Proyecto II

Curso de Inteligencia Artificial
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica

I. Objetivo

Aplicar un experimento que permita validar la hipótesis de que al aplicar técnicas de destilado de modelos de grandes volúmenes de parámetros en modelos más pequeños se pueden resolver tareas igual de complejas pero con modelos más eficientes.

II. Modelo de detección de anomalías

Para el desarrollo de este proyecto debe usar el dataset propuesto en *MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection*. Un dataset de escenarios industriales reales con diferentes tipos de anomalías en la forma de detección de defectos en objetos o texturas.

Seleccione 10 clases del dataset las que usted más prefiera, y con este subconjunto vamos a entrenar distintos modelos para resolver un problema de detección de anomalías.

III. Modelos

Cada modelo debe estructurar el proyecto utilizando la librería **Hydra** para la gestión modular de configuraciones, asegurando la correcta separación de hiper parámetros entre el modelo, el entrenamiento y los registros experimentales.

La estructura mínima recomendada del proyecto es la siguiente:

```
conf/
 - config.yaml
 - model/
     - vae.yaml
 - trainer/
     - default.yaml
 - logger/
     - wandb.yaml
```

Cada módulo de configuración deberá permitir la ejecución de experimentos con distintos parámetros del modelo, tales como:

- Dimensión del espacio latente (z).
- Cantidad de épocas, tamaño de batch, o cualquier hiperparametro que requiera.

Además debe utilizar Pytorch Ligthning para las personalizaciones de los entrenamientos y creación de los modelos basado en las mejores prácticas de diseño de software que permita una correcto diseño 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.

Cada modelo debe de ser entrenado unicamente con datos de la clase sin defectos, **no incluir clases anomalas en el entrenamiento de los modelos**.

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.

A. Modelo clasificador CNN (Scratch y DestilationComo pod)

Para el siguiente modelo debe de crear una estructura base siguiendo la estructura de RESNET-18 para las primeras 3 convoluciones (conv1, con2_x, con3_x) (ver Figura 1), de acá en adelante coloque un clasificador (FC layer) a su gusto para crear un clasificador entre las distintas clases.

Una vez diseñado el modelo debe proceder hacer dos variantes A y B.

El modelo A será entrenado desde 0 es decir al inicio tendrá pesos colocados aleatoriamente y comenzará su proceso de entrenamiento.

El modelo B será entrenando siguiendo un proceso de destilado del modelo RESNET-18 siguiendo la técnica **teacher-student** donde el modelo RESNET-18 sirve como teacher y el modelo B como student.

Para esto debe investigar como aplicar correctamente un proceso de destilado de un modelo.

Es importante un buen diseño de modelo que permita obtener el vector de embeddings de salida de las capas convolucionales. Pues son los que luego permitirán diseñar el detector de anomalías.

Cada modelo debe ser entrenado al menos con 3 hiperpametros distintos para obtener buenos modelos y no solamente la primera combinación que obtengan.

B. Modelo C - Embedding de un autoencoder

Diseño un modelo de autoencoder basado en U-net que reconstruya las imágenes del set de entrenamiento seleccionado y también permita obtener el embedding correspondiente.

Este será entrenado completamente desde 0.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

Fig. 1. Arquitectura de la red ResNet-18.

IV. Evaluación de anomalías

Una vez entrenados los modelos, se deben calcular las representaciones latentes (*embeddings*) de las imágenes del conjunto de validación para estimar una métrica que permita, posteriormente, identificar los datos anómalos en el conjunto de prueba.

1) Ejemplo: Mahalanobis Distance: Una manera estadística de determinar si una nueva muestra pertenece o no a la distribución de los datos normales consiste en utilizar la **distancia de Mahalanobis**.

Esta métrica mide la distancia entre un vector de características (embedding) y la distribución multivariada estimada a partir de las muestras normales, considerando las correlaciones entre dimensiones.

1. Estimación de la distribución normal. A partir del conjunto de validación o entrenamiento correspondiente a la clase sin defectos, se extraen los embeddings de cada imagen mediante el modelo previamente entrenado (para cada modelo).

