Aprendizaje Profundo Localmente Ponderado para el Reconocimiento de Emociones

por

Maria Fernanda Hernández Luquin

Tesis sometida como requerimiento parcial para obtener el grado de Doctor en

Ciencias, en el área de Ciencias Computacionales

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
Santa María de Tonantzintla, Puebla, CP 72840

Mayo 2024

Director:

Dr. Hugo Jair Escalante Balderas
Coordinación de Ciencias Computacionales INAOE

Índice general

Índice de figuras

Índice de Tablas

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo otorgado a través de la beca No. CVU 664689.

A los proyectos FORDECYT-PRONACES/2275/2020 y Ciencia Básica-S-26314.

A mi asesor el Dr. Hugo Escalate por compartirme sus conocimientos, por su tiempo, su ayuda y su paciencia que me brindó a lo largo de mi doctorado.

A mis sinodales por sus observaciones y comentarios.

Dedicatoria

A Dios por permitirme poder alcanzar mi doctorado.

A mis hijos José y Carolina que son el mejor regalo de mi vida.

A mi amado esposo Leonardo por ser un gran hombre y padre.

A mis papás, Rosalba y Fernando.

A mis hermanos Paola y Fernando.

Resumen

En este trabajo de tesis se presenta un esquema llamado *aprendizaje profundo localmente ponderado* o también llamado *Locally Weighted Deep Learning (LWDL)*. LWDL trata de adaptar una técnica de aprendizaje local en un modelo de aprendizaje profundo (*deep learning*) enfocado a tareas de clasificación de imágenes. La finalidad del LWDL es adaptar el aprendizaje local en la fase predictiva del modelo profundo para mejorar el desempeño en la tarea de clasificación. La idea se enfoca en tomar las mejores ventajas de los dos enfoques del aprendizaje profundo y el aprendizaje local para combinarlos y crear un modelo profundo más robusto.

Los modelos locales tiene la capacidad de crear funciones de aproximación que se basan en considerar únicamente aquellas instancias del conjunto de entrenamiento que son relevantes para una instancia de punto de consulta. Para el caso del aprendizaje profundo, la ventaja más atractiva es que son modelos complejos capaces de extraer de manera automática las características de los datos de entrada y de manera conjunta llevar el proceso de clasificación. Uno de los enfoques del aprendizaje profundo que particularmente ha tenido éxito en el procesamiento de imágenes son las redes neuronales convolucionales (CNN). Se considera que las CNNs contienen dos fases: la extracción de características y la fase de clasificación.

La solución que proponemos para mejorar la fase predictiva del modelo es adaptar un método incremental basado en el aprendizaje local para que la fase predictiva del modelo profundo construya su función de decisión basándose en información local. La razón es que un método incremental cuenta con fase de entrenamiento y prueba; así la función de decisión se construye basándose en aquellas instancias relacionadas entre sí.

El LWDL se evalúa en conjuntos de datos de referencia usados para el reconocimiento de objetos, digitos y emociones. Los resultados alcanzados demuestran que el LWDL mejora el rendimiento del modelo profundo en tareas de clasificación de imágenes y reconocimiento de emociones; qué en comparativa con el estado del arte, se demostró tener resultados competitivos contra modelos complejos usados para el reconocimiento de emociones en imágenes.

Abstract

In this thesis work, we present a schema called Locally Weighted Deep Learning (LWDL). It integrates a local learning technique into a deep learning model. The aim is to adapt the local learning into the predictive phase in the model of a deep network to improve the performance of the classification task. The idea is to take the best advantages of the two approaches of deep learning and local learning to combine them and create a more robust deep model.

A local model has the capability to create a decision boundary for the training dataset considering only the instances nearest to the query point. This kind of learning is not applied to a deep model; usually, deep networks are based on global learning, but its most attractive advantage is that they are complex models capable of automatically extracting features from the input data. One of the models with great success in image processing is the convolutional neural networks (CNNs). We can establish that the CNN contains two phases: feature extraction and classification or the predictive phase. The feature extraction is formed by convolutional layers that apply convolution operation over the input image to obtain feature maps that are processed by the predictive model phase. The advantage lies in both phases are worked jointly. The predictive phase contains one or several fully connected dense layers and an output layer. The layers contain neurons that, like a classifier based on a multi-layer neural network, use the entire data set to build the decision function. This type of learning is called *global learning*.

Our solution consists to improve the predictive phase of the model by adapting an incremental method (based on local learning) on a deep learning model. The reason is that an incremental method has a training and testing phase; thus the decision function is built based on instances close to the query point.

We evaluated the model in the reference data set used for object recognition, digit recognition, and emotion recognition. The results achieved show that the LWDL improves the performance of the deep model for some image classification tasks and in its comparison with the state of the art, it showed competitive results against complex models used for emotion recognition in images.

1

Introducción

1.1. Antecedentes

El aprendizaje automático (*Machine Learning*, ML) es un subcampo de la inteligencia artificial que permite a las computadoras la habilidad de aprender sin ser explícitamente programadas, siendo presentados muchos ejemplos relevantes sobre una tarea específica, para posteriormente, construir modelos estadísticos capaces de hacer predicciones sobre nuevos ejemplos (Géron, 2017).

Los modelos convencionales de ML requieren una ingeniería y experiencia en el dominio para diseñar un extractor de características que transforme los datos sin procesar, de tal manera que las representaciones sean adecuadas o se obtenga un vector de características a partir del cual el modelo de aprendizaje automático podrá detectar o clasificar patrones (LeCun et al., 2015). Las técnicas de aprendizaje en ML se pueden categorizar como: supervisado, no-supervisado, semi-supervisado y aprendizaje por refuerzo. De manera general, el aprendizaje supervisado consiste en una técnica donde los modelos son entrenados con conjuntos de datos previamente etiquetados. Es decir, para una tarea específica se muestran varios ejemplos (instancias) etiquetados (asociados a una clase) para que el algoritmo construya un modelo que sea capaz de generalizar ante nuevas instancias. El aprendizaje no supervisado consiste en una técnica donde los modelos son capaces de encontrar patrones sobre los datos de entrada no etiquetados, con fines como la visualización de los datos, comprensión de los datos o para encontrar correlaciones entre ellos. Las técnicas que utilizan el aprendizaje no supervisado son la reducción de dimensionalidad y los algoritmos de agrupamiento (*clustering*) (Chollet, 2021).

En el aprendizaje supervisado podemos encontrar dos tipos de técnicas de aprendizaje: global y

local. El aprendizaje global [1] comprende las fases de entrenamiento y prueba. En la fase de entrenamiento el modelo se construye utilizando todo el conjunto de datos de entrenamiento para generar un modelo que posteriormente en la fase de prueba sea capaz de generalizar a instancias nunca antes vistas, el conjunto de prueba se usa para estimar el desempeño de generalización.

En contraste, el aprendizaje local hace predicciones a partir de un subconjunto de los datos de entrenamiento, creando un modelo aproximado a la instancia de consulta (ver la Figura 1.1). El aprendizaje local está constituido de tres diferentes enfoques y son: las representaciones locales, la selección local y el aprendizaje localmente ponderado. Una representación local implica que cada nuevo punto de datos afecta a un pequeño subconjunto de parámetros y el responder una consulta también implicaría un pequeño subconjunto de parámetros. Algunos ejemplos de representaciones locales son las tablas de búsqueda y clasificadores basados en ejemplos o prototipos (Atkeson et al., 1997b).

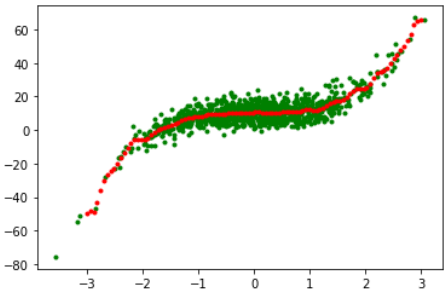


Figura 1.1: A partir de un conjunto de puntos se construye una gráfica de aproximación a través de la regresión lineal localmente ponderada.

La selección local se refiere a métodos que almacenan todos (o la mayoría) de los datos de entrenamiento en memoria y usan una función de distancia para determinar qué puntos almacenados son relevantes para la consulta. La función de la selección local es ubicar una única salida usando el vecino más cercano o usando un esquema de votación basado en la distancia. El aprendizaje localmente ponderado (también llamado *Locally Weighted Learning, LWL)* almacena explícitamente los datos de entrenamiento (al igual que los enfoques de selección local) y solo ajusta los parámetros a los datos de entrenamiento cuando se conoce una consulta. La característica crítica del aprendizaje localmente ponderado es que se utiliza un criterio de ponderación local con respecto a la ubicación de la consulta para ajustar algún tipo de modelo paramétrico a los datos. Aquí surge la confusión de las estructuras de modelos aparentemente

[1] Para explicaciones posteriores se establece el término de *aprendizaje global* al aprendizaje automático computacional que usa todos los datos disponibles para generar un modelo.

globales (por ejemplo, redes neuronales sigmoidales multicapa o las redes de función de base radial) se llaman modelos locales debido al criterio de entrenamiento que establece. El criterio establece que todos los datos pueden participar en la construcción del modelo local, siempre que los datos distantes importen menos que los datos cercanos. Por lo tanto, existen enfoques y representaciones globales que se pueden transformar en enfoques ponderados localmente utilizando un criterio de entrenamiento local (Atkeson et al., 1997a).

Las ventajas que presentan los métodos basados en el aprendizaje localmente ponderado (LWL) se refieren a que son flexibles e interpretables y tienen una configuración de parámetros simple que mejora el rendimiento en la predicción. Además, se pueden representar funciones no lineales con la ventaja de tener reglas simples en su entrenamiento como: el control de ajuste de parámetros, el suavizado, el rechazo de valores atípicos, entre otros. El proceso de modelado es fácil de entender y ajustar, debido a que se construye con puntos relacionados al punto de consulta. Cabe mencionar que una desventaja del método es que puede fallar en su generalización cuando se presenta una alta dimensionalidad en el espacio latente (Neruda and Kudová, 2005).

Por otro lado dentro del aprendizaje automático existe un subcampo llamado aprendizaje profundo (*Deep Learning, DL*) el cual se distingue por aprender características de los datos a través de múltiples capas de abstracción. Los datos sin procesar se ingresan en el nivel inferior y la salida deseada se produce en el nivel superior. El resultado del aprendizaje se obtiene través de muchos niveles de datos transformados, en el sentido que en cada capa, el algoritmo extrae automáticamente características visuales desde niveles inferiores que posteriormente son procesadas en niveles más profundos como se muestra en la Figura 1.2. En DL los modelos son construidos con técnicas de aprendizaje global, es decir, que los métodos supervisados en su fase de entrenamiento usan todo el conjunto de datos para crear un modelo que en la fase de predictiva será capaz de generalizar sobre instancias nunca antes vistas.

En este trabajo de investigación, se estudia la integración de un esquema de aprendizaje profundo localmente ponderado *(Locally Weighted Deep Learning, LWDL)*, que consiste en integrar una técnica de aprendizaje local en un enfoque de aprendizaje profundo. La finalidad de un modelo local es que se mejore la fase predictiva en un modelo de aprendizaje profundo. Un modelo local tiene la capacidad de crear funciones que agrupen los conjuntos de entrenamiento basándose en añadir pesos a aquellas instancias que sean relevantes para cierta clase. Una de las ventajas más atractivas de un método de DL es que es un método complejo capaz de extraer de manera automática las características, por lo tanto, éste tomará

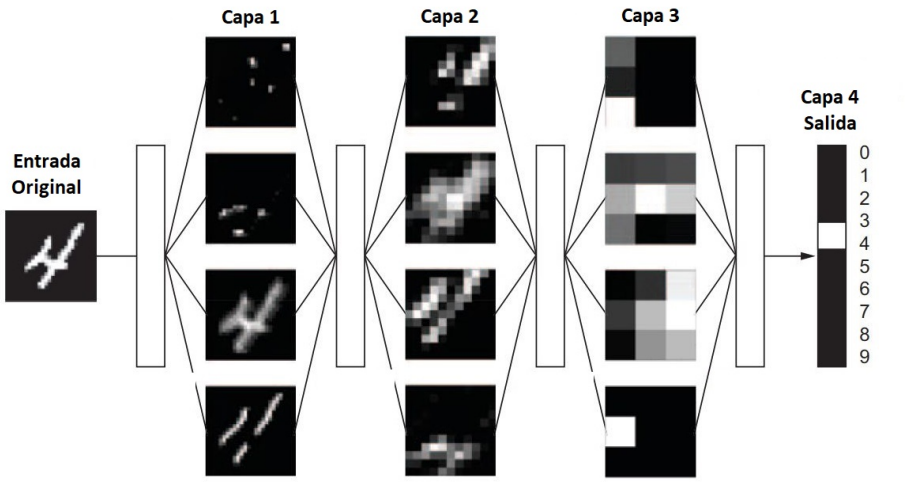


Figura 1.2: Representación de una red de aprendizaje profundo para la clasificación de dígitos. Figura reproducida de (Francois, 2017).

características importantes y relevantes del objeto que es analizado y las procesará en la fase predictiva; aquí surge la idea de añadir un esquema basado en métodos de LWL y se mejore la fase predictiva de los modelos. El modelo propuesto pretende ser benéfico en dominios dentro del alcance del LWL. Los dominios dentro del alance del LWL se consideran son: la clasificación con ejemplos de clase minoritaria o datos desbalanceados y clasificación de datos de grano fino.

Un hallazgo interesante de este trabajo es que el esquema LWDL demostró alcanzar resultados competitivos con respecto al estado del arte en el reconocimiento de emociones en conjuntos de datos con emociones básicas y compuestas; permitiendo demostrar que para el dominio de (*Emotion Recognition, ER*) el integrar el aprendizaje local, mejora el desempeño de los modelos en su fase predictiva con respecto a métodos convencionales del DL basados en aprendizaje global.

1.2. Motivación y justificación

Comúnmente, el aprendizaje localmente ponderado se aplica cuando se tienen problemas al construir clasificadores en los cuales es difícil separar las características entre clases, es decir, cuando sus atributos son muy similares entre sí (Yu and Grauman, 2017). El aprendizaje local es capaz de crear un modelo robusto e interpretable que generaliza correctamente basándose en instancias muy cercanas entre sí.

Basándonos en la naturaleza del aprendizaje local se considera que su uso puede ser benéfico y en

la tarea del reconocimiento de emociones (ER), mediante el reconocimiento de expresiones faciales (*Facial Expression Recognition, FER*) en imágenes. En el caso del ER en imágenes (siendo una de las aplicaciones consideradas dentro del alcance del LWL), se aborda usando diferentes modalidades que incluyen: voz, texto, signos vitales (EGG), reconocimiento de gestos, expresiones faciales e híbridos como se muestra en la Figura 2.13. Una de las formas más usadas es mediante el análisis de las expresiones faciales. Las expresiones faciales pueden ser capturadas de forma simple mediante una cámara para su análisis en modelos computacionales y su etiquetado se puede llevar a cabo por expertos simplemente con observar la imagen, pero esto nos da captura de emociones aparentes [2](ver Figura 1.4).

Figura 1.3: Diferentes fuentes para el reconocimiento de emociones.

El reconocimiento de emociones actualmente se aborda con enfoques del aprendizaje profundo obteniendo resultados sobresalientes. Los enfoques se dedican a resolver el ER sobre conjuntos de datos tomado bajo entornos controlados. Pero usualmente, este tipo de conjuntos de datos no son extensos. Esto es un problema en el momento de utilizar enfoques de DL, ya que los métodos tienden a generalizar mejor sobre conjuntos de datos extensos. De aquí surge la importancia de construir modelos de DL que sean capaces de reconocer con exactitud emociones en conjuntos de datos tomados bajo entornos no controlados. Además, se enfrenta el reto donde existe una variación como: cultural, étnica, racial, de género, de edad y de intensidad emocional. Las variaciones presentan retos incluso en la identificación de emociones

[2]Las emociones aparentes son aquellas que se perciben visualmente, de acuerdo a las convenciones establecidas en el reconocimiento de emociones, por lo que no es posible determinar si la emoción aparente es en realidad genuina.

Figura 1.4: Ejemplo de imágenes asociadas a diferentes emociones en la base de datos de expresiones faciales CK+.

entre los observadores (Matsumoto et al., 2002). Las variaciones motivan a pensar en que una técnica de aprendizaje local puede ser prometedora en ER, para crear clasificadores que sean robustos a la detección de esta variantes (como se ilustra en la Figura 1.5), ya que la localidad se encargará de agrupar aquellos conjuntos de rasgos que tengan similitudes entre sí, en función a su distancia más cercana.

El aprendizaje local almacena un subconjunto de datos de entrenamiento cercanos al punto de consulta para construir un modelo local capaz de hacer predicciones. La predicción de un valor o clase para una nueva instancia, se basa inicialmente en el cálculo de distancias o similitudes entre instancias relacionadas al entrenamiento. Internamente, el aprendizaje local hace transformaciones no lineales de los espacios, mediante el uso de aproximadores locales que se encargan del mapeo de entrada-salida de los datos cuando no son linealmente separables. Este tipo de aprendizaje tiene la ventaja que los clasificadores son robustos al ruido en los datos y evita la dificultad de la construcción de una función global sobre todo el conjunto de datos. Una desventaja en el aprendizaje local es que los modelos presentan problemas ante la alta dimensionalidad, haciendo intratable la aplicación de los métodos locales en altas dimensiones de los datos y derivando en problemas de regularización (Orr, 1995).

El presente trabajo de investigación propone un esquema que explote las ventajas del aprendizaje local, pero desarrollando técnicas que resuelvan la problemática de alta dimensionalidad al integrar en un enfoque de DL. Se planteó la hipótesis de que al aplicar un modelo de aprendizaje profundo que implique una nueva técnica de aprendizaje local puede mejorar el desempeño del modelo profundo en su fase predictiva. La técnica involucra el adaptar el concepto de aprendizaje local como lo hace los métodos incrementales en LWL sobre un modelo del aprendizaje profundo y resolver la problemática de la alta dimensionalidad que presentan los métodos de aprendizaje localmente ponderado.

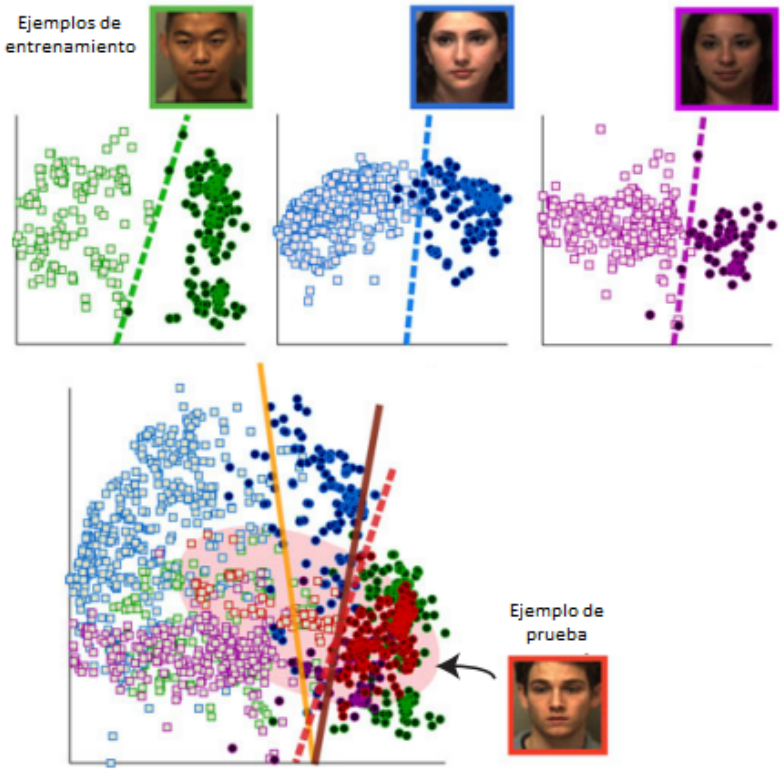


Figura 1.5: [Ejemplo de clasificador de expresiones faciales por sujeto. Gráfica de un clasificador basado en aprendizaje local donde separa adecuadamente las expresiones faciales casi perfectamente para cada sujeto, para mejorar el reconocimiento de emociones. Figura reproducida de (Chu et al., 2017b).

1.3. Planteamiento del problema

Los esquemas *Locally Weighted Deep Learning* actuales adaptan el concepto del aprendizaje local en un enfoque de aprendizaje profundo a través del uso de las unidades de función de base radial, como neuronas que determinan en la capa de salida la predicción de la clase. El esquema presenta una serie de problemáticas al integrar el aprendizaje local en un método profundo, tales están relacionadas a la construcción de las unidades RBF y se enfrentan a las siguientes condiciones:

- Los esquemas no generalizan correctamente cuando se enfrentan ante una alta dimensionalidad en el espacio latente.
- La selección de parámetros para la creación de las unidades RBF no muestran un indicio de ser los parámetros que mejor se ajusten en el esquema LWDL.

El problema se aborda con el desarrollo de modelos de aprendizaje profundo ponderado localmente creando técnicas nuevas que permitan reducir la alta dimensionalidad en el espacio latente y los aproximadores locales puedan mejorar la fase predictiva del modelo. El objetivo es superar las limitaciones de las soluciones ya existentes para obtener resultados que sean competitivos y alcancen un mejor desempeño comparado con el estado del arte en aplicaciones como el reconocimiento de emociones y dominios dentro del alcance del LWL. Los dominios dentro del alance del LWL se consideran son: la clasificación con ejemplos de clase minoritaria o datos desbalanceados y clasificación de datos de grano fino.

1.4. Preguntas de investigación

La propuesta de investigación plantea las siguientes preguntas:

1. ¿En qué dominios de aplicación los métodos de aprendizaje local convencionales tienen un mejor desempeño en comparación con los métodos de aprendizaje global?

2. ¿Cuáles enfoques de DL y LWL conforman el esquema LWDL?

3. Los métodos de aprendizaje local presentan problemas para generalizar en altas dimensionalidades en el espacio latente, por lo tanto, ¿Con qué técnicas se puede lidiar el problema de alta dimensionalidad en el esquema LWDL?

4. ¿En qué dominios un esquema LWDL mejorara la exactitud en la clasificación de imágenes?

5. En el caso de aplicaciones como el reconocimiento de emociones, se tienen conjuntos de datos tomados en entornos controlados y no controlados, donde la heterogeneidad de identidad, género, edad, etnia, iluminación y pose es mucho mayor. ¿El esquema LWDL tiene un desempeño competitivo en el reconocimiento de emociones en imágenes mediante el análisis de expresiones faciales?

1.5. Hipótesis

La integración del aprendizaje local de extremo a extremo en un enfoque de aprendizaje profundo (LWDL) obtiene un desempeño competitivo en comparación con el estado del arte, generando un modelo que es interpretable y robusto en el reconocimiento de emociones y en los dominios dentro del alance del LWL.

1.6. Objetivos

1.6.1. Objetivo General

Desarrollar un esquema de aprendizaje profundo localmente ponderado (LWDL) que obtenga un desempeño competitivo en comparación con los métodos tradicionales en el reconocimiento de emociones mediante imágenes y la clasificación de imágenes dentro del alcance de LWL.

1.6.2. Objetivos Específicos

1. Evaluar las ventajas que ofrecen los esquemas de aprendizaje local y global en términos de rendimiento en ER y dominios dentro del alcance de LWL.
2. Determinar los componentes de la estructura del esquema LWDL de extremo-a-extremo.
3. Diseñar el esquema LWDL que contengan aprendizaje local de extremo-a-extremo aplicado a ER y dominios dentro del alcance de LWL.
4. Desarrollar una estrategia que resuelva la problemática de alta dimensionalidad en el espacio latente y la construcción de los aproximadores locales en el esquema LWDL.
5. Implementación y evaluación del esquema LWDL en ER y dominios dentro del alcance de LWL.

Durante la investigación se hizo un estudio acerca de la integración del aprendizaje local en un método de aprendizaje profundo, usando las CNNs para mejorar la fase predictiva de la CNN. Para ello inicialmente se llevó a cabo un análisis comparativo del aprendizaje local contra el global para identificar las tareas donde el aprendizaje local tiene un mejor desempeño. Además, se determinaron las técnicas que son aptas para adaptar el aprendizaje local con el global. Se determinó que el uso de las CNNs era el mejor modelo de DL debido a que en gran medida se reduce la dependencia del uso de técnicas de preprocesamiento de imágenes y este tipo de redes permiten de manera conjunta hacer la extracción de características y la etapa de clasificación; y su arquitectura permite incluir el aprendizaje local en su fase predictiva, con la finalidad de mejorar el desempeño de la CNN. Se desarrolló el LWDL con una red CNN que contiene múltiples ramas, donde cada rama contiene módulos de aprendizaje local, creando el modelo MB-RBFN. LA MB-RBFN se evaluó en conjuntos de datos enfocados al ER a través del reconocimiento de emociones en expresiones faciales. Los conjuntos de datos usados son de emociones básicas y compuestas, siendo las emociones compuestas consideradas como tareas de grano fino y por el desbalanceo entre las

clases, son conjuntos desbalanceados. Los resultados en esta tarea fue competitivo en el reconocimiento de emociones básicas y superior para las emociones compuestas, con respecto al estado del arte. También se llevaron a cabo experimentos donde se hace uso del aprendizaje local en algoritmos de agrupamiento y arquitecturas como los auto-codificadores. Finalmente, se hizo una mejora al MB-RBFN que integra una función de pérdida en la red, donde se mide la distribución obtenida en cada rama con la deseada, la finalidad de la mejora es que se modele correctamente el espacio de características en cada rama.

1.7. Contribuciones

En esta investigación doctoral se obtuvieron las siguientes contribuciones:

- Un esquema de *Locally Weighted Deep Learning*. El esquema permite tener una técnica de aprendizaje local integrada en un modelo de aprendizaje profundo, donde impacta en la mejora del rendimiento de la fase predictiva del modelo profundo.

- Una metodología para el reconocimiento de emociones y la clasificación de imágenes en el alcance de LWL.

- La metodología propuesta se evaluó sobre la tarea del reconocimiento de emociones básicas y compuestas. Para el caso de grano fino y conjuntos desbalanceados se uso un conjunto de datos de emociones compuestas con clases minoritarias y el modelo propuesto superó el estado del arte.

1.8. Publicaciones

Las publicación en revista JCR derivada como producto del trabajo de tesis doctoral es:

- Hernández-Luquin, F., & Escalante, H. J. (2021). Multi-branch deep radial basis function networks for facial emotion recognition. *Neural Computing and Applications*, 1-15. **JCR Q1**.

La publicación de conferencia derivada del trabajo es:

- Hernández-Luquin, F., et al, (December,2022). Dog emotion recognition from images in the wild: DEBIw dataset and first results. *Ninth International Conference on Animal-Computer Interaction (ACI'22)*

1.9. Estructura del documento

El documento se estructura de la siguiente forma:

- En el Capítulo 2 se muestran los conceptos que fundamentan la investigación, partiendo de una descripción general del aprendizaje profundo y del aprendizaje localmente ponderado. Nos enfocamos al estudio de las redes neuronales convolucionales, debido a que es el modelo usado para integrar el aprendizaje localmente ponderado.
- En el Capítulo 3 se muestra la revisión del estado del arte sobre trabajos que adaptan la integración del aprendizaje local en modelos de aprendizaje profundos.
- En el Capítulo 4 se describe la estructura del esquema llamado aprendizaje profundo localmente ponderado (LWDL) que consiste en integrar una red de aprendizaje profundo llamada MB-RBFN que integra el uso de las redes de función de base radial en una CNN. Además, se explica la mejora propuesta para MB-RBFN.
- En el Capítulo 5 se describen los experimentos que se llevaron a cabo durante la investigación, mostrando una comparativa entre modelos de aprendizaje local contra global.
- El Capítulo 6 se explica las conclusiones y el trabajo futuro de esta investigación.

2

Marco Téorico

En este capítulo se presentan las bases teóricas que sustentan la investigación. Los temas que se abordan son conceptos del aprendizaje profundo, del aprendizaje localmente ponderado y del reconocimiento de emociones.

2.1. Aprendizaje Profundo

El aprendizaje profundo (*Deep Learning, DL*) es un sub-campo dentro del aprendizaje automático (*Machine Learning, ML*) que permite a los modelos aprender múltiples niveles de representación y abstracción a partir de los datos de entrada como imágenes, sonido y texto (Francois, 2017). Los métodos basados en DL han mejorado drásticamente el estado del arte en varias tareas donde el ML no pudo tener resultados sobresalientes y el DL ha mejorando considerablemente los resultados en tareas como el reconocimiento de imágenes, tareas de clasificación y procesamiento de texto por mencionar algunas. Los métodos convencionales de ML están limitados en la habilidad de procesar los datos de manera natural, es por ello que una de las ventajas más sobresalientes e interesantes del DL es que los datos de entrada no necesitan un procesamiento previo para ser procesados por los modelos debido a que el DL ha resultado ser muy bueno para descubrir estructuras intrínsecas en datos de alta dimensión y le permite generar modelos más robustos.

Históricamente, el concepto de aprendizaje profundo se originó a partir de la investigación de las redes neuronales artificiales (Bengio et al., 2009), siendo este uno de los principales enfoques con el cual el aprendizaje profundo puede ser explicado. La familia de métodos de aprendizaje profundo se ha vuelto cada vez más extenso que abarca modelos probabilísticos jerárquicos y una gran variedad de algoritmos

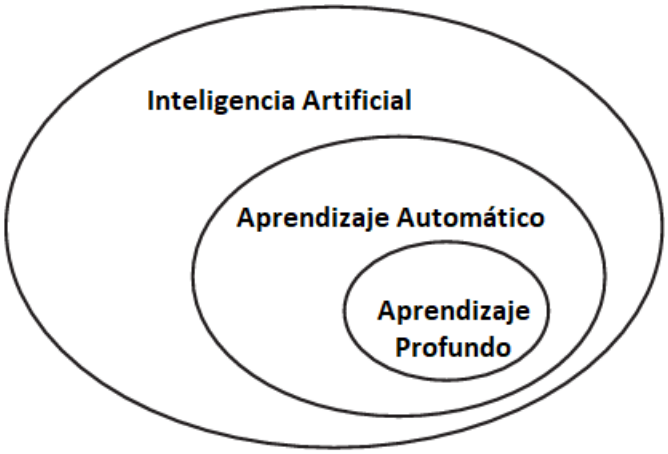


Figura 2.1: Representación del aprendizaje profundo como sub-campo del aprendizaje automático.

de aprendizaje supervisado, no supervisado, semi-supervisado y por refuerzo (Deng et al., 2014).

Las redes neuronales profundas *Deep neural networks, (DNNs)* son un ejemplo de un modelo de que aplica aprendizaje profundo (Ver Figura 2.2). Las DNNs se pueden definir como un perceptrón multicapa que consiste en una red neuronal artificial (*Artificial Neural Network, ANN*) formada por múltiples capas, de tal manera que tiene la capacidad para resolver problemas que no son linealmente separables. Generalmente, este tipo de red consiste en una capa de entrada, una capa oculta y una capa de salida (cuando se tiene una configuración simple este no se considera parte de una red profunda). Cada capa está compuesta por neuronas totalmente conectadas entre capas y se encargan de transferir los pesos a través de la red. Una arquitectura de aprendizaje profundo consiste en una representación de múltiples capas que aplican

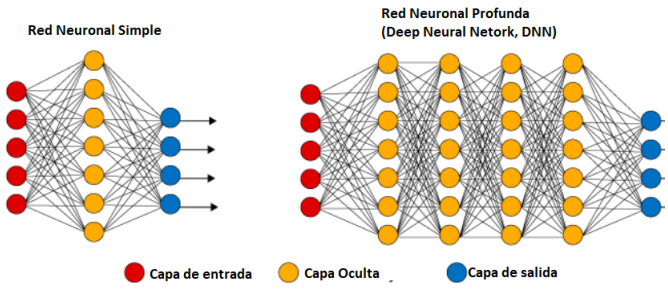


Figura 2.2: Representación de una arquitectura de red neuronal artificial.

funciones de activación para realizar transformaciones no lineales de las entradas que se puede describir de la siguiente manera:

$$f_l^{W,b} = f_l\left(\sum_{j=1}^{N_l} W_{lj}X_j + b_l\right) = f_l\left(W_l X_l + b_l\right), l \leq l \leq L \tag{2.1}$$

Donde el número de unidades ocultas está dado por N_l. El predictor se encarga de modelar un mapeo de alta dimensión F a través de la composición de funciones y se puede definir como:

$$Y(X) = F(X) = \left(f_1^{W_1,b_1} \circ \cdots \circ f_L^{W_L,b_L}\right) \tag{2.2}$$

La salida final es la respuesta de Y y puede ser categórica o numérica. La estructura explícita de una regla de predicción profunda es entonces:

$$\begin{aligned} Z^{(1)} &= f^{(1)}\left(W^{(0)}X + b^{(0)}\right), \\ Z^{(2)} &= f^{(2)}\left(W^{(1)}Z^{(1)} + b^{(1)}\right), \\ &\vdots \\ Z^{(L)} &= f^{(L)}\left(W^{(L-1)}Z^{(L-1)} + b^{(L-1)}\right), \\ Y(X) &= W^{(L)}Z^{(L)} + b^{(1)} \end{aligned} \tag{2.3}$$

Aquí, se define como: $Z^{(L)}$ la L-ésima capa, $W^{(L)}$ la matriz de pesos y $b^{(L)}$ el sesgo. $Z^{(L)}$ contiene las características ocultas extraídas, dicho de otra manera, el enfoque profundo usa predictores jerárquicos que comprenden una serie de transformaciones no lineales en L aplicadas a X. Cada una de las transformaciones L se refiere a una capa donde la entrada original es X, la salida de la primera transformación es la primera capa, y así sucesivamente hasta la salida Y como la capa $(L+1)$. Usamos $l \in \{1, \cdots, L\}$ para indexar las capas que se denominan capas ocultas. El número de capas L representa la profundidad de la arquitectura profunda.

