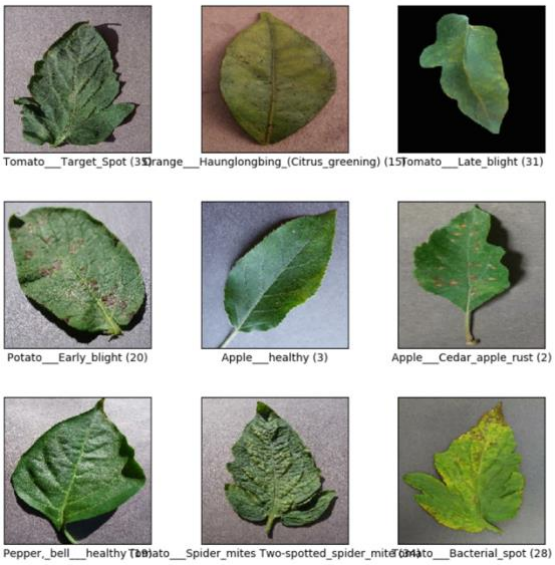
Proyecto 3 – Plant Disease Self-Supervision



En el área de biodiversidad es posible entrenar modelos de ML para ayudar en las tareas de clasificación. Generalmente se ha trabajado en clasificación de especies de plantas, pero también es posible detectar enfermedades de las plantas con base en fotos de sus hojas. Uno de los problemas es que no todos los datos que existen tienen labels, sin embargo sería un desperdicio no hacer uso de tantos datos sin labels. Necesitamos pensar maneras de lograr aprendizaje de esos datos, y complementar con los que si tienen labels, los cuales suelen ser mucho menos en la vida real.

Este proyecto consiste en comparar modelos de clasificación de enfermedades de plantas, con un total de 38 categorías. La diferencia es que unos de los clasificadores serán construidos sobre un autoencoder, que a su vez fue entrenado de forma self-supervised con un conjunto de datos sin labels. Adicionalmente haremos el modelo más robusto usando ruido en la imagen de entrada.

Plant Disease Dataset

Link: https://www.tensorflow.org/datasets/catalog/plant_village

Este set de datos consiste en 54mil~ imágenes divididas entre 38 categorías entre las cuáles hay hojas de plantas con o sin enfermedad. El set de datos permite entonces detectar si hay enfermedad o no por especie de planta. Nuestro problema entonces es un problema de clasificación de plantas, donde algunas clasificaciones señalan que la hoja está enferma.

Hipótesis

1. En ausencia de datos con labels, es posible entrenar un autoencoder reconstruyendo datos sin labels, hacer transfer learning con el subconjunto pequeño que sí tiene labels, y obtener mejores resultados que con solo usar el mismo subconjunto pequeño con labels directo para entrenar un clasificador.
2. Al agregar ruido a la entrada del autoencoder, es posible aprender representaciones más robustas que mejoran la clasificación y, por ende, la representación latente.

Experimentos

Los experimentos por realizar van a poner a prueba las hipótesis planteadas anteriormente. A continuación, se describen los dos experimentos a correr:

Experimento 1 (asociado a la hipótesis 1)

Para el primer experimento se debe tomar el set de datos y simular que una buena parte de este no tiene labels. Esto porque en la práctica, es más común tener acceso a datos sin labels que con labels. La parte del set de datos sin labels será usada para aprender a reconstruir los datos (imágenes) con un Autoencoder y aprender una representación latente. Luego, la parte restante con labels, será subdividida en training y testing, para entrenar 2 modelos de clasificación, donde uno de ellos será entrenado desde cero (sin usar nada del autoencoder, ni pesos pre-entrenados), y otro usando el encoder del autoencoder como base. El clasificador que usa el autoencoder como base tendrá a su vez dos variantes: una dejando los pesos aprendidos del encoder congelados durante el entrenamiento del clasificador, y otra permitiendo que cambien.

Deben crear entonces 2 corridas cambiando los porcentajes lo suficiente para poder observar la relación de rendimiento de los clasificadores desde cero versus basados en autoencoder, además de comparar el rendimiento si se usa el encoder como feature extractor (congelado) o si puede cambiar, y sacar sus conclusiones al respecto.

La siguiente tabla muestra un ejemplo de dos posibles corridas:

# Corrida	Datos sin labels (Porcentaje usado para entrenar el autoencoder)	Con labels (training) (Porcentaje usado para entrenar los clasificadores)	Con labels (testing) (Porcentaje usado para probar los clasificadores)	Clasificadores	Tipo de clasificador
1	80%	10%	10%	Clasificador A	Sin autoencoder
				Clasificador B	Pesos de autoencoder congelados
				Clasificador C	Pesos de autoencoder no congelado
2	50%	35%	15%	Clasificador D	Sin autoencoder
				Clasificador E	Pesos de autoencoder congelados
				Clasificador F	Pesos de autoencoder no congelado