Cada embedding puede representarse como un vector $\mathbf{z}_i \in \mathbb{R}^d$. Con todos los embeddings del conjunto normal se calcula la media $\boldsymbol{\mu}$ y la matriz de covarianza $\boldsymbol{\Sigma}$:

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{z}_i, \quad \boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{z}_i - \boldsymbol{\mu})(\mathbf{z}_i - \boldsymbol{\mu})^T$$

De esta forma se modela la distribución normal como una distribución gaussiana multivariada $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, que representa los datos normales en el espacio de embeddings.

2. Cálculo de la distancia de Mahalanobis. Para una nueva muestra con embedding $\mathbf{z}_{test}$, se calcula su distancia a la distribución normal.

Esta distancia mide qué tan alejada se encuentra la muestra del centro de la distribución de los datos sin defectos, considerando la forma y correlaciones de dicha distribución.

Apartir de acá debe de averiguar como clasificar una anomalía o una clase sin defectos utilizando comparación de la distancia (e.g tomar el percentil).

Nota: El estudiante puede implementar también otras estrategias de detección, como la distancia euclidiana, reconstrucción basada en error (*reconstruction loss*).

V. Modelos cuantizados

Adicionalmente se desea hacer comparaciones de los resultados entre los modelos originales y los modelos cuantizados para hacer comparaciones de rendimiento. Para esto, 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.

VI. Análisis de Outliers mediante DBSCAN - Clustering

Una vez identificado el mejor modelo de detección de anomalías —ya sea el clasificador CNN entrenado desde cero, su versión distilada mediante *teacher–student*, o el modelo autoencoder basado en U-Net proceda a utilizar sus *embeddings* como insumo para realizar un análisis adicional mediante técnicas de agrupamiento no supervisado. En particular DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*), un método basado en densidad que permite identificar regiones de alta concentración en el espacio latente y, simultáneamente, detectar puntos aislados que pueden interpretarse como outliers o anomalías.

Extraiga los embeddings generados por el modelo seleccionado para cada imagen del conjunto de prueba. En estos

espacios latentes, las imágenes normales suelen agruparse de forma natural alrededor de regiones de alta densidad, mientras que las anomalías producen vectores que tienden a ubicarse lejos de dichas regiones. Con el fin de facilitar tanto la visualización como la separación estructural, aplique reducción de dimensionalidad con PCA y t-SNE

Una vez obtenidas las representaciones latentes reducidas aplique DBSCAN Desde la perspectiva de la detección de anomalías, los puntos etiquetados por DBSCAN como ruido constituyen una indicación natural de potencial anomalía, ya que representan vectores que se encuentran en zonas de baja densidad del espacio latente.

Analice los resultados desde el punto de vista visual, y cuantitativa del resultado de la clasificación de anomalías.

RÚBRICA

Si el trabajo no se encuentra debidamente ordenado y presentado siguiendo una adecuada estructura para el informe, puede ser considerado como incompleto y cualquiera de las rúbricas se puede ver afectada ver Tabla I. Esto quiere decir que se evaluará el contenido del informe y se verificará contra los notebooks.

TABLE I
RÚBRICA

Criterio	**Puntaje Máx.**
Implementación de Modelo CNN Scratch y Destilado (Pytorch Lightning, Hydra y WandB)	15
Implementación de Autoencoder U-Net (Pytorch Lightning, Hydra y WandB)	10
Diseño experimental, múltiples entrenamientos y variación de hiperparámetros.	15
Comparación de modelos base: reconstruccion de imágenes, progreso de validación y entrenamiento, análisis de overfitting	10
Definición de evaluación de anomalías con embeddings	10
Comparación de mejores modelos de detección de anomalías	10
Comparación entre modelos originales y cuantizados (latencia, tamaño, rendimiento)	10
Comparación de análisis de anomalías con DBSCAN: t-SNE y PCA	10
Calidad de informe científico	15
Total	**105**