A través del estudio, han surgido varios enfoques notables en DL como las *Convolutional Neural Networks, (CNNs)*, *Recurrent Neural Networks, (RNN)* (incluyendo *Long Short-Term Memory, (LSTM)* y *Gated Recurrent Units, (GRU)*), *Auto-Encoder, (AE)*, *Deep Belief Networks, (DBN)*, *Generative Adversarial Networks, (GAN)*, *Deep Reinforcement Learning, (DRL)* (Alom et al., 2019) y los novedosos *Transformers* (Han et al., 2022).

DL ha logrado una importancia excepcional en la comunidad científica debido a la aplicabilidad a casi cualquier dominio, siendo capaz de resolver tareas asociadas con el campo del procesamiento de imágenes, visión por computadora, reconocimiento de voz, procesamiento del lenguaje natural, traducción automática, arte, imágenes médicas, procesamiento de información médica, robótica y ciberseguridad, todas con notables resultados. Aunque una desventaja es que los modelos de DL generalizan mejor cuando se tiene un gran volumen de datos para entrenar y por lo tanto, esto lo convierte en modelos más complejos que requieren más recursos en hardware a diferencia de los modelos tradicionales de ML.

2.1.1. Componentes principales de una red de aprendizaje profundo

Los componentes principales que contiene una red profunda son (Chollet, 2021):

- Ajuste de capas. Las capas son una unidad fundamental en las redes profundas que van cambiando dependiendo del tipo de función de activación que use.
- Funciones de activación. Las funciones de activación son una función limitadora o umbral que modifica el valor de la salida de la neurona, poniendo un límite en el valor del cual no debe sobrepasar antes de propagar su valor
- Funciones de pérdida. Las funciones de pérdida cuantifican la salida predicha (o la etiqueta) contra la salida real. Se utilizan funciones de pérdida para determinar la penalización por una clasificación incorrecta de un dato de entrada.
- Métodos de optimización. El entrenamiento de un modelo en aprendizaje automático implica encontrar el mejor conjunto de valores para el vector de parámetros como los valores de función de pérdida más bajo. El aprendizaje automático se puede ver como un problema de optimización, en el que se minimiza la función de pérdida con respecto a los parámetros de la función de predicción (según nuestro modelo).
- Ajuste de hiper-parámetros. Un hiper-parámetro se refiere a elegir libremente por el usuario algunas configuraciones que podrían mejorar el rendimiento.

Un enfoque ampliamente usado en el DL para el procesamiento y clasificación de imágenes son las Redes Neuronales Convolucionales (*Convolutional Neural Networks*, CNNs). Las CNNs han alcanzado una importancia notable en el DL debido a que tienen la capacidad de funcionar como extractores automáticos de características en datos presentados en forma de tensor, por ejemplo, las imágenes. Este tipo de arquitecturas han mostrado resultados sobresalientes en tareas de clasificación de imágenes, en la siguiente sección 2.2, se explica el funcionamiento de las CNNs.

2.2. Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (*Convolutional Neural Networks, CNNs*) son un tipo especial de red neuronal que aplica la operación matemática de convolución en las capas iniciales de la red. Las CNNs constan de varias capas que permiten un aprendizaje automático de las características ya que su

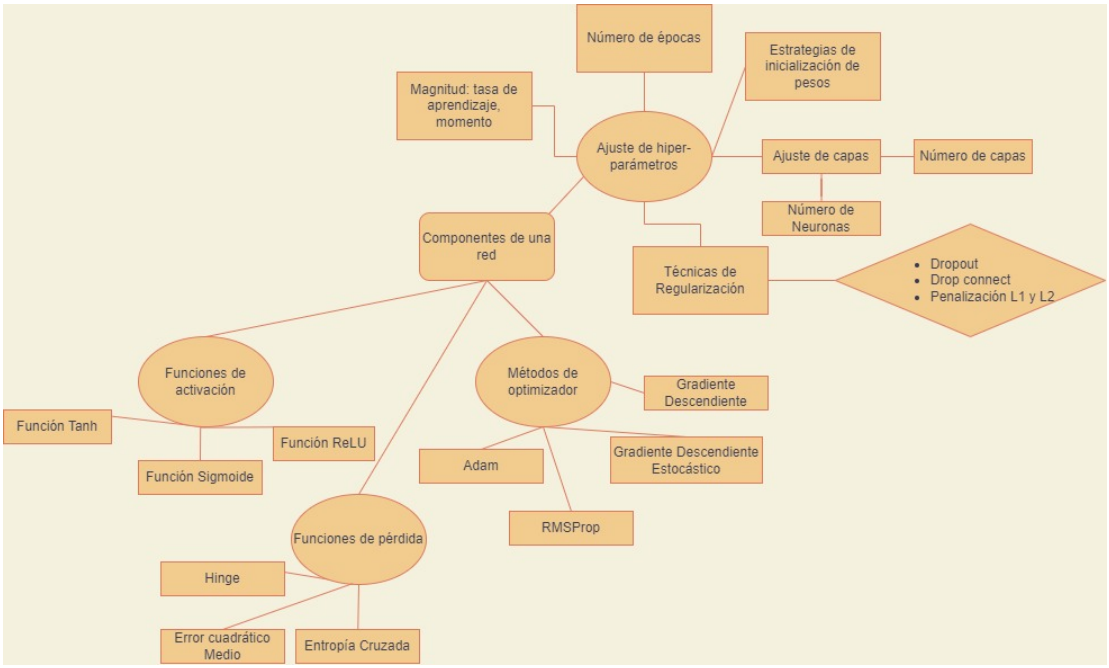


Figura 2.3: Parámetros comúnmente configurados en una red de aprendizaje profundo.

entrada son datos como imágenes. Las primeras etapas se componen de dos tipos de capas: capas convolucionales y capas *pooling*. Una primera capa de convolución aprenderá pequeños patrones locales como bordes, una segunda capa de convolución aprenderá patrones más grandes hechos de las características de las primeras capas y así sucesivamente (como se puede observar en la fig 2.4). Las unidades en una capa convolucional se organizan en mapas de características, dentro de los cuales cada unidad está conectada a parches locales en los mapas de características de la capa anterior a través de un conjunto de pesos denominado filtros. El resultado de esta suma ponderada local luego se pasa a través de una no linealidad como ReLU. Diferentes mapas de características en una capa usan diferentes filtros. Una vez que son obtenidos los mapas de características, los mapas se usan como patrones de entrada en la fase de clasificación de la CNN. La fase de clasificación se puede ver como una red neuronal se encarga de separar las características y categorizarla en una clase (Krizhevsky et al., 2012).

La capa convolucional implica la operación de convolución se le conoce como el detector de características de una CNN. La entrada a una capa convolucional son datos sin preprocesar, por ejemplo imágenes y genera una salida de imágenes *mapa de características* que se usan como entrada a otra capa convolucional. Generalmente se interpreta como un filtro donde el núcleo filtra datos de entrada para cierto tipo de información, siendo capaz de analizar información acerca de la posición del objeto, la invarianza a rotaciones, el análisis de bordes y las texturas en la imagen (vea la representación en la Figura 2.6).

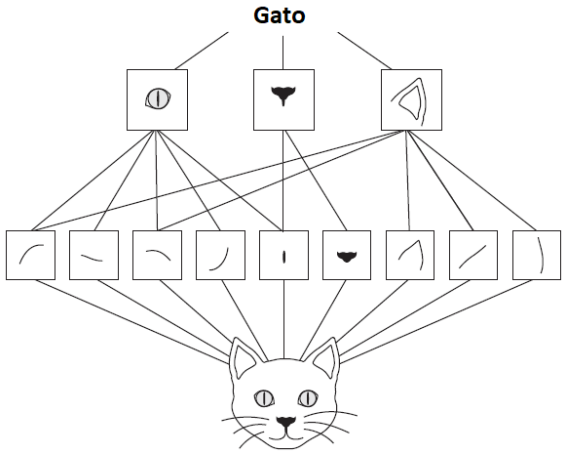


Figura 2.4: Representación de los patrones aprendidos por la CNN. La figura ilustra una representación de lo que la red aprende durante el entrenamiento, la red aprende bordes, texturas y formas. Figura reproducida de (Francois, 2017)

La operación matemática convolución en una imagen se denota como:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.4}$$

Se toma como entrada una imagen I, se aplica un kernel de convolución K sobre cada píxel en la posición i, j en la imagen y nos da un mapa de características de la imagen como salida S.
Otro componente en la capa convolucional son las funciones de activación. En el caso de una CNN, comúnmente se usa la función de activación llamada unidad lineal rectificada o ReLu (*Rectified Linear Unit*). Esta función de activación calcula la salida como se muestra en la Ecuación 2.5, la función calcula si la entrada está por debajo de cero, la salida es cero.

$$R(x) = max(0, x) \tag{2.5}$$

Además de la operación de convolución en la fase de extracción de características, se usa una capa llamada *pooling layer*. La capa *Pooling* se inserta entre las capas convolucionales para reducir progresivamente el tamaño espacial (ancho y alto) de la representación de los datos y poder controlar el sobreajuste. La operación más usada en esta capa es Max-Pooling (como se muestra gráficamente en la Figura 2.10). Esta operación implica la agrupación de un vecindario rectangular y su salida toma el valor máximo del píxel dentro del vecindario. Otras funciones de agrupación incluyen el promedio de un vecindario rectangular (Average-Pooling), la norma L2 de un vecindario rectangular y un promedio ponderado basado en la distancia desde el píxel central.

La fase de clasificación en una CNN es llamada capa totalmente conectada. La capa se comporta

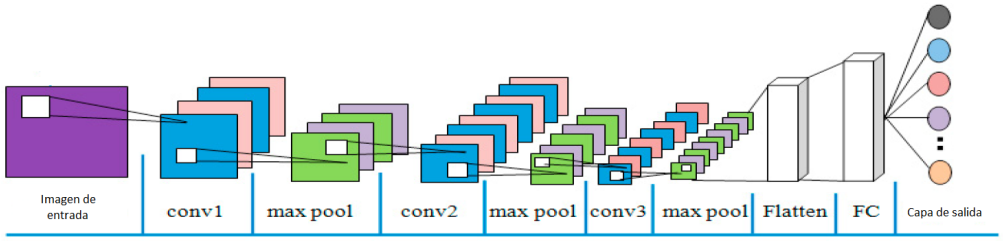


Figura 2.5: Modelo de una Red Neuronal Convolucional. Una CNN se define como una red de aprendizaje profundo que contiene dos fases la de extracción de características y la de predicción. La fase de extracción de características contiene capas convolucionales y la fase de predicción una red tipo MLP. Figura reproducida de (Santhoshkumar and Geetha, 2019).

al igual una red neuronal, todas las neuronas de la capa están conectadas con cada neurona de la capa anterior y se puede calcular como:

$$F(x) = \sigma(W * x) \tag{2.6}$$

Donde F es la salida de las unidades, $W \in \Re$ son los pesos de la red y $\sigma : \Re \to \Re$ es la función de activación de la red. La capa final generalmente es la capa en la que el error se puede propagar usando el algoritmo de retro-propagación (*Back Propagation*) y el desempeño de la red se incrementa. Aquí la red generalmente usa la función *softmax*, donde la salida se calcula como sigue:

$$S(x)_j = \frac{x^{x_i}}{\sum_{i=0}^{N} e^{x_i}} \tag{2.7}$$

$S(x) : R \to [0,1]N$, donde N es el tamaño del vector de entrada. Para $1 \leq j \leq N$. La capa de salida de una CNN tiene un tamaño igual al número de clases.

Algunas arquitecturas convolucionales novedosas son:

- **LeNet-5** (LeCun et al., 1998): Fue la primera CNN desarrollada por Yann LeCun por la década de los 90s. La arquitectura LeNet es la más conocida porque se utilizaba para leer códigos postales, dígitos, etc. En la Figura 2.9
- **AlexNet** (Krizhevsky et al., 2017): Es una red neuronal convolucional profunda que se publicó en el 2012 y fue utilizada por primera vez en el desafío de reconocimiento visual a gran escala llamado ***ILSVRC***, y se utilizó para tareas de clasificación de imágenes.
- **VGGNet** (Simonyan and Zisserman, 2014): Es una arquitectura de red convolucional que ha sido usada ampliamente como base en modelos CNN. Esta red es reconocida por sus dos variantes

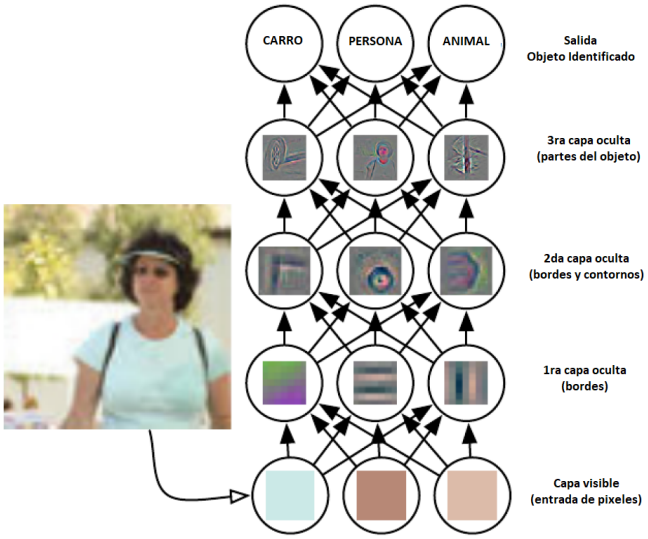


Figura 2.6: Representación del aprendizaje profundo como subcampo del aprendizaje automático. La figura ilustra como cada capa de la red profunda aprende distintas características a partir de una imagen de entrada. Figura reproducida de (Goodfellow et al., 2016)

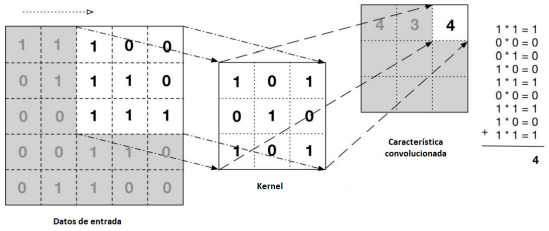


Figura 2.7: Ejemplo de la operación matemática de convolución sobre una imagen. Figura reproducida de (Patterson and Gibson, 2017).

VGG16 y VGG19. La variación consiste en el tamaño de la profundidad de la red que es 16 capas convolucionales y 19, respectivamente.

- **ResNet** (He et al., 2016a): Es una red neuronal convolucional que se planteó en el 2015 con la finalidad de lidiar con la problemática del *vanishing gradient*. La finalidad es que la problemática se resuelva añadiendo conexiones residuales que permiten que los gradientes pueden fluir directamente a través de las conexiones de salto hacia atrás desde las capas posteriores hasta los filtros iniciales.

- **GoogleNet**(Szegedy et al., 2015): Es una red propuesta por Google en el 2014 y fue usada para tareas de clasificación y detección en la competencia ILSVRC. Esta red convolucional particularmente propone un módulo entre sus capas convolucionales que hacen que la red no solo sea profunda, sino

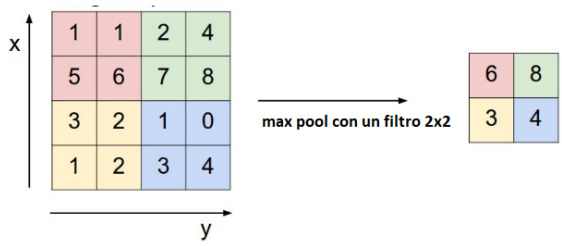


Figura 2.8: Ejemplo de la operación ***Max Pooling***. Figura reproducida de (Santhoshkumar and Geetha, 2019)

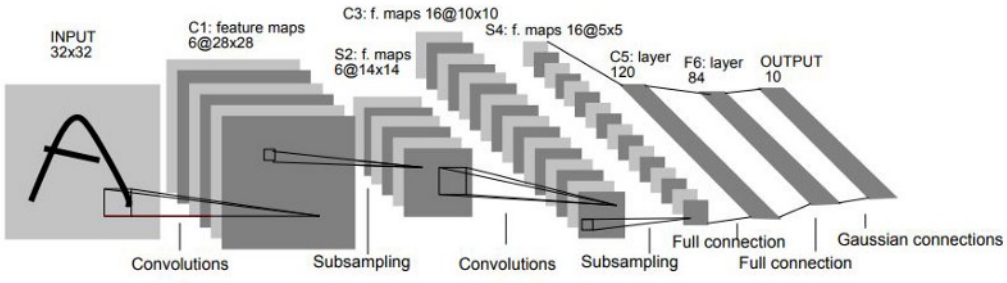


Figura 2.9: Arquitectura de la red convolucional LeNet-5, publicada por primera vez para el reconocimiento de digitos. Figura reproducida de (LeCun et al., 1998)

más ancha.

- **EfficientNet-B7** (Tan and Le, 2019): Es una red propuesta en el 2019, la red alcanza el estado del arte para los conjuntos de datos ImageNet(Krizhevsky et al., 2017), Cifar100(Krizhevsky et al., 2009) y Flowers(Nilsback and Zisserman, 2008). La red incluye un método de escalado que escala uniformemente todas las dimensiones de profundidad/ancho/resolución, permitiendo a la red ser más pequeña y más rápida que las redes convencionales existentes en su momento.

2.3. Aprendizaje Local

El aprendizaje local está constituido de tres enfoques principales y son: las representaciones locales, la selección local y el aprendizaje localmente ponderado. Una representación local implica que cada nuevo punto de datos afecta a un pequeño subconjunto de parámetros y el responder una consulta también implicaría un pequeño subconjunto de parámetros. Algunos ejemplos de representaciones locales son las tablas de búsqueda y clasificadores basados en ejemplos o prototipos (Atkeson et al., 1997b). La selección local se refiere a métodos que almacenan todos (o la mayoría) de los datos de entrenamiento en memoria y

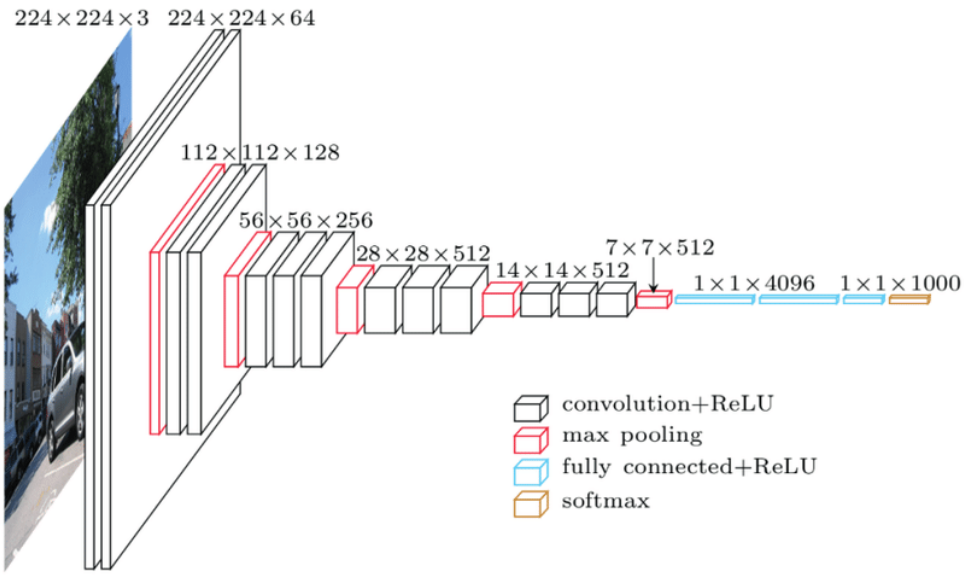


Figura 2.10: Arquitectura de la red convolucional VGGNet. Figura reproducida de (Simonyan and Zisserman, 2014).

usan una función de distancia para determinar qué puntos almacenados son relevantes para la consulta. La función de la selección local es ubicar una única salida usando el vecino más cercano o usando un esquema de votación basado en la distancia (Atkeson et al., 1997a).

El aprendizaje localmente ponderado almacena explícitamente los datos de entrenamiento (al igual que los enfoques de selección local) y solo ajusta los parámetros a los datos de entrenamiento cuando se conoce una consulta. La característica crítica del aprendizaje localmente ponderado es que se utiliza un criterio de ponderación local con respecto a la ubicación de la consulta para ajustar algún tipo de modelo paramétrico a los datos. Aquí surge la confusión de las estructuras de modelos aparentemente globales (por ejemplo, redes neuronales sigmoidales multicapa o las redes de función de base radial). Estos modelos se llaman modelos locales debido al criterio de entrenamiento que se establece y consiste en que todos los datos pueden participar en la construcción del modelo local, siempre que los datos distantes importen menos que los datos cercanos. Por lo tanto, existen enfoques y representaciones globales que se pueden transformar en enfoques ponderados localmente utilizando un criterio de entrenamiento local (Atkeson et al., 1997a).

Las ventajas que ofrecen los métodos basados en el aprendizaje local es que son modelos flexibles e interpretables y tienen una configuración de parámetros simple que mejora el rendimiento en la pre-

dicción. Además, se pueden representar funciones no lineales con la ventaja de tener reglas simples en su entrenamiento como: el control de ajuste de parámetros, el suavizado, el rechazo de valores atípicos, entre otros. El proceso de modelado es fácil de entender y ajustar, debido a que se construye con puntos relacionados al punto de consulta. Una desventaja de los métodos que incluyen el aprendizaje local es que fallan en su generalización cuando se presenta una alta dimensionalidad en el espacio latente (Neruda and Kudová, 2005).

2.3.1. Aprendizaje Localmente Ponderado

En ML un enfoque que adapta algoritmos de aprendizaje local es el aprendizaje localmente ponderado también llamado (*Locally Weighted Learning, LWL*). LWL es una técnica de aproximación de funciones donde se realiza una predicción mediante el uso de un modelo local aproximado en torno al punto de interés. Los modelos locales adaptan modelos para cada punto de interés, se va creando un modelo basado en las vecindades del punto. Es decir, para cada punto de los datos, se calcula un factor de ponderación que expresa la influencia entre sí de los datos para la predicción. En general, los puntos de los datos que están cerca del punto de consulta actual, reciben un peso mayor que los puntos de datos que están distantes (Atkeson et al., 1997b), como se muestra en la Figura 2.11.

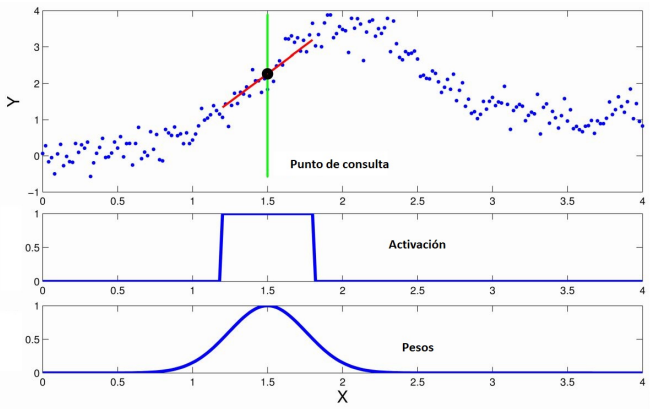


Figura 2.11: Ejemplo de Regresión Localmente Ponderada ***Locally Weighted Regression, (LWR)***, en la gráfica los puntos azules representan el conjunto de datos del entrenamiento (x, y) y los modelos lineales locales (líneas rojas). Figura reproducida de (Englert, 2012).

Hay adaptaciones de LWL que implican la combinación de métodos de aprendizaje local y global, creando clasificadores robustos como se muestran en la Tabla 2.1. Este tipo de combinaciones híbridas tienen la desventaja de ser combinados con métodos basados en instancias, significa que todos los datos

Modelos globales		Modelos locales e híbridos	
Suport Vector Machine	SVM	Support Vector Machine - kNN	SVM-kNN
Multi Layer Perceptron	MLP	Radia Basis Function Networks	RBF
Decision Tree	DT	Decision Tree - kNN	DT-kNN
Naive Bayes	NB	Naive Bayes - kNN	NB-kNN
Linear Regressor	LR	Locally Weighted Lineal Regressor	LWLR
		K-Nearest Neighbors	KNN
		Learning vector quantization	LVQ

Tabla 2.1: En la tabla se muestran algunos clasificadores basados en aprendizaje local, global e híbridos.

de entrenamiento se almacenan en memoria.

Existen algunos modelos de LWL que son no-paramétricos y la predicción actual se realiza mediante funciones locales que utilizan sólo un subconjunto de datos. Los modelos paramétricos aprenden una función que aproxima los datos de entrenamiento a la variable objetivo por un vector de parámetros cuyo tamaño es finito y fijado antes de observar cualquier dato. En los modelos no-paramétricos, la complejidad de su espacio de hipótesis crece según el número de instancias de datos a considerar. Por ejemplo, el algoritmo kNN hace que su complejidad sea una función del tamaño del conjunto de entrenamiento, lo cual es un algoritmo intratable en conjuntos de datos extensos. El objetivo detrás de LWL es que en lugar de construir un modelo global para todo el espacio de instancias, se construya un modelo local basado en datos vecinos al punto de consulta para cada punto de consulta.

Una característica atractiva del LWL es que los modelos son interpretables. El proceso de modelado es fácil de entender y por lo tanto, fácil de ajustar o controlar algunos parámetros de entrenamiento en el clasificador. Hay dos categorías principales en las que puede dividir los métodos de LWL. La primera categoría incluye los métodos LWL basados en memoria donde todos los datos de entrenamiento se guardan en la memoria para hacer su predicción, por ejemplo los algoritmos: *k-Nearest Neighbor, Weighted Average, y Locally Weighted Regression*. La segunda categoría incluye métodos LWL incrementales que no necesitan recordar ningún dato explícitamente, por ejemplo las redes tipo RBF (*Radial Basis Functions Networks, RBF*) (Atkeson et al., 1997a).

La desventaja de los modelos basados en LWL radica en que son sensibles cuando se presenta una alta dimensionalidad en los datos, también tienen problemas para generalizar adecuadamente cuando se tiene un conjunto de datos extenso (para los métodos basados en memoria). En el caso de los métodos LWL incrementales, es complicado establecer parámetros iniciales en la configuración de los algoritmos, aunque una ventaja de los métodos incrementales es que pueden ser adaptados en tareas de clasificación

donde el conjunto de datos es extenso y es capaz de generalizar adecuadamente (Colmenares, 2007).

A continuación, se da una descripción del funcionamiento de los métodos basados en aprendizaje local como: K-NN, RBF, SOMs y LVQ.

2.3.2. Método de los *K* vecinos más cercanos

El método de los *K* vecinos más cercanos (también llamado *K-Nearest Neighbor*, K-NN) es un tipo de aprendizaje basado en instancias donde el algoritmo supone que todas las instancias corresponden a puntos en el espacio n-dimensional $\Re^n$, y la función objetivo se aproxima localmente (Mitchell et al., 1990). Comúnmente, el vecino más cercano de una instancia se define en términos de la distancia euclidiana estándar, más precisamente, una instancia se puede describir como un vector de características en la forma:

$$\langle a_1(x), a_2(x) \cdots a_n(x) \rangle \tag{2.8}$$

Donde denotamos $a_n(x)$ como el valor del atributo n-ésimo de instancia x. Por lo tanto, la distancia entre dos instancias $d(x_i, x_j)$ se puede definir de la siguiente manera:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{2.9}$$

La función objetivo $f : \Re^n \to V$ del vecino más cercano se puede calcular como una valor discreto o un valor real, para un conjunto finito $V = v_1, \cdots, v_s$. El algoritmo de los k vecinos más cercanos toma cada ejemplo del conjunto de prueba x_q y calcula los k ejemplos más cercanos del conjunto de entrenamiento. Por ejemplo, si $k = 1$ entonces el algoritmo retorna el 1-vecino más cercano y asigna a $f(x_q)$ el valor de $f(x_i)$ donde x_i es la instancia de entrenamiento más cercana a x_q. Cuando se tienen valores grandes de k, el algoritmo asigna a x_q al valor más común en el conjunto de los k más cercanos.
La ventaja del algoritmo es que es fácil de interpretar sus resultados. La desventaja del algoritmo es que es un método basado en instancias, ya que no aprende explícitamente un modelo, en su lugar memoriza las instancias de entrenamiento, para ser posteriormente usadas como conocimiento en la fase de predicción. También, es insensible a los valores atípicos es decir, la precisión puede verse afectada por el ruido o las características irrelevantes.

2.3.3. Redes de Función de Base Radial

Las redes de Función de Base Radial, (también llamadas *Radial Basis Function, RBF*) son un tipo de red neuronal artificial construida a partir de funciones de kernel espacialmente localizadas. Las redes RBF pueden describirse como una combinación de los enfoques LWL (donde se hace una aproximación local

en el momento de la consulta) y redes neuronales (donde se forma una aproximación global a la función objetivo en el momento del entrenamiento) (Powell, 1987). Por lo tanto, en las Redes RBF su modelo de aprendizaje supervisado se realiza bajo el concepto de aproximación local.

La arquitectura de una Red RFB es simple y está compuesta de una capa de entrada, una capa oculta (en esta capa se definen las funciones RBF) y una capa de salida. En la Figura 2.12 se muestra la arquitectura típica de una Red RBF.

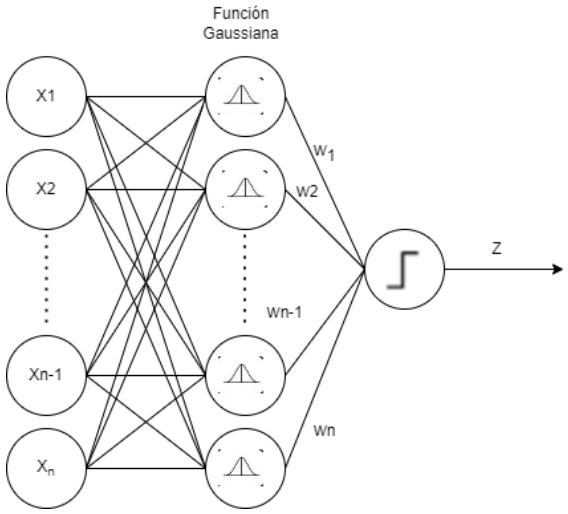


Figura 2.12: Arquitectura de Red de Función de Base Radial (RBFN). Consiste en un vector de entrada, una capa de neuronas RBF y una capa de salida.

Su funcionamiento básicamente consiste en que la capa de entrada transmite los ejemplos o patrones de entrenamiento y prueba hacia las capas ocultas. Es decir, el número de unidades de entrada es exactamente igual a la dimensionalidad d de los datos. Los cálculos en la capa oculta están basados sobre comparaciones entre vectores prototipos. Los vectores prototipo se obtienen a partir de un agrupamiento previo del conjunto de datos de entrada, tomando los centros del agrupamiento como los vectores prototipo. Cada capa oculta tiene un vector prototipo de dimensiones d. Para la i-ésima unidad oculta el vector prototipo es denotado por μ_i. Además, la i-ésima unidad oculta contiene un ancho de banda denotado por σ_i. Aunque los vectores prototipo son siempre específicos para unidades particulares, los anchos de banda de diferentes unidades σ_i a menudo se establecen en el mismo valor σ. Los vectores prototipo y los anchos de banda generalmente se aprenden de manera no supervisada o con el uso de una supervisión moderada. Entonces, para cualquier punto de entrada en el conjunto de entrenamiento X, la activación $\phi_i(X)$ de la

i-ésima unidad oculta se define de la siguiente manera:

$$h_i = \phi_i(\overline{X}) = e^{\left(-\frac{\|\overline{X}-\overline{\mu}_i\|^2}{2\cdot\sigma i^2}\right)} \forall \in \{1, \cdots, m\} \tag{2.10}$$

El número total de unidades ocultas se denota por m. Cada una de estas unidades m está diseñada para tener un alto nivel de influencia con puntos cercanos a su vector prototipo. Por lo tanto, se puede ver a m como un número de grupos utilizados o centroides para modelar las unidades RBF. Para entradas de baja dimensión, es típico que el valor de m sea mayor que la dimensionalidad de entrada d, pero menor que el número de puntos de entrenamiento n. Los pesos de las conexiones únicamente existen de los nodos ocultos a los nodos de salida y se establecen en w_i. Luego, la predicción $\overline{y}$ de la red RBF en la capa de salida se define de la siguiente manera:

$$\overline{y} = \sum_{i=1}^{m} w_i h_i = \sum_{i=1}^{m} w_i \phi_i(\overline{X}) = \sum_{i=1}^{m} w_i e^{\left(-\frac{\|\overline{X}-\overline{\mu}_i\|^2}{2\cdot\sigma i^2}\right)} \tag{2.11}$$

Una vez que se obtiene el valor predicho y, entonces se puede configurar una función de pérdida, por ejemplo, mínimos cuadrados. Los valores de los pesos $w_1, \cdots, w_m$ son aprendidos de forma supervisada.

Las ventajas una red RBF es que tiene un mejor desempeño cuando el volumen de datos de entrenamiento es grande. También este tipo de red se le reconoce como una red con alta eficiencia en la fase de entrenamiento; ya que su aprendizaje es más rápido debido a que el cambio de peso sólo afecta a la neurona oculta asociada a dicho peso, es decir, sólo a un grupo de patrones pertenecientes a la clase que representa a dicha neurona oculta. A diferencia del perceptrón multicapa o MLP, las redes RBF requieren una mayor cantidad de neuronas en los nodos ocultos para que la red tenga un mejor desempeño. Las redes RBF no son comúnmente utilizadas en aplicaciones que impliquen un alto volumen de patrones de entrenamiento.

2.3.4. Otros métodos de aprendizaje local

Existen otros métodos que se pueden considerar parte del aprendizaje local y son las redes SOMs, el algoritmo LVQ y el método de regresión lineal localmente ponderado. En esta sección se muestra una descripción de estos métodos.

