Curso: Inteligencia Artificial
Profesor: MSc. Steven Pacheco Portuguez
Semestre: II 2024
Valor: 25%
Fecha de entrega: 19 de noviembre

Proyecto III

Este proyecto tiene como objetivo aplicar técnicas de Transfer Learning para crear clasificadores multiclase de especies de aves. Estos clasificadores serán construidos utilizando primeramente arquitecturas autoencoders sin etiquetas de manera self-supervised, entrenado con el objetivo de eliminar ruido de las imágenes.

Recomendaciones:

En caso de no contar con GPU en una unidad local, pueden utilizar Google Colab para realizar el proyecto, en este caso debe subir el dataset dentro de Google Drive para poder accederlo.
Guia Colab

Birds 525 Dataset:

Para este proyecto debe utilizar el dataset Birds 525 Especies, que contiene una clasificación de imágenes de 525 especies de aves, con imágenes de entrenamiento de 130 por especie de dimensiones 224x224x3. Puede reducir la dimensionalidad de estas imágenes hasta un mínimo de 128x128x3 para reducir los tiempos de entrenamiento de los modelos.

Seleccione **solamente las 20 especies** con mayor cantidad de muestras, por lo tanto su clasificador solamente deberá clasificar estas especies. Además mueva unas 20 muestras del set de entrenamiento de cada especie al set de testing para incrementar su volumen, porque actualmente solo tiene 5 muestras para testing.

Hipótesis:

1. En ausencia de muestras etiquetadas, es posible entrenar un modelo autoencoder que reconstruya imágenes para aprender una representación latente base de cada clase para aplicar técnica de transfer learning y crear un clasificador más robusto con un subconjunto pequeño que si tiene labels, y obtener mejores resultados que solamente utilizar el mismo subconjunto pequeño para entrenar un clasificador.
2. Agregar ruido a las imágenes de entrada del autoencoder para crear un Denoising Autoencoder que permita aprender representaciones más robustas y generar mejores vectores del espacio latente para el clasificador.

Herramientas a utilizar:

Diseñe su modelo utilizando PyTorch Lightning, un framework basado en PyTorch con estructuras para abstraer detalles del entrenamiento haciéndolo escalable. Debe crear su propia clase de carga de datos utilizando “LightningDataModule” y su modelo utilizando “LightningModule”, acá debe

redefinir como mínimo los métodos de training_step, test_step, configure_optimizers. Adicionalmente utilice el Callback de EarlyStopping durante el proceso de entrenamiento para evitar Overfitting del modelo.

Para la ejecución y corrida con los diversos experimentos y configuraciones de hiperparámetros debe utilizar Hydra, un framework open-source que simplifica el desarrollo de experimentos basado en una configuración jerárquica.
Config Notebooks
Tutorial Simple

Debe utilizar la herramienta Weights and Biases, para tracear y visualiza varios aspectos del proceso de entrenamiento del modelo en tiempo real, como loss, accuracy, y otras métricas de evaluación que considere importantes, además de la reconstrucción de imágenes de cada autoencoder.
Logging wandb

Como tener todo en un mismo Jupyter Notebook puede ser complicado y extenso, pueden crear la jerarquía de archivos que requieran y utilizarlas como archivos auxiliarías en formato script para el diseño y control de los experimentos, sin embargo, la ejecución del entrenamiento debe estar en un Jupyter Notebook. Todos los archivos utilizados deben estar dentro del entregable.

Experimentos

Para ambos experimentos

Experimento 1 (Hipótesis 1)

Tome el set de datos de entrenamiento y simule que una parte de cada clase no contiene labels, denominado **set de datos sin labels**, el otro restante será denominado **set de datos con labels**.

Utilice el **set de datos sin labels** para reconstruir las muestras utilizando un Autoencoder para aprender una representación latente de los datos. El Autoencoder debe estar basado en la Arquitectura U-Net. Es obligatorio la permanencia de Skip Connections.

Con el **set de datos con labels** deberá entrenar dos modelos clasificadores. Cada clasificador debe de respetar las restricciones de dimensiones del autoencoder, puede utilizar cualquier arquitectura para este proceso. Debe utilizar el mismo corte de datos para los clasificadores.

El primer clasificador (A) será entrenado desde 0, mientras que el otro clasificador debe ser entrenado utilizando con base el Encoder del Autoencoder (B). Del modelo clasificador (B) debe generar dos variantes, congelar los pesos del autoencoder **después** del

entrenamiento y solamente ajustar los pesos del clasificador (B1), permitir que los pesos del autoencoder **después** del entrenamiento se ajusten (B2).

Convierta los tres modelos con mejores resultados de acuerdo a su criterio a modelos cuantizados, y realice una comparación de latencias en respuesta, tamaño, y rendimiento, incluya este análisis en su informe.

Debe realizar dos ejecuciones utilizando porcentajes diferentes del **set de datos sin labels** y el **set de datos con labels**, ejemplo:

1. 70% sin labels y 30% con labels,
2. 90% sin labels y 10% con labels

Debe realizar las comparaciones de los tres modelos, utilizando métricas para determinar el rendimiento, de acuerdo con sus resultados exponga sus conclusiones.

Experimento 2 (Hipótesis 2)

Genere un modelo de clasificación supervisado Denoising Autoencoder agregando "Salt and Pepper Noise" a las imágenes. Puede reutilizar uno de cortes de datos del experimento 1 (no es necesario hacer 2 corridas).

Además, debe realizar una visualización de los vectores latentes del Denoising Autoencoder del **set de datos sin labels** utilizando t-SNE para observar si el espacio latente aprendió a modelar las clases. Se esperaría que los vectores presenten datos de características similares, por lo tanto utilice un algoritmo de clustering (K-means) para otorgar etiquetas a cada una de las muestras utilizando unsupervised learning.

Entregables del proyecto:

- Jupyter notebook con la exploración de los datos y sus visualizaciones, además de incluir el resultado de aplicación de los modelos, así como la evaluación de métricas para cada. Debe incluir los resultados.
- Informe: Cree un informe en formato de artículo científico (Latex) donde describa los experimentos realizados y realice una comparación de los resultados de cada modelo para cada conjunto de datos.

  La estructura del informe básica es la siguiente: Abstract, Introducción, Metodología, Resultados, Discusión y Conclusiones, Bibliografía. Debe entregar el código fuente y el PDF.

- El no cumplir con las herramientas de desarrollo, archivos o formatos de documentación solicitados la revisión de las rúbricas descritas a continuación puede ser anulada. Es fundamental seguir las indicaciones.

Criterios	Puntuación máxima	Puntuación obtenida
Experimento 1		
Corrida Autoencoder 70%		
Clasificador pesos entrenados congelados	7.5	
Clasificador pesos autoencoder entrenados sin congelar	7.5	
Clasificador sin autoencoder entrenado.	5	
Corrida Autoencoder 90%		
Clasificador pesos entrenados congelados	7.5	
Clasificador pesos autoencoder entrenados sin congelar	7.5	
Clasificador sin autoencoder entrenado.	5	
Extras		
Modificar la arquitectura U-Net – Justificar	5	
Aspectos Generales		
Comparación modelos cuantizados	5	
Creación de DataModule	5	
Redefinición de métodos de LightningModule	5	
Logger de información de WandB	5	
Análisis de resultados	15	
Experimento 2		
Denoising Autoencoder	10	
Visualización espacio latente t-SNE	5	
Ejecución y visualización de K-Means del espacio latente	5	
Análisis de resultados	5	
	Extras	
Denoising Variational Autoencoder	10	