Clasificación de imágenes de teledetección mediante ELM kernel y perfiles morfológicos en GPU

Alberto S. Garea[1], Dora B. Heras[1], and Francisco Argüello[2]

Resumen— **Hoy en día el uso de sensores hiperespectrales se ha extendido a una gran variedad de aplicaciones tales como la clasificación de imágenes de teledetección. Recientemente se ha presentado un esquema de clasificación espectral-espacial (ELM-EMP) basado en *Extreme Machine Learning* (ELM) y Perfiles Morfológicos Extendidos (EMP) obtenidos utilizando Análisis en Componentes Principales (PCA) y operaciones morfológicas. En este trabajo se han introducido varias mejoras para incrementar la precisión de la clasificación del esquema original (ELM-EMP). En particular, se presenta un esquema que utiliza un clasificador ELM basado en *k*ernels (KELM-EMP) y se aplica una regularización espacial. Además, se ha realizado una implementación eficiente sobre Unidades de Procesamiento Gráfico (GPUs) de estos esquemas. En cuanto a esta proyección en GPU, se han aplicado diferentes técnicas como pueden ser el uso de librerías CUDA optimizadas y la ejecución de bloques asíncronos. Como resultado, la precisión obtenida por los dos esquemas (ELM-EMP-S y KELM-EMP-S) es mejor que para el esquema ELM-EMP original y el tiempo de ejecución se ha reducido significativamente.**

Palabras clave— **Teledetección; clasificación, datos hiperespectrales, *Extreme Machine Learning* (ELM), Análisis de Componentes Principales (PCA), Perfiles Morfológicos Extendidos (EMP).**

I. Introducción

LOS avances en la tecnología de sensores de imagen han hecho posible ampliar el rango del espectro electromagnético que puede ser capturado, pasando de unas pocas bandas en imágenes multiespectrales a cientos de bandas en imágenes hiperespectrales [1]. Este amplio rango proporciona más información que puede ser utilizada para mejorar las tareas de reconocimiento y clasificación de materiales. Debido a la alta dimensionalidad de las imágenes hiperespectrales, se necesitan técnicas específicas para aprovechar toda esa información [2]. En el campo de las redes neuronales usamos el término *Extreme Machine Learning*(ELM) para describir un tipo de *Single-hidden Layer Feedforward Neural Network* (SLFN) con pesos aleatorios [3]. Se ha visto en [4], [5] que una SLFN (con N nodos en la capa oculta) donde los pesos y los sesgos se eligen aleatoriamente puede aprender exactamente N observaciones distintas [6]. El mismo autor también propone el uso de la pseudoinversa en el algoritmo ELM para evitar la baja inferencia de los algoritmos de entrenamiento con retropropagación. En el ELM, la asignación aleatoria de los pesos entre la capa de entrada y la capa oculta produce una amplia variación en la precisión de la clasificación en diferentes pruebas usando el mismo número de nodos ocultos. Para evitar esto, [7] propone, para el algoritmo ELM, el uso de funciones *kernel* en sustitución de la capa oculta.

Como valor añadido, el algoritmo ELM es muy adecuado para ser implementado en Unidades de Procesamiento Gráfico (GPUs) de uso común y otras arquitecturas paralelas debido a que la mayoría de sus operaciones están relacionadas con operaciones matriciales que pueden ser calculadas en bloques independientes, es decir, sin dependencia de datos entre ellos. Heeswijk et al. [8] desarrolló una implementación en GPU de parte de un sistema de clasificación basado en la ejecución de un centenar de instancias de ELM. Más tarde se publicó una implementación eficiente, completamente en GPU, del clasificador hiperespectral ELM [9]. Sin embargo, no se han publicado implementaciones en GPU para la versión *kernel* del ELM.

Las precisión de los resultados proporcionados por una clasificación píxel a píxel puede ser mejorada aportando información espacial al clasificador [10], [2], [11], [12]. Esto significa que la decisión de asignar un píxel a una clase determinada se basa tanto en la característica espectral, que es el valor del píxel, como en cierta información extraída del vecindario del píxel, que se puede considerar información espacial. Entre los métodos espaciales podemos incluir los basados en segmentación, como la transformada *watershed* [13], [14], [9] y la segmentación basada en autómatas celulares (ECAS-II) [15], los basados en técnicas de agrupamiento por particiones [16], los basados en los bosques de expansión mínima [2] o los basados en filtrado local [17].

La información espacial también puede ser extraída de los datos hiperespectrales mediante herramientas morfológicas. Los operadores morfológicos más usados son la apertura y el cierre, que están basados en las operaciones fundamentales de erosión y dilatación. A partir de esas operaciones básicas se pueden construir los llamados Perfiles Morfológicos (MP) [18], [19], [20], [12]. Un MP contiene información de las estructuras de la imagen a diferentes tamaños de resolución. En la teledetección, los MPs se calculan normalmente en base a los datos obtenidos del Análisis en Componentes Principales (PCA) realizado sobre los datos hiperespectrales [21], [2], [22]. Si se conservan varias componentes prin-

[1]Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS), Universidade de Santiago de Compostela, Spain, {jorge.suarez.garea, dora.blanco}@usc.es.

[2]Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Spain, francisco.arguello@usc.es.

cipales, los MPs obtenidos para cada uno de esas componentes se pueden utilizar todos juntos en un Perfil Morfológico Extendido (EMP) [23], [24]. Esto se puede generalizar más con el fin de modelar la información espacial de forma más precisa. Por ejemplo, se puede crear un Perfil de Atributos (AP) morfológicos del mismo modo que un MP [25]. En [11] se crea un Perfil de Atributos Morfológicos Extendido (EMAP) usando filtros de atributos morfológicos. En [18] y [21] se presentan esquemas de clasificación espectral-espacial para imágenes de teledetección basados en SVM e introduciendo información proporcionada por un EMP. En el caso de [26] se utilizan características Gabor, en lugar de EMP, sobre las componentes generadas por el PCA para, posteriormente, concatenar el resultado con la imagen original y realizar la clasificación mediante el ELM *kernel*. Este último artículo también presenta un clasificador basado en KELM con Predicciones Multihipótesis (MH).

En [27] se presenta un esquema de clasificación espectral-espacial para imágenes hiperespectrales de teledetección basado en ELM y que además integra información proporcionada por un EMP. El perfil se crea a partir de las componentes generadas por el PCA. El esquema espectral-espacial propuesto permite asignar diferentes pesos a las características espectrales y a las espaciales.