Los mapas auto-organizados también llamados en inglés (*Self organizing maps, SOMs*) son un tipo de red neuronal artificial (ANN) que se usan para reducir la dimensionalidad y se entrenan utilizando el aprendizaje no supervisado para producir una representación discreta del espacio de entrada que son las muestras de entrenamiento de baja dimensión (típicamente bidimensional) llamada mapa. Los mapas

SOMs difieren de otras redes neuronales artificiales, ya que aplican el aprendizaje competitivo en oposición al aprendizaje de corrección de errores (como la propagación hacia atrás con descenso de gradiente) y en el sentido de que usan una función de vecindario para preservar las propiedades topológicas del espacio de entrada. La red SOM generalmente consta de dos capas de nodos: la de entrada y de salida. A diferencia de otras redes neuronales, la red SOM en la capa de entrada los nodos de origen están directamente conectados a la capa de salida sin ninguna capa oculta (Asan and Ercan, 2012). Los nodos en la capa de entrada denotan los atributos (características).

Otro método estrechamente relacionado a las redes SOM es el aprendizaje basado en prototipos llamado en inglés (*Learning Vector Quantization, LVQ*). La diferencia radica en que SOM es un método de agrupamiento y aprendizaje no supervisados en cambio LVQ es aprendizaje supervisado. LVQ utiliza uno o más prototipos para representar cada clase en el conjunto de datos. A nuevos puntos de datos, se les asigna la clase del prototipo más cercano a ellos. Por lo general, se usa la métrica de la distancia euclidiana. No hay limitación sobre cuántos prototipos pueden existir por clase, pero debe ser al menos 1 para cada clase. Este algoritmo contiene fase de entrenamiento y prueba.

Un método no paramétrico basado en el aprendizaje localmente ponderado es la regresión lineal localmente ponderada en inglés llamada *(Locally Wwigthed Lineal Regresor, LWLR).* La diferencia contra el regresor lineal (considerado parte de un modelo de aprendizaje global), es qe ajusta muchos sub-modelos por cada punto de consulta, distinto al regresor lineal que genera un único modelo sobre todo el conjunto de datos. La curva resultante final es el producto de todos esos modelos locales de regresión como se ilustra en la Figura 2.11.

Los algoritmos basados en el aprendizaje local se consideran pueden mejorar el desempeño en la tarea del reconocimiento de emociones. Dado que el reconocimiento de emociones es una aplicación que se considera dentro del alcance del aprendizaje local. A continuación, se aborda un estudio de la forma en la que se puede llevar a cabo el reconocimiento de emociones.

2.4. Reconocimiento de Emociones

Las emociones son estados de sentimiento con valencia afectiva negativa o positiva (Ortony et al., 1988). Juegan un papel importante en la vida humana y las interacciones sociales de cada persona, ya que constituyen una parte importante en la percepción y cognición humana (Imani and Montazer, 2019). Las

investigaciones neurológicas y el estudio de las funciones utilitarias dentro del cerebro humano muestran una relación evidente entre las emociones humanas y la toma racional de decisiones (Picard et al., 2001; Sander et al., 2005).

El reconocimiento automático de emociones (también llamado *Emotion Recognition, ER*) se ha abordado usando diferentes modalidades que incluyen: voz, texto, signos vitales (EGG), reconocimiento de gestos, expresiones faciales e híbridos como se muestra en la Fig 2.13. Una de las formas más usadas en ER es a través del análisis de las expresiones faciales.

Las expresiones faciales pueden ser capturadas de forma simple mediante una cámara para su análisis en modelos computacionales y su etiquetado se puede llevar a cabo por expertos simplemente con observar la imagen. Los conjuntos de datos de expresiones faciales que se usan para el reconocimiento de emociones contienen imágenes de emociones *planteadas* o *aparentes*. Para su construcción se pide a un grupo de participantes que expresen diferentes estados emocionales básicos. Un conjunto de datos de expresiones espontáneas se refiere a las expresiones cuando son naturales. La diferencia entre las expresiones espontáneas y aparentes radica en que difieren notablemente en términos de: intensidad, configuración y duración. En la mayoría de los casos, las expresiones aparentes son exageradas, mientras que las espontáneas son sutiles y difieren en apariencia.

Si bien, el reconocer emociones mediante signos vitales es una de las técnicas más acertadas, el medir estados emocionales involucra obtener señales vitales como: la presión arterial, la respiración, los electroencefalogramas, y los electrocardiogramas. La desventaja de este enfoque es que se requiere el uso de sensores físicos y usuarios con experiencia para el manejo de los equipos (Imani and Montazer, 2019), esto limita la movilidad de los participantes y distrae las reacciones emocionales de la persona.

Figura 2.13: Diferentes fuentes para el reconocimiento de emociones.

2.4.1. Reconocimiento de Expresiones Faciales

La expresión facial es una de las características más importantes en el reconocimiento de las emociones humanas. Una expresión facial consiste en una secuencia de señales no verbales en la comunicación e interacción entre humanos (IqbalQuraishi et al., 2012) para comunicar una emoción. Una expresión facial implica la contracción de músculos en la cara y se puede reconocer a partir de imágenes estáticas o una secuencia de imágenes o vídeos (Imani and Montazer, 2019). El humano puede asumir la emoción de alguien con el sólo hecho de observar su rostro. El reconocimiento de expresiones faciales tiene varias aplicaciones como son el comportamiento humano no verbal, la robótica, rehabilitación y la interacción humano-computadora (Jain et al., 2019).

El reconocimiento de expresiones faciales o *(Facial Expression Recognition, FER)* consiste en categorizar la expresión facial en diferentes clases para distinguir entre distintos gestos faciales e interpretar incluso estados mentales (Imani and Montazer, 2019). Las expresiones faciales se deben a deformaciones temporales de los elementos faciales como: la boca, las cejas, los ojos y la nariz. El grado de cambios en todas las regiones faciales determina indirectamente la intensidad de la emoción.

Existe un sistema que codifica las expresiones llamado *Sistema de codificación de acción facial (FACS)* (Ekman, 1977). FACS segmenta los efectos visibles de la activación muscular facial en unidades de acción (*Action Units, AU*) (Hamm et al., 2011) y se utilizan particularmente un conjunto de 46 unidades de acción principales con respecto a su intensidad y ubicación. Esas unidades de acción codifican las acciones fundamentales de los músculos individuales o grupos de músculos que se ven involucrados cuando está presente una expresión facial de una emoción en particular.

Figura 2.14: Ejemplos de la base de datos CK +. En la figura se presentan 8 emociones que expresan diferentes AU. Las emociones que se representan en la ilustración corresponden a: disgusto, felicidad, sorpresa, miedo, enojo, desprecio, tristeza y neutralidad, respectivamente. Figura reproducida de (Lucey et al., 2010).

FER generalmente se lleva a cabo en aplicaciones como estudios psicológicos, animaciones faciales, ciencia cognitiva, neurociencia, comprensión de imágenes, videojuegos, robótica, dispositivos de visión

por computadora y aprendizaje automático (Corneanu et al., 2016) y se ha abordado mediante el uso de modelos de aprendizaje profundo, alcanzado resultados sobresalientes en comparación con los métodos convencionales de ML. En el siguiente capítulo se ahondará en la tarea del ER y se presentarán los distintos enfoques aplicados para su solución.

2.4.2. Discusión

En el presente capítulo se presentaron las bases teóricas que sustentan este trabajo de investigación. Se realizó un análisis del aprendizaje profundo y el aprendizaje local, se revisó de qué forma se puede llevar a cabo la integración de estos dos enfoques en un modelo de aprendizaje profundo. Se observó que una prometedora aplicación es el uso de las CNNs en combinación con redes RBF. Se estudió que uno de los dominios dentro del alcance del aprendizaje local es el reconocimiento de emociones y como se lleva a cabo dicha tarea.

3

Trabajo relacionado

En esta sección presentamos una revisión de la literatura sobre los trabajos actuales relacionados a la investigación. Se tratan áreas como: tipos de aprendizaje local, la adaptación de métodos locales en el aprendizaje profundo y el reconocimiento de emociones en imágenes.

3.1. Métodos basados en Aprendizaje Localmente Ponderado

El aprendizaje localmente ponderado (LWL, *Locally Weighted Learning*) se puede dividir en cuatro categorías que incluyen aprendizaje basado en distancias, centroides, modelos locales ponderados e híbridos de modelos globales y locales, como se ilustra la Fig.3.1. Los algoritmos basados en distancias son algoritmos de aprendizaje automático que clasifican los ejemplos calculando las distancias entre ellos y una serie de ejemplos almacenados internamente (Mitchell et al., 2007), así los ejemplos más cercanos a la consulta tienen la mayor influencia en la clasificación asignada a la consulta. El aprendizaje basado en centroides se refiere a algoritmos de agrupación que son no supervisados, este trata de encontrar un número fijo k de agrupaciones en un conjunto de datos basándose en las similitudes de sus características. Los modelos híbridos consisten en entrenar modelos de aprendizaje global para un conjunto local de instancias relacionadas a la instancia de consulta (Segata et al., 2012; Yuan et al., 2008; Zanchettin et al., 2012; Zhang et al., 2006).

Existen trabajos que incluyen adaptaciones híbridas del aprendizaje local y global para mejorar las tareas de clasificación. Por ejemplo, la adaptación SVM-kNN resuelve tareas relacionadas con la clasificación de personalidad (Pratama and Sarno, 2015), clasificación de imágenes (Segata et al., 2012), tareas de

reconocimiento de escritura a mano (Zanchettin et al., 2012) y tareas relacionadas con la clasificación de texto (Yuan et al., 2008). El método SVM-KNN consiste en que para cada instancia de consulta, se toma un conjunto de los k-vecinos más cercanos; el conjunto se usa para entrenar un clasificador SVM que genere un modelo local para realizar la predicción. Las contribuciones anteriores de aprendizaje ponderado localmente se centran en cómo las funciones del clasificador pueden aproximarse utilizando cualquier esquema de codificación local. En el trabajo presentado por (Ladicky and Torr, 2011), el autor propone un clasificador SVM localmente lineal con un límite de decisión suave y una curvatura limitada. El esquema toma localmente un conjunto de datos y crea una función de decisión entre los datos para demostrar que aunque el problema no es linealmente separable, localmente en regiones suficientemente pequeñas, el límite de decisión es casi lineal. Por lo tanto, los datos se pueden separar razonablemente bien utilizando un clasificador localmente lineal. Otra adaptación al aprendizaje local en SVM es la integración de un kernel no-lineal mediante el producto de un kernel local y un kernel global usado para aprender características locales arbitrarias; el objetivo del aprendizaje del kernel es aprender conjuntamente los parámetros del kernel y SVM, así el aprendizaje de múltiples kernels locales aprende un kernel diferente; es decir, un clasificador para cada punto en el espacio de características (Jose et al., 2013). La adaptación de métodos globales como los clasificadores árboles de decisión (*Decision Tree, DT*) y Multinomial Naive Bayes (*Multinomial Naive Bayes, MNB*) han sido combinados con el método local kNN. Estos métodos se han propuesto para resolver tareas relacionadas a la clasificación de textos (Puurula and Bifet, 2012) y la clasificación de datos incompletos (Hsu and Srivastava, 2009).

En los trabajos presentados anteriormente, se concluye que el rendimiento de un clasificador que adapta un modelo híbrido reporta mejoras en ciertas tareas de clasificación. Esto puede ser debido a que, en la práctica, entrenar un modelo global con todo el conjunto de datos es lento y tratar múltiples clases no es tan natural como en un método local. Sin embargo, en la vecindad de un pequeño número de ejemplos y un pequeño número de clases, los métodos globales a menudo funcionan mejor que otros métodos de clasificación. Esta combinación es la que hace que un método híbrido mejore el desempeño del clasificador. Hoy en día, se han hecho intentos como los presentados en (Card et al., 2019; Li and Deng, 2018; Papernot and McDaniel, 2018; Vidnerová and Neruda, 2018; Zadeh et al., 2018) para adaptar el esquema LWL en modelos de aprendizaje profundo. En la siguiente sección, se realiza una revisión de la literatura con las adaptaciones del aprendizaje local en modelos de aprendizaje profundo.

Figura 3.1: Métodos de aprendizaje localmente ponderado.

3.2. Métodos de LWL en modelos de DL

Uno de los enfoques más explorados en el aprendizaje profundo (DL, *Deep Learning*) con el aprendizaje localmente ponderado (LWL, *Locally Weighted Learning*) son las CNNs. Los trabajos se enfocan en resolver la tarea de clasificación de imágenes usando ejemplos adversos. Los ejemplos adversos se introducen en (Szegedy et al., 2013) y se refiere a que aplicando una perturbación aleatoria imperceptible sobre una imagen de entrada, la predicción de la red entrenada no es capaz de generalizar correctamente. En el trabajo (Vidnerová and Neruda, 2018) se presenta un método donde se realiza la combinación de una arquitectura de red neuronal profunda (*Deep Neural Networks, DNNs*) y una red de función de base radial (RBFN) para clasificar correctamente ejemplos adversos. A pesar de que se añade el concepto de aprendizaje local en enfoques de DL, este trabajo únicamente define una concatenación de una red neuronal

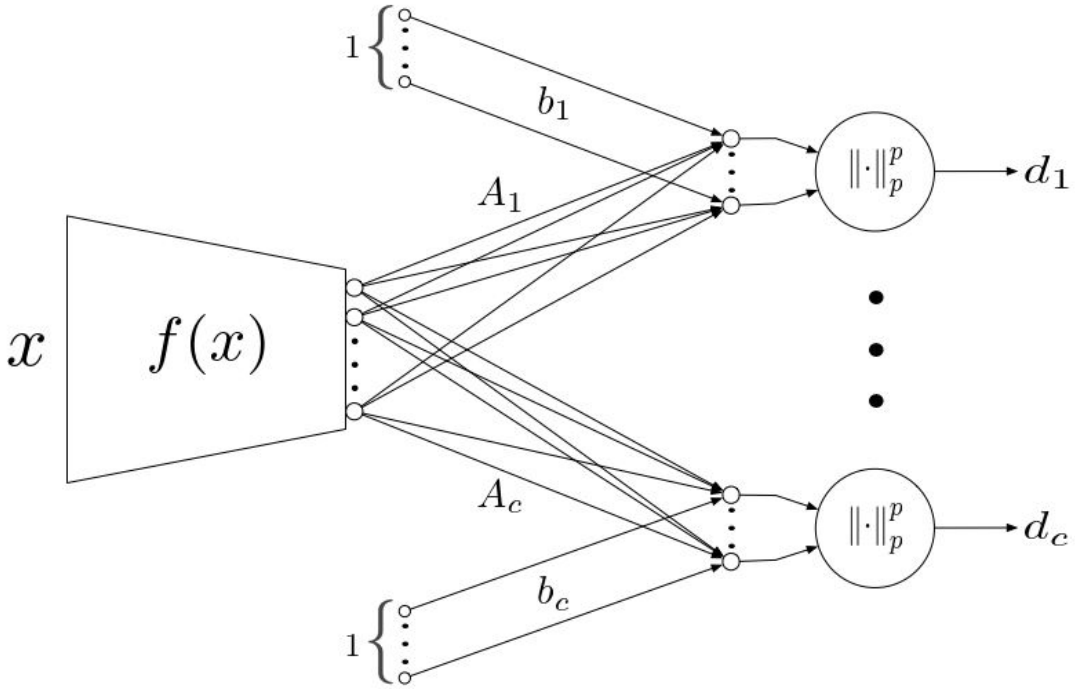


Figura 3.2: *Deep Radial Basis Function, Deep RBF* presentado por Zadeh et al. (2018). La figura ilustra una configuración de red neuronal profunda conectada con una capa de unidades RBF encargadas de calcular la predicción de la clase.

profunda (DNN) y una red de función de base radial (RBFN). Las redes no aprenden de manera conjunta, el entrenamiento es independiente en cada configuración de red.

Un trabajo interesante que adapta una red RBF profunda (*Deep RBF*) se presenta en (Zadeh et al., 2018). El autor propone una CNN y en su capa de salida aplica el concepto de unidades RBF donde se establece una unidad RBF para cada clase (ver Fig. 3.2). La finalidad es sustituir la función *softmax* por las unidades RBF, tomando como salida la unidad RBF a la clase más cercana. Además, se propone una función de costo que se adapta para hacer que la red RBF profunda sea resistente a múltiples ataques adversos, es decir, que generalice correctamente los ejemplos adversos. Una desventaja por considerar en este modelo es que los métodos de aprendizaje local tienden a tener problemas con alta dimensionalidad en los datos y el método propuesto únicamente es evaluado sobre un conjunto de datos que puede ser llevado a un espacio latente de baja dimensión.

El algoritmo K-NN también se ha evaluado en modelos de aprendizaje profundo como una representación del aprendizaje local. En (Papernot and McDaniel, 2018) se presenta una red profunda de los

k-vecinos más cercanos (DkNN). Este clasificador híbrido combina el algoritmo K-NN con representaciones de los datos aprendidos por cada capa de la red neuronal profunda. Sus contribuciones fueron la demostración de la interpretabilidad de la red DkNN, la medida de no conformidad en una predicción y la solidez para identificar ejemplos adversos. El método DkNN adapta una red profunda DNN y obtiene representación de cada capa de la CNN a partir de su salida. Posteriormente, para cada representación calcula los k-vecinos más cercanos; siendo estos los k-puntos de entrenamiento para cada representación de la capa y mide qué tan cercano se encuentra de la muestra de prueba. Así, para cada capa se recolecta una serie de etiquetas asignadas en cada capa y se decide con un esquema de votación a qué clase pertenece. Otro trabajo que adapta el método DkNN es (Sitawarin and Wagner, 2019), este trabajo se basa completamente en el método de Papernot and McDaniel (2018), la diferencia se basa en proponer una heurística para la inicialización del conjunto de ejemplos que se encuentran cercanos al conjunto de entrenamiento.

Otras adaptaciones del aprendizaje profundo que implican técnicas de aprendizaje local son las presentadas por (Card et al., 2019; Li and Deng, 2018). Únicamente se han enfocado en dar una interpretabilidad a las decisiones tomadas por los clasificadores cuando se tienen ejemplos adversos. Estos trabajos presentan la desventaja de que el concepto de localidad no se puede llevar a cabo de extremo a extremo. Dado que se requiere del almacenamiento en memoria de las instancias, donde para los enfoques de DL no es viable, ya que se requiere una gran cantidad de datos para que un modelo de aprendizaje profundo generalice correctamente. Además, en muchos casos, este tipo de aprendizaje no cuenta con una fase de clasificación.

Nótese que el aprendizaje profundo localmente ponderado no ha sido aprovechado para resolver tareas relacionadas al reconocimiento de emociones en imágenes. La adaptación puede ser benéfica para mejorar el desempeño en el reconocimiento de emociones porqué esto es debido a que se explotarían las ventajas de las CNNs que aprenden en conjunto aquellas características visuales que permiten hacer una clasificación correcta de las instancias. El aprendizaje local debe contener una fase de entrenamiento y prueba para ser integrado en la parte de clasificación que adapte criterios de entrenamiento ponderados localmente. El esquema que se propone como LWDL pretende adaptar estos dos enfoques incompatibles, a través de soluciones que permitan integrar el aprendizaje local en la fase predictiva de un modelo de aprendizaje profundo.

3.3. Reconocimiento de emociones

Actualmente, el reconocimiento de emociones (*Emotion Recognition, ER*) en imágenes es una tarea compleja que ha sido ampliamente estudiada con la adaptación de enfoques convencionales de ML o DL. La adaptación de enfoques convencionales de ML para ER incluye tres componentes: la detección facial, la extracción de características y la clasificación de la emoción como se ilustra en la Fig.3.3. En la extracción

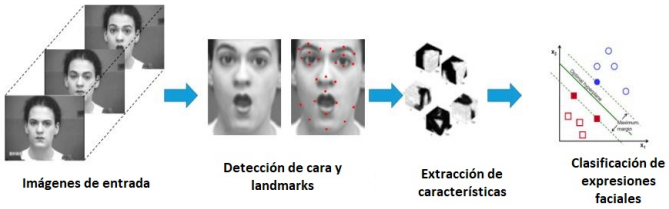


Figura 3.3: Enfoque convencional de ML usado en el reconocimiento de emociones en imágenes de expresiones faciales. A partir de imágenes de entrada (a), se detecta la región de cara y puntos de referencia faciales (b), para extraen de los componentes de la de las características espaciales y temporales de la cara (c) para utilizar algoritmos de clasificación. Figura reproducida de (Ko, 2018)

de características se emplean algoritmos que incluyen: Histogramas de gradientes (HoG) (Gunawan et al., 2015), Patrones Binarios Locales (LBP) (Happy et al., 2012), entre otros (Ghimire and Lee, 2013; Ghimire et al., 2017). En los métodos de clasificación se usan algunos métodos como SVM (Suk and Prabhakaran, 2014), AdaBoost (Ghimire and Lee, 2013) y DT (Wei et al., 2016), por mencionar algunos.

En el reconocimiento de emociones se usan enfoques como las *Redes Neuronales Convolucionales, CNNs* (Breuer and Kimmel, 2017), *Redes Neuronales Recurrentes, RNNs*, *Redes de memoria de corto plazo, LSTM* (Chu et al., 2017a) y métodos híbridos (Ebrahimi Kahou et al., 2015). Por ejemplo, en el trabajo propuesto por (Jain et al., 2019) se presenta un modelo basado en redes neuronales convolucionales (CNNs) donde adaptan módulos residuales en sus capas convolucionales. La red se evalúa sobre los conjuntos de datos Ck+ y JAFFE alcanzando un 95.23 % y 93.24 % de exactitud, respectivamente.

La combinación de redes CNN y LSTM se presenta en (Sepas-Moghaddam et al., 2019), donde el autor combina la red VGG-Face con dos redes neuronales recurrentes tipo LSTM. La red VGG-Face se usa para extraer descriptores de características en las imágenes. Posteriormente, los descriptores se adaptan como secuencias de entrada en cada una las redes LSTM. La clasificación se hace tomando las salidas de celdas LSTM como entradas para un clasificador *softmax*. En el trabajo (He and Zhang, 2018) se presenta

un modelo con dos arquitecturas CNN, una CNN binaria (B-CNN) y una CNN que tiene como objetivo clasificar 8 emociones (E-CNN). La red B-CNN se entrena para crear un modelo que clasifique en imágenes una escena como positiva o negativa. Los pesos obtenidos del modelo B-CNN son usados para entrenar la siguiente red E-CNN. La red E-CNN se encarga de entrenar un modelo que es capaz de reconocer 8 tipos de emociones, usando los pesos de las capas convolucionales de la red B-CNN para generar el modelo E-CNN. Otro trabajo que propone el uso de arquitecturas tipo CNN pero se adapta un preprocesamiento de la imagen es en (Pitaloka et al., 2017). El preprocesamiento consiste en la detección de rostros, cambio de tamaño, adición de ruido y normalización de datos previamente al entrenamiento de la CNN. El autor reporta que se obtuvo una mejora en el FER al adaptar el preprocesamiento. EL método es evaluado usando los conjuntos de datos CK+, JAFFE y MUG.

Para el FER en (Shao and Qian, 2019) el autor propone tres modelos de CNN: Light-CNN, dual-branch CNN y una CNN preentrenada. La red Light-CNN es una arquitectura que consiste en 6 módulos de convolución residuales. La red dual-branch CNN consta de tres módulos: dos módulos de ramificación CNN individuales y un módulo de fusión. La primera rama toma la imagen completa como entrada y extrae las características globales. La otra rama toma la imagen de la característica de textura preprocesada por LBP como entrada. Finalmente, el tercer módulo es una red de fusión que toma como entrada las características globales y de textura. La red preentrenada que se utiliza es ResNet101 y la red está entrenada sobre el conjunto de datos ImageNet. Se aplica la técnica Fine-Tuning para entrenar algunas capas y realizar un ajuste fino en algunas capas para extraer características más específicas. La salida se ajusta de acuerdo con el número de categorías de las emociones. Los tres modelos se entrenan para reconocer 7 emociones y se hace una comparativa entre ellos. El autor concluye que un modelo preentrenado mejora la exactitud en el reconocimiento de emociones. EL método es evaluado con CK+, BU-3DFE y FER2013 alcanzando una exactitud de 85.71 %, 48.17 % y 54.64 %, respectivamente. Los resultados de exactitud alcanzados con este modelo, no muestran una mejora en comparación con los modelos mencionados anteriormente. Además, ningún modelo se evalúa sobre conjunto de datos de ER que contengan emociones compuestas.

Todos estos trabajos han mostrado que los enfoques de DL mejoran el desempeño en el reconocimiento de emociones en comparación con los métodos convencionales de ML. Otro punto interesante es que los métodos usados en ER demostraron que una forma de obtener resultados sobresalientes es mediante la integración de modelos que se enfoquen en el análisis geométrico y visual de la cara. Sin embargo, en los trabajos revisados no se pone tanto énfasis en la etapa de clasificación, que es igualmente importante. En este trabajo nos enfocamos en tratar de mejorar el aspecto predictivo del modelo, mediante clasificación localmente ponderada.

Existe un trabajo basado en el aprendizaje localmente ponderado (LWL) que se enfoca al reconocimiento de emociones y hace una comparativa con un método de aprendizaje global. El trabajo presentado en (Tarnowski et al., 2017) hace una comparativa de exactitud en el reconocimiento de 7 emociones básicas. Los métodos que usa son los k-vecinos más cercanos y una red MLP. El autor llega a la conclusión que un modelo local se ajusta mejor que un método global. Pero el trabajo únicamente se evalúa sobre un conjunto de datos de un modelo 3D de la cara. El hecho de que un modelo local funcione bien, puede ser debido a que sea más sensible a separar los puntos de referencia faciales (*facial landmarks*) sobre un modelo 3D de la cara, ya que directamente realiza cálculos de distancias entre estos atributos y esto permite hacer un agrupamiento favorable de los *facial landmarks*. Por lo tanto, surge la necesidad de proponer un algoritmo basado en el aprendizaje local sobre un modelo de aprendizaje profundo.

Para el reconocimiento de emociones un método local que construya sus límites de decisión basados en la similitud calculada mediante distancias, permitirá hacer una separabilidad adecuada de los patrones dados las variaciones que se presentan en la emoción. El reconocimiento de emociones requiere que los clasificadores hagan una separación de patrones más fina. Por ejemplo, en el caso siguiente: tenemos una variación racial donde hay asiáticos y caucásicos. La forma de distinguir entre ellos emociones es diferente. Los asiáticos tienden mostrar signos tempranos característicos de intensidad emocional con los ojos. Los caucásicos involucran otros músculos faciales y no relacionados exactamente con los ojos Jack et al. (2012). Esto significa que en regiones específicas se brinda información importante para el reconocimiento de cierta emoción y que está ligada la dependencia racial debido a los rasgos. En este caso, se puede pensar que un método de aprendizaje local sería pertinente dada esta y algunas variaciones culturales, ya que se encargaría de construir límites de decisión basándose en el cálculo de similitudes entre las variaciones que se presenten entre emoción.

3.3.1. Discusión

El esquema LWDL se enfoca en adaptar el aprendizaje local en un método de aprendizaje profundo, evitando la problemática que presentan los modelos locales. Generalmente en la revisión previa, las adaptaciones que usan modelos locales en métodos globales tratan de abordar esta problemática con el uso de algoritmos basados en instancias. Este tipo de técnicas tienen la desventaja de no generalizar correctamente sobre conjuntos de datos extensos, ya que requiere almacenar los ejemplos en memoria. Para evitar el uso de métodos basados en instancias, se presenta una idea novedosa que incluye el uso de las redes de función de base radial y son integradas en un modelo de aprendizaje profundo, a pesar de ser un enfoque de LWDL, el concepto no es ampliamente analizado. El enfoque deja de lado la problemática inicial de este

tipo de modelos, el bajo desempeño en conjuntos de datos que representen un espacio latente de alta dimensión. Si hablamos del uso de las redes de función de base radial, encontramos que son la mejor opción por parte del aprendizaje local, debido a que su estructura es similar al perceptrón multicapa (MLP) [1] y la creación de su frontera de decisión se lleva a cabo con el uso de aproximadores locales permitiendo a su arquitectura ser usada en la fase predictiva de cualquier modelo profundo, siempre y cuando mantenga la estructura.

[1] MLP es considerado un modelo de aprendizaje global debido a que su frontera de decisión es creada usado todos los ejemplos de entrenamiento.

4

Aprendizaje profundo localmente ponderado

4.1. Introducción

La parte primordial de la investigación consiste en desarrollar un esquema de aprendizaje profundo localmente ponderado (LWDL) que mejore el desempeño de un modelo de aprendizaje profundo. Para ello, se realizó un estudio exhaustivo de las técnicas que conforman el esquema, es decir, qué elementos debe contener el LWDL, tanto de técnicas de aprendizaje local como global. A continuación, se explica detalladamente los componentes principales del esquema LWDL. Se comienza con una breve explicación acerca de las redes RBF, posteriormente se explica el modelo MB-RBFN que es el esquema LWDL. Así también, se encontrará información acerca de la integración del aprendizaje local en una red CNN y la solución a las problemáticas que enfrenta la integración. Finalmente, se explica la mejora que se hace al modelo MB-RBFN que consiste en añadir una función de costo sobre las ramas de la red basada en la divergencia de Kullback-Leibler.

4.2. Redes de Función de Base Radial

Como se mencionó en la Sección 2.3.3, las redes de Función de Base Radial o Redes RBF son un tipo de red neuronal artificial construida a partir de funciones de kernel espacialmente localizadas, su arquitectura básicamente consiste en una red de tres capas. Es una red simple que contiene una capa de entrada, una capa oculta que contiene unidades RBF y una capa de salida (ver la Figura 4.1). La unidad RBF

es un tipo de neurona que está asociada a un centro y un radio, se puede considerar como un prototipo en el espacio de entrada cuya posición se actualiza o aprende a partir de los datos. Estas unidades se usan comúnmente en redes RBF y modelos basados en LVQ. Generalmente en las redes RBF un conjunto de unidades define una capa y hay una unidad RBF por clase asociada al problema en cuestión (es decir, las unidades RBF a menudo se usan en lugar de las unidades *softmax* para la parte predictiva del modelo). La salida h_i de una unidad RBF i dada la entrada $\mathbf{x}_j$ se calcula de la siguiente manera:

$$h_i = \phi_i(\mathbf{x}_j) = e^{-\left(\|\mathbf{x}_j - \mu_i\|^2 \times (2\sigma i^2)^{-1}\right)} \tag{4.1}$$

donde $\mathbf{x}_j \in R^d$ es un vector $d-$dimensional de características y $\mu_\mathbf{i} \in R^d$, $\sigma_i \in R$ son el centro y el radio de la unidad RBF i, respectivamente.

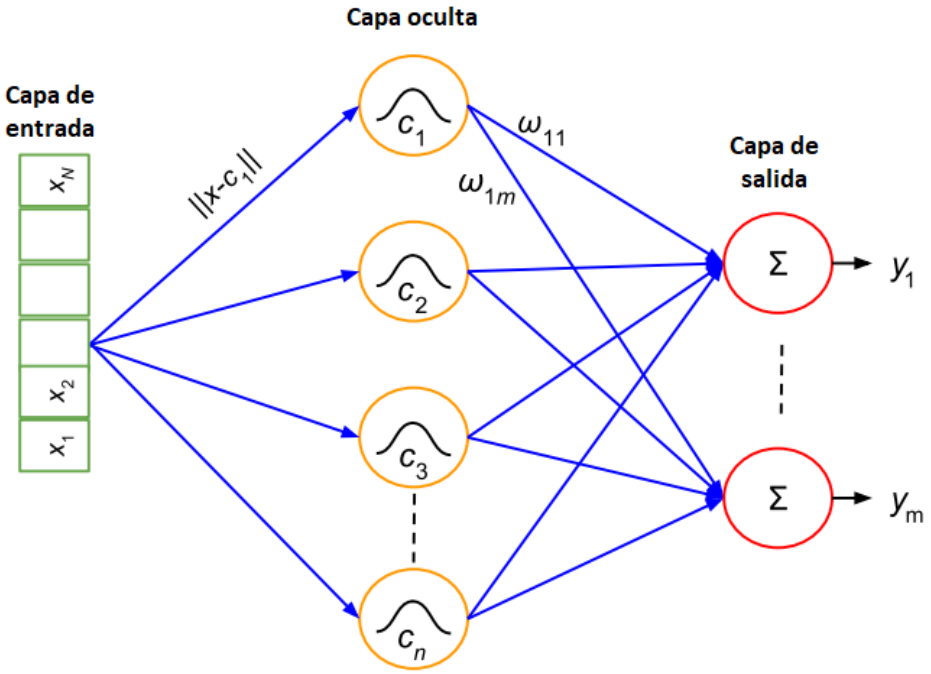


Figura 4.1: Diagrama de una red de función de base radial simple. La Red RBF contiene tres capas, la capa de entrada, la capa oculta o capa RBF y la capa de salida. Figura reproducida de (Faris et al., 2017).

Vector de entrada. El vector de entrada es un vector n-dimensional que debe ser clasificado por la red. Este vector es mostrado a cada neurona RBF de la red y también se le conoce como patrón de entrada.

Neuronas RBF. Cada neurona RBF contine un vector prototipo que es sólo uno de los vectores del conjunto de entrenamiento. Cada neurona RBF compara el vector de entrada con su prototipo y retorna un valor entre 0 y 1, que es una medida de similitud. Si la entrada es igual al prototipo, entonces la salida

de esa neurona RBF será 1. A medida que crece la distancia entre el vector de entrada y el prototipo, la respuesta cae exponencialmente hacia 0. En la Ecuación 4.1 se calcula la distancia euclidiana entre el vector de entrada y el prototipo. Y está ponderada por el centro y el radio de la unidad RBF. El resultado es el exponente de la base exponencial para calcular la salida de la unidad RBF. El valor de respuesta de la neurona también se denomina valor de activación. Al vector prototipo se le conoce como centro de la neurona. La forma de la respuesta de la neurona RBF es una curva de campana.

