Destilado de bases de datos de imágenes hiperespectrales basado en aprendizaje profundo para la solución de problemas de clasificación

Edinsson Fernando Gutiérrez Palomino

Trabajo de grado para optar al titulo de Ingeniero de Sistemas

Director:

Ph.D Henry Arguello Fuentes

Co-director:

Ph.D(c) Román Alejandro Jácome Carrascal

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2024

Dedicatoria

Dedico este trabajo de grado a mis padres: Fernando Gutiérrez Ortega y Blanca María Palomino Palomino, por ser para mí un sustento y un apoyo sufrido e incondicional, lo que hizo que esto fuera posible; también a mi hermano Joan Sebástián Gutiérrez Palomino por estar siempre ahí a mi lado y por último (pero no menos importante) a mi prima, la educadora Eddy Del Rosario Gutiérrez Zapata, por su grande ayuda y apoyo durante mi etapa de formación.

Agradecimientos

Al Señor Jesucristo por su invaluable ayuda, dirección y fortaleza en mi día a día, A mi tutor y co-director el *Ph.D(c)* Román Alejandro Jácome Carrascal por su valiosa guía durante todo el desarrollo del presente trabajo investigativo y al grupo de investigación en diseño de algoritmos y procesamiento de datos multidimensionales High Dimensional Signal Processing (HDSP) por la acogida y la formación que me han brindado durante estos últimos años.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Resumen

Título: Destilado de bases de datos de imágenes hiperespectrales basado en aprendizaje profundo para la solución de problemas de clasificación [*]

Autor: Edinsson Fernando Gutiérrez Palomino [**]

Palabras Clave: Destilación de base de datos, Imágenes hiperespectrales, Clasificación, Aprendizaje profundo

Descripción: El destilado (o condensación) de bases de datos es una técnica utilizada para reducir la dimensionalidad de los conjuntos de datos empleados en el entrenamiento de modelos de aprendizaje profundo, que, según Yu et al. (2024), fue inicialmente propuesta por Wang et al. (2020); pero que sin embargo, a día de hoy aún tiene limitaciones para su aplicación en datos de alta dimensionalidad (Lei and Tao, 2024) en los cuales se incluyen las imágenes hiperespectrales, un tipo de señal que se caracteriza (entre otras cosas) por proveer información a lo largo de una enorme gama de frecuencias del espectro electromagnético a una escala mayor que la de las imágenes multiespectrales, lo que la convierte en la señal de tipo imagen con mayor dimensionalidad; pero que a pesar de ello es ampliamente utilizada para resolver diversos problemas de clasificación de (entre otras cosas) materiales, tipos de suelo, sustancias y enfermedades, a menudo por medio de clasificadores basados en aprendizaje profundo, los cuales también son conocidos por su alto costo de entrenamiento debido a la enorme cantidad de parámetros optimizables que poseen, lo que dificulta aún mas la tarea de entrenamiento y clasificación en proyectos con capacidades de cómputo limitadas. Es debido a lo anterior que este trabajo investigativo presenta la implementación de un algoritmo de destilado basado en el aprendizaje profundo aplicado a imágenes hiperespectrales.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas en informática. Director: Ph.D Henry Arguello Fuentes.

Abstract

Title: Dataset distillation for classification of hyperspectral images datasets. *

Author: Edinsson Fernando Gutiérrez Palomino **

Keywords: Dataset distillation, hyperspectral image, classification, Deep learning.

Description: Dataset distillation (also known as dataset condensation) is a dimensionality reduction technique for training datasets of deep learning models, first proposed by Wang et al. (2020), according to Yu et al. (2024). This technique presents some limitations when applied to high-dimensional data (Lei and Tao, 2024), these include hyperspectral images, a type of signal that provides information across a wider range of frequencies in the electromagnetic spectrum compared to multispectral images, making them the type of image with the greatest dimensionality. Despite this, they are widely used to solve classification problems of materials, soil types, substances, and diseases, often using deep learning models. These models are known for consuming significant computational resources due to their high number of parameters to optimize, making the classification task harder for projects with limited computational resources. To this end, this research implements an algorithm for hyperspectral image classification dataset distillation based on deep learning.

---

* Bachelor Thesis

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Ph.D Henry Arguello Fuentes.

Introducción

Las imágenes hiperespectrales son arreglos de datos constituidos por tres dimensiones: 2 de ellas espaciales y una espectral (Rodríguez-Pulido et al., 2021). Dada una escena en específico, esta última dimensión relaciona la intensidad de luz reflejada en diferentes puntos con determinadas longitudes (o frecuencias) de la onda incidente. Dicho de otra forma, podemos ver las imágenes hiperespectrales cómo un arreglo de firmas espectrales distribuidas espacialmente a lo largo de un área (Bacca et al., 2023), siendo cada firma espectral, por definición, única para cada material y/o sustancia.

De lo anterior se deduce que la función de las imágenes hiperespectrales es permitir de manera simultánea la localización y la identificación de los diferentes materiales y sustancias presentes en una escena determinada, por lo que este tipo de imágenes ha sido ampliamente utilizados para la solución de diversos problemas de clasificación; como lo son por ejemplo el diagnóstico de enfermedades (Bi and Wang, 2019), agricultura de precisión (Rodríguez-Pulido et al., 2021), control de calidad en productos alimenticios (Rodríguez-Pulido et al., 2021), control y evaluación de desastres naturales (Woo et al., 2021), etc; esto a través del entrenamiento de modelos de aprendizaje automático.

Entre este tipo de modelos, los mas populares son aquellos basados en aprendizaje profundo, característicos por demandar una cantidad considerablemente grande de muestras de entrenamiento a fin de alcanzar un buen resultado en etapa de testeo (Lei and Tao, 2024), lo que sumado al enorme volumen de datos que ocupan las imágenes hiperespectrales, resulta en un pro-

ceso de entrenamiento que demanda una cantidad significativa de capacidad de cómputo, tiempo de procesamiento y consumo energético; una situación que afecta principalmente a proyectos con capacidades y presupuesto limitado, pero que además se seguirá dando cada vez más con el paso del tiempo debido al aumento gradual en los datos de entrenamiento (Lei and Tao, 2024) y la presión social a las organizaciones que desarrollan inteligencia artificial para que minimicen su huella de carbono y de agua (Tamburrini, 2022). Debido a esto surge la pregunta ¿cómo lograr un rendimiento aceptable para modelos de clasificación de imágenes hiperespectrales con recursos limitados?

Una manera de hacerle frente a este problema es la reducción del volumen de los datos de entrenamiento, tarea que tradicionalmente se ha realizado por medio de los denominados *métodos de coreset*, los cuales consisten en la creación de un subconjunto de datos de entrenamiento creado a partir del conjunto de entrenamiento original. Cada método de coreset difiere de otros por la forma en la que realizan el proceso de selección de las muestras a utilizar, algunos de ellos son coreset aleatorio y el método de kernel herding: con el primero este proceso de selección se realiza por medio de un muestreo aleatorio uniforme y el segundo busca seleccionar aquellas muestras que minimicen la distancia entre la media del subconjunto seleccionado y la del conjunto completo (Guo et al., 2022). Al utilizarse un conjunto mas pequeño, el entrenamiento es mas eficiente computacionalmente hablando, aunque a cambio requiera sacrificar rendimiento en etapa de testeo debido a la inevitable pérdida de información.

Para mitigar esta pérdida de rendimiento en el modelo, recientemente se ha propuesto el destilado de bases de datos (Lei and Tao, 2024), una alternativa a los tradicionales métodos de

coreset que plantea un escenario totalmente opuesto al que se presenta al momento de entrenar un modelo: en lugar de tener unos pesos variables a aprender y un conjunto de datos fijos, se tiene un modelo con pesos fijos y un conjunto de datos a aprender; en ambos casos buscando obtener un buen resultado en etapa de prueba. Existen técnicas de destilado de datos de entrenamiento basadas en coincidencia de meta-modelos, gradientes, trayectoria y distribución: en el caso de la coincidencia de meta-modelos se busca aprender un conjunto de datos a fin de optimizar la transferibilidad al conjunto real de determinado modelo clasificador entrenado con dichos datos (Sachdeva and McAuley, 2023); en el destilado con coincidencia de gradientes se busca que el gradiente de los datos inducidos por el conjunto aprendido imite el inducido por los datos reales (Lei and Tao, 2024); el destilado con coincidencia de trayectoria define una trayectoria de entrenamiento que depende de los parámetros del modelo clasificador, de modo que lo que se busca con este tipo de destilado es que la trayectoria de entrenamiento de los datos aprendidos coincida lo máximo posible con la trayectoria de los datos reales (Sachdeva and McAuley, 2023) y por último el destilado con coincidencia de distribución define una función embebido que comparte parámetros con el modelo clasificador, de manera que lo que se busca es optimizar la diferencia entre la distribución de la función embebido de los datos reales y aprendidos (Zhao and Bilen, 2023). Cómo es de intuir, con el destilado no se genera un subconjunto de las muestras originales; en su lugar se genera un conjunto nuevo con datos totalmente sintéticos los cuales condensan información relevante de los datos originales (Yu et al., 2024).

A pesar de ser una técnica prometedora y novedosa; existen algunas limitaciones al momento de ser aplicado en datos de alta dimensionalidad (Lei and Tao, 2024). Este hecho, sumado

a lo poco óptimo que resulta el entrenar un clasificador profundo con imágenes hiperespectrales, así como la utilidad práctica de este tipo de datos; motivó al presente proyecto a proponer, diseñar e implementar un método de destilado de conjuntos de datos de imágenes hiperespectrales para el entrenamiento de clasificadores basados en aprendizaje profundo. Para lograrlo se optó por seguir un enfoque basado en coincidencia de distribución debido a su superioridad en términos computacionales con respecto a otros enfoques (Zhao and Bilen, 2023).

La validación del método propuesto se realizó entrenando diferentes modelos clasificadores con los datos aprendidos, el conjunto real completo y con subconjuntos de este obtenidos a través de métodos de coreset con el fin de estimar su rendimiento en etapa de testeo. Al finalizar los experimentos se obtuvo un rendimiento medio superior por parte de los datos sintéticos con respecto a los datos de coreset para la mayoría de los casos de prueba.

1. Objetivos

Objetivo General

Desarrollar un algoritmo basado en aprendizaje profundo para la destilación de conjuntos de datos de imágenes hiperespectrales aplicada a problemas de clasificación

Objetivos específicos

Implementar redes de aprendizaje profundo para la clasificación de imágenes hiperespectrales

Formular matemáticamente la destilación de bases de datos para la tarea de clasificación de imágenes hiperespectrales como un problema de optimización empleando redes de aprendizaje profundo

Desarrollar un algoritmo de destilado de bases de datos de imágenes hiperespectrales para la tarea de clasificación

Evaluar el algoritmo diseñado en distintos escenarios; empleando diferentes bases de datos, modelos de aprendizaje profundo y tipos de clasificación, utilizando métricas como la precisión o el puntaje F1

2. Clasificación de imágenes Espectrales

La tarea de clasificación de imágenes en general puede agruparse en 3 categorías dependiendo del formato del conjunto de datos y del problema que se busca resolver: la primera, conocida cómo clasificación subpíxel, suave o difusa en la que a cada píxel se le asigna más de una etiqueta; la segunda, denominada clasificación por píxel, dura o nítida en la que a cada píxel individual se le asigna una única etiqueta y por último la clasificación por textura, contextual, espacial o superresolución dónde a cada vecindad de píxeles se le asigna una única etiqueta (Md. Palash Uddin and Hossain, 2021). Para este proyecto son de interés la clasificación por píxel y por textura.

Esta tarea de determinar el tipo de clasificación a realizar es de vital importancia ya que influye directamente en la arquitectura del modelo que deseemos emplear para ello, principalmente en su entrada, su salida y en los parámetros que estén conectados a estas. Para el caso en especifico de imágenes hiperespectrales, existen 3 tipos de modelos que son comúnmente usados para tareas de clasificación: están aquellos basados en núcleo, los cuales buscan llevar los datos a un espacio de orden mayor para definir en él una solución lineal al problema de clasificación (Camps-Valls and Bruzzone, 2005); también están los métodos semi supervisados basados en grafos, estos plantean la clasificación como un camino aleatorio en un grafo cuyos vértices son las imágenes hiperespectrales, de maneara que sus aristas deben ser optimizadas para así a partir de un conjunto de datos etiquetado lograr etiquetar otro que no lo está (Camps-Valls et al., 2007) y por último se tienen los métodos basados en aprendizaje profundo (Li et al., 2019), los cuales han tenido una enorme acogida en el ámbito de visión por computador; sin embargo, (tal y cómo profundizaremos

al final de esta sección) son computacionalmente ineficientes al momento de lidiar con enormes volúmenes de datos (Yu et al., 2024), es debido a esto que el presente proyecto se enfocará en dichos modelos.

2.1. Modelos de clasificación de imágenes hiperespectrales basados en aprendizaje profundo

Un modelo de aprendizaje automático $\mathscr{N}_\theta$ está formado a grandes rasgos por una secuencia de capas, las cuales a su vez cuentan con un conjunto asociado de parámetros (o pesos) generalmente optimizables con los cuales se realiza determinada transformación sobre la salida de la capa anterior. la cantidad de parámetros asociados a una capa así cómo la forma en que estos operan depende del tamaño de su entrada y de la naturaleza misma de la capa, pudiendo ser esta convolucional, pooling, flatten y densa.