En este artículo se han abordado principalmente tres tareas. La primera tarea ha sido el desarrollo de un esquema similar al publicado en [27], basado en ELM con funciones *kernel*. Estas funciones *kernel* sustituyen a la generación de pesos aleatorios en el esquema ELM original. La segunda ha sido la mejora, en términos de precisión, del esquema presentado en [27]. La tercera tarea ha sido la implementación, sobre GPU, de los esquemas descritos previamente para reducir el tiempo de ejecución. El resto del artículo se organiza como sigue: La sección 2 describe los algoritmos utilizados. En la sección 3 se presenta la implementación del esquema de clasificación sobre GPU. La evaluación se realiza en la sección 4. Y, finalmente, la sección 5 presenta las conclusiones.

II. Clasificación espectral-espacial basada en ELM

En esta sección se explican las diferentes etapas del esquema de clasificación que llamaremos ELM-EMP, como en [27] (Ver Fig. 1). Este proceso comienza con la extracción de la información espacial. El primer paso es reducir la dimensionalidad de la imagen hiperespectral usando PCA. El segundo paso es la construcción del EMP mediante operaciones morfológicas. Tanto las contribuciones espectrales como las espaciales se ajustan usando un método de combinación de características llamado concatenación ponderada. Esto permite asignar los pesos k_w y k_s a los datos espectrales y espaciales respectivamente. Finalmente, utilizamos ELM para la clasificación.

A. Clasificación basada en KELM

ELM fue originalmente desarrollado como una técnica de entrenamiento para un tipo de SLFN con pesos aleatorios [28], [6]. En [7] se propuso el uso de una función *kernel* en lugar del uso de pesos aleatorios. El objetivo era evitar la gran variación, en términos de precisión de la clasificación, producida en diferentes entrenamientos y debida a la aleatoriedad de los pesos. Además, [7] sugirió sumar un valor positivo $\frac{1}{C}$ (donde C es definido por el usuario) para calcular los pesos de $\boldsymbol{\beta}$ como:

$$\boldsymbol{\beta} = \mathbf{H}^{\mathrm{T}} \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\mathrm{T}} \right)^{-1} \mathbf{T}, \tag{1}$$

donde $\mathbf{H}$ es la matriz de salida de la capa oculta y $\mathbf{T}$ es la matriz objetivo para el conjunto de datos de entrenamiento. Con este valor positivo se tienden a obtener soluciones más estables, así como una mejora en el rendimiento. Por lo tanto, para $\boldsymbol{x} \in \mathbb{R}^d$, la función de salida del ELM es:

$$f(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta} = \boldsymbol{h}(\boldsymbol{x})\mathbf{H}^{\mathrm{T}} \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\mathrm{T}} \right)^{-1} \mathbf{T}, \tag{2}$$

donde $\boldsymbol{h}(\boldsymbol{x})$ es un mapeo de características. La matriz del *kernel* para ELM, $\boldsymbol{\Omega}$, puede ser representada como:

$$\begin{aligned} \boldsymbol{\Omega}_{\mathrm{ELM}} &= \mathbf{H}\mathbf{H}^{T} : \Omega_{ELM_{i,j}} \\ &= [\boldsymbol{h}(\boldsymbol{x}_i) \times \boldsymbol{h}(\boldsymbol{x}_j)] = [K(\boldsymbol{x}_i, \boldsymbol{x}_j)], \end{aligned} \tag{3}$$

donde x_i y x_j son muestras de entrenamiento, $i = 1, \ldots, N$, $j = 1, \ldots, N$, y $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ es la función *kernel*. Finalmente, la función de salida se puede escribir como:

$$f(\boldsymbol{x}) = \underbrace{\begin{bmatrix} K(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ K(\boldsymbol{x}, \boldsymbol{x}_N) \end{bmatrix}^{\mathrm{T}}}_{Kernel\_1} \left(\frac{\mathbf{I}}{C} + \underbrace{\boldsymbol{\Omega}_{\mathrm{ELM}}}_{Kernel\_2} \right)^{-1} \mathbf{T}. \tag{4}$$

Así, podemos resumir el ELM con *kernel* como:

Algoritmo ELM con *kernel*: Dado un conjunto de entrenamiento $\mathbf{J} = \{\boldsymbol{x}_i | \boldsymbol{x}_i \in \mathbb{R}^d, i = 1, \ldots, N\}$, una matriz de entrenamiento $\mathbf{T}$, una función *kernel* $K(\boldsymbol{u}, \boldsymbol{v})$ (e.g., $K(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\gamma||\boldsymbol{u} - \boldsymbol{v}||^2)$) y los parámetros definidos por el usuario C y γ,

1. Calcular $\left(\frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\mathrm{ELM}}\right)^{-1} \mathbf{T}$, con $\boldsymbol{\Omega}_{\mathrm{ELM}} = [K(\boldsymbol{x}_i, \boldsymbol{x}_j)]$, $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbf{J}$ y $K(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\gamma||\boldsymbol{u} - \boldsymbol{v}||^2)$.
2. Calcular $\begin{bmatrix} K(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ K(\boldsymbol{x}, \boldsymbol{x}_N) \end{bmatrix}^{\mathrm{T}}$ donde $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbf{J}$ y $\boldsymbol{x}$ hace referencia a todos los puntos del conjunto de datos de test.
3. Calcular la función de salida del ELM como en la Ecuación (4).

En el caso de nuestra imagen hiperespectral, cada muestra de entrenamiento representa un píxel de la

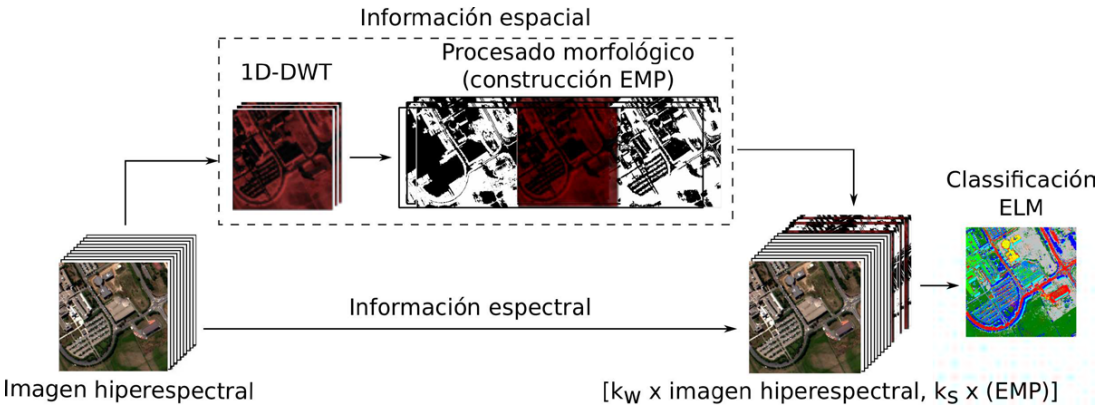

Fig. 1. Esquema de clasificación espectral-espacial basado en ELM y mapeo de características compuesto (ELM-EMP).

imagen seleccionado de forma aleatoria, que a su vez es un componente espectral y espacial ponderado del píxel. Los componentes espaciales proporcionados por el EMP son multiplicados por el factor k_s, mientras que los componentes de la imagen original son multiplicados por la constante k_w. La salida que se obtiene después de la fase de clasificación es la predicción de la clase a la que pertenece cada uno de los píxeles de la imagen.