Los nodos de salida. La salida de la red consta de un conjunto de nodos, uno por categoría o clases a clasificar. Cada nodo de salida calcula una especie de puntuación para la clase asociada. Por lo general, se toma una decisión de clasificación asignando la entrada a la categoría con la puntuación más alta. La puntuación se obtiene calculando una suma ponderada de los valores de activación de cada neurona RBF. Es decir, que un nodo de salida se asocia a un valor de peso con cada una de las neuronas RBF y multiplica la activación de la neurona por este peso antes de propagarlo como respuesta final. Dado que cada nodo de salida calcula la puntuación para una clase diferente, cada nodo de salida tiene su propio conjunto de ponderaciones. El nodo de salida normalmente otorgará un peso positivo a las neuronas RBF que pertenecen a su categoría y un peso negativo a las demás. La mayoría de los enfoques clásicos desplegados en la literatura para entrenar RBFN se realizan en dos etapas. En la primera etapa se determinan los centros y radios usando algoritmos de agrupamiento no supervisados, mientras que en la segunda etapa los pesos de conexión entre la capa oculta y la capa de salida se calculan de una manera tal como un criterio de error usando el error cuadrático medio común (MSE) que se minimiza en todo el conjunto de datos (Faris et al., 2017).

Función de activación de neuronas RBF. Cada neurona RBF calcula una medida de la similitud entre la entrada y su vector prototipo (tomado del conjunto de entrenamiento), generalmente se usa la métrica de distancia euclidiana. Los vectores de entrada que son más similares al prototipo devuelven un resultado más cercano a 1. Hay diferentes opciones posibles de funciones de similitud, pero la más popular se basa en la gaussiana.

El uso de las redes RBF se ha enfocado para resolver problemas de aproximación de funciones, reconocimiento de patrones y predicción de series temporales (Celikoglu, 2006). Tales redes tienen la propiedad de aproximación universal (Park and Sandberg, 1991), surgen naturalmente como soluciones a problemas de regularización (Poggio and Girosi, 1990) y se tratan bien en la teoría de la interpolación (Powell, 1985). Otras ventajas de este tipo de redes es que su estructura simple permite el aprendizaje por etapas, reduce el tiempo de entrenamiento y esto ha llevado a la aplicación de este tipo de redes a muchos problemas prácticos.

4.3. Componentes del LWDL

La estructura que compone el esquema **LWDL** en términos generales consiste en la integración de un método de aprendizaje local en un método de aprendizaje profundo. Para abordar la forma de llevar a cabo el esquema, se estudia el uso de las CNNs como parte de una red profunda y las redes RBF como parte del aprendizaje local. La finalidad es que se mejore la fase predictiva de la red profunda. Cada componente que conforma el esquema LWDL se analiza por separado para identificar los retos en la integración de los métodos, para ello se propone una serie de soluciones que a continuación son analizadas.

Comenzamos analizando acerca del aprendizaje localmente ponderado (LWL) y nos encontramos en dos variantes importantes, una consiste en métodos basados en instancias y otra en métodos incrementales. Los métodos basados en instancias (como se mencionó en la Sección 2) son métodos que mantienen todas las instancias de entrenamiento guardadas en memoria; esto presenta una desventaja cuando se tiene un conjunto de datos muy extenso, dado que se requiere mayor recurso computacional para hacer una predicción. En cambio, los métodos incrementales cuentan con una fase de entrenamiento y prueba; la construcción de su modelo se basa en crear límites de decisión con base a instancias relacionadas o muy cercanas entre sí.

Por lo tanto, de manera importante, se concluye que un *método incremental* se ajusta mejor para hacer una adaptación del aprendizaje local de extremo a extremo en un enfoque de DL; debido a que los métodos de DL tienden a generalizar mejor y evitar el sobre-ajuste cuando se entrena en conjuntos de datos muy grandes y es intratable un método basado en instancias.

Una de las formas de abordar el esquema *Locally Weighted Deep Learning, LWDL* (como se muestra en la Figura 4.2), es mediante la integración del enfoque tipo CNN, en combinación con algunos métodos de LWL como: las Redes de Función de Base Radial (RBFN) o adaptando aproximadores locales basados en algoritmos de agrupamiento (*Clustering*), e incluso utilizando clasificadores basados en distancias ponderadas (*Distance Weighted Learning*).

El enfoque propuesto de LWDL adapta el uso de las redes neuronales convolucionales con las redes de función de base radial. Se elige el uso de las redes de función de base radial debido a que contienen neuronas que funcionan como aproximadores locales y son semejantes a las redes MLP. La ventaja de las redes RBF es que tienen una estructura semejante a las redes MLP y permiten adaptar su fase

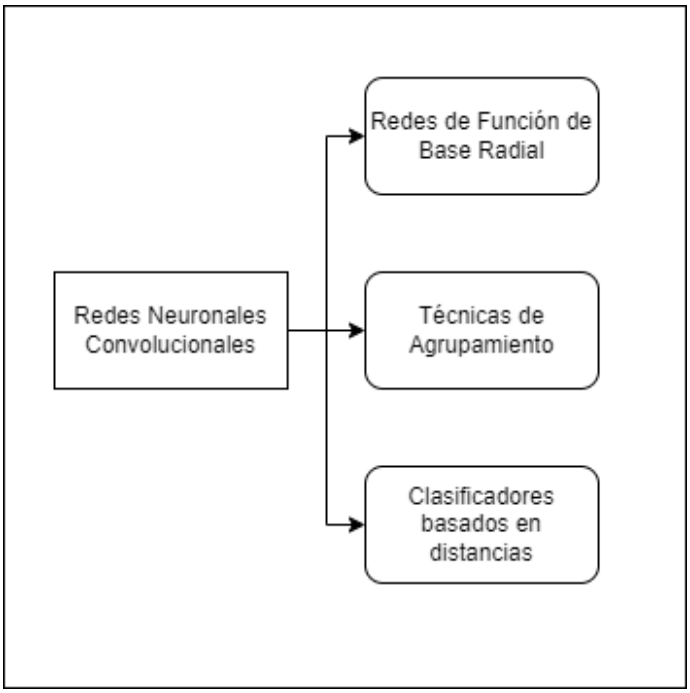


Figura 4.2: Métodos de aprendizaje localmente ponderado (LWL) que pueden ser adaptados sobre un enfoque de DL como las redes neuronales convolucionales (CNNs).

predictiva en un modelo profundo; dado que la fase predictiva de una CNN contiene una capa totalmente conectada que permite aproximar la salida de la clase a través de unidades *softmax*. Así, tales unidades pueden ser directamente reemplazadas por neuronas RBF que calculen la predicción de la clase y de esta forma se añade el término de localidad en una red profunda.

Actualmente, ya existen trabajos que integran el aprendizaje local en una red CNN (Papernot and McDaniel, 2018; Vidnerová and Neruda, 2018; Zadeh et al., 2018). La forma directa que se hizo es añadiendo en la última capa de la CNN unidades RBF que calculan la predicción de la clase; al modelo existente se le nombró red profunda de función de base radial (*Deep Radial Basis Function Network, Deep-RBF Network*). En la Figura 4.3 se muestra el diagrama propuesto de una *Deep-RBF Network*.

La idea del modelo toma la ventaja qué una CNN ofrece y es que contiene dos fases: la extracción automática de características visuales y la fase de clasificación. La red básicamente aprende características conforme a un perceptrón multicapa (MLP) que se considera son las capas totalmente conectadas de la CNN (considerada parte de la fase de clasificación). Las redes MLP funcionan globalmente, es decir, la salida de la red es decidida por todas las neuronas, a diferencia de las redes RBF donde la salida está deter-

minada por unidades especializadas en ciertos campos receptivos locales.

La configuración inicialmente se puede encontrar en el estado del arte, una propuesta hecha por Zadeh et al. (2018) llamada *Deep RBF*. La red añade una capa RBF totalmente conectada antes de la capa de salida. La capa RBF se encarga de recibir las características desde el *flatten* y la idea es que agrupe las características usando el concepto de localidad. En la Figura 4.4 se muestra la arquitectura propuesta por Zadeh et al. (2018).

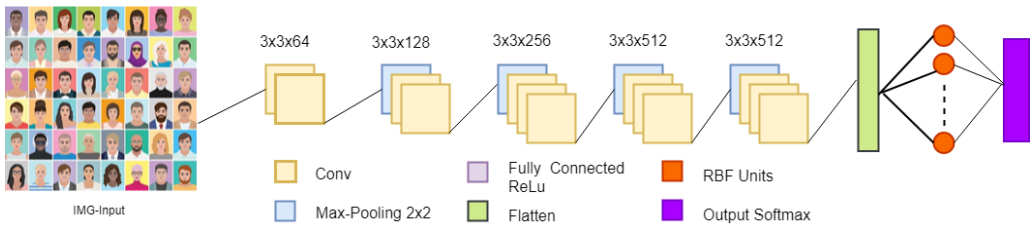


Figura 4.3: Descripción general una arquitectura de DL que integra el aprendizaje local de extremo a extremo usando unidades RBF. La arquitectura muestra una red tipo CNN donde en la fase predictiva se incluye el uso de una capa RBF, como parte de la integración del aprendizaje local en un método profundo.

La diferencia del primer esquema del aprendizaje local de extremo a extremo presentado por Zadeh et al. (2018) y el diagrama presentado en la Figura 4.3; es que Zadeh et al. (2018) usa el aprendizaje local para calcular la predicción a la clase a la que pertenece la instancia, es decir, en la capa de salida se adaptan unidades RBF que conforme el tamaño de las clases es el número de unidades RBF que se añaden a la red. Se elige como la salida el argumento mínimo de la neurona, significa que toma la clase que tenga la mínima distancia.

Si se desea integrar aproximadores locales en capas intermedias, como se explicó anteriormente se necesita tratar el tema de la alta dimensionalidad del espacio latente, obtenido en la fase de extracción de características de la CNN. La idea de añadir unidades locales entre capas intermedias es novedosa. Si conectamos directamente los aproximadores locales estos podrán no funcionar correctamente ante la alta dimensionalidad en el espacio latente; para ello, es necesario proponer técnicas que den solución a esta problemática. A continuación, se explican algunas técnicas que son propuestas para lidiar con la alta dimensionalidad en el espacio latente.

La problemática que se presenta al adaptar el aprendizaje local en una red profunda consiste en:

- Construcción de las unidades RBF. La definición de los centros de las unidades RBF y sus respectivos

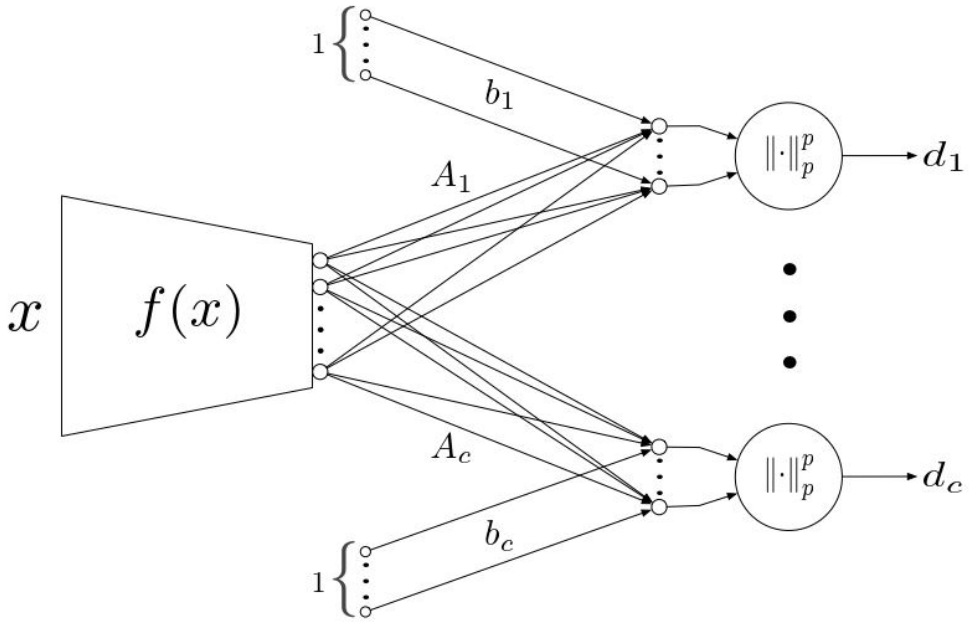


Figura 4.4: Descripción general una arquitectura de DL que integra el aprendizaje local de extremo a extremo usando unidades RBF y una capa intermedia totalmente conectada. Figura reproducida de Zadeh et al. (2018).

radios.

- Reducir el espacio latente de alta dimensión obtenido por la red neuronal convolucional.

En la literatura se han planteado algunos métodos para resolver la problemática mencionada anteriormente y para ello se proponen las siguientes técnicas:

Construcción de las unidades RBF. La construcción de las unidades RBF se puede llevar a cabo de dos formas:

1. A prueba y error: Las unidades RBF se pueden crear estableciendo algún número de unidades RBF ocultas, inicializando con valores aleatorios los centros y radios.
2. Algoritmos de agrupamiento: Se determinan las unidades RBF mediante la creación de prototipos. Los prototipos usan algoritmos de agrupamiento para construir los centros y radios a partir del conjunto de datos de entrenamiento.

El uso de algoritmos de agrupamiento para la construcción de las unidades RBF en un esquema LWDL no es óptimo debido a la gran cantidad de datos que se maneja en una red profunda. Generalmente,

las redes profundas usan conjuntos de datos extensos para generalizar correctamente, al ser así, la cantidad de características procesadas por la red profunda crece y se convierte en un problema que puede ser complejo a la hora de determinar los centros en conjunto de datos, ya que su espacio latente es de alta dimensión. Lo más conveniente para el esquema LWDL es establecer un valor aleatorio para los centros y que conforme a su entrenamiento, el centro vaya convergiendo hasta que se ajuste a un valor correcto; garantizando la minimización de la distancia entre el centro y el vector prototipo.

Los algoritmos de agrupamiento que se pueden usar para determinar los centros son:

- Agrupación espacial basada en densidad de aplicaciones con ruido (*Density-Based Spatial Clustering, DBSCAN*): El algoritmo determina un número de kernel basado en la densidad de la muestra con sus vecinos más cercanos. Estos kernels son usados para construir el agrupamiento. Se utilizarían estos kernel como el centro de cada unidad RBF y se obtendría el radio como la distancia del kernel al vecino más lejano que es considerado como el punto kernel.
- *K-Means Clustering*: El número de unidades RBF será proporcional al número de centroides del algoritmo de agrupamiento y será inicializado en el mismo punto. El radio será igual a la distancia del centroide al elemento más lejano del agrupamiento.
- *Mean-Shift Clustering*: El algoritmo puede determinar el número de clúster o ser un valor establecidos como parámetro, de igual forma que los demás métodos de agrupamiento, los centroides son usados para inicializar las unidades RBF.

Reducción de dimensionalidad del espacio latente. Es crucial trabajar en un problema importante en la creación del esquema LWDL y es la reducción de la dimensionalidad del espacio de características. Si no se reduce la dimensionalidad del espacio latente a ser procesada por el esquema LWDL, esto limitaría su uso al resolver problemas de clasificación simples donde el espacio sea menor. Para ello se propone la reducción de dimensionalidad a través de dos técnicas:

- **Autoencoders**: En el aprendizaje profundo, se puede utilizar un tipo de red neuronal llamada *autoencoders*. Los *autoencoders* o auto-codificadores son redes neuronales que se pueden usar para reducir los datos en un espacio latente de baja dimensión al apilar múltiples transformaciones no lineales. Esta reducción se puede llevar a cabo dentro del modelo de aprendizaje profundo de extremo a extremo. El espacio latente de baja dimensión se puede utilizar como entrada de la capa con aproximadores locales, para crear un aprendizaje local de extremo a extremo en un enfoque de aprendizaje profundo.

- **Multibranch Deep Radial Basis Function**. Idea propuesta para desarrollar sobre el esquema LWDL.

Multibranch Deep Radial Basis Function. Es la contribución principal del esquema LWDL y consiste en demostrar que la idea propuesta permite lidiar con la alta dimensionalidad en una CNN a través de la adaptación de módulos de unidades RBF al final de las capas convolucionales. En la Figura 4.5 se presenta una de las arquitecturas propuesta del esquema LWDL con múltiples ramas en una CNN. La red reduce la dimensionalidad del espacio latente al adaptar múltiples conexiones ponderadas por aproximadores locales en los mapas de características. Los aproximadores locales consisten en múltiples módulos RBF que se conectan a un subconjunto de los mapas de características para hacer un aprendizaje local de las características extraídas. En esta configuración especialmente se usan 16 módulos RBF de la última capa convolucional (el tamaño de 16 módulos se eligió arbitrariamente, en principio el número de módulos no depende del tamaño ni tipo de datos de entrada). Inicialmente está diseñado para que reciba como parámetros la red un número n módulos con n unidades. La finalidad de las ramas es que la capa RBF pueda procesar espacios de características de menos dimensión a diferencia si le pasamos el mapa directamente obtenido de la CNN.

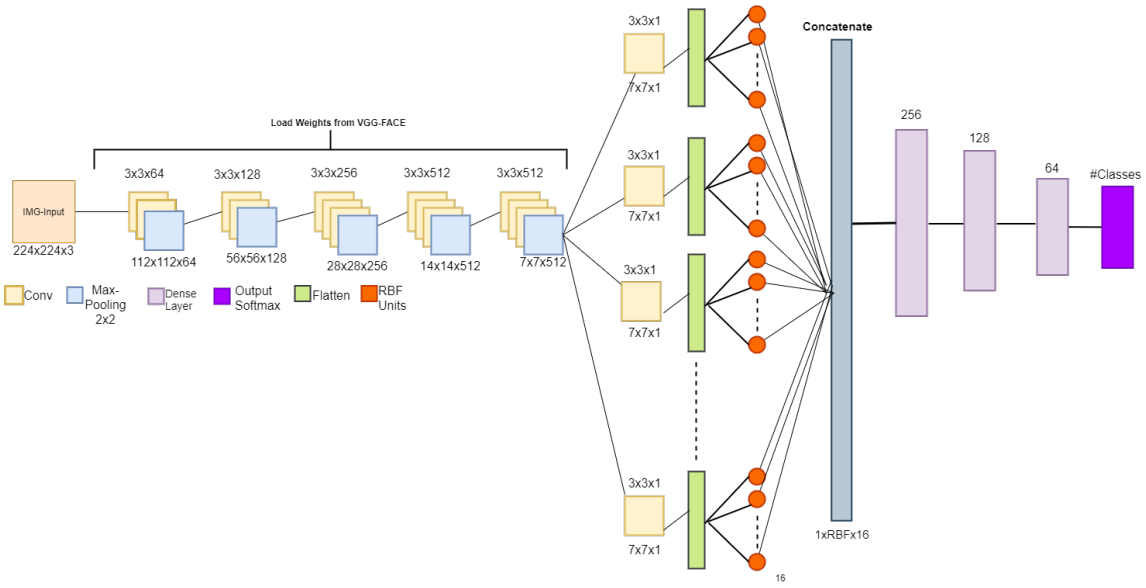


Figura 4.5: Multibranch Deep Radial Basis Function con 16 módulos RBF. En la figura se presenta una arquitectura donde se integra el aprendizaje local a través de multiples ramas con modulos basados en redes RBF. La red presenta una diferencia al agragar multiples capas densas de menor tamaño.

A continuación, se explica la red profunda multicapa de función de base radial que se propone como contribución principal de la investigación doctoral.

4.4. Modelo MB-RBFN

En esta sección se describe el modelo llamado *Multi-Branch RBF network*. El modelo forma parte de la contribución principal de la investigación dado que incorpora el uso de la información local a nivel instancia dentro de un proceso de aprendizaje profundo de extremo a extremo. El modelo usado está basado en una CNN para mejorar el desempeño en el reconocimiento de emociones en la tarea del reconocimiento de expresiones faciales (FER), aunque se realizaron experimentos en otros conjuntos de datos.

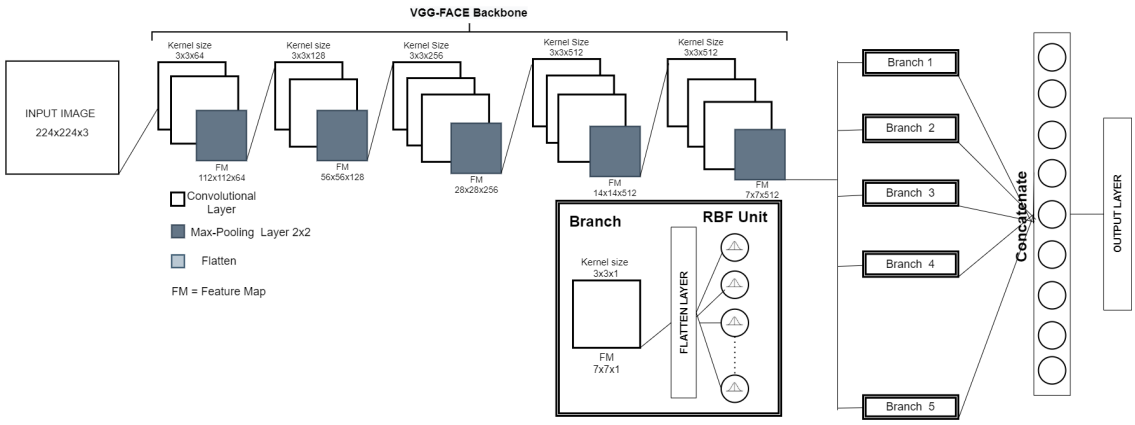


Figura 4.6: Diagrama de la arquitectura propuesta, la red adapta funciones de base radial en múltiples ramas.

La red consiste en una red neuronal convolucional (CNN) con múltiples ramas que adaptan unidades RBF, donde cada rama contiene un módulo RBF. El módulo RBF consta de una capa convolucional donde la entrada es el mapa de características obtenido de la última capa convolucional. La red inicial está basada en VGG-FACE (Parkhi et al., 2015) (como arquitectura *backbone*) conservando las capas convolucionales y cargando los pesos de la red preentrenada; posteriormente, se integran otros modelos *backbone*. La motivación detrás de adaptar múltiples ramas es para resolver la problemática a la que se enfrenta el método con la alta dimensionalidad de los datos ya que las unidades RBF fallan al generalizar, por lo tanto, las ramas se encargan de reducir la dimensión de los datos de entrada hacia las unidades RBF.

La hipótesis de este trabajo se basa en que la incorporación de información local a nivel de instancia en el proceso de aprendizaje de los modelos basados en CNN mejora el rendimiento en la tarea del reconocimiento de emociones mediante expresiones faciales (FER). La intuición detrás de esta hipótesis es que en FER pueden existir grupos de instancias que comparten similitudes entre sí en uno o más aspectos, por ejemplo, en términos de etnia o edad; por lo tanto, se basa en construir la predicción tomando en cuenta

aquellas instancias relacionadas entre sí.

- *La contribución del modelo es que la red contiene múltiples ramas que se encargan de agrupar atributos con una baja dimensionalidad, basándose en la técnica de aprendizaje local con el uso de una capa RBF. Así al tener múltiples capas de RBFs, cada una se encargará de procesar un vector de datos de baja dimensión y podría explotar información local fácilmente.*

En la Fig.4.6 se muestra la red propuesta MB-RBFN y como se puede observar, se elimina la combinación de capas totalmente conectadas con el uso de la capa RBF. Siendo la capa RBF encargada de hacer toda la fase predictiva del modelo, así eliminamos el sesgo que, al combinar con capas totalmente conectadas, éstas puedan encargarse de la predicción del clasificador.

En la construcción del modelo se utiliza una CNN como *backbone* (por ejemplo, VGG-face[1]) y se mejora el desempeño del *backbone* con una nueva capa formada por múltiples ramas de unidades RBF. Dicha capa RBF recibe como entrada mapas de características de las capas anteriores de la CNN y sus salidas son concatenadas y conectadas a una capa de salida *softmax* que realiza las predicciones sobre las clases consideradas. Esta mejora permite que una CNN incorpore implícitamente información local y se cree que pueda tener un impacto positivo en el rendimiento para tareas de clasificación.

El uso de la red VGG-Face como red de base o (*backbone*) es porqué es una CNN bien conocida y lo suficientemente genérica para el análisis facial que ha demostrado ser muy útil en FER y tareas relacionadas cuando se usa como modelo preentrenado; aunque cabe mencionar que no es una arquitectura de vanguardia. La decisión de usar un modelo genérico radica en que se requiere probar que el adaptar el aprendizaje local podría conducir a mejoras con un modelo estándar. Si se usan modelos más complejos o elaborados haría más complicado evaluar la mejora real debido a los módulos locales. VGG-Face es una arquitectura formada por una serie de capas convolucionales seguidas de capas totalmente conectadas que a su vez son seguidas por una capa *softmax* encargada del proceso de clasificación (Parkhi et al., 2015).

Para adaptar el aprendizaje local se modifican las últimas capas de la arquitectura de la siguiente manera: quitamos todas las capas completamente conectadas y en su lugar conectamos varias ramas de unidades RBF a la salida de la última capa convolucional. Dicha capa convolucional (ver Figura 4.6) devuelve como salida 512 mapas de características de dimensión 7×7 (se realizaron experimentos con otros

[1]La red inicial está basada en VGG-FACE (Parkhi et al., 2015) (como modelo *backbone*) conservando las capas convolucionales y cargando los pesos de la red preentrenada; posteriormente se integran otros modelos *backbone*.

valores ver Sección 5.1). Tomamos la activación de estos mapas como entradas a las múltiples ramas de RBF.

La motivación detrás de tener múltiples ramas de RBF es qué si se tuviera un sólo RBF, la entrada tendría una dimensión muy alta (es decir, $7 \times 7 \times 512 = 25088$) y la información local potencialmente útil se perdería o sería muy difícil de procesar. Por el contrario, tener múltiples ramas de RBF, cada una de las cuales toma como entrada un vector de baja dimensión, podría llevar a explotar fácilmente la información local. De hecho, se esperaría que cada rama pudiera capturar un patrón local diferente del resto (ver Sección 5.1.2). Por lo tanto, se propone procesar las salidas de la última capa de convolución de tal manera que cada rama tome una entrada de tamaño manejable. Específicamente, procesamos la última capa convolucional con un conjunto de filtros que producen una salida de 7×7 y usamos tantos de estos filtros como ramas se consideran en el modelo (ver Figura. 4.6). En las salidas se aplica una concatenación y un *flatten* para alimentar a las ramas de RBF en un orden fijo. Es importante mencionar que la mejora implementada en la red se ve reflejada en la evaluación cuantitativa del modelo. La comparativa que se hace es para medir el desempeño de la red es calculando los parámetros de la red y los *FLOPS* de las redes VGGFACE y MB-DRBF. Los resultados obtenidos para MB-DRBF demuestran que la mejora es significativa ya que para VGGFACE se entrenan alrededor de 145M de parámetros y para MB-DRBF 20M, mostrando una reducción de alrededor de 125M. El número de FLOPS alcanzados por VGGFACE son 19.6 MFLOPs y 9.8 MFLOPs para MB-DRBF, la diferencia entre los dos modelos es bastante significativa que rondan alrededor de los 9.8MFLOPs, esto demuestra cuantitativamente que el uso de las multi-ramas mejora el desempeño de la red.

Los centros de las unidades RBF y sus correspondientes radios son inicializados aleatoriamente. Posteriormente, el modelo es entrenado de extremo-a-extremo usando el algoritmo de retropropagación y el gradiente descendiente estocástico (ver detalles en la Sección 5.1) utilizando conjuntos de datos de referencia comúnmente usados en FER. Así también, se llevaron a cabo diferentes tipos de experimentos en los cuales incluyen el congelamiento de capas y la actualización de pesos sobre toda la arquitectura durante el entrenamiento. En el experimento se pudo observar que no existe una diferencia significativa en el desempeño del modelo cuando se actualizan los pesos en la arquitectura convolucional. Por lo tanto, se decide congelar los pesos en las capas convolucionales de la arquitectura VGG-Face y aprender únicamente el resto de los parámetros de la red. Cabe mencionar que la red está diseñada para contener un número mayor o menor de ramas a las mostradas en la Figura.4.6

Los parámetros adicionales introducidos por el modelo basado en RBF para el *backbone* de VGG-Face son las unidades y el radio de las unidades RBF (μ_i y σ_i). Estos parámetros se ajustan durante el

proceso de aprendizaje. Así mismo, los hiperparámetros adicionales son el número de ramas y el número de unidades por rama, la función de pérdida usada es la *categorical crossentropy*, el número de *batch* o lotes es igual a 32 y el número de épocas es igual a 100. Estos últimos deben ser dados por el usuario y dependen del conjunto de datos en cuestión. En la Sección 5.1.1 se muestra un estudio de ablación sobre estos dos hiperparámetros. Intuitivamente, una gran cantidad de ramas daría como resultado unidades más diversas que podrían capturar diferentes patrones, mientras que demasiadas unidades en cada rama pueden llevar a capturar información desequilibrada entre las unidades (consulte la Sección 5.1.2 para un análisis cuantitativo).

4.5. Modelo MB-RBFN con divergencia de Kullback-Leibler (KL)

En esta sección se describe una mejora propuesta para el modelo MB-RBFN. La mejora que se propone pretende mejorar el desempeño de la fase predictiva del modelo MB-RBFN, para ello, se plantea añadir una función de pérdida basada en la minimización de la entropía en cada una de las ramas de la red a través de la divergencia de Kullback-Leibler (KL). La idea propuesta mencionada en (Ghasedi Dizaji et al., 2017) añade la minimización de la entropía para mejorar su arquitectura que está basada en un autoencoder.

La divergencia KL se utiliza en este caso para medir la disimilitud entre dos distribuciones, la información obtenida en cada rama P y la información de la variable objetivo Q. En este caso lo que se trata de minimizar la distancia entre el vector embebido de características de cada rama y la salida esperada (que se propone como un vector característico que representa a cada clase y se considera la variable objetivo o *target* Q).

Para definir la función objetivo enfocada a la minimización de la entropía a través de la divergencia KL, se emplea una variable objetivo auxiliar Q para refinar las predicciones del modelo de manera iterativa. Para hacerlo, primero es usa la divergencia de Kullback-Leibler (KL) para disminuir la distancia entre la predicción del modelo P y la variable objetivo Q como sigue en la Ecuación 4.2.

$$L = KL(Q||P) = 1/N \sum_{i=1}^{N} \sum_{k=1}^{k} q_{ik} \log \frac{q_{ik}}{p_{ik}} \tag{4.2}$$

Para proponer este tipo de función de pérdida es importante añadir términos de regularización. El término de regularización f se calcula como la frecuencia en la cual una instancia es asignada a cierta

clase.

$$f_k = P(y = k) = \frac{1}{N} \sum_i q_{ik} \tag{4.3}$$

Donde f_k puede considerarse como la frecuencia de las asignaciones a la clase en la distribución objetivo. Asumiendo esto, se puede hacer cumplir el tener asignaciones balanceadas agregando el siguiente término en la función KL (ver Ecuación 4.4). Cabe mencionar que esta hipótesis se quiere plantear en cada rama. El primer término de la función minimiza la distancia entre las distribuciones de predicción del modelo, el segundo término equilibra la frecuencia de las clases en las asignaciones. Para ello se añade un término U que representa una distribución empírica de las etiquetas de la instancia.

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{k} q_{ik} \log \frac{q_{ik}}{p_{ik}} + q_{ik} \log \frac{f_k}{u_k} \tag{4.4}$$

Mientras que el primer término en la función objetivo minimiza la distancia entre las distribuciones de predicción objetivo y del modelo, el segundo término equilibra la frecuencia de las asignaciones a las clases en la variable objetivo.

Así se trata de garantizar tener predicciones más equilibradas. El problema para inferir la variable objetivo Q tiene el siguiente objetivo:

$$\min_Q \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{k} q_{ik} \log \frac{q_{ik}}{p_{ik}} + q_{ik} \log \frac{f_k}{u_k} \tag{4.5}$$

A través de un análisis matemático, como se demuestra en (Ghasedi Dizaji et al., 2017), se deriva la función objetivo (ver ecuación 4.5) para para demostrar que se puede resolver el problema con métodos como el gradiente descendiente y puede ser propagado el error mediante *backpropagation* y la función de pérdida se reduce a la siguiente función objetivo:

$$\min_Q -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} q_{ik} \log p_{ik} \tag{4.6}$$

Como se puede observar en la Ecuación 4.6, el problema puede ser considerado como la función estándar de la entropía cruzada. Por lo tanto, se aplica la función de la entropía cruzada en cada rama donde los parámetros de entrada de la función de pérdida son el espacio embebido de características obtenidas en cada rama y el vector de etiqueta de la instancia. Esta pérdida en cada rama se añade a la pérdida general de la CNN. Lo esperado de esta función de pérdida compuesta es que se modele mejor el espacio latente en cada rama de la CNN y esto beneficie en mejorar la predicción de la clase. En la Figura 4.7 se muestra un diagrama de la red MB-RBFN-KL.