2.1.1. Capas convolucionales

Sea $\boldsymbol{Z} \in \mathbb{R}^{m \times a \times d}$ un cubo de entrada en el que m y a son dimensiones espaciales, d la cantidad de canales y $\boldsymbol{z}_i$ es el i-ésimo mapa de características de $\boldsymbol{Z}$. Si Γ es la cantidad de filtros en esta capa convolucional y el j-ésimo filtro puede ser caracterizado por los pesos $\boldsymbol{w}_j$ y un sesgo b_j, la j-ésima salida de la capa convolucional se representa de la siguiente manera

$$\tilde{n}_j = \sum_{i=1}^{d} f(\boldsymbol{z}_i * \boldsymbol{w}_j + b_j), j = 1, 2, \ldots, \Gamma \tag{1}$$

dónde * es la operación de convolución y $f(\cdot)$ es una función de activación, esta última se usa para evitar el desvanecimiento del gradiente siendo ReLU la más usada (Li et al., 2019):

$$\sigma(\boldsymbol{z}) = \max(0, \boldsymbol{z}). \tag{2}$$

2.1.2. Capas Pooling

Reducen la información redundante en los mapas de características, lo que resulta en un modelo con menos parámetros (y por lo tanto mas ligero) y en una mayor abstracción de las características extraídas.

Supongamos que se tiene una vecindad en un mapa de características de tamaño de ventana de $p \times p$ denominada $\boldsymbol{S}$, el trabajo de la capa pooling es mapear $\boldsymbol{S}$ a un valor escalar. Entre los tipos de capas pooling mas conocidas están *max Pooling*, la cual mapea al valor máximo de $\boldsymbol{S}$ y *average pooling*, que mapea al valor promedio (Li et al., 2019).

2.1.3. Capas densas

Al final de las capas pooling y convolucionales, las cuales extraen principalmente propiedades espaciales de la imagen, el mapa de características resultante se suele aplanar (capa flatten) para convertirlo en un vector y así poder pasarlo a una serie de capas densas las cuales extraerán características aún mas profundas y abstractas para finalmente realizar la clasificación. Estas capas

pueden ser definidas de la siguiente manera (Li et al., 2019):

$$\tilde{\boldsymbol{N}}' = \sum_{i=1}^{\lambda} f(\boldsymbol{W}\boldsymbol{Z}' + b); \tag{3}$$

donde $\boldsymbol{W}$ son los pesos de la capa, $\boldsymbol{b}$ su sesgo y $\boldsymbol{Z}'$ es el mapa de características de entrada.

2.1.4. Entrenamiento de modelos para la clasificación

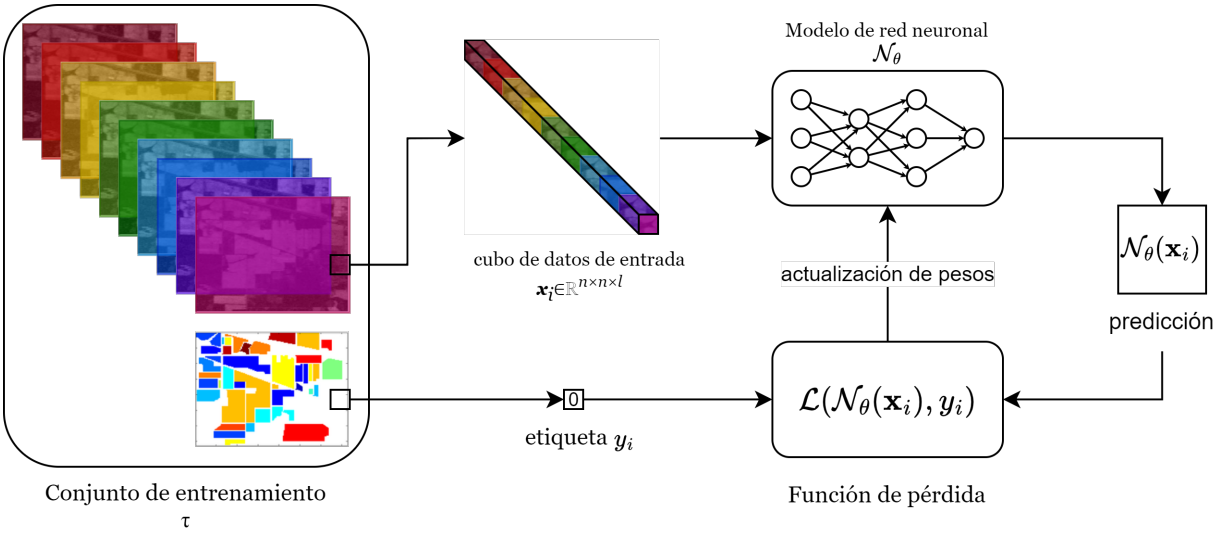
La arquitectura de determinado modelo clasificador profundo $\mathscr{N}_\theta$ está determinada por la secuencia de capas que lo conforman, pudiendo incluso conectarse capas de diferente naturaleza; algunas de las arquitecturas mas conocidas son de tipo auto-codificadores apilados, máquinas del Boltzman apiladas, red neuronal convolucional, red neuronal recurrente, redes generativas adversariales, transformadores, entre otros (Li et al., 2019). Una vez definida la arquitectura de $\mathscr{N}_\theta$, lo que sigue es realizar el entrenamiento de dicho modelo a fin de que los valores de sus parámetros se ajusten a la tarea de clasificación requerida, para ello se requiere mínimamente de 2 elementos: un conjunto de datos de entrenamiento $\tau = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_{|\tau|}, y_{|\tau|})\}$ que conste de $|\tau|$ pares imagen-etiqueta (con $\boldsymbol{x} \in \mathbb{R}^{n \times n \times l}$, y $y_i \in \{0, \ldots, c-1\}$ siendo n el ancho y alto de la imagen, l la cantidad de bandas espectrales y c la cantidad de clases) y una función de costo $\mathscr{L}_{\text{loss}}(\mathscr{N}_\theta(\boldsymbol{x}_i), y_i)$ coherente con el problema a tratar y que sea diferenciable con respecto a los pesos del modelo en cada entrada, es decir $\exists \nabla_\theta \mathscr{L}_{\text{loss}}(\mathscr{N}_\theta(\boldsymbol{x}_i), y_i) \forall i \in \{1, \ldots, |\tau|\}$. El objetivo del entrenamiento es encontrar un conjunto de parámetros θ (pesos) con los cuales se minimice la función de costo:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=0}^{|\tau|} \mathscr{L}_{\text{loss}}(\mathscr{N}_\theta(\boldsymbol{x}_i), y_i); \tag{4}$$

este proceso puede observarse gráficamente en la figura 1.

Figura 1

Proceso de clasificación con un método basado en aprendizaje profundo



*N*ota. Esta figura ilustra el proceso de clasificación por píxel (caso $n = 1$) y espacial (caso $n \in \mathbb{N} : n > 1$) de una imagen hiper espectral de l bandas por medio de un modelo de aprendizaje automático $\mathcal{N}_\theta$ el cual estima una etiqueta $\mathcal{N}_\theta(\boldsymbol{x}_i)$. Las imágenes utilizadas cómo referencia fueron tomadas del conjunto de datos Indian Pines (Graña et al., 2021). La etiqueta y ubicación del fragmento del conjunto de datos del cual se extrae el cubo de entrada es meramente ilustrativa.

2.1.5. Problemas de rendimiento de los métodos basados en aprendizaje profundo

A pesar de ser bastante usados, los clasificadores profundos no son conocidos por ofrecer un entrenamiento óptimo. Para ilustrar este hecho analicemos cuantas actualizaciones ρ se deben realizar al entrenar un modelo $\mathcal{N}_\theta$ de $|\theta|$ parámetros utilizando el ya mencionado conjunto de entrenamiento τ dividido en $|\tau|/|B^\tau|$ lotes, con tamaño de lote $|B^\tau| \in (0, |\tau|] \cap \mathbb{N}$:

$$\rho = \alpha |\theta| \frac{|\tau|}{|B^\tau|}; \tag{5}$$

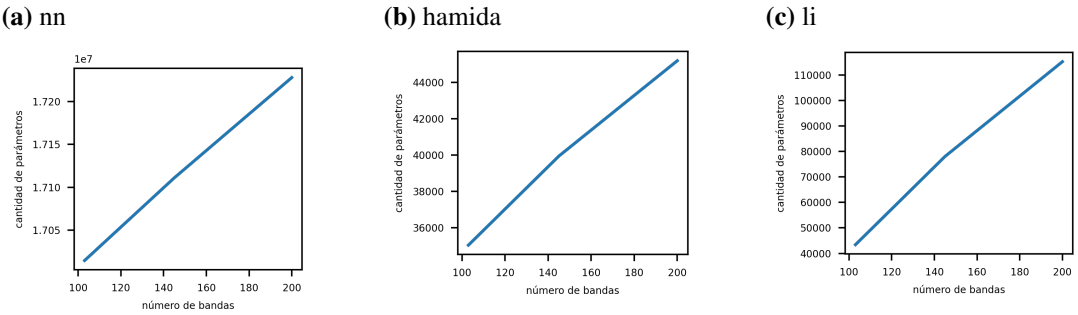
dónde α es la cantidad de épocas de entrenamiento.

En base a la ecuación 5 se deduce que la cantidad de actualizaciones es proporcional al número de parámetros del modelo, una cantidad que puede llegar a ser extremadamente alta, para ejemplificarlo primero hay que tomar en cuenta que al momento de inicializar un modelo la cantidad de parámetros optimizables de la capa de entrada (y por lo tanto también la cantidad total de parámetros del modelo) dependen del tamaño del cubo de datos de entrada $\boldsymbol{x}_i \in \mathbb{R}^{n\times n\times l}$ (ver figura 1). Ya habiendo aclarado esto, se tomaron del estado del arte 3 modelos de aprendizaje profundo como muestra, los cuales son nn, hamida y li (figuras 3, 4 y 5 respectivamente) con el fin de registrar la cantidad de parámetros optimizables que necesitan para ser entrenados con imágenes de 103, 145 y 200 bandas espectrales (la misma cantidad de bandas que poseen las imágenes de las bases de datos mencionadas en la sección 5.2); este registro se llevó a cabo manteniendo fijo el tamaño de parche n en cada modelo, siendo $n = 1$ para nn y $n = 5$ para hamida y li, por lo que para un mismo modelo solo se varió la cantidad de canales espectrales de entrada. Los registros se pueden observar en la figura 2, de manera que, por ejemplo, en la figura 2a se graficó la cantidad de parámetros a inicializar con nn manteniendo tamaños de entrada de $1 \times 1 \times 103$, $1 \times 1 \times 145$ y $1 \times 1 \times 200$ (variando únicamente la cantidad de bandas y no el alto ni ancho de la entrada); de manera similar los tamaños de entrada a considerar en la figura 2b (al igual que en la 2c) fueron $5 \times 5 \times 103$, $5 \times 5 \times 145$ y $5 \times 5 \times 200$.

En la figura 2 se puede observar como la cantidad de parámetros en los modelos hamida y li se cuentan por miles mientras que para el caso del modelo nn se cuentan por decenas de millones; además, esta cantidad presenta una clara tendencia a aumentar a medida que también lo hace la

Figura 2

Relación entre la cantidad de bandas y de parámetros a optimizar en un modelo



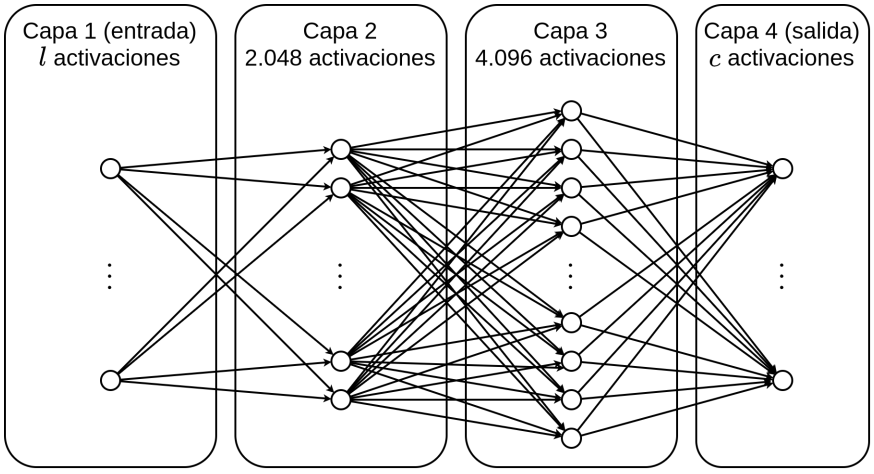
cantidad de bandas espectrales del modelo, haciendo que la necesidad de aprovechar mejor los recursos computacionales sea mas apremiante en imágenes hiperespectrales que en datos similares como lo son las imágenes multiespectrales, RGB y en escala de grises.

Si bien es posible reducir la cantidad de parámetros del modelo para aumentar la eficiencia computacional del entrenamiento, ya sea modificando sus capas ocultas o disminuyendo el número de bandas, por ejemplo reduciendo las imágenes a sus componentes principales (Md. Palash Uddin and Hossain, 2021); así como se podría jugar con otras variables cómo el tamaño del lote o la cantidad de épocas; para efectos de este proyecto es de especial relevancia el tamaño del conjunto de datos $|\tau|$.