B. Extracción de características

La principal función del PCA es reducir la dimensionalidad del conjunto de datos, constituido por un gran número de variables interrelacionadas, mientras se retiene, tanto como sea posible, la variación presente en el conjunto de datos.

Matemáticamente existen varios métodos para calcular el PCA. Se puede realizar usando *Single Value Decomposition* (SVD) sobre el conjunto de datos o mediante *Eigenvalue Decomposition* (EVD) sobre la matriz de covarianza del conjunto de datos, siendo este último el método utilizado en este documento. Ambos métodos necesitan que el conjunto de datos esté centrado [29].

Vamos a asumir que la matriz de datos $\mathbf{X}$ está centrada, es decir, a cada elemento de una columna de $\mathbf{X}$ se le ha restado la media aritmética de dicha columna, quedando ahora dicha media igual a cero. Por lo tanto, la matriz de covarianza $\mathbf{A} = \mathbf{X}^{\mathrm{T}}\mathbf{X}/n$, donde n es el número de características, puede ser diagonalizada:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}, \tag{5}$$

donde los elementos de la diagonal de $\mathbf{S}$ son los valores singulares de $\mathbf{A}$ y las primeras columnas de $\mathbf{U}$ y $\mathbf{V}$ son los vectores singulares derecho e izquierdo de $\mathbf{A}$ respectivamente. La proyección de los datos en el eje principal se conoce como Componentes Principales (PCs).

C. EMP

Las transformaciones morfológicas fueron propuestas para utilizar la información espacial en la clasificación de teledetección [30], [21], [23]. El MP fue introducido en [21] y extendido a imágenes multidimensionales en [23] para extraer información sobre el contraste y el tamaño de las estructuras presentes en la imagen. El MP de orden n de la imagen I se puede expresar como:

$$\begin{aligned}\mathrm{MP}^{(n)}(I) = \{&\gamma_r^{(n)}(I), \ldots, \gamma_r^{(1)}(I),\\ &I, \phi_r^{(1)}(I), \ldots, \phi_r^{(n)}(I)\},\end{aligned} \tag{6}$$

siendo $\gamma_r^{(i)}$ y $\phi_r^{(i)}$ los operadores de apertura y cierre por reconstrucción respectivamente, con un elemento estructural i desde 1 a n, cuyo tamaño incrementa normalmente en pasos de 1 o 2.

Cuando se usa la aproximación de MP sobre los datos hiperespectrales, se usan los m PCs más significativos como imagen base. El resultado es un EMP,

$$EMP_m^{(n)}(I) = \{(MP)_1^{(n)}(I), \ldots, (MP)_m^{(n)}(I)\}, \tag{7}$$

con $m(2n+1)$ componentes.

Una vez que se ha extraído de la imagen la información espacial, esta debe ser integrada en el clasificador.

III. Clasificación espectral-espacial de una imagen hiperespectral sobre GPU

En esta sección introduciremos algunos de los principios de la programación *Compute Unified Device Architecture* (CUDA), así como la implementación de los algoritmos propuestos en la sección II. Los *kernels* ejecutados en GPU se encuentran entre los símbolos <>. Los pseudocódigos también incluyen los acrónimos GM y SM para indicar que los *kernels* usan memoria global o memoria compartida respectivamente.

A. Fundamentos de programación CUDA

CUDA permite a las GPUs de Nvidia ejecutar programas a través de la invocación de funciones paralelas llamadas *kernels* [31]. Los *kernels* son ejecutados por hilos que a su vez se organizan en bloques. Los bloques se disponen en una rejilla donde se mapean a una jerarquía de procesadores CUDA en la GPU. Los

hilos tienen acceso a múltiples espacios de memoria, como pueden ser la memoria local y los registros. Cada bloque de hilos tiene un espacio de memoria compartida que es visible únicamente a los hilos del mismo bloque y cuyo tiempo de vida es igual al del bloque. Por último, todos los hilos pueden acceder al mismo espacio de memoria global.

El tiempo de vida de la memoria compartida dificulta el poder compartir datos entre bloques de hilos, lo que obliga a usar la memoria global que es más lenta. La arquitectura Kepler [31] incluye una jerarquía con dos niveles de caché. En este trabajo se han seguido diferentes estrategias para optimizar el rendimiento:

1. **Maximizar la ejecución paralela.** Organizando los algoritmos en bloques computacionalmente independientes.
2. **Mejorar la eficiencia en el uso de la jerarquía de memoria.** Para realizar el máximo número de cálculos con los datos ya almacenados en la memoria compartida.
3. **Reducir el número de sincronizaciones globales mediante el cómputo con bloques asíncronos.** En el cálculo del EMP, cada bloque se actualiza varias veces a través de una sincronización local antes de realizar una sincronización global.
4. **Añadir un borde a las regiones de datos.** Dado que en la fase de construcción del EMP cada píxel requiere datos de sus vecinos, cada región de datos se amplía con una frontera con el fin de minimizar las dependencias entre los bloques
5. **Superposición de operaciones en la CPU y la GPU.** A veces es posible solapar la ejecución de operaciones independientes en la CPU y la GPU, reduciendo el tiempo de ejecución. En particular, en el cálculo del ELM, la generación de los pesos aleatorios para los datos de entrada y las neuronas ocultas se solapa con la generación de la matriz $\mathbf{X}_{\text{test}}$.
6. **Explotar las librerías disponibles.** Dado que la mayoría de las operaciones de los algoritmos utilizados en este trabajo son operaciones con matrices, se han utilizado diferentes bibliotecas CUDA optimizadas para álgebra lineal y procesamiento de imágenes. En particular, **MAGMA** [32], que es una librería de álgebra lineal para arquitecturas heterogéneas usada para calcular la matriz $\mathbf{X}_{\text{test}}$ y la pseudoinversa de la matriz $\mathbf{H}$, y **CULA** [33], que es un conjunto de librerías algebraicas usadas para calcular los EVD en el algoritmo PCA. Por último, **CUBLAS** [34], que es una versión en GPU de la librería estándar *Basic Linear Algebra Subprograms* (BLAS) y que se utilizó para obtener la matriz de correlación en el cálculo de la PCA y para calcular la matriz de pesos en la fase de entrenamiento, tanto para el algoritmo ELM como para el KELM.