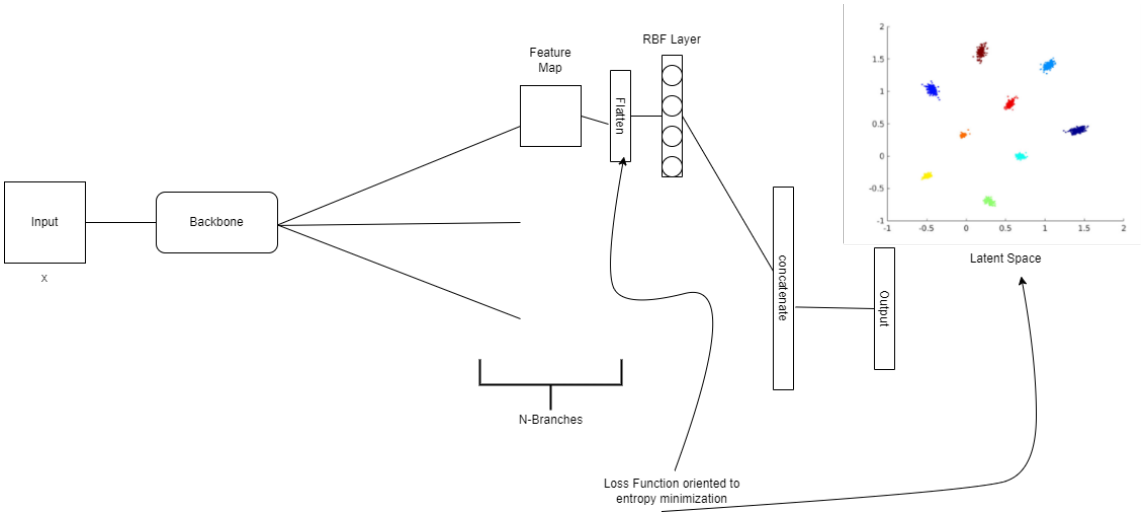


Figura 4.7: Diagrama del modelo MB-RBFN-KL basado en la arquitectura de red de función de base radial profunda de múltiples ramas.

En la Figura 4.7 se muestra una arquitectura de red troncal puede ser reemplazada por cualquier arquitectura de red neuronal convolucional y se mejora con múltiples ramas de unidades RBF que luego se conectan a una capa densa *softmax*. El diagrama demuestra que en el espacio embebido entre el mapa de características y la capa RBF se aplica una función de pérdida que minimiza la entropía y se aplica en cada rama, así cada rama va a contribuir con una pérdida y se sumará a la pérdida total que se toma al final de la red. La gráfica del espacio latente es una representación de la separabilidad entre las clases del espacio latente en cada rama, tal representación se plantea se realice al implementar la función de pérdida propuesta.

A continuación, se explica el algoritmo propuesto para añadir la minimización de la entropía de en la red MB-RBFN-KL a través de la divergencia KL.

4.5.1. Algoritmo minimización de entropía en la red MB-RBFN

Se propone un algoritmo para añadir una variante en la función de pérdida general. La red MB-RBFN-KL inicialmente utiliza una función de pérdida general que mide el desempeño de la red una vez obtenida la predicción de la clase contra una variable objetivo que representa de forma categórica la clase de la instancia. La métrica que se utiliza es la pérdida de la entropía cruzada también llamada *(categorical crossentropy, CE)*. Cabe mencionar que la técnica usada utiliza aprendizaje supervisado en todo momento, por lo tanto, es necesario conocer la etiqueta de la instancia para ser usada como parámetro de las funciones.

Algorithm 1 Algoritmo para función de pérdida en MB-RBFN-KL

Require: Conjunto de entrenamiento D

Ensure: Entrenamiento de MB-RBFN-KL

1: Inicializar los centros de las RBFs de forma aleatoria
2: **for** cada época **do**
3: Obtener el vector objetivo Q
4: Convertir la salida de la unidad RBF a un equivalente de unidades *softmax*

$$P_{ij} = \frac{exp(\frac{1}{|d_i|})}{\sum_{j=0}^{n} exp(\frac{1}{|d_j|})} \tag{4.7}$$

5: Aplicar la función de pérdida que minimice la entropía.
6: Actualizar los centros de las unidades RBF conforme al trabajo de Yang et al. (2017).
7: **end for**

A continuación, se explica con detalle cada uno de los pasos del algoritmo propuesto.

Inicializar centros. La inicialización de los centros de las unidades RBF se llevan a cabo con asignación de valores aleatorios de manera inicial. Se pretende que conforme el entrenamiento avance los centros converjan.

Obtener vector objetivo o Target Existirán tantas unidades RBF como clases en cada rama. Para calcular la pérdida, se calcula primero la variable *Target Q*. La variable Q simplemente está dada con la información de la etiqueta. La variable Q será de forma categórica de tal forma que sea procesada por la función *softmax*.

Convertir la salida de la RBF a un equivalente softmax La salida de la capa RBF se puede interpretar el resultado como la distancia que existe entre la instancia y los centros que representan las clases. Si se quiere aplicar una función de pérdida basada en la entropía cruzada se necesita transformar la salida de las unidades RBF (que es una distancia) a unidades *softmax*. La transformación se puede realizar a través del cálculo del inverso de la distancia y se aplica a la función *softmax*. Si aplicamos el *Inverso distancia* $\rightarrow \frac{1}{|d|}$ el valor de $|d| = 0$ entonces se puede decir que $\frac{1}{|d|} = \infty$. Esto significa que cuando el valor de la distancia entre el centro y la instancia tienda a cero, es decir que está muy cerca al centro, el inverso de esta distancia tiende a infinito. Así cuando apliquemos la función *softmax* esta asignará un valor cercano a 1 para ese caso y 0 para el resto. La Ecuación 4.8 muestra la transformación de la unidad RBF.

$$P_{ij} = \frac{exp(\frac{1}{|d_i|})}{\sum_{j=0}^{n} exp(\frac{1}{|d_j|})} \tag{4.8}$$

Actualización de los centros La actualización de los centros conforme al trabajo de Yang et al.

(2017) establece que el valor del centro nuevo será asignado al valor del centro actual menos una ponderación de las veces que la instancia ha sido asignado a una clase, afectado por la diferencia del centro actual menos el valor de la instancia embebida, la Ecuación 4.9 calcula el valor del centro.

$$C_k \leftarrow C_k - (\frac{1}{f_{ik}})(C_k - F(x_i)) \tag{4.9}$$

Donde:

- C_k es el centro que representa a cada clase.
- k es el número de clases.
- f_{ik} el conteo de veces que la instancia ha sido asignada a tal clase.
- $F(x_i)$ el *flatten*[2] de los mapas de características en cada rama

4.5.2. Función de pérdida para MB-RBFN-KL

Como se mencionó anteriormente, la función de pérdida a la que se redujo es la función estándar entropía cruzada (CE) y se usa para tareas de clasificación. La idea es que cada rama aplique la CE y los parámetros que recibe la función son P_{ij} y Q (la variable target). Recordando que P es la transformación calculada de distancias hacia una representación *softmax*.

La función de pérdida que se propone consta de dos partes: la pérdida total de la red y la pérdida de cada una de las ramas. En la Ecuación 4.10 se muestra el cálculo de la función de pérdida de toda la red.

$$L = L_t + \sum_{i=1}^{n} L_{bi} \tag{4.10}$$

Donde:

- L_t es la pérdida total de la red
- L_b es la pérdida de cada rama
- n el número de ramas

La pérdida total es calculada a partir de la información en la capa de salida de la red y la etiqueta de la instancia. La pérdida total se define como:

$$L_t = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} t_{ik} \log(P_{ik}) \tag{4.11}$$

[2] *flatten* significa que se convierten los datos en un arreglo unidimensional para que sea la entrada a la siguiente capa.

La pérdida de cada rama primero aplica los pasos propuestos en el algoritmo y posteriormente en cada una de las ramas se calcula la pérdida entre el espacio embebido en cada rama y el vector objetivo. La pérdida en cada una de las ramas se define como:

$$L_{b_i} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log(p_{ik}) \tag{4.12}$$

Donde:

- N es el número de instancias.
- K son las clases.
- t_{ik} las etiquetas de la instancia i para la clase k.
- P_{ik} Probabilidad *softmax* para la clase de la red final para cada instancia.
- q_{ikl} variable target de cada rama que también es la etiqueta de la instancia.
- p_{ikl} Cálculo del inverso de la salida de la RBF y transformación a *softmax*.

La función de pérdida compuesta por los dos términos se establece como sigue:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik}\log(P_{ik}) + \sum_{l=1}^{n} Lb_i \tag{4.13}$$

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik}\log(P_{ik}) - \sum_{l=1}^{n}\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log(p_{ik}) \tag{4.14}$$

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik}\log(P_{ik}) - \frac{1}{N}\sum_{l=1}^{n}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log(p_{ik}) \tag{4.15}$$

Después de realizar el análisis matemático, la función de pérdida final queda de la siguiente forma:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}[t_{ik}\log(P_{ik}) + \sum_{l=1}^{n} q_{ikl}\log(p_{ikl})] \tag{4.16}$$

La función de pérdida se aplica sobre el modelo MB-RBFN-KL para mejorar la fase predictiva del modelo, garantizando que se modelará mejor el espacio latente en cada una de las ramas, así la red mejora la predicción de la clase.

4.6. Discusión

El integrar el esquema LWDL implica el análisis de dos enfoques distintos que en esencia no son compatibles. En la Sección 4.3 se revisó la problemática a la que se enfrenta el adaptar el aprendizaje localmente ponderado en una CNN (usada como modelo de aprendizaje profundo), que consiste en añadir técnicas que lidien con el tema del alta dimensionalidad en el espacio latente, que en principio es la información que será procesada por métodos de aprendizaje local. Se concluye que el aprendizaje local no es un buen método para tareas de clasificación cuando el espacio de características es grande.

En la Sección 4.4 se presentó la red *Multi-Branch Deep RBF* como un modelo que mejora las CNNs mediante un mecanismo que le permite incorporar información local. El modelo propuesto se basa en la red VGG-Face como modelo de referencia para la extracción automática de características, donde la última capa convolucional de este modelo está conectada a múltiples ramas de unidades RBF, donde las salidas se concatenan y conectan a una capa *softmax*. El modelo propuesto se inicializa con la red VGG-Face para el reconocimiento de emociones y se ajustan las capas RBF.

Por último en la Seción 4.5 se presentó una mejora sobre MB-RBFN. La idea MB-DRBF-KL consiste en la adaptación de una red profunda tipo CNN y la integración de múltiples ramas, la funcionalidad de las ramas es reducir la dimensionalidad de la entrada a la capa RBF, que funciona como parte de la integración del aprendizaje local. Al utilizar las ramas reducimos los mapas de características que procesa cada capa RBF, y permite que las unidades RBF puedan generalizar de mejor forma.

Anteriormente, se había presentado un modelo que incluye tal descripción, la actualización de sus centros se llevaba de forma automática con los ajustes de la red. La MB-DRBF-KL permite la modificación de los pesos que son asignados en los centros a través de un cálculo que toma en cuenta la instancia evaluada y la distancia a cada representante, se aplica una especie de ponderación a través de la transformación a una función *softmax* al centro o representante más cercano. Además de modificar la forma de actualizar los centros, se construyó una función de pérdida que mide el desempeño en cada una de las ramas a través de la minimización de la entropía. Tal medición toma en cuenta la salida de la capa RBF y la etiqueta de la instancia. La suma de la pérdida en cada rama se añade a la pérdida total de la red.

5

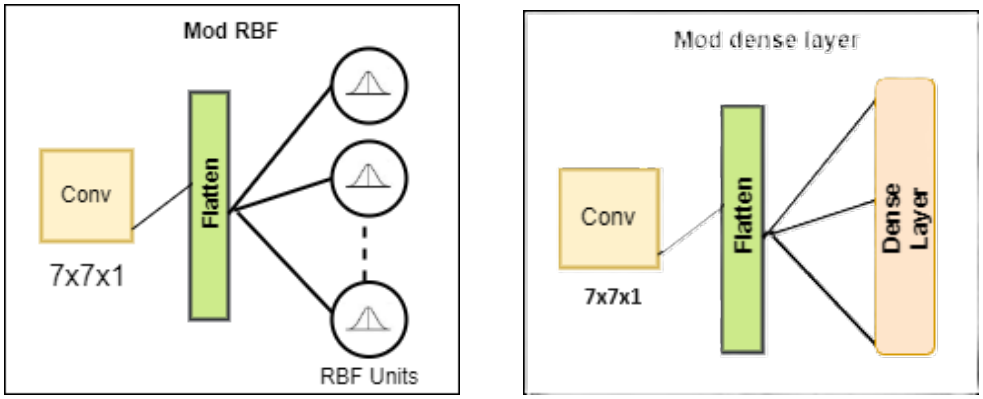
Evaluación Experimental

En esta sección se muestran los experimentos que se llevaron a cabo durante la investigación. Los experimentos están orientados a obtener una comparativa cuantitativa y cualitativa entre modelos de aprendizaje local vs aprendizaje global y del esquema LWDL. En la Sección 5.1 se muestran experimentos importantes realizados con el modelo propuesto MB-RBFN, en el cual se reporta un análisis de ablación y comparativas con el estado del arte para el modelo MB-RBFN. En la Sección 5.2 se encuentran resultados alcanzados con la mejora del modelo MB-RBFN-KL. Finalmente, en la Sección 5.3 se muestran los resultados obtenidos en el reconocimiento de emociones en perros. En el apéndice A se reporta la comparativa de evaluar modelos de aprendizaje local vs global en conjuntos de datos sintéticos que se muestra en la Sección A.1. En la Sección A.2 se implementa una mejora al algoritmo *K-Means* que consiste en adaptar el término radial (basado en como las unidades RBF lo hacen) en la construcción de los centroides a través de una función de pérdida. Otra forma de abordar el aprendizaje profundo con métodos locales es a través del uso de auto-codificadores que integran funciones de pérdida orientadas al agrupamiento que se muestra en la Secció A.3.

5.1. Evaluación experimental del modelo MB-RBFN

El objetivo del experimento es corroborar la hipótesis planteada en la Sección 4.4 que consiste en que si añadimos la técnica de aprendizaje local a través del uso de las redes RBF el modelo de aprendizaje profundo puede mejorar su fase predictiva. Esto con el uso de una red que contenga múltiples ramas que se encargar de agrupar atributos con una baja dimensionalidad.

La finalidad de la evaluación experimental es explicar y demostrar que los resultados alcanzados con el modelo propuesto MB-RBFN superan a los modelos básicos de referencia (que incluyen el aprendizaje global). En particular, comparamos el modelo propuesto con un modelo de referencia que reemplaza las ramas RBF por capas densas; esto se ilustra en la Figura 5.1, para robustecer el análisis. Es importante mencionar que esta comparación permitirá determinar los beneficios reales de tener información local en lugar de unidades completamente conectadas, como es estándar en los modelos CNN.



(a) Modulo con capa RBF.

(b) Modulo con capa densa.

Figura 5.1: Ilustración de las unidades utilizadas en el modelo propuesto (izquierda) y la variante de CNN, CNN de múltiples ramas (derecha). Cada módulo tiene un filtro convolucional seguido de una capa *flatten*. La diferencia de cada módulo es la capa utilizada después de la capa *flatten*: a) usa la capa RBF mientras que b) usa la capa densa con una función de activación ReLU.

Para la comparación experimental, se usan los siguientes conjuntos de datos de referencia que se han utilizado ampliamente en la literatura (ver (Li and Deng, 2020)): Real-world Affective Faces Database (RAF-DB), RAF-DB Compound (Li and Deng, 2019; Li et al., 2017), the Extended Cohn-Kanade Dataset (CK+) (Lucey et al., 2010), the Japanese Female Facial Expression (JAFFE) Dataset (Lyons et al., 1998; Lyons et al., 1998) y FER 2013 (Goodfellow et al., 2013). Además, se realizaron experimentos en un conjunto de datos desafiante que combina los conjuntos de datos CK+ y JAFFE. Las muestras de los conjuntos de datos considerados se muestran en las Figuras 5.2 y 5.3 y algunas estadísticas se presentan en la Tabla 5.1.

Los conjuntos de datos considerados comprenden una diversidad en términos del número de muestras, condiciones de captura de la imagen y complejidad. Los conjuntos de datos CK+, JAFFE, FER2013 y RAF-DB comprenden emociones básicas[1] e imágenes provenientes de la misma distribución. Los conjuntos de datos desafiantes son CK+-JAFFE y RAF-DB Compound, el primero formado por la fusión de imágenes de los conjuntos de datos CK+ y JAFFE, y el último considerando una clasificación de grano fino

[1] Tenga en cuenta que CK+ incluye la emoción *Desprecio* en lugar de la *neutral*, consulte la Figura 5.2.

Tabla 5.1: Estadísticas de los conjuntos de datos considerados para el experimento. Se muestra el número de clases (E) y muestras de entrenamiento, validación y prueba. Hay que tener en cuenta que para CK+-JAFFE el número de clases es 8 ya que incluimos la emoción *Desprecio* que solo está presente en CK+.

Dataset	#Ent.	#Val.	#Prueba	# E
CK+ (Lucey et al., 2010)	877	94	123	7
JAFFE (Lyons et al., 1998)	143	35	35	7
CK+JAFFE	1,020	129	158	8
FER 2013 (Goodfellow et al., 2013)	28,709	3,589	3,589	7
RAF-DB (Li and Deng, 2019)	12,271	3,068	3,068	7
RAF-DB Compound (Li et al., 2017)	3,162	792	792	11

para emociones, consulte la Figura 5.3 .

La intuición detrás de experimentar con el conjunto de datos combinado de CK+-JAFFE radica en que se quiere evaluar el rendimiento del modelo cuando hay diferencias evidentes entre las muestras de la misma categoría.

Figura 5.2: Imágenes de muestra asociadas a diferentes emociones para los conjuntos de datos CK+, JAFFE, FER2013 y RAF-DB. Hay que tener en cuenta que para el conjunto de datos CK+ se muestra una imagen de la emoción *Desprecio* en lugar de la *Neutral*, ya que es el único conjunto de datos sin la última emoción.

Por otro lado, se considera un conjunto de datos desafiante el RAF-DB Compound porque considera categorías de emociones compuestas (p. ej., *Temerosamente sorprendido, tristemente enojado, felizmente disgustado*, etc.), se consideran 11 categorías, vea la Figura 5.3. La idea de considerar este conjunto de datos es mostrar los beneficios de incorporar información local en el proceso de reconocimiento para abordar

Figura 5.3: Imágenes de muestra asociadas a diferentes emociones consideradas en el conjunto de datos RAF-DB Compound.

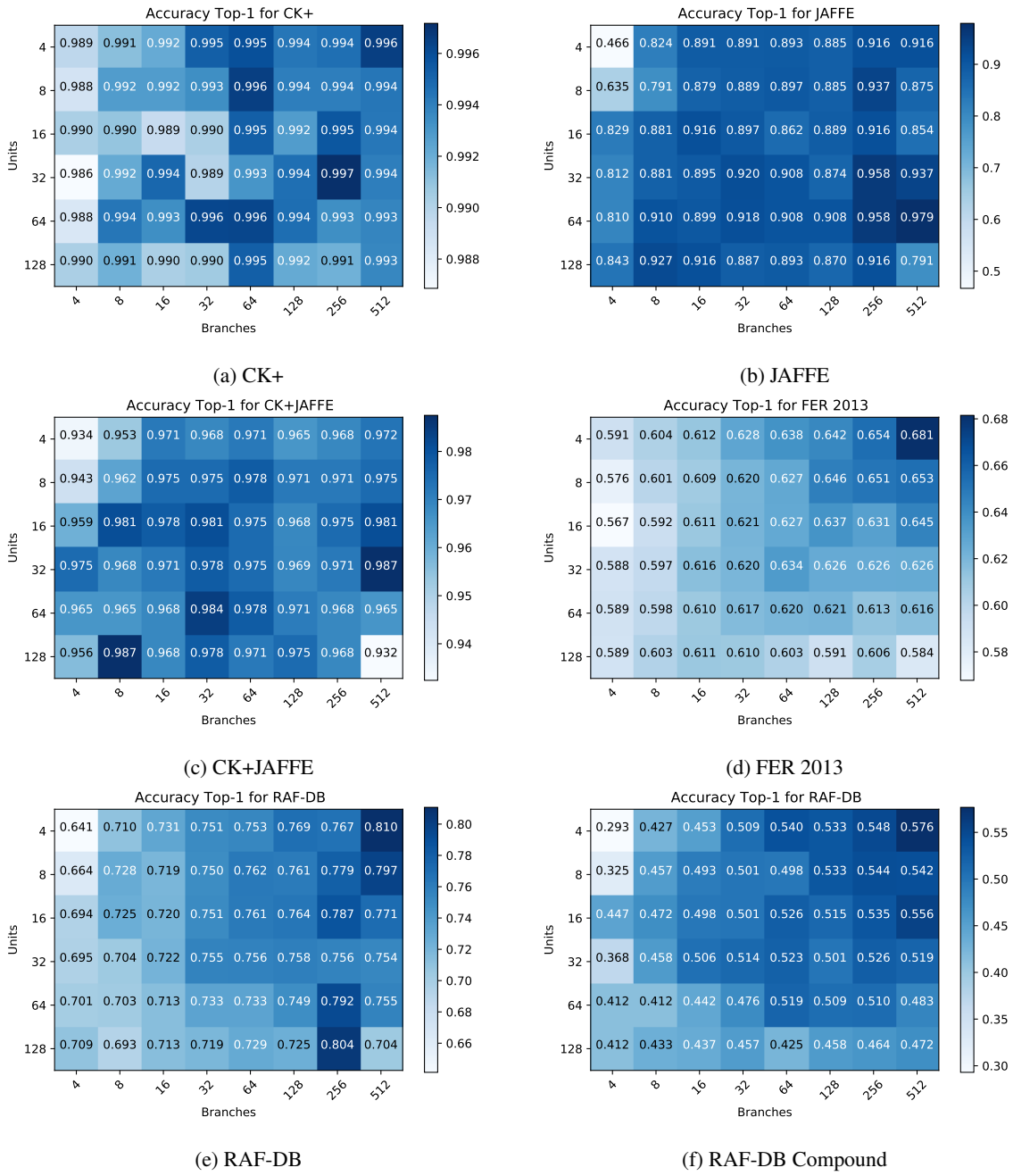
una tarea FER detallada. Se espera que el modelo propuesto sea más ventajoso en los dos conjuntos de datos considerados desafiantes.

Para todos los conjuntos de datos, usamos la métrica *top-1 accuracy* en el conjunto de prueba como medida de evaluación del rendimiento. Esto está de acuerdo con trabajos previos que utilizan los mismos conjuntos de datos. Se usaron las mismas particiones para entrenamiento y prueba en los conjuntos de datos donde estaban disponibles (RAF-DB, RAF-DB Compound y FER2013) y se usaron divisiones aleatorias del 80 % para entrenamiento y validación y del 20 % para prueba CK+ y JAFFE. Para los últimos conjuntos de datos se generaron múltiples particiones y se promediaron sus resultados en cada experimento.

El modelo se entrenó con el optimizador Adam (Kingma and Ba, 2014) con un tamaño de *batch* de 32 durante 100 épocas. El desempeño en la validación se utilizó para monitorear la convergencia del modelo. Determinamos el valor de σ experimentalmente como $\sigma = 0.0528$. El modelo fue entrenado en una laptop con tarjeta Nvidia GTX 2080 con 8Gb de VRAM, y un procesador I7 6700K con 32Gb de RAM.

5.1.1. Estudio de ablación

En esta sección se evalúa el desempeño del modelo propuesto al variar el número de ramas y unidades. En la Figura 5.4 se presentan los resultados de esta evaluación para los seis conjuntos de datos considerados. Los resultados se muestran como mapas de calor (cuanto más oscuro, mejor), el número de



(a) CK+

(b) JAFFE

(c) CK+JAFFE

(d) FER 2013

(e) RAF-DB

(f) RAF-DB Compound

Figura 5.4: Rendimiento del reconocimiento de FER al variar el número de ramas y unidades RBF por rama en el modelo propuesto.

ramas se especifica en el eje x y el número de unidades se muestra en el eje y.

Como puede verse en esta figura, se obtienen resultados mixtos para los diferentes conjuntos de

datos. Siendo el conjunto de datos CK+ el *más fácil* y el RAF-DB Compound el *más duro* en términos de rendimiento de reconocimiento. La diferencia entre el rendimiento más alto y el más bajo logrado para cada conjunto de datos deja en claro que es necesario ajustar adecuadamente estos dos parámetros (p. ej., comparar el rendimiento más alto y más bajo en las Figuras 5.4 (b) y (f)).

Aunque no se puede sacar una conclusión general sobre los valores de los parámetros, un patrón que parece estar presente en todos los conjuntos de datos es que cuanto mayor sea el número de ramas parece dar como resultado un mejor rendimiento del modelo. Además, parece que una pequeña cantidad de unidades RBF combinadas con una gran cantidad de ramas es una combinación de parámetros poco robusta. En general, el rendimiento obtenido en la mayoría de los conjuntos de datos es competitivo con el estado del arte.

5.1.2. Visualización de centros de las unidades RBF

Para el análisis cualitativo del modelo se realiza una visualización de centros RBF aprendidos para dos configuraciones de parámetros del modelo propuesto. En este caso se analiza para el conjunto de datos CK+JAFFE. Se elige este conjunto de datos en particular porque está formado por instancias de dos conjuntos de datos diferentes y esperamos que la información local sea particularmente útil. Además, se debe tener en cuenta que el rendimiento de este conjunto de datos no varió demasiado para las diferentes opciones de parámetros, como se muestra en la Figura 5.4 (c).

La Figura 5.5 muestra los centros para una configuración con 4 ramas y 8 unidades RBF por rama, el rendimiento reportado para esta configuración fue de 0.943 de exactitud. De esta figura se puede ver que los centros a través de las ramas son muy diferentes entre sí. ***Corroborando la hipótesis de que diferentes centros están modelando diferentes aspectos de los mapas de características de entrada***. Es solo para la rama 1 que parece haber similitudes entre los centros (columna 1, filas 4-7 de la gráfica de la izquierda). En general, parece que la información relevante se encuentra cerca del centro de la imagen (valores azules en el centro, amarillo para el fondo), lo que tiene sentido dado que la tarea abordada es ER. Sin embargo, hay algunos centros que también le dan mucha importancia a la región que rodea la cara (fondo azul).

La Figura 5.6 muestra los centros, pero para una configuración diferente: 8 ramas y 4 unidades RBF cada una, con un rendimiento reportado de 0.953 de exactitud. Una vez más, los centros parecen ser visualmente diferentes entre sí, aunque las diferencias entre las unidades RBF de la misma rama (filas, gráfico de la derecha) son menos notorias. Esto podría reflejar el hecho de que las ramas están capturando

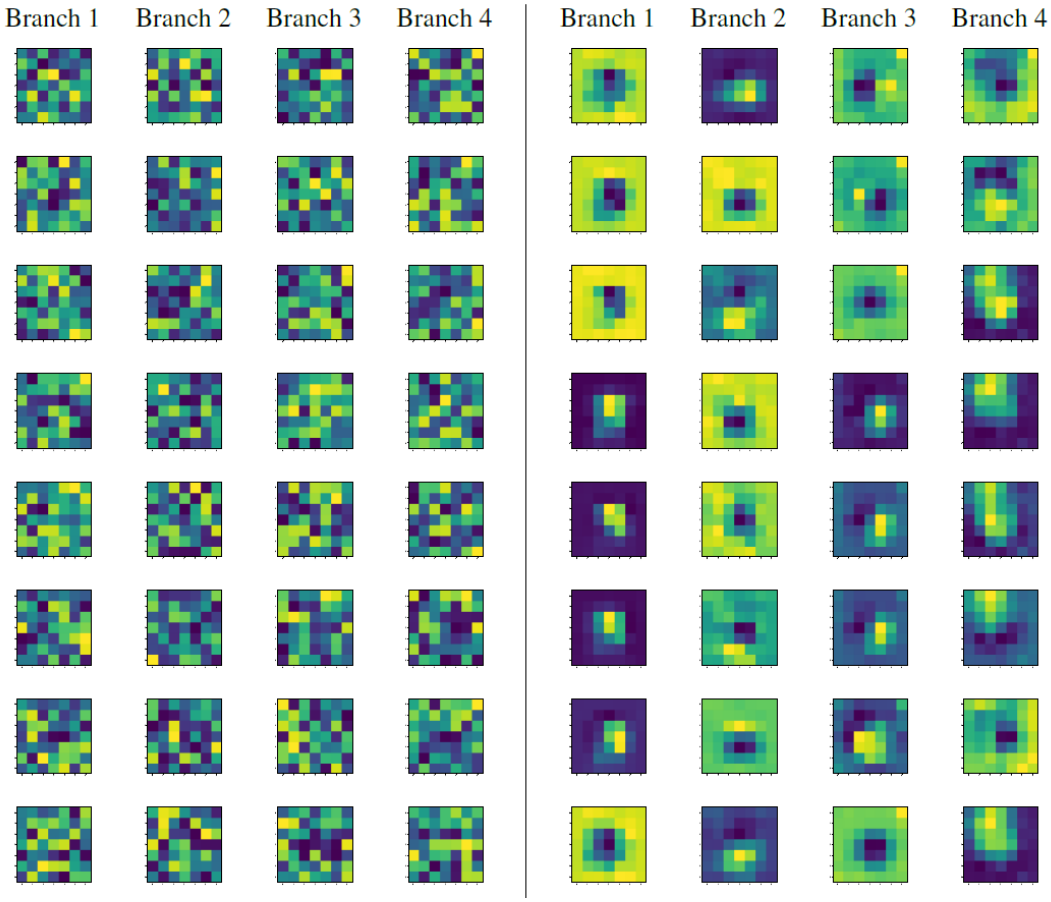


Figura 5.5: Visualización de los centros de las unidades RBF para un modelo con 4 ramas y 8 unidades RBF. El gráfico de la izquierda muestra los centros inicializados y el de la derecha muestra los centros aprendidos después de la convergencia. La ilustración muestra los mapas de características aprendidas por cada unidad RBF en las distintas ramas de la red.

patrones locales con diferencias sutiles entre las unidades RBF (excepto la rama 5, quinta fila en la Figura 5.6 que parece estar aprendiendo el mismo patrón en las 3 unidades RBF). De hecho, este tipo de centros dan como resultado un mejor rendimiento para el conjunto de datos abordado. Finalmente, vale la pena resaltar que en ambos casos los centros parecen converger a una representación útil, partiendo de números aleatorios (gráficos izquierdos en Figuras 5.5 y 5.6).

5.1.3. Comparativa con modelos de referencia

Como modelos de referencia consideramos: (1) el modelo backbone, VGG-Face, la red pre-entrenada ajustada a las nuevas clases, la última capa fue removida y reemplazada por una capa *softmax* con tantas

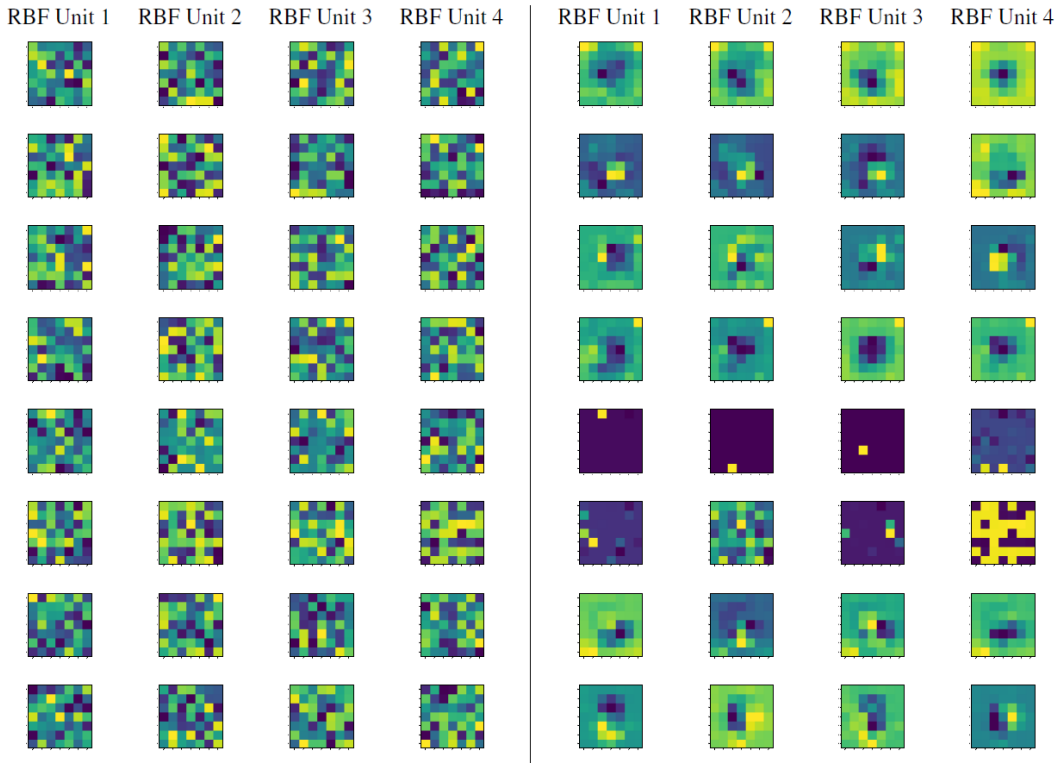


Figura 5.6: Visualización de los centros de las unidades RBF para un modelo con 8 ramas y 4 unidades RBF. El gráfico de la izquierda muestra los centros inicializados y el de la derecha muestra los centros aprendidos después de la convergencia.

unidades como clases; las capas completamente conectadas se someten al proceso de ajuste fino. (2) La CNN de ramas múltiples es un modelo en el que las ramas de las unidades RBF se reemplazan por capas densas (consulte la Figura 5.1). La idea es determinar si agregar parámetros al *backbone* es la causa de la mejora. En general, el objetivo de este experimento es evaluar los beneficios del modelo propuesto en comparación con los modelos de referencia que no incorporan información local.

La Tabla 5.2 muestra una comparación del rendimiento del modelo propuesto con otras variantes[2] de CNN que abordan la misma tarea. Reportamos el promedio y la desviación estándar obtenidas de 10 experimentos con diferente inicialización aleatoria.

En la Tabla 5.2 ***se observa claramente que el modelo propuesto supera a los dos modelos de***

[2]Tenga en cuenta que en la experimentación preliminar también se evaluaron otras variantes de CNN (incluido el modelo utilizado en (Zadeh et al., 2018) y otras configuraciones de CNN que dependían de diferentes procesos de ajuste fino). Sin embargo, reportamos solo las líneas de base más competitivas para la comparación.