Una reducción considerable del conjunto de entrenamiento tiene un riesgo potencial de generar un sobre ajuste en los pesos del modelo al momento del entrenamiento, este riesgo ha sido documentado como ≪la maldición de la dimensionalidad≫ o fenómeno de Hughes (Camps-Valls and Bruzzone, 2005); otra desventaja es la pérdida de información, lo que conlleva a una disminución del rendimiento del modelo en etapa de testeo; aunque, por otro lado dicha reducción tiene

Figura 3

Modelo nn



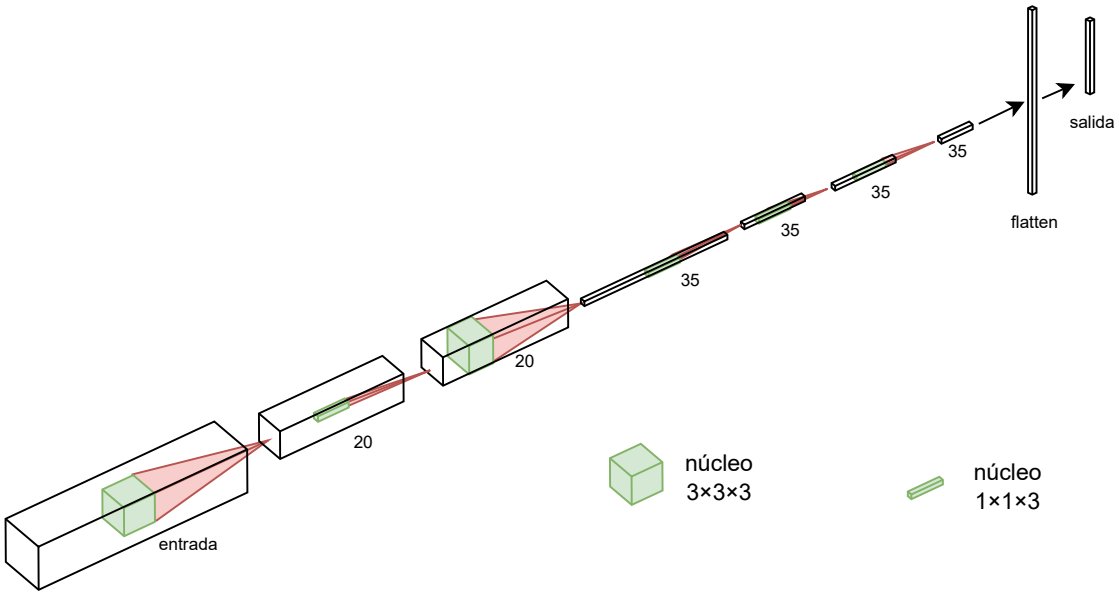
*N*ota. Se ilustra la arquitectura del modelo nn definido por Audebert et al. (2019), aquí c es la cantidad de clases y l el número de bandas de entrada.

beneficios en términos no solo de rendimiento computacional sino también en términos de almacenamiento y de rapidez de transmisión; por lo que la elección queda a criterio de las capacidades, y necesidades de cada proyecto en particular en el que se vaya a realizar la clasificación.

Tradicionalmente esta reducción se ha realizado muestreando un determinado número de muestras del conjunto original por medio de alguna técnica de selección de muestras de entrenamiento o método de coreset (Guo et al., 2022); sin embargo hace algunos años surgió una novedosa técnica llamada «destilado de bases de datos» la cual nos arroja una perspectiva diferente para llevar a cabo esta tarea (Lei and Tao, 2024).

Figura 4

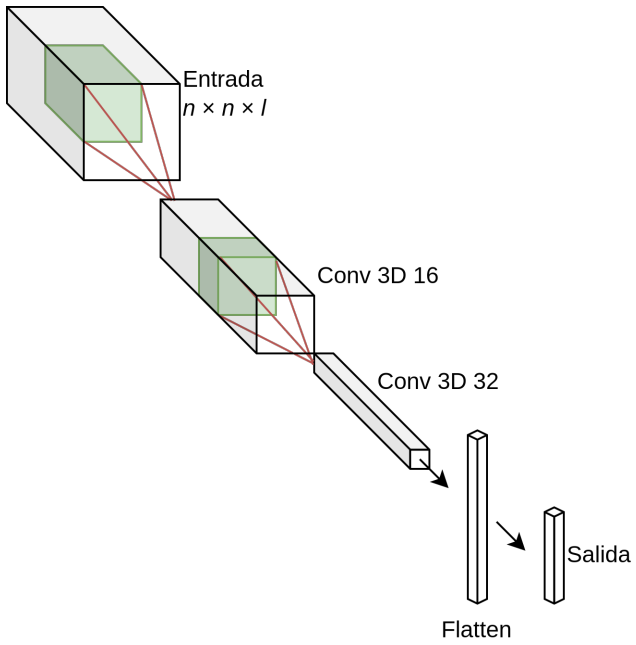
Modelo hamida



*N*ota. Se ilustra la arquitectura del modelo hamida el cual fue propuesto originalmente por Ben Hamida et al. (2018); este modelo está compuesto por una capa de entrada seguida de 6 capas convolucionales 3D con activación ReLU, una capa de aplanamiento y una capa densa de salida. El número debajo de cada capa indica la cantidad de mapas de características que esta posee.

Figura 5

Modelo li



*N*ota. Se ilustra la arquitectura del modelo li el cual fue propuesto originalmente por Li et al. (2017), dónde CONV 3D 16 y CONV 3D 32 indica una salida cuya primera dimensión es de tamaño 16 y 32 respectivamente. Este modelo consta de dos capas convolucionales 3D con activación ReLU, la primera de ellas cuenta con un núcleo de tamaño $3 \times 3 \times 7$ y la segunda de $3 \times 3 \times 3$.

3. Destilado de bases de datos

Se quiere condensar un conjunto de datos de entrenamiento $\tau = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{|\tau|}, y_{|\tau|})\}$ de $|\tau|$ pares imagen-etiqueta en un pequeño conjunto sintético $S = \{(\boldsymbol{s}_1, y_1), \ldots, (\boldsymbol{s}_{|s|}, y_{|s|})\}$ de tamaño $|S| \ll |\tau|$ de tal forma que el rendimiento en etapa de testeo que se obtiene al entrenar cierto modelo $\mathscr{N}_\theta$ con los datos S sea comparable con el rendimiento obtenido al entrenar con τ (figura 6), es decir

$$\mathbb{E}_{x\sim P_{\mathscr{D}}}\left[\ell\left(\mathscr{N}_{\theta^\tau}(x), y\right)\right] \simeq \mathbb{E}_{x\sim P_{\mathscr{D}}}\left[\ell\left(\mathscr{N}_{\theta^S}(x), y\right)\right] \tag{6}$$

Dónde $P_{\mathscr{D}}$ es la distribución de los datos reales, ℓ es la función de pérdida, $\mathbb{E}$ representa el valor esperado y θ^τ θ^S son los parámetros de $\mathscr{N}_\theta$ obtenidos al entrenar con τ y S respectivamente (Zhao and Bilen, 2023).

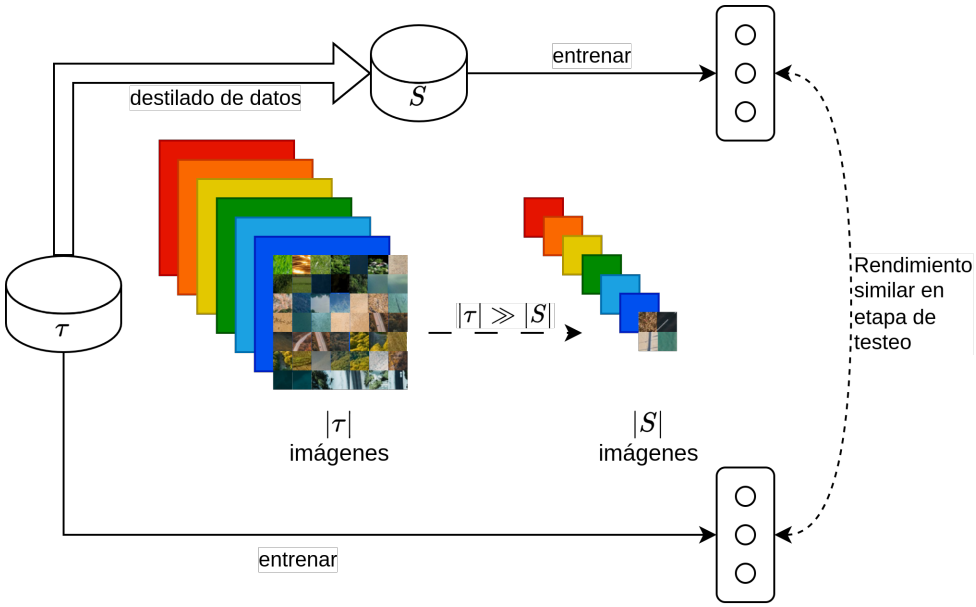
La ecuación 6 puede ser entendida como un problema de aprender a aprender en el que los parámetros θ^S son una función dependiente de los datos sintéticos S, de manera que podemos darle solución minimizando cierta función de pérdida relativa al problema del destilado $\mathscr{L}^\tau$ a través de los datos originales τ, por ejemplo, la ecuación 7 muestra como sería dicha función en un enfoque de doble optimización(Zhao and Bilen, 2023):

$$\begin{aligned} S^* &= \arg\min_S \mathscr{L}^\tau\left(\theta^S(S)\right) \\ \text{subject to } \theta^S(S) &= \arg\min_\theta \mathscr{L}^S(\theta). \end{aligned} \tag{7}$$

En la actualidad existen diferentes métodos de destilado de conjuntos de datos de entrena-

Figura 6

Marco de trabajo del destilado de bases de datos de clasificación de imágenes hiperespectrales



*N*ota. adaptado de Yu et al. (2024). Las imágenes utilizadas para representar los diferentes conjuntos de datos son meramente ilustrativas.

miento, algunos de los cuales se muestran a continuación:

3.1. Destilado con coincidencia de gradientes

Es un tipo de destilado con doble optimización que busca minimizar la diferencia entre los gradientes a través de S de la función de pérdida $\mathscr{L}$ con respecto a los parámetros θ del modelo $\mathscr{N}_\theta$ y sus equivalentes a través de τ, esto mientras los parámetros θ son actualizados:

$$S^* = \arg\min_S \mathrm{E}_{\theta_0 \sim P_{\theta_0}} \left[\sum_{t=0}^{T-1} D\left(\nabla_\theta \mathscr{L}^S(\theta_t), \nabla_\theta \mathscr{L}^\tau(\theta_t) \right) \right]$$
$$\text{sujeto a} \quad \theta_{t+1} \leftarrow \text{opt - alg}_\theta \left(\mathscr{L}^S(\theta_t), \varsigma_\theta, \eta_\theta \right); \tag{8}$$

dónde P_{θ_0} es la distribución de los parámetros de inicialización, $T \in \mathscr{N}^{++}$ es la cantidad de veces que debemos actualizar S en un solo ciclo del algoritmo, $\boldsymbol{\varsigma_\theta}$ es la cantidad de épocas en las que se entrenarán los pesos θ del modelo $\mathscr{N}_\theta$ en cada ciclo del algoritmo, $\boldsymbol{\eta_\theta}$ es la tasa de aprendizaje, opt - alg es un procedimiento de optimización especifico con ς pasos y $D(\cdot,\cdot)$ mide el error de la coincidencia del gradiente.(Zhao and Bilen, 2023)

3.2. Destilado con poda de parámetros

Este método inicia con el pre-entrenamiento de N modelos profesores con el conjunto τ durante cierta cantidad de épocas guardando los pesos θ en cada una de ellas, estos serán los pesos profesores, posteriormente, en cada paso de destilado k se pueden identificar las siguientes etapas: en la primera, conocida cómo entrenamiento de arquitectura profesor-estudiante, se muestrea un número de época i junto con sus respectivos pesos profesores θ_i para inicializar el modelo y posteriormente entrenar el conjunto de datos S por J épocas, los parámetros $\tilde{\theta}_{i,J}$ obtenidos de este entrenamiento serán conocidos cómo parámetros estudiantes; en la segunda etapa, conocida cómo coincidencia de parámetros profesor-estudiante, dada una cantidad k de iteraciones para entrenar los parámetros profesores, se realiza la vectorización de los parámetros $\tilde{\theta}_{i+J}$ y θ_{i+k} para posteriormente hacer una evaluación elemento a elemento de la similitud $\theta^z_{i+k}/\tilde{\theta}^h_{i,j}$ siendo h un índice, si esa similitud es menor que cierto umbral ε, entonces estos parámetros se marcan cómo difíciles de aprender y se procede a realizar la poda de los mismos según las siguientes expresiones:

$$\tilde{\theta}'_{i,J} = [\tilde{\theta}^1_{i,J}, \tilde{\theta}^2_{i,J}, \ldots, \tilde{\theta}^u_{i,J}], \tag{9}$$

$$\theta'_{i+k} = [\theta^1_{i+k}, \theta^2_{i+k}, \ldots, \theta^u_{i+k}] \tag{10}$$

y

$$\theta'_i = [\theta^1_i, \theta^2_i, \ldots, \theta^u_i] \tag{11}$$

dónde u representa el número de parámetros efectivos restantes; con estos parámetros podemos calcular la función de pérdida correspondiente a este método:

$$\mathscr{L} = \frac{\left\| \tilde{\theta}'_{i,J} - \theta'_{i+k} \right\|_2^2}{\left\| \theta'_i - \theta'_{i+k} \right\|_2^2}; \tag{12}$$

finalmente, en la etapa de la generación de datos destilados optimizados se hace la actualización del conjunto de datos S tomando en cuenta la función de pérdida calculada (Li et al., 2023).

