados limitados.

B. Dataset

- Butterflies 100 species. Imágenes de entrenamiento, validación y testing. Clases pueden estar desbalanceadas.
- Seleccionar solo las **30 especies con mayor cantidad de muestras**.
- Mover aproximadamente **20 muestras** del set de entrenamiento de cada una de estas 30 especies al set de testing (el testing set original es pequeño).
- Imágenes pueden ser redimensionadas a un mínimo de 128x128 píxeles (preferible 224x224).

C. Herramientas Requeridas

- **PyTorch Lightning**:
  - Crear clase que herede de `LightningDataModule` para cargar datos.
  - Crear clase que herede de `LightningModule` para el modelo.
  - Implementar al menos métodos `training_step`, `testing_step`, y `configure_optimizers`.
- **Hydra**: Gestión de experimentos y configuraciones de hiperparámetros vía archivos YAML. Permite definir y cambiar configuraciones (modelos, entrenamiento) fácilmente sin modificar el código.
- **Wandb (Weights & Biases)**: Rastreo y visualización de métricas (loss, accuracy, etc.) y comparación de experimentos.
- **Early Stopping**: Implementar callback en PyTorch Lightning para detener el entrenamiento y evitar *overfitting*. Otros callbacks (ej. disminución del *learning rate*) también pueden usarse.

D. Estructura del Código

- Jerarquía de archivos con scripts auxiliares para clases y funciones.
- La ejecución principal de los experimentos y entrenamiento debe estar orquestada desde un **Jupyter Notebook**.
- Todos los archivos utilizados deben estar en el entregable.

E. Hipótesis y Experimentos

E1. Hipótesis 1: Impacto del Autoencoder en el Transfer Learning con Datos Etiquetados Limitados:

a. Pregunta: ¿En ausencia de muestras etiquetadas, es posible entrenar un autoencoder para aprender representaciones latentes útiles para un clasificador, logrando resultados mejores que entrenar un clasificador solo con un pequeño subconjunto etiquetado?

b. Simulación de Datos: Tomar el set de entrenamiento y dividirlo artificialmente en:

- **Set de datos sin labels:** Una parte de cada clase, simulando la ausencia de etiquetas.
- **Set de datos con labels:** El resto, con etiquetas disponibles.

Dos cortes de datos a experimentar:

A. 70 % sin labels / 30 % con labels
B. 90 % sin labels / 10 % con labels

c. Paso 1: Entrenamiento del Autoencoder (Unsupervised Pre-training):

- Entrenar un **autoencoder** utilizando el **set de datos sin labels** para aprender una representación latente.
- El autoencoder debe basarse en la arquitectura **UNet**.
- Las **skip connections son obligatorias**.
- Se pueden considerar modificaciones a la arquitectura UNet (agregar otros módulos, cambiar *skip connections*, etc.).

d. Paso 2: Entrenamiento de Clasificadores (con Transfer Learning): Entrenar **tres modelos clasificadores** utilizando el **set de datos con labels**. Todos usarán la arquitectura del *encoder* del autoencoder como base para la extracción de características. La parte del clasificador (después del encoder, ej. *fully connected layers*) puede ser cualquier arquitectura adecuada.

F. Modelo A (Baseline - Desde Cero)

- Clasificador entrenado **desde cero**.
- Se toma la arquitectura del encoder del autoencoder, pero **sin los pesos preentrenados**.
- Se adjunta un clasificador (*fully connected*) y todo el modelo (*encoder* + clasificador) se entrena *end-to-end* desde cero con el set de datos con labels.

G. Modelo B (Transfer Learning con Encoder Preentrenado)

Clasificador utilizando el **encoder preentrenado** del autoencoder (del Paso 1). Se toman los pesos aprendidos por el autoencoder en el Paso 1. Se deben generar **dos variantes** del Modelo B:

A. **Modelo B1 (Encoder Fijo):**
   - Utilizar el encoder preentrenado como **extractor de características fijo**.
   - Los pesos del encoder se **congelan** (no se reentrenan).
   - Solo se entrena el clasificador adjunto (*fully connected*) con el set de datos con labels.

B. **Modelo B2 (Fine-tuning del Encoder):**
   - Utilizar el encoder preentrenado, pero **permitir que sus pesos se re-entrenen** (*fine-tuning*) durante el entrenamiento del clasificador.
   - Se entrena todo el modelo (*encoder* + clasificador) con el set de datos con labels, permitiendo ajustes tanto en el clasificador como en el encoder preentrenado.

a. Objetivo de Comparación: Comparar si el transfer learning con el autoencoder preentrenado (Modelos B1 y B2) produce mejores resultados que entrenar desde cero (Modelo A) con un subconjunto pequeño de datos etiquetados.

b. Modelos Finales: En total, se entrenarán **seis modelos finales** (A, B1, B2 para cada corte de datos: 70 % sin labels / 30 % con labels; y 90 % sin labels / 10 % con labels).