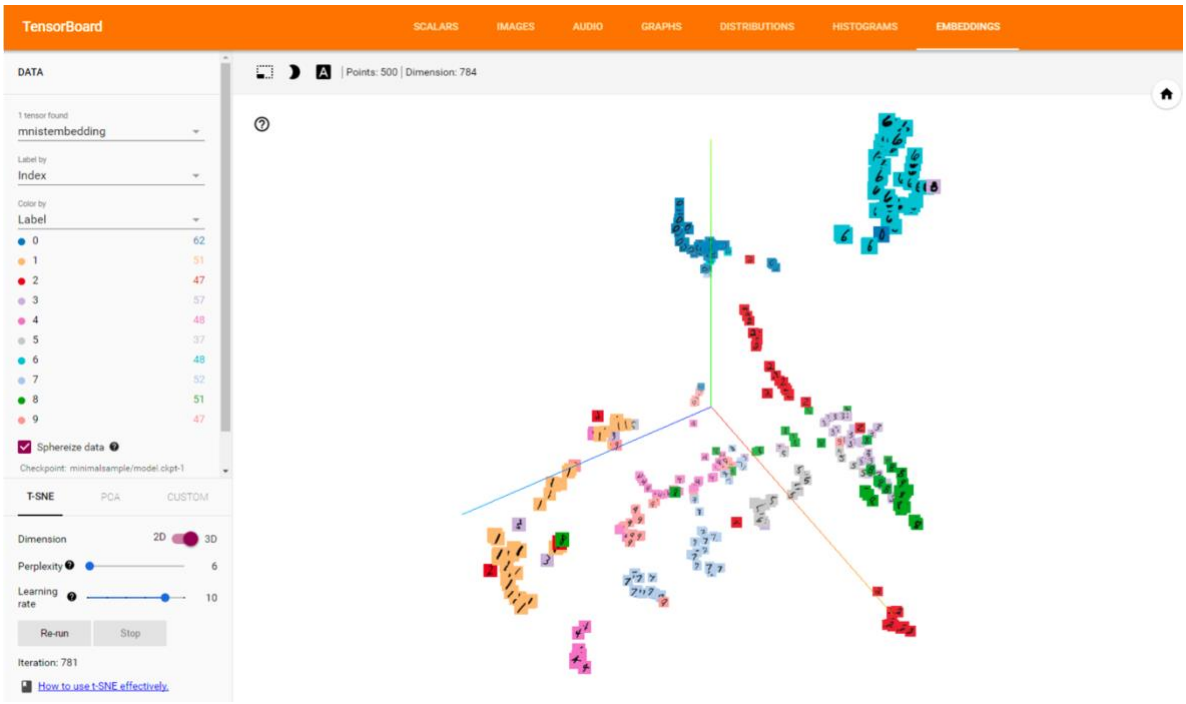
Es importante aclarar que se debe usar el mismo corte de training y testing entre clasificadores de la misma corrida para que la comparación sea justa. También se debe usar el split de forma estratificada.

Experimento 2 (asociado a la hipótesis 2)

Se deben comparar dos clasificadores basados en autoencoders, con la diferencia de que uno de los autoencoders fue entrenado agregando algún tipo de ruido a las imágenes de entrada (por ejemplo salt and pepper noise). Este tipo de autoencoder se conoce como Denoising Autoencoder. Se puede reutilizar alguno de los cortes de datos usados en el experimento 1.

Adicionalmente, se debe hacer una visualización de los vectores latentes de las imágenes "sin labels", mostrando los labels (que sí conocemos, solo que no los usamos en el entrenamiento para simular un faltante de labels) con colores o similares (Ver imagen adjunta abajo), para ver si en el espacio latente el modelo aprendió a modelar relativamente bien las clases, es decir, que items de la misma clase estén cerca. Por ejemplo, se esperaría que los vectores latentes del Denoising Autoencoder muestren clusters más claros que los del Autoencoder normal.

Para ello se puede usar alguna técnica de reducción de dimensionalidad para visualización de los vectores, como PCA o t-SNE.



Herramientas

Las redes deben implementarse en Pytorch, usando Python. Se recomienda el uso de GPUs en Google Colab, subiendo el set de datos en Google Drive. De tener GPU propio, todavía mejor. Otra opción para obtener GPUs es usar maquinas virtuales de la escuela, o bien en caso de tener la posibilidad, Azure o AWS son opciones viables. Para la visualización se puede usar Tensorboard, en particular la herramienta conocida como Projector.

Métricas/Conclusiones

Deben reportar Accuracy por clase y general, así como otras métricas como la matriz de confusión y sus derivados, en el caso de los clasificadores. En el caso de los autoencoders, deben elegir muy bien cuál loss function usar de reconstrucción y reportarla. Recuerden que los resultados deben ser en términos de test sets (aunque pueden mencionar los de training set, especialmente para hablar sobre Overfitting o Underfitting). Por cuestiones computacionales, no se recomienda hacer cross validation, salvo si cuentan con muy buen equipo computacional y el tiempo para hacerlo.

Pueden leer papers que han trabajado este mismo set de datos y darse ideas para aplicar en el proyecto, además de tomarlos como punto de comparación con sus propios experimentos.

Es importante escribir sobre las hipótesis y cómo los experimentos las prueban. Si podemos rechazar cada una de ellas o bien si no podemos rechazarlas de acuerdo a la evidencia de los experimentos.

Paper en Latex / PDF

Deben crear en latex (y entregar las fuentes de latex) un paper donde describen los experimentos realizados, su metodología. Deben incluir los resultados y por supuesto sus propias conclusiones. El documento debe seguir el template de la IEEE para publicaciones en ingeniería el cual pueden encontrar aquí:

https://www.ieee.org/conferences/publishing/templates.html

Evaluación

Tarea	Puntaje Máximo
Experimento 1, Corrida 1	
Clasificación con CNN directa sin autoencoder	10
Clasificación con CN basada en autoencoder, pesos congelados	10
Clasificación con CN basada en autoencoder, pesos sin congelar	10
Experimento 1, Corrida 2	
Clasificación con CNN directa sin autoencoder	10
Clasificación con CN basada en autoencoder, pesos congelados	10
Clasificación con CN basada en autoencoder, pesos sin congelar	10
Experimento 2	
Comparación de resultados de clasificadores basados en autoencoder con ruido y sin ruido	10
Visualización de vectores latentes	10
Puntos EXTRA: Visualización y comparación de vectores con un VAE	10
Paper y análisis de resultados global y comparativos, conclusiones	20
TOTAL	110