3.3. Destilado con coincidencia de distribución

Tanto el destilado con coincidencia de gradientes cómo el destilado con poda de parámetros tienen algo en común: una costosísima doble optimización, esto debido a que tanto los datos sintéticos S cómo los parámetros θ son optimizados en algún momento del entrenamiento. Una alternativa óptima al enfoque planteado en 7 fue propuesta por Zhao and Bilen (2023) y consiste en lo siguiente: sea $\boldsymbol{x}_i \in \mathbb{R}^d$ con $i \in \mathbb{N} \cap [1, |\tau|]$ una imagen de entrada que puede mapearse hacia un espacio de dimensión inferior por medio de un conjunto de funciones paramétricas $\psi_\vartheta : \Re^d \to \Re^{d'}$ (con $d' \ll d$ y siendo ϑ el un conjunto de parámetros) y además sea $\mathscr{A}(\cdot, \omega) : \omega \sim \Omega$ una función de aumento de datos con parámetros de aumento aleatorios ω (los cuales siguen una distribución de probabilidad Ω) la cual fue insertada para añadir diversidad al problema y así mitigar el riesgo

de sobreajuste y el desbalanceo de clases; entonces definimos la media de la distribución que sigue la función embebido de los datos τ y S respectivamente cómo

$$\mu_\tau = \frac{1}{|\tau|} \sum_{i=1}^{|\tau|} \psi_\vartheta \left(\mathscr{A} \left(\boldsymbol{x}_i, \boldsymbol{\omega} \right) \right) \tag{13}$$

y

$$\mu_S = \frac{1}{|S|} \sum_{i=1}^{|S|} \psi_\vartheta \left(\mathscr{A} \left(\boldsymbol{s}_i, \boldsymbol{\omega} \right) \right); \tag{14}$$

a partir de las cuales se define la función de pérdida del algoritmo de destilado cómo la discrepancia entre ambas distribuciones, o dicho de otra forma:

$$\arg\min_S \mathbb{E}_{\vartheta \sim P_\vartheta} \| \mu_\tau - \mu_S \|^2. \tag{15}$$

Esta optimización se debe realizar clase por clase y (cómo es de notar) en ningún momento de la misma se optimizan los parámetros ϑ de la función embebido, sino que más bien se muestrean de acuerdo a una distribución de probabilidad P_ϑ, por lo cual este método es más eficiente que aquellos basados en doble optimización; sin embargo, hasta ahora no había sido utilizado en imágenes hiperespectrales.

4. Método de destilado propuesto

Debido a la enorme ventaja que tiene el destilado con coincidencia de distribución sobre los otros dos métodos mencionados, este proyecto seguirá dicho enfoque para realizar la tarea del destilado de imágenes hiperespectrales. En este punto es importante denotar, basado en las ecuaciones 13 y 14, la necesidad de definir una función de aumento coherente con nuestro problema.

4.1. Función de aumento de datos

Esta función de aumento $\mathscr{A}(\cdot, \boldsymbol{\omega})$ se definió de tal forma que reciba un vector de parámetros aleatorios $\boldsymbol{\omega} \in \mathbb{R}^r \sim U(0{,}9\ ,\ 1{,}1)$ (dónde r es la cantidad de muestras que se desea generar) y su salida será un arreglo de tamaño $r+1$ cuyo primer elemento será la imagen original y el de ellos serán productos escalares de la misma, de esta forma la función de aumento para el conjunto de datos reales sería

$$\mathscr{A}(\boldsymbol{x}_i, \boldsymbol{\omega}) = [\boldsymbol{x}_i, \boldsymbol{\omega}_1 \boldsymbol{x}_i, \boldsymbol{\omega}_2 \boldsymbol{x}_i, \ldots, \boldsymbol{\omega}_r \boldsymbol{x}_i] \tag{16}$$

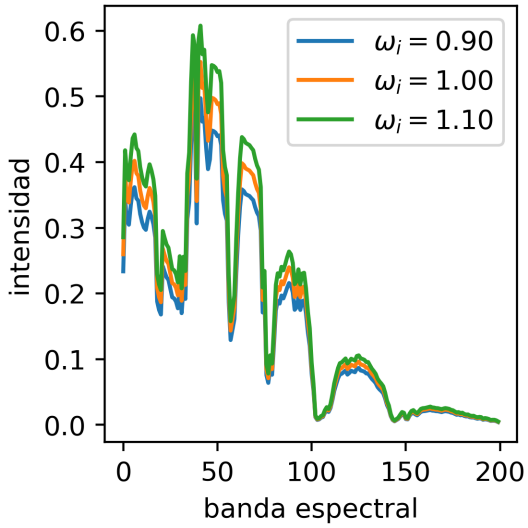
y para los datos destilados

$$\mathscr{A}(\boldsymbol{s}_i, \boldsymbol{\omega}) = [\boldsymbol{s}_i, \boldsymbol{\omega}_1 \boldsymbol{s}_i, \boldsymbol{\omega}_2 \boldsymbol{s}_i, \ldots, \boldsymbol{\omega}_r \boldsymbol{s}_i]; \tag{17}$$

esta transformación tiene un efecto de escalado en la firma espectral (figura 7) que no modifica su comportamiento de forma brusca.

Figura 7

Aumento de datos



*N*ota. Se tomó como muestra una firma real $\boldsymbol{x}_i$ seleccionada aleatoriamente del conjunto Indian Pines (Graña et al., 2021). La firma en el caso $\boldsymbol{\omega}_i = 1$ corresponde a la original.

4.2. Algoritmo de destilado propuesto

Ahora que hemos definido la función de aumento, es momento de echar un vistazo al problema de optimización definido en la ecuación 15. Lo primero que hay que denotar al respecto es que la media no es el único estadístico que describe una distribución, por lo que sería bueno incluir más características que describan aún mejor diferentes comportamientos propios de los datos, por ejemplo, existe un fenómeno que es considerado como una enorme fuente de error en el procesamiento de imágenes multiespectrales e hiperespectrales: la variabilidad espectral, es decir, todas aquellas variaciones existentes en cuanto a forma y escala a nivel espectral para diferentes píxeles de una misma clase debido a diferencias en la iluminación, ruido atmosférico, no uniformidad del

material, etc. (Ariolfo Camacho and Arguello, 2022).

Tomando en cuenta estas consideraciones, proponemos que para una representación aún mas fiel a los datos originales sería útil tomar en cuenta también la desviación estándar al momento de la optimización, esto sabiendo además que el método de Zhao and Bilen (2023) fue probado en imágenes RBG; sin embargo, las imágenes hiperespectrales tienen un tamaño considerablemente mas grande y por lo tanto un número de grados de libertad mucho mayor.

En base a los cambios anteriormente descritos, definimos la desviación estándar del espacio embebido cómo

$$\sigma_\tau = \sqrt{\frac{1}{|\tau|}\sum_{i=1}^{|\tau|}[\psi_\vartheta(\mathscr{A}(\boldsymbol{x}_i,\boldsymbol{\omega})) - \mu_\tau]^2} \tag{18}$$

para los datos reales y

$$\sigma_S = \sqrt{\frac{1}{|S|}\sum_{i=1}^{|S|}[\psi_\vartheta(\mathscr{A}(\boldsymbol{s}_i,\boldsymbol{\omega})) - \mu_S]^2} \tag{19}$$

para los sintéticos, los exponentes en ambas ecuaciones implican una operación de potencia elemento a elemento. Esto nos lleva al siguiente problema de optimización:

$$\min_S \mathbb{E}_{\vartheta\sim P_\vartheta}\left(\|\mu_\tau - \mu_S\|^2 + \|\sigma_\tau - \sigma_S\|^2\right). \tag{20}$$

La solución a la ecuación 20 se halla de manera similar al del método propuesto por Zhao and Bilen (2023): optimizando por K iteraciones y clase por clase por medio de un algoritmo basado en el descenso del gradiente estocástico (en este caso se utilizó gradiente descendiente con una tasa de aprendizaje η), este proceso se puede visualizar en la figura 8 y además está implementado en

el algoritmo 1.

Algorithm 1 Algoritmo propuesto para el destilado de imágenes hiperespectrales

Require: Una inicialización de S, modelo de red neuronal profunda $\mathscr{N}_\theta$ con parámetros θ que contiene el espacio embebido ψ_ϑ con parámetros ϑ, distribución de probabilidad P_ϑ, función de aumento diferenciable $\mathscr{A}_\omega$ con parámetros $\omega \sim \Omega$, cantidad de iteraciones K, tasa de aprendizaje η y un conjunto real de pares imagen hiper espectral-etiqueta τ

for $k = 0, \ldots, K-1$ **do**
  Muestrea $\vartheta \sim P_\vartheta$
  inicializar $\mathscr{L} = 0$
  for $c = 0, \ldots, C-1$ **do**
    Muestrea los pares de mini lotes $B_c^S \sim S$ $B_c^\tau \sim \tau$ y parámetros de aumento $\omega_c \sim \Omega$ para todas las clases c
    Calcular $\mu_{B_c^\tau} = \sum_{\{\boldsymbol{x},y\}\in B_c^\tau} \psi_\vartheta(\mathscr{A}(\boldsymbol{x},\omega))/|B_c^\tau|$
    Calcular $\mu_{B_c^S} = \sum_{\{\boldsymbol{s},y\}\in B_c^S} \psi_\vartheta(\mathscr{A}(\boldsymbol{s},\omega))/|B_c^S|$
    Calcular $\sigma_{B_c^\tau} = \sqrt{\sum_{\{\boldsymbol{x},y\}\in B_c^\tau} [\psi_\vartheta(\mathscr{A}(\boldsymbol{x},\omega)) - \mu_{B_c^\tau}]^2/|B_c^\tau|}$
    Calcular $\sigma_{B_c^S} = \sqrt{\sum_{\{\boldsymbol{s},y\}\in B_c^\tau} [\psi_\vartheta(\mathscr{A}(\boldsymbol{s},\omega)) - \mu_{B_c^S}]^2/|B_c^S|}$
    Computar $\mathscr{L} \leftarrow \mathscr{L} + \left\|\mu_{B_c^\tau} - \mu_{B_c^\tau}\right\|^2 + \left\|\sigma_{B_c^\tau} - \sigma_{B_c^S}\right\|^2$
  end for
  Actualizar $S \leftarrow S - \eta \nabla_S \mathscr{L}$
end for

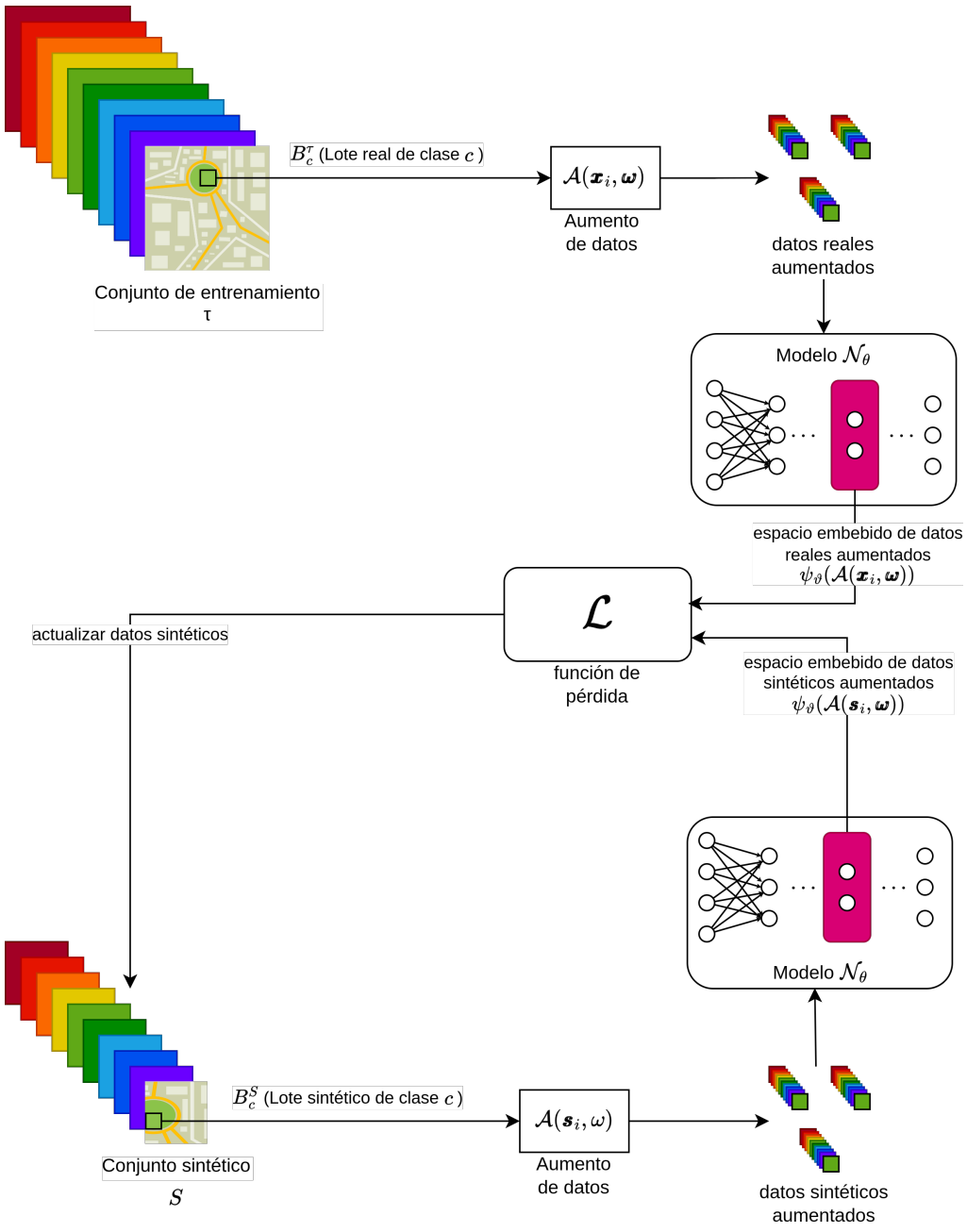
Este algoritmo recibe cómo parte de su entrada el conjunto de datos sintéticos inicializado, dicha inicialización generalmente se puede realizar de 2 maneras diferentes: inicializar el conjunto con ruido o con muestras del conjunto de datos real (inicialización por medio de un conjunto de coreset). Sea cual sea el camino que optemos para inicializarlo, el conjunto sintético deberá seguir el siguiente formato:

$$S = \{(\boldsymbol{s}_1, 0), (\boldsymbol{s}_2, 0), \ldots, (\boldsymbol{s}_v, 0), (\boldsymbol{s}_{v+1}, 1), (\boldsymbol{s}_{v+2}, 1), \ldots, (\boldsymbol{s}_{cv-1}, c), (\boldsymbol{s}_{cv}, c)\}; \tag{21}$$

dónde c es la cantidad de clases del conjunto de datos τ y v es la cantidad de muestras por clase que

Figura 8

Destilado de imágenes hiperespectrales con coincidencia de distribución



*N*ota. En esta figura se muestra el proceso de actualización de los datos sintéticos para cada clase, en cada iteración del algoritmo de destilado se debe hacer este procedimiento para todas las c clases del conjunto de datos.

tendrá el conjunto sintético, de manera que el tamaño del conjunto sintético $|S|$ se puede calcular de la siguiente manera:

$$|S| = cv. \tag{22}$$

Adicionalmente se necesita definir la función embebido ψ_ϑ, que en efectos prácticos es la salida de alguna capa intermedia de N_θ, de manera que el conjunto de parámetros ϑ incluirá únicamente a los parámetros de dicha capa y las anteriores.