G1. Hipótesis 2: Robustez de Representaciones Latentes con Denoising Autoencoder y Clustering:

a. Pregunta: ¿Agregar ruido (*salt-and-pepper*) a las imágenes (Denoising Autoencoder) mejora las representaciones latentes?

b. Paso 1: Entrenamiento del Denoising Autoencoder:

- Reutilizar uno de los cortes de datos del experimento 1 (Hipótesis 1).
- Entrenar un **denoising autoencoder** agregando ruido (ej. *salt & pepper noise*) a las imágenes de entrada antes de pasarlas al autoencoder.

c. Paso 2: Visualización del Espacio Latente:

- Después de entrenar el denoising autoencoder, visualizar los **vectores latentes** del set de datos **sin labels** utilizando **TSNE** (t-Distributed Stochastic Neighbor Embedding).
- Se espera que los datos con características similares (misma especie) se encuentren cerca en el espacio latente reducido.

d. Paso 3: Clustering:

- Aplicar un algoritmo de **clustering**, como **K-Means**, a los vectores latentes del set de datos sin labels para asignar etiquetas basadas en la agrupación.

e. Paso 4: Análisis de Clusters:

- Analizar los clusters resultantes.
- Se debe **comparar 10 imágenes de al menos cinco clusters**.
- Teóricamente, la mayoría de las imágenes dentro de un mismo clúster deberían ser similares o de la misma especie.
- El análisis debe indicar qué tan bien quedaron los clusters.

H. Resultados Finales y Entregables (Proyecto 3)

- Seleccionar los **tres mejores modelos** (entre las variantes de A, B1, B2 de la Hipótesis 1) basándose en métricas de rendimiento.
- Convertir estos tres mejores modelos a **modelos cuantizados**.
- Realizar **comparaciones** de **latencia en respuesta, tamaño y rendimiento** de los modelos cuantizados. Incluir este análisis en el informe.
- **Entregables**:
  - Jupyter Notebook con las exploraciones y ejecuciones.

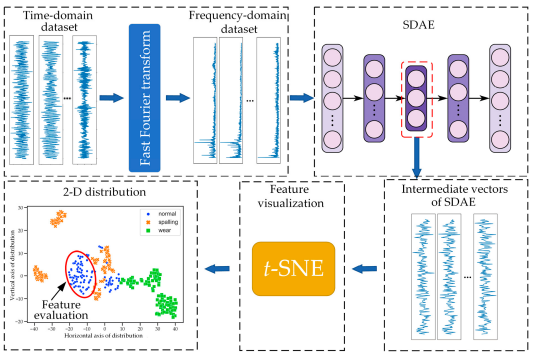

Figura 2. Diagrama de flujo del Denoising Autoencoder, visualización TSNE y clustering.

- Informe detallado con estructura definida.
- Todos los archivos de código (scripts auxiliares).

I. *Aspectos Extra (Proyecto 3) para Puntos Adicionales*

- Modificar la arquitectura de UNet y demostrar si mejora los resultados (puntos dados aunque los resultados sean peores, siempre que se presente también la versión base de UNet).
- Hacer que el denoising autoencoder sea un **Variational Autoencoder (VAE)**, modificando el encoder para que produzca la media y desviación estándar de una distribución.
- Comparar el VAE con el autoencoder estándar utilizando una métrica como la **divergencia de Kullback-Leibler (KL Divergence)** en la función de pérdida para evaluar la calidad del autoencoder.

J. *Nota*

El Proyecto 3 es **largo** y se recomienda empezar lo antes posible. Se realiza en **grupos**.

IV. Materia Vista en Clase: Análisis de Componentes Principales (PCA)

A. *Concepto*

Herramienta de **reducción de dimensionalidad** y **extracción de características**.

B. *Propósito de PCA*

- Aplicar transformaciones lineales a los datos para reubicarlos en direcciones (vectores) que capturan la **mayor variabilidad**.
- Reducir la dimensionalidad del dataset **sin perder tanta información**.
- Maximizar la varianza de los datos y disminuir la influencia de variables que aportan poca información.
- Eliminar variables correlacionadas y crear nuevas variables independientes (ortogonales).

C. *¿Cuándo utilizarlo?*

- Cuando se tienen muchas variables y no se sabe cuáles seleccionar.
- Cuando se quiere asegurar variables totalmente independientes.

D. *Limitación*

- La transformación lineal aplicada **no refleja características del mundo real**; los componentes resultantes no tienen un significado interpretable en el contexto original (ej. altura, peso).