Tabla 5.2: Comparativa del rendimiento del modelo propuesto (MB-RBFN) con modelos de referencia (VGG-Face y MB-CNN), se reporta la métrica *Top-1 accuracy* para la tarea de clasificación.

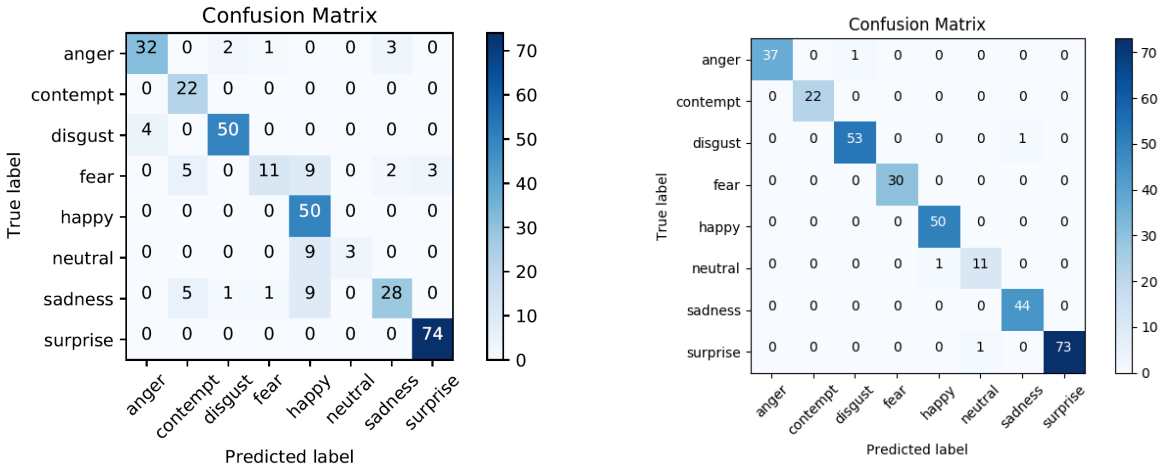
Datasets	VGG-Face	MB-CNN	MB-RBFN
CK+	0.8291 ± 0.003	0.8381 ± 0.051	**0.9964** ± 0.0037
JAFFE	0.6352 ± 0.012	0.5971 ± 0.032	**0.9796** ± 0.0314
CK+-JA	0.8341 ± 0.0018	0.8594 ± 0.021	**0.9872** ± 0.0024
FER13	0.4731 ± 0.035	0.6751 ± 0.0082	**0.6815** ± 0.0097
RAF	0.4289 ± 0.058	0.7237 ± 0.041	**0.810** ± 0.0014
RAF-C	0.2330 ± 0.0012	0.4739 ± 0.0034	**0.5768** ± 0.0074

referencia. Las diferencias en el rendimiento son significativas para la mayoría de los conjuntos de datos. Por ejemplo, el rendimiento de VGG-Face y el modelo propuesto para los conjuntos de datos RAF-DB y RAF-DB Compound, las diferencias en el rendimiento son significativas. Esto podría deberse a la falta de coincidencia entre los conjuntos de datos (tanto en términos de tipo de imágenes como de clases) utilizados para entrenar VGG-Face y los considerados para la evaluación; incluso cuando ajustamos las capas totalmente conectadas del modelo. En la comparativa se puede ver que la línea de referencia MB-CNN supera a VGG-Face en todos los conjuntos de datos excepto en JAFFE. Mostrando evidencia de que las capas añadidas a la arquitectura *estándar* VGG-Face mejoraron el rendimiento del reconocimiento.

Se realizó un análisis a fondo de las diferencias de rendimiento entre VGG-Face y MB-RBFN. La Figura 5.7 muestra las matrices de confusión para VGG-Face y el modelo propuesto en el conjunto de datos CK+-JAFFE. Se puede ver que VGG-Face comete muchos más errores en las categorías *ira, miedo* y *tristeza*. Nuestro modelo no clasificó 4 imágenes del conjunto de datos JAFFE y 6 de CK+, mientras que el modelo VGG-Face cometió 28 y 29 errores para las imágenes JAFFE y CK+, respectivamente. Esto representa el 58 % de las imágenes de JAFFE en el conjunto de prueba y solo el 10 % si se trata de imágenes CK+. Esto ilustra claramente los beneficios de incorporar información local en el modelo CNN: ***las muestras sub-representadas se clasifican mejor*** (se observó un comportamiento similar para el otro modelo de referencia).

Para analizar más a fondo estos errores, la Figura 5.8 muestra imágenes de las categorías *sorpresa* y *miedo*. Siendo sorpresa una de la mejor clasificada por ambos modelos[3] y esta última la clase más difícil para el modelo VGG-Face. Se puede ver en esta figura que las muestras para la categoría *sorpresa* comparten un patrón notable sin importar su origen: la boca está abierta en todos los casos, esto hace que el modelo *generic* VGG-Face clasifique correctamente la mayoría de las instancias de prueba en el conjunto

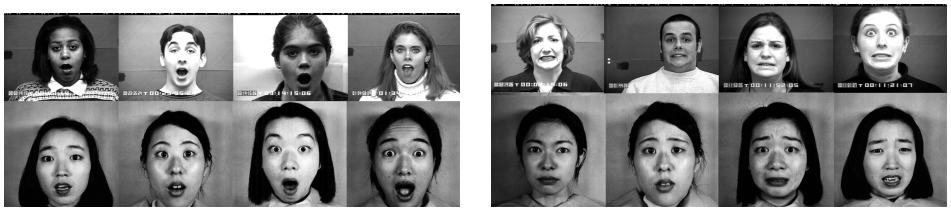
[3]Se incluyen imágenes para *sorpresa* en lugar de *feliz* porque para la última categoría se incluyó una sola imagen de prueba de JAFFE.



(a) VGG-Face (b) MB-RBFN

Figura 5.7: Matrices de confusión obtenidas por VGG-Face y el modelo propuesto para el conjunto de datos CK+-JAFFE.

de datos mixto. Sin embargo, para la categoría *miedo*, las imágenes provenientes de CK+ y JAFFE se ven visualmente diferentes entre sí, pero comparten similitudes dentro de cada conjunto de datos. Esto hace que esta clase sea particularmente desafiante para VGG-Face, mientras que el modelo propuesto puede clasificar correctamente cada instancia de esta clase. Esto podría deberse a la información local incorporada en el modelo, y se puede entender que este es el principal rasgo distintivo de la propuesta.



(a) Surprise (b) Fear

Figura 5.8: Imágenes del conjunto de prueba de CK+-JAFFE para las categorías *sorpresa* y *miedo*. La figura superior muestra las imágenes de CK+ y la inferior muestra las imágenes de JAFFE.

5.1.4. Evaluación del desempeño con diferentes *backbones*

En esta sección se muestra un análisis del desempeño del modelo MB-RBFN cuando se utilizan otros *backbones* en el modelo. El objetivo es mostrar evidencia que la mejora ofrecida por la integración de las múltiples ramas con unidades RBF es evidente cuando se incluye en un modelo pre-entrenado. Para este

estudio, se considera FaceNet (Schroff et al., 2015) y un modelo ResNet18 (He et al., 2016b). FaceNet es una CNN basada en una arquitectura inicial (Szegedy et al., 2014), la red que usamos ha sido entrenada previamente en 10 millones de imágenes faciales del conjunto de datos Celeb-1M. Además, se utiliza un modelo ResNet18 (He et al., 2016b) que fue entrenado en los conjuntos de datos de emociones que se están considerando para la evaluación.

Los resultados es este experimento se muestran en las Tablas 5.3 y 5.4 para los modelos FaceNet y ResNet18, respectivamente. Se reporta los resultados del desempeño obtenido por cada *backbone* para los modelos MB-CNN y MB-RBFN como se describe en la Sección 5.1.3. También se reportan los resultados de una sola ejecución con cada uno de los modelos y para cada uno de los conjuntos de datos como se muestra en la Tabla 5.2.

Tabla 5.3: Clasificación usando la métrica de *Top-1 Accuracy* para los conjuntos de datos considerados. Se realiza una comparativa del rendimiento para el modelo propuesto (MB-RBFN) con modelos de referencia (FaceNet y MB-CNN).

Datasets	FaceNet	MB-CNN	MB-RBFN
CK+	0.9873	0.9873	**0.9949**
JAFFE	0.9375	0.9375	**0.9792**
CK+JAFFE	0.9841	0.9841	**0.9909**
FER 2013	0.5895	0.6227	**0.6400**
RAF-DB	0.8211	**0.8413**	0.8269
RAF-DB C	0.5934	**0.6086**	0.5732

Tabla 5.4: Clasificación usando la métrica de *Top-1 Accuracy* para los conjuntos de datos considerados. Se realiza una comparativa del rendimiento para el modelo propuesto (MB-RBFN) con modelos de referencia (ResNet18 y MB-CNN).

Datasets	ResNet18	MB-CNN	MB-RBFN
CK+	0.9797	0.9822	**0.9889**
JAFFE	0.9166	0.8750	**0.9583**
CK+JAFFE	0.9638	**0.9638**	0.9567
FER 2013	0.5818	0.5798	**0.6004**
RAF-DB	0.7696	0.7702	**0.7734**
RAF-DB C	**0.4886**	0.4621	0.4532

De las Tablas 5.3 y 5.4, se puede ver que el modelo propuesto obtiene el rendimiento más alto en la mayoría de los conjuntos de datos. Sin embargo, esta vez hay dos conjuntos de datos en los que los

Tabla 5.5: Evaluación utilizando la métrica de exactitud *Top-1 accuracy* para la tarea de clasificación en los conjuntos de datos para el reconocimiento de emociones. La comparativa se refiere al rendimiento del modelo propuesto contra referencias del estado del arte.

CK+		**JAFFE**		**FER2013**		**RAF-DB**		**RAF-DB C**	
Ref.	Acc.	Ref.	Acc.	Ref.	Acc.	Ref.	Acc.	Ref.	Acc.
Chen et al. (2019)	0.9806	Li et al. (2020)	**0.9852**	Liang et al. (2020)	**0.7830**	Shi and Zhu (2021)	**0.9055**	Li et al. (2019)	**0.5884**
Minaee and Abdolrashidi (2019)	0.9800	Shima and Omori (2018)	0.9531	Li et al. (2020)	0.7582	Gera and Balasubramanian (2021)	0.8942	Li et al. (2017)	0.5795
Ravi et al. (2020)	0.9732	Minaee and Abdolrashidi (2019)	0.9280	Minaee and Abdolrashidi (2019)	0.7002	Zeng et al. (2018)	0.8690	Li et al. (2017)	0.5354
Wang et al. (2020)	0.9730	Zhang et al. (2020)	0.9238	Mollahosseini et al. (2016)	0.6640	Zhao et al. (2016)	0.8677	Liang et al. (2020)	0.5020
Meng et al. (2017)	0.9537	Videla and Kumar (2020)	0.7810	Shao and Cheng (2021)	0.6617	Liang et al. (2020)	0.758	Kollias et al. (2019)	0.4830
VGG-Face	0.8295		0.6352		0.4731		0.4289	-	0.2330
MB-CNN	0.8381		0.5971		0.6751		0.7237	-	0.4739
MB-RBF	**0.9964**		0.9796		0.6815		0.81	-	0.5758

backbones del modelo propuesto no funcionan tan bien. Con respecto a la red troncal de FaceNet (Tabla 5.3), el MB-RBFN obtuvo un rendimiento más bajo que el método MB-CNN en los conjuntos de datos de la RAF. Esto podría deberse al hecho de que el modelo de FaceNet se entrenó previamente en un conjunto de datos que es 4 veces más grande que el utilizado para el entrenamiento previo de VGG-Face. Por otro lado, cuando se utiliza la red troncal ResNet18 (Tabla 5.4), el MB-RBFN obtiene un rendimiento ligeramente más bajo que el MB-CNN para el conjunto de datos CK+-JAFFE, aunque la diferencia es bastante pequeña. Además, el modelo ResNet18 obtuvo un rendimiento más bajo que la propia red troncal para el conjunto de datos RAF-DB Compound.

5.1.5. Comparativa con el estado del arte

En esta sección, se muestra una comparación entre el desempeño alcanzado por el modelo y los modelos de referencia con el estado del arte. Para esta comparación se usaron los resultados obtenidos por VGG-Face como *backbone*. La razón es que con esta red se realizaron más corridas de experimentos que con otros *backbones*. La Tabla 5.5 muestra una comparativa con los resultados, para cada conjunto de datos considerado. Además, se reporta el desempeño de los recientes trabajos con el estado del arte; aunque cabe mencionar que para este caso se tomó el resultado individual reportado por cada modelo.

De la Tabla 5.5 se puede ver que solo en dos conjuntos de datos: FER2013 y RAF-DB (de los cinco considerados para esta evaluación), el modelo propuesto no logra un rendimiento competitivo con el estado del arte. Curiosamente, estos son precisamente los dos conjuntos de datos con mayor número de muestras con 28.807 y 12.271 respectivamente. Este resultado parece indicar que el modelo propuesto es particularmente útil para conjuntos de datos de tamaño medio-bajo. Además, el desempeño competitivo en CK+ y JAFFE parece sugerir que el modelo funciona mejor en pequeños conjuntos de datos registrados bajo condiciones controladas. Esto es parcialmente cierto ya que el modelo MB-RBFN logra un rendimiento competitivo en RAF-DB-C, que se ha registrado como un conjunto de datos *in the wild*.

Asimismo, dado que las referencias que se comparan se basan en modelos extremadamente complejos, mecanismos a la medida y procedimientos sofisticados, no es extraño que funcionen mejor cuando se dispone de suficientes datos. Como se mencionó anteriormente, el modelo propuesto logra un rendimiento muy competitivo en conjuntos de datos compuestos CK+, JAFFE y RAF-DB. En CK+, los resultados establecen un nuevo resultado de referencia y en los conjuntos de datos compuestos de JAFFE y RAF-DB, el modelo logra un rendimiento comparable. Es notable el rendimiento obtenido por el modelo propuesto en el conjunto de datos RAF-DB Compound, ya que este presenta un problema de clasificación muy fina con superposición entre clases (como se puede ver en las clases *Tristemente Disgustado* y *Tristemente Enojado*, ver Figura 5.9) y muy desequilibrado (4 clases comprenden 72 % de las muestras, y las 7 clases restantes con un 6 % o menos del total de muestras). Este resultado proporciona evidencia adicional de que el modelo es particularmente útil para este tipo de problemas.

Figura 5.9: Imágenes de muestra de dos de las clases en el conjunto de datos RAF-DB Compound, imagen tomada de `http://www.whdeng.cn/RAF/model1.html`.

Es importante remarcar que el modelo es ventajoso en términos de simplicidad, además, es posible que si evaluamos el modelo con un *backbone* más complejo, el rendimiento del modelo RBF de múltiples ramas podría ser incluso superior. Por último un punto a tener en cuenta es que el modelo no realiza ningún proceso de aprendizaje de características *ad hoc*: únicamente se basa en el modelo VGG-Face preentrenado, mientras que la mayoría de las otras referencias comprenden costosos procesos de aprendizaje desde cero o de ajuste fino, que a menudo utilizan datos externos adicionales.

5.1.6. Discusión

En esta sección se presentó una evaluación experimental del modelo de red *Multi-Branch Deep RBF*. Se realizaron experimentos en seis conjuntos de datos ampliamente utilizados para ER, dos de ellos eran variantes que presentaban desafíos particulares con los que luchan los métodos actuales. La evaluación experimental mostró que el modelo propuesto supera considerablemente a los modelos de referencia que incluían un modelo similar formado sólo por capas densas. La comparación con los modelos de referencia junto con una inspección visual de los centros aprendidos constituye evidencia de que la información local capturada por el modelo propuesto es útil para abordar la tarea de ER.

Por otro lado, el modelo propuesto se comparó favorablemente con metodologías recientes que se basan en técnicas y procedimientos mucho más complejas. Este es un resultado sobresaliente dado que el modelo propuesto se basa en un modelo de referencia muy genérico, pero efectivo: VGG-Face. Curiosamente, se demostró que el modelo propuesto ofrece más ventajas en conjuntos de datos con condiciones más desafiantes, a saber: tamaño de muestra pequeño-mediano, con alto desequilibrio de clases, superposición de clases y con imágenes provenientes de dos distribuciones diferentes. Los resultados obtenidos son, por lo tanto, alentadores y comprueban que es benéfico la incorporación de información local en el aprendizaje profundo.

5.2. Experimentos del modelo MB-RBFN-KL

El modelo MB-RBFN-KL se refiere a una mejora hecha al primer modelo MB-RBFN. La mejora consiste en incluir una función de pérdida a la red en las multi-ramas a través del uso de la divergencia Kullback-Leibler. Esta comparación entre distribuciones se da entre el target establecido para cada clase y el espacio latente obtenido en cada módulo RBF de la multi-rama. El objetivo del experimento es medir el desempeño del modelo sobre conjuntos de datos de referencia y compararlo contra la red ya establecida MB-RBFN y el *baseline* de la arquitectura.

5.2.1. Configuración de parámetros

El modelo utilizó los conjuntos de datos de referencia MNIST, Fashion MNIST, Cifar10 y Cifar100, comúnmente usados en tareas de clasificación de imágenes. Los conjuntos de datos tienen un tamaño de 10 clases excepto por Cifar100 que contiene 100 clases. Se usan para reconocer: dígitos, objetos y ropa.

Las imágenes son de baja resolución y tienen un tamaño de 32x32 a color (para el caso de Cifar10 y Cifar100) y de 28x28 a escala de grises (para el caso de MNIST y Fashion MNIST). Los conjuntos contienen un aproximado de 60,000 imágenes, lo cual es un tamaño considerable para medir el desempeño del modelo propuesto. En la Tabla 5.6 se muestra la distribución de los conjuntos en datos divido en datos de entrenamiento y prueba.

Tabla 5.6: Distribución de los conjuntos de datos de referencia divididos en muestras de entrenamiento y prueba.

Dataset	# Muestras de entrenamiento	# Muestras de prueba	# Clases
MNIST	60,000	10,000	10
Fashion MNIST	60,000	10,000	10
Cifar10	50,000	10,000	10
Cifar 100	50,000	10,000	100

El experimento se construye para demostrar que el nuevo modelo tiene un buen desempeño con respecto al modelo anterior (el MB-RBFN) cuando se añade una función de pérdida en cada rama. La propuesta tiene la finalidad de mejorar el modelado del espacio latente en cada rama y así poder separar correctamente las características y distinguir con más exactitud las clases. Para evaluar el nuevo modelo se utilizaron varias arquitecturas. La primera arquitectura contiene una red CNN como red de referencia. La segunda arquitectura utiliza el modelo de MB-RBFN. La tercer arquitectura consiste del modelo con la mejora el MB-RBFN-KL. Se usaron dos arquitecturas como *backbone* para las CNNs y son: VGG16 y ResNET50.

Arquitecturas evaluadas para hacer la comparativa entre modelos:

- VGG16 (Baseline)
- VGG16 con Multi-Rama que incluye capas RBF (MB-RBFN).
- VGG16 con Multi-Rama que incluye capas RBF y función de pérdida para las ramas y actualización de centros de RBF (MB-RBFN-KL).

El experimento se llevó a cabo usando la técnica de *Transfer Learning*, significa que bajo esta configuración una CNN previamente entrenada se le cargan los pesos de las capas convolucionales se congelan para posteriormente realizar un proceso de ajuste fino para minimizar la función de pérdida en el modelo; significa que sólo las capas asociadas con el proceso de clasificación se entrenan, tratando de adaptar el

modelo a cada conjunto de datos específico.

Para el caso de la red de base propuesta MB-RBFN y la mejora MB-RBFN-KL se aplica la misma técnica de transferencia con el modelo previamente para cada *backbone* utilizado y se actualizan los pesos de las ramas. Recordando que para el caso de MB-RBFN los centros de las unidades RBF se actualizan de forma automática conforme al optimizador, en el caso del modelo MB-DRBF-KL se actualizan conforme a un criterio establecido, que en términos generales incluye cuantas veces una instancia ha sido asignada a cierta clase (vea la Sección 4.5.1).

5.2.2. Evaluación del desempeño entre *backbones*

La métrica que se usa para evaluar la clasificación del modelo es la *(accuracy Top-1)*. En términos generales la métrica evalúa la exactitud con la que el modelo ha predicho la clase a cada instancia. La evaluación de MB-RBFN-KL con la arquitectura VGG16 se realiza con todos los conjuntos de datos de referencia que son: MNIST, Fashion MNIST, Cifar10 y Cifar100. En la Tabla 5.7 se observan tres resultados: *baseline* corresponde a la red de referencia entrenada con cada conjunto de datos, *MB-RBFN 8R* se refiere a la red con multi-ramas RBF donde se entrenó con 8 ramas y cada rama contiene tanta unidades RBF como clases existentes en el conjunto de datos. El número de ramas se eligió arbitrariamente y *MB-RBFN-KL 8R* se refiere al mismo caso con las 8 ramas pero sobre el modelo con la mejora integrada.

Los resultados entre los tres modelos como se puede observar en la Tabla 5.7 reporta que MB-RBFN-KL alcanza resultados sobresalientes en la evaluación del desempeño para todos los conjuntos de datos. A pesar de que son conjuntos de datos distintos, la red supera en todos los casos a los modelos que no integran la función de pérdida en las ramas. Incluso no importa si los conjuntos contienen grupos desbalanceados, el tamaño de las muestras es variable, o el número de clases es grande o pequeño.

El siguiente backbone con el que se realizaron los experimentos está basado en la red ResNET50 (He et al., 2016a). ResNET50 está preentrenada sobre el conjunto de datos IMAGENET (Deng et al., 2009), de igual forma se evaluó de las tres maneras usando ResNET50 como baseline y construyendo la MB-RBFN y MB-RBFN-KL.

Los conjuntos de datos que se utilizan son: Cifar10 y Cifar100. La razón es que la arquitectura soporta como mínimo un tamaño de entrada de una dimensión de 32x32x3. Para el caso de MNIST y Fashion-MNIST el tamaño de la imagen es de 28x28x1. La Tabla 5.8 muestra los resultados alcanzados en

Tabla 5.7: Tabla de comparativa de resultados obtenidos entre la evaluación de arquitecturas base VGG16 y arquitecturas que incluyen el aprendizaje profundo. MB-RBFN se refiere a la arquitectura base que contiene múltiples ramas donde cada rama contiene una capa RBF. En el caso de MB-RBFN-KL se refiere a la misma arquitectura que MB-RBFN pero con la diferencia que integra la idea de la función de pérdida por rama y la actualización de centros dinámicamente con cada instancia.

Dataset	Baseline	MB-RBFN 8R	MB-RBFN-KL 8R
Cifar10	0.7108	0.7293	**0.7799**
Cifar100	0.4015	0.4872	**0.5347**
Fashion-MNIST	0.9266	0.9839	**0.9902**
MNIST	0.9937	0.9883	**0.9947**

la evaluación con el *baseline*, MB-RBFN y MB-RBFN-KL. Como se puede observar la mejora MB-RBFN-KL supera a las demás configuraciones demostrando qué sin importar que CNN de referencia se use, si se incluye el término de localidad con ajuste en su función de pérdida en las ramas, la fase predictiva del modelo se ve mejorada. Cabe esperar que al usar ResNET50 incluso mejoró los resultados alcanzados con VGG16, debido a que se habla de una arquitectura más robusta y compleja que contiene conexiones residuales.

Tabla 5.8: Tabla de comparativa de resultados obtenidos entre la evaluación de arquitecturas base ResNET50 y MB-RBFN y MB-RBFN-KL. MB-RBFN se refiere a la arquitectura base que contiene múltiples ramas donde cada rama contiene una capa RBF. En el caso de MB-RBFN-KL se refiere a la misma arquitectura que MB-RBFN, pero con la diferencia que integra la idea de la función de pérdida por rama y la actualización de centros dinámicamente con cada instancia. El backbone ResNET50 es una red que contiene capas residuales en su arquitectura, y demostró ser un buen extractor de características.

	Baseline	MB-RBFN 8 Ramas	MB-RBFN-KL 8 Ramas
Cifar10	0.7632	0.7684	**0.7869**
Cifar100	0.4620	0.4723	**0.5473**

5.2.3. Discusión

La idea MB-RBFN-KL consistió en la adaptación de una red profunda tipo CNN y la integración de múltiples ramas, la funcionalidad de las ramas es reducir la dimensionalidad de la entrada a la capa RBF, que funciona como parte de la integración del aprendizaje local. Al utilizar las ramas reducimos los mapas

de características que procesa cada capa RBF, y permite que las unidades RBF puedan generalizar de mejor forma.

Anteriormente, se había presentado un modelo que incluye tal descripción, la actualización de sus centros se llevaba de forma automática con los ajustes de la red. La MB-RBFN-KL permite la modificación de los pesos que son asignados en los centros a través de un cálculo que toma en cuenta la instancia evaluada y la distancia a cada representante, se aplica una especie de ponderación a través de la transformación a una función SOFTMAX al centro o representante más cercano. Además de modificar la forma de actualizar los centros se construyó una función de pérdida que mide el desempeño en cada una de las ramas a través de la minimización de la entropía. Tal medición toma en cuenta la salida de la capa RBF y la etiqueta de la instancia.

Lo que podemos concluir hasta este momento, es que los resultados preliminares alcanzados por MB-RBFN-KL demostraron ser mejores que cuando solo usamos una red MB-RBFN. Por lo tanto, se demuestra para estos casos, que el aprendizaje local mejora la fase predictiva del modelo para los diferentes *backbones* usados.

El aprendizaje local ha demostrado obtener resultados competitivos, en este momento falta comprobar que la mejora (MB-RBFN-KL) sigue teniendo resultados sobresalientes en los conjuntos de datos que contienen datos con una alta superposición entre clases, conjuntos de datos con distribuciones mixtas en el conjunto de prueba y con altos índices de desequilibrio, emociones, grano fino y mostrar la comparativa entre las arquitecturas.

5.3. Experimentos para el reconocimiento de emociones en perros

El reconocimiento de emociones en perros a través de imágenes es una tarea poco abordada en la investigación. Generalmente esta tarea se lleva a cabo para reconocer los estados emocionales del animal y evitar condiciones fisiológicas no saludables. En este experimento se realiza el reconocimiento usando la arquitectura MB-RBFN y arquitecturas CNN de referencia, así también se incluye una solución de AutoML para tratar de encontrar el mejor modelo y los mejores parámetros que alcancen un buen desempeño en el reconocimiento de emociones en perros. Los modelos son entrenados con un conjunto de datos llamado DEBIw que contiene 15,999 imágenes de emociones en perros como: agresión, ansiedad, satisfacción y miedo. El conjunto se divide en seis subconjuntos de imágenes donde cada uno tiene un tamaño diferente.

En la Tabla 5.9 se muestra la distribución de cada subconjunto de datos y la cantidad de imágenes por emoción. Las imágenes que se usan para entrenar el modelo son imágenes a las que no se aplica ninguna técnica de preprocesamiento, segmentación, caracterización o marcado de puntos clave. La Figura 5.10 muestra imágenes de los diferentes subconjuntos, como se puede observar la imagen muestra las dificultades asociadas al problema de clasificación de imágenes, incluyendo: la naturaleza no normalizada de las imágenes en término de: tamaños, iluminación, fondo, posición del perro en la imagen, inclusión de múltiples objetos, incluso texto, etc.

Figura 5.10: Imágenes de muestra seleccionadas aleatoriamente para cada uno de los subconjuntos de datos considerados.

El objetivo del experimento es tener una metodología que pueda servir como un medio no invasivo, fácil de instrumentar y fácil de volver a entrenar para la implementación de sistemas computacionales conscientes de las emociones de los perros. Para ello en este experimento se hace una comparativa entre arquitecturas CNN para medir el desempeño en el reconocimiento de emociones en perros desde imágenes.

5.3.1. Clasificación automática de las emociones de los perros

La metodología adoptada para el análisis del comportamiento de los perros a partir de imágenes está basada en CNN para el análisis del conjunto de datos recopilados. La tarea se abordó como una de clasificación supervisada: dada una imagen de entrada, el objetivo es predecir el comportamiento del perro representado en la imagen. Se consideraron algunas variantes de modelos CNN (como arquitecturas

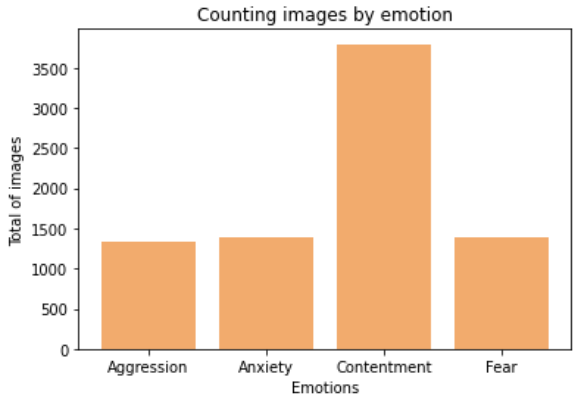


Figura 5.11: Número de imágenes etiquetadas por categoría.

Tabla 5.9: Número de imágenes en cada uno de los subconjuntos considerados.

#Muestras Dataset_dogs						
Subconjunto	A	B	C	D	E	F
Etiqueta \ Votos	2-7	3-7	4-7	5-7	6-7	7-7
Aggression	1328	829	518	465	409	149
Anxiety	1383	1239	931	791	676	235
Contentment	3793	2966	2088	1901	1722	778
Fear	1395	1094	756	704	637	327
Total	7899	6128	4293	3861	3444	1489

backbones) y se exploraron tres formas de construir tales modelos, también se evaluó el conjunto de datos con la arquitectura MB-RBFN para probar si el aprendizaje local tiene un mejor desempeño en el modelo cuando existe un desequlibrio entre las clases . Por último, se exploró el uso de una solución AutoML para optimizar los modelos CNN de referencia.

Las arquitecturas que fueron consideradas como *backbones* para entrenar el modelo de clasificación de perros son:

- VGG15
- ResNet50 y ResNet152
- InceptionNet
- EfficientNet

Para cada *backbone*, se evaluaron tres variantes del modelo, estas difieren en la forma en que los parámetros del modelo son ajustados. Las variantes son:

- **Aprendizaje desde cero**. Bajo esta configuración, las arquitecturas consideradas se entrenan desde cero, a partir de pesos aleatorios.
- **Transfer learning**[4]**.** Bajo esta configuración, un modelo preentrenado del correspondiente se carga la arquitectura *backbone*, luego se congelan los pesos de las capas convolucionales y se realiza un proceso de ajuste fino para minimizar la función de pérdida en el conjunto de datos de las emociones de los perros. Esto significa que solo se vuelven a entrenar las capas asociadas con el proceso de clasificación, tratando de adaptar el modelo a nuestro conjunto de datos específico.
- **Fine tuning.** Bajo esta variante, se carga un modelo previamente entrenado, luego se realizan varias rondas de ajuste fino. La diferencia con el enfoque anterior es que en este caso se actualizan todos los parámetros del modelo (es decir, en ninguna capa, los parámetros son congelados).
- **MB-RBFN**. Se uso la red profunda multi-rama RBF para medir el desempeño con cada arquitectura *backbone* propuesta, como parte del experimento de la comparativa del aprendizaje local. La MB-RBFN se entrenó con 4 ramas y cada rama contiene 4 unidades RBF (los que corresponden al número de clases) y la MB-RBFN usó como red *backbone* la arquitectura VGG16.

[4]Tenga en cuenta que, técnicamente, ambas variantes: transferencia de aprendizaje y ajuste fino son una forma de transferencia de aprendizaje y ambas realizan un ajuste fino de los parámetros. Usamos ambos nombres para distinguir uno de otro.

Todos los modelos se aprenden con descenso de gradiente estocástico, utiliza el optimizador de *Adam*, estableciendo un tamaño de lote de 32. Como función de pérdida para la optimización de los procesos se usa la entropía cruzada categórica.

Para el caso de AutoML se define como el subcampo del aprendizaje automático que tiene como objetivo automatizar tanto como sea posible el diseño de modelos de aprendizaje (Escalante, 2020). En el contexto del aprendizaje profundo y las CNN, esto también se denomina *Búsqueda de Arquitectura Neuronal* en inglés llamado (*Neural Architecture Search, NAS*). Los métodos NAS apuntan a automatizar el proceso de generación de modelos, reduciendo la dependencia del rendimiento del modelo en las habilidades del diseñador. Para encontrar un modelo de CNN competitivo, decidimos utilizar una metodología AutoML para diseñar automáticamente modelos de CNN efectivos. Específicamente, adoptamos la solución AutoKeras (Jin et al., 2019).

5.3.2. Resultados del reconocimiento de emociones en perros

El objetivo de estos experimentos es evaluar la viabilidad de la tarea de reconocer las emociones de los perros a partir de imágenes y determinar hasta qué punto las CNN pueden resolver la tarea planteada. Para todos los experimentos se consideró la partición aleatoria del conjunto de datos; dividiendo para cada uno de los subconjuntos de datos el $80\,\%$ de las muestras para el entrenamiento y el $20\,\%$ para prueba. Los experimentos se realizan con los 6 subconjuntos presentados en la Tabla 5.9.