Por último, la codificación completa del algoritmo propuesto puede ser consultada en el siguiente enlace: `https://github.com/Edinsson-G/destilado.git`.

5. Resultados

5.1. Configuración en general del método

Se puso en práctica el método propuesto utilizando los modelos nn, hamida y li (figuras 3, 4 y 5 respectivamente) tomando la salida de la última capa oculta de cada modelo (excluyendo las capas flatten en el caso de hamida y li) como espacio embebido $\psi_\vartheta \sim P_\vartheta$, esto debido a que si escogiéramos alguna capa anterior a ella, solo obtendríamos una representación parcial de los datos, mientras que al escoger dicha capa estamos obteniendo una representación completa (Zhao and Bilen, 2023). En cuanto a la distribución P_ϑ con la cual se muestrearán los parámetros de dicha función, esta depende del tipo de capa a la cual pertenece cada parámetro $\vartheta_j \in \vartheta$:

$$P_\vartheta = \begin{cases} U\left(-\sqrt[-2]{\gamma\prod_{i=0}^{2}\xi_i},\ \sqrt[-2]{\gamma\prod_{i=0}^{2}\xi_i}\right), & \text{si } \vartheta_j \text{ pertenece a una capa de convolución 3D} \\ U(-\sqrt[-2]{\beta}, -\sqrt[-2]{\beta}), & \text{si } \vartheta_j \text{ pertenece a una capa densa} \end{cases} \tag{23}$$

Siendo γ la cantidad de filtros de entrada, ξ_i la cantidad de elementos que tiene el núcleo convolucional en su i-ésima dimensión (para el caso de las capas convolucionales) y β la cantidad de entradas para en el caso de las capas densas.

Adicionalmente a la obtención de la función embebido, los modelos de aprendizaje profundo también se utilizaron para realizar validaciones y pruebas que permitieron hacer un seguimiento a la calidad de los datos sintéticos, para ello fue necesario dividir cada base de datos en 3 partes diferentes correspondientes al 64 %, 16 % y 20 % de las muestras a fin de destinarlas para tareas

de entrenamiento, validación y testeo respectivamente (de manera que τ solo corresponderá a esa parte que posee el 64 % del total de muestras).

A este conjunto de datos τ se le aplicó el algoritmo 1 con un tamaño de aumento $r = 1$ y un máximo de $K = 1{,}000$ iteraciones con un criterio de parada temprana de 300 iteraciones consecutivas sin tener una disminución en la función de costo y conservando los datos con los que se obtuvo la menor pérdida, además se utilizó un optimizador de descenso de gradiente estocástico con un momento de 0.5 y una paciencia de 100 iteraciones consecutivas sin ninguna disminución en la función de costo antes de disminuir su tasa de aprendizaje η a la décima parte de su valor.

Ya habiendo explicado la división de los datos para entrenamiento, es momento de hablar de los datos de validación. El propósito de estos es realizar un seguimiento al progreso del algoritmo a lo largo de las iteraciones, este seguimiento se realiza por medio de un entrenamiento del modelo $\mathscr{N}_\vartheta$ con una inicialización de pesos fija cada 10 iteraciones utilizando los datos sintéticos, al finalizar dicho entrenamiento se le pasan los datos de validación al modelo entrenado (a modo de lo que serían los datos de testeo en un entrenamiento tradicional de redes neuronales) para tener un estimado del progreso que se obtiene a lo largo del destilado. Este entrenamiento no corresponde a una doble optimización puesto que no afecta para nada el resultado del algoritmo 1 además de ser opcional su realización puesto que es de carácter meramente informativo.

Por último, tenemos los datos de testeo, estos se usan con el fin de medir la calidad de los datos sintetizados de la siguiente manera: al finalizar la ejecución del algoritmo 1 se usan los datos sintéticos S para entrenar 20 veces el modelo $\mathscr{N}_\vartheta$ con 20 inicializaciones aleatorias diferentes, y después de cada entrenamiento se le pasan los datos de testeo al modelo a fin de estimar el valor

de accuracy-1 alcanzado en cada entrenamiento individual y en últimas registrar su valor medio y error estándar. Este valor medio se compara con el obtenido tras realizar la misma cantidad de entrenamientos con τ y subconjuntos de este obtenidos al aplicar los métodos de coreset kernel herding y selección aleatoria clase por clase (puesto que el destilado también se realiza clase por clase).

Todo este procedimiento se aplicó con diferentes cantidades de muestras por clase v para cada par modelo-base de datos, aunque los valores exactos de muestras por clase seleccionados varían según cada caso para lograr un mejor análisis de la tendencia del accuracy-1 en etapa de testeo.

5.2. Bases de datos empleadas

5.2.1. Indian Pines

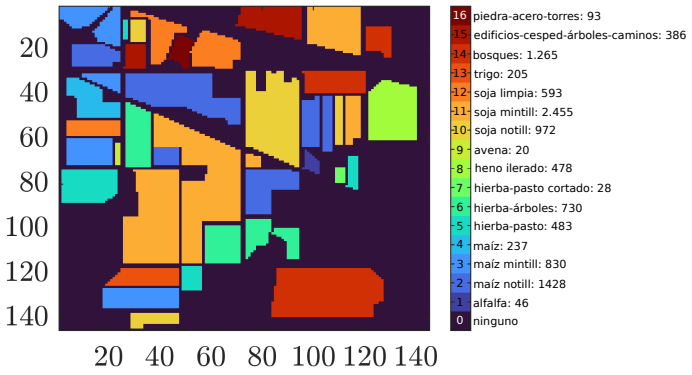
Tal y como se muestra en la figura 9b, este conjunto cuenta con imágenes hiperespectrales de 200 bandas clasificadas en 17 clases diferentes, de las cuales se omitió la clase número 0 (ninguno) y el resto de clases se re-etiquetaron restándole 1 a su valor para así quedar con un total de 16 clases etiquetadas entre 0 y 15 (lo mismo aplicó para las otras bases de datos). En la figura 9a se muestra una vista previa del cubo de datos completo en su versión corregida, del cual se obtuvieron un total de 6.512 firmas espectrales de entrenamiento para el modelo nn y 6.400 parches de $5 \times 5 \times 200$ para hamida y li los cuales pueden solaparse entre sí (lo mismo aplica para las otras bases de datos) (Graña et al., 2021).

Figura 9

conjunto Indian Pines

(a) una banda espectral

(b) máscara de etiquetas



*N*ota. la figura 9a fue tomada de Graña et al. (2021) mientras que la figura 9b fue adaptada de la misma fuente.

5.2.2. Pavia U

Con la versión corregida de este conjunto contamos con 27.275 firmas para entrenar el modelo nn y 26.670 parches de $5 \times 5 \times 103$ para hamida y li. Cada entrada cuenta con un total de 103 bandas espectrales y está clasificada en una de las 10 clases que se muestran en la figura 10b de las cuales se ignorará la clase de código 0. En la figura 10 se puede observar una vista previa del conjunto de datos (Graña et al., 2021).

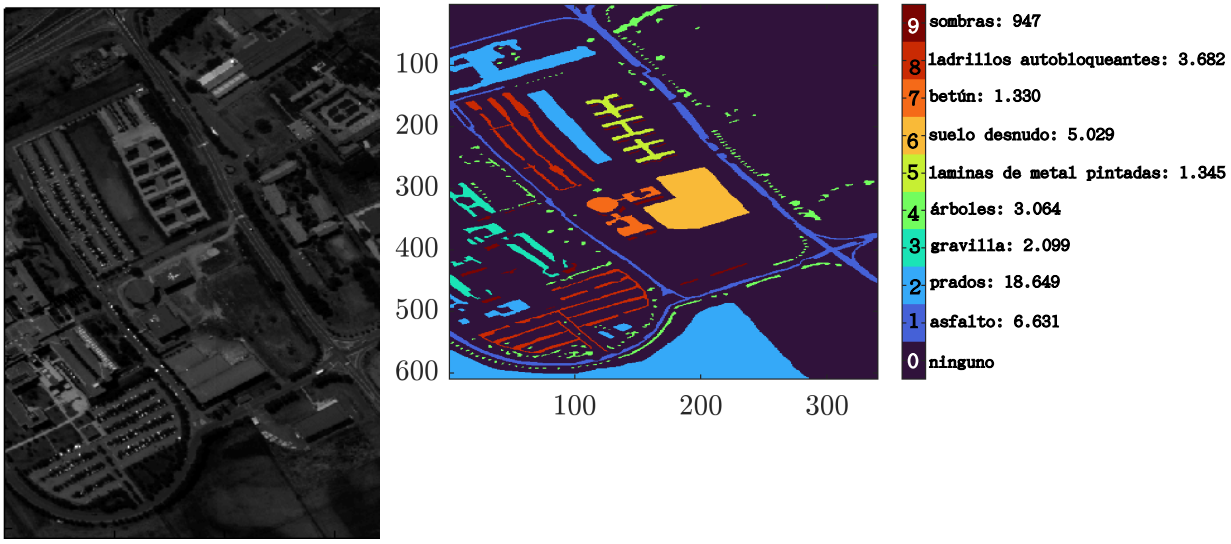
5.2.3. Botswana

Este conjunto contiene 2.079 firmas de entrenamiento para nn, mientras que para li y hamida se utilizaron 2.079 y 2.078 parches de entrenamiento respectivamente, estas muestras están clasificadas en 14 diferentes etiquetas que corresponden al tipo de cobertura del suelo en pantanos

Figura 10

conjunto Pavia U

(a) una banda espectral **(b)** máscara de etiquetas



*N*ota. La figura 10a fue tomada de Graña et al. (2021) mientras que la figura 10b fue adaptada de la misma fuente.

estacionales, pantanos ocasionales y bosques más secos de la parte distal del Delta del Okavango en Botsuana (o Botswana por su nombre en inglés) (Graña et al., 2021).

5.3. Resultados de simulación

En la subsecciones siguientes se muestra un resumen de los resultados mas importantes del algoritmo propuesto, para acceder a los resultados completos se puede consultar el enlace `https://mega.nz/folder/7EoXzB5S#wlK7n31MYFetr3i6Bwyblw`.

5.3.1. Inicialización del conjunto sintético

Se hizo una prueba inicial con el fin de determinar la inicialización que mejor da resultado en este caso: inicialización con ruido o con un conjunto de coreset.

Tal y como se muestra en las secciones siguientes, el método de coreset de kernel-herding (Chen et al., 2010) es superior al método de coreset aleatorio en términos de rendimiento en etapa de testeo, por lo cual este será el único método de coreset que escogeremos para realizar la prueba. En cuanto a la inicialización con ruido, se escogió una distribución de probabilidad $U(0,1)$ para generar dichas muestras.

Esta simulación se realizó aplicando el algoritmo 1 a los datos de entrenamiento de Botswana con el modelo nn para 1, 5 y 10 muestras por clase siguiendo el procedimiento mencionado en la sección 5.1. Tal cómo se puede observar en la figura 11, al inicializar con muestras del conjunto de datos real seleccionadas con el algoritmo de kernel herding se obtiene un resultado considerablemente superior a que si se inicializara con ruido, lo que en primer lugar sugiere que nuestro problema de optimización no es convexo y en segundo lugar significa que esta será la inicialización que se aplicará a todos los experimentos siguientes.

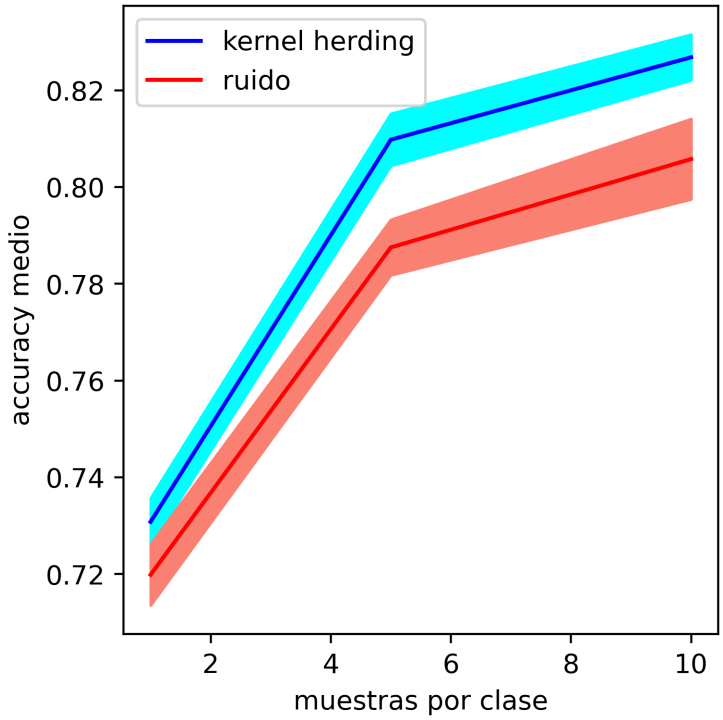
Esta práctica de inicializar el conjunto de datos sintéticos con muestras del conjunto real es muy frecuente en la literatura, de hecho, Zhao and Bilen (2023) inicializan el conjunto con muestras seleccionadas aleatoriamente (método de coreset aleatorio) afirmando que la discrepancia entre la distribución inicial y la final disminuye a medida que aumenta el tamaño del conjunto.