E. *Conceptos Clave*

- **Varianza:** Medida de la dispersión o variabilidad de los datos en una dirección. PCA busca direcciones con alta varianza.
- **Covarianza:** Medida de la relación lineal conjunta entre dos variables. La matriz de covarianza muestra la varianza de cada variable en la diagonal y la covarianza entre pares de variables fuera de la diagonal.
- **Correlación:** Versión estandarizada de la covarianza, que indica la fuerza y dirección de la relación lineal. La matriz de correlación tiene 1s en la diagonal (correlación perfecta de una variable consigo misma).
- **Transformación Lineal:** Una operación que mapea vectores de un espacio a otro, que puede cambiar su tamaño, dirección o sentido. Geométricamente, puede estirar o encoger el espacio.
- **Span:** El conjunto de todos los vectores que se pueden obtener como combinaciones lineales de un conjunto dado de vectores. Representa la "dirección.º el subespacio que abarcan.
- **Eigenvectors (Vectores Propios):** Vectores cuya dirección **no cambia** al aplicar una transformación lineal específica; solo pueden escalar (crecer o disminuir de tamaño) o cambiar de sentido. Son las direcciones importantes donde los datos varían más. Para cada dimensión de los datos, hay al menos un *eigenvector*. Los *eigenvectors* de PCA son ortogonales.
- **Eigenvalues (Valores Propios):** Un escalar asociado a cada *eigenvector* que indica **cuánto** el *eigenvector* escala (crece o disminuye) al aplicar la transformación lineal. En PCA, el *eigenvalue* representa la **varianza** de los datos en la dirección del *eigenvector* correspondiente. Los *eigenvectors* con los *eigenvalues* más grandes son los más importantes porque capturan la mayor variabilidad de los datos.

F. *Algoritmo de PCA*

A. **Estandarizar los datos:** Asegurar que cada característica tenga media 0 y varianza 1. Necesario para calcular la matriz de covarianza/correlación.
B. Calcular la **matriz de covarianza** o la **matriz de correlación** de los datos estandarizados.
C. Calcular los **eigenvalues y eigenvectors** de la matriz de covarianza/correlación.

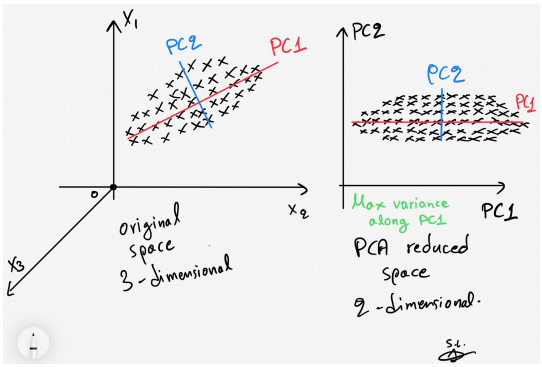

Figura 3. Diagrama conceptual que muestra datos dispersos y las direcciones de los componentes principales.

- Se resuelve la ecuación característica: $\det(\Sigma - \lambda I) = 0$, donde Σ es la matriz de covarianza/correlación, λ son los eigenvalues, e I es la matriz identidad.
- Una vez encontrados los eigenvalues, se resuelve un sistema de ecuaciones para encontrar los eigenvectors asociados.

D. **Seleccionar los K componentes principales:**
- Ordenar los eigenvectors en función de sus eigenvalues asociados en orden descendente.
- Seleccionar los K primeros (los que tienen los eigenvalues más grandes). Estos K eigenvectors formarán una nueva matriz de vectores.

E. **Transformar los datos:**
- Multiplicar la matriz de datos original (centrada o estandarizada) por la matriz de los K eigenvectors seleccionados.
- Esto proyecta los datos a un nuevo espacio dimensional definido por los K componentes principales. La nueva matriz resultante tiene los datos transformados, donde cada columna es un componente principal.

V. Conclusiones

Esta clase proporcionó una visión integral de tres pilares fundamentales en inteligencia artificial: la implementación de asistentes conversacionales avanzados, el desarrollo de modelos de clasificación de imágenes con transfer learning utilizando autoencoders, y la comprensión de la reducción de dimensionalidad con PCA. Se detallaron los requisitos para las asignaciones prácticas (Tarea 3 y Proyecto 3), enfatizando la importancia de las herramientas de desarrollo y la experimentación sistemática. La exploración de PCA reafirmó la necesidad de entender las propiedades estadísticas de los datos para optimizar su representación. La aplicación correcta de estas técnicas es esencial para mejorar el rendimiento de los algoritmos en problemas reales.

Referencias

[1] Ü-Net Architecture Explained" *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/u-net-architecture-explained/. [Accessed: Jun 4, 2025].

[2] "What is Retrieval Augmented Generation (RAG)?," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/what-is-retrieval-augmented-generation-rag/. [Accessed: Jun 4, 2025].

[3] "Build RAG Pipeline Using Open Source Large Language Models," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/build-rag-pipeline-using-open-source-large-language-models/. [Accessed: Jun 4, 2025].

[4] "Principal Component Analysis (PCA)," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/principal-component-analysis-pca/. [Accessed: Jun 4, 2025].