Algoritmo 1 Algoritmo KELM en GPU

Entrada: El conjunto de datos de la imagen hiperespectral $\mathbf{X}$, el conjunto de etiquetas $\mathbf{T}$ y los parámetros C y γ definidos por el usuario

▷ Fase de preprocesado
1: <Normalización del conjunto de datos> ▷ SM + GM
2: Selección de los puntos de entrenamiento ($\mathbf{X}_{\text{train}}$)
3: <Procesado de la matriz de objetivos ($\mathbf{T}_{\text{train}}$) > ▷ GM
▷ Cálculo de la función *kernel*
4: <Calcular $Kernel\_2$ (*Eq.* 4) > ▷ GM
5: <Calcular $\alpha = (\mathbf{I}/C + Kernel\_2)^{-1}\mathbf{T}_{\text{train}}$ > ▷ GM
6: <Preparar los datos de test ($\mathbf{X}_{\text{test}}$)> ▷ GM
7: <Calcular $Kernel\_1$ (*Eq.* 4) > ▷ GM
▷ Clasificación.
8: <Calcular $\alpha Kernel\_1$ > ▷ GM

B. Clasificacion basada en KELM sobre GPU

El algoritmo KELM se compone de tres fases principales: preprocesado, cálculo de las funciones *kernel* y clasificación. El pseudocódigo del Algoritmo 1 nos muestra la implementación en GPU tal y como se explica en la sección II-A.

En nuestro caso, el algoritmo KELM es parte del esquema KELM-EMP, siendo los datos de entrada el resultado de la unión y la normalización del conjunto de datos hiperespectrales y el EMP. En la fase de unión, los datos del perfil obtenidos a través del algoritmo EMP son multiplicados por un peso determinado. Después, en la normalización (línea 1 del pseudocódigo), a todos los puntos del conjunto de datos hiperespectrales se le resta el mínimo de dicho conjunto mientras que en el conjunto de datos del EMP la normalización se realiza por bandas, restando el mínimo de cada banda a los elementos de dicha banda. El *kernel* utilizado para el cálculo del mínimo utiliza memoria compartida y evita el conflicto de bancos, reduciendo el tiempo de ejecución.

Dado que el ELM y el KELM son algoritmos de aprendizaje supervisado, la selección de los píxeles de entrenamiento en el mapa de referencia se hace de forma aleatoria, escalando posteriormente los valores correspondientes del conjunto de datos entre [0:1] y almacenándolos en la matriz $\mathbf{X}_{\text{train}}$ (línea 2 del pseudocódigo). Finalmente se procesa la matriz de entrenamiento, donde cada fila representa una muestra y cada columna una clase, y para la que el valor 1 indica que pertenece a la clase y el valor 0 que no pertenece a la clase (línea 3 del pseudocódigo).

La segunda fase (ejecución de las funciones *kernel*) comienza con la primera función de *kernel*, donde $K(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\gamma||\boldsymbol{u} - \boldsymbol{v}||^2)$ y $\boldsymbol{u}, \boldsymbol{v}$ pertenece al conjunto de datos de entrenamiento. En primer lugar calculamos la matriz $\mathbf{X}_{\text{train}}(\mathbf{X}_{\text{train}})^{\text{T}}$, que contiene todos los productos entre dos elementos del conjunto de entrenamiento. Con la matriz anterior y el parámetro γ, definido por el usuario, calculamos la matriz $Kernel\_2$. El siguiente paso es generar una matriz identidad donde todos sus elementos son divididos por el parámetro C, también definido por el usuario. A continuación sumamos la matriz identidad modificada y la matriz $Kernel\_2$. Por último usamos la función de MAGMA llamada *magma_dgesv_gpu* para resolver el sistema de ecuaciones lineales y obtener α, donde:

$$\alpha = (\mathbf{I}/C + Kernel\_2)^{-1}\mathbf{T}_{\text{train}}.$$

La segunda fase continua con el cálculo de la otra función *kernel* del mismo modo que la anterior, pero esta vez $\boldsymbol{u}$ pertenece al conjunto de datos de test y $\boldsymbol{v}$ al de entrenamiento. Primero escalamos entre [0:1] el conjunto de datos de test y lo almacenamos en la matriz $\mathbf{X}_{\text{test}}$. A continuación calculamos la función *kernel* y creamos una matriz para almacenar $\mathbf{X}_{\text{test}}\mathbf{X}_{\text{train}}$, un vector para almacenar la diagonal de $\mathbf{X}_{\text{test}}\mathbf{X}_{\text{test}}$ y otro vector para almacenar la diagonal de $\mathbf{X}_{\text{train}}\mathbf{X}_{\text{train}}$. Por último calculamos la función *kernel* con todos los elementos de $\mathbf{X}_{\text{test}}$ y todos los elementos de $\mathbf{X}_{\text{train}}$.

En la última fase del algoritmo KELM únicamente necesitamos calcular el producto de $Kernel\_1$ por α para obtener la clasificación final.

C. Extracción de características sobre GPU

Para reducir la dimensionalidad del conjunto de datos se utiliza el algoritmo EVD (EVD-PCA) explicado en la sección II-B.

El algoritmo 2 incluye el pseudocódigo del algoritmo EVD-PCA en GPU. Suponemos que la matriz $\mathbf{X}$ contiene el conjunto de datos y que este se encuentra almacenado en la memoria global de la GPU. Es necesario preprocesar el conjunto de datos, es decir, deben estar centrados. Para ello restamos a cada píxel la suma de todos los píxeles de su banda dividida por el número de píxeles de dicha banda (lineas 1-3 del Algoritmo 2). En este paso, además del uso de memoria compartida para la eficiencia en el acceso a los datos, también se utiliza la función de CUBLAS *cublasSgemv*.

Una vez que la matriz $\mathbf{X}$ contiene los datos centrados, comienza la fase del PCA. En primer lugar se calcula la matriz de correlación de la matriz $\mathbf{X}$ usando la función de CUBLAS *cublasSsyrk* (linea 4). Esta función realiza la operación $\mathbf{X}\mathbf{X}^{\mathrm{T}}$, pero debido a que el resultado es simétrico, *cublasSsyrk* solo devuelve la matriz superior. El siguiente paso consistirá en completar la matriz inferior del cálculo anterior con los datos de la matriz superior (line 5).