En un primer experimento, se evalúa el rendimiento de las cuatro variantes de entrenamiento para todas las arquitecturas *backbone* consideradas. Se utiliza el optimizador Adam con una tasa de aprendizaje de 0,001 y un tamaño de lote de 32. Los modelos se entrenaron durante 100 épocas y se detuvieron si se observaba convergencia (sin cambios en el rendimiento de la validación durante 15 épocas). El conjunto de validación se fijó en el $20\,\%$ de los datos de entrenamiento. Para este experimento, se consideran los 6 subconjuntos de datos. Los resultados experimentales se muestran en la Fig. 5.12. El análisis de estos resultados se estudian según tres dimensiones: *modelos, entorno de aprendizaje* y *subconjunto de datos.*

Modelos. El modelo de mejor rendimiento varía según el entorno de aprendizaje. Cuando se entrena desde cero el mejor modelo es la CNN. Esto puede deberse al refuerzo de la función de pérdida en múltiples niveles para esta arquitectura, lo que puede ser benéfico cuando se entrena el modelo desde cero, dada la muestra relativamente pequeña. Como era de esperar, el rendimiento de este modelo disminuye cuando hay menos datos disponibles; compare los resultados en los conjuntos de datos A y F para la gráfica

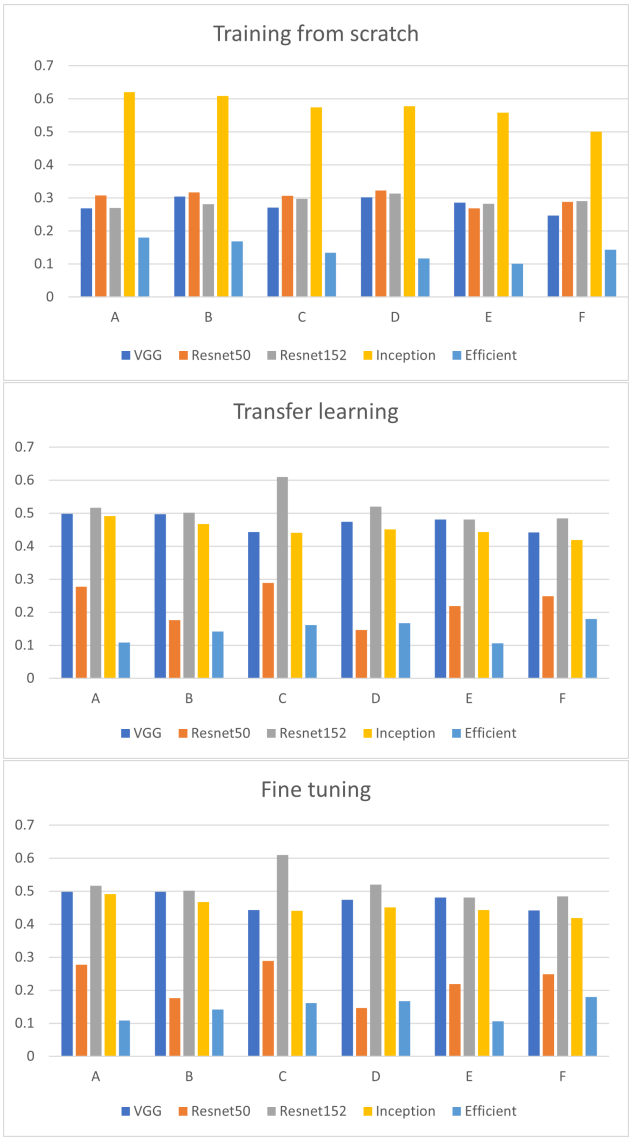


Figura 5.12: Rendimiento de macro f_1 para las arquitecturas consideradas (colores de barra) y conjuntos de datos (eje x en cada gráfico) obtenidos por modelos entrenados desde cero, transferencia de aprendizaje y ajuste fino.

superior en la Figura 5.12. El bajo rendimiento de EfficienNet dada esta configuración, también se debe al pequeño tamaño de la muestra. Este es un problema mucho más complicado que simplemente aprender los parámetros del modelo. Al comenzar con modelos preentrenados, el modelo ResNet152 obtuvo mejores resultados consistentemente, seguido por la arquitectura VGG y la CNN de Inception.

Hay que tener en cuenta que, como se mencionó anteriormente, las CNN como ResNet y VGG se han utilizado ampliamente en trabajos anteriores (Boneh-Shitrit et al., 2022; Chavez-Guerrero et al., 2022; Franzoni et al., 2019; Raman et al., 2021) relacionados al tema. Por lo tanto, los resultados obtenidos con tales métodos son indicativos del desempeño de los modelos que han sido evaluados en otras tareas y escenarios relacionados.

Configuración de aprendizaje. Como se mencionó anteriormente, aprender desde cero solo funciona para la CNN basada en Inception. De hecho, el modelo de inicio es el que muestra un comportamiento regular en los entornos de aprendizaje, lo que lo convierte en un candidato ideal para abordar el problema. Otro modelo regular, pero con bajo rendimiento, es EfficienNet, ya que logra el rendimiento más bajo en todos los entornos de aprendizaje y conjuntos de datos. Esto podría deberse al hecho de que la arquitectura EfficientNet preentrenada está demasiado optimizada para el conjunto de datos de ImageNet y no funciona bien con la recopilación de emociones de los perros.

Rendimiento entre subconjuntos. En términos de los diferentes subconjuntos, no se identificó una tendencia clara (aparte del rendimiento de InceptionNet en el entorno de aprendizaje desde cero). El rendimiento cae ligeramente cuando se pasa del conjunto de datos A al F en la configuración de transferencia de aprendizaje y ajuste fino, pero las disminuciones son bastante pequeñas. Esto es algo esperado ya que el principal beneficio de los modelos preentrenados es precisamente aliviar el problema de la muestra pequeña.

Tabla 5.10: Rendimiento de AutoML en subconjuntos. El mejor resultado obtenido para cada configuración por los modelos individuales se reporta para comparación.

	Inicio	**MB-RBFN**	**TL**	**FT**	**AutoML**
A	0.6202	0.4885	0.5171	0.5171	**0.671**
B	0.6091	0.4974	0.5016	0.5016	**0.6219**
C	0.5741	0.5131	**0.6095**	**0.6095**	0.6024
D	0.5776	0.4978	0.5198	0.5198	**0.5822**
E	0.5581	0.5397	0.4811	0.4811	**0.5813**
F	0.5004	0.5464	0.4846	0.4846	**0.5528**

En un segundo experimento, se evaluó el rendimiento de la red MB-RBFN para el reconocimiento de emociones en perros y al comparar con el rendimiento obtenido por CNN de referencia se puede obervar que MB-RBFN no alcanza resultados competitivos y muestra los resultados más bajos de todas las configuraciones de red. Para este experimento se ve que el aprendizaje local no funciona en conjuntos de datos con un alto desequilibrio entre las clases. Esto se puede ocasionar a que se usó una arquitectura CNN simple como *backbone*. En este experimento sería bueno extender el uso de diferentes *backbones* para hacer una comparativa más robusta entre los métodos y poder conlcuir si el aprendizaje local es benéfico o no en conjuntos de datos con alto desequilibrio entre las clases.

En un segundo experimento, evaluamos el rendimiento de la solución AutoML para el reconocimiento de emociones en perros y comparamos su rendimiento con el obtenido por el mejor modelo de referencia que se encuentra en el primer experimento que se muestran en la Tabla 5.10.

Como era de esperar, se obtuvieron mejores resultados con el modelo optimizado por AutoML. Curiosamente, la arquitectura seleccionada por AutoKeras(Jin et al., 2023) fue la arquitectura EfficientNet. Esto es contrario a la intuición ya que dicho modelo obtuvo el rendimiento más bajo en los experimentos anteriores. Esto hace evidente la importancia que tiene el proceso de optimización del modelo a la hora de abordar una nueva tarea. El mejor resultado general se obtuvo para el subconjunto A (2-7 votos) y el rendimiento disminuye a medida que hay menos imágenes disponibles. Esto está en línea con los resultados obtenidos por el modelo Inception cuando se entrena desde cero. En todos los casos, excepto en el subconjunto C, la solución AutoML logra el rendimiento más alto. Para el subconjunto C la diferencia con los modelos de referencia es despreciable.

5.3.3. Discusión

El objetivo de este trabajo fue identificar las emociones en imágenes de perros utilizando técnicas de aprendizaje automático supervisado en un conjunto de emociones en perros. Las imágenes no fueron segmentadas, marcadas o caracterizadas de ninguna manera antes de ser utilizadas. Esta condición aumentó la complejidad del problema porque los perros en las imágenes tienen muchas posturas y grados de oclusión. Además, las imágenes fueron tomadas desde diferentes distancias, ángulos, iluminación e incluso algunas imágenes incluían varios perros.

Los resultados obtenidos fueron notables, considerando la complejidad de esta tarea de clasifica-

ción. Obtuvimos alrededor del 70 % F-Métrica clasificando cuatro clases emocionales. AutoML mostró el rendimiento más alto en casi todas las configuraciones de evaluación en comparación con las otras seis arquitecturas de redes neuronales probadas. Se observó que AutoML es un enfoque de aprendizaje automático especialmente adecuado para la complejidad de esta tarea.

6

Conclusiones y trabajo futuro

En esta trabajo de tesis se presentó un esquema de aprendizaje profundo localmente ponderado *(Locally Weighted Deep Learning, LWDL)* que consiste en integrar una técnica de aprendizaje local en un enfoque de aprendizaje profundo. La finalidad es mejorar el desempeño de una red profunda a través de mecanismos que permitan incorporar la información local en la fase predictiva del modelo.

En la investigación se exploró como enfoque de aprendizaje profundo el uso de las CNNs para la tarea de clasificación. Se estudió los componentes de este tipo de redes y se analizó que enfoques forman parte del aprendizaje local y cuales pueden adaptarse con el aprendizaje profundo. En los hallazgos se encontró que la mejor opción para un enfoque basado en el aprendizaje local es usar métodos incrementales ya que este tipo de clasificadores contienen dos fases: la fase entrenamiento y la de prueba, en contraste con los métodos basados en memoria. Los modelos de aprendizaje profundo requieren combinar técnicas que permitan el entrenamiento y la validación de los modelos. Una parte importante en la investigación fue analizar qué características debe contener el modelo de aprendizaje local para que pueda ser llevado a un enfoque del aprendizaje profundo, ya que de forma inicial son dos modelos no compatibles; inicialmente hay que lidiar con temas de alta dimensión en el espacio de características obtenido por un modelo profundo; en contraste, por parte del aprendizaje local, hay que trabajar en el manejo de técnicas locales que permitan trabajar con conjuntos de datos extensos. Este último punto se considera importante porqué generalmente los clasificadores basados en el aprendizaje local no son buenos aproximadores cuando se tiene un conjunto de datos extenso y con un alto número de clases.

Como estudio por parte del aprendizaje local se demostró que el uso de las redes de función de

base radial son un tipo de red neuronal que su función está basada en el aprendizaje local. Las redes RBF contienen la fase de entrenamiento y de prueba, además su arquitectura es similar a las redes neuronales artificiales multi-capa. Las unidades RBF funcionan como aproximadores locales donde la neurona responde únicamente cuando se cumple el criterio de la minimización de distancia entre su centro y el vector de entrada; es decir, que mientras más cerca este la instancia al centro de la neurona, la neurona se activa.

Dados los puntos mencionados anteriormente, se observó que el proponer una red profunda basada en una CNN con múltiples ramas de unidades locales son la mejor opción para adaptar el esquema LWDL y reducir la dimensionalidad del espacio recibido por las unidades RBF. Se encontró que las ramas permiten tener mapas de características de menor dimensión. La CNN usa arquitecturas de referencia o *backbones* para realizar la extracción de características, esta información obtenida por las capas convolucionales son conectadas a múltiples ramas de unidades RBF que funcionan como aproximadores locales. La salida de las unidades RBF son concatenadas y finalmente se conecta a una capa de salida que aplica una función *softmax* para determinar la clase.

La contribución del trabajo está en la reducción del espacio de características procesado por la convolucional a través de las múltiples ramas y el uso de las unidades RBF como aproximadores locales.

El trabajo mostró un desempeño competitivo con el estado-del-arte en la tarea del reconocimiento de emociones (ER). El modelo fue evaluado en conjuntos de referencia usados en ER, donde los conjuntos variaban en el tamaño de la muestra, eran conjuntos desequilibrados o conjuntos extensos, también variaban en el número de clases, existen conjuntos que se consideran de grano-fino por el número extenso de clases que se clasificaban. En la mayoría de los casos se demostró que el uso del aprendizaje local mejoró el desempeño de la red, incluso se robusteció los resultados haciendo una comparativa con diferentes *backbones* donde se demostró que independiente de la arquitectura convolucional, el uso de las múltiples ramas RBF mejoraban sus resultados con respecto a sus redes de referencia.

A continuación, se resumen los principales hallazgos de este trabajo de investigación:

- La inclusión de información local, a través de las unidades RBF de múltiples ramas, mejora significativamente el rendimiento de un modelo de CNN. De hecho, el modelo propuesto supera a un modelo similar ampliado con una capa densa, lo que demuestra que las unidades RBF son las responsables de la mejora del rendimiento.

- El modelo propuesto es competitivo con métodos de última generación basados en arquitecturas y mecanismos más complejos, incluso cuando nos basamos en un backbone estándar y simple.
- El modelo propuesto demostró ser más ventajoso para conjuntos de datos con condiciones desafiantes que incluyen tamaño de muestra pequeño, alta superposición entre clases, conjuntos de datos con distribuciones mixtas en el conjunto de prueba y con altos índices de desequilibrio.
- Los centros de las unidades RBF de diferentes ramas capturan información local y esta información resultó muy útil para clasificar muestras provenientes de diferentes distribuciones. Visualmente se estudió que cada rama capturaba patrones locales diferentes.

6.1. Trabajo Futuro

Los hallazgos y resultados presentados en este trabajo de investigación son alentadores y motivan a seguir estudiando el LWDL. En particular, en trabajos futuros sería bueno explorar formas alternativas de incorporar información local en otros modelos basados en aprendizaje profundo. Explorar otras redes profundas con diferentes tareas a la clasificación de imágenes que pueda incluirse el aprendizaje local. Asimismo, sería interesante analizar las formas en que la información proporcionada por los centros de las unidades RBF puede ser utilizada para la explicabilidad e interpretabilidad del modelo. Seguir explorando dominios que estén dentro del alcance del aprendizaje local y el esquema LWDL alcance un alto desempeño. Finalmente, otra dirección de investigación interesante sería extender el método propuesto para que pueda usarse para mejorar el reconocimiento de emociones en imágenes de perros.

A

Apéndice

En esta parte del documento, se describen experimentos que conformaron parte de la investigación para ahondar sobre el aprendizaje local y el aprendizaje profundo. Los experimentos aquí mostrados forman parte de una pauta a seguir para ver que camino es mejor tomar al integrar el aprendizaje local en métodos de aprendizaje profundo. A continuación, se describen tres experimentos que incluyen el aprendizaje local.

A.1. Aprendizaje local en datos sintéticos

El objetivo del experimento es evaluar el aprendizaje local con el uso de las redes RBF en conjuntos de datos sintéticos, que cumplen con la particularidad de ser no linealmente separables y con traslape entre las clases; para posteriormente compararlos contra un método de aprendizaje global como un MLP. La finalidad es explorar si de primera instancia es benéfico el aprendizaje local en ciertos tipos de datos, para identificar técnicas que permitan incluir el aprendizaje local en métodos de aprendizaje profundo. El modelo local que se adapta es una *red de función de base radial* (RBF) y se proponen algunas variantes de la arquitectura.

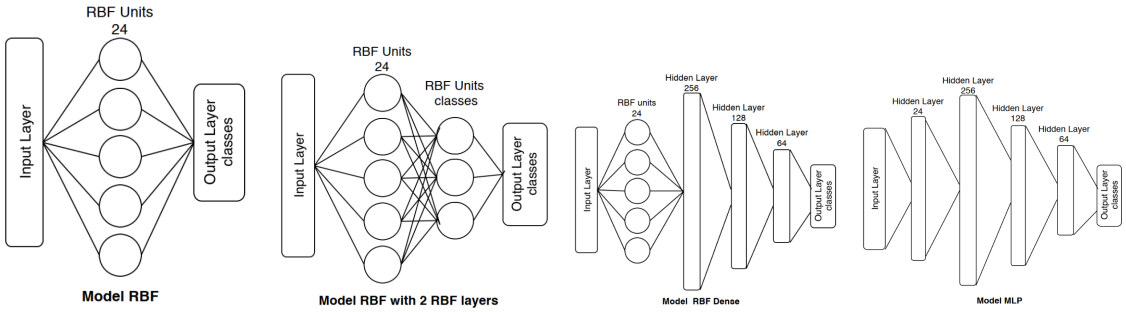
Las arquitecturas que se evalúan son las siguientes:

1. **Red RBF**. Las redes tipo RBF (véase Figura A.1a) son un tipo especial de arquitectura muy similar a las MLP, pero adaptan el aprendizaje local en su modelo. Este tipo de red se observa que es adecuada en varias aplicaciones, una de ellas es donde los datos son linealmente no separables.

2. **Red RBF a 2 capas**. Esta configuración propuesta (ver Figura A.1b) se adapta con la finalidad de mejorar el espacio de clasificación. La arquitectura en la primera capa RBF se encarga de aprender aquellos atributos más generales y la segunda capa RBF, el número de las unidades RBF es el número de clases a predecir. La segunda capa RBF se encarga de transformar el espacio de tal forma que cada unidad RBF se concentre en cada una de las clases. La capa de salida usa la función SOFTMAX para computar la probabilidad de la clase.

3. **Red RBF con múltiples capas totalmente conectadas**. Esta configuración es un híbrido del aprendizaje local y global (véase en la Figura A.1c). En este caso, la arquitectura se encarga de agrupar los atributos en la capa inicial de forma local, posteriormente se transmiten a neuronas a través de capas densas. En la capa final, se utiliza la función de activación *softmax* para predecir la clase.

4. **Red MLP**. La red MLP (ver Figura A.1d) su aprendizaje está basado en aprendizaje global. Y se plantea una configuración para resolver el problema de clasificación de datos sintéticos que contienen traslape entre las clases.

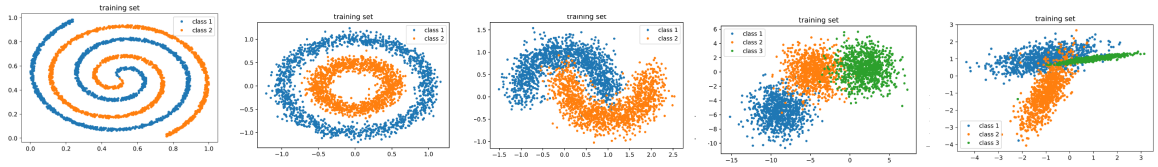


(a) (b) (c) (d)

Figura A.1: Redes RBF como métodos de aprendizaje local y MLP como aprendizaje global. La FiguraA.1a es una red RBF con tres capas, una de entrada, una capa RBF oculta y una capa de salida. La Figura A.1b es una red RBF con dos capas RBF ocultas. La Figura A.1c es una red MLP que contiene múltiples capas ocultas, pero la primer capa es una capa RBF, el resto de capas son capas densas totalmente conectadas. La Figura A.1d es una red neuronal profunda de 4 capas densas ocultas.

A.1.1. Creación de datos sintéticos

Para generar los conjuntos de datos sintéticos se utilizó la librería scikit-learn (Pedregosa et al., 2011) debido a que tiene funciones que permiten generan algunos conjuntos de datos sintéticos. Se calcularon 5 diferentes distribuciones en los valores de los puntos y para ello, se usaron diferentes tamaños de datos



(a) Spiral (b) Circle (c) Moons (d) Blobs (e) Multi Class

Figura A.2: Representación de los conjuntos de datos sintéticos.

considerando los siguientes números de puntos: 520,1000,2000,4000,6000,10000,20000,40000,80000,160000. En la Tabla A.1 se muestra la distribución de los conjuntos en porcentajes sobre el total de puntos.

Tabla A.1: Distribución de los conjuntos de datos sintéticos para entrenamiento, prueba y validación.

Dataset	**%Entrenamiento**	**%Validación**	**%Prueba**	**#Classes**
Spiral Circle Moons	64 %	16 %	20 %	2
Blobs Multi-class	64 %	16 %	20 %	3

A.1.2. Evaluación del aprendizaje local en datos sintéticos

Los conjuntos se evalúan en cada una de las configuraciones propuestas. En la Tabla A.2 se muestran los resultados obtenidos para cada modelo. En los hallazgos se observó que el aprendizaje local tiene un mejor resultado con respecto a uno global cuando las clases no son linealmente separables. En el caso de que exista un traslape de las clases, se observó que el aprendizaje global tiene un mejor desempeño; esto se debe a que el cálculo de la vecindad de los puntos que tienen traslape va a caer en clases erróneas.

En la Tabla A.2 se muestran los resultados en exactitud Top-1 para cada conjunto de datos con la evaluación de cada uno de los modelos propuestos de redes RBF y MLP (véase la Figura A.1). La tabla también muestra la dispersión de los datos a través de la desviación estándar. Un punto importante es que los resultados del experimento permiten explorar en qué tipo de datos el aprendizaje local se desempeña de forma sobresaliente, esto nos permitió darnos una idea global acerca de qué dominios podemos aplicar el aprendizaje local en modelos de aprendizaje profundo y tengamos resultados benéficos.

El aprendizaje local no se desempeña adecuadamente en conjuntos de datos donde existe un tras-

Tabla A.2: Resultados *Top-1* de exactitud de los métodos local vs global.

Dataset	Red RBF	Red RBF a 2 capas	RBF con capas FC	MLP
Spiral	0.6556 ± 0.0245	**0.9625 ± 0.193**	0.6021±0.0482	0.6166±0.0055
Circle	0.9932±0.0066	0.9945±0.0069	**0.9961±0.00032**	0.7667±0.0283
Blobs	0.5977±0.0372	0.5719±0.0332	0.6931±0.0214	**0.9626±0.0021**
Moons	**0.9706±0.0034**	0.9616±0.0051	0.9699±0.0033	0.925±0.0374
Multi-class	0.8976±0.0441	0.8795±0.0525	0.9079±0.0387	**0.9139±0.0366**

lape entre el espacio latente de las distintas clases. El estudio nos abre un panorama acerca de los datos de grano fino y es que el aprendizaje local no sea una buena técnica para resolver dicha tarea. Una aplicación bastante interesante donde se observó que el aprendizaje local tiene buenos resultados es en datos donde no existe una separabilidad lineal. Éste tipo de datos como los que se muestran en la Figura A.2b generalmente se utiliza una transformación del espacio latente a una dimensión más alta que permita trazar una separabilidad entre los datos. En éste caso el uso de aproximadores locales permiten tener buenos resultados en la tarea de clasificación.

De las arquitecturas evaluadas la que mejor resultado dio es la que mezcla el uso de la capa RBF y capas densas, es decir la que es un híbrido del aprendizaje local con el global. La idea de la arquitectura que combina capa RBF y capa densa es que se tiene integrado los dos tipos de aprendizaje, el local y el global. La arquitectura como se puede observar en la Figura A.1c integra una capa RBF como aprendizaje local seguido de la capa de entrada, posteriormente utiliza tres capas densas totalmente conectadas seguidas de la capa de salida. Los hallazgos permiten observar que en una arquitectura completamente basada en aprendizaje global se puede encontrar una mejora en su rendimiento si se añade el concepto del aprendizaje local. Esto nos conduce a plantear la hipótesis de mejorar el desempeño de un modelo profundo con la integración del aprendizaje local en su fase predictiva.

A.2. Aprendizaje local sobre algoritmos de agrupamiento

En la Sección 4.3 se explicó que una de las formas de llevar a cabo el esquema LWDL es el uso de las CNN en combinación con un método de LWL como las redes RBF, o usando adaptadores locales basado en algoritmos de agrupamiento (*Clustering*), así como el uso de clasificadores basados en distancias ponderadas. Se consideran estas variantes como parte del aprendizaje local debido a que usan el concepto de *vecindad* para calcular aproximaciones entre el espacio de características.

El objetivo del experimento es proponer una variante sobre un algoritmo de agrupamiento donde se pueda incluir el concepto del aprendizaje local para llevarlo a un método profundo. Para ello primero como parte de la investigación, se llevó a cabo una revisión del estado del arte sobre trabajos que adaptan el aprendizaje local usando algoritmos de agrupamiento sobre métodos de aprendizaje profundo. Los trabajos que existen abordan la tarea con el uso de auto-codificadores convolucionales que integran una función de pérdida orientada al agrupamiento *clustering*.

Una particularidad de los trabajos que adaptan la función de pérdida orientada al agrupamiento es que modelan el espacio latente de tal forma que las clases se agrupen entre sí, e incluso sin traslape entre los grupos, basando su función de pérdida muy similar a como lo hace el algoritmo K-Means (como se puede observar en la Figura A.3).

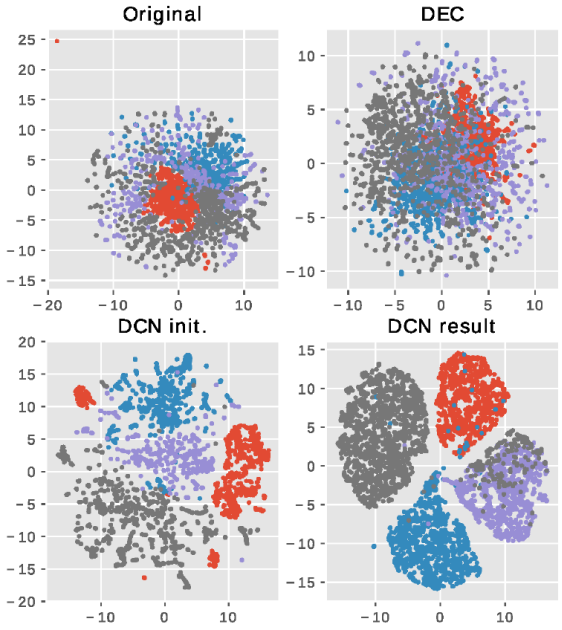


Figura A.3: Figura tomada de (Yang et al., 2017), donde muestra los resultados de su modelo al evaluarlo con un conjunto de datos sintético. Que consiste en la agrupación de clases a través de una función de pérdida.

Durante la investigación se aborda la idea de integrar un algoritmo que directamente sobre el algoritmo *K-Means* mejore los resultados con la adaptación del término radial en la construcción de sus centroides. La mejora consiste en añadir un término de radio para modelar el centro de cada clúster, simi-

lar a cómo lo hacen las unidades de base radial o RBF. Las unidades RBF consideran la distancia de cada instancia hacia cada neurona. En esa dirección surgió el planteamiento de un algoritmo llamado *Radial K-Means*.

La idea consiste en integrar el término de radio en las agrupaciones para que, al momento de agrupar una instancia, se tome en cuenta a su centroide más cercano y su radio. El contexto es el mismo que los modelos anteriores de *K-Means*, pero con el hallazgo de lidiar con los valores atípicos para mejorar la asignación a los grupos, algo que naturalmente no *K-Means* no hace.

El algoritmo utiliza el promedio y la desviación estándar para determinar el valor atípico en el conjunto de datos para cada grupo. Los valores atípicos obtenidos en cada iteración son eliminados de la siguiente iteración y una vez que el algoritmo termina de iterar, el punto más lejano de cada grupo se convierte en el siguiente radio. Posteriormente, se recalcula el agrupamiento con base en el radio y el centro de cada grupo. El centro de cada grupo es determinado de forma similar a como lo hace *K-Means* con el promedio de sus instancias. Otro punto importante es identificar en qué casos el método tiene un alto desempeño y compararlo de forma inicial con el propio algoritmo K-Means. La finalidad del experimento es probar que el aprendizaje local funciona en otros enfoques distintos a una CNN. Una vez que los resultados son analizados, se plantea estudiar si es o no conveniente llevarlo a un nuevo método de aprendizaje profundo.

A.2.1. Conjuntos de datos para agrupamiento

El modelo propuesto es evaluado sobre conjuntos de datos sintéticos de referencia comúnmente utilizados en algoritmos de agrupamiento. Los conjuntos de datos se muestran en la Figura A.4 (en la columna llamada *Target*) y estos fueron obtenidos de (Fränti and Sieranoja, 2018).

Inicialmente estos conjuntos de datos fueron propuestos para medir un estándar o punto de referencia del algoritmo *K-Means*. Específicamente, se medía el rendimiento considerando los siguientes factores: la superposición de grupos, el número de grupos, la dimensionalidad del conjunto y el desequilibrio en el tamaño de los grupos. Por ejemplo, en el conjunto de datos A de A1 hasta A3 varía el número de grupos y tiene un tamaño de 3000 a 7500 puntos con 20-30-50 grupos respectivamente. Como se puede observar en la Figura A.4 en la columna *Target* los grupos varían en superposición, densidad de puntos, desequilibrio de puntos en los grupos. En la Tabla A.3 se muestra la distribución del conjunto de datos usado para el

Tabla A.3: Conjunto de datos de referencia para medir el desempeño en algoritmos de agrupamiento.

Dataset	#Clusters
A1	20
A2	35
A3	50
Aggregation	7
Flame	2
G2-2-30	2
G2-2-50	2
R15	15
S1	15
S2	15
S3	15
S4	15
Unbalance	8

agrupamiento o *Clustering*.

A.2.2. Comparativa con algoritmos de referencia

Para medir el desempeño del algoritmo propuesto, usamos las métricas estándares para la evaluación del agrupamiento. Específicamente utilizamos el *Rand Index (RI)*, *Adjusted Rand Index (ARI)*, *Adjusted Mutual Info Score (AMI)* y *Homogeneity Score (HS)*. Como se puede observar en la Tabla A.4, el algoritmo propuesto tiene resultados competitivos con *K-Means*. Existe únicamente un conjunto de datos en el cual el desempeño es mayor en comparación con *K-Means*, pero en el resto de los conjuntos, los resultados están muy cercanos entre sí.

La razón por la cual el método propuesto mejoró en el caso particular del conjunto *Aggregation*, se puede deber a que este no contiene traslape entre los clústeres y al parecer al existir un traslape, el concepto de radio tiene un conflicto al asignar el grupo cuando los centroides están muy cercanos entre sí y tienen el mismo radio como sucedió en el peor caso con *G2-2-50*. En la Tabla A.4 se muestran los resultados

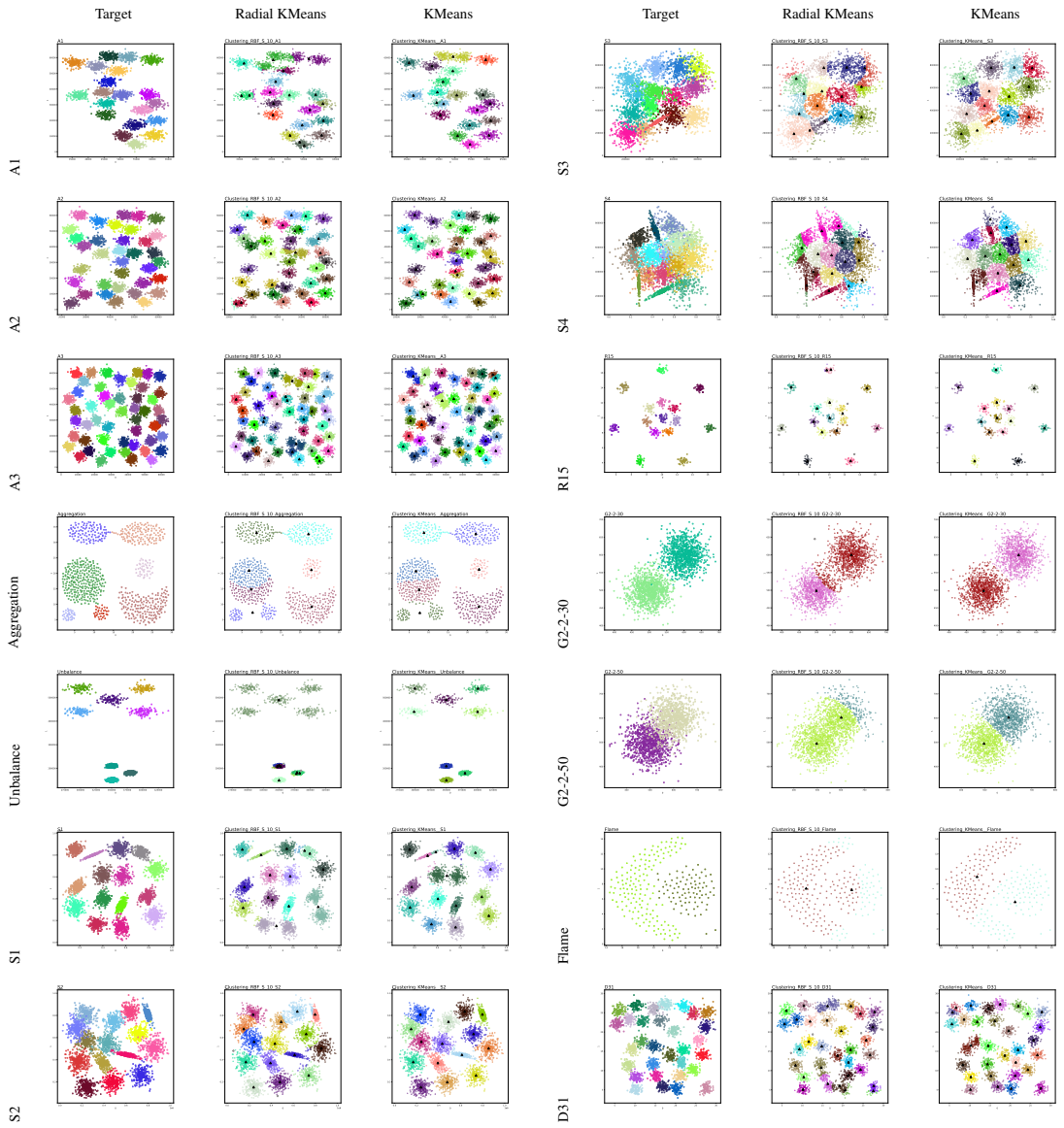


Figura A.4: Resultados gráficos de la evaluación del algoritmo *Radial K-Means* contra *K-Means*. En la figura la primera columna muestra el gráfico que corresponde al conjunto de datos original, la segunda columna corresponde al resultado de agrupar con el método propuesto y la tercera columna muestra el resultado de agrupar con el método *K-Means*. Cada grupo se representa con un color distinto, para el caso de los centros estos se muestran en color negro. (Ver a color)

de las métricas con cada conjunto de datos, reportando una comparativa entre el método propuesto y el resultado obtenido por el algoritmo K-Means, así también se puede ver el número de clústeres usado por cada conjunto de datos.