5.3.2. Indian Pines

Al entrenar los 3 modelos de red neuronal anteriormente mencionados con todos los pares imagen-etiqueta de entrenamiento pertenecientes al conjunto Indian Pines, se obtiene un accuracy-1 en etapa de testeo de $97,99\,\% \pm 1,3 \times 10^{-3}$; $98,81\,\% \pm 2,7 \times 10^{-3}$ y $99,04 \pm 2,12 \times 10^{-3}$ para

Figura 11

Comparación entre inicialización con ruido e inicialización con kernel herding



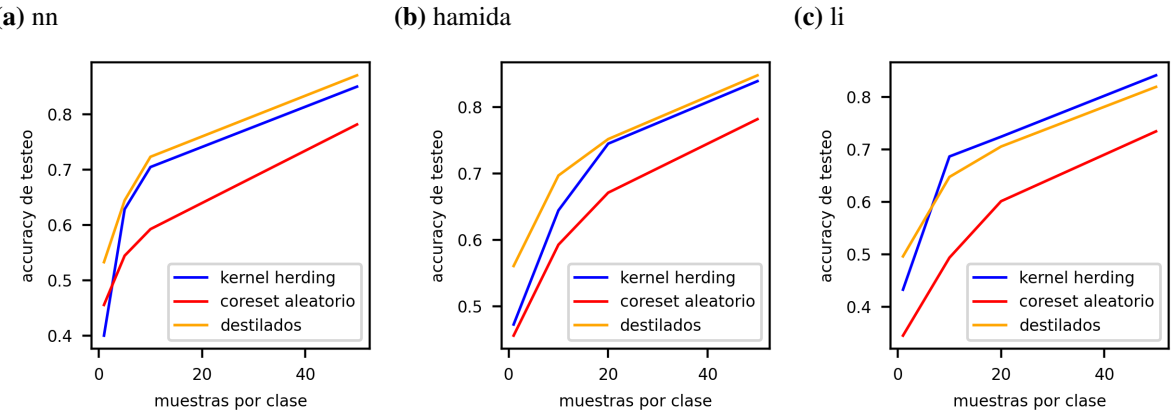
*N*ota. Se tomó como muestra para esta figura el conjunto de datos de Botswana con el clasificador nn. La palabra ≪herding≫ en la leyenda de esta imagen indica una inicialización con ese método de coreset.

nn, hamida y li respectivamente.

En cuanto a los datos destilados, las simulaciones se realizaron con valores de 1, 5, 20 y 50 muestras por clase con el modelo nn (figura 12a) y 1, 10, 20 y 50 muestras por clase para los modelos hamida y li (figuras 12b y 12c respectivamente). En la figura 12 y tabla 1 se observa cómo nuestro método supera a los 2 métodos de coreset seleccionados al utilizar los modelos nn y hamida; aunque al utilizar el modelo li esto solo ocurre para los conjuntos sintéticos mas pequeños.

Figura 12

Resultados para conjuntos destilados y coreset de IndianPines



Cuadro 1

Desviaciones estándar del rendimiento obtenido con el método kernel herding y el método propuesto para el conjunto de datos Indian Pines

Modelo	muestras por clase	kernel herding	destilados
nn	1	$1,6 \times 10^{-2}$	$8,2 \times 10^{-3}$
hamida	1	$\underline{2,14 \times 10^{-3}}$	$2,61 \times 10^{-2}$
li	1	$5,4 \times 10^{-2}$	$2,67 \times 10^{-2}$
nn	5	$\underline{5,08 \times 10^{-3}}$	$7,1 \times 10^{-3}$
hamida	10	$3,83 \times 10^{-2}$	$\underline{2,45 \times 10^{-2}}$
li	10	$7,36 \times 10^{-2}$	$6,62 \times 10^{-2}$
nn	20	$\underline{5,78 \times 10^{-3}}$	$6,62 \times 10^{-2}$
hamida	20	$3,01 \times 10^{-2}$	$2,98 \times 10^{-2}$
li	20	$1,44 \times 10^{-2}$	$1,44 \times 10^{-2}$
nn	50	$5,08 \times 10^{-3}$	$\underline{3,10 \times 10^{-3}}$
hamida	50	$1,93 \times 10^{-2}$	$1,79 \times 10^{-2}$
li	50	$1,43 \times 10^{-2}$	$9,95 \times 10^{-3}$

Nota. Se subrayan las desviaciones estándar mas pequeñas de cada tamaño de conjunto.

Un comentario acerca del método de coreset aleatorio es que, si bien es el método que peores resultados da, su rendimiento no es tan bajo como algunos pueden llegar a pensar al ver la palabra ≪aleatorio≫ en su nombre. Lo que no hay que olvidar es que lo que se selecciona aleatoriamente es un subconjunto de índices a partir de unas muestras reales ya recolectadas. De manera que, por más que el muestreo sea aleatorio, no deja de tratarse de un conjunto que contiene muestras reales. De hecho, en teoría, existe la posibilidad de que un subconjunto creado a partir del método de coreset aleatorio sea exactamente igual a uno creado a partir de kernel herding, e incluso es aún mayor la probabilidad de que estos compartan muestras. Otro factor a tomar en cuenta es que dicho muestreo aleatorio se realiza clase por clase, lo que genera un balanceo de datos siempre y cuando exista la cantidad de muestras requeridas para cada clase.

Después de haber aclarado esto, es momento de echar un vistazo mas a fondo de lo que ocurrió con la distribución de la función embebido después del destilado, esto tomando en cuenta que en el algoritmo 1 la optimización se realiza sobre la distribución de la función de embebido (mas no directamente sobre los datos), así que a fin de visualizar que tanta diferencia hay entre la distribución que entró al algoritmo (evaluada en los datos de kernel-herding) y la que se obtuvo de él (con los datos destilados), en la figura 13 se muestran sus 2 componentes principales junto con los de la distribución de la función embebido del conjunto real completo, esto para algunas clases y (cómo es de notar) en el caso de prueba de 10 muestras por clase, en dicha gráfica se evidencia un comportamiento de la distribución de la función embebido de los datos sintéticos totalmente coherente con sus semejantes de los conjuntos coreset y real.

Para finalizar el análisis de los resultados relacionados con Indian Pines, se seleccionó tam-

Figura 13

Distribución de la función embebido ψ_ϑ evaluada con los conjuntos de datos relacionados con Indian Pines.

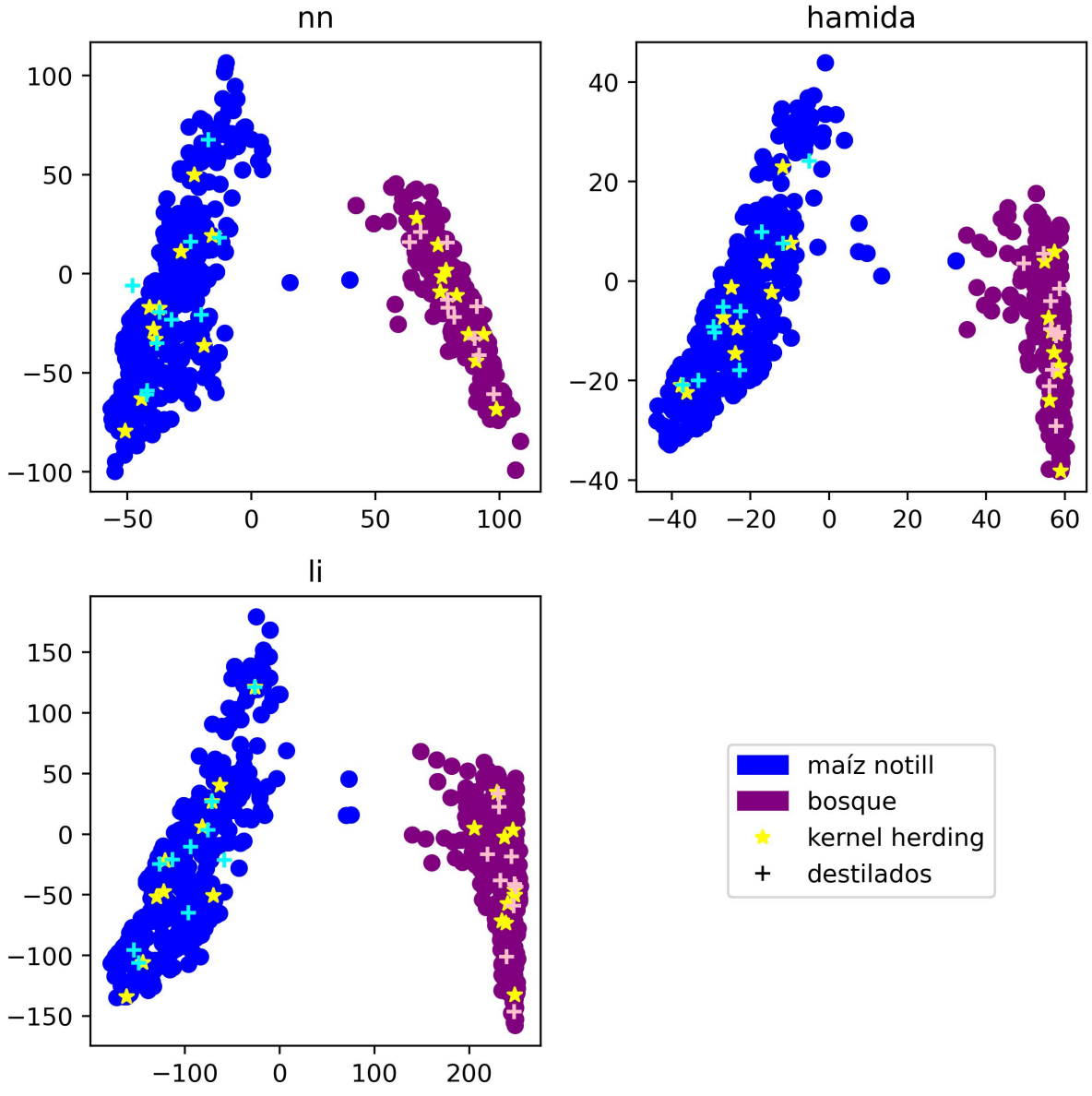
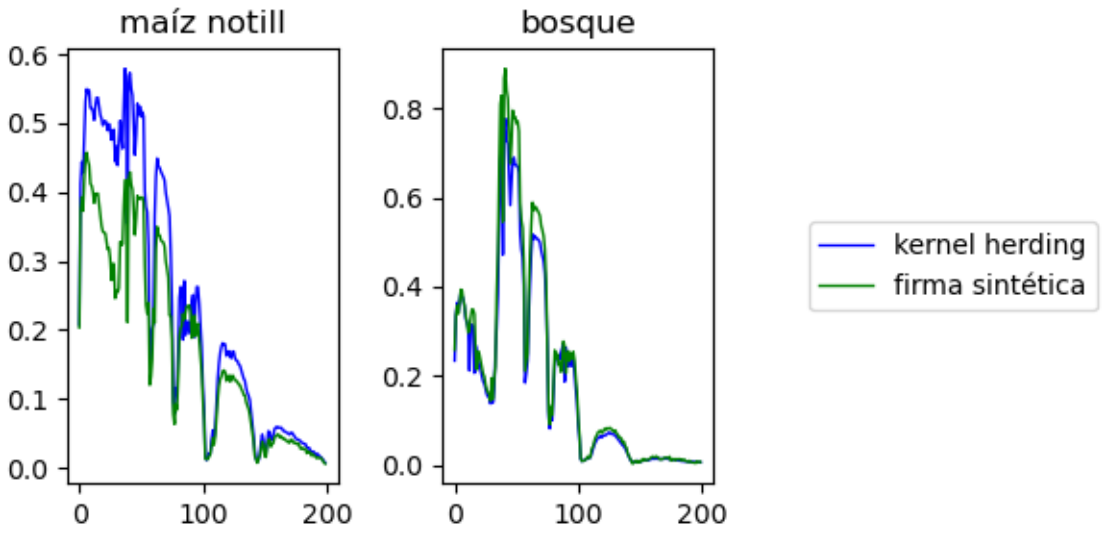


Figura 14

Firmas sintéticas contra firmas de kernel herding de Indian Pines.



bién el caso de prueba de 10 muestras por clase y el clasificador de nn en específico con el objetivo de realizar una comparación similar a la anterior pero esta vez utilizando las firmas sintéticas directamente. Los resultados se pueden visualizar en la figura 14.

5.3.3. Botswana

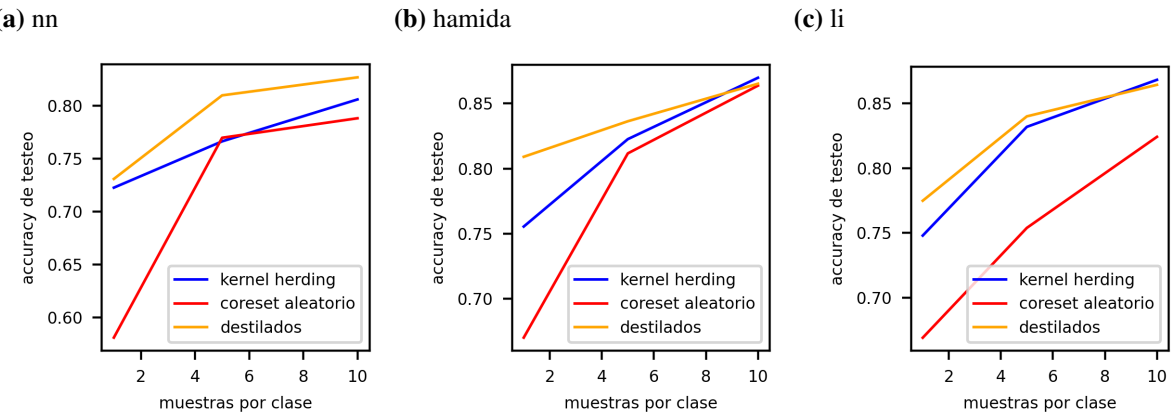
Al entrenar los modelos nn, hamida y li con el conjunto real completo se obtienen valores de accuracy-1 en etapa de testeo de $92{,}46\,\% \pm 2{,}04 \times 10^{-3}$; $91{,}47\,\% \pm 3{,}6 \times 10^{-3}$ y $90{,}54\,\% \pm 2{,}14 \times 10^{-3}$ respectivamente. En cuanto a los resultados con los datos destilados y de coreset, estos pueden ser visualizados en la figura 15 y el cuadro 2 para conjuntos de 1, 5 y 10 muestras por clase; en este caso se seleccionó esta escala debido al que (en comparación con Indian Pines) se observó una mayor rapidez de crecimiento del accuracy de testeo en función de la cantidad de

muestras por clase.