El algoritmo PCA continua con el cálculo de

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T,$$

usando la función de CULA *culaDeviceSgesvd*, donde los elementos de la diagonal de $\mathbf{S}$ son los valores singulares de $\mathbf{X}\mathbf{X}^T$ y las primeras columnas de $\mathbf{U}$ y $\mathbf{V}$ son los vectores singulares derechos e izquierdos de $\mathbf{X}\mathbf{X}^T$ respectivamente (linea 6). El último paso es la obtención de los PCs mediante la función de CUBLAS *cublasSgemm*, que realiza el producto entre $\mathbf{X}$ y $\mathbf{V}$ (linea 7).

Algoritmo 2 Algoritmo EVD-PCA sobre GPU

Entrada: El conjunto de datos $\mathbf{X}$ se almacena inicialmente en la memoria global

▷ Fase de centrado del conjunto de datos

1: <Crear un vector de unos> ▷ GM
2: Obtener la suma de todos los píxeles en cada banda del conjunto de datos ▷ SM + GM
3: <Centrado del conjunto de datos> ▷ SM

▷ Fase PCA

4: Calcular la matriz de covarianza $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ del conjunto de datos centrados ▷ SM + GM
5: <Completar la matriz triangular inferior de $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ > ▷ GM
6: Calcular la matriz de auto-vectores $\mathbf{V}$ ▷ SM + GM
7: Obtener los PCs mediante $\mathbf{X}\mathbf{V}$ ▷ SM + GM

D. EMP sobre GPU

El EMP es el conjunto de MPs creado por medio de operaciones de apertura y cierre por reconstrucción sobre cada una de las bandas obtenidas en la fase del PCA.

La implementación realizada se basa en la propagación de bloques asíncronos [35], donde se realizan múltiples barridos en ambas direcciones al mismo tiempo. La principal ventaja de esta implementación asíncrona en GPU [36], para la reconstrucción morfológica (que consisten en actualizaciones intra e inter-bloque), es que se realizan tantas actualizaciones como sean posibles con los datos disponibles en la memoria compartida antes de realizar una sincronización entre los hilos de los bloques. Los datos actualizados dentro de un bloque de memoria compartida se pueden reutilizar, lo cual es mucho más rápido que las actualizaciones de memoria global [37]. En esta versión asíncrona, el número de sincronizaciones globales se reduce en comparación con la versión síncrona del algoritmo [35].

IV. Resultados

Esta sección muestra los resultados experimentales obtenidos por los clasificadores. Los algoritmos propuestos han sido evaluados en un PC con un quadcore Intel Xeon E5-2609v2 a 2.5 GHz y 15 GB de RAM. El código se ha compilado usando gcc version 4.8.4 con soporte OpenMP (OMP) 3.0 bajo Linux y 4 hilos. Se ha utilizado la librería OPENBLAS [38] para acelerar las operaciones de los distintos algoritmos. En cuanto a la implementación en GPU, se ha utilizado una tarjeta NVIDIA GrForce GTX Titan con 14 SMXs de 192 procesadores CUDA cada uno. Se usó la versión 7.5 del conjunto de herramientas CUDA para Linux.

La precisión de los resultados está expresada en porcentaje en términos de precisión total (OA), que es el porcentaje de píxeles clasificados correctamente, precisión media (AA), que se calcula como la media de las precisiones de las diferentes clases, y el coeficiente kappa [39], que es el porcentaje de acierto corregido por la cantidad de acierto que sería de esperar debido al azar.

Los resultados de rendimiento se expresan en términos de tiempo de ejecución (en segundos) y aceleración. Los resultados son la media de 100 ejecuciones. Los resultados se muestran como valor medio y desviación estándar de las 100 ejecuciones. El tiempo de ejecución para los códigos en GPU no incluye la transferencia de datos CPU-GPU, solo los tiempos de computación.

Los algoritmos han sido probados sobre tres imágenes de teledetección: Una imagen obtenida por el sensor ROSIS de 103 bandas de la Universidad de

Pavia (Pavia Univ.), con una dimensión espacial de 610×340 píxeles, una imagen obtenida por el sensor AVIRIS de 220 bandas y 145×145 píxeles tomada sobre el Noroeste de Indiana (Indian Pines) y una imagen obtenida mediante AVIRIS de 204 bandas y 512×217 píxeles del valle de Salinas, California (Salinas).

Para poder realizar la comparación, el número de muestras de entrenamiento es el mismo que en el trabajo de referencia [27]. La tabla I muestra información de las imágenes utilizadas, incluyendo sus dimensiones y el número y porcentaje de muestras de entrenamiento. La selección de las muestras de entrenamiento se ha realizado de forma aleatoria, además de que dichas muestras no son tenidas en cuenta cuando se evalúa la precisión. El número de neuronas de la capa oculta empleadas en el ELM es de 1000 para Pavia Univ., 300 para Indian Pines y 350 para Salinas en todos los casos [27].

Para el cálculo del EMP, los mejores resultados fueron los obtenidos considerando 7 componentes principales que se extrajeron previamente mediante análisis en componentes principales. Se consideran además 7 operaciones de apertura y cierre por reconstrucción, usando discos de radio incremental (el tamaño de los elementos estructurales es de 3, 5, 9, 13, 17, 21 y 25, dando lugar a un total de 105 componentes). El mapeo de características compuestas es de tipo concatenación ponderada, es decir, se le da un peso a la información espacial y otro a la espectral. El peso para la característica espectral (k_w) es 1, mientras que el peso para la característica espacial (k_s) se ha ajustado para cada imagen por prueba y error en valores desde 0.5 hasta 10. Los mejores resultados se obtuvieron con k_s 1, 5 y 3 para la Universidad de Pavia, Indian Pines y Salinas respectivamente. Tal y como se menciona en la Sección II-A, con el fin de maximizar la capacidad de discriminación del clasificador, cada conjunto de datos χ fue ajustado al rango $[0, max(\chi) - min(\chi)]$. Este proceso se llevó a cabo de forma independiente para los datos hiperespectrales y para cada uno de los componentes individuales del EMP. Finalmente, después de la concatenación de las dos características, todo el conjunto de datos se escaló dentro del rango de [0,1].

En primer lugar hemos realizado una comparación entre el esquema original basado en ELM (ELM-EMP) y la versión que incluye una post-regularización espacial (ELM-EMP-S). Esta post-regularización consiste en asignar a cada píxel la clase mayoritaria de entre sus vecinos. En nuestro caso se han tenido en cuenta a los 8 vecinos de cada píxel. La tabla II muestra que esta técnica mejora los resultados de precisión de la clasificación con un bajo coste en el tiempo de ejecución, pasando de un valor para OA de 92.81% a 95.05% en el caso de la imagen de Indian Pines y de un 99.65% a 99.82% en el caso de la imagen de Pavia Univ.. Por lo tanto, a partir de ahora los esquemas aquí estudiados incluyen dicha post-regularización.