Tabla A.4: Resultados de las métricas RI, ARI, AMI, y HS para evaluar el desempeño del modelo propuesto (*Radial K-Means*) contra el algoritmo KMeans para los conjuntos de datos de referencia.

Dataset	RI		ARI		AMI		HS		#Clusters
	Radial K-Means	*K-Means*	*Radial K-Means*	*K-Means*	*Radial K-Means*	*K-Means*	*Radial K-Means*	*K-Means*	
A1	0.9802	**0.9929**	0.8087	**0.9275**	0.9130	**0.9715**	0.9328	**0.9671**	20
A2	0.9930	**0.9948**	0.8790	**0.9088**	0.9468	**0.9648**	0.9559	**0.9622**	35
A3	0.9947	**0.9975**	0.8709	**0.9389**	0.9495	**0.9802**	0.9592	**0.9785**	50
Aggregation	**0.9348**	0.9283	**0.7884**	0.7656	**0.8984**	0.8808	0.8568	**0.9320**	7
Flame	0.5958	**0.7155**	0.1771	**0.4311**	0.1037	**0.3920**	0.1157	**0.4054**	2
G2-2-30	0.8583	**0.9816**	0.7166	**0.9632**	0.6753	**0.9244**	0.6812	**0.9245**	2
G2-2-50	0.5150	**0.8485**	0.5304	**0.6970**	0.1304	**0.5888**	0.2185	**0.5890**	2
R15	0.9882	**0.9991**	0.9073	**0.9927**	0.9597	**0.9938**	0.9727	**0.9942**	15
S1	0.9777	**0.9891**	0.8395	**0.9162**	0.9305	**0.9597**	0.9608	**0.9496**	15
S2	0.9822	**0.9923**	0.8580	**0.9382**	0.9059	**0.9462**	0.9075	**0.9467**	15
S3	0.9379	**0.9660**	0.5538	**0.7273**	0.7243	**0.7952**	0.7482	**0.7963**	15
S4	0.9112	**0.9473**	0.4201	**0.5937**	0.6125	**0.7039**	0.6534	**0.6992**	15
Unbalance	0.9720	**0.9972**	0.9299	**0.9992**	0.8981	**0.9984**	0.8852	**0.9987**	8

A.2.3. Discusión

Los resultados obtenidos con el método propuesto *Radial K-Means* no presentan una mejora con respecto al algoritmo *K-Means*, pero el método muestra tener un buen desempeño en la tarea de detección de valores atípicos. El algoritmo considera los valores atípicos como aquellos puntos que se encuentran más alejados de la agrupación y que tienen una densidad menor. Estos son excluidos durante el agrupamiento, con la finalidad de que no influyan en el promedio del cálculo de los representantes (centroides), como se puede observar en la Figura A.4. Esto es algo que el método *K-Means* no puede realizar.

Una mejora directa al algoritmo *K-Means* no se puede llevar a cabo con la adaptación de la técnica de aprendizaje local usando el concepto radial para la construcción de sus centroides. Por lo tanto, no tiene beneficio llevarlo a un método de aprendizaje profundo. Una forma de llevar a cabo el aprendizaje local en un modelo de DL sería a través de una función de pérdida enfocada al agrupamiento.

A.3. Auto-codificador con función de pérdida orientada al agrupamiento

En la investigación se estudió el uso de funciones de pérdida orientadas al agrupamiento sobre métodos de aprendizaje profundo, con la finalidad de explorar la integración del aprendizaje local en DL. En la literatura se encontró un trabajo donde los autores proponen un método que simultáneamente realiza una reducción de dimensionalidad (que es un punto clave para el esquema LWDL) y un modelado del espacio latente a través del algoritmo *K-Means*. La forma en la que se llevó a cabo es mediante la integración de una función de pérdida que mide el desempeño del auto-codificador y otra que mide el desempeño del agrupamiento del espacio latente como se ilustra en la Figura A.5).

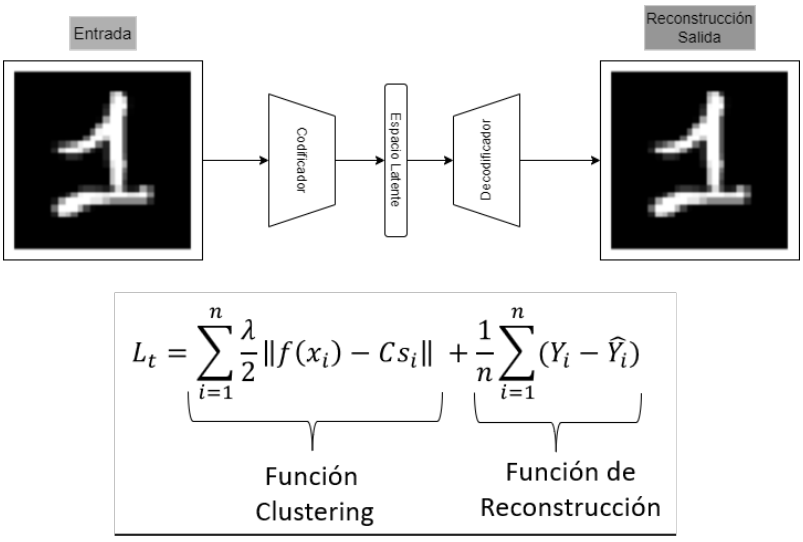


Figura A.5: Autoencoder con función de pérdida orientada al agrupamiento.

Para ello se desarrolló el auto-codificador propuesto por Yang et al. (2017) este contiene una estructura basada en un MLP. El codificador contiene una entrada con una dimensión de 784 características y tres capas de 50, 50 y 2000 neuronas respectivamente (ver Figura A.6). El espacio latente contiene 10 neuronas, que representan el número de clases que se procesaran. El decodificador contiene la misma estructura que el codificador.

El conjunto de datos que se utilizó para entrenar y probar el modelo es MNIST (LeCun et al., 1998). MNIST contiene 60,000 imágenes de entrenamiento y 10,000 de prueba; en escala de grises y representan dígitos del 0 al 9, siendo un total de 10 clases.

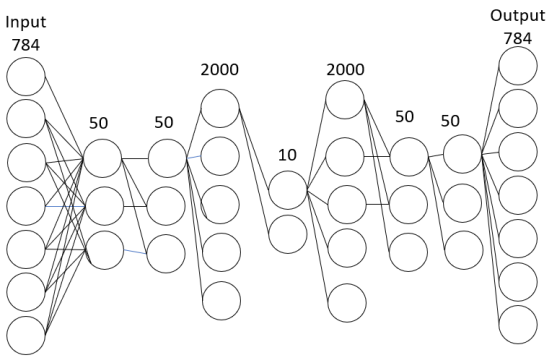


Figura A.6: Arquitectura basada en un MLP para un autoencoder, la red contiene una capa de entrada con una dimensión de 784 neuronas y tres capas totalmente conectadas con 50,50 y 200 neuronas respectivamente. Con un espacio latente de 10 unidades. En el decodificador se tiene la misma configuración que el codificador.

El modelo se entrenó durante 50 épocas con 10 valores de semillas diferentes para calcular un promedio del desempeño y una desviación estándar del desempeño del modelo. Se utilizó como optimizador el algoritmo Adam (Kingma and Ba, 2014).

Para medir el desempeño del auto-codificador se utilizó la pérdida basada en el error cuadrático medio (MSE) en la función de pérdida. En el caso de la pérdida del agrupamiento, este consiste en adaptar el principio usado por el algoritmo *K-Means*. Se calcula la distancia entre el espacio latente $f(x_i)$ hacia cada centroide C. Se añade un término s_i que toma los valores de 0 y 1, donde sí la distancia calculada es la menor a cualquier centroide, a este se le asigna un valor de 1 en caso contrario 0. λ es un parámetro que funciona como regulador entre la reconstrucción y el agrupamiento.

Los centros son calculados con la Ecuación A.1, el término c_k refiere al conteo de veces que la instancia ha sido asignada a un clúster o grupo k, el conteo influye como un parámetro de control que conforme el conteo incrementa. La idea es que la modificación del centro no cambie abruptamente y tienda a estabilizar el centro. La actualización del centro m_k toma el valor del centro anterior y este centro se mueve tomando en cuenta la diferencia entre el centro en corriente y la instancia transformada al espacio latente $f(x_i)$.

$$m_k \leftarrow m_k - (1/c_k)(m_k - f(x_i))s_i \tag{A.1}$$

El término λ es evaluado experimentalmente con diferentes valores dónde: $\lambda = \{$ 1e-6, 1e-5, 1e-4, 0.001, 0.01, 0.05, 0.1, 1, 10, 100 $\}$.

A.3.1. Resultados Experimentales del auto-codificador

Para evaluar el desempeño del auto-codificador se hace midiendo la reconstrucción de la imagen usando la pérdida MSE y para el desempeño del clustering con métricas como: ACC, MNI, ARI y homogeneidad. Como se puede observar en la Ecuación A.1 se incluye un término denominado λ que se encarga de equilibrar la agrupación. El término λ se calcula experimentalmente, es decir se asignan un grupo de valores y se va calculando la pérdida total.

Posteriormente, son necesarias métricas para robustecer el experimento y en la investigación se propone realizar pruebas de hipótesis basada en la *Paired t-Test* para verificar que añadir el clustering en la función de pérdida es benéfico para el modelo.

En el experimento se hace una comparativa en el auto-codificador la cual incluye el añadir el agrupamiento (clustering) y cuando no se incluye. En la Tabla A.5 se muestra el resultado MSE obtenido de la reconstrucción de la imagen para los dos casos. El valor de $\lambda = 0$ se refiere a cuando no se usa clustering en la función de pérdida y el resto de los valores de λ es cuando la función de pérdida incluye el clustering.

Tabla A.5: Tabla de resultados obtenidos de la evaluación de la reconstrucción alcanzada por el auto-codificador, los datos reportan métricas de error cuadrático medio para cada valor diferente de λ.

λ	**MSE**
0	0.04871 ± 0.00386
1e-6	0.04900 ± 0.00480
1e-5	0.04884 ± 0.00559
0.0001	0.05696 ± 0.00712
0.001	0.11107 ± 0.03213
0.01	0.13917 ± 0.01636
0.1	0.14265 ± 0.00921
0.05	0.13744 ± 0.01537
1	0.13969 ± 0.00951
10	0.13903 ± 0.00596
100	0.13952 ± 0.00501

Los resultados obtenidos muestran que añadir el agrupamiento en la función de pérdida aumenta el valor del error en la reconstrucción, lo importante es hacer las pruebas de hipótesis para corroborar tales

datos. En principio, siempre se tendrá un valor mayor cuando el término de agrupamiento se añade a la pérdida, ya que este se suma a la reconstrucción. En la Tabla A.6 se muestran los resultados que arrojan de las pruebas de hipótesis.

Las pruebas de hipótesis se encargan de comparar dos distribuciones y verifica si pertenecen a la misma distribución o no. La hipótesis que se plantea para este experimento es que es mejor el implementar el agrupamiento y la reconstrucción. Cuando las distribuciones resulten ser diferentes significa que se valida la hipótesis que en este caso mostró un resultado significativo aplicar la reconstrucción junto con el agrupamiento. En el caso contrario, por ejemplo, cuando se rechazó la hipótesis, que las distribuciones eran las mismas, se entiende que es lo mismo adaptar o no adaptar el agrupamiento.

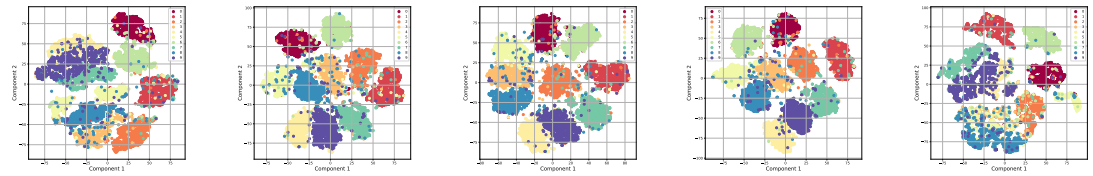
Tabla A.6: Tabla de resultados de las pruebas de hipótesis realizadas para en el experimento del autocodificador.

Lambda	Reconstruction Lambda = 0 (R)	Reconstruction+Clustering (R+C=RC)	Hipótesis	Conclusión
1e-6	0.04871 ± 0.00386	0.04900 ± 0.00480	R<RC	Same distributions (fail to reject H0)
1e-5	0.04871 ± 0.00386	0.04884 ± 0.00559	R<RC	Same distributions (fail to reject H0)
0.0001	0.04871 ± 0.00386	0.05696 ± 0.00712	R<RC	Different distributions (reject H0)
0.001	0.04871 ± 0.00386	0.11107 ± 0.03213	R<RC	Different distributions (reject H0)
0.01	0.04871 ± 0.00386	0.13917 ± 0.01636	R<RC	Different distributions (reject H0)
0.05	0.04871 ± 0.00386	0.13744 ± 0.01537	R<RC	Different distributions (reject H0)
0.1	0.04871 ± 0.00386	0.14265 ± 0.00921	R<RC	Different distributions (reject H0)
1	0.04871 ± 0.00386	0.13969 ± 0.00951	R<RC	Different distributions (reject H0)
10	0.04871 ± 0.00386	0.13903 ± 0.00596	R<RC	Different distributions (reject H0)
100	0.04871 ± 0.00386	0.13952 ± 0.00501	R<RC	Different distributions (reject H0)

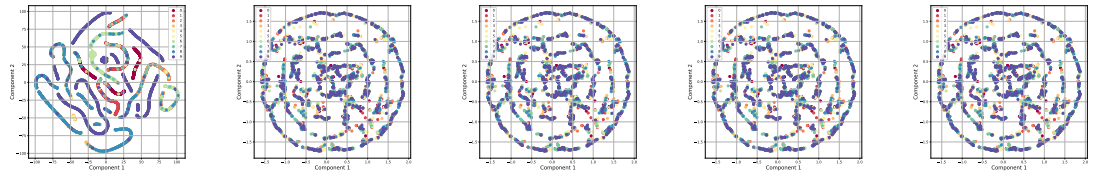
Para el desempeño del clustering validaremos los resultados de dos formas, tanto visuales como cuantitativos. En la Tabla A.7 se muestran los resultados obtenidos con la métricas propuestas para medir el desempeño del clustering sobre el conjunto de datos de prueba. Lo que se realiza es llevar cada instancia del conjunto de prueba a la representación obtenida por el espacio latente y se evalúa cada métrica. El mejor modelo se obtiene cuando se aplica clustering a la función pérdida con un valor de $\lambda = 0.0001$, alcanzando la mejor exactitud o en inglés *accuracy* en el modelo con respecto al resto. Se observa que el espacio latente se modela mejor cuando se añade el termino de clustering a diferencia de cuando no lo contiene. En la Figura A.7 se muestran los espacios latentes para cada valor de λ.

Tabla A.7: Tabla de resultados de la evaluación de las métricas de clustering: ACC, MNI, ARI y Homogeneidad en la representación del espacio latente del autoencoder, mostrando resultados de cuando se integra el clustering en la función de pérdida y cuando no. Las evaluaciones se realizan para cada valor de lambda.

Lambda/Seed	**ACC**	**MNI**	**ARI**	**Homogeneity**
0	0.54578 ± 0.04637	0.50172 ± 0.03278	0.35590 ± 0.03881	0.49120 ± 0.03346
1.00E-06	0.56337 ± 0.03822	0.51690 ± 0.02962	0.36813 ± 0.03325	0.50627 ± 0.02982
1.00E-05	0.64711 ± 0.06402	0.57957 ± 0.02767	0.45907 ± 0.04885	0.57203 ± 0.03121
0.0001	**0.79755 ± 0.10642**	**0.71870 ± 0.05348**	**0.66205 ± 0.09930**	**0.71464 ± 0.05965**
0.001	0.36417 ± 0.22425	0.34262 ± 0.29808	0.24429 ± 0.21820	0.32448 ± 0.28392
0.01	0.19766 ± 0.177442	0.09967 ± 0.21013	0.07381 ± 0.15572	0.09722 ± 0.20500
0.05	0.25641 ± 0.233442	0.17145 ± 0.27853	0.13642 ± 0.22538	0.16653 ± 0.27021
0.1	0.19185 ± 0.16637	0.11320 ± 0.24411	0.07259 ± 0.15520	0.10588 ± 0.22745
1	0.25100 ± 0.29793	0.15883 ± 0.33752	0.14921 ± 0.32148	0.15731 ± 0.33479
10	0.23249 ± 0.25716	0.15222 ± 0.32146	0.12860 ± 0.27589	0.14223 ± 0.30182
100	0.23353 ± 0.25430	0.15053 ± 0.32008	0.12662 ± 0.27104	0.14214 ± 0.30214



AE sin clustering | AE Lambda 1e-6 | AE Lambda 1e-5 | AE Lambda 0.0001 | AE Lambda 0.001



AE Lambda 0.01 | AE Lambda 0.1 | AE Lambda 1 | AE Lambda 10 | AE Lambda 100

Figura A.7: Gráficas de visualización de resultados obtenidos por el espacio latente del autoencoder. Los resultados fueron transformados utilizando el algoritmo t-SNE para la reducción del espacio a 2-D. En cada imagen se muestra una descripción sobre el valor de lambda que involucra la función de pérdida, el primer caso es la visualización del espacio latente sin clustering. En algunos ejemplos es claro la separación de los grupos que representan cada una de las clases con colores diferentes, en el caso donde no hay una diferencia clara, el método no converge.

A.3.2. Discusión

Los resultados reportan qué si se modela el espacio latente a través de una función de pérdida orientada al agrupamiento y se aprende conjuntamente con el modelo, se mejora la separabilidad del espacio de características y el modelo mejora los resultados en su predicción. Se puede pensar que esta mejora se aproveche para para realizar una clasificación más exacta, ya que en estas tareas es importante tener un espacio latente bien agrupado por clases. Sí llevamos el concepto de la función de pérdida a un esquema LWDL puede ser benéfico en el rendimiento del modelo que lo integre.

Bibliografía

Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal, and Vijayan K Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292, 2019.

Umut Asan and Secil Ercan. *An Introduction to Self-Organizing Maps*, pages 299–319. 01 2012. doi: 10.2991/978-94-91216-77-0_14.

Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning. In *Lazy learning*, pages 11–73. Springer, 1997a.

Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning for control. In *Lazy learning*, pages 75–113. Springer, 1997b.

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2 (1):1–127, 2009.

Tali Boneh-Shitrit, Shir Amir, Annika Bremhorst, Daniel S. Mills, Stefanie Riemer, Dror Fried, and Anna Zamansky. Deep learning models for automated classification of dog emotional states from facial expressions, 2022. URL `https://arxiv.org/abs/2206.05619`.

Ran Breuer and Ron Kimmel. A deep learning perspective on the origin of facial expressions. *arXiv preprint arXiv:1705.01842*, 2017.

Dallas Card, Michael Zhang, and Noah A Smith. Deep weighted averaging classifiers. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 369–378. ACM, 2019.

Hilmi Berk Celikoglu. Application of radial basis function and generalized regression neural networks in non-linear utility function specification for travel mode choice modelling. *Mathematical and Computer Modelling*, 44(7-8):640–658, 2006.

Víctor Ocyel Chavez-Guerrero, Humberto Perez-Espinosa, María Eugenia Puga-Nathal, and Veronica Reyes-Meza. Classification of domestic dogs emotional behavior using computer vision. *Computación y Sistemas*, 26(1), 2022.

Yuedong Chen, Jianfeng Wang, Shikai Chen, Zhongchao Shi, and Jianfei Cai. Facial motion prior networks for facial expression recognition. In *2019 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2019.

Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

Wen-Sheng Chu, Fernando De la Torre, and Jeffrey F Cohn. Learning spatial and temporal cues for multi-label facial action unit detection. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 25–32. IEEE, 2017a.

Wen-Sheng Chu, Fernando De la Torre, and Jeffrey F Cohn. Selective transfer machine for personalized facial expression analysis. *IEEE transactions on pattern analysis and machine intelligence*, 39(3):529–545, 2017b.

G Colmenares. Función de base radial. radial basis function (rbf)[en línea], 2007.

Ciprian Adrian Corneanu, Marc Oliu Simón, Jeffrey F Cohn, and Sergio Escalera Guerrero. Survey on rgb, 3d, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1548–1568, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

Samira Ebrahimi Kahou, Vincent Michalski, Kishore Konda, Roland Memisevic, and Christopher Pal. Recurrent neural networks for emotion recognition in video. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 467–474, 2015.

Paul Ekman. Facial action coding system. 1977.

Peter Englert. Locally weighted learning. In *Seminar Class on Autonomous Learning Systems*, 2012.

Hugo Jair Escalante. Automated machine learning - a brief review at the end of the early years. *CoRR*, abs/2008.08516, 2020. URL `https://arxiv.org/abs/2008.08516`.

Hossam Faris, Ibrahim Aljarah, and Seyedali Mirjalili. Evolving radial basis function networks using moth–flame optimizer. In *Handbook of neural computation*, pages 537–550. Elsevier, 2017.

Chollet Francois. Deep learning with python, 2017.

Pasi Fränti and Sami Sieranoja. K-means properties on six clustering benchmark datasets, 2018. URL `http://cs.uef.fi/sipu/datasets/`.

Valentina Franzoni, Alfredo Milani, Giulio Biondi, and Francesco Micheli. A preliminary work on dog emotion recognition. In *IEEE/WIC/ACM International Conference on Web Intelligence-Companion Volume*, pages 91–96, 2019.

Darshan Gera and S Balasubramanian. Landmark guidance independent spatio-channel attention and complementary context information based facial expression recognition. *Pattern Recognition Letters*, 145:58–66, 2021.

Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. .ºReilly Media, Inc.", 2017.

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, pages 5736–5745, 2017.

Deepak Ghimire and Joonwhoan Lee. Geometric feature-based facial expression recognition in image sequences using multi-class adaboost and support vector machines. *Sensors*, 13(6):7714–7734, 2013.

Deepak Ghimire, Joonwhoan Lee, Ze-Nian Li, and Sunghwan Jeong. Recognition of facial expressions based on salient geometric features and support vector machines. *Multimedia Tools and Applications*, 76 (6):7921–7946, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*, pages 117–124. Springer, 2013.

Alexander AS Gunawan et al. Face expression detection on kinect using active appearance model and fuzzy logic. *Procedia Computer Science*, 59:268–274, 2015.

Jihun Hamm, Christian G Kohler, Ruben C Gur, and Ragini Verma. Automated facial action coding system for dynamic analysis of facial expressions in neuropsychiatric disorders. *Journal of neuroscience methods*, 200(2):237–256, 2011.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

SL Happy, Anjith George, and Aurobinda Routray. A real time facial expression classification system using local binary patterns. In *2012 4th International conference on intelligent human computer interaction (IHCI)*, pages 1–5. IEEE, 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016b.

Xuanyu He and Wei Zhang. Emotion recognition by assisted learning with convolutional neural networks. *Neurocomputing*, 291:187–194, 2018.

Kuo-Wei Hsu and Jaideep Srivastava. An empirical study of applying ensembles of heterogeneous classifiers on imperfect data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 28–39. Springer, 2009.

Maryam Imani and Gholam Ali Montazer. A survey of emotion recognition methods with emphasis on e-learning environments. *Journal of Network and Computer Applications*, page 102423, 2019.

Md IqbalQuraishi, J Pal Choudhury, Mallika De, and Purbaja Chakraborty. A framework for the recognition of human emotion using soft computing models. *International Journal of Computer Applications*, 40(17): 50–55, 2012.

Rachael E Jack, Oliver GB Garrod, Hui Yu, Roberto Caldara, and Philippe G Schyns. Facial expressions of emotion are not culturally universal. *Proceedings of the National Academy of Sciences*, 109(19):7241–7244, 2012.

Deepak Kumar Jain, Pourya Shamsolmoali, and Paramjit Sehdev. Extended deep neural network for facial emotion recognition. *Pattern Recognition Letters*, 120:69–74, 2019.

Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM, 2019.

Haifeng Jin, François Chollet, Qingquan Song, and Xia Hu. Autokeras: An automl library for deep learning. *Journal of Machine Learning Research*, 24(6):1–6, 2023. URL `http://jmlr.org/papers/v24/20-1355.html`.

Cijo Jose, Prasoon Goyal, Parv Aggrwal, and Manik Varma. Local deep kernel learning for efficient non-linear svm prediction. In *International conference on machine learning*, pages 486–494, 2013.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Byoung Ko. A brief review of facial emotion recognition based on visual information. *sensors*, 18(2):401, 2018.

Dimitrios Kollias, Viktoriia Sharmanska, and Stefanos Zafeiriou. Face behavior à la carte: Expressions, affect and action units in a single network. *CoRR*, abs/1910.11111, 2019. URL `http://arxiv.org/abs/1910.11111`.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Lubor Ladicky and Philip Torr. Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 985–992, 2011.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Jing Li, Kan Jin, Dalin Zhou, Naoyuki Kubota, and Zhaojie Ju. Attention mechanism-based cnn for facial expression recognition. *Neurocomputing*, 411:340–350, 2020.

Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *arXiv preprint arXiv:1804.08348*, 2018.

Shan Li and Weihong Deng. Reliable crowdsourcing and deep locality-preserving learning for unconstrained facial expression recognition. *IEEE Transactions on Image Processing*, 28(1):356–370, 2019.

Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing*, 2020.

Shan Li, Weihong Deng, and JunPing Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2584–2593. IEEE, 2017.

Yingjian Li, Yao Lu, Jinxing Li, and Guangming Lu. Separate loss for basic and compound facial expression recognition in the wild. In *Asian Conference on Machine Learning*, pages 897–911. PMLR, 2019.

Liqian Liang, Congyan Lang, Yidong Li, Songhe Feng, and Jian Zhao. Fine-grained facial expression recognition in the wild. *IEEE Transactions on Information Forensics and Security*, 16:482–494, 2020.

Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops*, pages 94–101. IEEE, 2010.

M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205, 1998. doi: 10.1109/AFGR.1998.670949.

Michael Lyons, Miyuki Kamachi, and Jiro Gyoba. The Japanese Female Facial Expression (JAFFE) Database, April 1998. URL `https://doi.org/10.5281/zenodo.3451524`.

David Matsumoto, Theodora Consolacion, Hiroshi Yamada, Ryuta Suzuki, Brenda Franklin, Sunita Paul, Rebecca Ray, and Hideko Uchida. American-japanese cultural differences in judgements of emotional expressions of different intensities. *Cognition & Emotion*, 16(6):721–747, 2002.

Zibo Meng, Ping Liu, Jie Cai, Shizhong Han, and Yan Tong. Identity-aware convolutional neural network for facial expression recognition. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 558–565. IEEE, 2017.

Shervin Minaee and Amirali Abdolrashidi. Deep-emotion: Facial expression recognition using attentional convolutional network. *arXiv preprint arXiv:1902.01019*, 2019.

Tom Mitchell, Bruce Buchanan, Gerald DeJong, Thomas Dietterich, Paul Rosenbloom, and Alex Waibel. Machine learning. *Annual review of computer science*, 4(1):417–433, 1990.

Tom Michael Mitchell et al. *Machine learning*, volume 1. McGraw-hill New York, 2007.

Ali Mollahosseini, David Chan, and Mohammad H Mahoor. Going deeper in facial expression recognition using deep neural networks. In *2016 IEEE Winter conference on applications of computer vision (WACV)*, pages 1–10. IEEE, 2016.

Roman Neruda and Petra Kudová. Learning methods for radial basis function networks. *Future Generation Computer Systems*, 21(7):1131–1142, 2005.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

Mark JL Orr. Regularization in the selection of radial basis function centers. *Neural computation*, 7(3): 606–623, 1995.

Andrew Ortony, G Clore, and Allan Collins. The cognitive structure of emotions. cam (bridge university press. *Cambridge, England*, 1988.

Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. .º'Reilly Media, Inc.", 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rosalind W. Picard, Elias Vyzas, and Jennifer Healey. Toward machine emotional intelligence: Analysis of affective physiological state. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):1175–1191, 2001.

Diah Anggraeni Pitaloka, Ajeng Wulandari, T Basaruddin, and Dewi Yanti Liliana. Enhancing cnn with preprocessing stage in automatic emotion recognition. *Procedia computer science*, 116:523–529, 2017.

Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

Michael JD Powell. Radial basis functions for multivariable interpolation: a review. *Algorithms for the Approximation of Functions and Data.*, 1985.

Michael JD Powell. Radial basis functions for multivariable interpolation: a review. *Algorithms for approximation*, 1987.

Bayu Yudha Pratama and Riyanarto Sarno. Personality classification based on twitter text using naive bayes, knn and svm. In *2015 International Conference on Data and Software Engineering (ICoDSE)*, pages 170–174. IEEE, 2015.

Antti Puurula and Albert Bifet. Ensembles of sparse multinomial classifiers for scalable text classification. In *Proceedings of the 2012 ECML/PKDD Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification, Bristol*, 2012.

Srinivasan Raman, Rytis Maskeliūnas, and Robertas Damaševičius. Markerless dog pose recognition in the wild using resnet deep learning model. *Computers*, 11(1):2, 2021.

Rahul Ravi, SV Yadhukrishna, et al. A face expression recognition using cnn & lbp. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 684–689. IEEE, 2020.

David Sander, Didier Grandjean, Gilles Pourtois, Sophie Schwartz, Mohamed L Seghier, Klaus R Scherer, and Patrik Vuilleumier. Emotion and attention interactions in social cognition: brain regions involved in processing anger prosody. *Neuroimage*, 28(4):848–858, 2005.

R Santhoshkumar and M Kalaiselvi Geetha. Deep learning approach for emotion recognition from human body movements with feedforward deep convolution neural networks. *Procedia Computer Science*, 152: 158–165, 2019.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298682. URL https://doi.org/10.1109/CVPR.2015.7298682.

Nicola Segata, Edoardo Pasolli, Farid Melgani, and Enrico Blanzieri. Local svm approaches for fast and accurate classification of remote-sensing images. *International journal of remote sensing*, 33(19):6186–6201, 2012.

Alireza Sepas-Moghaddam, Ali Etemad, Paulo Lobato Correia, and Fernando Pereira. A deep framework for facial emotion recognition using light field images. In *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 1–7. IEEE, 2019.

Jie Shao and Qiyu Cheng. E-fcnn for tiny facial expression recognition. *Applied Intelligence*, 51(1):549–559, 2021.

Jie Shao and Yongsheng Qian. Three convolutional neural network models for facial expression recognition in the wild. *Neurocomputing*, 355:82–92, 2019.

Jiawei Shi and Songhao Zhu. Learning to amend facial expression representation via de-albino and affinity. *arXiv preprint arXiv:2103.10189*, 2021.

Yoshihiro Shima and Yuki Omori. Image augmentation for classifying facial expression images by using deep neural network pre-trained with object image database. In *Proceedings of the 3rd International Conference on Robotics, Control and Automation*, pages 140–146, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Chawin Sitawarin and David Wagner. On the robustness of deep k-nearest neighbors. *arXiv preprint arXiv:1903.08333*, 2019.

Myunghoon Suk and Balakrishnan Prabhakaran. Real-time mobile facial expression recognition system-a case study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 132–137, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL `http://arxiv.org/abs/1409.4842`.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

Paweł Tarnowski, Marcin Kołodziej, Andrzej Majkowski, and Remigiusz J Rak. Emotion recognition using facial expressions. *Procedia Computer Science*, 108:1175–1184, 2017.

Lakshmi Sarvani Videla and PM Ashok Kumar. Facial expression classification using vanilla convolution neural network. In *2020 7th International Conference on Smart Structures and Systems (ICSSS)*, pages 1–5. IEEE, 2020.

Petra Vidnerová and Roman Neruda. Deep networks with rbf layers to prevent adversarial examples. In *International Conference on Artificial Intelligence and Soft Computing*, pages 257–266. Springer, 2018.

Kai Wang, Xiaojiang Peng, Jianfei Yang, Debin Meng, and Yu Qiao. Region attention networks for pose and occlusion robust facial expression recognition. *IEEE Transactions on Image Processing*, 29:4057–4069, 2020.

Wei Wei, Qingxuan Jia, and Gang Chen. Real-time facial expression recognition for affective computing based on kinect. In *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, pages 161–165. IEEE, 2016.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.

Aron Yu and Kristen Grauman. Fine-grained comparisons with attributes. In *Visual Attributes*, pages 119–154. Springer, 2017.

Pingpeng Yuan, Yuqin Chen, Hai Jin, and Li Huang. Msvm-knn: Combining svm and k-nn for multi-class text classification. In *IEEE international workshop on Semantic Computing and Systems*, pages 133–140. IEEE, 2008.

Pourya Habib Zadeh, Reshad Hosseini, and Suvrit Sra. Deep-rbf networks revisited: Robust classification with rejection. *arXiv preprint arXiv:1812.03190*, 2018.

Cleber Zanchettin, Byron Leite Dantas Bezerra, and Washington W Azevedo. A knn-svm hybrid model for cursive handwriting recognition. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

Jiabei Zeng, Shiguang Shan, and Xilin Chen. Facial expression recognition with inconsistently annotated datasets. In *Proceedings of the European conference on computer vision (ECCV)*, pages 222–237, 2018.

Hao Zhang, Alexander C Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2126–2136. IEEE, 2006.

Hepeng Zhang, Bin Huang, and Guohui Tian. Facial expression recognition based on deep convolution long short-term memory networks of double-channel weighted mixture. *Pattern Recognition Letters*, 131:128–134, 2020.

Xiangyun Zhao, Xiaodan Liang, Luoqi Liu, Teng Li, Yugang Han, Nuno Vasconcelos, and Shuicheng Yan. Peak-piloted deep network for facial expression recognition. In *European conference on computer vision*, pages 425–442. Springer, 2016.