En dichos resultados es de notar cómo nuestro método nuevamente funciona y supera los métodos de coreset de kernel herding y coreset aleatorio en la mayoría de los casos de prueba, aunque, en el caso de hamida y li (figuras 15b y 15c respectivamente) esto solo pasa con los conjuntos más pequeños.

Figura 15

Resultados para conjuntos destilados y coreset de Botswana



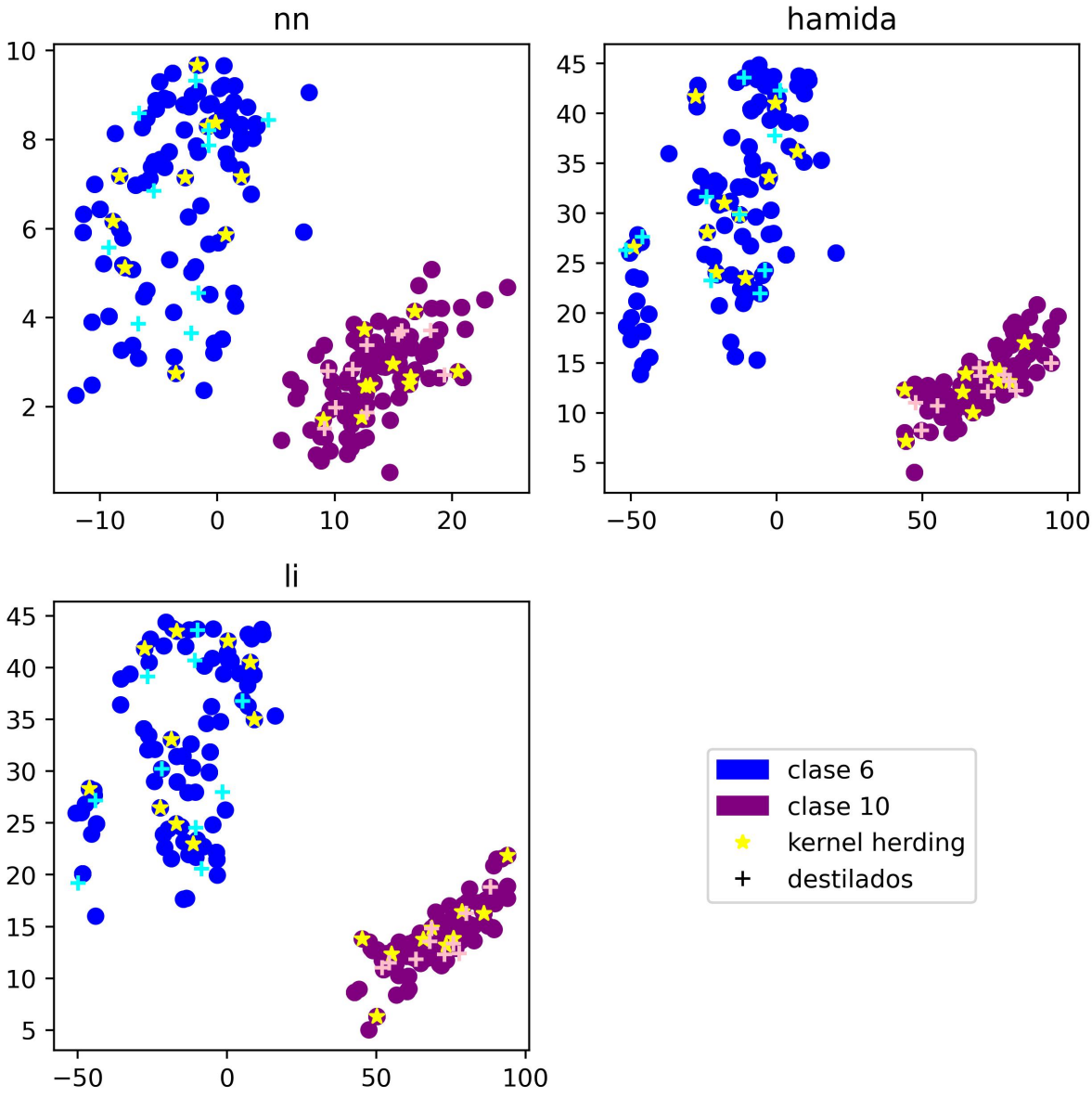
Por último y de la misma forma que se hizo con el conjunto de datos de Indian Pines, en la figura 16 se grafican los componentes principales de la función de embebido y en la figura 17 se muestra la comparación entre algunas firmas antes y después del destilado.

5.3.4. PaviaU

Los valores de precisión-1 obtenidos al entrenar los modelos nn, hamida y li con las muestras de entrenamiento de Pavia U son $84,36\,\% \pm 3,75 \times 10^{-4}$; $83,43\,\% \pm 3,75 \times 10^{-4}$ y $84,25\,\% \pm 2,44 \times 10^{-4}$ respectivamente; mientras que los valores de precisión obtenidos al momento de en-

Figura 16

Distribución de la función embebido ψ_ϑ evaluada con los conjuntos de datos relacionados con Botswana.



Cuadro 2

Desviaciones estándar del rendimiento obtenido con el método kernel herding y el método propuesto para el conjunto de datos Botswana

método	muestras por clase	modelo		
		nn	hamida	li
kernel herding	1	$\underline{2,24 \times 10^{-3}}$	$2,39 \times 10^{-3}$	$4,04 \times 10^{-3}$
destilado	1	$4{,}93 \times 10^{-3}$	$1{,}02 \times 10^{-2}$	$1{,}81 \times 10^{-2}$
kernel herding	5	$6,5 \times 10^{-3}$	$1,96 \times 10^{-2}$	$\underline{3,57 \times 10^{-3}}$
destilado	5	$5{,}40 \times 10^{-3}$	$1{,}37 \times 10^{-2}$	$3{,}60 \times 10^{-3}$
kernel herding	10	$\underline{1,83 \times 10^{-3}}$	$2,36 \times 10^{-3}$	$4,04 \times 10^{-3}$
destilado	10	$4{,}73 \times 10^{-3}$	$5{,}18 \times 10^{-3}$	$5{,}83 \times 10^{-3}$

*N*ota. Se subrayan las desviaciones estándar mas pequeñas de cada tamaño de conjunto.

trenar con los conjuntos de coreset y destilados pueden visualizarse en la gráfica 18 junto con su respecto error estándar en el cuadro 3. Es de señalar que nuevamente nuestro método supera a los métodos de coreset para la mayoría de los casos de prueba. Adicionalmente, se puede observar un curioso comportamiento en el método de coreset aleatorio en las figuras 18a y 18c, un comportamiento que no necesariamente debería ser extraño debido a que este método (cómo se infiere por su nombre) no discrimina entre las muestras mas representativas de las que no lo son.

En la figura 19 se muestra la distribución de la función embebido ψ_ϑ para los datos reales, kernel herding y destilados, mientras que la figura 20 muestra una comparación entre algunas firmas sintéticas y su respectiva inicialización de kernel herding.

5.3.5. Importancia experimental de la desviación estándar

En el capitulo 4 se dieron argumentos teóricos del por qué debería tomarse en cuenta la desviación estándar de la función embebido ψ_ϑ en la optimización de los datos sintéticos S, por

Figura 17

Firmas sintéticas contra firmas de kernel herding de Botswana.

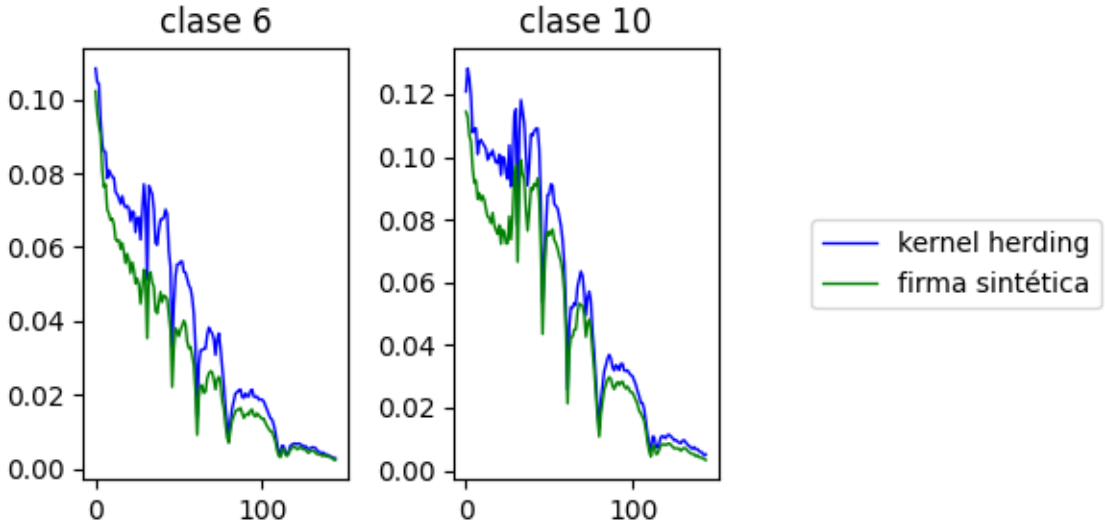
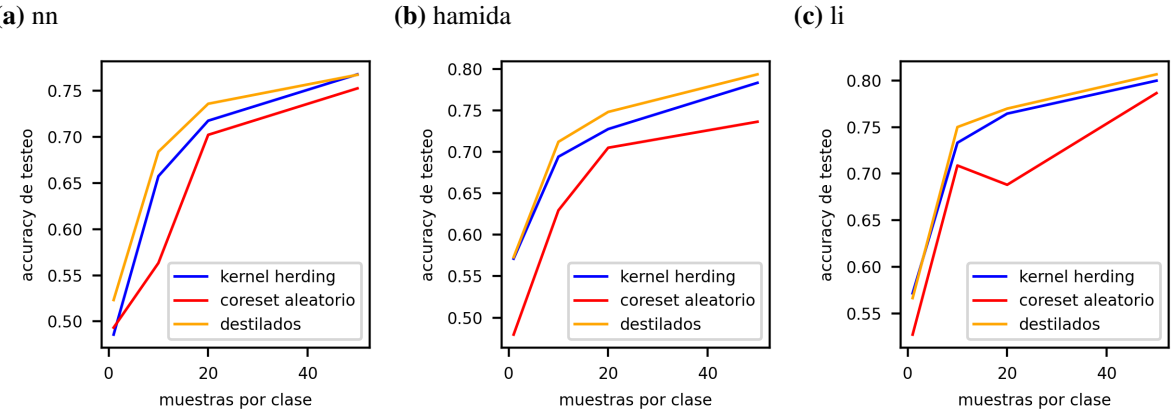


Figura 18

Resultados para conjuntos destilados y coreset de Pavia U



Cuadro 3

Desviaciones estándar del rendimiento obtenido con el método kernel herding y el método propuesto para el conjunto de datos Pavia U

Método	muestras por clase	modelo		
		nn	hamida	li
kernel herding	1	$3,21 \times 10^{-2}$	$2,5 \times 10^{-2}$	$2,45 \times 10^{-2}$
destilado	1	$\underline{4,00 \times 10^{-3}}$	$2,47 \times 10^{-2}$	$2,96 \times 10^{-2}$
kernel herding	10	$7,17 \times 10^{-3}$	$8,76 \times 10^{-3}$	$8,65 \times 10^{-3}$
destilado	10	$5,99 \times 10^{-3}$	$7,39 \times 10^{-3}$	$\underline{5,73 \times 10^{-3}}$
kernel herding	20	$5,18 \times 10^{-3}$	$1,64 \times 10^{-2}$	$\underline{3,84 \times 10^{-3}}$
destilado	20	$4,04 \times 10^{-3}$	$1,33 \times 10^{-2}$	$3,9 \times 10^{-3}$
kernel herding	50	$2,95 \times 10^{-3}$	$6,24 \times 10^{-3}$	$5,23 \times 10^{-3}$
destilado	50	$\underline{1,69 \times 10^{-3}}$	$6,92 \times 10^{-3}$	$7,29 \times 10^{-3}$

*N*ota. Se subrayan las desviaciones estándar mas pequeñas de cada tamaño de conjunto.