En la tabla III se comparan, en términos de precisión de la clasificación y tiempo de ejecución en GPU, los esquemas propuestos, ELM-EMP-S y KELM-EMP-S (KELM-EMP con regularización), con otros esquemas proyectados sobre GPU y disponibles en la literatura. Todos los experimentos han sido realizados en las mismas condiciones experimentales. Algunos de esos esquemas, como ELM+wat [9], usan ELM como clasificador mientras que otros, como SVM+wat [14] y WT-EMP [18], utilizan SVM, que es un clasificador bastante utilizado en la literatura. Todos los experimentos mostrados en la tabla se han realizado bajo las mismas condiciones experimentales descritas al inicio de la sección. En la tabla, los mejores resultados se muestran en negrita. El esquema ELM implementado en [9] realiza una regularización después de la clasificación, del mismo modo que los esquemas propuestos ELM-EMP-S y KELM-EMP-S. El esquema ELM+wat [9] combina la información proporcionada por el esquema ELM con información espacial obtenida a través de un algoritmo watershed. Los esquemas SVM [14] y SVM+wat [14] usan SVM como clasificador, pero el segundo añade información espacial usando un algoritmo de segmentación *watershed*. Por último, en el esquema WT-EMP [18] se crea un EMP a partir de las características extraídas mediante wavelets que se combina con la imagen original, sin ruido, en un vector que es procesado por el clasificador SVM.

En la tabla III podemos ver que los esquemas propuestos en este artículo (ELM-EMP-S y KELM-EMP-S) obtienen resultados muy cercanos en términos de precisión de la clasificación, siendo el esquema KELM-EMP-S ligeramente superior en las imágenes de Pavia Univ. e Indian Pines.

En las tablas IV y V se muestran los tiempos de ejecución y aceleraciones de las versiones GPU sobre las versiones OpenMP optimizadas para los esquemas propuestos (ELM-EMP-S y KELM-EMP-S). Ambos esquemas incluyen post-regularización. En el caso del esquema KELM-EMP-S, los valores para los parámetros definidos por el usuario C y γ para las tres imágenes hiperespectrales son: $C = 10^8$ y $\gamma = 10$ para Pavia Univ., $C = 10^6$ y $\gamma = 10$ para Indian Pines y $C = 10^8$ y $\gamma = 12$ para Salinas. Tal y como se mostraba en la tabla III, las precisiones alcanzadas por el esquema KELM-EMP-S para las dos primeras imágenes (Pavia Univ. e Indian Pines) son mejores que las obtenidas por el esquema ELM-EMP-S, sin embargo, podemos observar que requiere más tiempo para obtener el resultado. En cuanto a las aceleraciones, se han alcanzado valores de 4.92× para la imagen de Pavia Univ., mientras que para la imagen de Indian Pines el valor obtenido es una aceleración de 1.28×. Esto se debe a que la imagen de Indian Pines es 4.5 veces más pequeña que las otras dos utilizadas en este artículo y, por lo tanto, el número de hilos necesarios es también menor, lo que dificulta ocultar el coste en las transferencias de datos y, por tanto, aumenta el tiempo de ejecución.

Basada en los resultados de la clasificación

TABLA I
INFORMACIÓN SOBRE LAS IMÁGENES DE TELEDETECCIÓN UTILIZADAS PARA EL TEST.

	sensor	# clases	dimensiones	# muestras	# muestras de entrenamiento
Pavia Univ.	ROSIS	9	610×340×103	42776	3921 (9.17%)
Indian Pines	AVIRIS	16	145×145×220	10249	625 (6.10%)
Salinas	AVIRIS	16	512×217×204	54129	1076 (1.99%)

TABLA II
COMPARATIVA, EN TÉRMINOS DE PRECISIÓN DE LA CLASIFICACIÓN Y TIEMPOS DE EJECUCIÓN EN GPU, ENTRE EL ESQUEMA ELM-EMP Y ELM-EMP-S.

Esquema	Pavia Univ.				Indian Pines			
	OA(%)	AA(%)	Kappa(%)	t(s)	OA(%)	AA(%)	Kappa(%)	t(s)
ELM-EMP	99.65 ±0.08	99.60 ±0.06	99.52 ±0.11	2.29 ±0.08	92.81 ±0.76	95.02 ±0.65	91.77 ±0.86	0.72 ±0.13
ELM-EMP-S	99.82 ±0.09	99.76 ±0.05	99.75 ±0.13	2.30 ±0.10	95.05 ±0.74	96.44 ±0.56	94.32 ±0.85	0.72 ±0.12

TABLA III
PRECISIONES Y TIEMPOS DE EJECUCIÓN EN GPU PARA LAS DISTINTAS IMÁGENES. LAS DESVIACIONES ESTÁNDAR VAN SEGUIDAS DEL SÍMBOLO ±. LAS MEJORES PRECISIONES SE MUESTRAN EN NEGRITA.

Esquema	Pavia Univ.			Indian Pines			Salinas		
	OA(%)	AA(%)	Kappa(%)	OA(%)	AA(%)	Kappa(%)	OA(%)	AA(%)	Kappa(%)
KELM-EMP-S	99.83 ±0.07	99.79 ±0.04	99.77 ±0.09	95.39 ±0.80	96.82 ±0.63	94.72 ±0.92	99.16 ±0.18	99.05 ±0.20	99.06 ±0.20
ELM-EMP-S	99.82 ±0.09	99.76 ±0.05	99.75 ±0.13	95.05 ±0.74	96.44 ±0.56	94.32 ±0.85	99.21 ±0.25	99.09 ±0.18	99.12 ±0.28
ELM	96.94 ±0.31	96.13 ±0.34	95.83 ±0.43	82.25 ±1.57	89.85 ±0.98	79.93 ±1.73	93.63 ±0.29	96.38 ±0.25	92.89 ±0.32
ELM+wat	97.20 ±0.33	96.42 ±0.43	96.30 ±0.44	80.64 ±2.46	79.90 ±3.04	78.40 ±2.73	93.60 ±0.32	96.41 ±0.27	92.86 ±0.36
SVM	79.43 ±0.96	86.62 ±0.86	76.70 ±1.05	79.43 ±0.96	86.62 ±0.86	76.70 ±1.05	88.45 ±0.47	92.37 ±0.50	87.08 ±0.54
SVM+wat	96.18 ±0.52	96.58 ±0.25	94.81 ±0.70	87.65 ±1.26	91.41 ±1.78	85.96 ±1.42	- -	- -	- -
WT-EMP	98.70 ±0.18	98.72 ±0.13	98.22 ±0.24	89.73 ±1.14	94.12 ±0.67	88.28 ±0.073	- -	- -	- -

TABLA IV
TIEMPOS DE EJECUCIÓN Y ACELERACIÓN PARA EL ESQUEMA ELM-EMP-S.

		PCA (s)	EMP (s)	ELM (s)	Total (s)
Pavia Univ.	CPU	0.43	14.93	14.97	30.33
	OMP	0.28	4.35	6.70	11.33
	GPU	0.05	1.08	1.17	2.30
	Speedup GPU vs. OMP: 4.92×				
Indian Pines	CPU	0.13	1.52	0.82	2.47
	OMP	0.07	0.44	0.4	0.91
	GPU	0.06	0.58	0.07	0.71
	Speedup GPU vs. OMP: 1.28×				
Salinas	CPU	0.60	8	3.96	12.56
	OMP	0.31	2.30	2.06	4.67
	GPU	0.07	0.70	0.22	0.99
	Speedup GPU vs. OMP: 4.71×				

obtenidos por cada esquema de clasificación también realizamos la prueba estándar de McNemar [40], [26]. Se emplea para verificar la relevancia estadística de las diferencias de precisión entre pares de esquemas. La tabla VI presenta los resultados del test, donde cada valor Z compara el clasificador KELM-EMP-S con cada uno de los métodos mostrados en la tabla. La diferencia en precisión entre cada par de clasifi-

TABLA V
TIEMPOS DE EJECUCIÓN Y ACELERACIÓN PARA EL ESQUEMA KELM-EMP-S.

		PCA (s)	EMP (s)	KELM (s)	Total (s)
Pavia Univ.	CPU	0.43	14.93	51.61	66.97
	OMP	0.28	4.35	14.92	19.55
	GPU	0.05	1.08	2.80	3.93
	Speedup GPU vs. OMP: 4.97×				
Indian Pines	CPU	0.13	1.52	1.22	2.87
	OMP	0.07	0.44	0.56	1.07
	GPU	0.06	0.58	0.15	0.79
	Speedup GPU vs. OMP: 1.35×				
Salinas	CPU	0.60	8.00	8.92	17.52
	OMP	0.31	2.30	2.81	5.42
	GPU	0.07	0.70	0.62	1.39
	Speedup GPU vs. OMP: 3.90×				

cadores es vista como significativa con un nivel de confianza del 95% si $|Z| > 1.96$, y con un nivel de confianza del 99 % si $|Z| > 2.58$. Un valor positivo de Z indica que el primer clasificador supera al segundo ($Z > 0$). Podemos observar que sólo en un caso, cuando comparamos los esquemas KELM-EMP-S y ELM-EMP-S, las pequeñas diferencias en la precisión no son estadísticamente relevantes, lo que indica que ambos sistemas de clasificación son similares en términos de precisión.

V. CONCLUSION

En este artículo se presenta un esquema de clasificación para imágenes hiperespectrales de teledetección basado en *kernels* llamado KELM-EMP desarrollado a partir de un esquema publicado anteriormente (ELM-EMP) por los autores[27]. Estos esquemas realizan una reducción de la dimensionalidad de la imagen hiperespectral mediante análisis en componentes principales, seguida de la construcción de un perfil morfológico extendido. A continuación se utiliza una concatenación ponderada para combinar los datos del perfil con los datos de la imagen hiperespectral sobre los que se ejecuta el clasificador. En este artículo se propone también añadir un proceso de regularización espacial al final de la clasificación, de modo que se eliminan clasificaciones erróneas de píxeles aislados. Una última aportación de este artículo es proponer proyecciones de los algoritmos ELM-EMP-S y KELM-EMP-S sobre GPU, utilizando CUDA y aplicando técnicas de mejora como la ejecución asíncrona de algunas secciones del algoritmo.

Los resultados de precisión de la clasificación muestran que el esquema basado en KELM con regularización espacial (KELM-EMP-S) obtiene precisiones altas alcanzando un valor de 99.83% de precisión media para la imagen de Pavia Univ. En cuanto a la ejecución en GPU se consiguen aceleraciones altas en comparación con la ejecución de las versiones OpenMP alcanzando un speedup de 4.97× para la imagen de Pavia Univ.

AGRADECIMIENTOS

Agradecemos al profesor Chen Chen, de la Universidad de Texas, por su colaboración compartiendo su código MATLAB con nosotros. Este trabajo fue financiado en parte por el Ministerio de Ciencia e Innovación, Gobierno de España, cofinanciado con fondos FEDER a través de la Unión Europea bajo el contrato TIN 2013-41129-P, y por la Xunta de Galicia, mediante el programa de consolidación y estructuración de unidades de investigación competitivas ref. 2014/008.

REFERENCIAS

[1] David Landgrebe, "Hyperspectral image data analysis," *Signal Processing Magazine, IEEE*, vol. 19, no. 1, pp. 17–28, 2002.
[2] Mathieu Fauvel, Yuliya Tarabalka, Jón Atli Benediktsson, Jocelyn Chanussot, and James C Tilton, "Advances in spectral–spatial classification of hyperspectral images," *Proceedings of the IEEE*, vol. 101, no. 3, pp. 652–675, 2013.
[3] Guang-Bin Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.
[4] Shinichi Tamura and Masahiko Tateishi, "Capabilities of a four–layered feedforward neural network: four layers versus three," *Neural Networks, IEEE Transactions on*, vol. 8, no. 2, pp. 251–255, 1997.
[5] Guang-Bin Huang, "Learning capability and storage capacity of two–hidden–layer feedforward networks," *Neural Networks, IEEE Transactions on*, vol. 14, no. 2, pp. 274–281, 2003.
[6] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.
[7] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang, "Extreme learning machine for regression and multiclass classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 513–529, 2012.
[8] Mark Van Heeswijk, Yoan Miche, Erkki Oja, and Amaury Lendasse, "Gpu-accelerated and parallelized elm ensembles for large-scale regression," *Neurocomputing*, vol. 74, no. 16, pp. 2430–2437, 2011.
[9] Javier López-Fandiño, Pablo Quesada-Barriuso, Dora B Heras, and Francisco Arguello, "Efficient elm-based techniques for the classification of hyperspectral remote sensing images on commodity gpus," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 8, no. 6, pp. 2884–2893, 2015.
[10] Antonio Plaza, Jón Atli Benediktsson, Joseph W. Boardman, Jason Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Mathieu Fauvel, Paolo Gamba, Anthony Gualtieri, et al., "Recent advances in techniques

TABLA VI
RELEVANCIA ESTADÍSTICA DE LAS DIFERENCIAS EN LA PRECISIÓN DE CLASIFICACIÓN ENTRE KELM-EMP-S Y LOS ESQUEMAS PROPUESTOS EN LA TABLA.