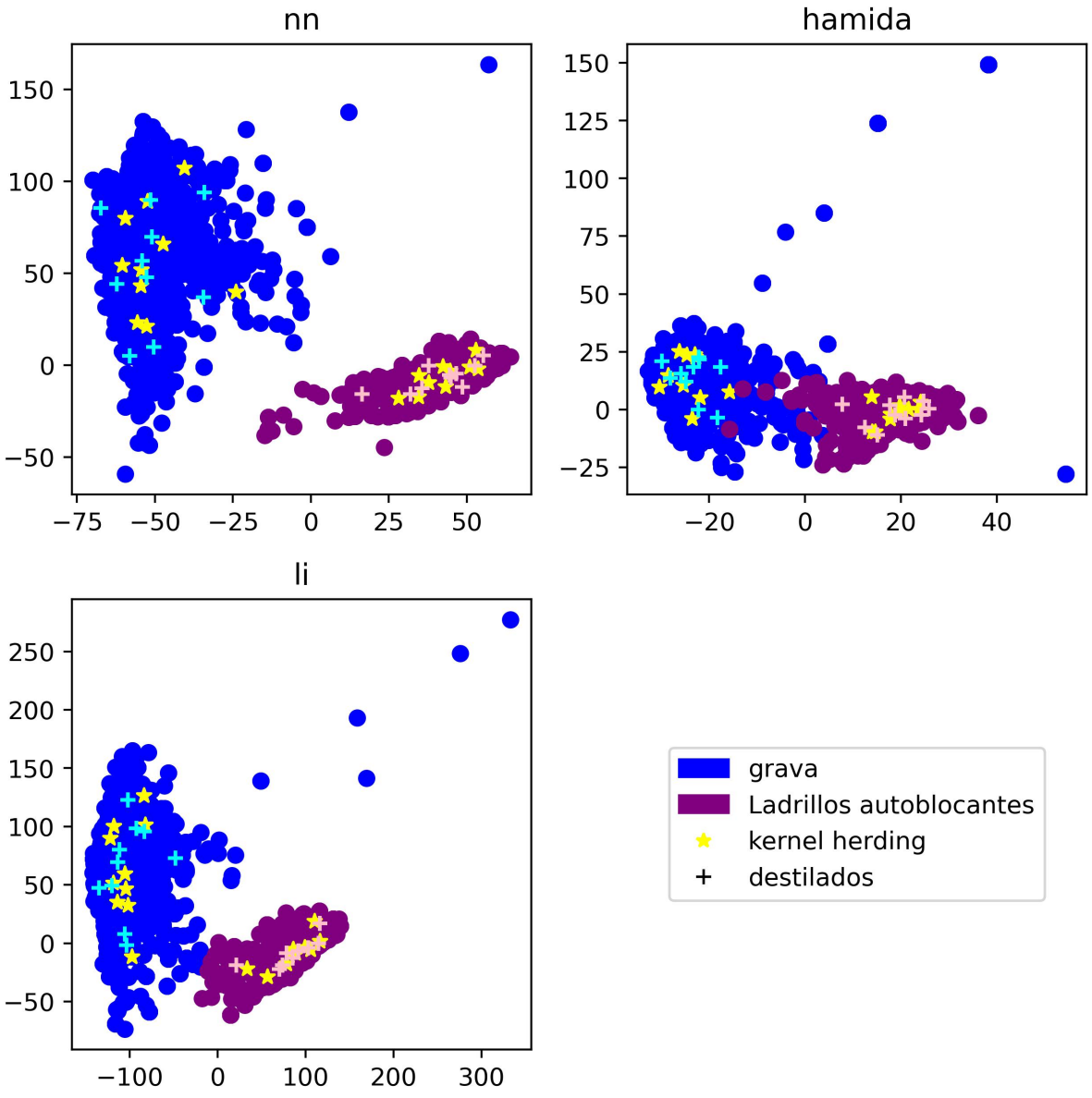
lo cuál se añadió este estadístico en la ecuación 20 dándole igual importancia que el término de la media; sin embargo, es posible hacer una modificación a dicha ecuación con el propósito de darle una importancia personalizada, de manera que definimos un término regularizador $\delta \in [0,1]$ que representará la importancia que se le da a la desviación estándar con respecto a la media en el problema del destilado de imágenes hiperespectrales, de tal forma que la nueva función de pérdida se define de la siguiente manera:

$$\min_{S} \mathbb{E}_{\vartheta \sim P_\vartheta} \left((1-\delta) \left\| \mu_\tau - \mu_S \right\|^2 + \delta \left\| \sigma_\tau - \sigma_S \right\|^2 \right). \tag{24}$$

Para poner a prueba la relevancia de dicha variable se destiló el conjunto de datos de Indian Pines utilizando el modelo hamida y función de pérdida definida en la ecuación 24 en lugar de la

Figura 19

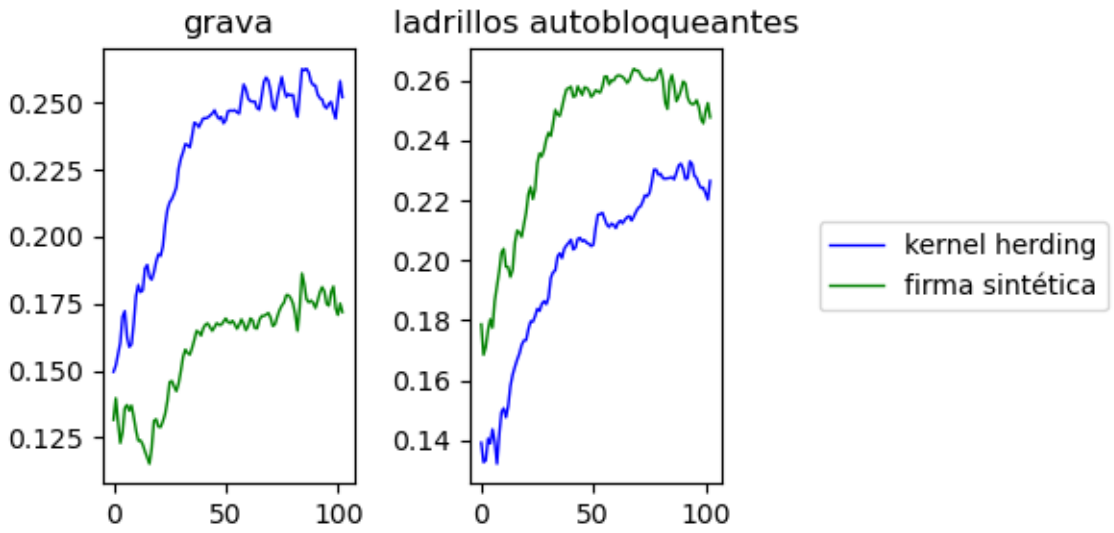
Distribución de la función embebido ψ_ϑ evaluada con los conjuntos de datos relacionados con Pavia U.



definida por 20. Esta prueba se realizó para valores de $\delta \in \{0;0,25;0,5;0,75;1\}$ cuyos resultados pueden visualizarse en la tabla 4. Nótese que en el caso $\delta = 0$ el problema de optimización es el

Figura 20

Firmas sintéticas contra firmas de kernel herding de Pavia U.



*N*ota. estas firmas corresponden al conjunto de entrenamiento del modelo nn.

mismo que propuso Zhao and Bilen (2023) y en el caso $\delta = 0{,}5$ la función de costo es un factor escalar de la ecuación 20 por lo que serían problemas de optimización equivalentes.

Habiendo aclarado esto, si tomamos el método propuesto por Zhao and Bilen (2023) como referencia, podemos observar que nuestro método ($\delta = 0,5$) lo supera en 3 de los 4 casos de prueba; sin embargo, la configuración $\delta = 0{,}75$ si logró superarlo en todos ellos (incluso con un error estándar menor) siendo así la configuración que mejor dio resultado, mientras que la configuración $\delta = 1$ fue el caso opuesto (tal y como era de esperarse).

Cuadro 4

Variación experimental de la importancia de la desviación estándar en la ecuación 24

δ	cantidad de muestras por clase			
	1	10	20	50
0	$55{,}24\,\% \pm 1{,}94 \times 10^{-2}$	$69{,}79\,\% \pm 2{,}89 \times 10^{-2}$	$75{,}36\,\% \pm 3{,}24 \times 10^{-2}$	$84{,}87\,\% \pm 2{,}22 \times 10^{-2}$
0.25	$54{,}98\,\% \pm 2{,}85 \times 10^{-2}$	$70{,}28\,\% \pm 3{,}00 \times 10^{-2}$	$74{,}99\,\% \pm 2{,}82 \times 10^{-2}$	$85{,}12\,\% \pm 1{,}83 \times 10^{-2}$
0.5	$55{,}46\,\% \pm 3{,}11 \times 10^{-2}$	$69{,}9\,\% \pm 2{,}93 \times 10^{-2}$	$75{,}24\,\% \pm 2{,}81 \times 10^{-2}$	$\underline{85{,}2\,\% \pm 1{,}69 \times 10^{-2}}$
0.75	$\underline{55{,}79\,\% \pm 2{,}84 \times 10^{-2}}$	$\underline{70{,}3\,\% \pm 2{,}66 \times 10^{-2}}$	$\underline{75{,}75\,\% \pm 2{,}55 \times 10^{-2}}$	$85{,}07\,\% \pm 1{,}43 \times 10^{-2}$
1	$52{,}01\,\% \pm 2{,}31 \times 10^{-2}$	$69{,}22\,\% \pm 2{,}94 \times 10^{-2}$	$74{,}82\,\% \pm 2{,}62 \times 10^{-2}$	$84{,}52\,\% \pm 1{,}67 \times 10^{-2}$

*N*ota. Se subrayan los mayores valores de precisión para cada tamaño del conjunto.

6. Trabajo Futuro

Debido a lo relativamente nuevo que es el método del destilado y a las limitaciones ya mencionadas en su aplicación a datos de alta dimensionalidad, aún hay mucho por explorar en este campo. Por ejemplo, la aplicación del destilado a problemas inversos, la propuesta de más algoritmos que aumenten aún más la diferencia en el rendimiento con respecto a los métodos de coreset, e incluso problemas de clasificación en los cuales, para una misma entrada, se obtenga un arreglo de etiquetas.

7. Conclusiones

El método propuesto para el destilado de imágenes hiperespectrales mostró un rendimiento medio superior a los métodos del estado del arte escogidos en la mayoría de los casos de prueba, lo que significa que el algoritmo aquí propuesto puede generar nuevos datos a partir de los originales de tal manera que con estos sea posible realizar entrenamientos mas rápidos y eficientes con una pérdida de rendimiento razonable en etapa de testeo.

Adicionalmente, se dejó en claro la importancia que tiene la desviación estándar de la función embebido en este tipo de métodos, de modo que el darle la importancia adecuada a este estadístico al momento de la optimización ayuda a aumentar el rendimiento medio de los datos en etapa de testeo y reducir su error estándar.

Bibliografía

Ariolfo Camacho, E. V. and Arguello, H. (2022). Hyperspectral and multispectral image fusion addressing spectral variability by an augmented linear mixing model. *International Journal of Remote Sensing*, 43(5):1577–1608.

Audebert, N., Le Saux, B., and Lefèvre, S. (2019). Deep learning for classification of hyperspectral data: A comparative review. *IEEE Geoscience and Remote Sensing Magazine*, 7(2):159–173.

Bacca, J., Martinez, E., and Arguello, H. (2023). Computational spectral imaging: A contemporary overview. Copyright - © 2023. This work is published under http://arxiv.org/licenses/nonexclusive-distrib/1.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Última actualización - 2023-04-13.

Ben Hamida, A., Benoit, A., Lambert, P., and Ben Amar, C. (2018). 3-d deep learning approach for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 56(8):4420–4434.

Bi, X. and Wang, H. (2019). Early alzheimer's disease diagnosis based on eeg spectral images using deep learning. *Neural Networks*, 114:119–135.

Camps-Valls, G., Bandos Marsheva, T. V., and Zhou, D. (2007). Semi-supervised graph-based

hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 45(10):3044–3054.

Camps-Valls, G. and Bruzzone, L. (2005). Kernel-based methods for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 43(6):1351–1362.

Chen, Y., Welling, M., and Smola, A. (2010). Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, page 109–116, Arlington, Virginia, USA. AUAI Press.

Graña, M., Veganzons, M., and Ayerdi, B. (2021). `https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Indian_Pines`.

Guo, C., Zhao, B., and Bai, Y. (2022). Deepcore: A comprehensive library for coreset selection in deep learning. Copyright - © 2022. This work is published under http://creativecommons.org/licenses/by-nc-nd/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Última actualización - 2022-08-17.

Lei, S. and Tao, D. (2024). A comprehensive survey of dataset distillation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):17–32.

Li, G., Togo, R., Ogawa, T., and Haseyama, M. (2023). Dataset distillation using parameter pruning. Copyright - © 2023. This work is published under http://arxiv.org/licenses/nonexclusive-

distrib/1.0/ (the “License”). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Última actualización - 2023-08-23.

Li, S., Song, W., Fang, L., Chen, Y., Ghamisi, P., and Benediktsson, J. A. (2019). Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6690–6709.

Li, Y., Zhang, H., and Shen, Q. (2017). Spectral–spatial classification of hyperspectral imagery with 3d convolutional neural network. *Remote Sensing*, 9(1).

Md. Palash Uddin, M. A. M. and Hossain, M. A. (2021). Pca-based feature reduction for hyperspectral remote sensing image classification. *IETE Technical Review*, 38(4):377–396.

Rodríguez-Pulido, F. J., Gordillo, B., Heredia, F. J., and González-Miret, M. L. (2021). Cielab – spectral image matching: An app for merging colorimetric and spectral images for grapes and derivatives. *Food Control*.

Sachdeva, N. and McAuley, J. (2023). Data distillation: A survey. *ArXiv*, abs/2301.04272.

Tamburrini, G. (2022). The ai carbon footprint and responsibilities of ai scientists. *Philosophies*, 7(1).

Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. (2020). Dataset distillation.

Woo, H., Acuna, M., Madurapperuma, B., Jung, G., Woo, C., and Park, J. (2021). Application of

maximum likelihood and spectral angle mapping classification techniques to evaluate forest fire severity from uav multi-spectral images in south korea. *Sensors and Materials*, 33:1.

Yu, R., Liu, S., and Wang, X. (2024). Dataset distillation: A comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):150–170.

Zhao, B. and Bilen, H. (2023). Dataset condensation with distribution matching. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6503–6512.