		ELM-EMP-S	ELM	ELM+wat	SVM	SVM+wat	WT-EMP
Z	Pavia Univ.	0.615	33.94	32.31	64.30	38.25	19.72
	Indian Pines	1.55	31.27	26.86	89.76	89.52	88.76
	Salinas	-1.48	50.99	74.32	51.09	-	-

for hyperspectral image processing," *Remote sensing of environment*, vol. 113, pp. S110–S122, 2009.
[11] Mauro Dalla Mura, Alberto Villa, Jón Atli Benediktsson, Jocelyn Chanussot, and Lorenzo Bruzzone, "Classification of hyperspectral images by using extended morphological attribute profiles and independent component analysis," *Geoscience and Remote Sensing Letters, IEEE*, vol. 8, no. 3, pp. 542–546, 2011.
[12] J.A. Palmason, J.A. Benediktsson, J.R. Sveinsson, and J. Chanussot, "Classification of hyperspectral data from urban areas using morpholgical preprocessing and independent component analysis," in *International Geoscience And Remote Sensing Symposium*, 2005, vol. 1, pp. 176–179.
[13] Yun Zhang, Xuezhi Feng, and Xinghua Le, "Segmentation on multispectral remote sensing image using watershed transformation," in *Image and Signal Processing, 2008. CISP'08. Congress on*. IEEE, 2008, vol. 4, pp. 773–777.
[14] Yuliya Tarabalka, Jocelyn Chanussot, and Jón Atli Benediktsson, "Segmentation and classification of hyperspectral images using watershed transformation," *Pattern Recognition*, vol. 43, no. 7, pp. 2367–2379, 2010.
[15] Blanca Priego, Francisco Bellas, and Richard J. Duro, "ECAS-II: A hybrid algorithm for the construction of multidimensional image segmenters," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.
[16] Yuliya Tarabalka, Jón Atli Benediktsson, and Jocelyn Chanussot, "Spectral–spatial classification of hyperspectral imagery based on partitional clustering techniques," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 47, no. 8, pp. 2973–2987, 2009.
[17] Xudong Kang, Shutao Li, and Jón Atli Benediktsson, "Spectral–spatial hyperspectral image classification with edge-preserving filtering," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 52, no. 5, pp. 2666–2677, 2014.
[18] Pablo Quesada-Barriuso, Francisco Argüello, and Dora B. Heras, "Spectral–spatial classification of hyperspectral images using wavelets and extended morphological profiles," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 7, no. 4, pp. 1177–1185, 2014.
[19] Martino Pesaresi and Jón Atli Benediktsson, "A new approach for the morphological segmentation of high–resolution satellite imagery," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 39, no. 2, pp. 309–320, 2001.
[20] Pierre Soille and Martino Pesaresi, "Advances in mathematical morphology applied to geoscience and remote sensing," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 40, no. 9, pp. 2042–2055, 2002.
[21] Jón Atli Benediktsson, Martino Pesaresi, and Kolbeinn Amason, "Classification and feature extraction for remote sensing images from urban areas based on morphological transformations," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 41, no. 9, pp. 1940–1949, 2003.
[22] Prashanth Reddy Marpu, Mattia Pedergnana, Mauro Dalla Mura, Stijn Peeters, Jón Atli Benediktsson, and Lorenzo Bruzzone, "Classification of hyperspectral data using extended attribute profiles based on supervised and unsupervised feature extraction techniques," *International Journal of Image and Data Fusion*, vol. 3, no. 3, pp. 269–298, 2012.
[23] Jón Atli Benediktsson, Jón Aevar Palmason, and Johannes R. Sveinsson, "Classification of hyperspectral data from urban areas based on extended morphological profiles," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 43, no. 3, pp. 480–491, 2005.
[24] Giorgio Licciardi, Prashanth Reddy Marpu, Jón Atli Benediktsson, and Jocelyn Chanussot, "Extended morphological profiles using auto–associative neural networks for hyperspectral data classification," in *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2011 3rd Workshop on*. IEEE, 2011, pp. 1–4.
[25] Mauro Dalla Mura, Jón Atli Benediktsson, Björn Waske, and Lorenzo Bruzzone, "Morphological attribute profiles for the analysis of very high resolution images," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 48, no. 10, pp. 3747–3762, 2010.
[26] Chen Chen, Wei Li, Hongjun Su, and Kui Liu, "Spectral-spatial classification of hyperspectral image based on kernel extreme learning machine," *Remote Sensing*, vol. 6, no. 6, pp. 5795–5814, 2014.
[27] Francisco Argüello and Dora B. Heras, "ELM-based spectral-spatial classification of hyperspectral images using extended morphological profiles and composite feature mappings," *International Journal of Remote Sensing*, vol. 36, no. 2, pp. 645–664, 2015.
[28] Guang-Bin Huang, Dian Hui Wang, and Yuan Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
[29] Hervé Abdi and Lynne J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.
[30] Martino Pesaresi and Ioannis Kanellopoulos, "Detection of urban features using morphological based segmentation and very high resolution remotely sensed data," in *Machine Vision and Advanced Image Processing in Remote Sensing*, pp. 271–284. Springer, 1999.
[31] Nvidia, "Nvidia kepler gk110 architecture," 2012.
[32] MAGMA, "Matrix algebra on gpu and multicore architectures," 2015.
[33] Nvidia, "Cula tools," 2015.
[34] Nvidia, "Cuda toolkit documentation: Cublas," 2015.
[35] Pablo Quesada-Barriuso, Dora B. Heras, and Francisco Argüello, "Efficient 2D and 3D watershed on graphics processing unit: block-asynchronous approaches based on cellular automata," *Computers & Electrical Engineering*, vol. 39, no. 8, pp. 2638–2655, 2013.
[36] Pablo Quesada-Barriuso, Francisco Argüello, Dora B. Heras, and Jón Atli Benediktsson, "Wavelet–based classification of hyperspectral images using extended morphological profiles on graphics processing units," 2014.
[37] D. Kirk and W. Hwu Wen-mei, *Programming massively parallel processors: a hands–on approach*, vol. 1, Elsevier Science, Applications of GPU Computing Series, 2010.
[38] OpenBLAS, "An optimized blas library," 2015.
[39] John A. Richards, *Remote sensing digital image analysis*, vol. 3, Springer, 1999.
[40] Giles M Foody, "Thematic map comparison," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 5, pp. 627–633, 2004.

